CMB-ML: A Cosmic Microwave Background Dataset for the Oldest Possible Computer Vision Task

Supplementary Material

A. More Background on the CMB

A.1. Selected CMB Missions

In the main body of this work, we refer to some of the surveys which have gathered CMB information. We provide more detail on those here.

There have been at least four pioneering satellite surveys designed with the purpose of observing the CMB signal from space: RELIKT-1, COBE (the COsmic Background Explorer), WMAP (the Wilkinson Microwave Anisotropy Probe), and the Planck mission. RELIKT-1 was a Soviet CMB satellite that observed at a single frequency for six months in 1983 and 1984. It was difficult to conclude that dipole and quadrupole measurements had a cosmological origin from this experiment alone. The COBE satellite operated from 1989 to 1993, confirming that CMB radiation follows a nearly perfect black body spectrum and has faint temperature anisotropies [33].

The WMAP satellite operated between 2001 and 2010, gathering data which supported the ΛCDM (Lambda Cold Dark Matter) cosmological model and provided narrow error bars for many of its governing parameters [19]. E-mode polarization of the CMB had previously been observed by ground-based experiments, but WMAP was the first to map it across the full sky.

The Planck mission satellite was operational between 2009 and 2013. This mission created maps of the microwave sky at higher resolution and of both E- and "B-modes" of polarization. It is important to note that these B-modes are dominated by foregrounds and lensing effects. Primordial B-modes in the CMB signal, which would indicate gravitational waves in the very early stages of the universe, have yet to be detected. Data from the Planck mission allowed further constraint of some cosmological parameters [6], and enabled better study of foregrounds components [10].

We emphasized satellite-based microwave detection. There are also observations made with both ground-based instruments as well as balloon-based instruments [17, 97]. These experiments are accompanied by additional challenges in data-processing, as those have different scanning strategies, they observe only portions of the sky, and they must handle noise which comes from the atmosphere. These experiments outnumber satellite experiments by roughly an order of magnitude. Because of the wide variety of scientific missions and approaches, we leave discussion to other resources.

Future surveys include CMB-S4 and LiteBird. CMB-S4 is a terrestrial survey, while LiteBIRD is a satellite. Among other goals, these missions will refine maps of B-mode polarization (with special attention to the primordial component), constrain key cosmological parameters, and address long-standing tensions that arise from other sources of cosmological observations. They may also provide tests for theories of the very early universe. More information about plans for these can be found in [1, 14].

A.2. Planck Instruments

The work of CMB-ML is centered around the Planck mission. Some of the settings for different frequency channels may seem strange, so we provide some more background on it. The Planck spacecraft has two instruments – the Low Frequency Instrument (LFI) and High Frequency Instrument (HFI). The instruments differ in the type of detector used. A total of nine frequency bands were observed.

The HFI had 52 detectors for the higher six frequency bands: 100, 140, 217, 353, 545, 853 GHz. These detectors use "bolometers," which are highly sensitive thermometers. These were able to measure weaker signals, but required extremely low temperatures (0.1K). Of the six bands, only the first four included polarization-sensitive detectors. The 52 detectors include pairs for measuring perpendicular polarization states, and redundant multiples to improve signal-to-noise ratio (and in case of failure). The highest frequency bands did not include detectors for polarization, as they were for mapping foregrounds in order to aid signal cleaning.

The LFI used 22 detectors for the three lower frequency bands: 30, 44 and 70 GHz [9]. These detectors use "high-electron-mobility transistors" (HEMTs), solid-state devices which can convert microwaves to electrical signals, operating at high frequency. Signals are stronger at these lower frequencies, so these could operate at higher temperatures (20K) with less concern for thermal noise. All of these detectors are sensitive to a particular polarization direction, so each detector was paired against another with perpendicular polarization. Because this instrument operated at higher temperatures, it was able to operate after the supply of helium (for cooling) ran out.

A.3. Mapmaking

Packets of raw data acquired by the detectors are sent each day to the Mission Operations Center (MOC). These were processed to extract all necessary signals and convert them

into time-ordered information (TOI). This TOI was used to generate models of instrumentation parameters (as reduced instrument models, "RIMO"). It was then cleaned to remove glitches and some amount of noise, while flagging periods of time that contain anomalies. Finally, the TOI was compressed from time samples into sky maps for Stokes parameters I, Q and U (temperature and polarization) for each frequency channel. Following [20], TOI for a detector, d, may be related to a map of the true sky signal, s, through

$$d = Ps + n, \tag{3}$$

where P encodes the scanning of the detector and orientation of the telescope at each point in time, and n is an average of the noise. Thus, the map-making problem can be viewed as searching for a linear operator L to generate map m:

$$m = Ld. \tag{4}$$

This brief description simplifies the process greatly. Sophisticated analysis is needed to combine multiple individual detectors for a given frequency, such that systematics are minimized. For the Planck Collaboration, different processing was used for LFI [9] and HFI instruments [8].

A.4. Components

There are many microwave-producing phenomena. In Fig. 3, we showed detailed views of four of them, illustrating different *classes* of the components. In Fig. 8, full-sky maps for these components are presented. In this section, we briefly describe the *particular* component signals used in CMB-ML.

All galactic components are diffuse and anisotropic. They include:

- Thermal dust: Small particles of dust emit microwaves as they cool.
- Synchrotron: Charged particles emit microwaves as they move through magnetic fields.
- "AME": Originally "Anomolous Microwave Emission," this is now commonly believed to be from very-rapidly spinning, nano-scale sized dust which has some electrostatic charge [31]. It has distinct spectral properties and spatial distribution from thermal dust.
- Free-free: When charged particles are deflected by nearby ions (without being captured), they lose kinetic energy and emit microwaves in the process.
- CO: Carbon-monoxide molecules transition between rotational states, emitting microwaves at discrete wavelengths.

Extragalactic components are always isotropic, and may be either diffuse or point sources. They include:

- Cosmic Infrared Background (CIB): When most stars were formed they emitted light which has since cooled. This radiation is diffuse because it has been obscured by dust and comes from many galaxies in the same line-of-sight.
- Thermal Sunyaev-Zel'dovich effect (tSZ): As CMB photons pass through hot ionized gas in galaxy clusters, they gain energy, resulting in a lower signal at low EM frequencies and higher signal at high frequencies. At the resolution of CMB-ML, these are point sources.
- Kinematic Sunyaev-Zel'dovich effect (kSZ): As CMB photons are scattered by electrons in either galaxy clusters or large-scale ionized gas, if whatever causes the scatter has a bulk velocity relative to the CMB path, then this will shift the CMB intensity.
- Radio Galaxies: Some galaxies have a core that emits "relativistic jets" of charged particles and photons. These galaxies are "radio" galaxies because the signal dominates radio waves.

A.5. Power spectra

Of critical concern for CMB analysis is the scale at which correlations are observed. For instance, at roughly 1° ($\approx \ell = 200$), we see a large peak in the power spectrum as well as "lumps" of this size in the maps, as described in Fig. 3. The angular power spectrum is defined as

$$C(\theta) \equiv \left\langle \frac{\Delta T}{T_0}(\hat{\mathbf{m}}) \frac{\Delta T}{T_0}(\hat{\mathbf{n}}) \right\rangle, \hat{\mathbf{m}} \cdot \hat{\mathbf{n}} = \cos\theta, \tag{5}$$

where $\frac{\Delta T}{T}(\hat{\mathbf{x}})$ is the anisotropy in direction $\hat{\mathbf{x}}$, θ is an amount of angular separation, and the ensemble average is over all directions and observer positions [55]. This can also be written, per [67], as a two-point correlation function:

$$C(\theta) = \sum_{\ell=2}^{\infty} \frac{2\ell+1}{4\pi} C_\ell P_\ell(\cos\theta), \tag{6}$$

relating angle θ to the "multipole moment" ℓ and power spectrum coefficients C_ℓ. Here, $C_\ell = \left\langle |a_{\ell m}|^2 \right\rangle$ and the $a_{\ell m}$ are spherical harmonic coefficients, defined as

$$\frac{\Delta T}{T}(\hat{\mathbf{x}}) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} a_{\ell m} Y_{\ell m}(\hat{\mathbf{x}}), \tag{7}$$

where $Y_{\ell m}$ are spherical harmonic basis functions.

Returning to the example of the first peak in the power spectrum at $\ell \approx 200$ and the "lumps" which are $\approx 1^\circ$, we have a maximum at $C(1^\circ)$ or C_{200}.

This is all to say, a spherical harmonic transformation (SHT) can allow us to convert the CMB anisotropy from real space to a harmonic domain, encoded as a series of $a_{\ell m}$'s. Averaging these spherical harmonic coefficients, $a_{\ell m}$, over all m for a particular ℓ, gives C_ℓ, which describes the variance at some angular scale (a function of ℓ). By

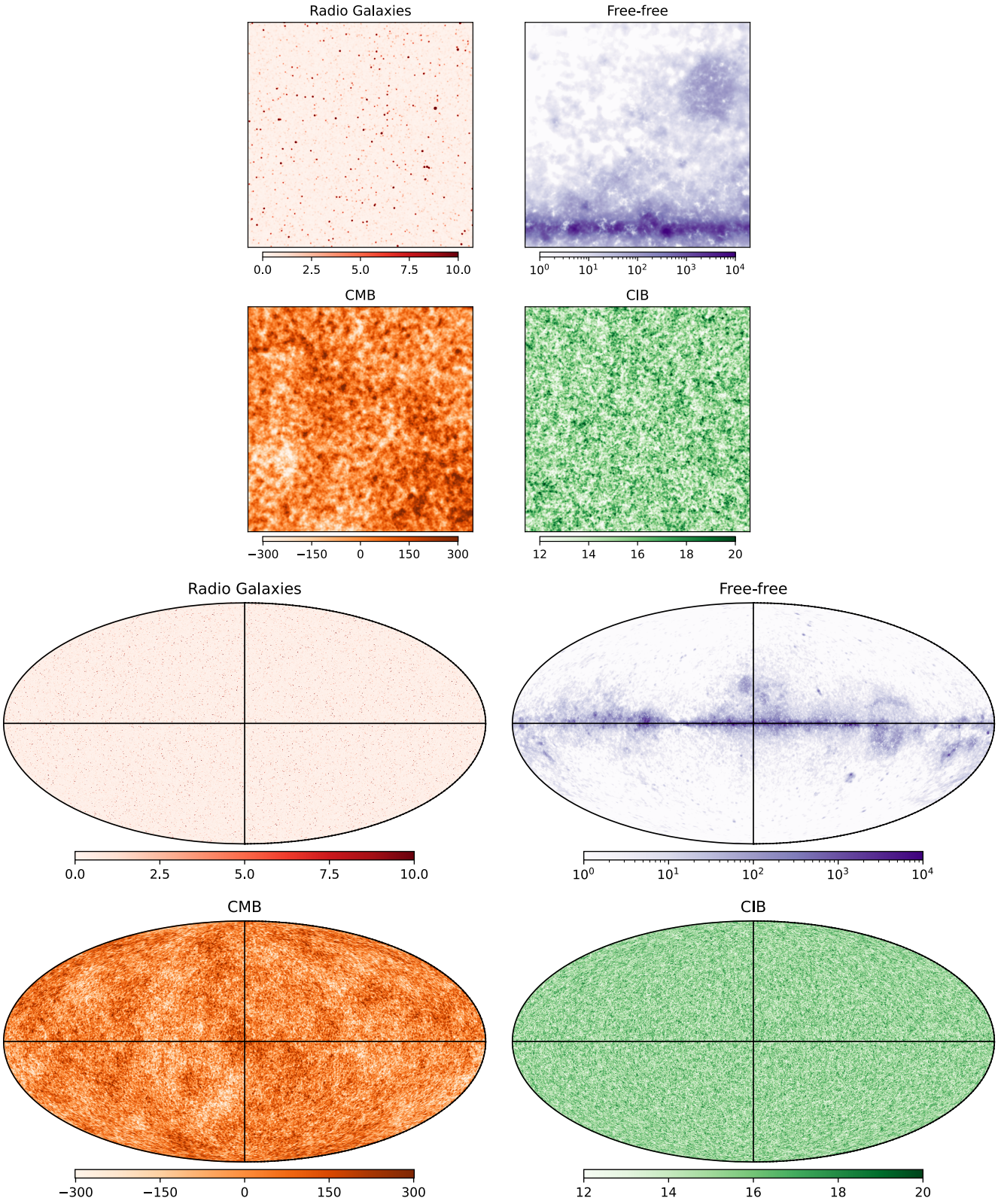

Figure 8. Detail view images, top, are the same as in the main body of the work. Full sky images, bottom, are provided as they may help orient the reader to the scale. The region in the detail view is above and to the left of galactic center, for all images.

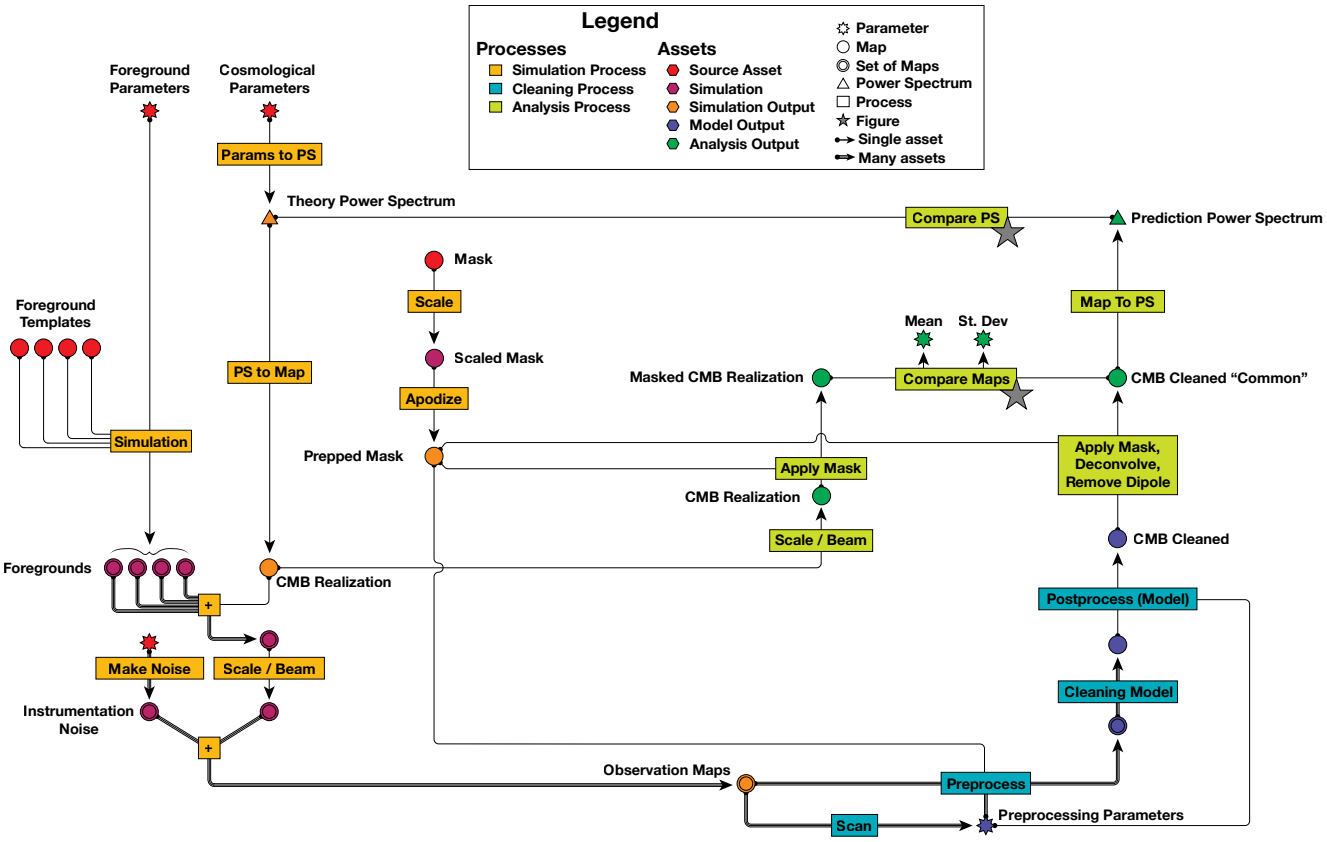

Figure 9. The CMB-ML pipeline. "Many assets" refers to having maps or parameters for each observation frequency, while "single asset" refers to those without frequency dependence.

considering a vector of $\mathbf{C}_\ell$ for all ℓ, we are provided with a useful summary statistic of spatial variations.

The SHT is a key tool in the map-processing toolbox. It is also a tool to use carefully. While the transformation is exact for continuous functions, in practice a maximum multipole moment, $\ell_{\rm max}$, must be chosen. This results in a bandwidth limit, analogous to more general sampling in Fourier analysis. If the true signal has information at higher multipoles than can be presented in the sampled signal (*e.g.*, $\ell > \ell_{\rm max}$), that information cannot be properly captured and the result may have artifacts (such as aliasing or ringing).

B. Architecture

As discussed previously, the pipeline consists of simulation, inference, and analysis stages. Each of these is further subdivided into tasks, where a task takes some input and produces some single output. This simplified further separation into a tiered architecture, where separate layers exist for the application flow, the physics logic, and the parameters.

The architecture was developed to avoid conflicts between stages while enabling flexibility. This helps to manage parameters, such as the $N_{\rm side}$, which are used across the stages. Storing file names separately and consistently ensures results are reproducible. The architecture also helps to make segments of the pipeline, such as the model used, interchangeable. Furthermore, the creation of data splits is performed in a way that single splits can have distinct characteristics. This would facilitate testing on multiple test splits, each with different classes of noise, for instance.

The system as a whole may be viewed as a directed acyclic graph. This is illustrated in Fig. 9. More information about particular stages will be given in other sections. The structure described in that flowchart is reflected in the code. Each process is implemented as an Executor object. Top-level Python scripts create a Pipeline object, then load each Executor into it. When the Pipeline is started, all Executors are initialized to ensure that configurations are available. Early stages of the pipeline are set aside to check for conflicting configurations. This pipeline setup has its advantages, including that an executor producing some asset (such as a trained model) can be easily disabled if only downstream segments of the pipeline are needed in subsequent runs.

The system is coordinated using Hydra[7] [105] to manage yaml configuration files. Configurations are modular; some are stored at the top level, but most are at the level of one of the following: `local_system`, `file_system`, `pipeline`, `scenario`, `splits`, `model/sims`, `model/pyilc`, `model/cmbnncs`. Further, the top-level configuration also contains interpolation parameters which are loaded by lower-level configurations for convenience. It is the hierarchy that allows consistent sets of parameters to be used.

File paths are generally set up automatically. The `local_system` configuration must be set for the locations of assets and the dataset. All file locations are defined relative to those directories.

Executors manage individual processes in the pipeline. The pipeline configuration yamls parameterize how they are run. This allows logic and parameters to remain separate. Each Executor will need to operate on some set of input assets and will generally produce some output asset. To avoid repetition (and accompanying errors), file locations are defined just for the output of an Executor; the input file locations are found from the Executor that produced the file. A special Executor exists to define source assets (e.g., Planck's raw files). Each asset has a file type. Handlers are defined to manage file I/O for the different formats of maps, power spectra, parameter files, and output figures. Stages in the pipeline yaml also track other parameters for that Executor, if there is no other suitable location.

To correlate code with output, every run of the code results logs being written to output directories (the `make_stage_log` can disable this behavior). Because there are many parameters with interplay between pipeline stages, this allows issues to be tracked to particular changes in the code. The stage logs include `logging` output from all stages run in the current session. The simulation stages with random seeds include the seeds for repeatability. The logs also include a mirror copy of the Hydra configurations and all imported code from within the CMB-ML codebase. This should ensure repeatability as long as the logs are preserved.

In subsequent sections, **bold-faced text** refers back to the pipeline diagram of Fig. 9.

C. Simulation details

C.1. Components

Maps are produced at a resolution of $N_{\text{side}} = 512$, corresponding to roughly 3 mega-pixels. This is chosen in keeping with previous work [80, 101] as well as because of computational resource limits. Either of the networks described in those works will run on modern GPUs with 40GB of VRAM, but a resolution $N_{\text{side}} = 1024$ map would require several GPUs.

The simulation pipeline first creates the target CMB map. For each simulation, **cosmological parameters** are drawn from the WMAP9 Monte Carlo Gibbs sampling chains, particularly the "$\Lambda\text{CDM} + \text{MNU}$" extended model [48]. It contains 603,936 correlated samples for each parameter. To preserve correlations, a single index is randomly chosen and all parameter values at that index are used together (one index per simulation instance). Parameters used are $H_0, \Omega_b h^2, \Omega_c h^2, \tau, n_s$, and $\sum m_\nu$. The default CAMB value for A_s was used in all simulations. Because the Planck mission narrowed the distribution, the more broad WMAP9 chains are used as a "reasonable broad" distribution to prevent overfitting. Using those parameters, a **theory power spectrum** is determined using the CAMB library[8] [49, 62]. The PySM3 `CMBLensed` object is used to create a **CMB realization**, including lensing effects by means of the TayLens [72] code.

Other foreground components are also generated by the PySM3 library, using the string presets. For **foreground parameters**, we use all the low-complexity galactic foregrounds (`d9, s4, f1, a1, co1`) [40] and all extragalactic foregrounds (`cib1, tsz1, ksz1, rg1`) [39]. All components are generated at an $N_{\text{side}} = 2048$ and the signal level is estimated for each center frequency of the nine nominal detector frequencies of the Planck mission, all converted to the common K_{CMB} equivalent temperature. The signals are summed, then each observation is downgraded to $N_{\text{side}} = 512$.

We then consider instrumentation effects. The Planck mission detectors had varying beam $FWHM$ because of physical practicalities. They chose map resolutions such that there were at least 2.5 pixels per beam $FWHM$. Decisions for CMB-ML were made from the opposite point of view. For computational reasons, the resolution of $N_{\text{side}} = 512$ was found to be large but practical. At this resolution, each pixel is approximately $6.87'$ per side. To use the Planck beam sizes mean that the maps are significantly undersampled, leading to issues with spherical harmonic transforms.

Many comparable datasets reportedly use common beams (the same size for all frequencies). While perfectly appropriate in astrophysical applications, it hides the inability of an algorithm to handle varied detector sizes. Instead, CMB-ML uses different beam sizes, without undersampling. Beginning with the conservative approximation of three pixels per beam $FWHM$ [92], a minimum beam FWHM of $20.6'$ is appropriate for the narrowest (857 GHz) detector. Because the Planck Collaboration used $FWHM = 4.64'$ for 857 GHz, we scale all other beams by the ratio $20.6/4.64$. This gives very wide beams, to

[7] https://hydra.cc/docs/intro

[8] https://camb.info

Table 2. Comparison of Planck and CMB-ML Beam FWHMs

Channel (GHz)	Beam FWHM Planck (arcmin)	CMB-ML (arcmin)
30	33.1	75.0
44	27.9	65.0
70	13.1	55.0
100	9.68	43.0
143	7.30	32.4
217	5.02	22.3
353	4.94	22.0
545	4.83	21.5
857	4.64	20.6

the point of having questionable utility, for the 30, 44, and 70 GHz detectors. For these we set arbitrary values. Values are listed in Tab. 2. Using varied $FWHM$ produces a dataset that will adequately illustrate trade-offs that models must make when working with the high-precision real-world datasets.

Scaling and application of beam effects occurs at the same time in SHT space using a PySM3 utility, `apply_smoothing_and_coord_transform()`.

Other works have preserved the Planck beams [101] or smoothed to a common beam [4, 80]. The first approach disregards beam effects entirely, as the simulations used in [101] are at too low a resolution. The second approach removes a confounding effect which may bias results toward one method or another. A simple ablation study of this may be interesting; the framework is in place to make this straightforward (but a distraction from the present context).

C.2. Noise

Critical attention to systematics (noise) was considered essential in order to get realistic power spectra results. The Planck collaboration produced simulated noise contributions beginning with TOI and proceeding through the full processing pipeline. Unfortunately, the software for this procedure could not be found and only 300 realizations were released. The importance of realistic noise is emphasized in related work on the `SRoll` algorithm [7, 30], and in other astrophysics work, e.g., [43] where shot noise is modeled in galaxy simulations.

The noise used in CMB-ML is correlated anisotropic noise, produced in two stages. We apply the same process for each detector. The first stage produces spatial anisotropy, and is fairly straightforward. First, a simple map is produced wherein every pixel is drawn from a Gaussian distribution with a mean of zero and variance one ($\sim \mathcal{N}(0, 1)$). This is then scaled to match the variance reported in Planck's observation maps for that detector, (*e.g.* as shown in Fig. 4) by multiplying by its square-root. The result is spatially anisotropic. However, no correlations exist in the map.

To produce correlations, we follow a slightly more complicated process. We first need a description of how noise is correlated. To produce this, we look to [8] for the FFP10 bank of simulations produced by the Planck Collaboration. This contains a fixed number of simulation maps for each detection channel, produced using a full end-to-end pipeline.

Because we want to produce unique noise for each CMB-ML simulation, we capture the statistics of these maps in the following way. We first calculate the average of 100 of these maps, so that we can isolate stationary signals which show up for some frequency channels. We then, for each of the 100 maps, subtract the average, apply a narrow galactic plane mask (masking $10°$ around the galactic equator) and determine the power spectrum, N_ℓ, of this. The maps are at high resolution (either $N_{\text{side}} = 1024$ or 2048), so we can get N_ℓ for thousands of ℓ. A complete covariance matrix would be overspecified, so we instead use scikit-learn's `PCA` (Principal Component Analysis) module for a summary of the covariance, with the added benefit of fast run-time. The average map, average power spectrum, target principle components, and associated target variances give us sufficient information to capture the distribution of these maps.

When it is time to create new white noise maps an anisotropic white noise map is created, then filtered to apply the correlation. We first create an anisotropic white noise map following the method at the start of this section. We then convert that to the $a_{\ell m}$ domain and get the power spectrum, $N_\ell^{(w)}$. Next, assuming our summary of the target distribution has k components, we draw $\mathcal{N}(0, 1)^k$ values and scale them to the target variances. We multiply these by the principle components and add the average power spectrum. This gives a synthetic $N_\ell^{(t)}$ from our approximation of Planck's noise power spectra. We follow a method inspired by [35] to create the filter,

$$f_\ell = \sqrt{\frac{N_\ell^{(t)}}{N_\ell^{(w)}}}. \tag{8}$$

We apply this filter to the anisotropic white noise $a_{\ell m}$ and perform an inverse spherical harmonic transformation, producing a map in pixel space. We subtract the monopole (average pixel value) of this map. Last, we add the average map for this channel of the Planck simulations, restoring the anisotropic portions of the noise.

This noise is added on top of the scaled and beam-smoothed maps to produce the final **observations maps**.

C.3. Masking

Masking is needed to handle the sharp discontinuities and high variance at the galactic plane and point sources. Two **prepped masks** are produced for CMB map analysis. Both are derived from the CNILC cleaning in the Planck 2015 data release (the `TMASK` field from the "COM_CMB_IQU-nilc_2048_R3.00_full.fits" file). The mask is downgraded from $N_{\text{side}} = 2048$ to 512, with a threshold at 0.9 for pixels at the border of the mask [3]. All values in this first mask are 0 or 1. This causes artifacts when computing SHTs, so a second apodized mask is also produced. We use the package NaMaster[9] [15] to perform 'C1' apodization at $20.6'$. The smoothed mask is used when calculating power spectra. The boolean mask is used when calculating pixel-domain statistics. The boolean mask is also provided to the CNILC method, which then performs its own apodization.

C.4. Simulation implementation

Because configurations are modular, multiple top-level configurations can pull the same parameters from lower-level configurations. We include several different top-level scripts that do precisely this. The first of these scripts creates simulations only (as these are often needed to run only once).

After the initial configuration checks, a series of executors prepare to create the noise maps. It is possible to create simulations with anisotropic white noise, in which case only executor `B_make_noise_cache.py` is needed. In the next three, Planck's original noise simulations are acquired, then average maps are created, then the noise models themselves are created. These run at a fairly high cost, as 900 simulations are needed (assuming 100 simulations per detector channel) which are downloaded from the Planck Legacy Archive. For this reason, noise models and average maps are included in the repository.

The remaining executors are straightforward. One creates observations and CMB maps, another creates noise (based on the noise type set in the simulation configuration), and a third creates the final observation maps. The final executor creates the masks needed for later stages.

D. Baseline Parameters

D.1. Baseline implementation

Again, multiple top-level scripts run the baseline models. One script will run cmbNNCS and analysis on all results from cmbNNCS alone. Because PyILC configures global Matplotlib settings with unlisted packages, one script imports and uses PyILC and a second does not import it for all analysis. A final script runs joint analysis on results from both, producing Fig. 7 and Tab. 1.

[9] https://github.com/LSSTDESC/NaMaster

When implementing a new method, the example given with cmbNNCS should be followed in general. This set of executors illustrates **preprocessing** and **postprocessing** data in bulk (the map files are large; preprocessing in a PyTorch dataloader would slow training considerably). Serial executors illustrate simple methods to do this, while parallel executors (using the `mulitprocessing` python library) are able to do this much more quickly. Executors for both training and inference largely follow common PyTorch patterns, with a few exceptions to match the original work [101]. A PyTorch `Dataset` subclass is provided and used by cmbNNCS. This subclass is written to be agnostic to the particular format needed for any method used.

Methods which are based on common patterns in the astrophysics community may instead refer to PyILC. This method has a single executor. It first creates a configuration file for the method, then runs the method in a working directory. The output prediction is copied to an appropriate location, along with the configuration that would produce such a result. The working directory is then cleared deliberately.

It is our sincere hope that these two examples are sufficient to get the intrepid researcher able to use CMB-ML to produce **CMB Cleaned Maps**. More detail on analysis is in the following section. We first look at details for each method.

D.2. cmbNNCS

cmbNNCS was run with the following settings:

- Features unscaled
- Target (CMB) scaled down by a factor of 5
- Initial learning rate 10^{-1}, with exponential decay to 10^{-6}
- Repeats each batch three times
- 120 epochs
- Batch size of 12
- UNet8
- Using 100, 143, 217, 353 GHz detectors only

We made every attempt to duplicate the work of the original authors, with the exception of their fully stochastic sampling method. The number of epochs is determined from the number of iterations (10,000) listed in that work. We did include the atypical repetition of batches (this is clearly marked in the Executor). We use PyTorch's `LambdaLR` to manage the exponentially decaying learning rate in the same way that it would have functioned with fully stochastic sampling.

Note that cmbNNCS scales the target signal by a blanket factor of 5. This occurs in place of more common standardization or min-max scaling, which could require some **scan** stage to produce **preprocessing parameters**.

D.3. CNILC Parameters

CNILC was run with the following settings:

- `ELLMAX` ($\ell_{\max}$): 1024
- `perform_ILC_at_beam`: 20.6
- `beam_FWHM_arcmin`: From simulation
- `ILC_bias_tol`: 0.001
- `taper_width`: 0
- `ellpeaks`: $[200, 715, 1025]$
- Mask: Non-apodized mask
- Frequencies: All except 30 GHz

`Ellpeaks` and `ELLMAX` were found by a coarse grid search. Values for `ELLMAX` were explored from 900 up to the band limit at 1460 (equivalent to the bandwidth limit at $20.6'$. The final value of 1024 minimizes error (and is $2N_{\text{side}}$, indicating a connection to bandwidth limit guidelines). Values for the lowest ℓ_{peak} were explored between 100 and 300. Below 200, inaccuracy at the edges of the masks causes error. Above 200, it appears that variance at higher multipole moments dominates $\ell < 4$ signal, resulting in large residuals that raise the error. The central $\ell_{\text{peak}} = 715$ minimized the error, with almost negligible variation on the order of 10^{-3} when changing ℓ_{peak} in increments of 200. Adding further needlet windows (ℓ_{peak}'s) also increased error at a similar scale and increased the run time by 50%.

Other parameters were set based on domain knowledge. The `ILC_bias_tol` was found to allow a remaining dipole when used at greater values of 0.001. Parameters for the beam window FWHMs come from the simulation parameters. The mask without apodization is used, as PyILC apodizes the mask separately for each needlet window.

E. Analysis

E.1. Analysis implementation

To facilitate fair comparison between methods, maps must be analyzed in the same way. This occurs in a series of executors. Most of these are included in the top-level baseline scripts. A final top-level script exists to report across different baseline models.

To this end, we produce a **Common CMB Cleaned** map by applying the smoothed mask, handling beam effects, and removing the dipole.

Many methods convert between pixel and harmonic domains. As mentioned at the end of Sec. A.5, bandwidth limits exist when considering SHTs. To prevent artifacts, maps are often convolved with a beam (similar to gaussian smoothing). Of the baseline methods included, the CNILC follows this paradigm. Thus, the output of that method is convolved and needs to be deconvolved. The cmbNNCS method does not use beam convolution.

As noted in Sec. 4.3, beam convolution can be helpful for analysis. These convolutions are performed using simple Gaussian beams from the healpy library.

Furthermore, it is convention to disregard the lowest two multipole moments, $\ell = 0, 1$. Proper estimation of these values is difficult for a multitude of reasons, from instrumental to fundamental. The monopole ($\ell = 0$), which is the mean temperature of the CMB, is better measured with outside data. The dipole ($\ell = 1$) is often used for calibration, but observations are influenced by external effects unrelated to the true CMB signal. Thus, both of these are typically removed for cosmological work.

These stages proceed as follows. Beam effects are handled first. The filter $B_{\text{target}}/B_{\text{source}}$ is made, where each B is a Gaussian beam with the appropriate full-width half-max value. The map is converted to $a_{\ell m}$ space, the beam is applied with healpy's `almxfl` function, and the $a_{\ell m}$'s are converted back to a map. Next, the mask is applied. Last removal of the monopole and dipole is performed with healpy's `remove_dipole()`. This completes production of analysis-ready **common CMB cleaned** maps.

The process of preparing a map for further analysis is simplified for the CMB realization. These maps were produced at high $N_{\text{side}} = 2048$, so they are downgraded and beam convolved ("**Scale/Beam**") in the same way the observation maps were. The same mask is then applied.

With these preparations, it is possible to **compare maps** at the pixel level. For each simulation, we find the difference between the processed prediction and ground-truth maps and use that difference with all the error metrics listed. Similarly to preprocessing and postprocessing the maps in the baseline stages, calculations are parallelized using `multiprocessing`.

Another set of stages perform analysis at the power spectrum level. Like in the pixel domain, we compare ground truth and prediction. Here, we also compare to a reference distribution of theory power spectra. Ideally, the distribution of spectra would reflect the full distribution of cosmological parameters. As a surrogate, we use the theory power spectra of the training data set as a representative sample. These bands are displayed in output figures and only used for reference.

We then calculate power spectra for each prediction. Executors were developed to compute similar statistics as for error in pixel space, but they were found to be misleading. While predictions from both methods deviate from ideal C_ℓ values around similar multipoles, one method increases without a bound (CNILC), while the other approaches zero (cmbNNCS). Metrics obscure this when the error is bounded in just one case.

Finally, some stages produce presentation figures and tables. Many of these are placed within the top-level scripts for individual baselines. There are also a few which gather the reports output by individual baseline methods and produce single output. Any executor which produces an output which can not be used for subsequent stages is prepended with a digit instead of a letter to indicate this distinction.

E.2. More results

More results are presented on the following pages. To show model response to different simulations at the map level, refer to Fig. 10. Similarly, Fig. 11 shows the power spectra for these predictions. In both cases, those are from the first four test split instances. Note that while there are minor differences, each model produces consistent results. Last, see Fig. 12 for difference maps at various beam convolutions.

F. Figures

All figures were produced by the authors. We describe the provenance of the data and manner of production briefly. Notebooks, where referenced, are at github.com/CMB-ML/paper_figures-ICCV2025.

Figure 1. Data files from the Planck Collaboration. It is easiest to give the filenames: `LFI_SkyMap_XXX-BPassCorrected_1024_R3.00_full.fits` for either `030`, `044`, or `070` frequencies; `HFI_SkyMap_XXX_2048_R3.01_full.fits` for `100`, `143`, `217`, `353-psb`, `545`, or `857` frequencies. The CMB map is from `COM_CMB_IQU-nilc_2048_ R3.00_full.fits`. All are the `I_STOKES` field.

Notebook: `f1_obs_and_target.ipynb`.

Figure 2. Images reproduce the canonical HEALPix description from [38], building off of the tutorial at [113].

Notebook: `f2_HEALPix.ipynb`.

Figure 3. Component maps are from PySM3. Provenance details available at pysm3.readthedocs.io/en/latest/models.html.

Notebook: `f3_components.ipynb`

Figure 4. Data is from the 100 GHz map listed for Figure 1, using the `II_COV` field.

Notebook: `f4_planck_variance.ipynb`

Figure 5. Data is from the maps listed for Figure 1.

Notebook: `f5_fwhm.ipynb`

Figure 6. Data is output from CMB-ML. Individual figure elements arranged in vector graphics software.

Notebook: `f6_dataset_results.ipynb`

Figure 7. Figure is output from CMB-ML, at the `PostAnalysisPsCompareFigExecutor` stage.

Figure 8. Same as Figure 3.

Figure 9. Manually created in vector graphics software.

Figure 10. Elements of figure are output from CMB-ML, at the `ShowSimsPostIndivExecutor` stage. Manually arranged in vector graphics software.

Figure 11. Same as figure 7.

Figure 12. Data is output from CMB-ML and rendered in the notebook `f6_dataset_results.ipynb`.

G. Release Notes

G.1. License

Code for CMB-ML is released under the MIT license. The derived dataset, CMB-ML-512-1450, is under CC-BY-4.0 license. We acknowledge the use of ESA Planck data obtained from the Planck Legacy Archive. Data is used under terms as described in that archive. The Planck Collaboration and ESA bear no responsibility for CMB-ML data.

Datasets are stored as a series of tarballs for each instance for convenient access. A single file contains all links required, along with MD5 hashes. Scripts are provided which automatically download the files, verify hashes, and extract contents.

G.2. Datasets

The only dataset suited for benchmark results is CMB-ML-512-1450, which is at a resolution of $N_{\text{side}} = 512$ (as described in the main text). When downloaded, the dataset is approximately 360 GB. An additional 700 GB of data are created when producing the dataset.

We recognize the need for fast debugging. Two small datasets are available at resolutions of $N_{\text{side}} = 32$ and 128. The miniscule resolution of 32 is suitable only for testing code. The resolution 128 dataset on which cmbNNCS and PyILC can run, though the results should not be used for publication. At this lower resolution, cmbNNCS must use the UNet5 network, as $128 = 2^7$ and the maps cannot be downscaled eight times. These smaller datasets may change without notification and should not be used for publication.

G.3. Maintenance

Hosting (Code) The repository at github.com/CMB-ML/cmb-ml is under active use and development. For reproducibility, see github.com/CMB-ML/cmb-ml/releases/tag/v0.1.1. The DOI 10.5281/zenodo.16510258 matches commit `5018b05`, in the `archive-iccv2025` branch. During review, reviewers were provided with access to the blinded repository (now) at github.com/CMB-ML/cmb-ml.

Hosting (Dataset) The CMB-ML-512-1450 dataset is hosted at utdallas.box.com/v/cmb-ml-512-1450. We recommend using the automated scripts in the repository.

Erratum Issues found in the dataset are noted at github.com/CMB-ML/cmb-ml in the main README.

Maintenance Datasets are versioned; any mistakes found will be corrected for future versions, but the original version will remain for a period of five years. Please watch the repository for updates or contact the first author for more information.

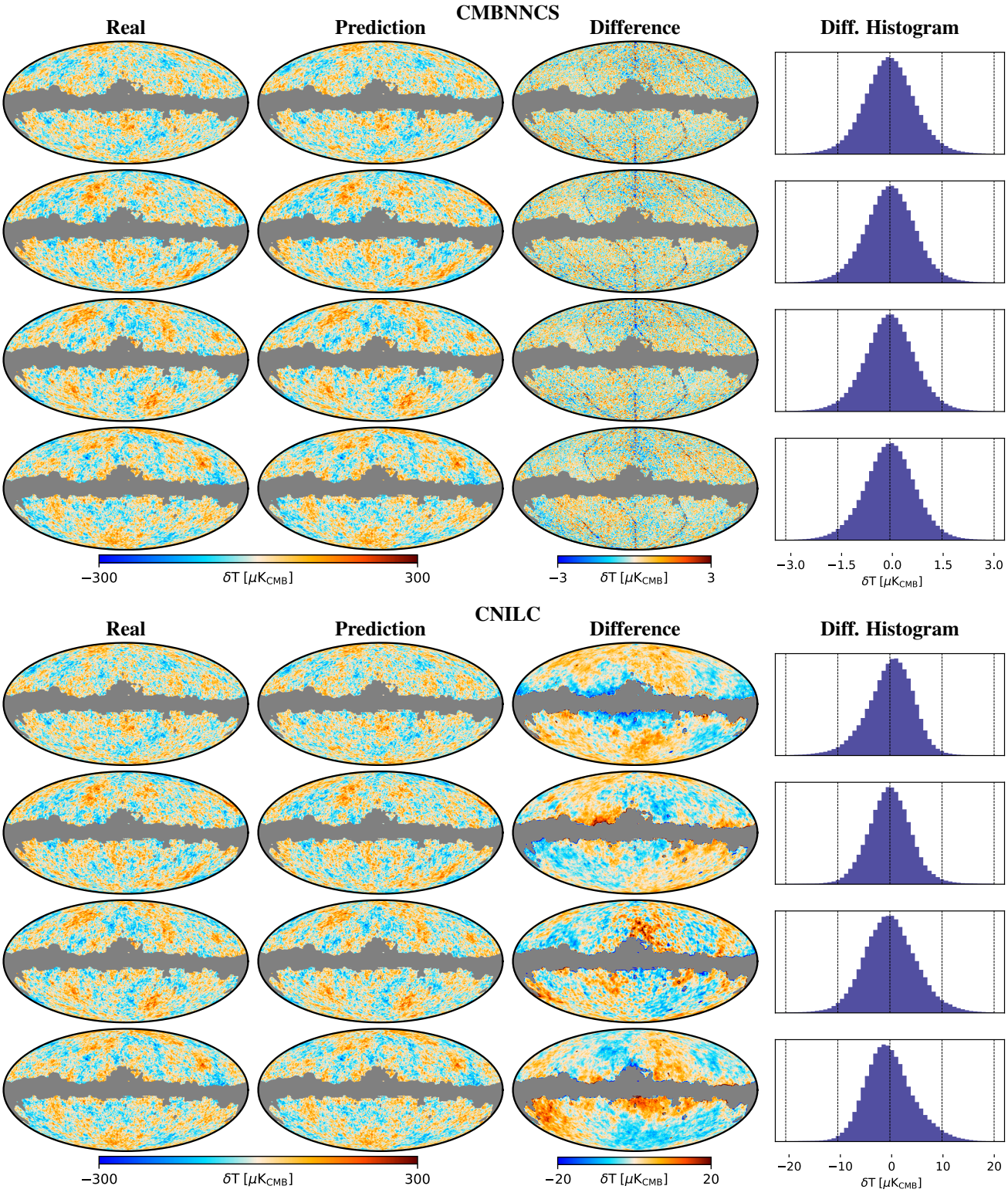

Figure 10. Results for the first four test simulations (0000–0003) using cmbNNCS (top rows) and CNILC (bottom rows). All realizations and predictions are smoothed to $60'$.

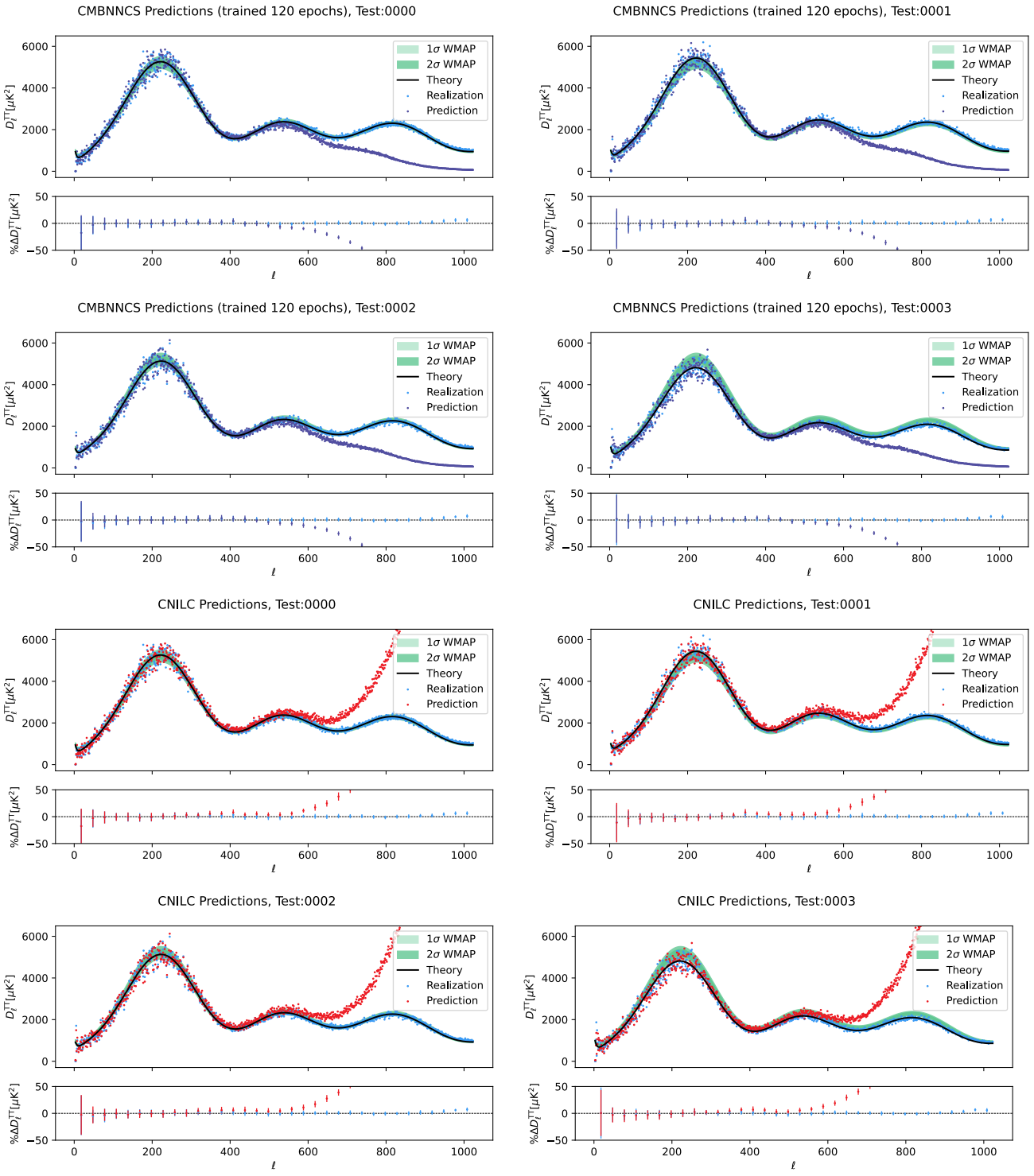

Figure 11. Power spectra results for the first four test simulations (0000−0003) using cmbNNCS (top rows) and CNILC (bottom rows). The realization spectrum refers to the spectrum of the underlying CMB signal, processed in a similar manner as the predictions. Recalling that each simulation has a CMB signal with a unique theory power spectrum, it can be observed that methods have error modes independent of the spectrum.

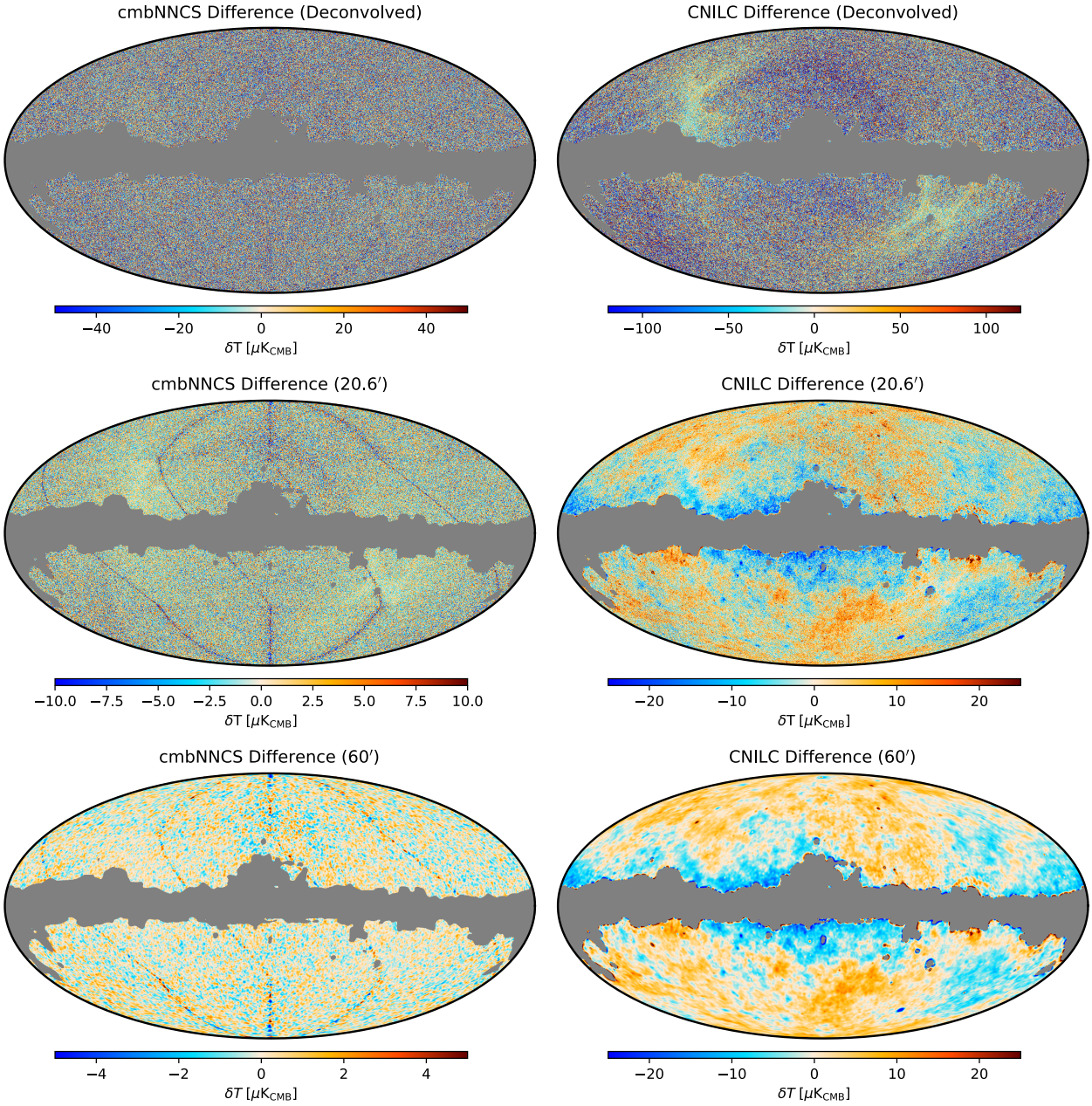

Figure 12. Detailed view of differences between realizations and predictions, with different levels of smoothing. All are for the first simulation in the Test split. Note that cmbNNCS and CNILC have different scale bars. **Left**: The cmbNNCS method clearly shows discontinuities due to the top-level pixel rearrangement. It is also possible to see the effect of residuals at $20.6'$ smoothing. At $60'$ smoothing, the only recognizable source of error is discontinuity. **Right**: The CNILC method shows greater sensitivity to residuals, especially when fully deconvolved. When smoothed, errors due to mask boundaries, point sources, and diffuse foregrounds are visible.