Fast Wiener filtering of CMB maps with Neural Networks

Moritz Münchmeyer
Perimeter Institute for Theoretical Physics
31 Caroline St N, Waterloo, Canada
mmunchmeyer@perimeterinstitute.ca

Kendrick M. Smith
Perimeter Institute for Theoretical Physics
31 Caroline St N, Waterloo, Canada
kmsmith@perimeterinstitute.ca

Abstract

The cosmic microwave background (CMB) is our main source of information about the big bang and the early universe. The computational bottleneck in optimal CMB data analysis is Wiener filtering, an expensive operation which is usually implemented using the conjugate gradient method. We show how a neural network can be trained to Wiener filter masked CMB maps to high accuracy. We propose an innovative neural network architecture, the WienerNet, which guarantees linearity in the data map. Our method does not require Wiener filtered training data, but rather learns Wiener filtering from tailored loss functions which are mathematically guaranteed to be minimized by the exact Wiener filter. Once trained, the neural network Wiener filter is a factor $\sim$1000 faster than the standard conjugate gradient method. Wiener filtering is used in many optimal CMB analyses, including power spectrum estimation, lensing and non-Gaussianities, and our method could be used to speed them up by orders of magnitude with minimal loss of optimality. The method should also be useful to analyze other statistical fields in cosmology and beyond.

1 Introduction

During the last few decades, analysis of cosmic microwave background (CMB) data has transformed our understanding of cosmological physics. The basic building block of optimal CMB data analysis is the Wiener filter, a linear operation whose input is a noisy, partial-sky pixelized CMB map, and whose output is the maximum likelihood CMB sky. Wiener filtering has been an essential ingredient in many landmark CMB results, for example optimal power spectrum analysis [1], limits on higher N-point functions [2], and gravitational lensing results [3].

The Wiener filter is defined as follows. Let $d = s + n$ be the observed CMB map, where s is the true CMB sky, and n is a noise realization. The Wiener filtered map y_{WF} is defined by:

$$y_{\rm WF} = S(S+N)^{-1}d \tag{1}$$

where S is the signal covariance matrix (given by the CMB power spectrum) and N is the noise covariance matrix (given by the experiment). We use an index-free notation where CMB maps are represented as length-$N_{\rm pix}$ vectors (where $N_{\rm pix}$ is the number of sky pixels) and denoted with lower-case $(s, d, y, \cdots)$, and operators are denoted with upper-case (S, N). In this paper, we assume that the noise covariance N is diagonal in pixel space, i.e. the noise is assumed uncorrelated between pixels, but is allowed to be anisotropic, an often used but not perfect approximation for real experiments. We represent the mask as a limiting case of anisotropic noise, by taking the noise level to be infinite in masked pixels.

Direct inversion of the $N_{\rm pix}$-by-$N_{\rm pix}$ matrix in Eq. (1) is computationally impossible for current CMB maps, with $N_{\rm pix} = \mathcal{O}(10^8)$ pixels. Therefore, conjugate gradient solvers [4] are usually employed

Second Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019), Vancouver, Canada.

to perform Wiener filtering of CMB data (see e.g. [5]), but the computational cost is still high, and Wiener filtering a large ensemble of maps is very expensive even with large computing resources. In this paper we show how neural networks can be trained to Wiener filter CMB maps to a very good approximation, at a tiny fraction of the computational costs of the conjugate gradient method.

2 Neural network architecture and loss functions

Our network architecture is inspired by techniques from deep learning image segmentation. Our proposed neural network architecture, the WienerNet, is shown in Fig. 1 and employs a series of convolutions. For the task of Wiener filtering, convolutions are a natural choice since without a mask the Wiener filter is a low-pass filter, which can be implemented in position space by a convolution. We use a *UNet* [6] architecture, with skip connections which pass information directly from lower level encoders to lower level decoders. The UNet architecture is shown in the blue parts of Fig. 1. These features are part of the standard deep learning image analysis toolkit developed over the last years. We now describe our innovations that make them applicable to Wiener filtering.

A key property of the Wiener Filter is that it is linear in the input map, which guarantees that a Gaussian field is transformed into a Gaussian field. Conventional neural networks have non-linearities, which build the highly non-linear functions needed in standard image analysis. In our architecture, we entirely drop activation functions in the map processing path (blue parts of Fig. 1), which is thus manifestly linear. The Wiener filter depends on the mask and noise of the experiment. In particular the mask breaks statistical homogeneity and thus translation invariance. To allow the network to learn this dependence on the mask, we process it in a second UNet type pathway (red parts of Fig. 1). The mask pathway can be non-linear, and we insert activation functions of *ReLU* type after every convolution. The two pathways communicate from the non-linear to the linear layer (never the other way round), by the weight-dependent pixelwise multiplication defined in Fig. 1.

Let y denote the output map from the neural network. Since the WienerNet is constrained to be linear in d (for a fixed mask), we can write

$$y = Md \tag{2}$$

for some matrix M which is learned during training, by minimizing a loss function $J(s, d, y)$ which can depend on the training data (s, d) and the output map y.

We will say that a loss function $J(s, d, y)$ is "viable" if it satisfies the following property. When $\langle J(s, d, y)\rangle$ is minimized over all choices of matrix M in Eq. (2), then the minimum occurs for $M = S(S + N)^{-1}$. Here, the expectation value $\langle\cdot\rangle$ is taken over an infinitely large training set. In this paper, we will consider three possibilities J_1, J_2, J_3. It can be shown with ordinary matrix calculus that each of these loss functions is viable, in the sense just defined.

$$J_1(d, y) = \frac{1}{2}(y - y_{\rm WF})^T A(y - y_{\rm WF}) \tag{3}$$

$$J_2(s, y) = \frac{1}{2}(y - s)^T A(y - s) \tag{4}$$

$$J_3(d, y) = \frac{1}{2}(y - d)^T N^{-1}(y - d) + \frac{1}{2}y^T S^{-1} y \tag{5}$$

In Eqs. (3), (4), A is an arbitrary positive definite matrix. We choose $A = I$ by default, but have experimented with choices which downweight low-frequency Fourier modes, where Wiener-filtered maps are numerically larger.

The idea behind these loss functions is as follows. Training on J_1 corresponds to minimizing the mismatch between the neural network output y and the Wiener filtered map $y_{\rm WF}$. This makes the training process intuitive, but computationally expensive since we need to Wiener-filter each training realization using the conjugate gradient algorithm. Training on J_2 corresponds to minimizing the mismatch between the neural network y and the true CMB sky s. The loss function J_3 is motivated by statistical considerations: we have

$$J_3(d, y) = -\log P(s|d)_{s=y} + \text{const.} \tag{6}$$

where $P(s|d)$ is the posterior likelihood of true CMB sky s, given noisy data realization d. Thus, minimizing J_3 can be interpreted as training the network to output the maximum likelihood CMB sky. We conveniently evaluate the signal term in Fourier space and the noise term in position space.

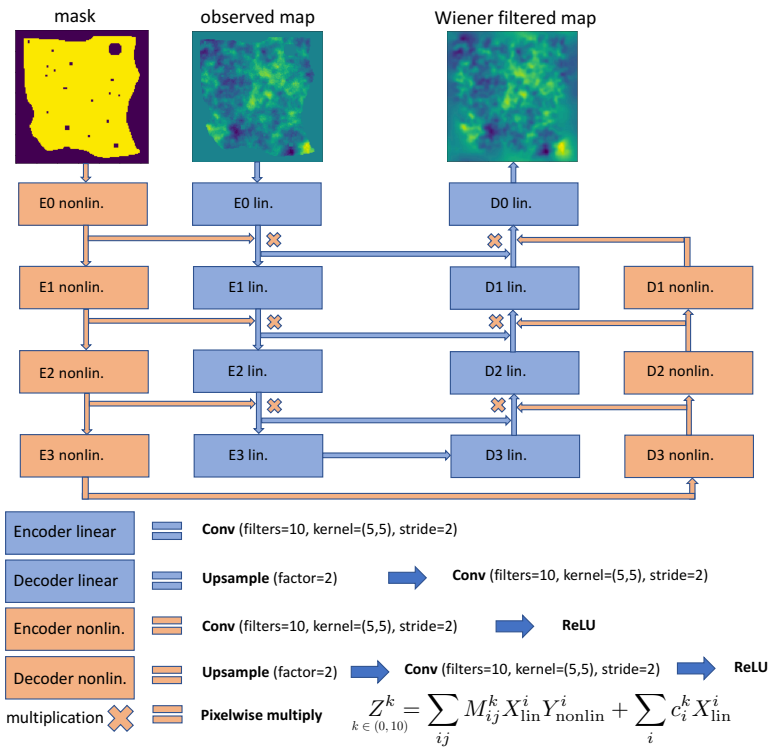

Figure 1: Network architecture of the WienerNet with a non-linear (red) and linear (blue) path with encoders (E) and decoders (D). Our default architecture has two more layers of encoders/decoders than shown in this figure. Bottom: Definition of the building blocks. Encoders downsample by a factor of two (stride=2), and decoders upsample by a factor of two, except encoder 0 and decoder 0 which have stride 1. The multiplication operation takes an input from the linear CMB path X and the non-linear mask path Y and multiplies them pixelwise according to the learned weights M^k_{ij} and c^k_i.

Empirically, we have found that both J_2 and J_3 work well, but J_3 performs significantly better in the high noise regime. Intuitively this happens because the J_3 difference term $(y - d)$ is much less noisy than the J_2 difference term $(y - s)$ in the high noise regime. In this way, bringing physical knowledge into the loss function, through the covariance matrices S and N, helps the network to learn the task even when the noise exceeds the signal by orders of magnitude. All results in the following sections are generated with J_3.

3 Data sets and results

Our training data consists of simulated CMB temperature and polarization maps, generated with the `quicklens`[1] package. The CMB temperature field T is a real valued scalar field describing the temperature in each pixel. CMB polarisation can be expressed in terms of the scalar fields (E, B), known as E-modes and B-modes. In our default configuration, we simulate (T, E, B) sky patches of size 5 deg $\times$ 5 deg, at a resolution of 128 $\times$ 128 pixels. This is small enough so that we can use the flat sky approximation, where spherical harmonics can be replaced with ordinary Fourier transforms. We have added realistic experimental noise levels, specified below. We have generated a typical experimental mask, representing sky cuts and smaller point-like sources, which can be seen in Fig. 1. We have verified that our results are very similar when using other masks of similar type. In total we create 10000 training maps as well as 1000 validation maps for each training setup. One could easily create much more training data, but we found good convergence with this number of maps.

[1] https://github.com/dhanson/quicklens

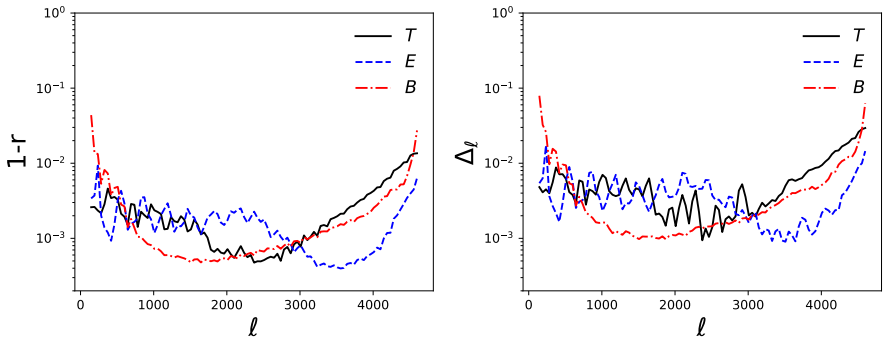

Figure 2: Quality measures for filtering 128x128 pixel CMB temperature and polarisation maps, obtained from 300 test maps. To illustrate different noise levels we show temperature (T) with 35μK-arcmin and polarisation (E,B) with 1μK-arcmin noise. The cross-correlation coefficient (left) between the neural network output and the exact Wiener filter is larger than 99% (with the exception of a small region in B modes where it drops to 95%). Similarly the power in the normalized difference map (right) between exact results and neural network results is at the 1% level. This means e.g. that cosmological parameter estimation based on our method would be about 99% optimal.

After training is complete, we compare the network output to the exact Wiener filtered maps (obtained using the conjugate gradient method) for 300 test maps. Training the network, which we implemented in TensorFlow, takes about 30 hours on our GeForce GTX 1080 Ti graphics card. Analyzing the 300 test maps with the trained neural network takes a few seconds, while exactly Wiener filtering them takes a few hours. This is a massive speed advantage of a factor of $\sim$1000, and the main advantage of the neural network.

For each map in the test set, we compare the neural network output y to the exact Wiener filtered map y_{WF} using two different metrics. The cross correlation coefficient r_ℓ between y and y_{WF} is shown as a function of angular wavenumber ℓ in Fig. 2 (left). We show the entire ℓ range of the sky patch from the lowest Fourier mode up to the Nyquist frequency of the sky patch ($\ell_{Nq} = 4608$ for the 5° side length). The quality of Wiener filtering is excellent, with about 99% cross correlation on all scales, independent of whether they are signal or noise dominated. We show the power spectrum Δ_ℓ of the difference map $(y - y_{WF})$, normalized by the total power, in Fig. 2 (right). Up to $\ell \simeq 3500$, the power in the difference map is at the 1% level. Overall we find excellent results in temperature as well as polarisation, even though the polarisation task involves learning to separate E and B modes in the presence of a mask, a notoriously difficult problem. It is likely that a larger neural network, for example with more feature maps, would improve the results even further.

4 Applications

After training, the neural network Wiener filters a map $\sim$1000 times faster than the exact conjugate gradient method. This massive speed improvement will allow us to make key CMB data analysis much more efficient, and solve problems that were previously thought computationally intractable. If one wanted to Wiener Filter only a single CMB map, due to the training time, our approach would not be beneficial. In practice however, CMB data analysis requires the determination of biases, normalizations and likelihoods with Monte Carlo methods. For example, to construct a power spectrum likelihood for the WMAP experiment, 10^5 Wiener filtered maps were needed [1]. For such analyses our approach is very beneficial.

Our results show that neural networks are an exceptionally powerful tool for the problem of Wiener filtering CMB maps, a particular type of maximum likelihood image reconstruction problem in cosmology. Other problems of this type include interferometric image reconstruction [7], galaxy shape estimation [8], CMB lensing reconstruction [9], reconstructing cosmological initial conditions from large-scale structure [10,11], and many others. Applications of neural networks to cosmological data analysis are still in their infancy, and it will be fascinating to explore these research directions further.

References

[1] Bennett, C. L., et. al., Astrophysical Journal Supplement, Volume 208, Issue 2, article id. 20, 54 pp. (2013).

[2] Y. Akrami et al, to appear in Astronomy & Astrophysics, arXiv:1905.05697.‘

[3] N. Aghanim et al, to appear in Astronomy & Astrophysics, arXiv:1807.06210.

[4] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, Cambridge University Press, New York, NY, USA, 2007, 3rd ed.

[5] K. M. Smith, O. Zahn, and O. Doré, Phys. Rev. D76, 043510 (2007), 0705.3980.

[6] O. Ronneberger, P. Fischer, and T. Brox, arXiv e-prints arXiv:1505.04597.

[7] K. Bouman, M. Johnson, D. Zoran, V. Fish, S. Doeleman, W. Freeman, IEEE Conference on Computer Vision and Pattern Recognition (2016) 913.

[8] R. Mandelbaum et al, Monthly Notices of the Royal Astronomical Society, 450 (2015) 2963.

[9] J. Caldeira, W. Wu, B. Nord, C. Avestruz, S. Trivedi, K. Story, Astronomy and Computing 100307 (2019) 28.

[10] J. Jasche, B. Wandelt, Monthly Notics of the Royal Astronomical Society 432 (2013) 894.

[11] Y. Feng, U. Seljak, M. Zaldarriaga, arXiv e-prints arXiv:1804.09687.