UNIVERSITÀ
DI TRENTO

UNIVERSITY OF TRENTO

PHYSICS DEPARTMENT

MASTER DEGREE IN PHYSICS

Joint Program between UniTn and SISSA

Enhancing the Resolution of CMB Foreground Models: a Stochastic and Iterative Approach

---

Supervisor:
Dr. Nicoletta Krachmalnicoff
Co-supervisor:
Prof. Albino Perego

Graduant:
Marianna Foschi

December 17, 2021

UNIVERSITY OF TRENTO
Physics Department

Master Degree in Physics

Enhancing the Resolution of CMB Foreground Models: a Stochastic and Iterative Approach

by Marianna Foschi

Abstract

The cosmic microwave background radiation (CMB) is currently one of the main sources of information about the early evolution of our universe, since the patterns in the CMB anisotropies carry the imprint of the primordial fluctuations, dating back to the epoch of inflation. One of the last predictions of the inflationary framework that still remains to be observed is the existence of primordial gravitational waves, which should have left a trace in the CMB polarization B-modes anisotropies. Therefore, detecting a B-modes signal, or putting a strict upper limit to its amplitude, is currently one of the main goals of future CMB experiments. The detection of primordial B-modes is made difficult because of contamination by gravitational lensing and foreground emitting sources. The major foreground contaminant is the galactic emission, in particular the thermal dust emission, which is predominant in the frequency range of CMB observations. Modeling correctly this foreground is crucial to calibrate component separation and delensing algorithms and obtain a clean CMB signal.

Here I present an extension to the Python package ForSE ("Foreground Scale extender") for modeling polarized emission from the galactic thermal dust foreground, in the context of CMB observations. The peculiarity of ForSE, compared to other models is that, by exploiting an algorithm based on a generative adversarial neural network, it is able to produce a full sky map of the polarized dust emission with non-Gaussian small scale features, up to a resolution of 12 arcminutes, which is higher than the resolution of best available experimental maps made from Planck data. My thesis aims at improving the ForSE package on two fronts: first by adding stochasticity in the foreground model, so to introduce variability in the foreground maps, and secondly by iterating the model to generate maps with a resolution of 3 arcminutes. As a result, I was able to obtain a procedure that generates different realizations of thermal dust polarization foreground maps down to scales of 12 and 3 arcminutes. The statistical properties of the maps were evaluated by computing angular power spectra, comparing Minkowski functionals and studying the distribution of pixel values. Minkowski functionals were also used as a tool to evaluate the training of the neural network. The foreground maps generated by this model, will be used in future work to tune algorithms that reconstruct the gravitational lensing potential from CMB maps, to perform delensing of the CMB signal.

Acknowledgements

I would like to thank my supervisor Dr. Nicoletta Krachmalnicoff for assigning me this very interesting project, for tolerating my delays and for her constant guidance and availability as well as patience in solving IT problems. A thanks also to Dr. Puglisi for his help in using computational resources and to my co-supervisor Prof. Perego for his interest in the project. On a more personal note, a great support while working on the thesis has come from my boyfriend Gabriele who constantly encouraged me, and, together with my flatmates Antonio and Vittorio, made our home the best place to find the balance between focus and relax.

The Acknowledgments section of a master's thesis is traditionally the place where the author can thank people who gave significant contributions not for the completion of the thesis itself but during the years of study leading up to it. In this regard, I would like to thank my father Damiano for the humbling discussions on mathematics and my mother Silvia for tolerating such conversations at the dinner table. I am grateful that he showed me along the years the fascination and fun of science and logic. My years at university were fruitful also thanks to the now physicists Mauro, Filippo, Francesco, Davide, Zeno and Emiliano, from the "Cetriolini", who made up the most brilliant, supporting, talented, fun, nerd and caring group of friends that I could have wished for. In particular I am thankful to Francesco for the time we spent studying and discussing physics together. A word also for Prof. Rinaldi, for his helpful and appreciated career advice and guidance.

This work used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

Contents

Introduction

One of the open questions in cosmology is whether the theory of cosmic inflation is correct and, if so, what are the details of the inflationary epoch. Among the predictions of inflation, there is the production of primordial gravitational waves, the trace of which should have remained imprinted in the patterns of the cosmic microwave background B-modes polarization signal. Detecting B-modes is difficult because of contamination from galactic foregrounds and distortion from gravitational lensing. In this context, the ability to model realistic maps of the synchrotron and thermal dust galactic emissions is crucial to build component separation algorithms that perform foreground cleaning and delensing of the primordial microwave background. One of the current models for polarized thermal dust emission, named ForSE, uses an algorithm based on convolutional neural networks, to create foreground maps with non-Gaussian small scales features at a resolution of 12 arcminutes, starting from Planck foreground polarization maps at 80 arcminutes.

The purpose of this thesis is to extend the scope of the ForSE package by adding two functionalities. The first consists in introducing stochasticity in the model that generates the foreground map, so that it is possible to create different foreground map realizations. Having a dataset of various statistically equivalent maps is useful to estimate the variance of the parameters computed from them. The second goal is to use the model iteratively, to create foreground maps at the even higher resolution of 3 arcminutes, which is close to the target resolution of the next future cosmic microwave background experiments.

The present text is articulated in two main parts. Part I provides an introduction to the topics of cosmology and computer science, that are necessary to understand the thesis, as well as a description of the previous work on the subject. Chapter 1 gives an overview on the cosmic microwave background and galactic foregrounds. It starts by explaining, in the context of the current cosmological model, what is the CMB, how it formed, what is its phenomenology and how it is detectable today. It goes on to describe the relevance of CMB polarization B-modes in relation to primordial fluctuations and the inflationary theory, stating the difficulties in detecting their signal and the sources of contamination. A quick description of the most important CMB experiments and up to date findings is presented in the second part of the chapter. Particular focus in given to the current knowledge about galactic foregrounds, including the available observations and models. Chapter 2 is dedicated to the topic of neural networks. It explains, from the basics, the concepts of deep learning, network training and the structural components of a neural network. Particular attention is paid to convolutional and generative adversarial networks and their employment in image recognition and image generation. The end of the chapter is dedicated to an overview of the most notable recent applications of convolutional neural networks in the fields of cosmological simulations and astronomical observations. Chapter 3 presents the "Foreground Scale Extender" (ForSE) Python package, developed by Dr. Krachmalnicoff and Dr. Puglisi, which provides a model

for thermal dust galactic emission based on neural networks. The characteristics of the model, the methodologies used for its development and the principal results are discussed.

Part II contains the core of the thesis' work. It gives motivation for the importance of the subject, describes the methodologies used and analyzes the obtained results. Chapter 4 addresses the implementation of stochasticisty in the ForSE model, in order to produce different realizations of foreground maps. The performance of the network training is discussed and the statistical properties of resulting maps are analyzed. Chapter 5 focuses on the iteration of the approach described in chapter 4, to produce foreground maps with higher resolution. The details of the iteration precedure are explained and the resulting maps are shown. Finally, chapter 6 shows how the results from the previous chapters can be used to produce full sky foreground maps and discusses the characteristics of such maps.

Part I

Chapter 1

Cosmic Microwave Background

The purpose of this chapter is to present the theoretical and observational framework within which the work of this thesis is located. The first part, will give a general overview on the topics of cosmological expansion, primordial fluctuations and inflation as well as a description of the phenomenology of the cosmic microwave background, including decoupling, polarization, harmonic decomposition and lensing (Baccigalupi, 2021, Krachmalnicoff and Poletti, 2021, Hu and White, 1997). In particular, the relevance of the detection of CMB polarization B-modes will be highlighted by explaining how the latter are linked to primordial gravitational waves and inflation (Kamionkowski and Kovetz, 2016). In the second part, the problem of foreground contaminants to the CMB will be presented, along with an overview of the current foreground data and models. The description of cosmological processes is given from a more qualitative point of view, taking for granted the reader's knowledge of the formalism, while the characterization of the quantities pertaining the CMB radiation is carried out in more detail.

1.1 An Expanding Universe, Photon Decoupling and CMB Formation

On the very large scales, the distribution of matter and energy in the universe appears to be homogeneous and isotropic and can be described by the perfect fluid stress-energy tensor (SET):

$$T_{\mu\nu} := (\rho + p)u_\mu u_\nu + p g_{\mu\nu} \,, \tag{1.1}$$

where ρ is the energy density, p the pressure and u_μ the 4-velocity of the fluid. The metric of a spacetime manifold with a homogeneous and isotropic distribution of energy and matter is the Friedmann-Robertson-Walker (FRW) metric:

$$\mathrm{d}s^2 = \mathrm{d}t^2 - a^2(t)\left[\frac{\mathrm{d}r^2}{1-kr^2} + r^2\left(\mathrm{d}\theta^2 + \sin^2\theta \mathrm{d}\phi^2\right)\right] \,, \tag{1.2}$$

which describes a universe that can be foliated into maximally symmetric spatial hypersurfaces with curvature $K = -1, 0$, or 1, which expands with the scale factor $a(t)$. From the Einstein's equations (Einstein, 1916):

$$G_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu} \tag{1.3}$$

the rate of expansion of the scale factor is regulated by the Friedmann equations:

$$H^2(t) := \frac{\dot{a}^2}{a^2} = \frac{8\pi G}{3}\rho - \frac{K}{a^2}\,, \tag{1.4}$$

$$\frac{\ddot{a}}{a} = -\frac{4\pi G}{3}(\rho + 3p)\,, \tag{1.5}$$

together with the equation of state for the perfect fluid:

$$p = \omega\rho\,. \tag{1.6}$$

Different energy sources are present in the universe and they may be distinguished into the categories of matter, i.e. collisionless, non-relativistic dust particles with $\omega = 0$, radiation, which consists of ultrarelativistic particles with $\omega = 1/3$, and dark energy, characterized by $\omega = -1$. From the previous three equations it results that the energy density scales differently with the scale factor for different sources: for radiation $\rho_R \propto a^{-4}$, for matter $\rho_M \propto a^{-3}$, while for dark energy $\rho_{DE} = \text{const}$. As the universe expands, the fraction of the total energy density relative to each component varies. In particular, for sufficiently small values of the scale factor, radiation was the dominant component. Then, with growing scale factor, it became matter and lastly dark energy. So we speak of a radiation dominated era (RDE), during which the scale factor expanded as $a(t) \propto t^{1/2}$, a matter dominated era (MDE), in which $a(t) \propto t^{2/3}$, and a dark energy dominated era, where the scale factor grows exponentially in time as $a(t) \propto e^{Ht}$ (Carrol, 2013).

At early times during the RDE, when the density of particles was high enough, all the different relativistic species were kept in thermal equilibrium by particle interactions. As the universe expanded, the energy density and the temperature associated with the plasma of particles decreased. When the energy scale was not sufficient to keep the rate of interactions higher than the rate of expansion, the particles involved in the interaction decoupled, meaning that they stopped interacting and began free streaming across the expanding universe. The first species to decouple were neutrinos, which during the RDE ceased to interact weakly with baryons and formed the cosmic neutrino background. Later on, during the MDE, also photons decoupled from baryonic matter, creating what is now the cosmic microwave background radiation (Peebles, 1993).

Before decoupling, photons were kept in thermal equilibrium with protons and electrons by electromagnetic (EM) interactions. But at redshift $z = 1100$, the energy scale was sufficiently low to allow for protons and electrons to form neutral atoms, under a process called recombination, and, as a consequence, photons started to stream freely. The radiation present at the time of recombination kept travelling in all directions, with energy decreasing because of the ongoing expansion of the universe, until now, when it is detectable in the microwave range at a temperature of $T = 2.725\,K$. The set of points where the CMB photons that are detected today on earth were originally emitted make up a spherical surface, called the last scattering surface (LSS). The decoupling was not instantaneous, it occurred over an interval $\Delta z = 100$, which determines the thickness of the LSS. By observing the CMB radiation, it is possible to gain information about the density distribution at the decoupling epoch, as well as about all the phenomena and objects that affected the CMB between the LSS and now, notably reionization and gravitational lensing.

The cosmic microwave background is partially linearly polarized, due to the scattering processes at the LSS. Indeed, since decoupling was not instantaneous, the photons could scatter off electrons multiple times, covering a distance equal to the

thickness of the LSS, before the last scatterer. Thomson scattering is the classical elastic scattering of a photon off a charged particle. Its differential cross section is proportional to the square of the product between the polarization vector of the incident wave and that of the scattered wave:

$$\frac{d\sigma_T}{d\Omega} \propto |\varepsilon \cdot \varepsilon'|^2 \tag{1.7}$$

Therefore, in the directions orthogonal to the polarization of the incident radiation, the scattered wave has higher intensity and is polarized parallel to the incident one. If the incoming radiation is isotropic, the Thomson scattered radiation does not have a net polarization. On the contrary, for anisotropic radiation, in particular in the presence of quadrupole intensity anisotropies, Thomson scattering acts as a polarization mechanism.

Anisotropies present at the time of the LSS left an imprint in the CMB that is visible still today, both in total intensity and polarization. The intensity of the CMB radiation in a particular sky direction depends on the energy density in the point where the correspondent photons left the LSS. Photons last scattered from points with over or under-densities, hence different gravitational potentials, are observed with slightly different redshifts. At the LSS, quadrupole anisotropies in the density generated quadrupole anisotropies in the radiation intensity, that resulted in the polarization of the CMB radiation.

Today, after removing the average value and the dipole anisotropy caused by the peculiar velocity of the earth with respect to the CMB, the fluctuations of the CMB's total intensity are of the order of $10^{-5}\,K$. The polarization fraction, i.e. the ratio between the amplitude of the polarized signal and the total intensity signal is 10%, so the amplitude of the polarized signal is of the order of $10^{-6}\,K$.

The causes of anisotropies are not to be attributed exclusively to the processes dating back to decoupling. For example, the photons' distribution may be altered along their path by gravitational weak lensing due to the newly formed large scale structures or by the plasma of particles ionized after the appearance of the first stars in the reionization period.

1.2 Polarization and Harmonic Expansion of the CMB Field

From an observational point of view, the CMB signal is an EM radiation coming to the Earth from every direction in the sky. So two quantities can be measured for every sky position: the total intensity of the EM field and the direction of its polarization. The intensity can be described by a scalar field ($\Theta(\hat{n})$), defined on a sphere spanned by the sky coordinates[1], while the polarization is represented by a spin-2 field, also defined on the sphere. If we call (A_x, A_y) the components of the amplitude of the EM field along axis $\hat{x}$ and $\hat{y}$ orthogonal to $\hat{n}$, the total intensity is defined as:

$$\Theta(\hat{n}) := \left(\langle A_x^2 \rangle + \langle A_y^2 \rangle\right)(\hat{n}) \tag{1.8}$$

[1]The system of sky coordinates used in this thesis is that of galactic longitude and latitude (l,b).

The two components of the polarization field are usually expressed in terms of the Stokes parameters for linear polarization, which are defined as:

$$Q(\hat{n}) := \left(\langle A_x^2\rangle - \langle A_y^2\rangle\right)(\hat{n}) = \Theta_x(\hat{n}) - \Theta_y(\hat{n}) \tag{1.9}$$

$$U(\hat{n}) := \left(\langle A_{x'}^2\rangle - \langle A_{y'}^2\rangle\right)(\hat{n}) = \Theta_{x'}(\hat{n}) - \Theta_{y'}(\hat{n}) \tag{1.10}$$

where (x, y) and (x', y') are two pairs of perpendicular axis rotated 45° apart[2]. Under a right handed rotation ψ around the axis $\hat{n}$, the Q and U fields mix with one another:

$$Q \longrightarrow Q\cos 2\psi + U\sin 2\psi \tag{1.11}$$

$$U \longrightarrow -Q\sin 2\psi + U\cos 2\psi \tag{1.12}$$

so it might be better to work with the two fields $(Q \pm iU)(\hat{n})$, which under rotations only acquire a phase factor:

$$(Q \pm iU)(\hat{n}) \longrightarrow e^{\mp 2i\psi}(Q \pm iU)(\hat{n}) \tag{1.13}$$

Like every function on the sphere, also the CMB fields can be decomposed into a basis of scalar and rank-2 tensor spherical harmonics[3]:

$$T(\hat{n}) = \sum_{\ell m} a_{\ell m} Y_{\ell m}(\hat{n}) \tag{1.14}$$

$$(Q + iU)(\hat{n}) = \sum_{\ell m} a_{\ell m}^{+2} Y_{\ell m}^{+2}(\hat{n}) \tag{1.15}$$

$$(Q - iU)(\hat{n}) = \sum_{\ell m} a_{\ell m}^{-2} Y_{\ell m}^{-2}(\hat{n}) \tag{1.16}$$

The coefficients $a_{\ell m}, a_{\ell m}^{\pm 2}$, with $\ell = 0, 1, \ldots$ and $m = -\ell, \ldots, +\ell$ are sufficient to fully characterize the total intensity and polarization signals. This decomposition is based on the Q and U observables that are directly linked to the polarization vector. However, a decomposition on a different basis might serve to highlight different characteristics of the two polarization fields. It is customary to define new coefficients:

$$E_{\ell m} = \frac{1}{2}\left(a_{\ell m}^{+2} + a_{\ell m}^{-2}\right) \tag{1.17}$$

$$B_{\ell m} = \frac{1}{2i}\left(a_{\ell m}^{+2} - a_{\ell m}^{-2}\right) \tag{1.18}$$

(Zaldarriaga and Seljak, 1997). The E and B coefficients are named in this way because they have a distinct behaviour under the parity transformation $\hat{n} \to -\hat{n}$: E coefficients acquire a sign $(-1)^{\ell}$ while B coefficients get a sign $(-1)^{\ell+1}$, just like the electric and magnetic fields. Stretching further the analogy with electromagnetism, E-modes are curl-less fields, while B-modes are divergence-less fields.

[2] In general, polarized radiation is described also by the Stokes parameter V for circular polarization, but the polarization of the CMB is only linear.

[3] Regarding the multipole decomposition, note that higher values of ℓ correspond to spherical harmonics with smaller scale features. The correspondence between the variability angular scale θ of a spherical harmonics, and the multipole ℓ is not precise, because a particular harmonic may have features that vary on small scales in one direction and on large scales in another. Anyway, a rough estimate of the correspondence may be given by the relation $\ell = \pi/\theta$, especially on small scales. This relation will be used in the following, whenever needed.

The angular power spectrum of the CMB signal is usually computed in the E-modes, B-modes decomposition:

$$C_\ell^{\Theta\Theta} = \frac{1}{2\ell+1}\sum_m |a_{\ell m}|^2 \qquad C_\ell^{\Theta E} = \frac{1}{2\ell+1}\sum_m (a_{\ell m}E^*_{\ell m}) \tag{1.19}$$

$$C_\ell^{EE} = \frac{1}{2\ell+1}\sum_m |E_{\ell m}|^2 \qquad C_\ell^{\Theta B} = \frac{1}{2\ell+1}\sum_m (a_{\ell m}B^*_{\ell m}) = 0 \tag{1.20}$$

$$C_\ell^{BB} = \frac{1}{2\ell+1}\sum_m |B_{\ell m}|^2 \qquad C_\ell^{EB} = \frac{1}{2\ell+1}\sum_m (E_{\ell m}B^*_{\ell m}) = 0 \tag{1.21}$$

The correlation between total intensity (T) and B-modes or E-modes and B-modes is 0 because the interactions that involve the CMB photons are parity invariant, so modes with different parity behaviour are not correlated. The relevant angular power spectra to compute are the pure TT, EE or BB modes and the TE correlation. The temperature and polarization primordial anisotropies of the CMB radiation are determined by the early universe fluctuations (see section 1.3), so they should be Gaussian as well, meaning that the harmonic coefficients have Gaussian distributions with variances:

$$\langle a_{lm}a^*_{l'm'}\rangle = C_l^{\Theta\Theta}\delta_{ll'}\delta_{mm'} \qquad \langle a_{lm}E_{l'm'}\rangle = C_l^{\Theta E}\delta_{ll'}\delta_{mm'} \tag{1.22}$$

$$\langle E_{lm}E^*_{l'm'}\rangle = C_l^{EE}\delta_{ll'}\delta_{mm'} \qquad \langle a_{lm}B_{l'm'}\rangle = 0 \tag{1.23}$$

$$\langle B_{lm}B^*_{l'm'}\rangle = C_l^{BB}\delta_{ll'}\delta_{mm'} \qquad \langle E_{lm}B^*_{l'm'}\rangle = 0 \tag{1.24}$$

However, if non-Gaussianities were present in the primordial fluctuations, they should also appear in the CMB anisotropies. The presence of non-Gaussianities in the CMB are also due to gravitational lensing and foreground contamination (see sections 1.3.1 and 1.5), and it is important to take them into account when measuring the primordial ones.

1.3 Primordial Fluctuations, Inflation and B-modes

It is widely assumed that, before the radiation, matter and dark energy dominated eras, there was another period of exponential expansion called inflation. The assumption of an inflationary epoch provides an elegant solution to a few problems in cosmology. The most relevant ones are the flatness problem, namely the fact that the universe is almost flat even though flatness is an unstable state, the horizon problem, i.e. the fact that regions of the universe too distant from each other to have ever been in causal contact appear homogeneous and thermalized, and the problem of missing observations of magnetic monopoles. The inflationary hypothesis also provides an explanation for the origin of the large scale structures in the universe, predicting that they formed from the gravitational collapse of primordial quantum fluctuations that were stretched by inflation to very large scales (Guzzetti et al., 2016).

Apart from the observations of flatness, super-horizon homogeneity and missing monopoles, that first motivated the inflationary paradigm, other predictions of the theory are that:

- The density fluctuations at the end of inflation are adiabatic, meaning that the ratio between the fluctuations and the average value of the density is the same for all species of particles.

- The primordial fluctuations are mostly an isotropic Gaussian random field, with a power spectrum of the form:

$$\left\langle \Delta(\bar{k})\Delta^*(\bar{k}')\right\rangle \propto \delta(\bar{k}-\bar{k}')\frac{P(k)}{k^3} \tag{1.25}$$

  for a fluctuation Δ. Depending on the inflationary model, the presence of a small amount of non-Gaussianities is also predicted.

- The spectrum of the fluctuations is almost scale invariant, i.e. it follows a power law:

$$P(k) \propto k^{n_s-1} \tag{1.26}$$

  where the spectral index n_s is very close to 1, making $P(k)$ almost constant.

- Inflation generated tensor perturbations of the spacetime metric, i.e. primordial gravitational waves. The amplitude of these fluctuations varies among different inflationary models and is quantified by the tensor-to-scalar ratio:

$$r := \frac{\Delta_t^2}{\Delta_s^2} \tag{1.27}$$

  between the amplitude of the tensor fluctuations and the amplitude of the scalar ones.

All these predictions have been proven consistent with observational data, except for primordial gravitational waves, which have not been detected yet. Measuring a tensor-to-scalar ratio different from zero would mean having the last piece of evidence in support of the inflationary theory.

The privileged source for information about the inflationary epoch is the CMB, because it retains the imprint of the fluctuations' distribution at the time of recombination. This is particularly true for what concerns the observations of primordial gravitational waves, whose traces can be found in the B-modes signal.

Primordial perturbations are the fluctuations in space of the components of the stress-energy tensor (i.e. density, pressure, shear and velocity of the primordial plasma) and of the metric tensor. The time evolution of these perturbations is determined by the linearized Einstein's equations[4], which relate the variations of the SET to the metric variations and their derivatives. The fluctuations of both tensors can be distinguished in scalar, vector and tensor perturbations, depending on their behaviour under rotations. Scalar perturbations are fluctuations of the energy density of the plasma. At the last scattering, quadrupole scalar anisotropies with $(\ell = 2, m = 0)$ cause a net irrotational flow of photons, which produces through Thomson scattering a curl-free polarization pattern in the radiation, i.e. CMB polarization E-modes. Vector perturbations are fluctuations in the plasma's vortical motions and, because of the Doppler effect, they would cause quadrupole moments with $(\ell = 2, m = \pm 1)$ in the temperature, that would generate a polarization pattern in the CMB with both a curl-less and a divergence-less component, hence both

[4]Non-linear structures, such as galaxies or clusters of galaxies, formed only relatively late in the universe timescale, so the linearized equations are sufficient to describe the evolution of perturbations from primordial fluctuations to decoupling.

E and B-modes. However vector perturbations are washed out during inflation, because they evolve only into decaying modes on super-horizon scales[5]. So no vector perturbation is present at the time of decoupling and their trace is not detectable in the CMB polarization modes. Tensor perturbations are the deviations of the spatial components of the metric tensor from the FRW metric, namely gravitational waves (GW). These perturbations are transverse and traceless, so they stretch the distance between test particles in directions orthogonal to the propagation of the wave, conserving the area of the plane orthogonal to the wavevector. Primordial GWs produce quadrupole tensor anisotropies with $(\ell = 2, m = \pm 2)$ in the temperature at the LSS. The resulting polarization affects both E and B-modes in the CMB.

Summing up, CMB polarization E-modes can be produced by scalar and tensor perturbations, while CMB polarization B-modes are produced only by tensor perturbations. This is why the detection of a B-mode signal in the CMB polarization is an evidence of primordial gravitational waves and therefore a confirmation of the inflation theory. From the detection of primordial B-modes it is possible to compute the tensor-to-scalar ratio, which is linked to the energy scale of inflation, enabling us to distinguish among different inflationary models (Baumann, 2012, Kamionkowski and Kovetz, 2016).

1.3.1 B-modes Power Spectrum and Lensing Contamination

The experimental detection of B-modes is not an easy task. The difficulty lies in the fact that the primordial signal, if present, is covered by B-modes produced by gravitational lensing and by contamination from the galactic foreground. Overlooking contaminants to the CMB polarization anisotropies, the B-modes power spectrum should present two distinct wide peaks, one at large angular scales ($\ell \lesssim 20$) due to re-scattering at reionization and the other at the degree scale ($\ell \sim 80$, $\theta \sim 2^\circ$), dating back to recombination. In addition, there is a strong contribution to the signal due to B-modes induced by gravitational lensing.

The fact that scalar perturbations cannot excite polarization B-modes, is true only in the linear regime of cosmological perturbation, that lasts until the formation of large scale structures by gravitational collapse. When density perturbations are not linear anymore, the photons' paths are bent by the gravitational potential of large scale objects present nearby, in a way that transfers power from the E-modes to B-modes, under what is called E/B mode mixing due to weak lensing. Even in the absence of primordial B-modes, after the effect of weak lensing, a signal in B-modes appears. On average, the photons emitted at the LSS undergo a total deflection of a few arcminutes due to the repeated deviations caused by incoherent gravitational lenses. Therefor, weak lensing affects the B-modes power spectrum with a peak at small angular scales ($\ell \sim 1000$). The impact of lensing is not negligible, indeed, the amplitude of the lensing signal at the angular scales of the recombination bump is equivalent to the amplitude of primordial B-modes with $r \simeq 0.02$ (Hanson et al., 2013, Louis et al., 2017). Therefore, it is important to tune correctly the algorithms that reconstruct the lensing potential and then perform delensing on the CMB total intensity and polarization maps. The features of the B-modes power spectrum are shown in fig. 1.1, where one can see the reionization and recombination bumps, and the lensing contribution. The primordial B-modes' spectrum depends linearly on

[5]The evolution of primordial perturbations has different behaviours depending on the type of perturbation (scalar, vector, tensor) and whether the perturbation scale is larger or smaller than the Hubble radius.

the tensor-to-scalar ratio and in the figure it is plotted for two different values r. The E-modes and temperature spectra are also plotted for scale comparison.

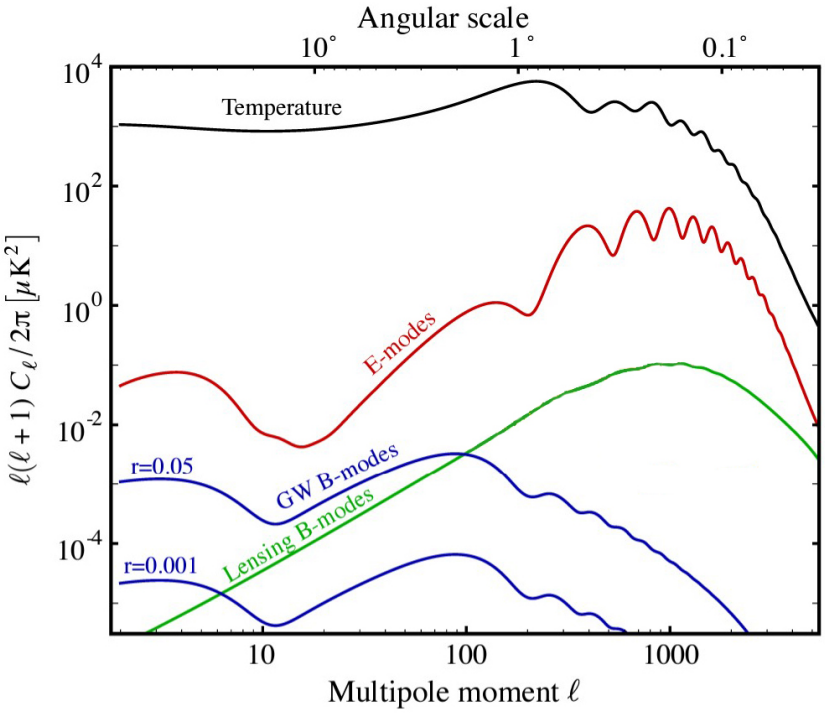


FIGURE 1.1: Expected angular power spectra of the CMB temperature, E-modes and B-modes signals. For B-modes possible primordial power spectra are plotted for two different values of r and the contribution of lensing B-modes is also shown. Adapted from Bersanelli et al., 2012.

Another source of non-primordial B-modes comes from the foreground emissions that mix with the CMB radiation. They will be discussed more extensively in section 1.5.

1.4 Overview of Past and Future CMB Experiments

Since the discovery in 1964 of the existence of a cosmic microwave background radiation (Penzias and Wilson, 1965, Dicke et al., 1965), many different experiments were performed to measure the CMB signal and its black body spectrum, as well as build maps of the temperature and polarization anisotropies and the corresponding power spectra. The ultimate main goal of CMB investigations is to determine as precisely as possible the cosmological parameters of the ΛCDM model, which is the current model of our universe.

CMB observations may be carried out by ground-based telescopes, balloon-borne probes or space satellites. The most relevant advances in our understanding of the CMB come from the full sky data recorded by three satellites launched by NASA and ESA: COBE, WMAP and Planck. The first one was launched in 1989. It measured the almost perfect black body spectrum of the CMB radiation (Bennett et al., 1993) and detected the small anisotropies. WMAP's mission started in 2001 and produced full sky maps of the CMB anisotropies in total intensity and polarization at five different frequencies, and angular power spectra of the anisotropies with clearly defined peaks (Bennett et al., 2013). The resolution of the sky maps was further improved

by the results from Planck satellite (Planck Collaboration I, 2020), active between 2009 and 2013. Most of today's knowledge on CMB comes from Planck's observations, which produced full sky maps in total intensity (for nine frequency bands) and in polarization (for seven frequency bands) with resolutions down to the degree scale. Multi-frequency observations, in the range between 30 and 857 Hz, were

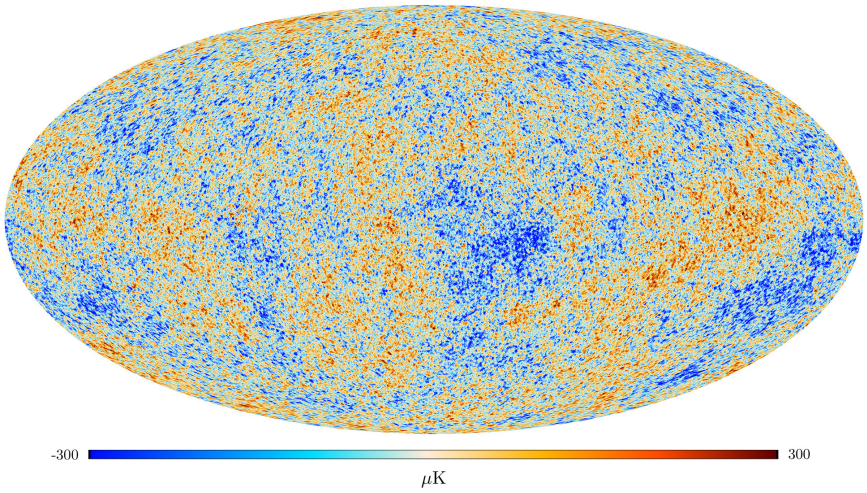


FIGURE 1.2: Temperature fluctuations of the CMB after removal of monopole, dipole and foregrounds. Adapted from ESA and the Planck Collaboration https://www.cosmos.esa.int/web/planck/picture-gallery.

used to perform component separation to isolate the different foreground sources and remove them from the total signal. Fig. 1.2 shows the famous map of CMB temperature anisotropies obtained with data from Planck, after the removal of foregrounds. From the maps, the total intensity and E-modes power spectra as well as the TE cross correlation were determined up to multipoles $\ell = 2500$ for the first one and $\ell = 2000$ for the second two. The power spectra of the lensing potential was also computed up to $\ell = 1000$. Planck data allowed to determine almost all the cosmological parameters of the ΛCDM model down to a 1% error or lower. Regarding the tensor-to-scalar ratio, using the total intensity and polarization power spectra from Planck, together with data from the BICEP2/Keck Array experiment, the constraints on r are set to $r < 0.044$ (Tristram et al., 2021).

With the completion of the Planck mission and the last data release in 2018 (Planck Legacy), the power spectra of temperature and E-modes have been determined up all physically relevant scales (Planck Collaboration V, 2020). Fig. 1.3 shows the combined results for the angular power spectra from all the recent experiments. It is clear that the peaks and oscillations of the temperature and E-modes spectra are fully characterized, while the B-modes spectrum is poorly defined and is dominated by the lensing signal. So the CMB community has shifted the focus on the goal (among few others) of measuring as precisely as possible the B-modes power spectrum at all scales from the full sky up to the arcminutes, in the quest for primordial gravitational waves. Detecting a tensor-to-scalar ratio different from zero, would mean to learn many information about the physics of the very early universe, such as the energy scale and expansion rate of inflation. Otherwise, setting a stricter upper limit on the value of r would rule out many of the simple theories of inflation.

Currently, various experiments are under preparation. Some will involve observations from earth-based telescopes, while other will use satellite probes. The

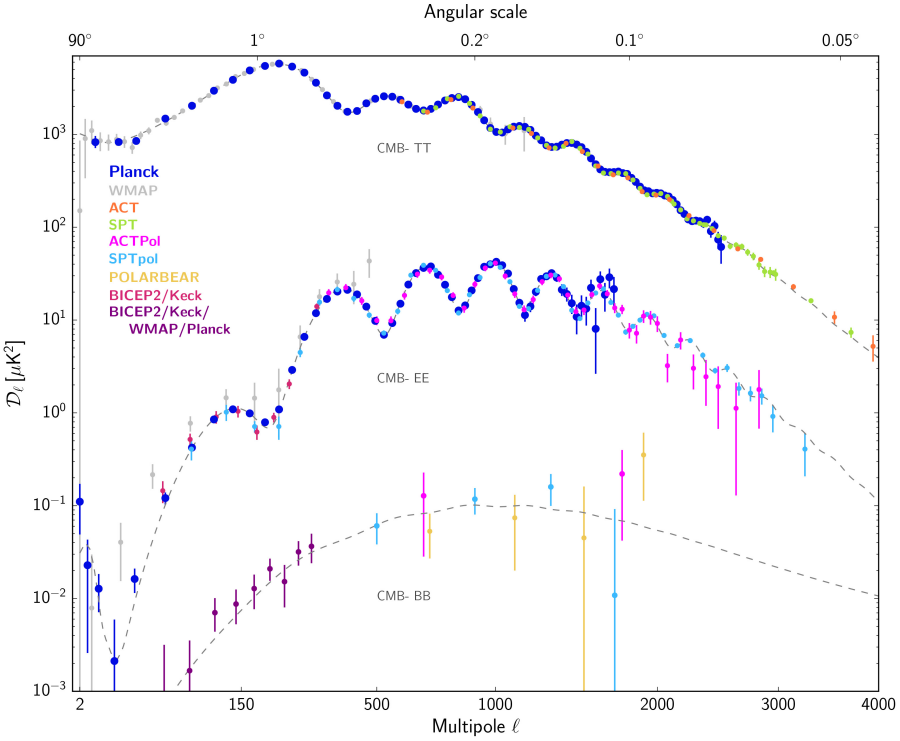


FIGURE 1.3: Compilation of CMB angular power spectrum measurements from recent experiments. The dashed line shows the best-fit ΛCDM model to the Planck temperature and E-mode and B-mode polarization power spectra. Planck Collaboration I, 2020.

Cosmic Microwave Background Stage-4 (CMB-S4) experiment will use dedicated telescopes at the South Pole and in the Atacama plateau to scan a 3% portion of the sky, with expected start date in 2029. Their goal is to reach an uncertainty of $\sigma(r) \sim 0.001$ on the tensor-to-scalar ratio, by measuring the recombination bump at the degree scales (Abazajian et al., 2016). Another earth-based experiment, at the Simons Observatory in the Atacama desert, will soon (2022) begin observations of the CMB anisotropies using small-aperture and large-aperture telescopes. The latter will scan 40% of the sky at the arcminute resolution (multipoles up to $\ell \sim 1000$), to reconstruct the gravitational lensing potential and remove the distortion effect from the CMB maps at degree scales ($\ell \in [30, 500]$), which are provided by the small-aperture telescopes on 10% of the sky. The experiment aims at building polarization maps with one order of magnitude less noise than the Planck ones, so to measure the tensor-to-scalar ratio up to $\sigma(r) \sim 0.003$ (The Simons Observatory Collaboration, 2019). Also the LiteBIRD satellite, planned to launch in 2027, will measure the B-modes power spectra. It will observe the CMB on the full sky and target both the reionization and the recombination bumps, up to multipoles $\ell \sim 200$, corresponding to sub-degree angular scales (Sugai et al., 2020).

1.5 Foreground Contamination to the CMB

In addition to weak lensing, another source of non-primordial B-modes comes from the foreground emissions that cover the CMB. Examples of contaminant sources are the atmospheric and human-made EM radiations, the infrared cosmic background,

active galactic nuclei emission, intracluster gas emission and, most importantly, the emission from our galaxy.

The emission is highest in directions pointing towards the galactic plane, but it is not negligible at any latitude. It is present at all frequencies (see fig. 1.4) and affects the CMB signal down to the degree scales, while at lower scales it affects the signal of the lensing potential, which has to be estimated carefully to perform delensing (Beck, Errard, and Stompor, 2020). The foreground emissions have different frequency dependencies and are relevant in different frequency ranges, so it is possible to separate the various foreground components from multi-frequencies measurements of the incoming radiation and component separation algorithms. The importance of estimating correctly the galactic foreground became evident in 2014 when the BICEP2 collaboration had to backtrack on the claim of a primordial B-modes detection, after realizing that the galactic emission was not modeled correctly (BICEP2 Collaboration, 2014, BICEP2/Keck and Planck Collaborations, 2015).

At the moment, from WMAP and Planck data, the galactic foreground radiation is resolved up to the degree scale on the full sky, while at lower scales, the amount of noise in the galactic emission polarization maps does not allow to extract any information outside small portions of the sky (Planck Collaboration X, 2016). The following subsections will give a description of the knowledge and observations of the galactic foregrounds as well as the foreground models that are currently employed for component separation algorithms.

1.5.1 Current Galactic Foreground Data

Galactic emission arises from two different sources: synchrotron emission and thermal dust emission. The first one originates from relativistic electrons accelerated along the galactic magnetic fields, it is dominant at low frequencies ($< 100\,GHz$) and is highly polarized (polarization fraction up to 20%). Full sky polarization maps of this foreground component, provided by WMAP and Planck, reach a resolution of a few degrees (Planck Collaboration XXV, 2016), while maps with less noise are available only on limited portions of the sky.

The second source is thermal dust emission, emitted by asymmetric dust grains in the interstellar medium, that are aligned along the lines of the galactic magnetic field. This emission is highly polarized too, with E-mode twice as powerful as B-modes (Planck Collaboration XXXVIII, 2016), and is dominant at frequencies $> 100\,GHz$. Today the best dust foreground maps are those made from Planck observations at 353 GHz (Planck Collaboration XLVIII, 2016, Planck Collaboration XXX, 2016). The dust foreground polarization maps in Q and U have a formal resolution of 5 arcminutes, but actually their features are resolved up to the scale of ten arcminutes only at sky latitudes close to the galactic equator, where the signal is stronger, while at higher latitudes the signal-to-noise ratio (SNR, i.e. the ratio between the power of the signal and the power of the noise) decreases, making features distinguishable only up to the degree scale. Fig. 1.5 presents the foreground full sky maps for synchrotron (in green) and dust (in red), obtained from Planck data using a component separation algorithm named Commander.

Since both emissions depend of the shape of the galactic magnetic field, synchrotron and thermal dust emissions show correlation up to the degree scale (Planck Collaboration XI, 2020). Their angular power spectra follow a power law:

$$C_{s/d}(\ell) \propto \ell^{\alpha_{s/d}} \tag{1.28}$$

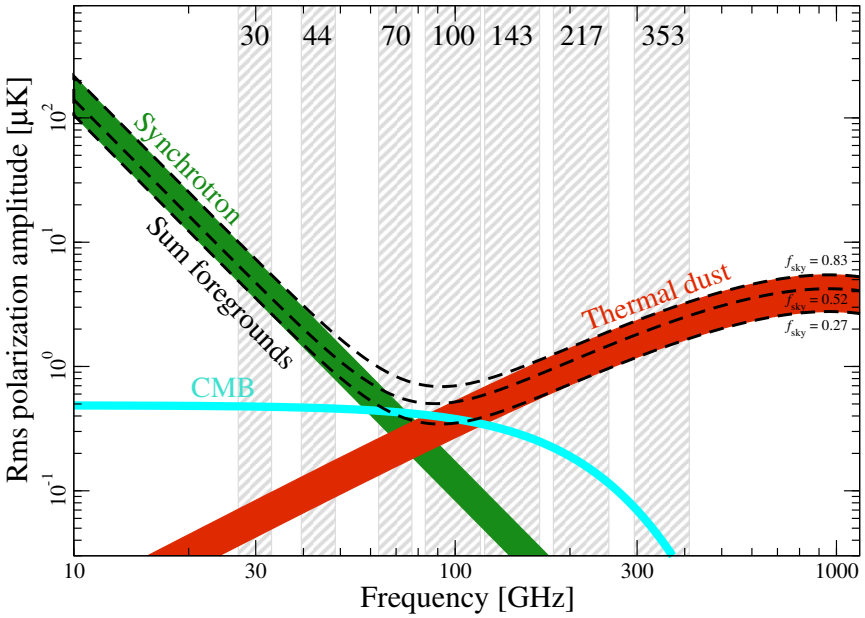


FIGURE 1.4: Polarization amplitude root mean square (rms) as a function of frequency and astrophysical components, evaluated at a smoothing scale of 40′ FWHM. The gray vertical bands indicate the frequency ranges of Planck measurements. The green band indicates polarized synchrotron emission, and the red band indicates polarized thermal dust emission. The cyan curve shows the CMB rms for a ΛCDM model with $\tau = 0.05$, and is strongly dominated by E-mode polarization. The dashed black lines indicate the sum of foregrounds evaluated over three different masks with fraction of sky coverage 0.83, 0.52, and 0.27. The widths of the synchrotron and thermal dust bands are defined by the largest and smallest sky coverages. Planck Collaboration IV, 2020.

with $\alpha_s = -3$ for synchrotron (Krachmalnicoff, N. et al., 2018, Jew and Grumitt, 2020) and $\alpha_d = -2.4$ for dust, which can be expected to hold up to scales lower than those currently resolved. The spectral energy distribution, i.e. the intensity of the radiation as function of the emitted frequency, instead, is different for the two components and makes their separation possible: it follows a power law for synchrotron emission and it is that of black body for dust (fig. 1.4).

The frequency dependence of the polarization amplitude of synchrotron and dust emissions are shown in fig. 1.4, together with the CMB E-modes polarization amplitude, against the frequency bands of the Planck measurements. It is clear synchrotron emission dominates at low frequencies, while dust emission dominates at high frequencies.

1.5.2 Galactic Thermal Dust Foreground Models

Modeling thermal dust foregrounds is not trivial, because of the complex features in the emission maps, together with the fact that the emission presents significant non-Gaussianities (i.e. non-null n-point correlation functions) and time variability (Coulton and Spergel, 2019)). Taking into account non-Gaussianities in the foreground models is important. Indeed, since weak lensing also induces non-Gaussianities in the radiation pattern, not accounting for them in the foreground model could lead to a biased reconstruction of the lensing potential and hence an incorrect estimation of the B-modes spectrum (Beck, Errard, and Stompor, 2020). Also, not considering non-Gaussianities in the foreground might affect the possible detection of primordial non-Gaussianities in the CMB polarization signal.

In preparation for the new CMB B-modes experiments described in section 1.4, foreground models are necessary in order to test the data analysis pipeline, i.e. the component separation, foreground subtracting, lensing reconstruction and delensing algorithms. The foreground models should have a resolution equal at least to the target resolution of the future observations, ideally at the arcminutes scales, which is difficult to achieve, since the available foreground maps reach resolutions at the degree scale.

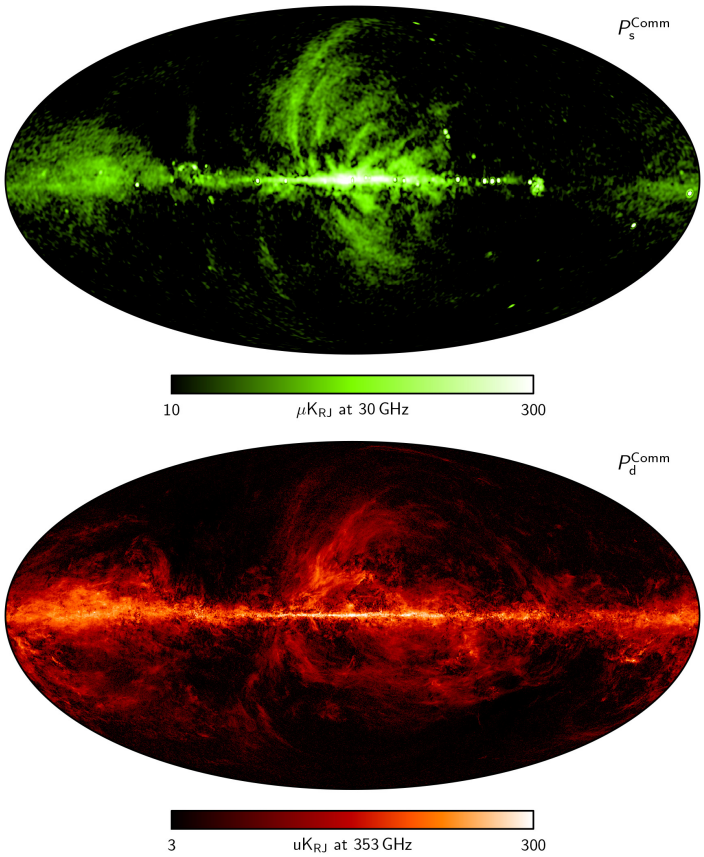


FIGURE 1.5: Commander 2018 polarized synchrotron (upper figure) and thermal dust (lower figure) amplitude maps at respectively 40′ and 5′ FWHM resolution, evaluated at a mono-chromatic reference frequency of respectively 30 and 353 GHz. Planck Collaboration IV, 2020.

Currently there are mostly two approaches for creating models of the galactic emission. The most common models of polarization dust emission are based on foreground maps and power spectra from previous experiments (WMAP and Planck). These models (such as the Python Sky Model, Thorne et al., 2017 and the Planck Sky Model, Delabrouille, J. et al., 2013) are reliable up to the scales at which the experimental foreground maps are signal dominated, which means up to the degree scale. Below that limit, smaller scale features can be added to the model in the form of a Gaussian field. This is done by extrapolating the power law spectra of the emission maps down to the desired scale and adding to the maps a Gaussian random field realization with a power spectra correspondent to the extrapolated one. This method allows to have models that have small scale features with the correct amplitude, but oversimplified statistics. Another approach consists in building the foreground emission from direct modeling of the interstellar medium through magnetohydrodynamics simulations, where turbulence is introduce by supernovae

explosions. This method, proposed by Kim, Choi, and Flauger, 2019, is able to generate galactic foregrounds with realistic properties, such as the E-B-modes asymmetry. However it has the downside of not producing a realization of the foreground with the large scale morphology of the real one. A combination between the two methods was carried out by Vansyngel, F. et al., 2017, who modeled the galactic magnetic field with a Gaussian turbulent component and a power law power spectrum, using constraints on the simulation parameters so that the polarization power spectra of the emission reproduced the spectra of real one. This model can produce foreground maps with resolution up to 0.5°.

Recently, Krachmalnicoff and Puglisi, 2021 proposed a new method for thermal dust foreground modeling in total intensity and polarization, based on the use of deep convolutional generative adversarial neural networks. The model is able to produce foreground maps with real large scale feature and small scale non-Gaussian features up to 12 arcminutes of resolution. The work of this thesis (presented in part II) consist in improving this model (which is described in detail in chapter 3) by adding variability and reaching the higher resolution of 3 arcminutes.

Chapter 2

Neural Networks

Putting aside for a moment the subject of CMB and galactic foregrounds, this chapter will give an overview of the concepts of deep learning and neural networks that are useful to describe the algorithms used in the thesis (Krachmalnicoff, 2020). It is not a comprehensive review of the topic, rather it presents the essential definitions and explanations required to understand the Deep Convolutional Generative Adversarial Networks (DCGANs), that were implemented in the code and data analysis scripts.

2.1 Introduction to Neural Networks

Traditional programming mostly consist in developing algorithms which encode a specific function in order to perform a task, taking some data as input and returning some data as output. Differently, the field of machine learning (ML) aims at building algorithms that learn by themselves which is the best function to perform the required task, without being explicitly programmed. A particular subfield of ML is Deep Learning (DL), which uses a variegated set of ML algorithms, characterized by a complex logical structure with multiple layers of nodes and connections, that deal with huge amounts of input data. Such algorithms are called Artificial Neural Networks (ANN), or simply Neural Network (NN), because they were initially inspired by and have a similar structure to the biological networks of neurons in the human brain.

Before a NN algorithm is able to perform the task it is designed for, a preparatory phase called training is required, so that the NN "learns" how to produce the correct output. There are two main ways in which this can be done: supervised and unsupervised learning. Supervised learning is based on the use of labeled datasets of input/output pairs. It is usually applied to situations where the ML algorithms has to learn the underlying relation between elements from two datasets. Unsupervised learning, instead, is performed without labeled example data and it is used to identify unknown patters in a dataset. The most common unsupervised learning methods are clustering, association and dimensionality reduction, while supervised learning includes two main methods: classification, when the output set contains a finite number of elements, and regressions, when the output set consist of one or more continuous parameters. In the following, only supervised learning will be discussed, because this is the method relevant to the network used in the thesis. For a NN, supervised learning consists in using the input/output pairs as reference examples to optimize the network parameters so that the NN learns to reproduce the correct output for any given input. The training procedure will be explained in detail in the following sections.

2.1.1 Feed-Forward Neural Network

In data analysis, there exist particular classes of tasks that can be done very easily by a human being but are hard to encode in an algorithm. Usually they involve the extraction of semantic information from spatially and temporally distributed data. Example of these tasks can be understanding the main topic of a written text, recognizing words from an audio signal or identifying the subject of a picture. These tasks fall under the computer science fields of semantic, speech and image recognition. These are all situations where neural networks can be way more effective than traditional algorithms and solve the task much faster than a human being.

For a neural network, the problem of data recognition can be defined as follows. One wants to implement a function f that maps an input x to the correspondent output $y = f(x)$, but the explicit form of f is not known. The goal then becomes to find the best approximation to the function f. This task can be accomplished by a feed-forward NN, which is a NN that has no recursive loops in its structure, but just receives an input, processes it across a sequence of logical layers and than returns an output. To perform the function approximation, the NN defines a function $f^*(x, \theta)$ that has to be optimized in the θ parameter space until it is close enough to the desired function f. NNs are able to simulate extremely different and complex functions because they apply iteratively a linear combination of the input data and a non-linear activation function. From the non-linearity, the complexity of the possible functions arises.

2.1.2 The Perceptron and Fully Connected Feed-Forward NN

The basic blocks that make up a neural network are nodes characterized by a set of parameters and an activation function, together with the ability to receive input and send output in the form of arrays or matrices.

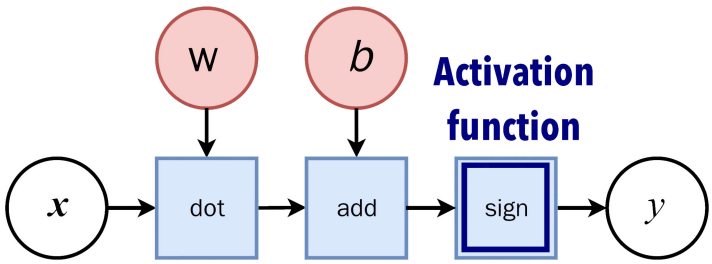


FIGURE 2.1: Diagram of the Perceptron model. White circles contain input/ouput data, red ones contain the parameters and blue boxes the functions applied. Krachmalnicoff, 2020.

The first and one of the simplest nodes to be defined is the Perceptron (Rosenblatt, 1958, illustrative diagram in fig. 2.1), which works according to the following sequence of actions:

1. it receives an input x in the form of an array of values,
2. it makes a linear combination of the input values using a parameter b and a vector of parameters w

$$w \cdot x + b\,,$$

3. it computes the output y by applying an activation function, in this case the step function Θ, to the linear combination

$$y = \Theta(w \cdot x + b)\,.$$

Many perceptrons can be assembled in parallel to build a layer of the neural network and the layers can be assembled in sequence to create a fully connected multi-layer feed-forward neural network, where the output of each layer is used as a input to the following layer (fig. 2.2). Fully connected means that every node in a layer is connected by a non-zero weight to every node in the previous and the following layer. In this way, the neural network is composed by an input layer

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_{\text{input}}} \end{bmatrix}, \tag{2.1}$$

consisting of an array of dimension n_{input}, a sequence of hidden layers, each with a parameter vector b_i of dimension n_i and a parameter matrix w_i of dimension $n_{i-1} \times n_i$:

$$b_i = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n_i} \end{bmatrix} \qquad w_i = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n_i} \\ w_{21} & w_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ w_{n_{i-1}1} & & & w_{n_{i-1}n_i} \end{bmatrix}, \tag{2.2}$$

and an output layer y, often in the form of a scalar value. All the coefficients in the b vectors and w matrices are called the parameters, or weights of the neural network, while the number and dimension of the hidden layers define the architecture of the NN and are two of the NN hyperparameters, i.e. the parameters that determine the structure of the NN and the features of the learning process. Neural networks can have up to thousands of layers and tens of millions of parameters. The higher the number of layers, the deeper the NN is said to be.

2.1.3 Learning Process

Before a NN is ready to perform its task, it must undergo three preparatory steps through which it will "learn" how to produce the correct output, using the dataset of labeled input/output pairs. These are:

1. training
2. validation
3. testing

and, accordingly, the dataset is divided into training dataset, containing the greater fraction of the data (around 80%), validation dataset and testing dataset, usually of the same size (around 10% each). During training, the NN parameters are optimized so that the NN output is as close as possible to the output of the labeled data. Validation is the step in which the NN hyperparameters are tuned by comparing the performance of different setups. Testing measures the performance of the optimized

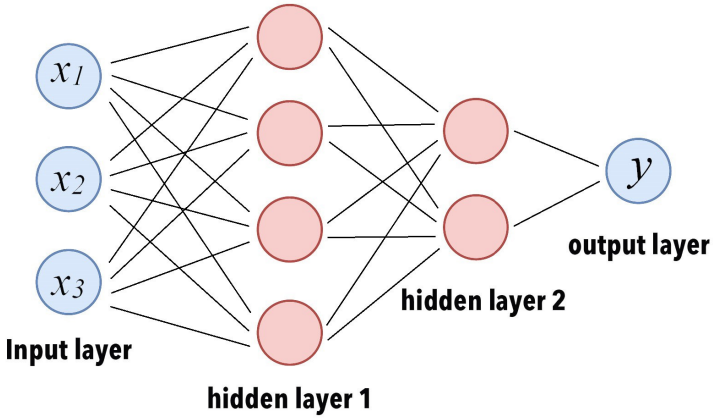


FIGURE 2.2: Schematic representation of a feed-forward NN. Blue nodes represent input/ouput data, while red circles represent the output data from each layer. Each black line before layer i represents the weights $w_{n,m}^{(i)}$, $b_n^{(i)}$ that are applied to the layer's input before activation. A network of this kind is called fully-connected because each node is connected to all nodes in the previous and the following layers. Note that this is a very simple architecture, in real applications networks can have up to $\sim 10^3$ layers and $\sim 10^7$ parameters. Adapted from Krachmalnicoff, 2020.

network. The following paragraph will present these steps more in detail, focusing on possible issues that may arise.

Training The processing of the input across hidden layers is the way by which the NN applies the function $f^*(x, \theta)$. Let's see how one can measure how far $f^*(\theta)$ is from f and how it can be optimized. Considering a pair of labelled input/output data $(x^j, \hat{y}^j)$, where $\hat{y}^j$ is called the ground truth, and the corresponding NN output y^j obtained from input x^j, one can define the loss function:

$$\mathcal{L}(y^j, \hat{y}^j) \tag{2.3}$$

that measures the distance between the NN output and the ground truth. The loss function concerns a single pair of labeled data, but, starting from it, the cost function can be defined as:

$$\mathcal{J} = \frac{1}{N} \sum_{j=0}^{N} \mathcal{L}(y^j, \hat{y}^j) \tag{2.4}$$

that is the average of the loss function over the whole training set. The cost function depends on the NN output, which in turn depends on the NN weights b_i and w_i. The training step consists in finding the NN weights that minimize the cost function. This is a simple optimization problem, made difficult however by the fact that the dimension of the parameter space can be extremely large (up to $\sim 10^7$ parameters).

Optimization algorithms The problem is solved by iterative approaches such as the gradient descent, or one of its many variations. Applied to this situation, the basic gradient descent algorithm consists of the following steps:

1. Initialize weights bs and ws.
2. Apply the NN to each input x^j and obtain output y^j.

3. Compute the cost function $\mathcal{J}$ on the outputs.
4. Check if the cost function is smaller than a fixed tolerance. If it is, stop training.
5. If not, update the weights to:

$$w \to w - \alpha \frac{\partial \mathcal{J}}{\partial w} \tag{2.5}$$

$$b \to b - \alpha \frac{\partial \mathcal{J}}{\partial b}\,. \tag{2.6}$$

   where α is the learning rate of the optimization algorithm.
6. Repeat from point 2.

The learning rate is a hyperparameter that controls how fast the model changes at each optimization step. The difficult part of the algorithm above is the computation of the derivatives in point 5. Indeed, in order to compute the derivative with respect to each weight, one has to apply the chain rule starting from the weights of the last layers back to those in the firsts. This process takes the name of backpropagation.

Each iteration of the gradient descent takes a lot of time, because computing the full gradient with respect to all parameters is computationally expensive. To reduce the time required for an iteration, it is possible to compute the gradient only with respect to a subset of variables, that are chosen randomly at every iteration. This method is called stochastic gradient descent (SGD) algorithm and it enables faster iterations at the expense of the rate of convergence. For the optimization of cost functions, the stochastic gradient descent, or modifications of it, are the preferred algorithms. To train the neural network used in this thesis work, a popular variation of the SGD, named Adam, was chosen (Kingma and Ba, 2017). This particular algorithm, assigns a different learning rate α_i to each weight, instead of a global one for all weights, and updates each of the α_i's depending on the average and variance of the last gradients with respect to weight i. Such a set up is appropriate for optimization problems with a large set of parameters and noisy gradients.

Validation, Testing and Overfitting. The validation step is meant to find the best hyperparameters for the NN architecture, by testing the network on a different dataset than the one used for training. Therefore, during validation, the weights are kept fixed, while the hyperparameters are adjusted by hand, to see which set up yields the lowest cost function over the validation dataset. The tunable hyperparameters include: the number of layers and their dimension in terms of nodes, the connections among nodes, the learning rate, the weight of the regularization term, the dropout rate and the number of elements in each minibatch (see definitions in the following paragraph). In the NN, the activation function, which is usually a variation of the step function, can be modified as well. A desirable characteristic of the activation function is that it yields a non-zero derivative when backpropagating (fig. 2.3).

The testing step takes place after training and validation. During testing nothing is updated in the NN, but the performance of the network is evaluated over a new dataset, to give a measure of how well the network can execute the classification or regression.

A possible problem when training a NN is overfitting the parameters, in the sense that they are optimized for the data in the training set but they are not adequate for a new generic dataset. Also the validation step can lead to overfitting the network

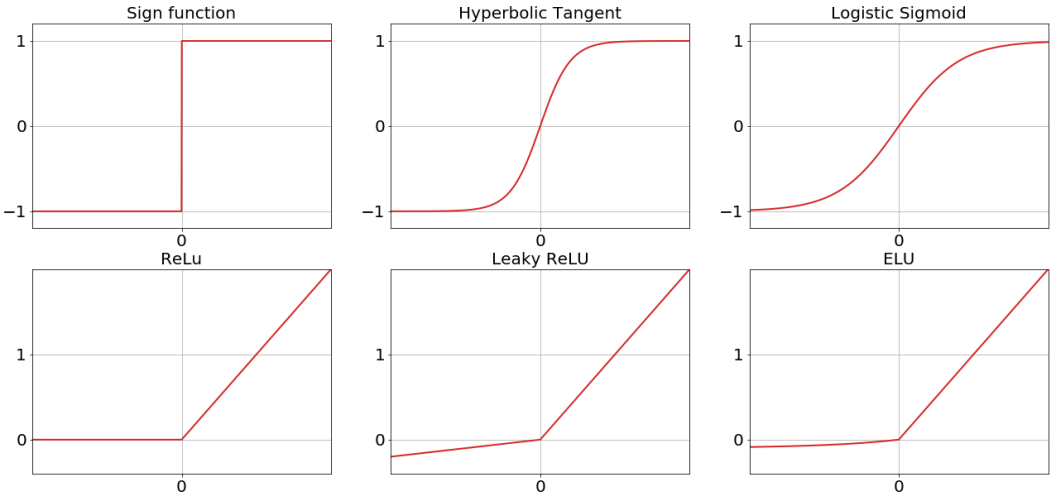


FIGURE 2.3: Examples of the most common activation functions. Usually, functions that have non-zero or infinite derivatives (such as in the four cases on the left) are preferred. Activation functions of the ReLU type (such as the three cases in the second row) are useful to have a linear response, but limited to a certain interval of values.

to the validation dataset. This is the reason why the training, validation and testing must be performed on independent datasets.

The number of iterations, during training, beyond which overfitting might start to happen can be shown by plotting the cost function calculated over the validation or testing set as a function of the number of iterations of the optimizing algorithm. Indeed, while the cost function over the training set is expected to decrease continuously, the cost function over the validation set is expected to reach a minimum and then grow again (fig. 2.4). Before the minimum, the NN weights are underfitting the data, after the minimum, they are overfitting it, hence the minimum marks the optimal amount of fitting, at least for what concerns the optimization of the cost function.

Another source of overfitting is the possible complexity of the loss function. To avoid it, a regularization term R, weighted by the hyperparameter λ, might be added to the cost function

$$\mathcal{J} = \frac{1}{N}\sum_{i=0}^{N}\mathcal{L}_i + \lambda R(\mathcal{L}) \tag{2.7}$$

in order to increase the cost function when the complexity of the loss function increases.

Dropout, Batch Normalization, Batch Segmentation Another possible problem when training, is that the network might rely on the weights of some specific nodes while others remain irrelevant. To prevent this, one can switch off a random selection of nodes at each iteration of the optimization algorithm so that the processing of the input is distributed across the whole network. This technique is called dropout.

A technique to improve training is to divide the training set in minibatches and use a different minibatch at each iteration of the optimization algorithm. The cycle of using all the minibatches is called an epoch. Many epochs might be needed to complete the training.

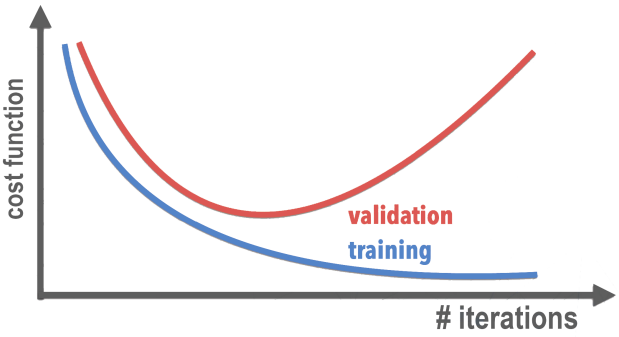


FIGURE 2.4: Qualitative example of overfitting due to the number of iterations of the optimization algorithm during training. The optimal fitting is found at the number of iterations corresponding to the minimum in the cost function computed over the validation or training set. Adapted from Krachmalnicoff, 2020.

When performing numerous numerical calculations on a machine, it is often advisable to work with normalized data in order not to risk overflow or approximation errors. So it is recommended to normalize the NN input to unit values. In addition to this, renormalizing the volume of data before each activation layers has shown to make the NN training faster and more stable. This procedure is called batch normalization and requires the following steps:

1. Before the application of the activation function, the output $z_i^{(k)}$ of a neuron i in layer k, is re-centered and re-scaled to the average $\mu_i^{(k)}$ and standard deviation $\sigma_i^{(k)}$ of the variable $z_i^{(k)}$ over the whole training set:

$$\bar{z}_i^{(k)} = \frac{z_i^{(k)} - \mu_i^{(k)}}{\sqrt{\sigma_i^{(k)2} + \epsilon}} \tag{2.8}$$

   so that the new output $\bar{z}_i^{(k)}$ has zero mean and unit variance over the training set.

2. The linear transformation:

$$\tilde{z}_i^{(k)} = \gamma^{(k)} \bar{z}_i^{(k)} + \beta^{(k)} \tag{2.9}$$

   is applied to $z_i^{(k)}$ and the result passed to the activation function. $\gamma^{(k)}$ and $\beta^{(k)}$ are optimizable weights of the network.

Batch normalization may be applied to the whole training set or to minibatches, if the training set is subdivided in that way.

2.2 Convolutional Neural Networks

There are many different setups and families of neural networks, adapted to specific computational tasks or particular data formats. This section will present Convolutional Neural Networks (CNN), which are NNs specifically structured to deal with input and/or output in the form of pixel-based images. First introduced by LeCun (LeCun et al., 1989), these kind of networks are nowadays applied in many different fields where image recognition, image reconstruction or image simulation are useful.

2.2.1 CNN Components, Hyperparameters and Architecture

The neural networks described in the previous section are structured to handle data shaped in the form of a 1-dimensional array and indeed each layer could be thought as a one dimensional array of nodes. Images, however, are in the form of 2-dimensional matrices, for monochromatic pictures, or triplets of 2-dimensional matrices for multi-color pictures[1]. It is true that a matrix of any dimension could be reshaped in the form of a vector, but it would be inefficient and ineffective to handle pictures in this way, because the spatial information encoded in nearby pixels would be lost. Therefore CNN were specifically adapted to pass data from layer to layer, maintaining it in the form of a 2 or 3-dimensional matrix.

CNN Layers Let us consider a square picture of size $L \times L \times 3$, with elements x, and a filter, i.e. a matrix of size $F \times F \times 3$, with elements w. The size of the picture must be greater than a few multiples of the filter size and the depth of the filter (i.e. its third dimension) must be the same as the picture depth (in this case 3), or in general as the depth of the 3D matrix to which it is applied. A convolutional layer of a CNN performs a discrete convolution between the 3 image color channels and the filter. By convolution, it is intended that the filter is gradually shifted along every possible position on the image and for ever position the sum of the pixelwise product:

$$\sum_i x_i w_i \tag{2.10}$$

among pixels covered by the filter is calculated and saved as a pixel value in a new picture (or matrix). Therefore the output from the convolution layer is a matrix of size $(L - F + 1) \times (L - F + 1) \times 1$. In reality, many filters of the same size, let's say N, can be applied in the same convolution layer, each producing an output matrix, making the size of the total output matrix $(L - F + 1) \times (L - F + 1) \times N$. For a schematic representation of these concepts see fig. 2.5. Then, the activation function

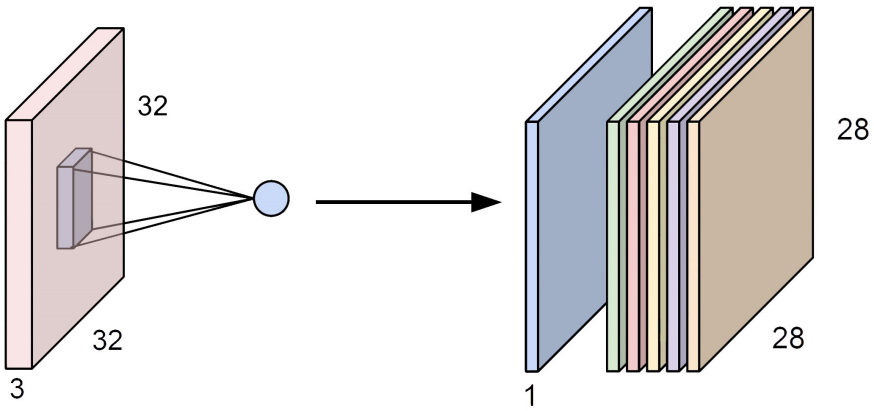


FIGURE 2.5: A convolution layer in a CNN is based on the action of filters. A filter (represented by the small blue rectangle) runs across all possible positions over the input data matrix (red rectangle). An output value, given by the dot product in eq. 2.10, corresponds to each position. The size of the output matrix is determined by the size of the input one and the size of the filter, while its depth depends on the number of filters applied. Krachmalnicoff, 2020.

is applied to the value in each pixel to produce the layer's output, which will be

[1] Colorspaces are 3-dimensional vector spaces, so each color can be defined by three parameters.

also the next layer's input. Each pixel in the newly created image then acts as a node of the neural network, while the coefficients w in the filters are the weights of the NN. Differently from the previous NNs, here each node is not connected with every node in the previous layer, but only on a subset of them, named the receptive field, determined by the shape of the filter. During the optimization procedure, the weights in the filters are optimized to detect characteristic features present in the images. Their values remain constant during a convolution, which implies that the relevance of a particular pattern, learnt by the filter, doesn't depend on its location in the image. Also, filters are not rotated, so each of them detects a particular feature with a particular orientation. The CNN is able to detect and learn as many patterns or features as the number of filters.

Other kinds of layers that are used in the CNN's design are pooling layers, which aim only at reducing the dimensionality of the data by downsampling methods, such as average value or extreme value downsampling. Convolution layers and pooling layers are usually used in CNNs that take images as inputs and perform classification. However, in CNNs where the output is an image as well, also deconvolution and upsampling layers are used. These kinds of layers just perform the reverse actions as those described above.

Usually the architecture of a CNN consists of a sequence of alternating convolution layers and pooling layers plus some fully connected layers. As the input is processed further in the layers, the dimension of the image is reduced while the image depth grows. See fig. 2.6 as an example of a CNN architecture

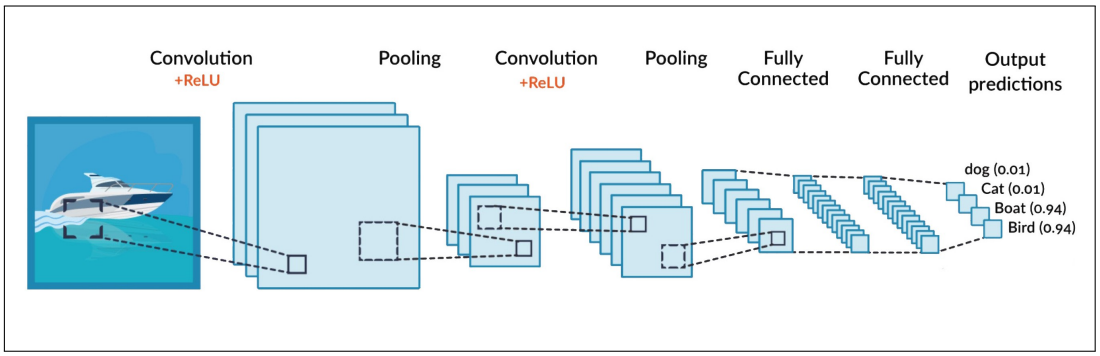


FIGURE 2.6: Example of a possible CNN architecture. The diagram illustrates the evolution of the data size and depth as it advances across fully connceted, pooling and convolution layers. This specific example refers to an image classification CNN. Krachmalnicoff, 2020.

Hyperparameters The hyperparameters that must be chosen in a convolution layer are:

- the size of the receptive field F, i.e. the size of the first two dimensions of a filter;
- the number N of different filters;
- the stride S, i.e. the length, in pixel units, of the displacement of the filter between two consecutive positions;
- the zero padding P, i.e. the number of pixel columns and rows with zero values that are added to the input image, if a precise output size is required;

while those for a pooling layer are:

- the downsampling ratio, i.e. the ratio between the size of the output image and the input image;
- the downsampling method, e.g. average value, extreme value, etc.

A convolutional layer with hyperparameters F, N, S and P, transforms an input matrix of volume $L \times L \times D$ into an output matrix of volume $L' \times L' \times N$ with:

$$L' = \frac{L - F + 2P}{S} + 1 \tag{2.11}$$

2.3 Generative Adversarial Neural Networks

A particular use of CNNs is in Generative Adversarial Networks (GANs) which, as the name says, consists in a framework for data generation through an adversarial process between two competing neural networks. Presented in 2014 by Goodfellow et al., it has now become one of the main tools for image generation (Goodfellow et al., 2014). In the following, GANs will be discussed in the specific case where the neural networks involved are CNNs.

A GAN is made of two neural networks that are trained together. One is the discriminator (D), whose task is to distinguish between images that belong to a reference set (the training set) and images that don't. The other is the generator (G), whose task is to create images that are as similar as possible to those in the reference set, so that D cannot distinguish among them. More formally, D is a function that receives an image (2D array) as input and returns a value between 0 and 1 which represents the probability that the image belongs to the training set as opposed to being generated by G. G, instead, is a function that maps samples from a random noise distribution (in the form of a 1D or 2D array), taken as input, to output images belonging to a certain distribution. During training, the weights of D are optimized so that images from the training set are assigned a probability of 1 and images made by G are assigned a probability of 0, while at the same time the weights of G are optimized so that D makes a mistake (i.e. D assigns 1 to images from G). The competition between these two optimization processes leads to a unique point in the parameter space where the weights are optimized so that images from the training set and images from G are both assigned the same probability, which turns out to be $1/2$.

Let's put this into formulas. We call x an image belonging to the training set and p_X the underlying distribution of reference images. In the same manner, we call z a noise realization and p_Z the distribution from which the sample Z is drawn. The output of the generator is $G(z)$ while that of the discriminator is $D(x)$ for reference images and $D(G(z))$ for generated images. A possible loss function can be defined over pairs of reference and generated images as:

$$\mathcal{L}_{D,G}(x, z) = \log(D(x)) + \log(1 - D(G(z))) \tag{2.12}$$

with a correspondent cost function:

$$\mathcal{J}_{D,G} = \langle \log(D(x)) \rangle_{p_X} + \langle \log(1 - D(G(z))) \rangle_{p_Z} \tag{2.13}$$

A loss function of this kind is called binary cross-entropy loss function. Binary because it is used in a binary classification of an input in two categories, cross-entropy because it contains the Von Neumann entropy formula summed for two different variables. The overall training process aims at finding the minimum over all the

possible functions G of the maximum over all the functions D of $\mathcal{J}_{D,G}$. The simultaneous training of D and G occurs at alternating steps. First, keeping G fixed, D undergoes $k \gtrsim 1$ steps (k is an hyperparameter) of the chosen iterative optimization method, then G undergoes 1 step of the optimization method, and this is all repeated until convergence is reached. The balance between the optimization steps of the two networks is set up so that D remains close to its optimal value but doesn't reach it, in order to avoid overfitting, while G changes slowly compared to D. A possible implementation of the GAN training algorithm, is the following:

1. sample m reference elements x_i from p_X,
2. sample m noise realizations z_i from p_Z,
3. update only the weights of D using a gradient method with:

$$\nabla_{w_D} \frac{1}{m} \sum_{i=1}^{m} [\log (D(x_i)) + \log (1 - D(G(z_i)))] \tag{2.14}$$

4. repeat steps 1-3 for k times
5. sample m noise realizations z_i from p_Z,
6. update only the weights of G using a gradient method with:

$$\nabla_{w_G} \frac{1}{m} \sum_{i=1}^{m} [\log (1 - D(G(z_i)))] \tag{2.15}$$

7. repeat from point 1 until convergence.

Fig. 2.7 shows a possible structure for a GAN network.

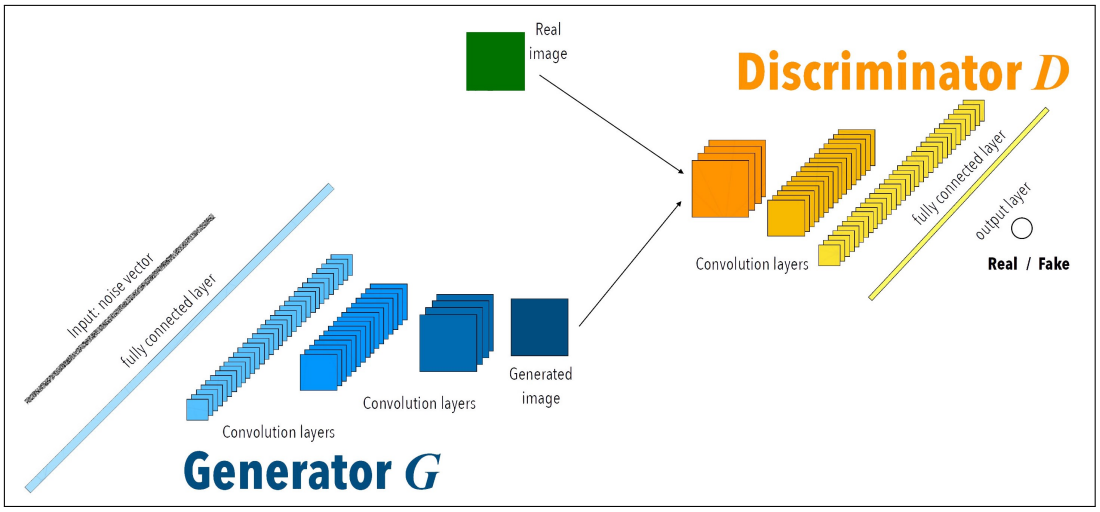


FIGURE 2.7: Example of a possible GAN architecture example. The Generator NN (in blue) evolves an input noise vector across deconvolution layers to produce an image. The Discriminator NN (in orange) receives an image as input and, after a sequence of convolution layers, classifies it into real or fake.

2.3.1 Deep Convolutional GANs

Convolutional Generative Adversarial Networks look like a powerful tool for generating artificial but realistic images. However, a problem is that, in general, the training of convolutional GANs is unstable and may result in non effective generators. In particular, training is difficult when trying to upscale a GAN network to deeper networks and to handle higher resolution images (which means a greater number of input output data). To ensure an efficient and effective training, Radford and Metz established a set of characteristics that the architecture of the convolutional GAN should display (Radford, Metz, and Chintala, 2016). A networks of this type is named deep convolutional generative adversarial network (DCGAN) and it should have the following features:

- All upsampling and downsampling layers in the network should be convolutional or deconvolutional layers rather than pooling or upsampling functions. This allows the network to learn its own down(up)sampling rather than having a deterministic one.
- Input and output layers should be connected directly to convolution layers, rather than to an intermediate fully connected layer. This increases convergence speed, without reducing stability too much.
- Apply batch normalization to each layer except to the generator's output and the discriminator's input. Normalization is helpful for a correct flow along the gradient in the optimization process and ensures that the generator doesn't collapse to one point (i.e. it returns the same output for any input). A overuse of normalization, though, might lead to model instability (i.e. the generator returns completely different outputs for similar inputs). The above prescription finds the balance between the two extremes.
- Implement the Tanh activation function in the generator's output layer, the ReLU activation function in all other layers of the generator and the LeakyReLU in all the discriminator's layers.

The training of DCGANs has proven to be successful in image recognition and image generation applied to images of handwritten digits, faces, house interiors and various subjects. So DCGANs' present a solid architecture that promisingly can be applied to the most various image sets.

2.4 Neural Networks Applications in Cosmology

Astrophysics and cosmology are fields where very often data comes in the form of images, e.g. sky maps or objects surveys. So it doesn't come as a surprise that in the last years GAN-based algorithms have found many application for different kinds of simulations, such as increasing resolution of observational or simulated images, adding or separating features from sky maps or estimating parameters from complex models. The following paragraphs give a short and non-comprehensive overview of some interesting applications.

Dark matter simulations On the large scales of the universe, gravity shaped the distribution of dark matter into complicated structures, known as cosmic web. The current theoretical model for the evolution of the universe is the lambda cold dark

matter model, which gives a mechanism for the growth of structures from the primordial fluctuations after the inflationary period. The ΛCDM predictions about the distribution of structures depend on the values of the model parameters. Therefore, it is important to simulate the evolution of structures for different values of the model parameters, in order to confront the latter with the estimates of parameters from galaxy surveys and CMB observations. The formation of large scale, dark matter structures, for a given set of cosmological parameters, can be studied with classical N-body simulations, that evolve the system, i.e. trace the particles' positions and velocities in a volume of space, from a certain initial condition up to the present epoch. N-body algorithms are, by their nature, computationally very expensive, with computational time growing fast with resolution or scale. Recently, some huge numerical simulations have been carried out (e.g. Springel et al., 2005), but there is still the need to obtain large scale, high resolution results in feasible computational time. Convolutional neural networks offer an efficient solution and can be useful in different ways. For example, a GAN can be trained on a set of 2D or 3D square arrays that represent a snapshot of the particle density from a dark matter N-body simulation (Ullmo, Decelle, and Aghanim, 2021, Rodríguez et al., 2018). The GAN learns the patterns and features of the particle density function and then is able to generate other 2D or 3D snapshots that are statistically indistinguishable from the original ones (fig. 2.8). In this way new data is generated, without the need of additional simulations. Another way in which GANs can be employed in dark matter simulations is by using them to increase the resolution of a simulation. Indeed, a GAN can be trained to learn the function that links the density distribution resulting from a simulation at low spatial resolution (LR), to the density distribution from a simulation with the same initial conditions but higher spatial resolution (HR). Once the GAN is trained, it is possible to obtain HR simulations at the computational cost of LR simulations only (Kodi Ramanah et al., 2020, Li et al., 2021).

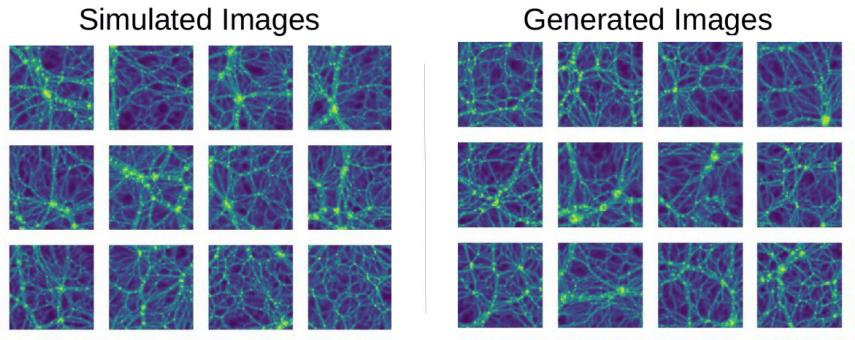


FIGURE 2.8: On the left, images from 2D dark matter N-body simulations. On the right, images generated from a GAN trained on the reference set of images on the left. Adapted from Ullmo, Decelle, and Aghanim, 2021.

Dark matter and galaxy distribution Simulating the complete evolution of galaxies is even more computationally expensive than simulating dark matter because, in addition to gravity, also hydrodynamics, electromagnetic forces and radiative processes must be taken into account. It has been shown that there is a strong correlation between the galaxy and dark matter distributions (Wechsler and Tinker, 2018). So CNNs, with a particular U-net architecture (Ronneberger, P.Fischer, and Brox, 2015), can be trained on input/output pairs of correspondent dark matter/galaxy distributions, in order to learn the underlying relation between the two. Then the

trained network can be applied to new dark matter distributions, to produce galaxy distributions in a shorter computational time (Zhang et al., 2019).

Galaxy features recovery and classification Large sky surveys provide catalogs of numerous images of astronomical objects. Background and instrumental noise limit the resolution of these images, especially for distant objects. Classical techniques, such as deconvolution, can be employed, to recover object features ruined by noise, but their effectiveness is limited. Convolutional Networks then may come into play. A convolutional GAN can be trained on pairs of degraded (noisy) and non-degraded pictures to recover the mapping from pictures with few noisy features to pictures with many sharp features. Once the network is trained, the mapping can be applied to other degraded images. This method has been applied by Schawinski et al., 2017 to images of galaxies from the Sloan Digital Sky Survey (SDSS). They showed that GANs were able to recover many more and less noisy features than conventional methods. In the context of galaxy surveys, CNNs are also useful to classify the type of galaxy present in a picture based on its shape (e.g. Zhu et al., 2019). For this kind of task, the network needs a supervised learning, during which it is trained on catalogs of galaxies already classified by hand.

Galactic foreground In CMB data analysis, convolutional networks are useful in modeling galactic foreground, in the prospect of separating the foreground emission from the background signal. Deep convolutional neural networks (DCNN) have found application for reconstructing masked parts of sychrotron and dust emission maps (Puglisi and Bai, 2020). They showed effective in inpainting a non-Gaussian signal, without altering the rest of the statistics of the emission. Using a DCGAN, Aylor et al., 2020, were able to model total intensity galactic dust foreground emission with non-Gaussian features, training the networks on maps from Planck satellite. The DCGAN was trained on output square maps covering 1% of the sky, and it learned to generate emission maps with the same features of the training set. The work showed promising results for extending the application to polarized foreground emission maps and to larger portions of the sky. These works show that CNNs are a powerful tool to generate non-Gaussianities in foreground emissions and reproduce different realizations of foreground maps.

Just to give an idea of the wide range of applicability of GANs in cosmology and astrophysics, other successful applications can be mentioned, such as the use of GANs to create weak-lensing convergence maps that are statistically indistinguishable from the simulated ones (Mustafa et al., 2019), or the application of unsupervised DCGANs for recognizing chemical characteristics of exoplanets atmospheres(Zingales, T. and Waldmann, 2018).

Chapter 3

ForSE Package

As explained in section 1.5, galactic emission is well known at scales greater than 1°, but detailed radiation maps are not available at sub-degree angular scales for the full sky (or for large portions of it). In preparation for the next generation of CMB experiments, it is important to characterize as precisely as possible the galactic emissions at all angular scales, since it has a critical impact on the observations of gravitational lensing and primordial B-modes. To obtain a clean CMB signal, the foreground emission from the galaxy must be removed from the total signal, both in total intensity and polarization. The component separation algorithms that perform this procedure, require a solid and detailed model of the galactic emission, otherwise there is risk of introducing spurious signals in the observations of the background. Foreground models are also used in the algorithms that estimate the gravitational lensing potential in order to delens the CMB maps.

Models of the galactic emission are complicated, because of the high degree of non-Gaussianities, time variability and the complicated emission features at all scales. Available models are based on templates of the galactic emission (e.g. the Python Sky Model and the Planck Sky Model), which allow to simulate the emission at different microwave frequency channels. The maps from which these models are created are not reliable at sub-degree scales because of the noise that contaminates the signal. A way around this, is to extrapolate the power spectrum of the emission template with a power law model up to multipoles corresponding to sub-degree scales. Then to create a Gaussian field realization that has the same power spectrum and add it in the sub-degree scales of the model. The models created with this procedure have a realistic amplitude (i.e. a correct power spectra), so at first order they are correct, but the higher order n-point correlation functions, which determine the statistics of the features in the emission maps, are not the same. Another possibility for creating galactic emission models is to start from magnetohydrodynamical numerical simulations of the interstellar medium. This method builds a model that includes non-Gaussian statistics, but has no correspondence with the large scale (super-degree) features of the radiation. As shown by Beck, Errard, and Stompor, 2020, neglecting to model non-Gaussianities in the synchrotron and dust emissions models can lead to biased estimates of the B-modes power spectrum and of the lensing potential, which in turn leads to incorrect delensing of the CMB polarized signal.

Recently, Krachmalnicoff and Puglisi, 2021 presented a new approach to galactic foreground modeling, in particular thermal dust emission modeling, making use of Neural Networks. They developed a Python package, called "Foreground Scale Extender" (ForSE), based on a DCGAN that was trained to learn the mapping from the galactic emission maps in low resolution (i.e. maps with super-degree scales features) to the corresponding maps in high resolution (i.e. maps with sub-degree

scales features)[1]. The learning was carried out over parts of the sky where maps are available in both low resolution and high resolution. Afterwards, the trained network generates the high resolution maps for parts of the sky which previously were availale only in low resolution. The strength of this approach, which was applied both to total intensity and polarization signals, is the ability to generate maps with small scale features that display non-Gaussianities and large scale features alike the real ones.

The topic of this thesis builds on the work of the ForSE package. So, in the following, I will explain the methodology used in its development, regarding network architecture, network training, image analysis and data analysis, and present the main results obtained.

3.1 Application of DCGANs to Thermal Dust Emissions

The goal of the project was to train a DCGAN to receive as input a low resolution map of the thermal dust emission from the galactic plane and return as output a high resolution version of that same map, for both total intensity and polarization maps. The maps used to train the network were obtained with component separation methods from Planck maps at a frequency of 353 Hz (Planck Collaboration XLVIII, 2016), and precisely they are the *2018 Stokes I, Q and U Thermal Dust map from GNILC*, available at the Planck Legacy Archive[2]. These maps have an angular resolutions that varies depending on the signal-to-noise-ratio, with an effective beam full-width-half-maximum (FWHM) varying from 5 arcminutes to 80 arcminutes.

To create a dataset of input-output pairs over which the network could be trained, the above fullsky maps were decomposed in square patches, using the Healpy[3] Python package, based on the HEALPix (Hierarchical Equal Area isoLatitude Pixelation) pixelization scheme (Gorski et al., 2005). Each patch has dimension of $320px \times 320px$, and covers a $20° \times 20°$ angle of the sky. Each patch is a monochromatic image, meaning that it can be represented by a square matrix of real values, with size corresponding the number of pixels per side.

We can assume that a patch M of an emission map, with angular resolution r_{SS}, is given by the sum of a patch M_{LS} encoding the large scale structures up to a resolution r_{LS}, plus a patch $M_{SS} = M_{LS} \cdot m_{SS}$ containing the small scale structures m_{SS} up to a resolution r_{SS}, modulated by the large scales structures.

The patch decomposition is:

$$M = M_{LS} + M_{SS} = M_{LS}(1 + m_{SS}) \tag{3.1}$$

or equivalently:

$$\frac{M}{M_{LS}} = \tilde{m}_{SS} = 1 + m_{SS} \tag{3.2}$$

The goal of the NN training is to learn the underlying relation that maps patches M_{LS} to patches $\tilde{m}_{SS}$.

Note that, if not otherwise specified, by *resolution* it is intended the resolution of the features shown in the patch of the full-sky map, i.e. the scale of the smallest

[1]The package, with code, input data and results, is available for public access at https://github.com/ai4cmb/ForSE.

[2]Planck Legacy Archive: https://pla.esac.esa.int/#home.

[3]Healpy documentation: https://healpy.readthedocs.io/en/latest/.

features, not the pixel resolution, which is referred to rather as *number of pixel*, *pixel size* or similar expressions. It is also important to specify that the two are not related. Indeed it is possible to have maps with the same resolution and different pixel size or vice versa.

3.1.1 DCGAN Architecture

The network setup used in ForSE is that of a DCGAN, very similar as the one described in Radford, Metz, and Chintala, 2016 (see chapter 2). The main difference is that the generator takes as input not an array of noise, but rather a large scale (LS), low resolution map M_{LS}. The generator's output is a small scale (SS), high resolution map $\tilde{m}_{SS}^{gen}$ that should resemble the reference set SS maps $\tilde{m}_{SS}^{real}$.

The architecture of the GAN is composed as follows. The generator G consists of

- 1 convolution layers, with a 5×5 kernel, no stride and 64 filters, which takes input in the form of 320×320 matrices;
- 1 convolution layers, with a 5×5 kernel, stride equal to 2 and 128 filters;
- 1 convolution layers, with a 5×5 kernel, stride equal to 2 and 256 filters;
- 3 upsampling layers combined with convolution ones, with dimensions symmetric to the first three layers.

The discriminator D has

- 3 convolution layers identical to those in the generator;
- a flattening layer that arranges the output of the previous layer in a 1D array;
- an output node, densely connected to the previous layer

In all encoding layers (first three) for both G and D, the activation function is the LeakyReLU. Batch normalization is applied at all steps. The output node in D is activated with a sigmoid function. The loss function is binary cross-entropy, the optimization algorithm is Adam. For a schematic idea of the GAN structure refer to fig. 3.1.

3.1.2 DCGAN Training and Testing

The DCGAN approach to thermal dust emission can be applied both to maps of the signal in total intensity and to maps of polarized signals in Q and U decomposition. However, the truly relevant application is the second one. Indeed, the next generation of CMB experiments will focus on the polarized signal to look for traces of primordial B-modes and to reconstruct the gravitational lensing of the CMB. Also, for polarized radiation, the resolution of foreground maps is consistently worse than in total intensity maps, because the signal is less powerful. For GNILC (Generalized Needlet Internal Linear Combination) thermal dust emission maps in total intensity, the FWHM of the effective beam ranges between 5 and 22 arcminutes, with more than 40% of the sky were it is below 12 arcminutes. For the same maps in polarization, the FWHM ranges between 5 and 80 arcminutes, staying lower than 12 arcminutes in less than 9% of the sky. Therefore, the NN resolution enhancement can be tested first on the total intensity map and successively applied to the polarization maps. Below I will describe the methodology for the training on total intensity maps and later explain the specifics of its application to polarization maps.

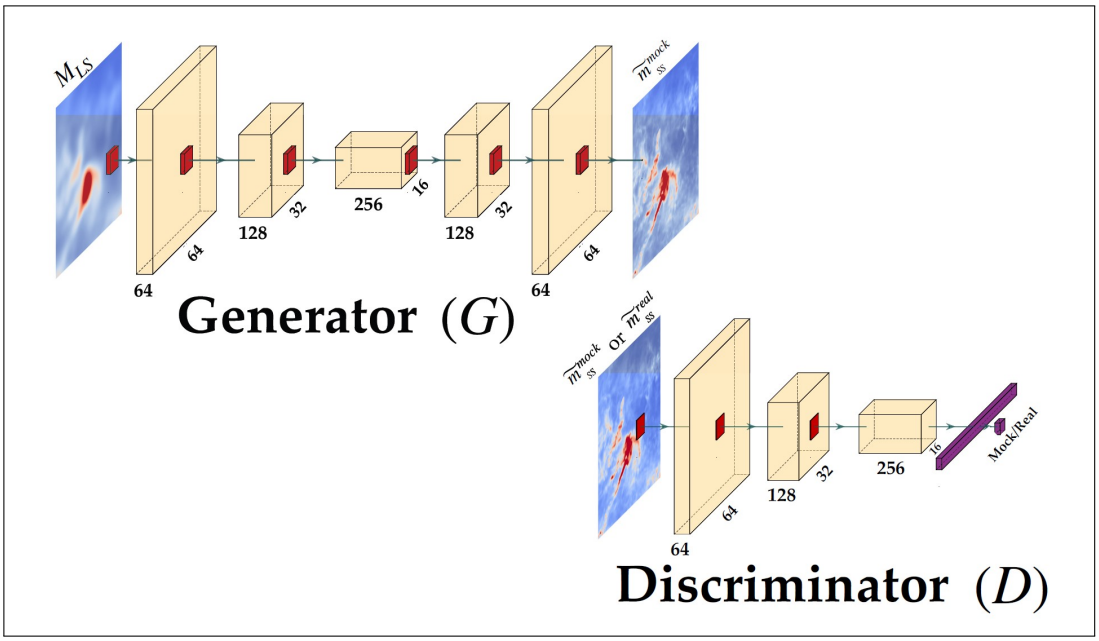


FIGURE 3.1: Explanatory diagram of the architecture of the DCGAN used in ForSE. Arrows represent (de)convolution plus activation layers, red rectangles are filters and yellow parallelepipeds represent the volume of the data matrix at each step. Krachmalnicoff and Puglisi, 2021.

Total Intensity Training A specific area of the whole full sky map was selected with the requirement of having a SNR above 8 and galactic latitude above 10°[4]. Pairs of high-resolution/low-resolution patches $(M_{LS}, \tilde{m}^{real}_{SS})$ in the training set were selected only if at least 80% of the patch was covered by such mask (fig. 3.2). To create a small scale (SS) map and a large scale (LS) map, two smoothings of the original map were applied, one with a beam of 12′ FWHM (for SS) and one with 80′ FWHM (for LS). A total of 350 randomly located and oriented pairs of square patches of $20° \times 20°$ side and 320×320 px were created from the smoothed full sky maps in the masked area. Each patch was normalized in $[-1, +1]$ before training.

From this set, the training was performed with batches of $N_b = 16$ pairs of patches alternating the optimization of the two networks in the following way:

1. For N_b times, an LS patch M_{LS} was given as input to G, which generated an SS output patch $\tilde{m}^{gen}_{SS}$, which was given as input to D. The cost function was computed over the N_b outputs from D. The parameters of G were optimized to maximize the probability that the patches $\tilde{m}^{gen}_{SS}$ were classified as real by D.

2. Afterwards, $N_b/2$ patches $\tilde{m}^{gen}_{SS}$ and $N_b/2$ patches $\tilde{m}^{real}_{SS}$ were given as input to D and the cost function computed over the N_b outputs from D. The parameters of D were optimized to maximize the probability of classifying the patches correctly.

The two steps of the above training process were iterated for 10^5 times, and the weights of D and G were saved every 10^3 iterations. To select the best NN parameter setup among the 10^4 saved ones, the generated patches were compared to the

[4]Portions of the sky too close to the galactic plane are overwhelmed with foreground signal and thus are not exploitable for CMB measurements.

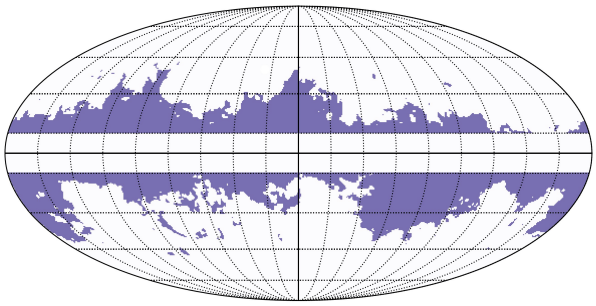


FIGURE 3.2: Sky mask used for selecting the square patches used in the training set. Patches were selected only if at least 80% of the patch was covered by the colored area. Krachmalnicoff and Puglisi, 2021.

training set, using a statistical analysis based on the Minkowski functionals. The following section will describe in more detail what are these functionals and why they are apt to quantify the statistical properties of images.

Polarization Training For polarization maps, the area of the sky where the resolution is above 12′ is very small (9% of the sky) and mostly close to the galactic plane. So it was not possible to create an other mask from which realize enough patches for the training set. To go around the problem, an assumption was introduced, which is that the total intensity and the polarized field have morphological features that share the same statistic. This is reasonable because the source of the polarized and unpolarized emissions is thought to be the same (i.e. grains of dust) and their two-point correlation functions are similar. Though reasonable, it is still an arbitrary assumption, motivated by the fact that with the available data this is the best guess we can do. Also, it should be clarified that the fact that the fields share the same statistics does not mean that they have also the same morphology, i.e. that they present spatially correlated realizations of that statistic.

Using this assumption, the training of the DCGAN for the Q and U maps can be done exactly as before, but using Q(U) maps smoothed at 80′ as large scales patches (M_{LS}), paired with total intensity maps smoothed at 12′ as small scale patches ($\tilde{m}^{real}_{SS}$). Again the 350 patches in the training set were selected from the masked area. The training procedure was carried out in the same way as before, as well as the testing of the training results.

3.2 Minkowski Functionals

Evaluating quantitatively the features of noisy images and comparing them to statistical models is not always a trivial task. When the image contains structures with a high degree of symmetry and order, it might be sufficient to compute variance, two-point correlation functions and Fourier transforms of the density field (see definition below) to capture the main characteristics. These descriptors, however, give information only about characteristic distances, failing to capture morphological features. Then, for images where patterns are of random and disordered nature, they become insufficient and we need to define more adequate mathematical tools. Here I will follow the definition of Minkowski functionals and their application to pixel-based images given in Mantz, Jacobs, and Mecke, 2008.

Monochromatic 2D images can be thought as a density function $\rho(x)$ defined on a compact convex set of a two dimensional vector space: $x \in C \subset \mathbb{R}^2$. The density

function takes values in a finite interval that represents the intensity of the color, on a grayscale from black to white. For the moment, we will treat ρ as a continuous function over a continuous domain, while later we will generalize to a discretized function over a quantized domain, such as in the case of pixel-based images. To describe the spatial features of structures created by the density function, we can introduce, from the field of integral geometry, a family of morphological descriptors, called Minkowski functionals.

In a d dimensional space, for a compact set A, $d+1$ Minkowski functionals $m_0, \ldots, m_d$ can be defined. Calling ∂A the boundary of A, $\omega_d = \pi^{d/2}/\Gamma(1+d/2)$ the volume of the ball in d dimensions and R_i the principal radii of curvature for $i = 1, \ldots, d-1$, Minkowski functionals can be defined as:

$$m_0(A) := \int_A \mathrm{d}V \tag{3.3}$$

for the first one and as:

$$m_\nu(A) := \frac{\omega_{d-\nu}}{\omega_\nu \omega_d} \frac{(\nu-1)!(d-\nu)!}{d!} \int_{\partial A} \sum_{\{i_1,\ldots,i_{\nu-1}\}} \frac{1}{R_{i_1} \cdot \ldots \cdot R_{i_{\nu-1}}} \mathrm{d}S \tag{3.4}$$

for $\nu \geq 1$.

In the application to 2D pictures, the relevant case is $d = 2$, for which we have:

$$m_0(A) = \int_A \mathrm{d}^2\vec{r} \tag{3.5}$$

$$m_1(A) = \frac{1}{2\pi} \int_{\partial A} \mathrm{d}\vec{r} \tag{3.6}$$

$$m_2(A) = \frac{1}{2\pi^2} \int_{\partial A} \frac{1}{R} \mathrm{d}\vec{r} \tag{3.7}$$

The first functional gives the covered area of the set, the second functional is proportional to the length of its perimeter while the third one is proportional to its Euler characteristic, which is the difference between the number of connected domains and holes.

An important step that allows the generalization of these definition to discrete domains, is a completeness theorem by Hadwiger that states the following.

Theorem 1 (Hadwiger) *All and only the functionals $\mathcal{F}(A)$, $A \subset \mathbb{R}^n$, A compact and convex, that can be written as a linear combination of Minkowski functionals*

$$\mathcal{F}(A) = \sum_{\nu=0}^{d} f_\nu M_\nu(A) \quad , \quad f_\nu \in \mathbb{R} \tag{3.8}$$

possesses the properties of additivity, motion invariance and continuity.

Additivity means that the functional of the union of two subset is the sum of the functional of each subset minus the functional of the intersection of the two:

$$\mathcal{F}(A \cup B) = \mathcal{F}(A) + \mathcal{F}(B) - \mathcal{F}(A \cap B) \tag{3.9}$$

The motion invariance property requires that the functional is independent of the spatial position and orientation of the subset:

$$\mathcal{F}(gA) = \mathcal{F}(A) \tag{3.10}$$

where $g \in \mathcal{G}$ is an element of the translations and rotations group. Continuity implies that if a sequence of subsets converges, $A_n \to A$ for $n \to \infty$[5], then also the functional of the sequence converges:

$$\mathcal{F}(A_n) \to \mathcal{F}(A) \tag{3.11}$$

The first property is relevant because it allows to extend the computation of functionals over convex sets to the computation of functionals over unions of convex sets and a pixel-based image can be described as a real valued function over a domain made by the union of compact convex sets, (for example each pixel is a compact convex set). The third property, which means that if a set A_n is an approximation to the set A, then also the functional $\mathcal{F}(A_n)$ is an approximation to $\mathcal{F}(A)$, is useful because, in pixel-based images, complex curved shapes are approximated by a tessellation of square pixels (from now on called pixelization). Finally, the second property is not specifically related to discretized sets, but just states that the evaluation of spatial features carried out by Minkowski functionals doesn't take into consideration position and orientation of shapes, which is an important attribute when working with homogeneous and isotropic random fields.

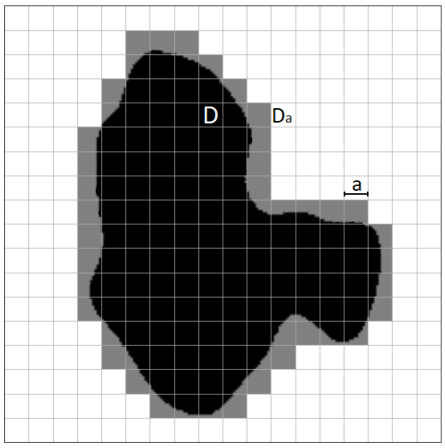


FIGURE 3.3: Pixelization example for computing discrete Minkowski functionals. As the pixel side length a tends to 0, the area of the pixelization D_a tends to the area of the shape D. The perimeter of the pixelization, however doesn't converge to that of the shape, so other methods must be used to compute it. If a is sufficiently small, the Euler characteristic of the pixelization coincides exactly with that of the original shape.

Let's discuss how to compute in practice the three Minkowski functionals of a 2D generic domain, named D, approximated with a pixelization of squares with side a, named D_a. As an example one can look at fig. 3.3.

- Computing the first Minkowski functional means to compute the area of D. Since the area of D_a tends to the area of D for $a \to 0$, the functional $m_0(D_a)$ approximates correctly the value $m_0(D)$.
- The computation of the second Minkowski functional requires a more complex approach because the perimeter of D_a doesn't converge to the perimeter of D. To approximate correctly the perimeter P of a shape D from its pixelization D_a, a common algorithm, named marching squares (Lorensen and Cline, 1987), is

[5]Convergence is intended as convergence of the Hausdorff distance of the subsets A_n

usually used. The algorithm approximates D with a polygon, based on interpolations between pixel values and it provides an approximated perimeter P_a that converges to P for $a \to 0$. The functional $m_1(P_a)$ approximates correctly the value $m_1(P)$.

- The last Minkowski functional is easier to compute because it is purely a topological descriptor. The Euler characteristic can be computed as the difference between the numbers N_+ and N_- of concave and convex angles of D_a, divided by four times the number of pixels N_{px}: $m_2(D_a) = m_2(D) = \frac{1}{2\pi}\frac{(N_+ - N_-)}{4N_{px}}$.

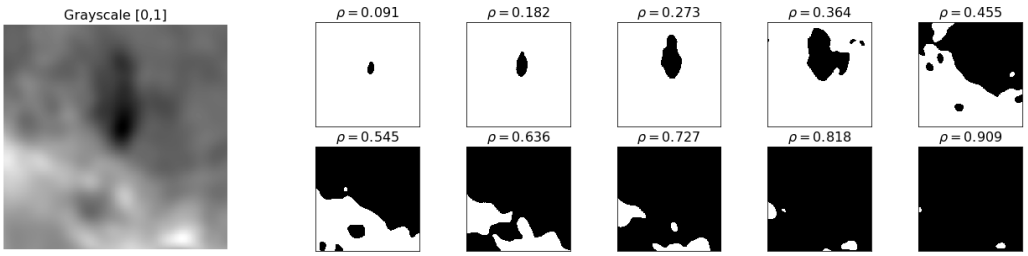


FIGURE 3.4: Thresholding example. The color scale of the image on the left contains 256 values, parametrized between 0 and 1. Thresholding an image means to set all pixels with color values lower than a threshold ρ to black and all pixels with higher color values to white. Pictures on the right are thresholded versions of the original picture, for ten different threshold values.

Up to this point, pixelized images were assumed to be 2-color, black and white pictures. One can apply the above concepts also to 256-color, grayscale images by first performing thresholding. Thresholding consists in choosing a value ρ on the grayscale (which is conventionally made of 256 values between 0 and 1) and then setting to 0 (black) all pixels with values lower than ρ and to 1 (white) all pixels with higher values. In this way, a two-color image is created. For a given 256-color grey-scale image, 256 different thresholdings can be performed, resulting in 256 different images (see fig. 3.4). Minkowski functionals can be computed for the black pixel subsets present in each thresholded image, making them function of the thresholding value: $m_i = m_i(\rho)$.

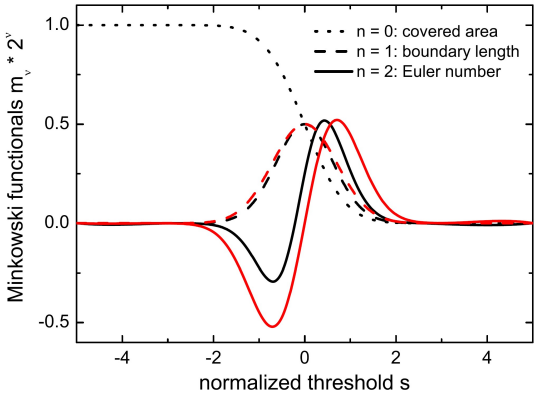


FIGURE 3.5: Exact Minkowski functionals $m_n(s)$ as functions of the normalized threshold $s = \rho/\sqrt{2\sigma^2}$. In black the random lattice, while in red the Gaussian random field. For $n = 0$ the two functionals coincide, but greater differences appear as n increases. Mantz, Jacobs, and Mecke, 2008.

For some of the most common models of statistical fields, such as the random lattice model and the Gaussian random field, the dependence of Minkowski functionals can be calculated analytically. Figure 3.5 shows the exact ρ dependence of the three Minkowski functionals for the random lattice and the Gaussian random field.

3.3 DCGAN Training Results

3.3.1 Total Intensity Results

To check how well the trained DCGAN was able to replicate the statistical properties of real small scale patches from the total intensity map, Minkowski functionals were computed on all the 350 real patches $\tilde{m}_{SS}^{real}$ and on the corresponding 350 generated patches $\tilde{m}_{SS}^{gen}$. Average value and standard deviation of the functional were computed as function of threshold for all three functionals at each saved iteration of the training procedure. The superposition of the areas included between ± 1 STD from the average functionals of the real patches and the average functionals of the generated ones gives a measure of how the training was successful. The NN parameters corresponding to the iteration that gave the highest percentage of superposition in Minkowski functional were chosen as those of the trained network. Fig. 3.6, shows the comparison of Minkowski functional of the real SS patches and of the SS patches generated from the trained network. The agreement is pretty good, with percentages of 76%, 84%, 91% for m_0, m_1 and m_2 respectively. In fig. 3.7, three examples

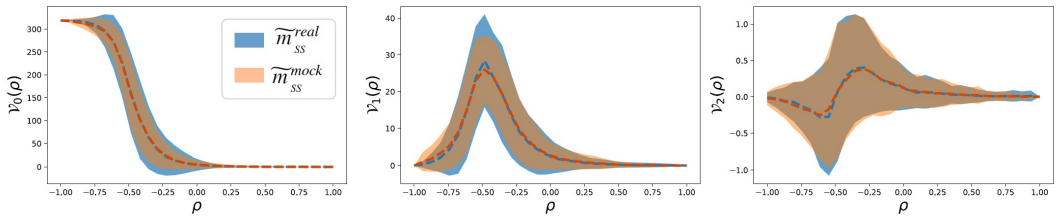


FIGURE 3.6: First three Minkowski functionals (here denoted as ν_n) as function of the threshold ρ, computed for both real (blue) and generated (orange) SS patches of the total intensity map. The diagrams show the average functional over the 350 patches from the training set and the 350 corresponding generated patches, plus and minus 1 STD. Krachmalnicoff and Puglisi, 2021.

are shown of LS input patches (M_{LS}), with corresponding real SS patch ($\tilde{m}_{SS}^{real}$) and generated SS output patch ($\tilde{m}_{SS}^{gen}$).

Since the training over total intensity maps resulted successful, the NN approach for generating small scales was extended also to polarization maps.

3.3.2 Polarization Results

With the assumptions and methods described in section 3.1.2, two different DCGANs were trained for Q and U maps to learn the mapping from Q/U large scale patches ($M_{LS}^{Q/U}$) to total intensity small scale patches ($\tilde{M}_{SS}^{Ireal}$). Again, the parameters of the trained networks were chosen by looking at the superposition of Minkowski functionals. The two sets of images on which Minkowski functionals were computed, were the set of 350 patches of real total intensity small scales and the set of 174 small scale patches generated by the trained DCGANs. Results were positive, for both Q and U trainings: the best superposition percentage were (79%, 85%, 89%) and (87%, 84%, 87%) respectively.

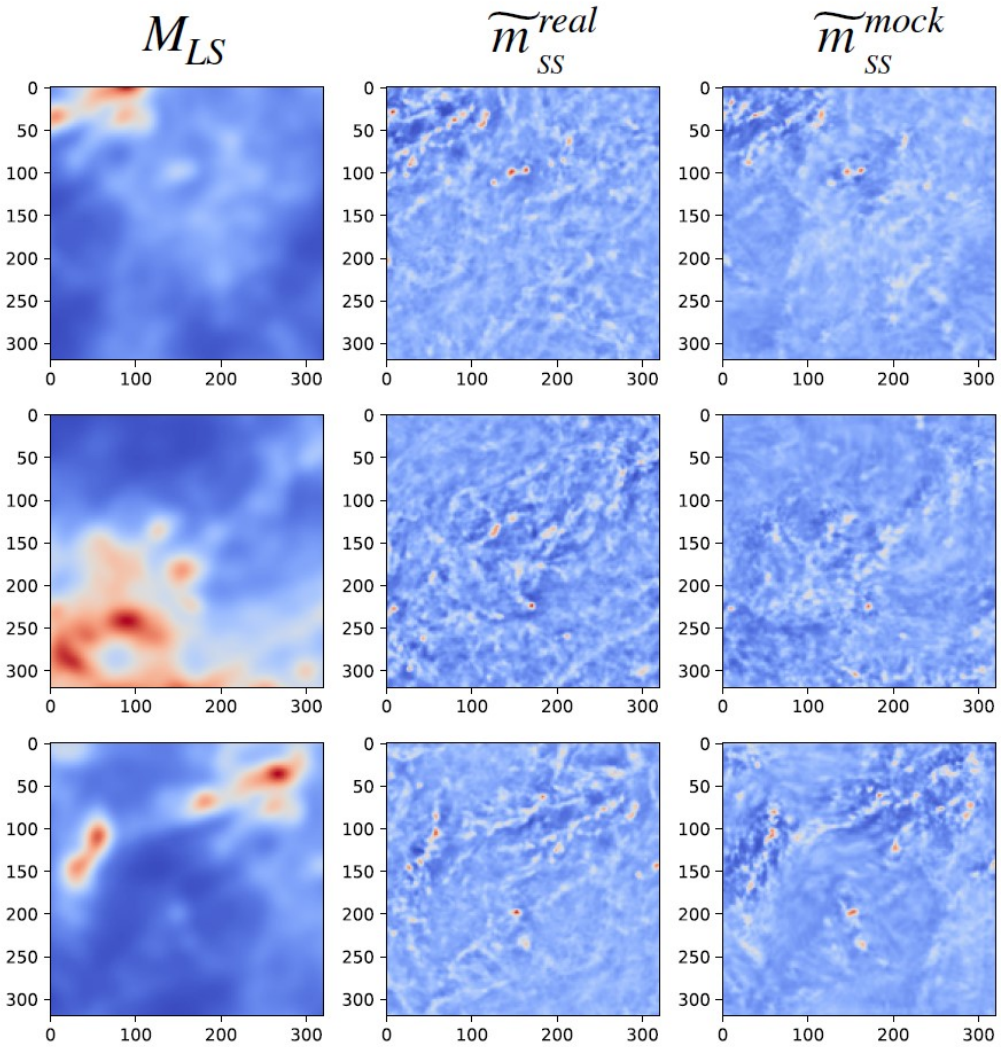


FIGURE 3.7: Patches from total intensity maps. Left column: large scale patches with features up to 80′ used as input to the DCGAN. Right column: small scale patches with features up to 12′ obtained as output from the DCGAN (here denoted by $\tilde{m}^{mock}_{SS}$ instead of $\tilde{m}^{gen}_{SS}$). Middle column: real small scale patches with features up to 12′. Both the real and the generated SS maps show trace of the LS structures, meaning that the network learnt to transpose this kind of structural information from input to output. Krachmalnicoff and Puglisi, 2021.

In the polarization case, the goal was not only to demonstrate that the DCGAN was able to create small scale patches with the same statistics of the real ones, but to actually create such patches in the areas of the sky where high resolution maps are not available.

So the full sky Q and U maps, smoothed at 80′, were divided in a set 174 slightly overlapping square patches. These LS maps were given as input to the trained DCGANs and corresponding SS maps were obtained for any area of the sky. Minkowski functionals were computed for the generated 174 SS patches and compared to Minkowski functionals of the 350 total intensity SS patches of the training set (fig. 3.8). As expected from the positive training results, the overlapping of the functionals is pretty good. The small scales structures were also compared to realizations of a Gaussian random field with a power spectra following a power law extrapolation of the Q/U large scale power spectra up to 12′ resolution. From fig. 3.8, it is clear that

the small scale structures generated with the DCGAN approach are different from the Gaussian small scale structures, making the DCGAN approach a valid method for generating non-Gaussian small scales. A more intuitive way to see that the structures generated by the NN are non-Gaussian is by looking at fig. 3.9 which compares for Q and U an example of LS patch ($M_{LS}^{Q/U}$), generated SS patch ($\tilde{M}_{SS}^{Q/U\,gen}$) and Gaussian SS patch ($\tilde{M}_{SS}^{Q/U\,gss}$).

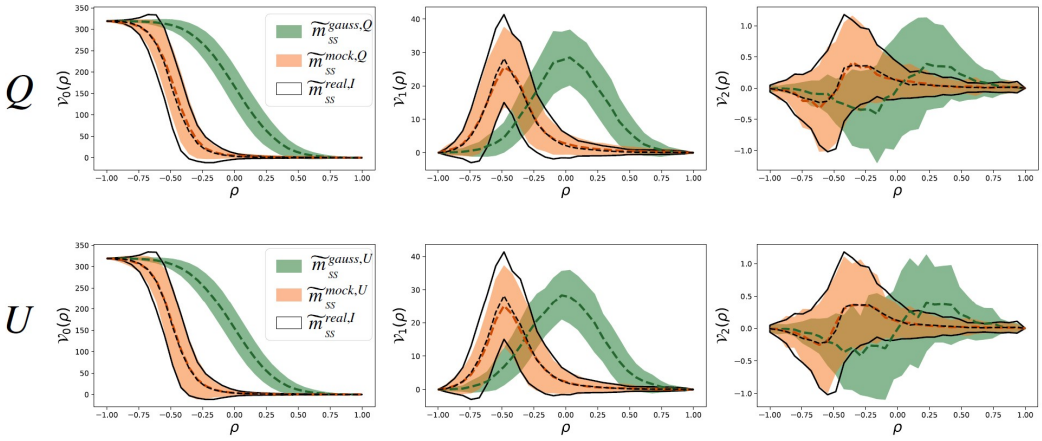


FIGURE 3.8: First three Minkowski functionals (here denoted as ν_n) as function of the threshold ρ, computed for both real SS patches of the total intensity maps (black), generated SS patches of the polarization map in Q and U (orange) and Gaussian SS patches (green). The diagrams show the average functional over the 350 patches from the training set, over the 174 generated patches, over the 174 Gaussian realization patches, plus and minus 1 STD. Krachmalnicoff and Puglisi, 2021.

3.3.3 Reconstructing Full Sky Maps

To rebuild the new full sky maps with resolution everywhere up to 12', from the set of 174 NN-generated patches at 12', a couple of steps must be performed.

The first one is to rescale the output SS patches from a $[-1, +1]$ range to the correct amplitude in physical units, which can be done by re-setting each patch's average and STD value to those of the Gaussian SS patches. Then, each SS patch must be re-multiplied by the corresponding LS patch, in order to obtain a map with both large and small scales: $M^{Q/U\,gen} = M_{LS}^{Q/U} \cdot \tilde{m}_{SS}^{Q/U\,gen}$. To further refine the rescaling, the maps $M^{Q/U\,gen}$ can be renormalized so that the power spectra of the small scales equals the power spectra of the Gaussian small scales (see sections 4.1 and 4.3.1).

The second one consists in merging effectively the overlapping edges of the patches, when putting them back together in the full sky map. The problem is that the small scales generated close to the border of each patch are different from those at the border of the neighbouring patches, so a simple averaging would show border effects. An apodization mask must be created, which ensures that patches are merged smoothly, without discontinuities (see chapter 6). The averaging of different small scales, however, tampers with the power spectra at high multipoles, which appears lower than before the process. The effect is more pronounced for patches at high latitudes, because the overlapping area between them is greater. The power loss, in any case, remains below 50% for all latitudes except for patches at the galactic poles which have a higher loss.

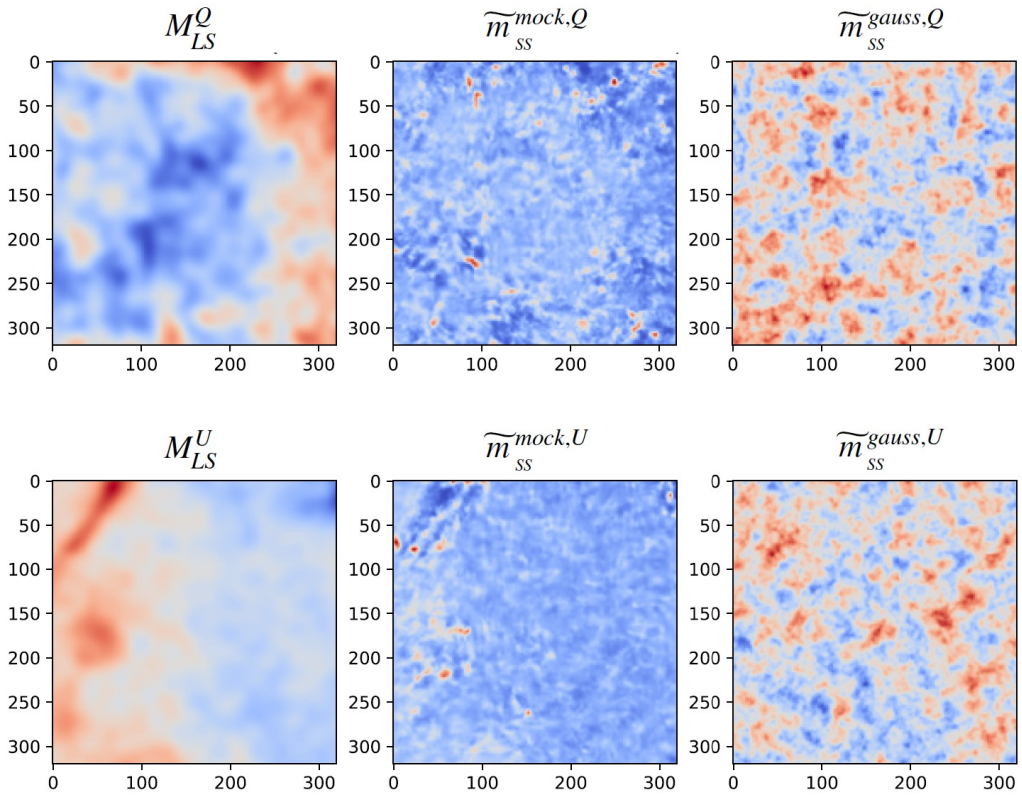


FIGURE 3.9: Patches from polarization Q and U maps. Left column: large scale patches with features up to 80′ used as input to the DCGAN. Middle column: small scale patches with features up to 12′ obtained as output from the DCGAN (here denoted by $\tilde{m}_{SS}^{mock}$ instead of $\tilde{m}_{SS}^{gen}$). Right column: Gaussian field realization small scale patches with features up to 12′. Krachmalnicoff and Puglisi, 2021.

3.4 Discussion and Developments

The DCGAN approach proved successful in increasing the resolution of CMB foreground thermal dust emission maps, by introducing small scale features with non-Gaussian statistics and the correct power spectra, in low resolutions maps that had only large scale structures. The parameters and the setup of the trained networks are publicly available together with the scripts for visualising the maps and computing Minkowski functionals and power spectra. The obtained Q and U full sky maps with 12′ DCGAN-generated features are also available for public access online[6]. They can be useful for foreground simulations or delensing algorithms, expecially for testing the data analysis pipeline in preparation of future CMB experiments (section 1.4).

In that context, there are some possibilities for further extensions of the ForSE package. One is to find a way to introduce stochasticity in the SS maps generated by the neural network. In that way, one could obtain maps with different realizations of the small scale structure distribution, and so have more synthetic realistic data for simulations. Having many different realizations of thermal dust maps would make it possible to determine the variance of the quantities computed from the maps, in particular the power spectra. An other possible development is to iterate the process of small scales creation to reach even smaller scales, assuming a fractal structure in the morphology of emission maps. Having foreground maps with resolution up

[6]Link to the maps: https://portal.nersc.gov/project/sobs/users/ForSE/.

to the arcminute scale is important to meet the expected resolution goals of future observations (e.g. Simons Observatory, LiteBIRD, Stage-4).

Elaborating on these two ideas, introducing stochasticity and iterating the approach to smaller scale, has been the main focus of my thesis. The next chapters will explain in detail how they were developed and what results were obtained, providing more in depth motivation about why this work is relevant in the introduction to each chapter.

Part II

Chapter 4

Implementing Stochasticity

The main difference between the original DCGAN architecture (Radford, Metz, and Chintala, 2016) and the architecture of the DCGAN used in ForSE is the kind of input received by the generator network. In the original set up, the input consisted of a noise vector, i.e. an array of random values, while in the ForSE network the input is a large scale patch, i.e. a 2D matrix containing structured information. Once training is over, for a specific input M_i, the neural network always returns the same corresponding output $\tilde{m}_i$. In other words, the DCGAN works in a deterministic way.

This might not be a desirable quality. The small scale structures introduced in the foreground maps with the NN approach are just a random realization of the distribution underlying the real small scale field, i.e. there is no point by point correspondence between NN-generated small scales and the real ones, but the two fields share the same statistics (up to the limits of the network training). So it would be useful to have a method for generating different realizations of the small scale field, so that one could obtain any desired number of maps with the same large scales but different small scales. Having many different maps would make it possible to estimate the errors on the statistical properties computed from them. For example, we could be able to compute error bars on the maps' power spectra.

A possible way to introduce stochasticity in the network would be to introduce a layer of random noise in the architecture of the GAN. The downside of this method is that it would increase the instability of the network and make training, in particular backpropagation, more difficult. An alternative is to apply a layer of noise directly on the input maps. There are different ways to do this. One is to train the network directly on a pure noise input, like the original DCGAN. The output in this case would have only small scale structures, with no trace or spatial correlation with the large scale ones. Another possibility is to add a layer of noise on top of the input maps. The amount of noise can be decided by tuning the SNR between the noise layer and the large scale maps. A different training has to be performed for each choice of SNR and the resulting generated small scales are expected to depend on it.

I followed both approaches: the training with pure noise only, from now on referred to as *noise training*, and the training with noisy large scale patches, from now on referred to as *LS + noise training*. The training done in ForSE, using pure large scale patches, will be referred to as *LS training*.

Programming Language, Packages and Computing Facilities The algorithm for building and training the ForSE network was written in Python[1], using the Keras[2] library and TensorFlow[3] backend. Keras contains high-level functions that implement

[1]Python documentation: https://www.python.org/.
[2]Keras documentation: https://keras.io/.
[3]TensorFlow documentation: https://www.tensorflow.org/

the building blocks of deep learning algorithms, such as the layers of a network, the cost function or the optimization algorithm. The low-level operations such as matrix operations or convolutions are implemented by the tensor manipulation framework Tensorflow. The operations required for training and using a neural network are computationally expensive, so that they cannot be carried out by personal computers or standard workstations. Therefore the training of the ForSE network and all other related computations were performed by on the supercomputer Cori at the National Energy Research Scientific Computing Center (NERSC)[4]. All the data manipulation and analysis that didn't require long computational times, as well as the code for data visualization, were written and organized in Jupyter Notebooks.

The additional code that I have written, uses the same language, libraries and facilities specified above.

4.1 DCGAN Training

As a preliminary step, before starting with the new kinds of trainings, I repeated the LS training for Q and U maps exactly as described in section 3.1.2. A first reason was to become acquainted with the procedures for running code on the NERSC facility, but it was also an important confirmation of the results found in Krachmalnicoff and Puglisi, 2021. Repeating the training also allowed me to have a LS trained network to which compare the results of the other trainings. Overall, I ran the following trainings (tab. 4.1):

- two LS trainings, for Q maps and U maps,
- one noise training,
- six LS + noise trainings for Q maps and U maps, with SNR = 1, 2, 10.

Note that for the noise training it was sufficient to train only one network for Q and U maps, because the output reference set consisted in both cases of the total intensity small scale patches and the input does not contain any information on the large scales.

In all cases, the training procedure remained exactly the one described in section 3.1.2, except for the input. In the LS training, the input maps were the large scale patches $M_{LS}^{Q/U}$ made from the masked area in Q and U maps and normalized between -1 and 1. In the noise training, the input maps were 2D matrices M_{noise} of the same size as the LS patches ($320px \times 320px$), where each pixel was a random value uniformly distributed between -1 and +1. In the LS + noise training the input maps were given by the sum of the normalized large scale patches plus the noise matrices divided by the SNR:

$$M_{LS}^{Q/U} + \frac{1}{SNR} \cdot M_{noise} \tag{4.1}$$

with the whole sum renormalized again between -1 and +1.

The LS trainings and the noise training were repeated for 100.000 iterations, while the LS + noise trainings were repeated for 150.000 iterations, because in the first trial trainings they were observed to converge in a longer time. Minkowski functionals were computed for the resulting small scale patches and compared to the Minkowski functionals computed over the total intensity small scale patches,

[4]NERSC web page: https://www.nersc.gov/.

training	input	output	SNR
LS	$M_{LS}^{Q/U}$	$\tilde{m}_{SS}^{T}$	∞
LS + noise	$\left(M_{LS}^{Q/U} + 0.1 \cdot M_{noise}\right)$	$\tilde{m}_{SS}^{T}$	10
LS + noise	$\left(M_{LS}^{Q/U} + 0.5 \cdot M_{noise}\right)$	$\tilde{m}_{SS}^{T}$	2
LS + noise	$\left(M_{LS}^{Q/U} + \cdot M_{noise}\right)$	$\tilde{m}_{SS}^{T}$	1
noise	M_{noise}	$\tilde{m}_{SS}^{T}$	0

TABLE 4.1: List of DCGAN trainings. In all cases the networks are trained to reproduce the total intensity small scale patches, from inputs of large scale patches with different levels of noise.

just as described in section 3.3. The overlapping of the areas included within ± 1 STD above or below the average of each functional was again used to determine the best iteration. Fig. 4.1 shows the overlap of Minkowski functionals as function of the number of iterations, for each functional and for each training. The parameters chosen for the trained networks were those for which the average Minkowski superposition (red line in the diagrams) reached its maximum.

The plots show that the best results were obtained when no noise was added to the input maps, as one could expect, with the maximum superposition decreasing as the SNR decreases. Anyway, overall good results were obtained also in the LS + noise and noise trainings except for the U maps training with SNR = 2, which can be taken as an example of a case when the training was unsuccessful. The training of a neural network, indeed, is not a deterministic process, stochasticity being introduced in the selection of training data, in the optimization method or in the dropout. So, even if a training is performed with the same parameters and setup, it might lead to different results each time it is done. In this case, the training was not repeated again due to a lack of computational time[5]. A sign that indicates that convergence in the training is reached, is a clear shift of the value around which the average Minkowski superposition oscillates. In the LS trainings this happens around 15.000-20.000 iterations, while in the LS + noise, SNR = 10, training this happens around 120.000-130.000, justifying the choice of a higher number of iterations. For the LS + noise trainings with lower SNRs it could be reasonable to think that convergence would be reached with a even higher number of iterations, but increasing the training iterations, and hence the computational time, was not feasible. Regarding the convergence of the noise training, a clear jump in average superposition happened right at the beginning of the training process, with no later improvement. So probably the Minkowski overlap percentage could not be further improved with more iterations. Another point to remark is that the introduction of noise in the NN, even if at the input level, rather than in the network's architecture, brings instability to the model. In the LS trainings, wide oscillations around the average superposition value are episodic, while in the other trainings they are continuous.

To express quantitatively the results of the DCGAN trainings, tab. 4.2 displays the average superposition, in percentage, between Minkowski functionals of the

[5] A LS + noise training with 150.000 iterations needs approximately 30 hours to complete on the NERSC supercomputer and time slots so big require a long waiting time before being granted.

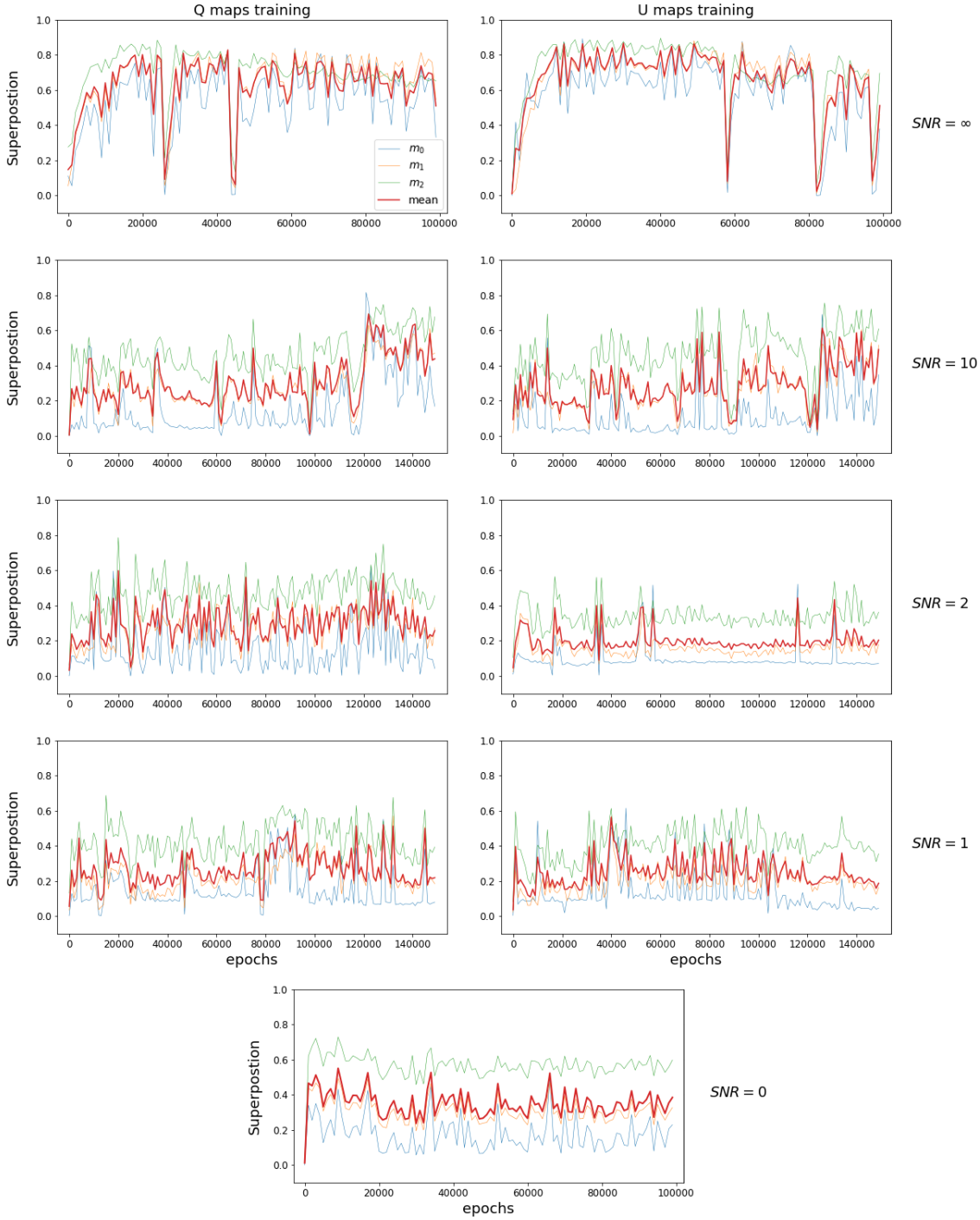


FIGURE 4.1: Superposition of Minkowski functionals, in percentage, as function of the number of iterations of the training algorithm. The thin lines in blue, orange and green line refer, in order, to the three Minkowski Functionals, while the red thick line refers to the average among the functionals.

generated SS patches in Q and U and of the real SS patches in total intensity, for each training as well as the number of iterations corresponding to the trained network.

training	m_0	m_1	m_2	**average**	iteration
LS (Q)	83%	83%	83%	83%	43.000
LS (U)	87%	86%	88%	87%	30.000
LS (Q) + noise, SNR = 10	75%	63%	69%	69%	122.000
LS (U) + noise, SNR = 10	69%	52%	63%	61%	126.000
LS (Q) + noise, SNR = 2	48%	53%	79%	60%	20.000
LS (U) + noise, SNR = 2	52%	37%	44%	44%	116.000
LS (Q) + noise, SNR = 1	58%	47%	58%	54%	92.000
LS (U) + noise, SNR = 1	50%	57%	61%	56%	40.000
noise	43%	49%	73%	55%	9.000

TABLE 4.2: Superposition, in percentage, of the Minkowski Functionals of the generated SS patches in Q and U and the real SS patches in total intensity, for each training.

The diagrams of the Minkowski functionals to which the percentages in tab. 4.2 refer to are shown in fig. 4.2, fig. 4.3 and fig. 4.4. These diagrams compare Minkowski Functionals, computed on different sets of small scale (12′ resolution) patches. The dashed line represents the average functional over each set of patches, while the colored area represents 1 STD above or below the average functional. The functionals in black/white are the ones of the real total intensity small scale patches, that were the target set of images. The blue plots are the functionals of the Gaussian field small scale patches. The orange, the green and the red plots, represent respectively the functionals of the SS patches generated by the trained DCGAN in the cases of LS training, LS + noise training, noise training. The plots of Minkowski function-

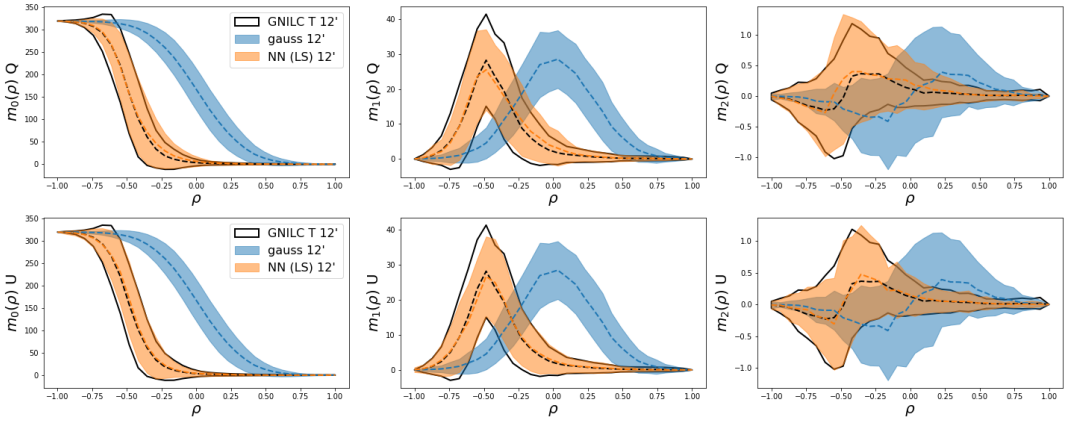


FIGURE 4.2: Minkowski functionals of the SS patches generated by the LS trained network (orange), compared to those of the real total intensity SS patches (black) and the Gaussian field realization of SS patches (blue). The dashed lines mark the average functional, while the contours of the colored area represent $\pm$ 1 STD above or below the average. First line is for Q maps, second line for U maps.

als in the LS training case show clearly the successful results of the training, since the generated patches yield an almost perfect overlap with the patches in the training set for both Q and U. In the LS + noise training and in the noise training cases the overlap becomes less exact, in accordance with the percentages in tab. 4.2. Still,

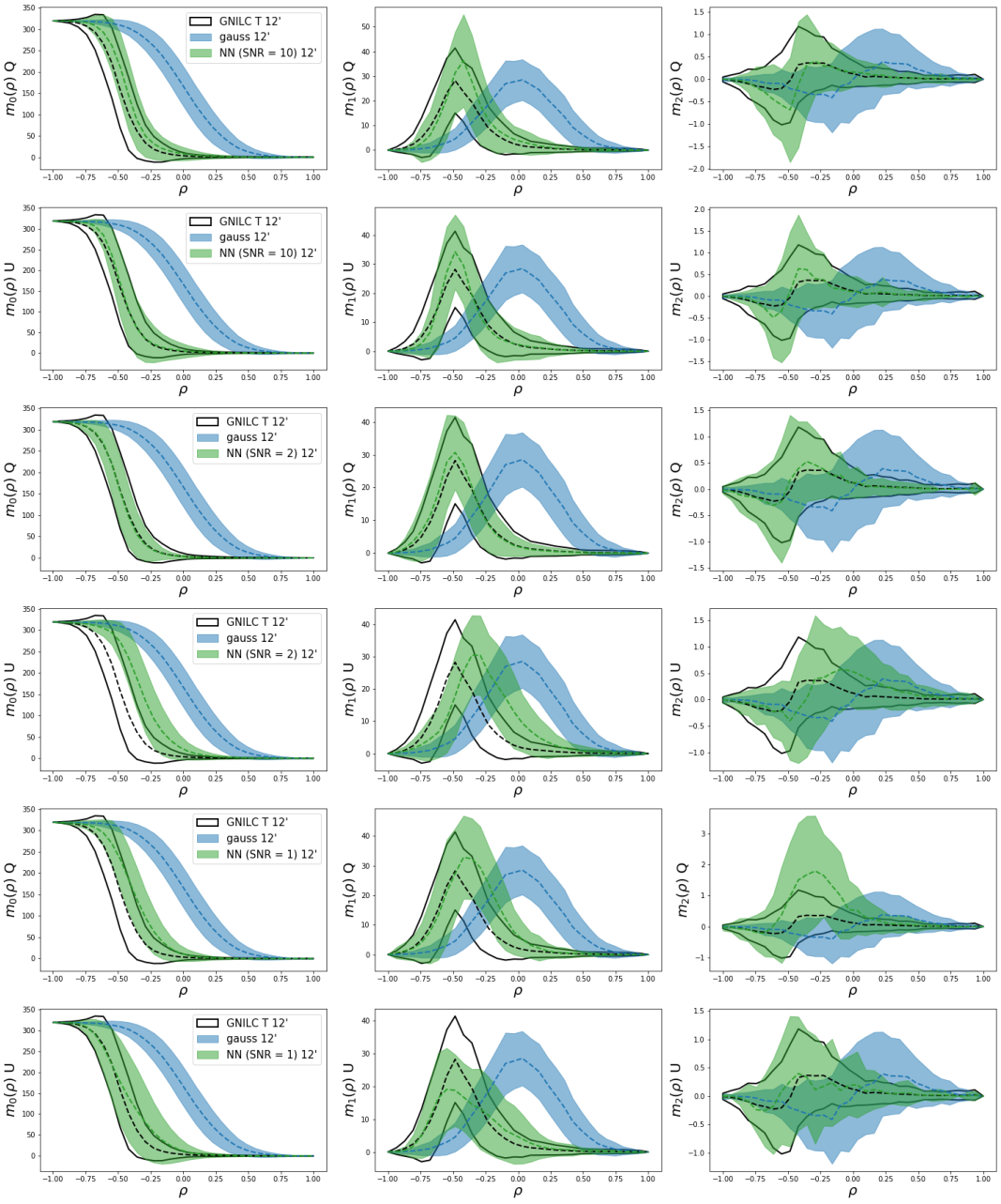


FIGURE 4.3: Minkowski functionals of the SS patches generated by the LS + noise trained network (green), compared to those of the real total intensity SS patches (black) and the Gaussian field realization of SS patches (blue). The dashed lines mark the average functional, while the contours of the colored area represent $\pm$ 1 STD above or below the average. The first two lines are the SNR = 10 case (first for Q, second for U), the second two for the SNR = 2 case, the last two for the SNR = 1 case.

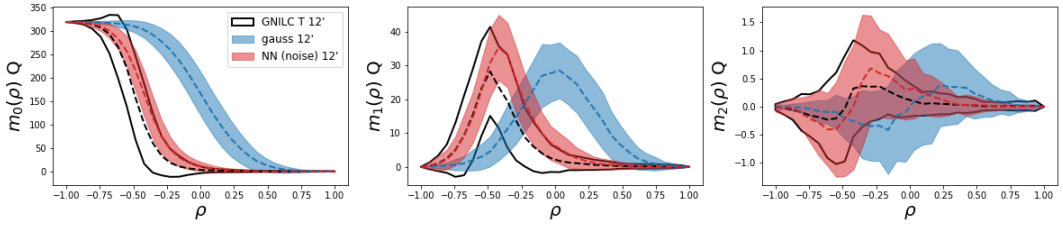


FIGURE 4.4: Minkowski functionals of the SS patches generated by the noise trained network (red), compared to those of the real total intensity SS patches (black) and the Gaussian field realization of SS patches (blue). The dashed lines mark the average functional, while the contours of the colored area represent $\pm$ 1 STD above or below the average.

in every training, the functionals of the generated patches are much more similar to those of the total intensity patches than are those of the Gaussian patches.

A more intuitive way to see that the DCGAN-generated small scales, for Q and U, are quite different from the Gaussian small scales is by looking at a comparison between example patches. Fig. 4.5 presents two examples of output patches for all trainings in both Q and U. The first line in the picture shows the LS patch used as input to the trained DCGAN. From the second to the fifth line, there are, in order, the SS output patches from the LS training, the LS + noise training (SNR from 10 to 1) and the noise training. The last line shows the Gaussian patches. The small scales in each column are the NN outputs corresponding to the input in the first line, apart for the noise training patches. The first two columns refer to the same portion of the sky (the first column with the Q map, the second with the U map), and the same holds for the third and fourth column. The large scale maps in the first line and the Gaussian maps in the last one are normalized between -1 and +1, while the small scale maps are shown exactly as returned by the network.

For all trainings with LS or LS + noise inputs, the generated small scales retain trace of the large ones, in variable measure according to the value of the SNR. This characteristic was present also in the real SS maps in total intensity, so we can assume that it is a realistic feature of SS maps. This means that the neural network learnt correctly the mapping from large to small scales. However, there might be cases where it is preferable to have small scale patches with no trace of the large ones. Then, it might be better to use the noise trained DCGAN instead, which gives away one of the realistic features that the NN is able to imitate, in favour of a more general applicability. An example of such a situation will be described in the next chapter when discussing the iteration of the DCGAN approach.

A problem when working with DCGANs is that the output is returned rescaled in an interval determined by the image set of the activation function and therefore requires a renormalization to the correct scale. In the case of DCGAN training for polarization maps, the only reference for the correct amplitude of the field in the SS patches are the Gaussian SS patches, which to first order have the correct amplitude, even though having a different statistics. The rescaling was performed in the following way:

1. Each of the 174 SS patches $\tilde{m}_{SS}^{gen}$ resulting as output from one of the trained DCGANs was paired with a Gaussian field SS patch $\tilde{m}_{SS}^{gauss}$.

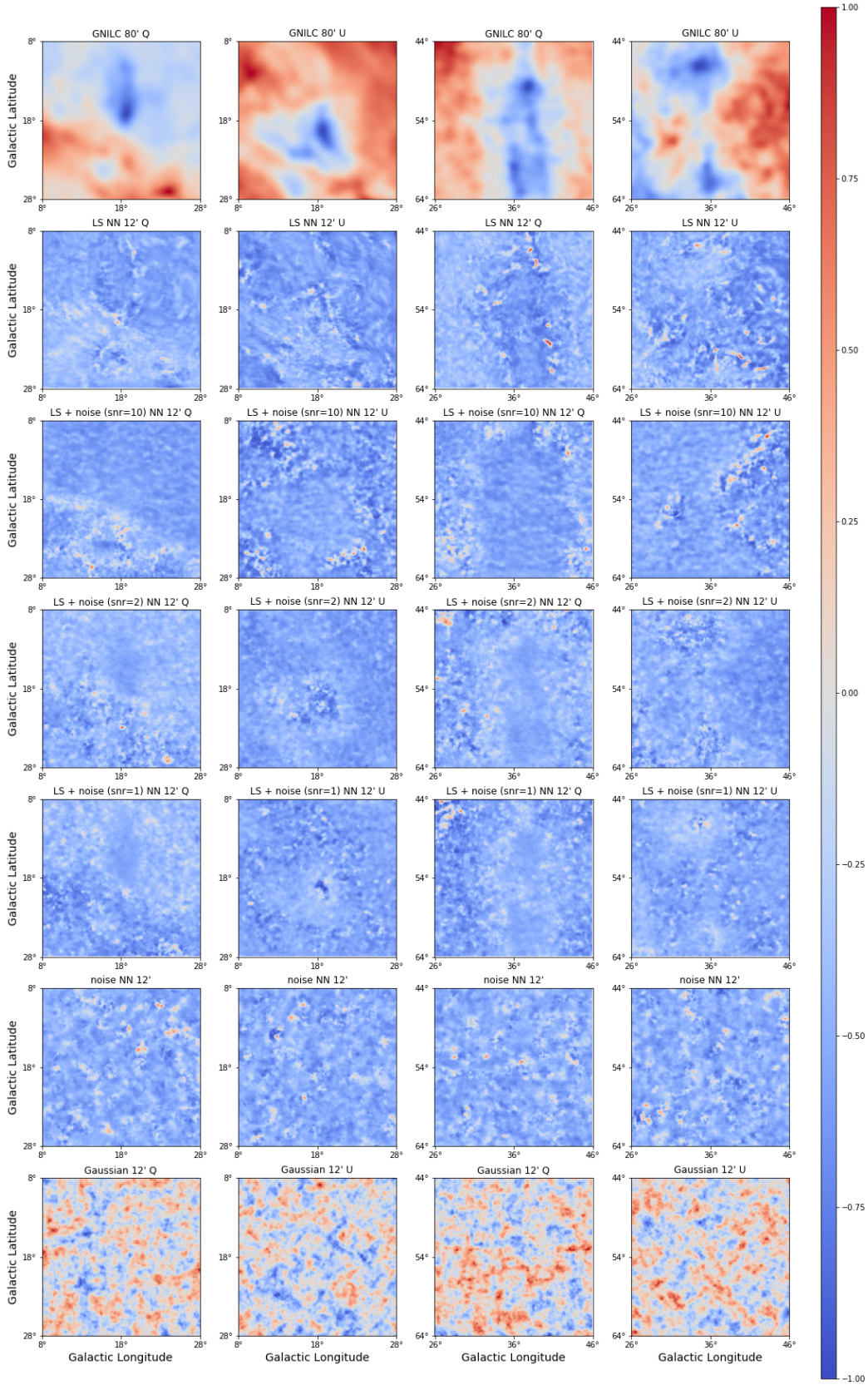


FIGURE 4.5: Comparison of small scale patches generated by the DCGANs with different trainings and the Gaussian small scale patches. The first line shows the LS patch used as input, from the second to the fifth line, there are, in order, the SS output patches for the LS training, the LS + noise training (SNR from 10 to 1) and the noise training. The last line shows the Gaussian patches. The small scales in each column are the NN outputs corresponding to the input in the first line, apart for the noise training patches.

2. The output patch was rescaled so that the average and the STD of the pixels' values was the same as the average and STD of the Gaussian patch:

$$\tilde{m}_{SS}^{gen} \quad \longrightarrow \quad \left(\tilde{m}_{SS}^{gen} - \left\langle \tilde{m}_{SS}^{gen} \right\rangle\right) \cdot \frac{STD\left(\tilde{m}_{SS}^{gauss}\right)}{STD\left(\tilde{m}_{SS}^{gen}\right)} + \left\langle \tilde{m}_{SS}^{gauss} \right\rangle \tag{4.2}$$

4.2 Characterization of the Output Maps

4.2.1 Distribution of the STD

Besides looking at Minkowski functionals , an other way to examine the statistics of the small scale field generated by the DCGAN, is to compute the standard deviation of the distribution of values for each pixel. In particular, from the 174 generated SS patches, after rescaling, it is possible to compute a pixelwise STD, i.e. compute at every pixel coordinates (x, y), $x, y = 1, 2, \ldots, 320$ the standard deviation across the 174 patches of the values of that pixel. The result is a standard deviation map, that reveals patterns, variability and average value of the STD of the field. We are interested in the distribution of the STD, rather than the average value or the STD itself, because both of them were fixed by the rescaling. In fig. 4.6 one finds the STD maps for all trainings as well as for the Gaussian field.

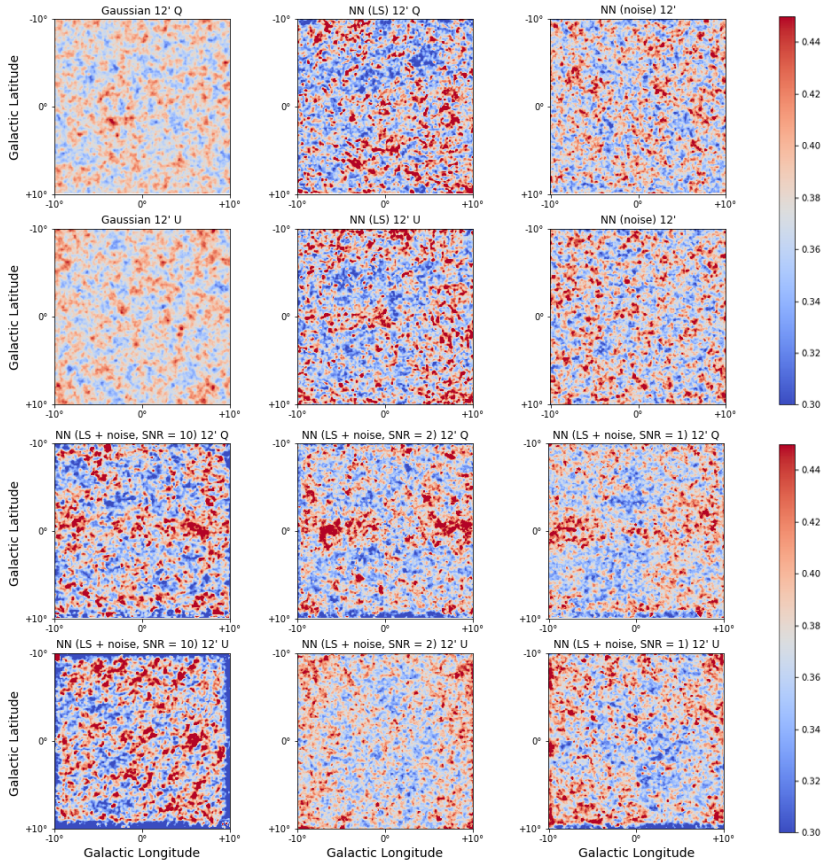


FIGURE 4.6: Pixelwise standard deviation maps for each set of output SS patches. The value represented by a pixel in these maps is the STD of that pixel's value in the set of 174 output patches. The first and third lines refer to Q maps, the second and fourth to U maps.

training	STD's average value	STD's standard deviation
LS (Q)	0.375	0.045
LS (U)	0.377	0.043
LS (Q) + noise, SNR = 10	0.375	0.044
LS (U) + noise, SNR = 10	0.372	0.059
LS (Q) + noise, SNR = 2	0.376	0.040
LS (U) + noise, SNR = 2	0.379	0.029
LS (Q) + noise, SNR = 1	0.377	0.032
LS (U) + noise, SNR = 1	0.378	0.037
noise	0.378	0.034
Gaussian field	0.378	0.020

TABLE 4.3: Standard Deviation of Pixel Values, averaged over all pixels. The average and standard deviation of the pixelwise STD are reported for all trainings and for the Gaussian field case.

A first thing to notice is the presence of border effects in all the cases where LS patches were used in the input. This is an undesired spurious effect that may be attributed to the decomposition procedure from full sky maps to square patches. Furthermore, the STD maps of the LS and LS + noise trainings reach higher and lower values than those of the noise training (with the exception of the SNR = 2, U training, which anyway was not successful, as previously discussed.). The wider range of STD values may be attributed to the presence of LS imprints in the SS patches generated from those trainings. In any case, for all the trainings, the STD ranged in a wider interval compared to the Gaussian patches. The standard deviation computed over all pixels of the pixelwise standard deviation is indeed lower for the Gaussian SS patches and is generally higher for the SS patches resulting from trainings with a high SNR (tab. 4.3).

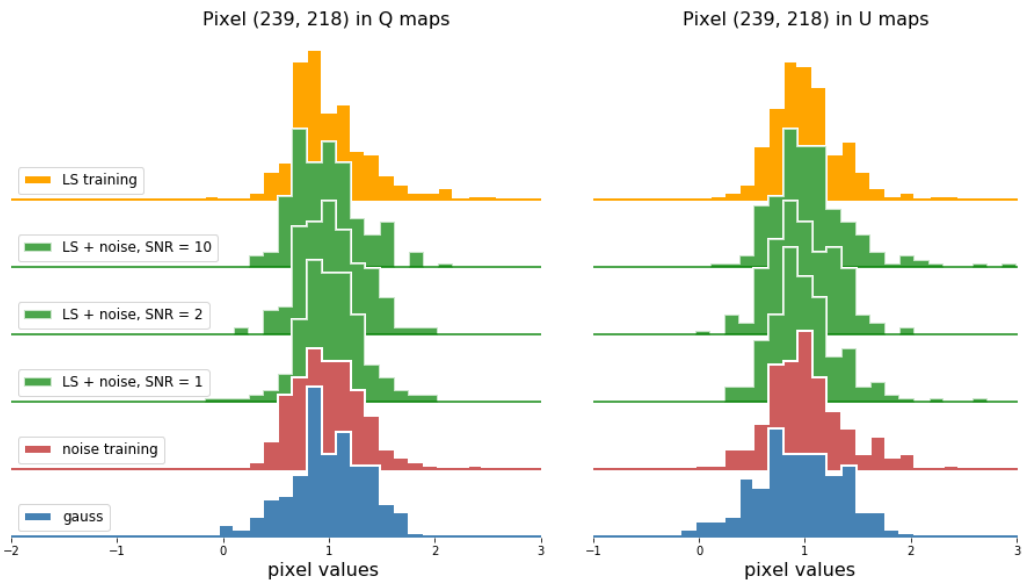


FIGURE 4.7: Distribution of the values of a randomly selected pixel in the output SS patches. The figure compares the results from different trainings and from the Gaussian field case. First column refers to Q maps, second column to U maps.

In fig. 4.7, there is an example of the distribution of values for a random pixel over the 174 SS patches. The figure compares the distribution of the same pixel value for each different training, in both Q an U[6]. The standard deviations of these distributions are the values appearing in the maps in 4.6.

4.2.2 Variability of Output Maps as Function of the Noise

When analysing the trainings with noisy inputs, an aspect to look at is how much the output patches change if the specific noise realization that was used in the input is changed for another one. For the noise training, this is just the variability of the SS output patches, since the input consists of noise only, while for the LS + noise trainings it means to compare the output patches corresponding to inputs with the same LS patch, but different noise realizations. It is important that the variability is sufficient to justify the introduction of noise in the input: if the trained network collapsed all inputs to the same output the introduction of noise would have failed its purpose.

In rows two to four, fig. 4.8 shows three examples of SS output patches for each trained network (only for the Q maps, U maps would show similar results). The first line displays the input LS patch given to the network in each case. The last line presents a standard deviation map, in the fashion of those in fig. 4.6, where the value of each pixel is the STD of the pixel's value in 100 output SS maps generated by the same LS input with 100 different noise realizations. It appears that, all the output patches present the same trace of large scales, if they were present in the input, but completely different small scales, as was hoped. The standard deviation of the pixel values is higher in the presence of a large scale structure, because the pixel values undergo higher oscillations in the proximity of a structure. This kind of inhomogeneity fades as the SNR of the input patch decreases, until disappearing in the noise training case. Fig. 4.9 reports the Minkowski functionals, as usual with average $\pm$ 1 STD, corresponding to the shown patches. The plots show more variability for functionals with a higher index, which is consistent with the fact that the higher the index the more complex the shape of the functional. In any case, the functionals appear as they would if the corresponding images were samples from the same distribution.

4.3 Power Spectra

The last analysis performed on the generated SS patches was the computation of their power spectra (defined in section 1.2). A generic LS patch (patch n. 98, with sky longitude and latitude $(18°, 18°)$) was picked as an example and the power spectrum was computed on the corresponding outputs returned by every trained network. The power spectra were computed using the NaMaster code, implemented in Python by the pymaster[7] library. This code uses the pseudo-C_ℓ formalism (Alonso et al., 2019) to calculate the angular power spectrum of functions defined only on a limited portion of the sphere, and in particular in the flat-sky limit, which is the case relevant to the $20° \times 20°$ square patches. Since the spectra are calculated only on a limited portion of the sky, their estimates for ℓs corresponding to scales greater

[6]Note that for the cases of noise training and Gaussian field the distributions in the Q and U columns are equivalent.

[7]Pymaster documentation: https://namaster.readthedocs.io/en/latest/#.

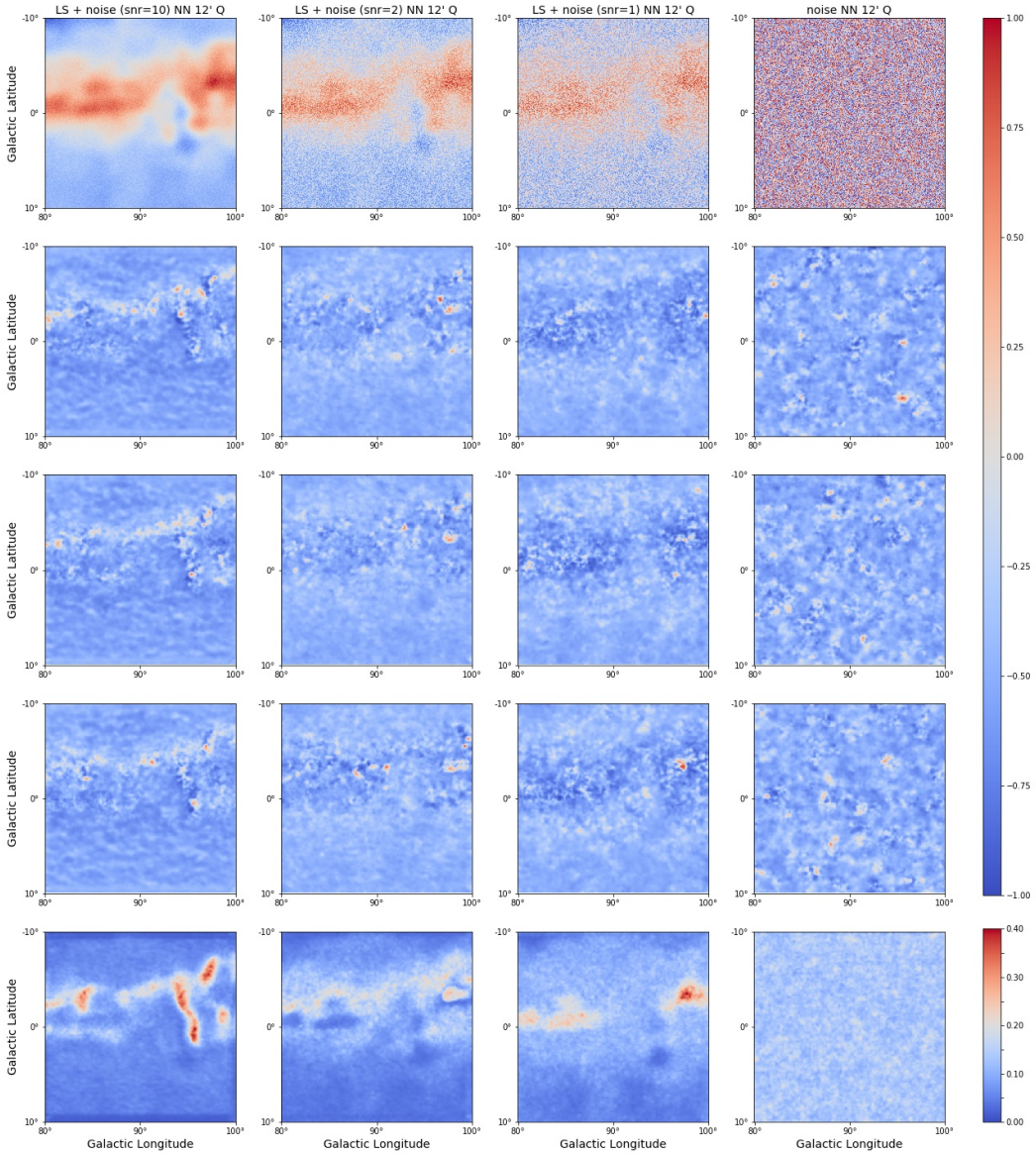


FIGURE 4.8: Three examples of SS output patches for the DCGAN trained with noisy inputs. The first line displays in the LS input patches normalized between -1 and +1. The second, third and fourth lines contain the output SS patches before any rescaling. In the last line there are STD maps, computed over 100 different realizations of small scale output maps above. The SNR decreases from left to right.

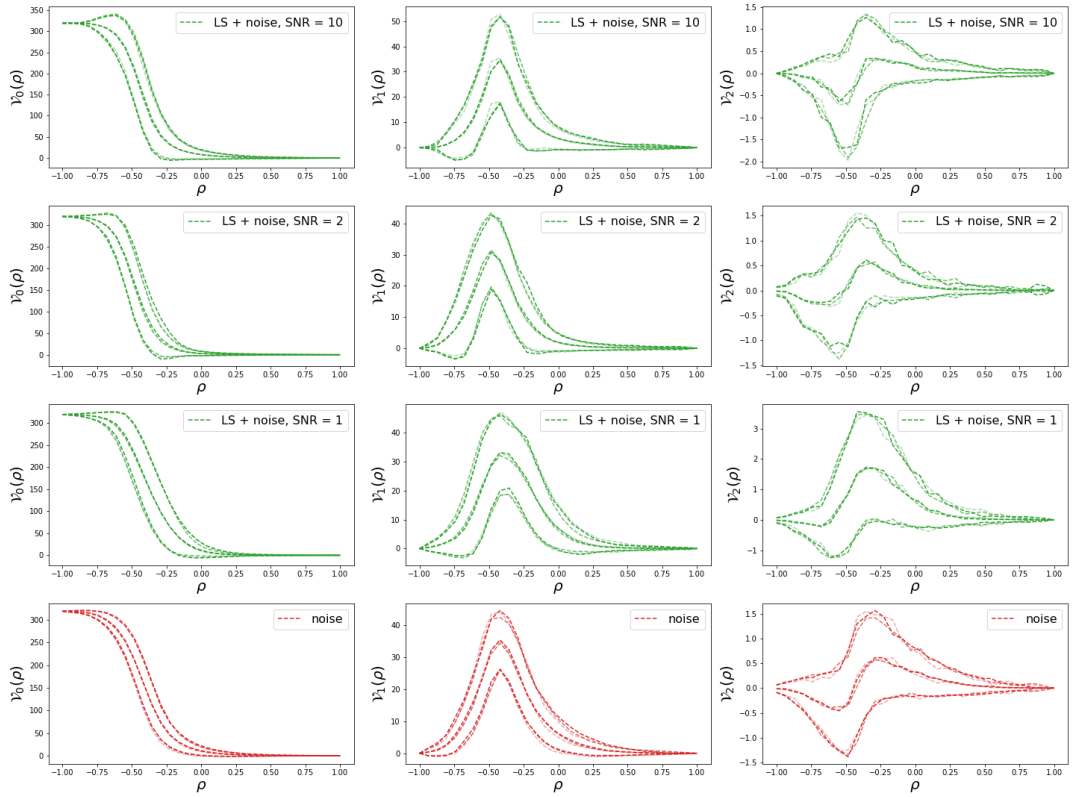


FIGURE 4.9: Plot of Minkowski functionals that show the variability of the output of the DCGANs trained with noise in the input. The green plots refer to the LS + noise training (from high to low SNR), the red plots are for the noise training.

than the side of the patches are not accurate. Indeed the power contained in low frequency modes might be incorrectly distributed among different modes, or between E and B modes. To limit this spurious mode mixing effects, an apodization window is applied to each map when computing the spectrum. The window must be continuous, smooth and go from values of 0 on the border to values of 1 at the center of the image. The one used here (fig. 4.10) follows the function:

$$W_{apo}(x) = \begin{cases} \frac{1}{2}\left(1 - \cos\left(\pi\sqrt{\frac{1-\cos\eta}{1-\cos\eta_0}}\right)\right) & \text{if } x < 1 \\ 1 & \text{otherwise} \end{cases} \tag{4.3}$$

where η is the angular separation between a point and the border of the window and $\eta_0 = 2^\circ$ a constant. The power spectra C_ℓ^{EE} and C_ℓ^{BB} were computed in the E-B decomposition, for values of ℓ in $[20, 1020]$, binned in intervals of $\Delta\ell = 40$. Fig. 4.11 shows the angular power spectra for SS output patches, corresponding to the LS input patch 98, after renormalization to the Gaussian mean and STD. The blue line represents the Gaussian field small scales, the yellow line the small scales generated from the LS training, the green line the ones generated from the LS + noise training and the red line those from the noise training. To simplify the reading of the plot, of all the LS + noise training cases, only the one with SNR = 1 is shown. Anyways, the trainings with different SNRs yielded results very similar to the SNR = 1 case, so from now on only the latter will be considered in the analysis. For the trainings with noise in the input as well as for the Gaussian patches, the power spectra shown in the diagram is the average spectra over 100 different realizations of the SS patches. For the LS + noise training, this meant that 100 different SS patches were generated

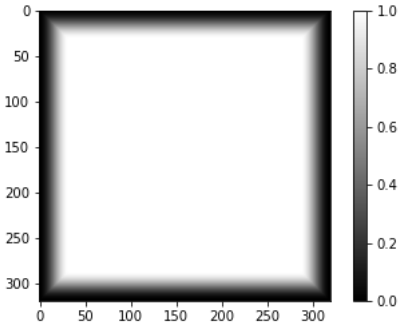


FIGURE 4.10: Apodization window used for the computation of the angular power spectra of flat square patches. Krachmalnicoff and Puglisi, 2021

starting from an input having the same LS + patches, but different noise realizations. For the noise training and for the Gaussian patches, this only meant generating 100 SS patches from 100 different noise inputs or picking 100 different realizations of the Gaussian field. The dotted lines represent the average power spectra, while the colored bands cover the area enclosed between 1 STD above or below the average. For the LS training case, the dotted line represents just the power spectra of that particular SS patch.

The interesting point here is to observe the trend of the STDs and to check how close the averaged spectra are to the spectra of the LS training patch. A first good results is the fact that all the spectra of the generated patches are very similar to each other, even overlapping in various points. This means that even if the DCGAN was trained using different inputs, the network is able to produce very similar output distributions. Instead, compared to the Gaussian field patches, the generated patches have more power in the small scales. Looking more closely, the spectra of the LS training falls almost everywhere within 1 STD from the LS + noise spectra and within 1 STD from the spectra of the noise training for roughly half of the points. This means that the LS + noise trainings produce SS patches with a spectrum that has a underlying distribution compatible with the distribution of the spectra of the patches from the LS training. Therefore we can use it effectively to estimate the STD of the power spectra of the DCGAN-generated small scales. The STDs of the power spectra are shown in fig. 4.12. For the LS + noise trainings, the STD is lower in the

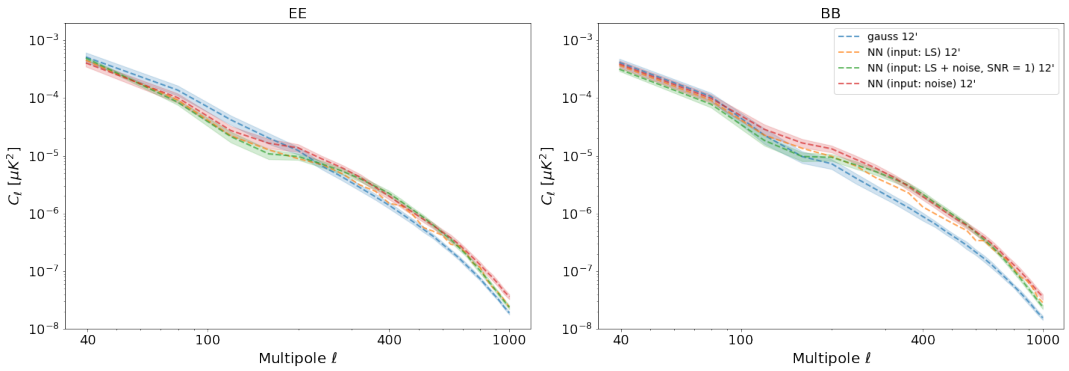


FIGURE 4.11: Angular power spectra of E and B modes for one small scale patch. The blue line is for the Gaussian field, the yellow one for the LS training, the green one for LS + noise training and the red one for the noise training. In all cases except the LS training, the plotted line shows the average spectra and STD of 100 different realizations.

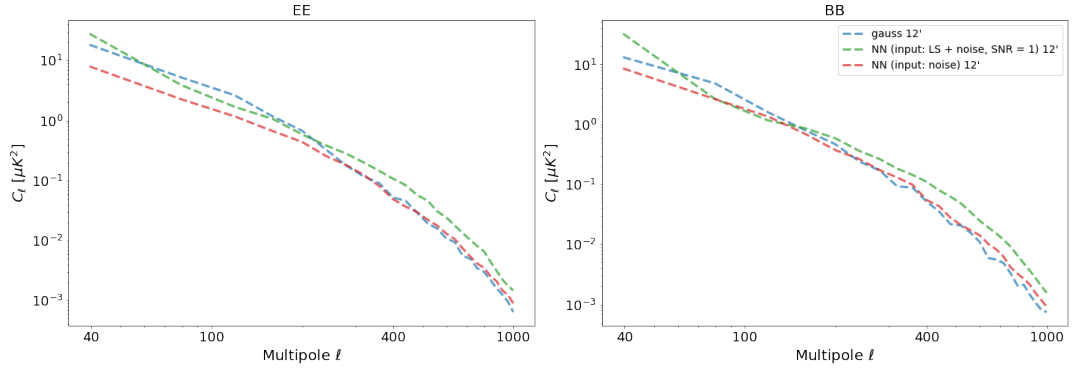


FIGURE 4.12: Standard deviation of the angular power spectra of E and B modes for one small scale patch. The blue line is for the Gaussian field, the green one for LS + noise training and the red one for the noise training. For all cases, the STD was computed over 100 different realizations.

large scales, which is reasonable since the patches generated from those trainings bare trace of the input large scales. Both the noise and the LS + noise trainings have a higher STD than the Gaussian field in the small cases, which means that the neural network generates small scales with a wider variability.

4.3.1 Refine the rescaling

A refinement of the rescaling of the SS patches can be done using the power spectra. The point of this second renormalization is to obtain generated SS patches with power spectra as similar possible to that of the Gaussian field SS, which is the correct power spectra for the emission maps. This is achieved by the rescaling:

$$\tilde{m}_{SS}^{Q/U} \longrightarrow \left(\tilde{m}_{SS}^{Q/U} - \left\langle \tilde{m}_{SS}^{Q/U} \right\rangle\right) \cdot \sqrt{\frac{\left\langle C_\ell^{gauss} \right\rangle_{\ell>160}}{\left\langle C_\ell^{Q/U} \right\rangle_{\ell>160}}} + \left\langle \tilde{m}_{SS}^{gauss} \right\rangle \tag{4.4}$$

where the factor $\sqrt{\left\langle C_\ell^{gauss} \right\rangle_{\ell>160}} / \sqrt{\left\langle C_\ell^{Q/U} \right\rangle_{\ell>160}}$ is the square root of the ratio between the power spectra of the Gaussian patches and that of the NN generated patches, averaged over all ℓs greater than 160 (corresponding to $\sim 1°$), to account only for the small scales. Note that the power spectra here are not computed in the E-B decomposition, but singularly on the Q and U fields.

4.3.2 Reintroducing Large Scales

After the creation of small scale patches and their renormalization, the next step is to insert them into the large scale maps. This is done by multiplying each SS patch $\tilde{m}_{SS}$ with the corresponding LS patch M_{LS}, in the correct units, to get a map with both large scales (80 arcminutes) and small scales (12 arcminutes):

$$M^{Q/U} = M_{LS}^{Q/U} \cdot \tilde{m}_{SS}^{Q/U} \tag{4.5}$$

Example results are shown in fig. 4.13. The first two lines display results for the example patch n.98 (sky coordinates: $(18°, 18°)$), the second line for patch n.139 (sky coordinates: $(36°, 54°)$). The first and the third row refer to Q maps, the second and the fourth row to U maps. In the first column there are large scale input maps at 80'.

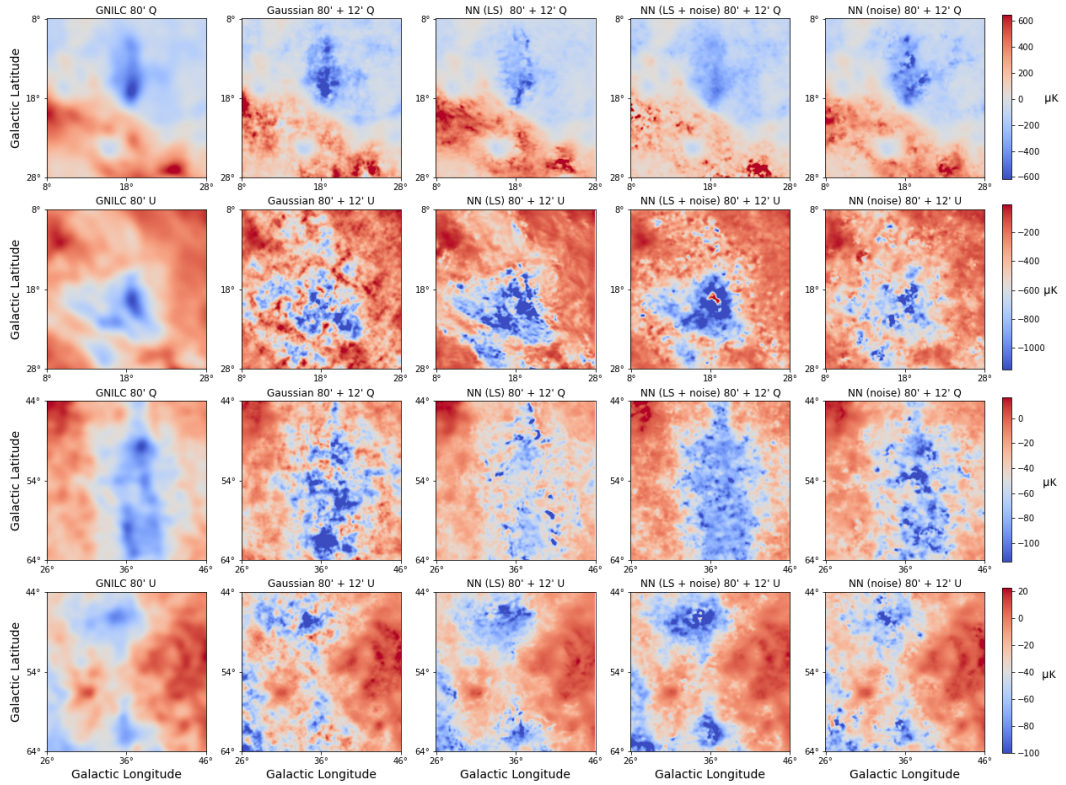


FIGURE 4.13: Example of LS + SS Patches. The first two lines show results for the example patch n.98, sky coordinates: $(18°, 18°)$, the second line for patch n.139, sky coordinates: $36°, 54°$.

From the second to the last column there are large scale plus small scale maps, up to 12′, for all different cases of small scales. These patches $M^{Q/U}$ with LS + SS are ready to be used to reconstruct full sky maps up to a resolution of 12 arcminutes.

The angular power spectra of the LS + SS patches can be computed in the same way as was done for the SS patches. Fig. 4.14, shows the spectra of the example patch (n.98). Again, the blue line is for the Gaussian field, the yellow one for the LS training, the green one for LS + noise training and the red one for the noise training. In all cases except the LS training, the plotted lines show the average spectra of 100

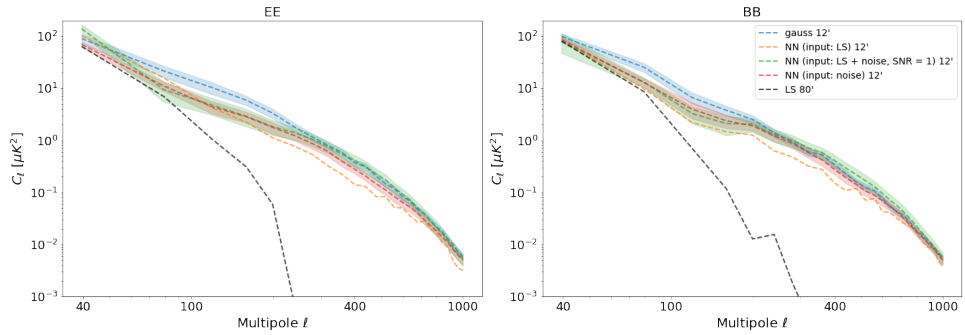


FIGURE 4.14: Angular power spectra of E and B modes for one large scale patch. The blue line is for the Gaussian field, the yellow one for the LS training, the green one for LS + noise training, the red one for the noise training and the black one for the original large scale maps. In all cases except the LS training and the LS maps, the plotted lines show the average spectra and STD of 100 different realizations.

different realizations and the colored bands are the area between 1 STD above or below the average. As before, only the case SNR = 1, is shown for the LS + noise training, because the results for the other cases were very similar.

All the small scale spectra are very similar to each other, especially for $\ell > 200$, were they almost coincide. So, this means that, for any training, the DCGAN is able to enhance the resolution of large scale maps by adding non-Gaussian small scales, that, after an appropriate rescaling, have the expected power spectrum (i.e. that of the Gaussian field realization).

Chapter 5

Iterating to 3 arcminutes

The DCGAN approach for modeling the small scale features of the thermal dust emission maps, allows one to generate features up a scale of 12 arcminutes. This resolution is set by the resolution of the total intensity SS patches in the training set, which in turn was determined by the maximum resolution present in the total intensity full sky maps for a sufficiently large portion of the sky. In the spherical harmonics multipole expansion (see section 1.2), the 1D angular scale $\theta = 12'$ corresponds to the multipole $\ell = \pi/\theta = 900$. Post-Planck CMB experiments currently under preparation, such as the Simons Observatory, LiteBIRD and Stage-4 (see section 1.4), aim at measuring the polarized power spectrum at multipoles corresponding to the arcminute scales.

It would be helpful to have models of the galactic emission that provide maps with resolution up to a few arcminutes and a possible way to achieve this is by repeating the DCGAN approach. The idea of iterating the creation of smaller scales relies on the fact that the neural network is not sensitive to the physical scale of the smallest features in an image (i.e. the resolution), but only to the ratio ($R = L/r$) between the angular size of the side of the patch (L) and the resolution (r). So by resizing correctly the 12′-scale patches, one can use them again, as an input to the neural network. The trained NNs were able to upscale the maps' resolution from 80′ to 12′, which means a decrease of the resolved scale by a factor of $f = 0.15$. Applying it again to 12′-scale maps would mean being able to generate 1.8′-scale maps. Actually, as it will be explained in this chapter, the iteration of the resolution upscaling requires a few technical steps, during which part of the resolution is lost, so that, in the end, the final resolution that can be reached is 3 arcminutes.

Applying the DCGAN to the emission maps for a second time, is possible if we assume that the structures at 12′ and the ones at 3′ have similar characteristics. Saying that the structures at different scales share the same statistics means that the maps of thermal dust emission have a fractal structure. This is reasonable because the angular power spectrum of the emission, which is the Fourier transform of the two-point correlation function, follows the same power law down to the arcminute scale, so one can assume that the same happens for the higher n-point correlation functions which define the statistics of the field. In any case this is the best guess that we are able to make with the available data.

There are different possibilities for iterating the SS generating process. First of all, there are five available networks (one for the LS training, three for the LS + noise trainings and one for the noise training). However, to use the networks with large scales in the input, we need that the fractal assumption holds, not only between scales at 12′ and scales at 3′, but also between the large scales at 80′ and the scale of the new patches that will be used as input to the NN. This is important because the networks are specifically trained to recognize the exact features of the patches at

80′ and translate them into the SS features at 12′. So if a different image is given as input to the NN, the latter will not be able to return a correct output. Therefore, the simplest way to introduce even smaller scales in the emission maps is by using the noise trained DCGAN which doesn't require this additional assumption.

This chapter will describe the steps required to iterate the process of small scales creation and analyze the results obtained.

5.1 Procedure for iterating the DCGAN Approach

Up to now, to pass from low resolution to high resolution maps, the DCGAN took input patches with:

side length	$L_{in} = 20^\circ = 1200'$
resolution	$r_{in} = 80'$
pixels per side	$n_{in} = 320\,px$

and the output patches were returned with:

side length	$L_{out} = 20^\circ = 1200'$
resolution	$r_{out} = 12'$
pixels per side	$n_{out} = 320\,px$

For the input maps, the ratio between side length and resolution is:

$$R_{in} = L_{in}/r_{in} = 15$$

while for the output maps it is:

$$R_{out} = L_{out}/r_{out} = 100$$

To use again the output patches as inputs for the NN, they must be divided into smaller sub-patches, with a side length such that the new ratio $R = L/r$ is equal to R_{in}, because that is an information that the network learned to recognize. The side length L of the sub-patches would be:

$$L = R_{in} \cdot r_{out} = 180'$$

which is approximately 6.67 time smaller than the side L_{out}. However, dividing the side of the patch by a non-integer number would make the use of DCGANs very complicated, because it would result in the use of non-square patches. So the most convenient way to make sub-patches is to divide the side by a number, smaller than 6.67, that is a divisor of the number of pixels per side of the output map, i.e. $n = 320px = 2^6 \cdot 5\,px$. The patches were divided into $4 \times 4 = 16$ sub-patches, which allows one to generate 3′ scales (as explained below). An other choice could have been to divide them into 5×5 sub-patches. With this narrower subdivision we would have reached a resolution of 2.6′, but the 13% increase in resolution would come at a cost of an increase of computational time of least 50% and so the option was ruled out for the time being. Repeating the work with more and smaller sub-patches, to create even smaller scales, remains a possibility for future work.

Sub-patches with side length $L = 5^\circ$, resolution $r = 12'$ and pixels per side $n = 80\,px$, are not ready yet to be fed to the DCGAN. First of all, the pixels per side of

the new input patches must be $n = 320\,px$, because the DCGAN architecture is built for an input of that size, so the sub-patches need a $(4\times)$ upscale. Secondly, the ratio R must be equal to R_{in}, so the sub-patches must be smoothed down to a resolution $r = L/R_{in} = 5^\circ/15 = 20'$. Since we are using the noise trained DCGAN, the sub-patches that are going to be used as the new large scale patches are constructed from the 174 Q and U patches $M^{Q/U}_{(noise)}$, that have features up to 12′, generated by the noise trained DCGAN (for an example, see the last column of fig. 4.13).

The sequence of operations required to iterate the production of small scales can be divided into two main steps: creating the new input patches at 20′ from the $M^{Q/U}_{(noise)}$ patches and adding to them the output patches at 3′. Both steps require various intermediate operations in which the side length, resolution or pixels per side are changed. So in the following, for the sake of clarity, a set of patches will be denoted by $M(X', Y^\circ, Zpx)$, where the triplet of numbers (X, Y, Z) indicates respectively the resolution (in arcminutes), the side length (in degrees) and the pixels per side (in px). For example, the patches $M^{Q/U}_{(noise)}$ will be denoted by $M^{Q/U}(12', 20^\circ, 320px)$. Let's now see all these steps in detail:

1. **Upsampling.** The DCGAN's architecture is set so that the number of pixels per side for input/output patches is $n = 320$. So, if the patches $M^{Q/U}(12', 20^\circ, 320px)$ are going to be divided into sub-patches with side length 4 times smaller, the number of pixels per side must be multiplied by 4. Note that even if we are increasing the number of pixels we are not increasing the resolution of the map, which means we don't want to introduce new information. So the upscaling is performed with no interpolation method, by just repeating the pixel values in the corresponding 4×4 pixels of the upsampled patch. After this step we obtain upsampled maps $M^{Q/U}(12', 20^\circ, 1280px)$.

2. **Smoothing.** Since the patches $M^{Q/U}(12', 20^\circ, 1280px)$ are going to be divided into sub-patches with side length $L = 5^\circ$, in order to have the same resolution-to-side-length ratio $R = 15$ of the original LS patches at 80′, their resolution must be smoothed to $r = 20'$. The chosen smoothing method is Gaussian blur, i.e. performing a discrete convolution between the image and a 2D Gaussian. The method is implemented using the `ndimage.gaussian_filter`[1] function from the Python library Scipy, which takes as input the image matrix and the size of the Gaussian sigma in pixel units. To smooth a map with the effective beam's FWHM = 12′ into a map with FWHM = 20′, the map must be convoluted with a (Gaussian) beam of FWHM = $\sqrt{(20')^2 - (12')^2} = 16'$ (see appendix A). For a Gaussian, a FWHM= 16′ corresponds to $\sigma = 6.79'$, which in our case amounts to $\sigma = 7.24\,px$ on the pixelized image. After smoothing, we obtain patches $M^{Q/U}(20', 20^\circ, 1280px)$. An example of the effect of the smoothing filter is shown in fig. 5.1.

3. **Division in sub-patches.** The patches $M^{Q/U}(20', 20^\circ, 1280px)$ must now be divided into sub-patches with a side length 4 times smaller. The original patches could be fully covered by a tessellation of $4 \times 4 = 16$ sub-patches. However, after the introduction of the 3′ scales, border effects would appear at the edge of each sub-patch. To avoid this, sub-patches could be created so that each of them overlaps for half of its side length with the sub-patches on the left,

[1] Scipy Gaussian filter documentation: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter.html.

right, top and bottom. In this way, each portion of the original patch is covered by 4 different sub-patches, except for the portions close to the side, which are covered only by 2 sub-patches, and the portions at the angles, which are covered by only 1 sub-patch. With this method, after the introduction of the 3′ scales, the sub-patches can be apodized back together. With this kind of overlap, seven sub-patches per side are needed, for a total of $7 \times 7 = 49$ sub-patches. So for each of the 174 patches $M^{Q/U}(20', 20^\circ, 1280px)$, 49 sub-patches $M^{Q/U}(20', 5^\circ, 320px)$ are created.

4. **Generating and adding 3′ scale patches.** At this point, for both Q and U maps, there are a total of $174 \times 49 = 8526$ sub-patches $M^{Q/U}(20', 5^\circ, 320px)$ that take up the role of the $M_{LS}^{Q/U}$ patches at 80′ that were used as input to the DCGAN in the previous chapter. The noise trained DCGAN generates output patches starting only from a noise matrix, so it is sufficient to feed 8526 different noise realizations to the DCGAN to get the same number of 3 arcminutes output patches. These patches take up the role of the $\tilde{m}_{SS}$ patches at 12′ generated in the previous chapter. To add the 3′ features in the 20′ sub-patches, each sub-patch $M_{LS(20')}^{Q/U}$ is multiplied by one output patch $\tilde{m}_{SS(3')}^{Q/U}$. Note that, before the multiplication, the patches with 3′ features have been renormalized with the two-step rescaling described in sections 4.1 and 4.3.1. Resulting from this step are maps $M^{Q/U}(3', 5^\circ, 320px)$, containing both 20′ and 3′ features and which are the equivalent of the maps $M^{Q/U}$ with 80′ + 12′ features obtained in the previous chapter. An example of the addition of 3′ scales to the 20′ scale is shown in fig. 5.2 for Q maps (U maps are similar), where the effect of DCGAN generated small scales is compared to that of Gaussian small scales.

5. **Recomposing the full patches.** The last step consists in recomposing together the sub-patches with side length of 5° into patches with 20° side length. Blending the sub-patches together is done by applying to them different apodization masks, depending on the position taken by the sub-patch on the full patch, e.g. center patch, side patch, angle patch (all the kinds of masks are shown in fig. 5.5a). It is important that the apodization masks are smooth, to avoid border effects, and that they sum to 1 when they overlap with each other in any point of the recomposed patch. The profile of the apodization masks follows the function:

$$W_{apo}(x) = 1 - \cos^2(x) \tag{5.1}$$

on every side where the corresponding patch overlaps with another patch, with x being the distance from the border of the mask. After the recomposition we obtain the final patches with enhanced resolution $M^{Q/U}(3', 20^\circ, 1280px)$. The reason for reconstructing $20^\circ \times 20^\circ$ patches, rather than keeping the $5^\circ \times 5^\circ$ ones, is that in this way, for building full sky maps with 3′ features, the same code for the 12′ full sky maps can be used (see section 6.1).

The above points are briefly listed in tab. 5.1 with the characteristics of the patches resulting from each step.

5.1.1 Checks on the Procedure

To make sure that the above procedure was producing maps with the correct resolution, without altering the statistics of the structures, a couple of tests were performed.

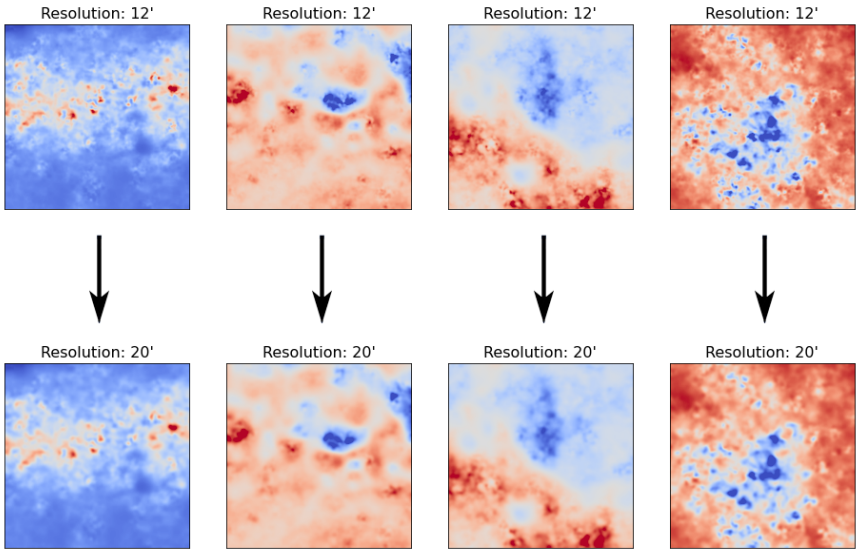


FIGURE 5.1: Four examples of the effect of the Gaussian filter, applied on Q and U maps. The filter performs a convolution between the image and a Gaussian beam with a chosen σ in pixel units. The first line displays maps at 12′ resolution, before the smoothing, while the second line shows the same maps at 20′, after the smoothing.

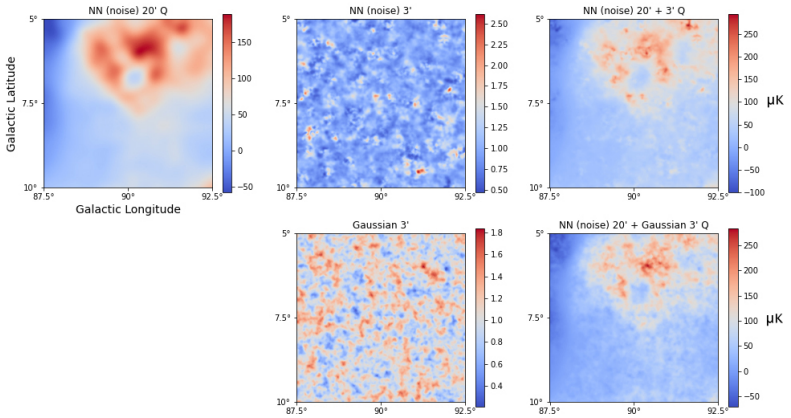


FIGURE 5.2: Example, for Q maps, of the addition of small scales (3′) to large scales (20′) maps. The LS patches in the left column are multiplied for the SS patches in the middle column to produce the LS + SS patches in the right column. The figure shows a comparison between the introduction of DCGAN generated small scales and Gaussian field small scales.

	resolution r	side length L	pixels per side n
initial patches	12′	20°	320 px
upsampling	12′	20°	1280 px
smoothing	20′	20°	1280 px
division in sub-patches	20′	5°	320 px
adding 3′scale patches	3′	5°	320 px
recompose full patches	3′	20°	1280 px

TABLE 5.1: Steps for iterating the DCGAN approach to reach the 3 arcminutes resolution. The last three columns refer to the characteristics of the Q-U patches after each step.

First, the resolution of an example patch was checked before and after the smoothing, by computing the angular power spectra. Fig. 5.3 shows E and B power spectra for patch n. 98 (sky coordinates $(18°, 18°)$) before smoothing (blue line) and after (red line). The smoothed one, of course, drops faster for smaller scales (higher mul-

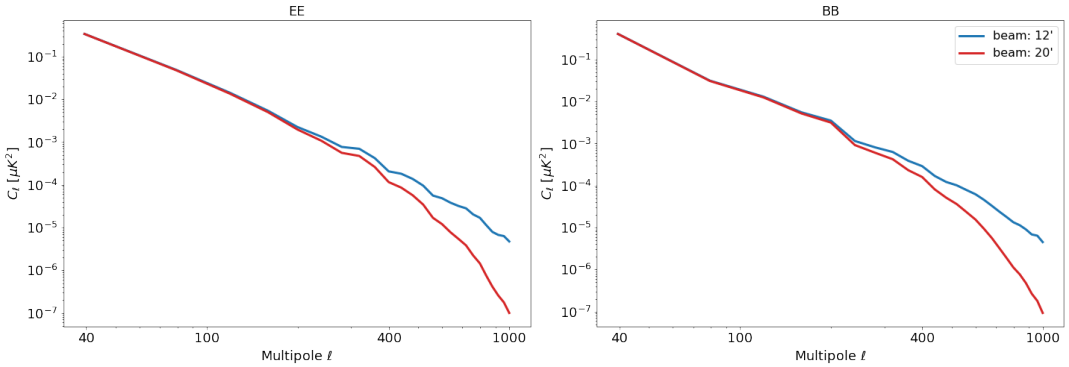


FIGURE 5.3: Angular power spectra of an example patch before (blue) and after (red) the smoothing that reduced its resolution from 12′ to 20′.

tipoles). Then the spectra were computed again, but this time correcting for the fact that the images were convoluted with a Gaussian beam with FWHM equal to 12′ for the pre-smoothing map and 20′ for the post-smoothing map[2]. The results are shown in fig. 5.4. The two power spectra overlap exactly, meaning that the maps' resolutions were actually 12′ and 20′.

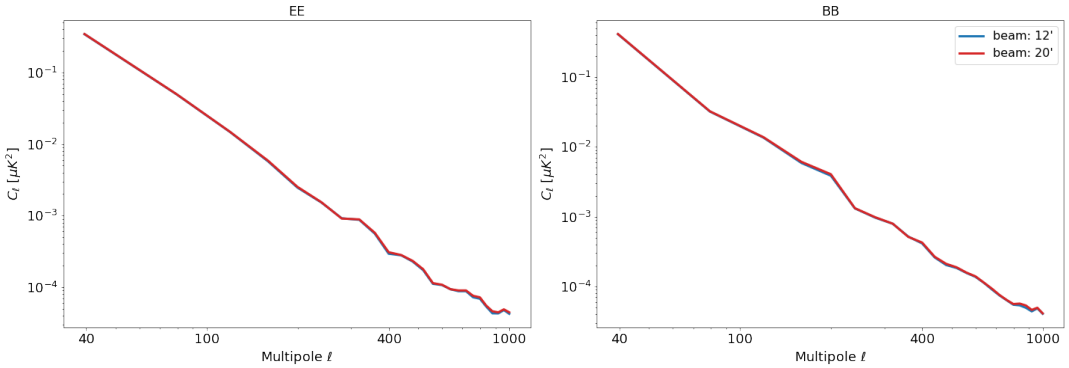
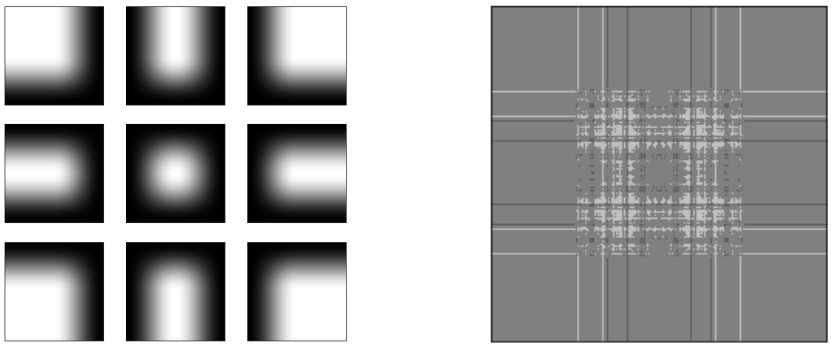


FIGURE 5.4: Angular power spectra, corrected for beams of 12′ and 20′, of an example patch before (blue) and after (red) the smoothing that reduced its resolution from 12′ to 20′.

Another test was done to ensure that the apodization masks were actually weighting correctly the patches to which they were applied. Nine of them were summed together, arranged 3×3, with an overlap of half of the side of a mask (see fig. 5.5). The sum of the masks should be equal to 1 everywhere. The result is presented in fig. 5.5b. The image shows some patterns, but the minimum and maximum values deviate from the unit at the 16^{th} decimal cifre, which is a completely neglectable error.

[2]Accounting for an instrumental beam is an option already present in the NaMaster functions.



(A) . (B) .

FIGURE 5.5: Apodization masks for patch reconstruction. Figure (A) shows all the different kinds of masks: the ones for sub-patches in the center, on the side and on an angle of the full patch. The color white corresponds to a value of 1, black to 0. In figure (B), the patches on the left are summed together, in the same position, with an overlap of half of the side of a mask.

5.2 Results of the Iteration

The maps resulting from the iteration of the DCGAN approach for creating small scales are shown in fig. 5.6. The figure presents a comparison, for Q and U maps, of an example patch (n. 83, sky coordinates $(108°, 0°)$ at different resolutions. In the first columns there are the initial large scale maps at 80′, in the third one there are the maps with resolution up to 12′ built with small scales generated by the noise trained network, the second column shows the smoothing with a Gaussian beam of the maps in the third column down to 20′, while in the last column there are the maps with features up to 3′ generated by the noise trained network iterated a second time. A zoomed detail of the maps in fig. 5.6 is shown in fig. 5.7. To focus the attention on the small scales it is convenient to plot the maps at 20′, 12′ and 3′, divided by the large scale maps at 80′. This is done in fig. 5.8, where the last three columns show, in order, the 20′ maps, the 12′ maps and the 3′ maps. The first column shows, as a comparison, the Gaussian field maps at 3′, also divided by the 80′ maps.

An issue in these last maps is the presence of points with values completely out of scale, noticeable in the plots as darker lines. They appear because of the division by the large scales patches, which in some points have values very close to 0. The presence of these points and their arrangement in lines along the 0 value level set of the 80′ maps, spoil the computation of Minkowski functionals, so they cannot be used to analyse the statistics of the patches. In any case the different statistic between 3′ Gaussian field maps and 3′ small scales generated by the NN, is intuitively noticeable from the plotted maps. Besides, the small scale patches at 3′ are generated in the exact same way as the small scales generated at 12′ so they yield the same non-Gaussian statistics.

It is also interesting to compare the angular power spectrum of the maps at different scales. In fig. 5.9, one can find the spectra in E and B for the example patch, at resolutions of 80′, 20′, 12′ and 3′. The spectra of all the maps appear to follow the same power law function, up to the scale corresponding to their effective beam. This is a good result, because it means that, also when applied the second time to reach scales of 3′, the trained DCGAN is able to generate features of the thermal dust emission maps with non-Gaussianities and with the correct power spectra.

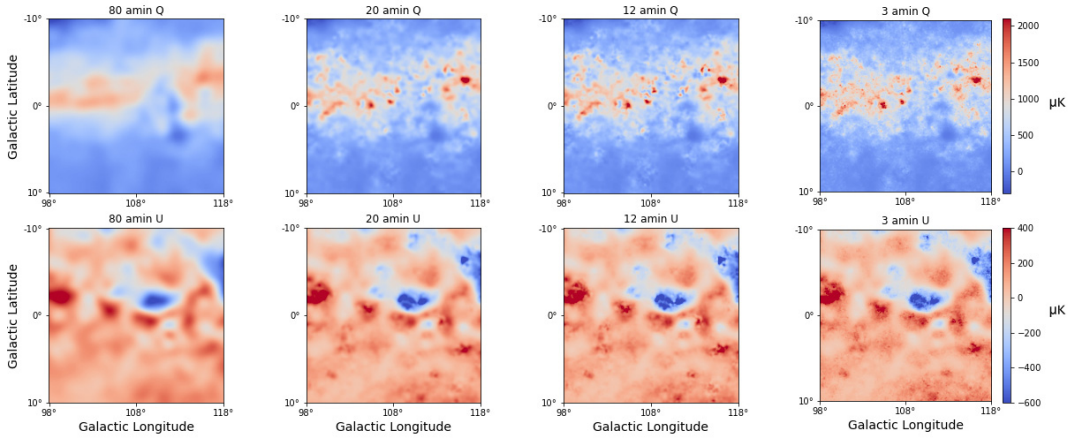


FIGURE 5.6: Comparison between Q-U maps at 80′, 20′, 12′ and 3′. Example patch n.83, sky coordinates $(108°, 0°)$. In the first column, the original GNILC maps smoothed at 80′. The small scales in the 12′ maps (third column) and in the 3′ maps (fourth column) are generated by the noise trained DCGAN. The maps at 20′ (second column) are created by a Gaussian smoothing of the maps at 12′.

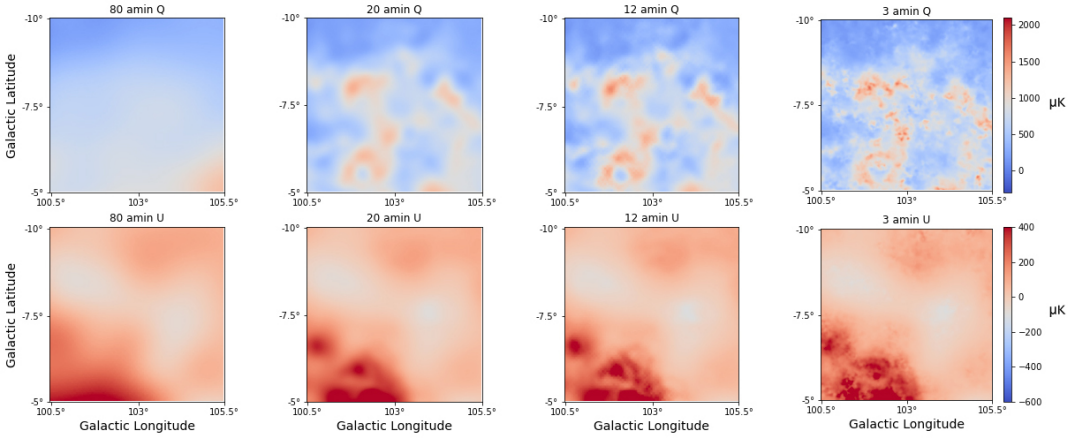


FIGURE 5.7: Detail from maps in fig. 5.6 (4 times zoom).

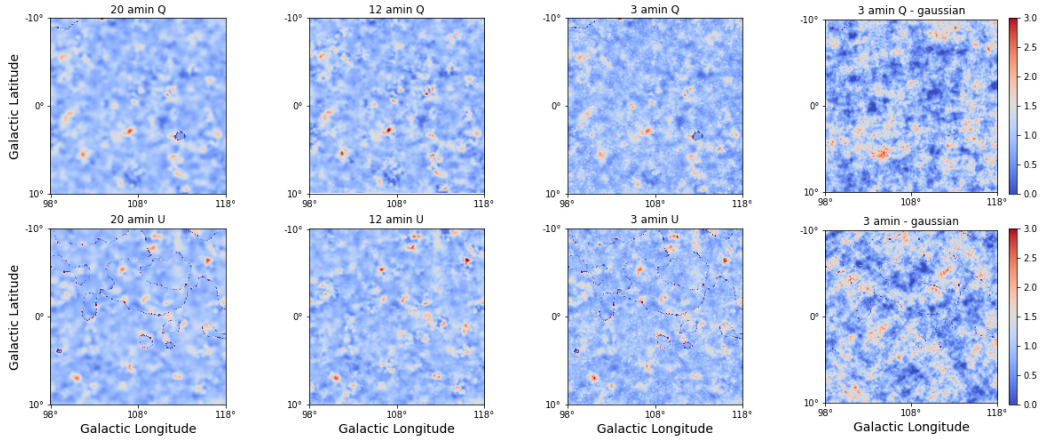


FIGURE 5.8: Comparison between Q-U maps at 20′, 12′, 3′ and 3′ Gaussian maps, all divided by the corresponding maps at 80′. Example patch n.83, sky coordinates $(108°, 0°)$.

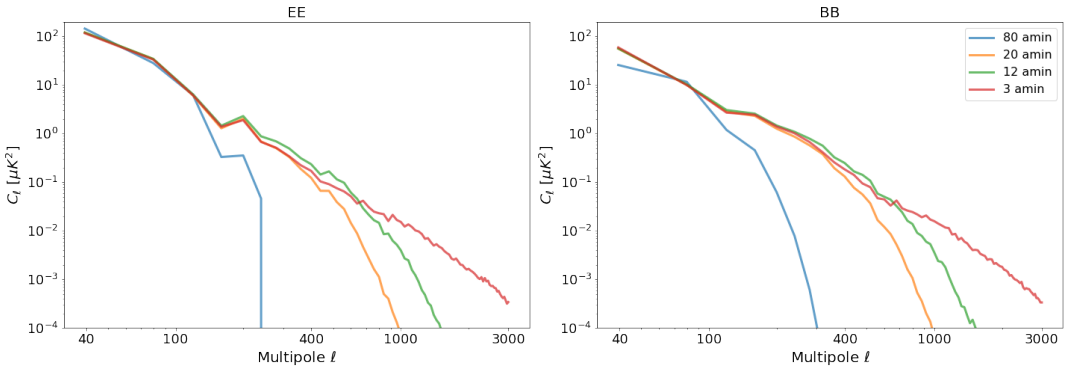


FIGURE 5.9: Angular power spectra in E and B for Q and U maps at 80′, 20′, 12′ and 3′ (example patch).

Chapter 6

Building Full Sky Maps

The two main goals of the thesis were to create full sky polarization maps of the thermal dust emission with features at 12′ generated by a neural network that had a stochastic variability and to iterate the method to generate also scales at 3′. So after creating and analysing the $20^\circ \times 20^\circ$ patches at 12′ and 3′, the last step is to compose them back together to form full sky maps.

The issue with the full sky recomposition regards the overlapping areas between the patches, which are of the scale of 2°. The problem is that the neural network generates small scales using only the information contained in one patch, with no knowledge of the content of the neighboring patches. Therefore the small scales generated close to the border of a patch don't have any continuity with those of the adjacent patches. So, before composing the patches together, an apodization window like the one in eq. 4.3 and fig. 4.10 is applied to them so that the features at the edges of the patches merge smoothly into one another. However, averaging the generated features together causes a loss in the small scales (high multipoles) angular power spectra, expecially in the points were more patches overlap with each other.

This chapter will explain how the recomposition is done and analyze the characteristics of the full sky maps.

6.1 Details of the Reprojection Procedure

For all the operations required to pass from flat sky patches to a full sky map, I used the reprojection script developed by Giuseppe Puglisi for the ForSE package (Krachmalnicoff and Puglisi, 2021). Here I will summarize the functioning of the reprojection code used for converting an HEALPix fullsky map to an array of flat, square patches of $20^\circ \times 20^\circ$ and viceversa.

The HEALPix[1] (Hierarchical Equal Area isoLatitude Pixelation) subdivision of the sphere consists in a tessellation of the sphere with 12 curvilinear quadrilaterals of the same area (see fig. 6.1 as a reference). The number of pixels can be increased by powers of 4, by dividing in 2 the side of each pixel. So the resolution of a HEALPix map depends on the number of pixels per side in one of the 12 basic quadrilaterals (which is always a power of 2).

To convert full sky maps to square patches, the code uses functions from the Healpy library to project a portion of the HEALPix sphere into square patches that cover a $20^\circ \times 20^\circ$ area . The projection is done at 174 different locations on the sphere. The centers of the projected patches are located along circles with the same latitude,

[1] HEALPix documentation: https://healpix.sourceforge.io/

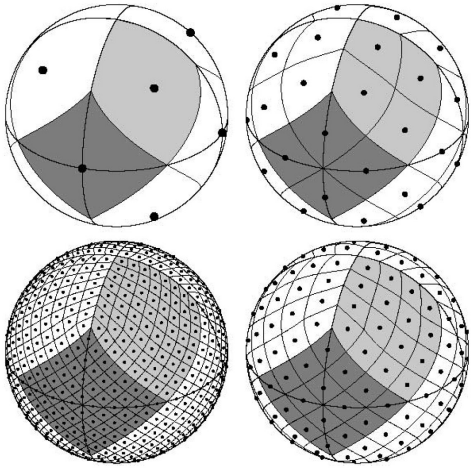


FIGURE 6.1: HEALPix Tesselation Scheme, example with increasing number of pixels per side.

18° distant from one another. The number of equally spaced patches per circle depends on the latitude (7 patches at the poles, 10 at $\pm 72°$ and 20 at other latitudes), in a way that the sphere is covered more homogeneously possible while still allowing enough overlap between the patches. So, in regions close to the poles, the areas where two or more patches overlap are wider and the number of overlapping patches in one point is higher than in regions close to the equator. This implies that the smoothing of the small scale features is more pronounced in proximity of the poles (see 6.2.2). To convert the square patches into full sky maps, the opposite projection is performed, at the same locations, but with the application of the apodization window, to merge the patches seamlessly.

The patches with 12′ resolution have 320 px per side, while patches with 3′ resolution have 1280 px per side. In the reprojection to full sky HEALPix maps, they are converted, respectively, in maps with 2048 and 4096 pixels per side. Conventionally, a feature of a sky map is considered resolved if it is covered by at least 3 pixels in the map. Therefore, maps with $N_{side} = 2048\,px, 4096\,px$, whose pixels have sizes of $S_{px} = 1.71', 0.859'$ resolve features up to $r = 5.13', 2.58'$, which are resolutions sufficient to conserve all the information contained in the 12′ and 3′ patches respectively.

6.2 Full Sky Maps Results

The results of the full sky recomposition are shown in fig. 6.2. From the top down, in the first three rows, one finds the original HEALPix Q and U maps with 80′ resolution, the maps with 12′ resolution and small scales generated by the noise trained network, the maps with 3′ resolution made by the iteration of the DCGAN. To give more relevance to the small scales and to check the possible presence of border effects from the patch overlaps, in the last three rows, one can see the subtraction between high resolution and low resolution maps (in order (12′-80′), (3′-80′) and (3′-12′)). The introduction of small scales both at 12′ and at 3′ doesn't change the large scale features of the emission maps and no border effects are visible in any map. This means that the apodization maps used to recompose the $20° \times 20°$ from the $5° \times 5°$ sub-patches and the mask used to go full sky were apt for the purpose.

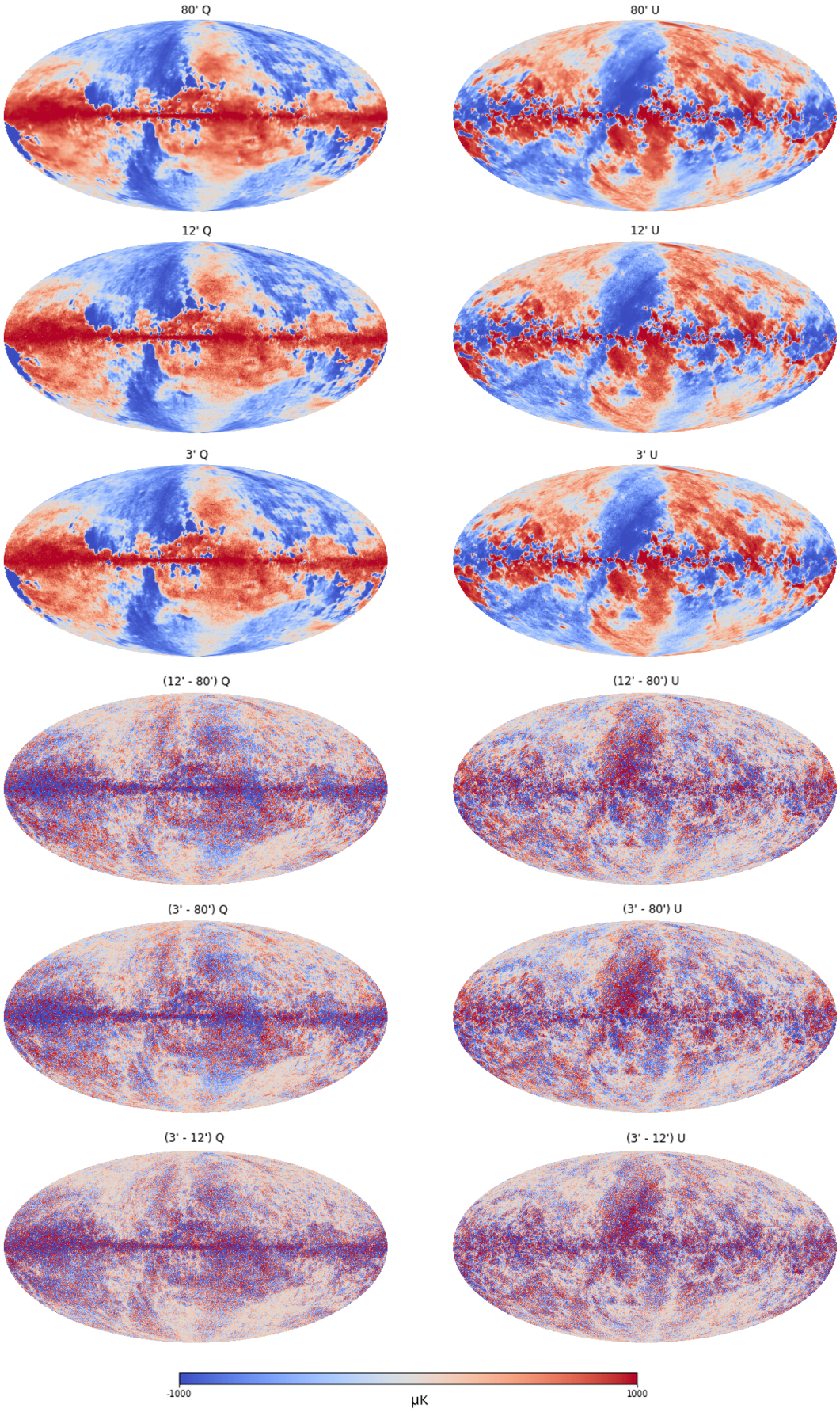


FIGURE 6.2: Full sky maps after the patch recomposition. First column is for Q maps, second is for U maps. The first three lines display, in order, the maps with resolutions up to 80′, 12′ and 3′. In the second three lines there are the subtraction maps (12′-80′), (3′-80′) and (3′-12′).

6.2.1 Full Sky Maps Power Spectra

Angular power spectra in E and B modes were computed also for the full sky maps, this time using the `sphtfunc.anafast` function from the Healpy library. The results are shown in 6.3 for the power spectra of the Q and U maps at 80′, 12′ and 3′. Note that the three spectra coincide exactly until the scale of the order of each map's resolution. This is trivial for the two maps with the lowest resolution, because the large scales at 80′ are always the same. For the map at 3′, however, attention was paid to make sure that the 12′ patches from which the 20′ input sub-patches were created were the same used to build the 12′ full sky map. The small and fast oscillations of the spectra at low multipoles are a spurious effect due to the particular function used to compute the spectra. The relevant thing to observe, instead, is that all the maps have a power spectra that follows the same power law distribution up to the multipoles corresponding to each map's resolution scale, i.e. $\ell = 135, 900, 3600$.

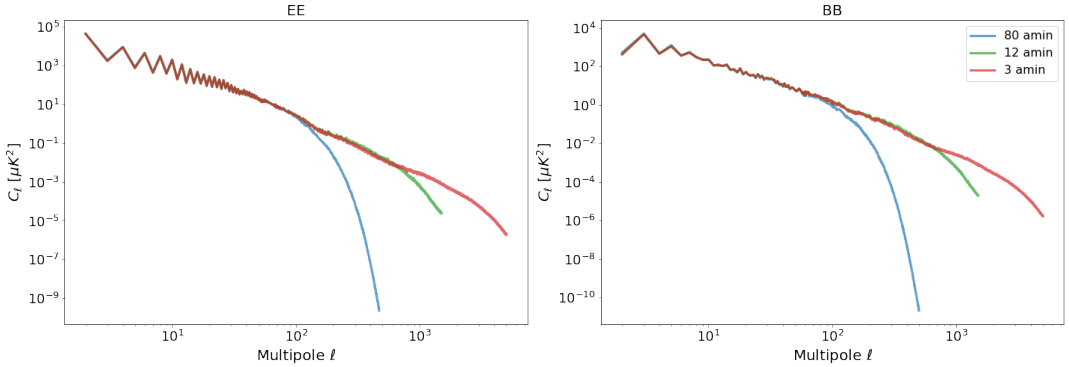


FIGURE 6.3: Angular power spectra in E and B for Q and U full sky maps at 80′ (blue), 12′ (green) and 3′ (red).

6.2.2 Quantifying the Angular Power Loss

To have a first qualitative idea of how much power is lost in the small scales because of averaging the overlapping patches, one can compare the zoomed full sky maps at different resolutions for a window positioned on one pole and for a window on the equator. The comparison is shown in fig. 6.4. It is evident that the small scales at 12′, and especially those at 3′, are sharper at the equator rather than at the pole.

To estimate the power loss more quantitatively, a possible way is to re-project again the full sky maps into square patches and then compare the angular power spectra of the original square patches and that of the newly re-projected ones. One can define the quantity $r_{\pm b}(\ell)$ as the ratio between the C_ℓ coefficients for the original patches and the re-projected ones, averaged among all patches at the same latitude $\pm b$:

$$r_{\pm b}(\ell) := \left\langle \frac{C^{(orig)}_{\pm b,l}(\ell)}{C^{(repr)}_{\pm b,l}(\ell)} \right\rangle_l \tag{6.1}$$

Values of $r_{\pm b}(\ell)$ greater than 1 indicate the presence of a loss in power. The ratio is computed as a function of ℓ to see at what scales the loss is greater and as a function of b to determine at which latitudes the effect becomes important, since the overlap area and the number of the overlapping patches grow with increasing $|b|$. The values of the ratio $r_{\pm b}(\ell)$, for E and B modes, are plotted in fig. 6.5 for the patches with 3

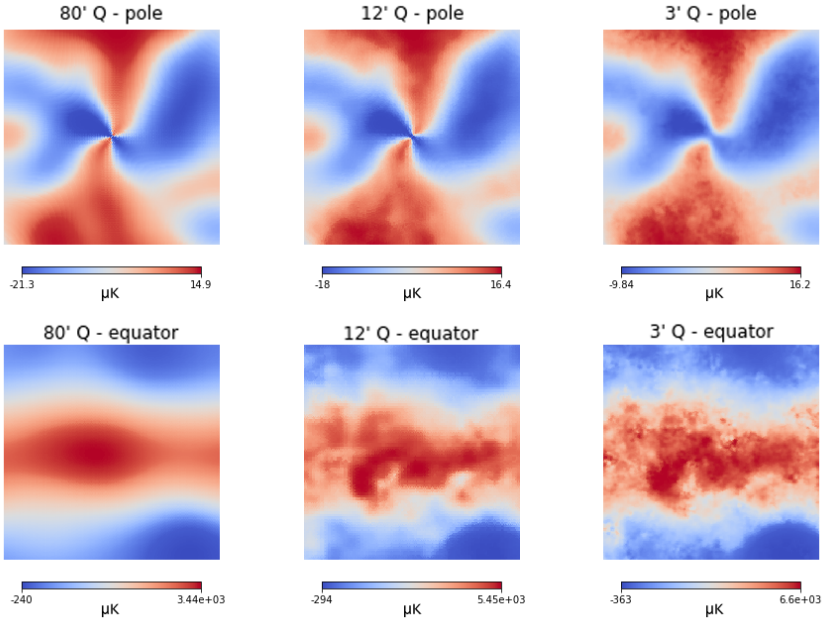


FIGURE 6.4: Comparison between zoomed portions of Q full sky maps positioned one at a pole and the other at the equator for maps with 80′, 12′ and 3′ resolutions.

arcminutes resolution. As expected, $r_{\pm b}(\ell)$ increases with increasing values of the latitude, as well as with higher multipoles, reaching a plateau for all latitudes around $\ell \sim 900$ ($\theta \sim 12'$). However it remains below the value of 2 for all latitudes, except for the patches at the poles.

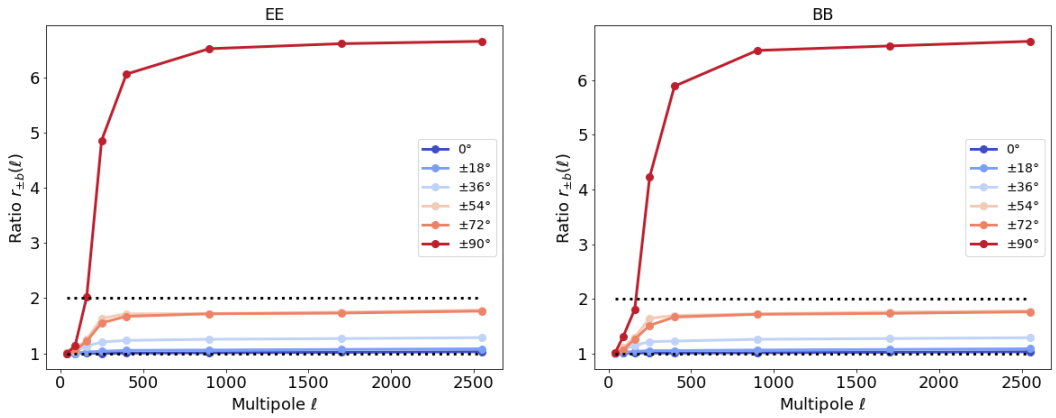


FIGURE 6.5: Ratio between the angular power spectra of the square patches at 3′ resolution, before and after the full sky reprojection. The ratio quantifies the power lost in the reprojection process and it is shown as function of the multipoles for different values of galactic latitude.

Conclusions

The quest for detecting primordial B-modes and measuring the tensor-to-scalar ratio of the primordial gravitational waves predicted by inflation requires precise measurements and realistic modeling of the CMB foregrounds. Primordial B-modes are contaminated by B-modes produced by lensing, which peak around small angular scales at $\ell \sim 1000$. Algorithms that estimate the lensing potential from CMB signal were developed with the purpose of performing delensing and cleaning the B-modes signal. For calibration they require accurate models of the CMB foregrounds, in particular of the galactic emission.

With this thesis I present a method for introducing variability in the galactic foreground model ForSE, based on a GAN network, which is able to produce an output map with non-Gaussianities at a resolution of 12 arcminutes, starting from an input map with resolution of 80′ arcminutes. This was achieved by adding noise to the input square maps that were used during training and to the input patches on which the trained network was applied. The introduction of noise generated instability in the network training and increased the number of iterations required to reach convergence. However, the similarity between the NN-generated output maps and the small scale maps used as a training set was still satisfactory for all training cases, with the best results obtained for low levels of noise. Anyway, even the trainings performed with high SNRs or with noise input only, produced output maps with small scales containing non-Gaussianities. Another result obtained by this work is the successful iteration of the procedure for creating small scale feature maps, to build foreground maps with the even higher resolution of 3′ arcminutes. Reaching such a resolution is relevant because the target resolution of the next CMB experiments, that are currently under preparation, is around the order of the arcminutes. The small scale square $20^\circ \times 20^\circ$ maps at 12′ and 3′ returned by the DCGAN were rescaled to the correct physical amplitude and then combined to form full sky maps in the HEALPix pixelation scheme.

Further improvements could be made in the future to extend the scope of this work. For example, the iteration of the algorithm was conducted using the network trained with noise only input, but the procedure is set up so that one can try to reach the same result with a GAN trained with large scales in the input. Also, having more computational power or longer computational time available, one could repeat the iterative procedure, but dividing the 12′ patches in 5×5 sub-patches, rather than 4×4 sub-patches, so to reach a resolution of 2.6′ instead of 3′. Furthermore, to better characterize the non-Gaussianities in the small scale maps, a useful thing would be to implement a code to compute the bispectrum or higher order correlation functions, to fully characterize the statistics of the generated small scale features. Lastly, issue that remains open is how to compensate for the angular power lost in the reprojection procedure, which was significant at high latitudes.

In future work, the full sky maps at 3′, produced with the method here described,

will be used as foreground models to test algorithms for estimating the weak lensing potential, in the prospect of performing delensing on CMB maps. The lensing reconstruction algorithms are sensitive to non-Gaussianities in the polarized emission, since this is characteristic of lensing B-modes. At the moment, the experimental foreground maps are not available with a resolution up to the arcminute scale and the current foreground models do not include non-Gaussianities at the small scales, even though it is expected that the real emission has a non-Gaussian statistics at all scales. Therefore it is not known if the presence of non-Gaussianities from the foreground contaminants might affect the estimate of the lensing potential and introduce a bias in the delensing procedure and so a bias on the estimate of the tensor-to-scalar ratio r. Hopefully the maps produced in this work might help to clarify whether the lensing algorithms are affected by foreground non-Gaussianities at small scales, so that in case they can be re-calibrated.

Appendix

A Gaussian Smoothing

Saying that a map has an effective beam with $\sigma = \sigma_a$ is equivalent to saying that the map, with virtually infinite resolution, is convoluted with a Gaussian window function, with $\sigma = \sigma_a$. In the harmonic domain, the convolution of two functions becomes the product of the two functions, and a Gaussian function with $\sigma = \sigma_a$ becomes a Gaussian function with $\sigma = \frac{1}{\sigma_a}$. In particular, in the convolution with the Gaussian beam, the coefficients $a_{\ell m}$ of the harmonic expansion of the function representing the map with infinite resolution get multiplied by a factor $e^{-\ell(\ell+1)\sigma^2}$. To to turn a map with effective beam with σ_a into a map with effective beam with $\sigma_b > \sigma_a$, one has to convolute the initial map with a Gaussian beam with $\sigma_c = \sqrt{\sigma_b^2 - \sigma_a^2}$:

$$a_{\ell m} e^{-\ell(\ell+1)\sigma_a^2} e^{-\ell(\ell+1)\sigma_c^2} = a_{\ell m} e^{-\ell(\ell+1)(\sigma_a^2+\sigma_b^2-\sigma_a^2)} = a_{\ell m} e^{-\ell(\ell+1)\sigma_b^2} \tag{A.1}$$

Note that both the FWHM and the σ can be used to determine the width of the Gaussian beam, and since they are proportional to each other, under the relation:

$$\text{FWHM} = 2\sqrt{2\log 2}\,\sigma \tag{A.2}$$

working with one or the other is completely equivalent.

Bibliography

Abazajian, K. N. et al. (2016). *CMB-S4 Science Book, First Edition*. arXiv: 1610.02743 [astro-ph.CO].

Alonso, D. et al. (Jan. 2019). "A unified pseudo-C_ℓ framework". In: *Monthly Notices of the Royal Astronomical Society* 484.3, pp. 4127–4151. ISSN: 0035-8711. DOI: 10.1093/mnras/stz093. eprint: https://academic.oup.com/mnras/article-pdf/484/3/4127/27747342/stz093.pdf. URL: https://doi.org/10.1093/mnras/stz093.

Aylor, K. et al. (Oct. 2020). "Cleaning our own dust: simulating and separating galactic dust foregrounds with neural networks". In: *Monthly Notices of the Royal Astronomical Society* 500.3, 3889–3897. ISSN: 1365-2966. DOI: 10.1093/mnras/staa3344. URL: http://dx.doi.org/10.1093/mnras/staa3344.

Baccigalupi, C. (Jan. 2021). *Linear Cosmological Perturbations and Cosmic Microwave Background Anisotropies - Lectures at SISSA - Astrophysics and Cosmology PhD school*.

Baumann, D. (2012). *TASI Lectures on Inflation*. arXiv: 0907.5424 [hep-th].

Beck, D., J. Errard, and R. Stompor (2020). "Impact of polarized galactic foreground emission on CMB lensing reconstruction and delensing of B-modes". In: *Journal of Cosmology and Astroparticle Physics* 2020.06, pp. 030–030. DOI: 10.1088/1475-7516/2020/06/030. URL: https://doi.org/10.1088/1475-7516/2020/06/030.

Bennett, C. L. et al. (1993). "Scientific results from the Cosmic Background Explorer (COBE)". In: *Proceedings of the National Academy of Sciences* 90.11, pp. 4766–4773. ISSN: 0027-8424. DOI: 10.1073/pnas.90.11.4766. eprint: https://www.pnas.org/content/90/11/4766.full.pdf. URL: https://www.pnas.org/content/90/11/4766.

Bennett, C. L. et al. (Sept. 2013). "Nine-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Final Maps and Results". In: *The Astrophysical Journal Supplement Series* 208.2, p. 20. ISSN: 1538-4365. DOI: 10.1088/0067-0049/208/2/20. URL: http://dx.doi.org/10.1088/0067-0049/208/2/20.

Bersanelli, M. et al. (Aug. 2012). "A coherent Polarimeter Array for the Large Scale Polarization Explorer Balloon Experiment". In: *Proceedings of SPIE - The International Society for Optical Engineering* 8446. DOI: 10.1117/12.925688.

BICEP2 Collaboration (June 2014). "Detection of B-Mode Polarization at Degree Angular Scales by BICEP2". In: *Phys. Rev. Lett.* 112 (24), p. 241101. DOI: 10.1103/PhysRevLett.112.241101. URL: https://link.aps.org/doi/10.1103/PhysRevLett.112.241101.

BICEP2/Keck and Planck Collaborations (Mar. 2015). "Joint Analysis of BICEP2/Keck Array and Planck Data". In: *Phys. Rev. Lett.* 114 (10), p. 101301. DOI: 10.1103/PhysRevLett.114.101301. URL: https://link.aps.org/doi/10.1103/PhysRevLett.114.101301.

Carrol, S. M. (2013). *Spacetime and Geometry: An Introduction to General Relativity*. Pearson Education.

Coulton, W. R. and D. N. Spergel (Oct. 2019). "The bispectrum of polarized galactic foregrounds". In: *Journal of Cosmology and Astroparticle Physics* 2019.10, pp. 056–056. DOI: 10.1088/1475-7516/2019/10/056. URL: https://doi.org/10.1088/1475-7516/2019/10/056.

Delabrouille, J. et al. (2013). "The pre-launch Planck Sky Model: a model of sky emission at submillimetre to centimetre wavelengths". In: *Astronomy & Astrophysics* 553, A96. DOI: 10.1051/0004-6361/201220019. URL: https://doi.org/10.1051/0004-6361/201220019.

Dicke, R. H. et al. (July 1965). "Cosmic Black-Body Radiation". In: *Astrophysical Journal* 142, pp. 414–419. DOI: 10.1086/148306.

Einstein, Albert (July 1916). "The Foundation of the Generalised Theory of Relativity". In: *Annalen der Physik* 354, pp. 769–822.

Goodfellow, I. J. et al. (2014). *Generative Adversarial Networks*. arXiv: 1406.2661 [stat.ML].

Gorski, K. M. et al. (Apr. 2005). "HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere". In: *The Astrophysical Journal* 622.2, 759–771. ISSN: 1538-4357. DOI: 10.1086/427976. URL: http://dx.doi.org/10.1086/427976.

Guzzetti, M. C. et al. (Apr. 2016). "Gravitational waves from inflation". In: *Rivista del nuovo cimento* 39.9. DOI: DOI10.1393/ncr/i2016-10127-1.

Hanson, D. et al. (Sept. 2013). "Detection of B-Mode Polarization in the Cosmic Microwave Background with Data from the South Pole Telescope". In: *Phys. Rev. Lett.* 111 (14), p. 141301. DOI: 10.1103/PhysRevLett.111.141301. URL: https://link.aps.org/doi/10.1103/PhysRevLett.111.141301.

Hu, W. and M. White (1997). *A CMB Polarization Primer - Lectures at Princeton*. URL: http://background.uchicago.edu/~whu/polar/webversion/polar.html.

Jew, L. and R. D. P. Grumitt (May 2020). "The spectral index of polarized diffuse Galactic emission between 30 and 44 GHz". In: *Monthly Notices of the Royal Astronomical Society* 495.1, pp. 578–593. ISSN: 0035-8711. DOI: 10.1093/mnras/staa1233. eprint: https://academic.oup.com/mnras/article-pdf/495/1/578/33237338/staa1233.pdf. URL: https://doi.org/10.1093/mnras/staa1233.

Kamionkowski, M. and E. D. Kovetz (2016). "The Quest for B Modes from Inflationary Gravitational Waves". In: *Annual Review of Astronomy and Astrophysics* 54.1, pp. 227–269. DOI: 10.1146/annurev-astro-081915-023433. URL: https://doi.org/10.1146/annurev-astro-081915-023433.

Kim, C. G., S. K. Choi, and R. Flauger (July 2019). "Dust Polarization Maps from TIGRESS: E/B Power Asymmetry and TE Correlation". In: *The Astrophysical Journal* 880.2, p. 106. DOI: 10.3847/1538-4357/ab29f2. URL: https://doi.org/10.3847/1538-4357/ab29f2.

Kingma, D. P. and J. Ba (2017). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs.LG].

Kodi Ramanah, D. et al. (May 2020). "Super-resolution emulator of cosmological simulations using deep physical models". In: *Monthly Notices of the Royal Astronomical Society* 495.4, 4227–4236. ISSN: 1365-2966. DOI: 10.1093/mnras/staa1428. URL: http://dx.doi.org/10.1093/mnras/staa1428.

Krachmalnicoff, N. (Dec. 2020). *Neural Networks: Theory and Practice - Lectures at SISSA - Astrophysics and Cosmology PhD school*.

Krachmalnicoff, N. and D. Poletti (May 2021). *B-modes in CMB polarization - Lectures at SISSA - Astrophysics and Cosmology PhD school*.

Krachmalnicoff, N. and G. Puglisi (Apr. 2021). "ForSE: A GAN-based Algorithm for Extending CMB Foreground Models to Subdegree Angular Scales". In: *The Astrophysical Journal* 911.1, p. 42. ISSN: 1538-4357. DOI: 10.3847/1538-4357/abe71c. URL: http://dx.doi.org/10.3847/1538-4357/abe71c.

Krachmalnicoff, N. et al. (2018). "S-PASS view of polarized Galactic synchrotron at 2.3 GHz as a contaminant to CMB observations". In: *Astronomy & Astrophysics* 618, A166. DOI: 10.1051/0004-6361/201832768. URL: https://doi.org/10.1051/0004-6361/201832768.

LeCun, Y. et al. (1989). "Handwritten Digit Recognition with a Back-Propagation Network". In: *NIPS*.

Li, Y. et al. (2021). "AI-Assisted Superresolution Cosmological Simulations". In: *Proceedings of the National Academy of Sciences* 118.19. ISSN: 0027-8424. DOI: 10.1073/pnas.2022038118. eprint: https://www.pnas.org/content/118/19/e2022038118.full.pdf. URL: https://www.pnas.org/content/118/19/e2022038118.

Lorensen, W. and H. Cline (Aug. 1987). "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *ACM SIGGRAPH Computer Graphics* 21, pp. 163–. DOI: 10.1145/37401.37422.

Louis, T. et al. (June 2017). "The Atacama Cosmology Telescope: two-season ACTPol spectra and parameters". In: *Journal of Cosmology and Astroparticle Physics* 2017.06, pp. 031–031. DOI: 10.1088/1475-7516/2017/06/031. URL: https://doi.org/10.1088/1475-7516/2017/06/031.

Mantz, H., K. Jacobs, and K. Mecke (Dec. 2008). "Utilising Minkowski Functionals for Image Analysis: a marching square algorithm". In: *J. Stat. Mech. P12015 J. Stat. Mech* 12. DOI: 10.1088/1742-5468/2008/12/P12015.

Mustafa, M. et al. (May 2019). "CosmoGAN: creating high-fidelity weak lensing convergence maps using Generative Adversarial Networks". In: *Computational Astrophysics and Cosmology* 6.1. ISSN: 2197-7909. DOI: 10.1186/s40668-019-0029-9. URL: http://dx.doi.org/10.1186/s40668-019-0029-9.

Peebles, P.J.E. (1993). *Principles of Physical Cosmology*. Princeton University Press.

Penzias, A. A. and R. W. Wilson (July 1965). "A Measurement of Excess Antenna Temperature at 4080 Mc/s." In: *Astrophysical Journal* 142, pp. 419–421. DOI: 10.1086/148307.

Planck Collaboration I (2020). "Planck 2018 results - I. Overview and the cosmological legacy of Planck". In: *Astronomy&Astrophysics* 641, A1. DOI: 10.1051/0004-6361/201833880. URL: https://doi.org/10.1051/0004-6361/201833880.

Planck Collaboration IV (Sept. 2020). "Planck 2018 results - IV. Diffuse component separation". In: *Astronomy & Astrophysics* 641, A4. DOI: 10.1051/0004-6361/201833881. URL: https://doi.org/10.1051/0004-6361/201833881.

Planck Collaboration V (2020). "Planck 2018 results - V. CMB power spectra and likelihoods". In: *Astronomy & Astrophysics* 641, A5. DOI: 10.1051/0004-6361/201936386. URL: https://doi.org/10.1051/0004-6361/201936386.

Planck Collaboration X (Oct. 2016). "Planck 2015 results - X. Diffuse component separation: Foreground maps". In: *Astronomy & Astrophysics* 594, A1. DOI: 10.1051/0004-6361/201525967. URL: https://doi.org/10.1051/0004-6361/201525967.

Planck Collaboration XI (2020). "Planck 2018 results - XI. Polarized dust foregrounds". In: *Astronomy & Astrophysics* 641, A11. DOI: 10.1051/0004-6361/201832618. URL: https://doi.org/10.1051/0004-6361/201832618.

Planck Collaboration XLVIII (Dec. 2016). "Planck intermediate results. XLVIII. Disentangling Galactic dust emission and cosmic infrared background anisotropies". In: *Astronomy & Astrophysics* 596, A109. ISSN: 1432-0746. DOI: 10.1051/0004-6361/201629022. URL: http://dx.doi.org/10.1051/0004-6361/201629022.

Planck Collaboration XXV (Oct. 2016). "Planck 2015 results - XXV. Diffuse low-frequency Galactic foregrounds". In: *Astronomy & Astrophysics* 594, A25. DOI: 10.1051/0004-6361/201526803. URL: https://doi.org/10.1051/0004-6361/201526803.

Planck Collaboration XXX (Feb. 2016). "Planck intermediate results - XXX. The angular power spectrum of polarized dust emission at intermediate and high Galactic latitudes". In: *Astronomy & Astrophysics* 586, A133. DOI: 10.1051/0004-6361/201425034. URL: https://doi.org/10.1051/0004-6361/201425034.

Planck Collaboration XXXVIII (2016). "Planck intermediate results - XXXVIII. E- and B-modes of dust polarization from the magnetized filamentary structure of the interstellar medium". In: *Astronomy & Astrophysics* 586, A141. DOI: 10.1051/0004-6361/201526506. URL: https://doi.org/10.1051/0004-6361/201526506.

Puglisi, G. and X. Bai (Dec. 2020). "Inpainting Galactic Foreground Intensity and Polarization Maps Using Convolutional Neural Networks". In: *The Astrophysical Journal* 905.2, p. 143. ISSN: 1538-4357. DOI: 10.3847/1538-4357/abc47c. URL: http://dx.doi.org/10.3847/1538-4357/abc47c.

Radford, A., L. Metz, and S. Chintala (2016). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. arXiv: 1511.06434 [cs.LG].

Rodríguez, A. C. et al. (Nov. 2018). "Fast cosmic web simulations with generative adversarial networks". In: *Computational Astrophysics and Cosmology* 5.1. ISSN: 2197-7909. DOI: 10.1186/s40668-018-0026-4. URL: http://dx.doi.org/10.1186/s40668-018-0026-4.

Ronneberger, O., P.Fischer, and T. Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: LNCS 9351, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a.

Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 65.6, 386–408. DOI: 10.1037/h0042519.

Schawinski, K. et al. (Jan. 2017). "Generative Adversarial Networks recover features in astrophysical images of galaxies beyond the deconvolution limit". In: *Monthly Notices of the Royal Astronomical Society: Letters*, slx008. ISSN: 1745-3933. DOI: 10.1093/mnrasl/slx008. URL: http://dx.doi.org/10.1093/mnrasl/slx008.

Springel, V. et al. (June 2005). "Simulations of the formation, evolution and clustering of galaxies and quasars". In: *Nature* 435.7042, 629–636. ISSN: 1476-4687. DOI: 10.1038/nature03597. URL: http://dx.doi.org/10.1038/nature03597.

Sugai, H. et al. (May 2020). "Updated Design of the CMB Polarization Experiment Satellite LiteBIRD". In: *Journal of Low Temperature Physics* 199 (3), pp. 1107–1117. ISSN: 1573-7357. DOI: 10.1007/s10909-019-02329-w. URL: https://doi.org/10.1007/s10909-019-02329-w.

The Simons Observatory Collaboration (Feb. 2019). "The Simons Observatory: science goals and forecasts". In: *Journal of Cosmology and Astroparticle Physics* 2019.02, pp. 056–056. DOI: 10.1088/1475-7516/2019/02/056. URL: https://doi.org/10.1088/1475-7516/2019/02/056.

Thorne, B. et al. (May 2017). "The Python Sky Model: software for simulating the Galactic microwave sky". In: *Monthly Notices of the Royal Astronomical Society* 469.3, pp. 2821–2833. ISSN: 0035-8711. DOI: 10.1093/mnras/stx949. URL: https://doi.org/10.1093/mnras/stx949.

Tristram, M. et al. (Mar. 2021). "Planck constraints on the tensor-to-scalar ratio". In: *Astronomy & Astrophysics* 647, A128. ISSN: 1432-0746. DOI: 10.1051/0004-6361/202039585. URL: http://dx.doi.org/10.1051/0004-6361/202039585.

Ullmo, M., A. Decelle, and N. Aghanim (July 2021). "Encoding large-scale cosmological structure with generative adversarial networks". In: *Astronomy and Astrophysics* 651, A46. ISSN: 1432-0746. DOI: 10.1051/0004-6361/202039866. URL: http://dx.doi.org/10.1051/0004-6361/202039866.

Vansyngel, F. et al. (July 2017). "Statistical simulations of the dust foreground to cosmic microwave background polarization". In: *Astronomy & Astrophysics* 603, A62. DOI: 10.1051/0004-6361/201629992. URL: https://doi.org/10.1051/0004-6361/201629992.

Wechsler, R. H. and J. L. Tinker (2018). "The Connection Between Galaxies and Their Dark Matter Halos". In: *Annual Review of Astronomy and Astrophysics* 56.1, pp. 435–487. DOI: 10.1146/annurev-astro-081817-051756. URL: https://doi.org/10.1146/annurev-astro-081817-051756.

Zaldarriaga, M. and U. Seljak (Feb. 1997). "All-sky analysis of polarization in the microwave background". In: *Physical Review D* 55.4, 1830–1840. ISSN: 1089-4918. DOI: 10.1103/physrevd.55.1830. URL: http://dx.doi.org/10.1103/PhysRevD.55.1830.

Zhang, X. et al. (2019). *From Dark Matter to Galaxies with Convolutional Networks*. arXiv: 1902.05965 [astro-ph.CO].

Zhu, X. P. et al. (Apr. 2019). "Galaxy Morphology Classification with Deep Convolutional Neural Networks". In: *Astrophysics and Space Science* 364.4. ISSN: 1572-946X. DOI: 10.1007/s10509-019-3540-1. URL: http://dx.doi.org/10.1007/s10509-019-3540-1.

Zingales, T. and I. P. Waldmann (Nov. 2018). "ExoGAN: Retrieving Exoplanetary Atmospheres Using Deep Convolutional Generative Adversarial Networks". In: *The Astronomical Journal* 156.6, p. 268. ISSN: 1538-3881. DOI: 10.3847/1538-3881/aae77c. URL: http://dx.doi.org/10.3847/1538-3881/aae77c.