Journal of Cosmology
and Astroparticle
Physics

PAPER • OPEN ACCESS

Cosmic topology. Part IVa. Classification of manifolds using machine learning: a case study with small toroidal universes

To cite this article: Andrius Tamosiunas *et al* JCAP09(2024)057

View the article online for updates and enhancements.

You may also like

- Cosmic topology. Part IIIb. Eigenmodes and correlation matrices of spin-2 perturbations in orientable Euclidean manifolds
  Amirhossein Samandar, Javier Carrón Duque, Craig J. Copi et al.

- Fractality in cosmic topology models with spectral action gravity
  Pedro Guicardi and Matilde Marcolli

- Erratum: Cosmic topology. Part I. Limits on orientable Euclidean manifolds from circle searches
  Deyan P. Mihaylov, Yashar Akrami, Craig J. Copi et al.

This content was downloaded from IP address 24.210.196.49 on 26/10/2025 at 16:55

Journal of Cosmology and Astroparticle Physics
An IOP and SISSA journal

Received: *April 10, 2024*
Revised: *August 10, 2024*
Accepted: *August 20, 2024*
Published: *September 25, 2024*

Cosmic topology. Part IVa. Classification of manifolds using machine learning: a case study with small toroidal universes

The COMPACT collaboration

Andrius Tamosiunas,[a] **Fernando Cornet-Gomez**,[a] **Yashar Akrami**,[b,a,c] **Stefano Anselmi**,[d,e,f] **Javier Carrón Duque**,[b] **Craig J. Copi**,[a] **Johannes R. Eskilt**,[g,c] **Özenç Güngör**,[a] **Andrew H. Jaffe**,[c] **Arthur Kosowsky**,[h] **Mikel Martin Barandiaran**,[b] **James B. Mertens**,[a] **Deyan P. Mihaylov**,[a] **Thiago S. Pereira**,[i] **Samanta Saha**,[a] **Amirhossein Samandar**,[a] **Glenn D. Starkman**,[a] **Quinn Taylor**[a] and **Valeri Vardanyan**[j]

[a]*CERCA/ISO, Department of Physics, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, Ohio 44106, U.S.A.*

[b]*Instituto de Física Teórica (IFT) UAM-CSIC, Campus de Cantoblanco UAM, C/ Nicolás Cabrera 13–15, Madrid 28049, Spain*

[c]*Astrophysics Group & Imperial Centre for Inference and Cosmology, Department of Physics, Imperial College London, Blackett Laboratory, Prince Consort Road, London SW7 2AZ, United Kingdom*

[d]*INFN, Sezione di Padova, via Marzolo 8, Padova I-35131, Italy*

[e]*Dipartimento di Fisica e Astronomia "G. Galilei", Università degli Studi di Padova, via Marzolo 8, Padova I-35131, Italy*

[f]*Laboratoire Univers et Théories, Observatoire de Paris, Université PSL, Université Paris Cité, CNRS, Meudon F-92190, France*

[g]*Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029 Blindern, Oslo N-0315, Norway*

© 2024 The Author(s). Published by IOP Publishing Ltd on behalf of Sissa Medialab. Original content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

https://doi.org/10.1088/1475-7516/2024/09/057

[h]Department of Physics and Astronomy, University of Pittsburgh,
Pittsburgh, Pennsylvania 15260, U.S.A.
[i]Departamento de Física, Universidade Estadual de Londrina,
Rod. Celso Garcia Cid, Km 380, Londrina 86057-970, Paraná, Brazil
[j]Kavli Institute for the Physics and Mathematics of the Universe (WPI), UTIAS,
The University of Tokyo,
Chiba 277-8583, Japan

E-mail: andrius.tamosiunas@case.edu, fernando.cornetgomez@case.edu, yashar.akrami@csic.es, stefano.anselmi@pd.infn.it, javier.carron@csic.es, craig.copi@case.edu, j.r.eskilt@astro.uio.no, ozenc.gungor@case.edu, a.jaffe@imperial.ac.uk, kosowsky@pitt.edu, mikel.martin@uam.es, james.mertens@case.edu, deyan.mihaylov@case.edu, tspereira@uel.br, samanta.saha@case.edu, amirhossein.samandar@case.edu, glenn.starkman@case.edu, qxt42@case.edu, valeri.vardanyan@ipmu.jp

Abstract: Non-trivial spatial topology of the Universe may give rise to potentially measurable signatures in the cosmic microwave background. We explore different machine learning approaches to classify harmonic-space realizations of the microwave background in the test case of Euclidean E_1 topology (the 3-torus) with a cubic fundamental domain of a size scale significantly smaller than the diameter of the last scattering surface. This is the first step toward developing a machine learning approach to classification of cosmic topology and likelihood-free inference of topological parameters. Different machine learning approaches are capable of classifying the harmonic-space realizations with accuracy greater than 99% if the topology scale is half of the diameter of the last-scattering surface and orientation of the topology is known. For distinguishing random rotations of these sky realizations from realizations of the covering space, the extreme gradient boosting classifier algorithm performs best with an accuracy of 88%. Slightly lower accuracies of 83% to 87% are obtained with the random forest classifier along with one- and two-dimensional convolutional neural networks. The techniques presented here can also accurately classify non-rotated cubic E_1 topology realizations with a topology scale slightly larger than the diameter of the last-scattering surface, if enough training data are provided. While information compressing methods like most machine learning approaches cannot exceed the statistical power of a likelihood-based approach that captures all available information, they potentially offer a computationally cheaper alternative. A principle challenge appears to be accounting for arbitrary orientations of a given topology, although this is also a significant hurdle for likelihood-based approaches.

Keywords: CMBR theory, Machine learning

ArXiv ePrint: 2404.01236

Contents

JCAP09(2024)057

1 Introduction

Einstein's general theory of relativity (GR) in combination with cosmological observations can be used to constrain the average local geometry of the Universe [1, 2]. A related but separate question is that of the global cosmic topology (see, e.g., ref. [3]). A universe with non-trivial topology would possess a number of features generally not considered in the standard model of cosmology. As an example, on a manifold with non-trivial topology, any point would have spacelike curves that could not be continuously deformed to a point [4, 5]. If the length of the shortest such curve through the observer were short enough, this would result in an observer detecting a multitude of copies (*clone images*) of astronomical sources [6–8]. For instance, light from a far-away galaxy would reach an observer via multiple paths, resulting in multiple observed images of the mentioned galaxy. Similarly, if the length of the shortest closed loop through an observer were sufficiently less than the diameter of the last-scattering surface (LSS), an observer would detect the so-called circles-in-the-sky effect, referring to matched patterns in the cosmic microwave background (CMB) temperature fluctuations around pairs of circles on the celestial sphere [9–12]. Even if the topology scale were larger than the diameter of the LSS, there might still be information encoded in the CMB fluctuations, as discussed in refs. [13–16]. Several observational searches for these signatures of non-trivial topology have been performed, including Wilkinson Microwave Anisotropy Probe (WMAP) and *Planck* searches for the circles-in-the-sky effect [17–23]. Similarly, a Bayesian search that relies on comparing the pixel-pixel correlations in the observed CMB temperature map to those expected to be induced in manifolds with non-trivial topology was performed using *Planck* survey data [22–24]. The outlined observational efforts, as of yet, have not detected any evidence for matched circle pairs in the CMB data. This, combined with the measurements of the local geometry of the Universe, allows us to constrain the set of allowed topology classes. Specifically, if the Universe is spatially flat, the set of allowed topologies consists of 18 classes (often denoted as E_1–E_{17} for the Euclidean non-trivial topology classes and E_{18} for the trivial topology class, i.e., the covering space). Each of the 18 classes can then have up to 6 free parameters, corresponding approximately to the lengths of the sides of the fundamental domain[1] and the angles between them, plus 6 additional possible degrees of freedom corresponding to the position and orientation of the observer [4, 5, 25–27]. The key challenge from an observational perspective then is to distinguish the effects of these 18 topology classes on the CMB anisotropies.

The CMB temperature fluctuations ΔT can be expanded in terms of spherical harmonics, such that

$$\Delta T(\theta, \phi) = \sum_{\ell,m} a_{\ell m} Y_{\ell m}(\theta, \phi), \tag{1.1}$$

where θ and ϕ are the polar and azimuthal angles on the sky, $Y_{\ell m}$ are the spherical harmonics, and $a_{\ell m}$ are the complex harmonic-space coefficients with ℓ and m the multipole and azimuthal numbers, respectively. Having decomposed the observed CMB temperature fluctuation pattern in spherical harmonics, we note that, if the sky is the result of a Gaussian process, then all the information about the temperature anisotropies is captured by the 2-point angular correlation matrix $C_{\ell m \ell' m'} = \langle a_{\ell m} a^*_{\ell' m'} \rangle$. In the case of trivial topology, which is isotropic, this correlation matrix is diagonal, $C_{\ell m \ell' m'} = C_\ell \delta^{(\mathrm{K})}_{\ell \ell'} \delta^{(\mathrm{K})}_{m m'}$, where C_ℓ is the angular power spectrum and $\delta^{(\mathrm{K})}_{ij}$ is the Kronecker delta. However, this is not the case when considering non-trivial topology — the assumption of statistical isotropy is broken, resulting in non-zero off-diagonal terms in the correlation matrix.

Given that these non-diagonal correlations are a key signature of non-trival topology, significant work has gone into understanding and classifying such features, e.g., in refs. [14–16, 26]. One can also quantify the information related to non-trivial topologies in the form of Kullback-Leibler (KL) divergence for given correlation matrices. Previous work [14–16] has demonstrated that even for a non-trivial topology with a scale slightly larger than the diameter of the LSS, the KL divergence is larger than unity, i.e., a given CMB realization from a non-trivial topology could still, in principle, be distinguished from the one from the trivial topology. However, even though we can demonstrate that the information related to non-trivial topologies exists and is encoded in the correlations between the different

[1]Technically, we should refer to the Dirichlet domain, which is a specific fundamental domain. For details see ref. [16].

JCAP09(2024)057

harmonic coefficients, deducing the topology classes based on an individual $a_{\ell m}$ realization is a complicated inverse problem. This problem cannot generally be approached analytically and other techniques, such as those based on artificial intelligence, are likely required.

In this work we present a set of machine learning algorithms to distinguish and classify CMB realizations from different topologies in harmonic-space ($a_{\ell m}$ coefficients) and correlation space ($a_{\ell m}a^*_{\ell' m'}$ for each realization). While the problem of distinguishing different topologies using harmonic-space realizations or CMB maps has been considered before (e.g., in refs. [28, 29]), to the best of our knowledge this is the first application of machine learning techniques to this problem. We start by numerically generating a set of harmonic-space realizations for the cubic E_1 topology of different sizes, where by size we refer to the length of the fundamental domain in units of the diameter of the LSS. Specifically, we generate cubic E_1 realizations in four size classes, $L \in \{0.05, 0.1, 0.5, \infty\} \times L_{\mathrm{LSS}}$, where L_{LSS} is the diameter of the LSS and the last class corresponds to the covering space. To classify the realizations, we test the following machine learning algorithms: random forests, extreme gradient boosting classifier (`XGBoost`), one-dimensional (1D) convolutional neural networks (CNNs), two-dimensional (2D) CNNs, and complex 2D CNNs. In each case we obtain the results for the algorithm trained and tested on randomly Wigner-rotated and non-rotated realizations. Additionally, we present a small set of results for harmonic-space realizations with the length scale of $L \gtrsim L_{\mathrm{LSS}}$ in order to set preliminary expectations for the effectiveness of machine learning for classifying large universes with non-trivial topologies.

To be clear, E_1 with a fundamental domain of small size (i.e., $L \lesssim L_{\mathrm{LSS}}$) is observationally excluded [15, 17–23]. However, our objective in this paper is not to extend current observational limits to larger fundamental domains, but to begin developing the machine learning techniques that may ultimately be necessary to do so, and to identify the challenges that must be addressed. To this end, we do not make explicit use of the circles-in-the-sky signature, which does not extend to domains larger than the LSS. We also note that an analogous problem has been explored using a Bayesian likelihood approach (see, e.g., ref. [28]), and while we do not explore such methods here, we find it valuable to compare the results obtained in this work with the corresponding results from the likelihood-based approaches. We compare the two different types of methods in section 5.

We demonstrate that the machine learning methods which we have tested are all effective at classifying harmonic-space realizations with $> 99\%$ accuracy for realizations that have coordinate axes aligned with the edges of the cubic fundamental domain of the torus. We find that the results depend significantly on whether the orientation of the torus is already known. For the realistic case when realizations have been randomly rotated, we find that topologies with larger fundamental domains, e.g., with $L = 0.5 \times L_{\mathrm{LSS}}$, are more challenging to distinguish from the covering space. Finally, we show that our methods are also effective when we classify non-rotated $a_{\ell m}$ E_1 realizations with the size scale slightly larger than the diameter of the LSS, i.e., for $L = 1.01 \times L_{\mathrm{LSS}}$, as long as the dataset is sufficiently large. We identify this difficulty in dealing with coordinate rotations as a key challenge to be addressed in future work.

The layout of the paper is as follows. In section 2 we discuss the theoretical background, the process of generating the E_1 topology realizations, and the general features of the resulting datasets. Section 3 outlines the machine learning algorithms we employ. Section 4.1

summarizes the results obtained for realizations with the size scale L significantly smaller than that of the diameter of the LSS. The results for realizations with the size scale $L \gtrsim L_{\mathrm{LSS}}$ are discussed in section 4.2. Future work, implications for observational topology searches, and a comparison with likelihood-based approaches are discussed in section 5. Appendix A contains extra information on the training procedures as well as the settings used to train the algorithms. The methodology for generating the realizations with $L \gtrsim L_{\mathrm{LSS}}$ is discussed in appendix B.

2 The dataset

2.1 Properties of the E_1 topology

To generate simulated realizations of the CMB on a manifold, we require the eigenmodes of the Laplacian on that manifold.[2] The eigenmodes for the Euclidean topologies have been studied extensively (e.g., see ref. [16]). The E_1 eigenmodes $\Upsilon^{E_1}_{\boldsymbol{k}}$ are the subset of the E_{18} eigenmodes that respect the E_1 symmetries,

$$\Upsilon^{E_1}_{\boldsymbol{k}}\left(g^{E_1}_{A_j}\boldsymbol{x}\right) = \Upsilon^{E_1}_{\boldsymbol{k}}(\boldsymbol{x}), \quad j = 1, 2, 3\,. \tag{2.1}$$

Here $g^{E_1}_{A_j}$ is a generator for the E_1 topology (described in detail in sections 2 and 3.1 of ref. [16]) i.e., a translation. Since in this paper we will confine our attention to cubic E_1 manifolds, these three translations are in orthogonal directions, which we can take to be aligned along the three coordinate axes, and are of equal length, L. L is therefore also the side length of the cubic periodic box that is the most convenient fundamental domain for cubic E_1. The symmetry condition (2.1) causes the discretization of the allowed wave vectors. In a cubic periodic box of side length L, this corresponds to

$$\boldsymbol{k}_{\boldsymbol{n}} = \frac{2\pi}{L}(n_1, n_2, n_3), \tag{2.2}$$

where the wave vectors $\boldsymbol{k}_{\boldsymbol{n}}$ are labeled by a triplet of integers $\boldsymbol{n} = (n_1, n_2, n_3)$. The E_1 eigenmodes are therefore given by

$$\Upsilon^{E_1}_{\boldsymbol{k}_{\boldsymbol{n}}}(\boldsymbol{x}) = e^{i\boldsymbol{k}_{\boldsymbol{n}}\cdot(\boldsymbol{x}-\boldsymbol{x}_0)}, \quad \text{for } \boldsymbol{n} \in \mathcal{N}^{E_1}, \tag{2.3}$$

where $\mathcal{N}^{E_1} \equiv \{(n_1, n_2, n_3) \mid n_i \in \mathbb{Z}\} \setminus (0, 0, 0)$.

Given the discretization of the allowed wave vectors due to the E_1 symmetries, the spherical-harmonic coefficients can be obtained by

$$a_{\ell m} = \frac{4\pi}{L^3} \sum_{\boldsymbol{n}\in\mathcal{N}^{E_1}} \delta^{\mathcal{R}}_{\boldsymbol{k}_{\boldsymbol{n}}} \xi^{E_1;\hat{\boldsymbol{k}}_{\boldsymbol{n}}}_{k_{\boldsymbol{n}}\ell m} \Delta_\ell\left(k_{\boldsymbol{n}}\right), \tag{2.4}$$

where $\xi^{E_1;\hat{\boldsymbol{k}}_{\boldsymbol{n}}}_{k_{\boldsymbol{n}}\ell m} \equiv e^{-i\boldsymbol{k}_{\boldsymbol{n}}\cdot\boldsymbol{x}_0} i^\ell Y^*_{\ell m}\left(\hat{\boldsymbol{k}}_{\boldsymbol{n}}\right)$ and $\delta^{\mathcal{R}}_{\boldsymbol{k}_{\boldsymbol{n}}}$ (primordial density fluctuations) are Gaussian, random, statistically independent variables of zero mean with variances determined by the primordial power spectrum $\mathcal{P}^{\mathcal{R}}$,

$$\left\langle \delta^{\mathcal{R}}_{\boldsymbol{k}_{\boldsymbol{n}}} \delta^{\mathcal{R}*}_{\boldsymbol{k}'_{\boldsymbol{n}}} \right\rangle = \mathcal{P}^{\mathcal{R}}(k_{\boldsymbol{n}})\delta^{(\mathrm{K})}_{\boldsymbol{k}_{\boldsymbol{n}}\boldsymbol{k}'_{\boldsymbol{n}}}\,. \tag{2.5}$$

[2] In principle, both the scalar (spin-zero) and tensor (massless spin-two) eigenmodes, but in this paper we confine ourselves to scalar perturbations.

Note that $\Delta_\ell(k_{\boldsymbol{n}})$ in is the corresponding transfer function. The harmonic-space covariance matrix in the E_1 topology then has the form

$$C_{\ell m \ell' m'} = \frac{(4\pi)^2}{L^3} \sum_{\boldsymbol{n} \in \mathcal{N}^{E_1}} \Delta_\ell\left(k_{\boldsymbol{n}}\right) \Delta^*_{\ell'}\left(k_{\boldsymbol{n}}\right) \frac{2\pi^2 \mathcal{P}^{\mathcal{R}}\left(k_{\boldsymbol{n}}\right)}{k^3_{\boldsymbol{n}}} \xi^{E_1;\hat{\boldsymbol{k}}_{\boldsymbol{n}}}_{k_{\boldsymbol{n}} \ell m} \xi^{E_1;\hat{\boldsymbol{k}}_{\boldsymbol{n}}*}_{k_{\boldsymbol{n}} \ell' m'}. \tag{2.6}$$

In the rest of this work we will refer to the covariance matrix generated from individual $a_{\ell m}$ realizations as $\mathcal{C}_{\ell m \ell' m'} = a_{\ell m} a^*_{\ell' m'}$ as opposed to the analytic covariance matrix given in.

2.2 Generating $a_{\ell m}$ realizations

The $a_{\ell m}$ realizations can be generated by numerically evaluating (2.4) (following the approach previously applied in ref. [16]). In summary, we sum over all the Fourier modes that contribute to each spherical harmonic coefficient by generating the primordial density fluctuations, i.e., the $\delta^{\mathcal{R}}_{\boldsymbol{k}_{\boldsymbol{n}}}$, for each wave vector $\boldsymbol{k}_{\boldsymbol{n}}$. The transfer function along with the primordial power spectrum are obtained by using `CAMB` [30, 31] with *Planck* 2018 best-fit ΛCDM cosmological parameters [32]. Theoretically, we would need to perform the sums in the equations over an infinite set of wave numbers, however, in practice, we can obtain accurate results by choosing a cutoff value $k_{\max}$, at which we stop the summation. We choose $k_{\max}$ such that wave vectors with $k \leq k_{\max}(\ell)$ would contribute to at least 99% of the ΛCDM angular power spectrum, $C^{\Lambda\mathrm{CDM}}_\ell$, as generated by `CAMB`.

We generate the $a_{\ell m}$ realizations for four classes of the E_1 topology. We focus on small cubic E_1 manifolds and investigate the accuracy of different machine learning techniques in classifying the manifolds and distinguishing between them and the covering space. The chosen four classes are $L \in \{0.05, 0.1, 0.5, \infty\} \times L_{\mathrm{LSS}}$, where the last one corresponds to the $L \to \infty$ limit, i.e., the covering space. We expect that classifying realizations with $L > L_{\mathrm{LSS}}$ will be significantly more difficult. This is based on the fact that an observer in a universe of such size would not observe matched circle pairs in the CMB, so there are no (nearly) perfectly matched pixels (i.e., on a pixelated temperature map).

Similarly, the KL divergence and the signal-to-noise ratio statistic (introduced in section 5.3 of ref. [16]) for manifolds with $L > L_{\mathrm{LSS}}$ quickly approach unity [16]. This indicates that classifying the realizations with $L > L_{\mathrm{LSS}}$ will likely require machine learning methods that are different from those outlined in this work. We plan to explore this in future work. Nevertheless, examining $L \leq 1$ gives us an opportunity to explore the specific challenges we are likely to face in applying machine learning classification methods to cosmic topology.

In each case studied in the present work, the $a_{\ell m}$ realizations are generated with the multipoles in the range $\ell \in [2, 100]$. However, for different machine learning algorithms we use different subsets of this dataset, depending on the memory constraints (for full details, see appendix A). In order to generate the covering space realizations, we use the `synalm` function available as part of the `healpy`[3] package [33, 34]. We use the *Planck* 2018 best-fit cosmological parameters (the same parameters as those used for computing the transfer function of the E_1 realizations).

An important nuance when it comes to training machine learning models is data ordering. By this we refer to the order of the $a_{\ell m}$ entries as stored in the data array. The order

[3]Available at http://healpix.sourceforge.net.

JCAP09(2024)057

matters as it can determine whether certain features of the data appear locally or non-locally in the data vector. This is especially important for convolutional neural networks, where certain non-local features might not be captured depending on the size of the filters and the data arrays. To investigate such effects we consider two types of data orderings, (m, ℓ) and (ℓ, m). The (m, ℓ) ordering refers to the default `HEALPix` [33, 34] ordering, i.e., $a_{\ell m} = \{a_{00}, a_{10}, a_{20}, \ldots, a_{\ell_{\max}0}, a_{11}, a_{21}, \ldots, a_{\ell_{\max}\ell_{\max}}\}$, while the (ℓ, m) ordering is $a_{\ell m} = \{a_{00}, a_{10}, a_{11}, a_{20}, a_{21}, \ldots, a_{\ell_{\max}\ell_{\max}}\}$. We denote a specific entry in a particular data vector by $\mathcal{I}_{\ell,m}$ for (ℓ, m) ordered dataset and by $\mathcal{I}_{m,\ell}$ for (m, ℓ) ordered dataset, where

$$\mathcal{I}_{\ell,m} = \ell(\ell+1)/2 + m, \qquad \mathcal{I}_{m,\ell} = m(2\ell_{\max} + 1 - m)/2 + \ell. \tag{2.7}$$

In order to train a 2D CNN, we require data formatted in 2D. We generate the needed dataset by evaluating $\mathcal{C}_{\ell m \ell' m'}$ for each set of $a_{\ell m}$. The four-dimensional matrix is then flattened, such that for each 2D (ℓ, ℓ') "block" the corresponding (m, m') values are shown inside the block, resulting in the characteristic *checkerboard* pattern seen (partially) in figure 3 and (more clearly) in figure 4. We generate the 2D data in the same two ordering schemes as are used for the $a_{\ell m}$ realizations. The $\mathcal{C}_{\ell m \ell' m'}$ data is generated for $\ell \in [2, 20]$. For larger $\ell_{\max}$, the dataset quickly becomes unmanageable. We expect this data format to be especially useful, as previous work indicates that for the larger $(L > L_{\mathrm{LSS}})$ manifolds that we eventually want to be able to classify the key features of non-trivial topologies for scalar temperature fluctuations are stored in the correlations between relatively low multipoles.

An important choice when generating the CMB realizations is the orientation of the coordinate system. We will find that this choice strongly affects the results of the machine learning classification. To take this into consideration, we generate two sets of realizations, one where the $a_{\ell m}$ are unrotated compared to the natural frame of the cubic E_1 domain, and one where each realization is randomly rotated (i.e., by applying the Wigner D-matrices $D^{\ell}_{m\bar{m}}(\theta_0, \phi_0, \psi_0)$). In the unrotated case, the default orientation is such that the closest clone images are in the x, y, and z directions, i.e., the observer's coordinate system is aligned along the three coordinate axes (see refs. [16, 35] for a wider discussion of this point).

In order to prepare our training, validation, and test datasets, we numerically generate 40,000 $a_{\ell m}$ realizations, i.e., the same 10,000 realizations per class ordered in $(\ell,\ m)$ and $(m,\ \ell)$ ordering schemes (see section 3 for further details). We also generate an augmented dataset in order to investigate the effects on the classification accuracy of an increased total dataset size and of rotations. This is done by taking each of the aforementioned 40,000 realizations and randomly rotating it, choosing the three Euler angles so that the direction of the z axis is drawn from a uniform distribution on the sky, and the orientation of the x and y axes with respect to that z axis is drawn from a uniform distribution from 0 to 2π. This is repeated 10 times for each of the initial 40,000 realizations, resulting in a total of 400,000 realizations. Different subsets of this augmented dataset are then used to train the different algorithms, as outlined in appendix A.

In addition to the datasets described above, we generate an extra set of $a_{\ell m}$ realizations with $L \gtrsim L_{\mathrm{LSS}}$ by employing an alternative method of Cholesky decomposition (see refs. [16, 36, 37]). The primary motivation for using an alternative method for generating the data here is the fact that numerically evaluating for topology scales of $L \gtrsim L_{\mathrm{LSS}}$ is not viable given the computational resources available to us. An alternative method for generating harmonic-space realizations is that of computing the temperature auto-correlation matrix, for given values of L and $\ell_{\max}$, and then using Cholesky decomposition to factor the correlation matrix into a lower triangular matrix along with its conjugate. This method allows generating harmonic-space realizations corresponding to the topology size scale $L \gtrsim L_{\mathrm{LSS}}$ significantly faster, with the caveat that the maximum multipole value is lower, e.g., $\ell_{\max} \approx 30$. This is not an issue, since we already know [16, 38] that for $L > L_{\mathrm{LSS}}$ most of the information in the CMB for distinguishing compact topologies from the covering space is at $\ell \lesssim 30$.

Specifically, we generate a number of datasets for two topology classes, cubic E_1 topology with $L = 1.01 \times L_{\mathrm{LSS}}$ and the covering space. This particular size scale is chosen as a test case for a manifold that is sufficiently large for there to be no matched circle pairs, but not too large (as we know that for $L \gtrsim 1.1 \times L_{\mathrm{LSS}}$ the KL divergence and the signal-to-noise ratio would be significantly small [16], and hence, an alternative classification approach would likely be required). The realizations are generated with $\ell_{\max} = 30$. In order to investigate how the classification accuracy depends on the dataset size, a total of 6 datasets are generated, ranging in their total number of realizations between 200 and 200,000. As before, we generate two versions of each dataset — an unrotated and a randomly rotated one. In each case we chose to work with the realizations in the (ℓ, m) ordering. Further technical details, as well as a comparison between the realizations obtained by the two outlined methods, are described in appendix B.

2.3 Features of the $a_{\ell m}$ realization data

Here we lay out some of the key features of the generated datasets. Figure 1 shows the mean and the standard deviation for each class in the dataset. For each class real and imaginary parts are shown in $(\ell,\ m)$ ordering. Similarly, figure 2 shows random data samples in (m, ℓ) ordering. The first two even m values are marked in the figure for each class. As expected, we find that larger E_1 realizations, e.g., realizations with $L = 0.5 \times L_{\mathrm{LSS}}$, look nearly identical to the covering space realizations. Meanwhile, smaller realizations, such as $L = 0.05 \times L_{\mathrm{LSS}}$, show clear differences when compared to the trivial topology class. One such feature is the visible suppression of the variance of the imaginary parts for even m values. The same effect is seen even more clearly in figure 1. This feature is more noticeable in small E_1 ($L = \{0.05–0.1\} \times L_{\mathrm{LSS}}$) realizations, less visible for the larger E_1 realizations ($L = 0.5 \times L_{\mathrm{LSS}}$), and not present at all in the covering space realizations. Note that this is a feature that may be specific to cubic E_1 and we do not expect it to necessarily be present in non-cubic, randomly rotated realizations or in other topologies.

Figure 3 shows a selection of randomly chosen $\mathcal{C}_{\ell m \ell' m'}$ realizations, while figure 4 shows the analytic covariance matrices, as defined in, for different sizes of the E_1 topology. The figures all show the absolute values of correlation normalized by the ΛCDM power spectrum, i.e., by $\left(C_\ell^{\Lambda\mathrm{CDM}} C_{\ell'}^{\Lambda\mathrm{CDM}}\right)^{-1/2}$. The key standout feature is the *checkerboard* pattern observed in figure 4, which indicates off-diagonal correlations between different multipoles. The relatively orderly and local distribution of features (i.e., nearby in the $C_{\ell m \ell' m'}$ matrix given the (ℓ, m) ordering) hints at this data format being advantageous over some other ways of representing the data. For example, one could consider training an algorithm directly on the temperature

JCAP09(2024)057

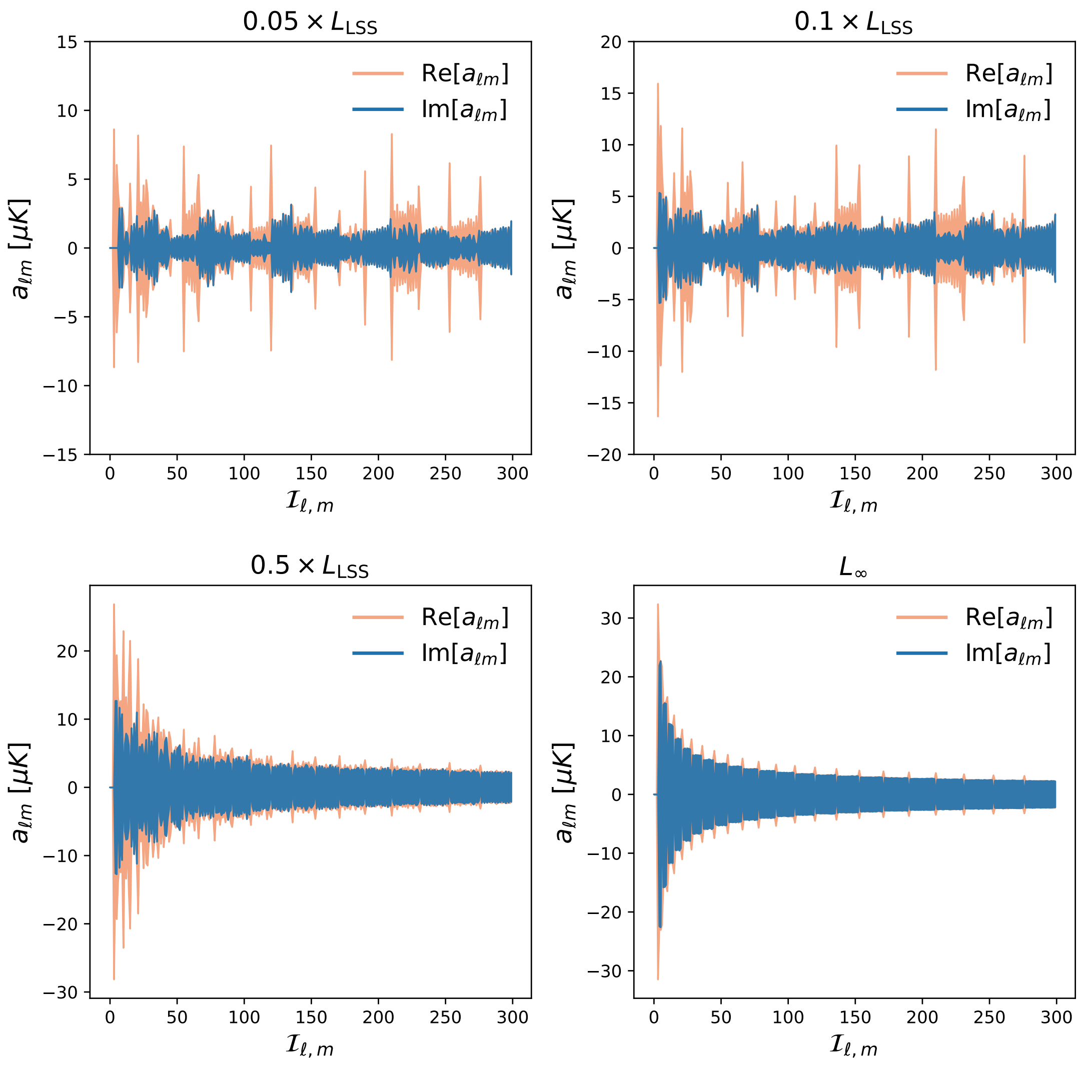

Figure 1. Samples of each dataset class, i.e., $L \in \{0.05, 0.1, 0.5, \infty\} \times L_{\rm LSS}$, in (ℓ, m) ordering for the cubic E_1 topology. For each class, the mean and the standard deviation over the full dataset (10,000 realizations) are plotted and we show the real and imaginary parts of the $a_{\ell m}$ for the first 300 values in order to illustrate the local features of the dataset. The L_∞ class corresponds to the covering space (or trivial topology).

JCAP09(2024)057

fluctuation `HEALPix` maps for each topology class (for instance by using `DeepSphere` [39]). While it is true that for the small topology scales that we are considering here (i.e., $L < L_{\rm LSS}$) the correlations might be clearly represented in the real (or pixel) space, extracting this information is non-trivial. It generally requires specific algorithms (e.g., for finding the matched-circle pairs) and a sufficiently large value of $\ell_{\rm max}$ (generally at least $\ell_{\rm max} \approx 250$–$300$). Extracting this information with machine learning is also complicated by the fact that the correlated pixels generally appear non-locally in the map and are difficult to capture by the convolutional filters. This is especially true in the regime of $L > L_{\rm LSS}$ (which is our ultimate target given the existing negative circle-search results), where we expect correlations to be the clearest in (three-dimensional) Fourier space, and hence, in harmonic space. For these two

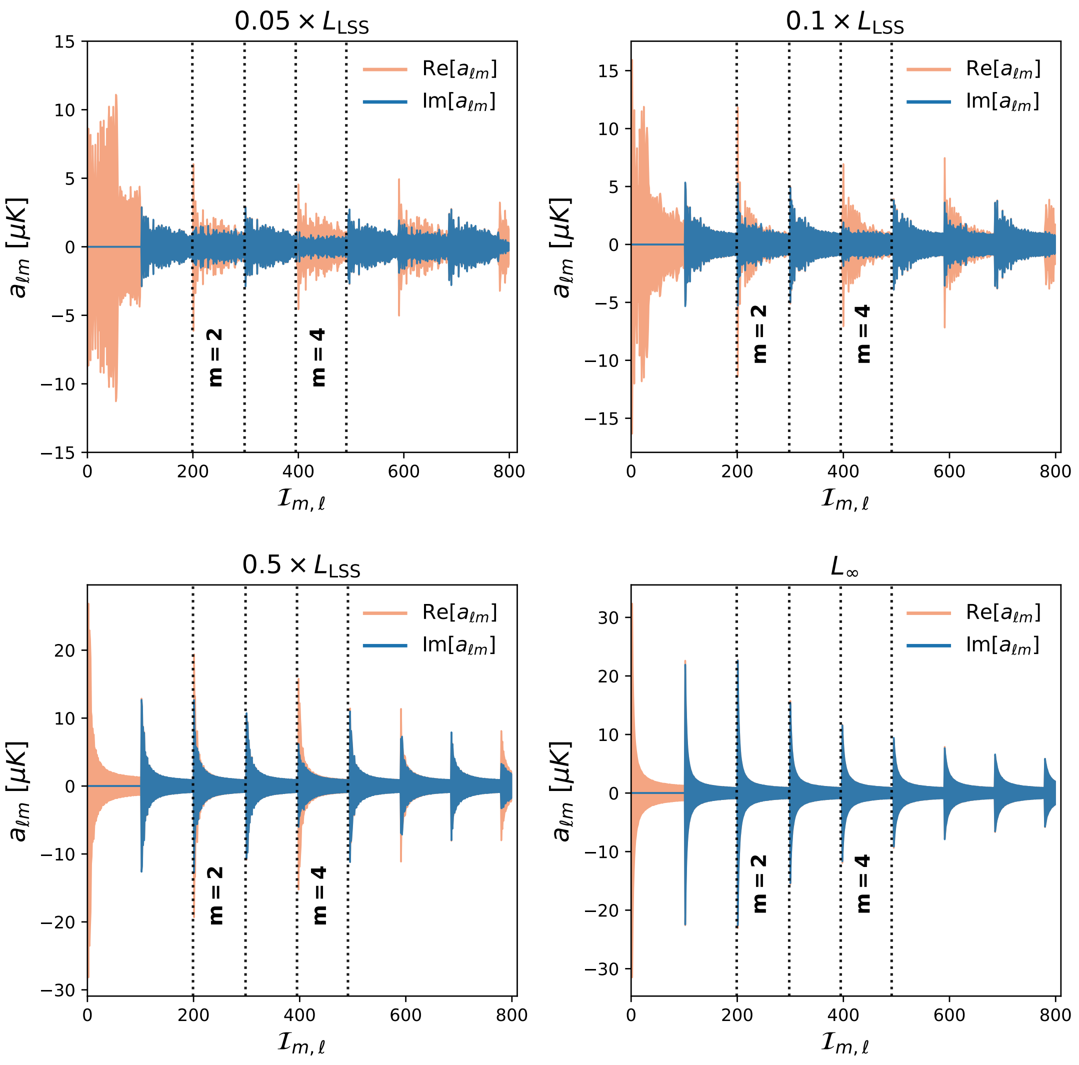

Figure 2. As in figure 1, but for (m, ℓ) ordering. For each sample we show the first several m values. In particular, $m = 2$ and $m = 4$ are marked by dashed vertical lines to demonstrate the suppression of the imaginary values with respect to the corresponding real values.

reasons, we choose to work specifically with harmonic space realizations with the maximum multipole value in the range of $\ell_{\max} = 50$–100 (and $\mathcal{C}_{\ell m \ell' m'}$ realizations with $\ell_{\max} = 20$).

3 Machine learning algorithms

To classify the different realizations of the covering space and the E_1 topology we use a number of different machine learning algorithms. Here we summarize the key features of those algorithms along with the details of the training procedure for each case. Further technical details on the training procedure and the data preparation are provided in appendix A.

3.1 Random forests and extreme gradient boosting classifier trained on $a_{\ell m}$ data

Random forests refers to an ensemble algorithm that is widely used for classification and regression problems. Specifically, it is an ensemble of individual decision trees, the predictions

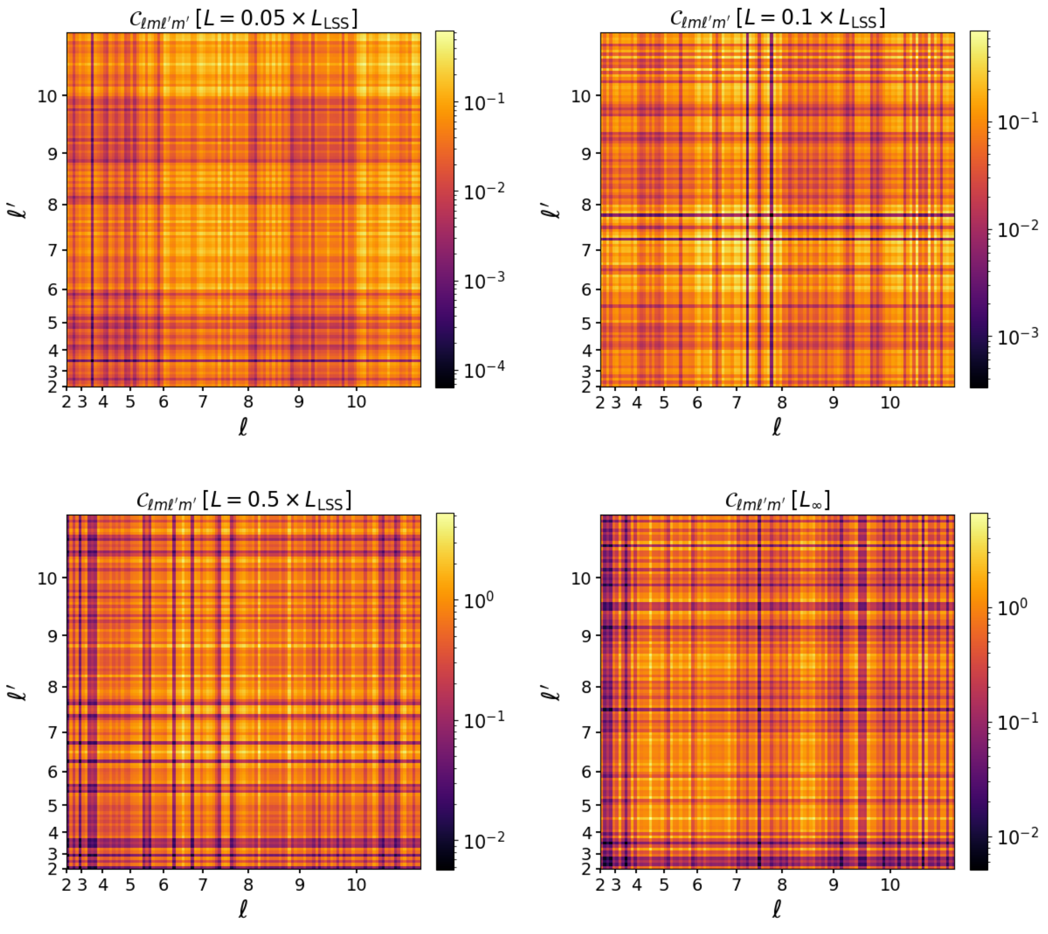

Figure 3. A selection of random samples of the absolute value of $\mathcal{C}_{\ell m \ell' m'}$ (represented as a 2D array) for each class in our dataset. For each (ℓ, ℓ') block the matrix elements show the corresponding (m, m') in increasing order, i.e., $-\ell \leq m \leq \ell$ (see figure 4 for a clearer view of the block structure). Each plot is rescaled by $\left(C_\ell^{\Lambda\rm CDM} C_{\ell'}^{\Lambda\rm CDM}\right)^{-1/2}$, with $C_\ell^{\Lambda\rm CDM}$ the ΛCDM angular power spectrum.

of which are combined using ensemble voting. The main features of the algorithm along with common uses are described in detail in refs. [40–42]. During the training procedure, a random subset of the training data vectors is sampled (with replacement). For a given subset of the data, a decision tree is generated by determining the optimal way to split the data values, such that the classification accuracy is maximized. The optimal splits are evaluated based on a chosen statistic, generally Gini impurity, entropy, or mean squared error for regression tasks. Each tree is constructed until the specified stopping condition is met (e.g., maximum depth of the tree or a minimum number of samples in a leaf node). Once the specified number of trees are generated, their predictions are combined in an ensemble vote. This is generally a majority vote or an average over all predictions.

Random forests correct some of the common issues observed when using decision trees, namely their tendency to overfit when the number of features is high. This is an advantage for our dataset, where the number of features (i.e., $a_{\ell m}$ entries) is comparable to the size of the dataset (i.e., the number of realizations in the training dataset). In addition, it is generally

JCAP09(2024)057

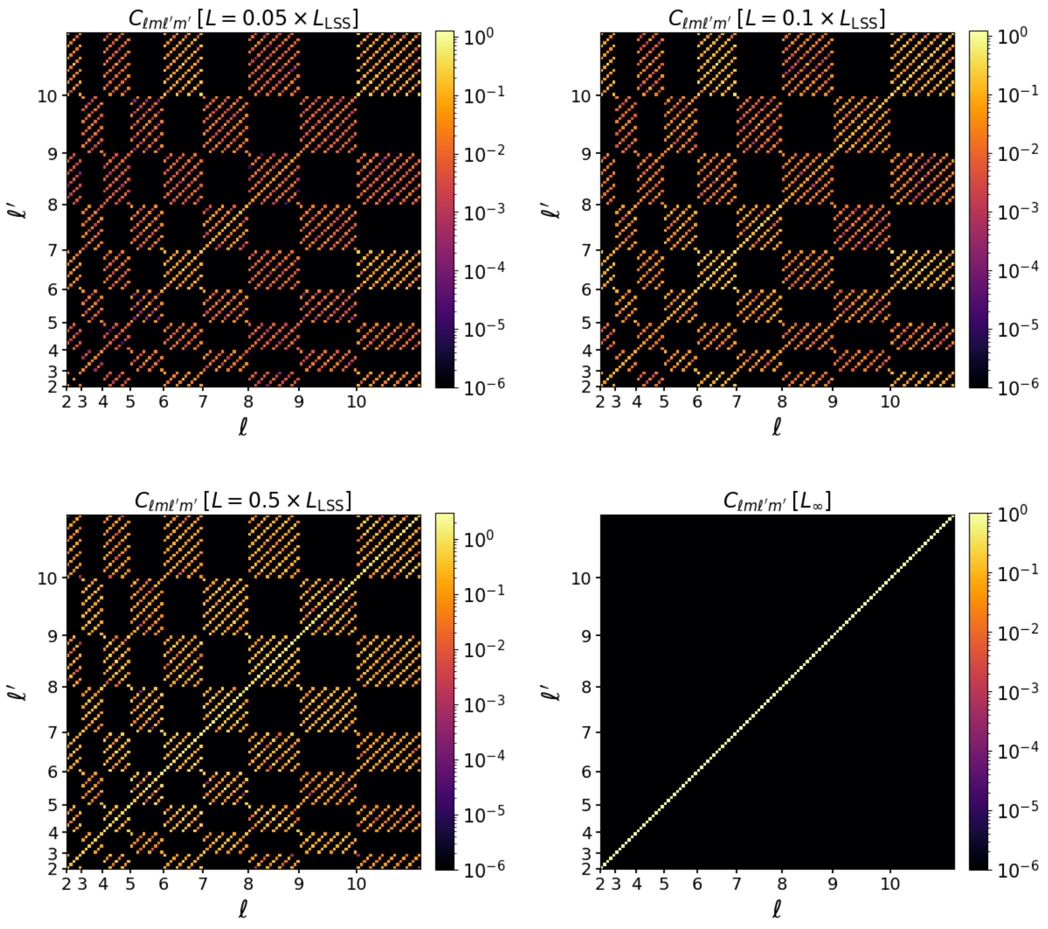

Figure 4. The analytic covariance matrix (2.6) for each class in our dataset. The absolute value is shown in each case, and each covariance matrix is rescaled by $\left(C_\ell^{\Lambda\rm CDM} C_{\ell'}^{\Lambda\rm CDM}\right)^{-1/2}$.

found that multiple uncorrelated models (i.e., random forests) perform significantly better for classification tasks than individual models (i.e., individual decision trees) do. Random forests are also known to be more interpretable than some of the well-known *black box* algorithms, e.g., neural networks. As a concrete example, one can calculate the feature importance for a given feature in our dataset. This refers to a Gini impurity measure of how often a randomly chosen element would be incorrectly classified. For a given node in a random forest classifier, the Gini impurity refers to the uncertainty of that particular node with respect to the target variable. It can be expressed as $G = 1 - \sum_{i=1}^{C} p_i^2$, with C the number of classes and p_i the probability of class i in that particular node. The feature importance is then calculated as the sum of the Gini impurity decrease over all nodes in all decision trees where the feature at hand was used to make a classification decision. For our specific dataset of $a_{\ell m}$ realizations, the feature importance score then tells us which particular values of ℓ and m are most significant when making the classification decision. In other words, the feature importance statistic indicates which angular scales on the CMB sky hold the key information that differentiates non-trivial topology realizations from one another and from the corresponding covering space realizations.

In this work, we train a random forest classifier available from the `scikit-learn` library [43]. The training is performed on both the randomly rotated (100,000 data samples) and the unrotated samples (a dataset of 40,000 samples). For both datasets, we consider $a_{\ell m}$ values for $\ell \in [2, 100]$. The data is split into 80% training and 20% test datasets. The training is performed with 3500 estimators (number of decision tress). For the split criterion, we use Gini impurity. The data samples are prepared by splitting the complex $a_{\ell m}$ into their real and imaginary parts (in (ℓ, m) ordering), which are appended into a single data vector. In addition to the real and imaginary parts of each realization, we consider many other features that can be constructed from the available data, e.g., C_ℓ values, average $a_{\ell m}$ values for a given ℓ, and the eigenvalues of the $\mathcal{C}_{\ell m \ell' m'}$ matrix corresponding to a given realization. We find that the eigenvalues of $\mathcal{C}_{\ell m \ell' m'}$ are particularly informative. This is the case, as, given the structure of the $a_{\ell m} a^*_{\ell' m'}$ matrix for a given realization, all but one of the eigenvalues are zero. The only non-zero eigenvalue corresponds to the square of the absolute value of the $a_{\ell m}$ data vector, i.e., $||a_{\ell m}||^2 \equiv \sum_{\ell m} |a_{\ell m}|^2$, which is a rotation-invariant quantity. Hence, in order to improve the classification results, we append $||a_{\ell m}||^2$ to the training and test data arrays. Further training parameter settings and the motivation behind choosing their values are outlined in appendix A.1.

A class of algorithms that share many similarities with random forests is that of *gradient boosting*. Gradient boosting here specifically refers to combining multiple models trained sequentially such that every new model corrects the classification errors made by the existing ensemble. In practice, gradient boosting is generally implemented using multiple weak learners, such as decision trees. The initial tree is generated by choosing the most important features of the dataset based on the decrease of a chosen loss function. The feature with the highest loss decrease then becomes the root node. Additional nodes of the tree are then added iteratively based on their loss value until the specified maximum depth of the tree is reached. The outer layers of the tree (leaf nodes) then hold the prediction of the tree and can be used to calculate the residual values by comparing to the data labels in the test dataset. The subsequent trees during the training procedure are trained in an analogous way, the only difference being that they are trained on the residual data. This feature, in particular, allows the algorithm to correct the errors made during the previous training iteration.

A particularly useful implementation of gradient boosting is offered by the Python package `XGBoost` [44, 45]. In recent years, `XGBoost` has been used to generate state-of-the-art results in classification and regression for many astrophysics applications, often producing similar or better results than those obtained with commonly used neural network algorithms. In addition, `XGBoost` has been used as a winning algorithm for multiple Kaggle competitions. Being based on ensembles of decision trees, the predictions made by `XGBoost` are generally not prone to overfitting for a large number of features. Similarly, having an ensemble of decision trees allows us to calculate the relative feature importance in a fashion analogous to the random forest algorithm.

We train an extreme gradient boosting classifier provided by the `XGBClassifier` class from the `XGBoost` Python package. The training settings for the classifier are generally analogous to the one used for random forests, with the main exception being that we use 2000 decision tree classifiers (rather than 3500 for the random forest classifier). The data is split into 80% training and 20% test datasets. The complex values are split into the real and

JCAP09(2024)057

imaginary parts, which are appended into a single data vector. The non-zero eigenvalue of the $\mathcal{C}_{\ell m \ell' m'}$ is also appended to each data vector. The full list of settings used, along with how the particular values are chosen, is described in appendix A.1.

3.2 1D convolutional neural networks trained on $a_{\ell m}$ data

CNNs refer to the class of neural networks that employ convolutions in order to extract features from N-dimensional data. The key building blocks of CNN algorithms are the convolutional layers, which apply convolution operations to the input data. Specifically, these layers include filters or kernels of a chosen size that slide across the input data and perform convolutions, which extract a set of feature maps. Extra features can then be extracted in the subsequent layers, which are ultimately combined in the final set of (fully connected) layers of the network in order to generate a prediction. In addition to the mentioned convolutional layers, the networks often include extra elements, such as pooling layers that downsample the feature maps in order to reduce complexity and to extract the key information. Dropout layers are also often added to randomly switch off a certain fraction of neurons in order to avoid overfitting.

1D CNNs refer to the sub-class of neural networks specifically applied to 1D datasets by performing 1D convolution operations. Due to the structure of 1D CNNs, it is an algorithm of choice for many signal processing tasks ranging from audio classification to astrophysical signal processing. Similarly, 1D CNNs have been widely applied for the analysis of sequential data, such as for speech recognition, time-series analysis, and natural-language processing. In general, 1D CNNs share mostly the same structure as their 2D counterparts, the main difference being that the layers are strictly one-dimensional (`Conv1D`, `MaxPooling1D`, etc.). Extra layers are sometimes added specifically for the analysis of sequential or time-series data, e.g., long-short-term-memory (LSTM) layers.

We train the 1D CNN algorithm on the two mentioned datasets: the 40,000 unrotated $a_{\ell m}$ realizations and the 100,000 randomly rotated (and augmented via extra rotations) data samples. For each realization, we extract the $a_{\ell m}$ data corresponding to $\ell \in [2, 50]$ in (ℓ, m) ordering (see appendix A.2 for a discussion of why these particular values are chosen). In order to deal with complex values of the $a_{\ell m}$ realizations, we split the $a_{\ell m}$ into the real and imaginary parts, which are appended into a single array. In addition, for the rotated dataset, the eigenvalues of the $\mathcal{C}_{\ell m \ell' m'}$ matrix corresponding to each $a_{\ell m}$ realization vector are also appended to the data array (see appendix A.2 for further technical details).

We build and implement the 1D CNN architecture using `TensorFlow` with `Keras` [46, 47]. Generally, we expect the key information related to non-trivial topologies to be stored as correlations between the $a_{\ell m}$ entries for different ℓ's and m's. In order to capture this information, we choose an architecture with multiple kernels of different sizes (see table 3). In addition to the 1D convolution layers, we also add a down-sampling layer (`MaxPool1D`) followed by a dropout layer (with a dropout scale of 0.3) to reduce overfitting. The three final layers are fully connected dense layers, which combine the extracted features into a final classification prediction. The model is compiled using the `Adam` optimizer along with the sparse categorical cross entropy loss function. The model is trained for 40 epochs, which we find to be sufficient for the model to converge. The weights corresponding to the model with

JCAP09(2024)057

the highest validation accuracy during the training procedure are saved as the best model. The test accuracy along with the confusion matrix are calculated for the full test dataset.

3.3 2D convolutional neural networks trained on $\mathcal{C}_{\ell m\ell' m'}$ data

Previous results in the literature indicate that one of the key signatures of non-trivial topology is the non-local correlation pattern between different ℓ's [14–16, 38]. Generally, we expect this information to be easier to capture in two dimensions, i.e., by analyzing the auto-correlation data. In (ℓ, m) ordering, this results in 2D correlations resembling a checkerboard pattern seen in figure 3. While we do expect that the correlations are more easily captured in two dimensions, there is a natural trade-off due to memory constraints — we are limited to rather low values of $\ell_{\max}$, i.e., $\ell_{\max} \lesssim 20$–30. This is the case as, even for $\ell_{\max} = 20$, the 2D $\mathcal{C}_{\ell m\ell' m'}$ array is significantly larger than the 1D vector of $a_{\ell m}$ with even $\ell_{\max} = 100$ (42849 data entries per realization versus 5151).

We prepare the data by calculating a 2D realization $\{\mathcal{C}_{\ell m\ell' m'}\}$ for $\ell, \ell' \in [2, 20]$, for each 1D realization $\{a_{\ell m}\}$. Following the approach described in ref. [16], each 2D realization is rescaled by $\left(C_\ell^{\Lambda\mathrm{CDM}} C_{\ell'}^{\Lambda\mathrm{CDM}}\right)^{-1/2}$, with $C_\ell^{\Lambda\mathrm{CDM}}$ the ΛCDM covering-space angular power spectrum. Since the resulting correlation values are complex, we store the real and imaginary parts as different channels. In addition to the imaginary and real values, we add an extra data channel corresponding to the absolute values of $\mathcal{C}_{\ell m\ell' m'}$, resulting in each data sample having the shape $(207, 207, 3)$. Due to memory constraints, we perform the training on the original dataset of 40,000 unrotated and the 40,000 randomly Wigner-rotated realizations (without data augmentation).

In an attempt to capture the correlations between the different multipoles, we choose to work with the `ResNet-50` architecture available from the `Keras` model library [48]. Originally designed by Microsoft Research, `ResNet-50` has demonstrated state-of-the-art results for image classification tasks. Due to the use of convolutional layers with different filter sizes and pooling operations, `ResNet-50` based architectures are known to work well with data having features that are correlated at different scales. Another advantage of the `ResNet-50` architecture is that the model available on `Keras` library allows working with pre-trained model weights obtained by training on the `ImageNet` dataset,[4] which generally results in a faster training procedure. A more in-depth discussion of the `ResNet-50` architecture and the settings of the training procedure are available in appendix A.3.

3.4 2D complex convolutional neural networks trained on $\mathcal{C}_{\ell m\ell' m'}$ data

Both the coefficients $a_{\ell m}$ of the expansion (1.1) and the elements of the covariance matrix (2.6) are complex. In the previous sections, we split the complex inputs into real and imaginary parts. In this section, we explore the use of complex-valued neural networks (CVNNs) [49], a set of neural networks capable of handling complex numbers. It has been proven that CVNNs have a stronger generalization power than real-valued neural networks (RVNNs) [50]. The findings in ref. [51] indicate that CVNNs outperform RVNNs across various architectures and data structures. The authors also claim that the accuracy of CVNNs demonstrates a

[4]Available at https://image-net.org/index.php.

JCAP09(2024)057

statistically higher mean and median, coupled with lower variance compared to RVNNs, and that when no regularization technique is applied, CVNNs exhibit reduced overfitting.

The CVNN library [49] generalizes the RVNNs using `Tensorflow` as back-end to enable the use of complex inputs, weights, and activation functions. The architecture used in the present work is based on the one described in ref. [52], where it was used to extract the cosmological parameters Ω_M and σ_8 from noisy weak-lensing maps. The network consists of 15 2D convolutional complex layers grouped in sets of 2, 3, and 5 layers, with different kernel sizes, followed by a 2D average pooling layer in between sets (see table 4). Lastly, two dense layers output the prediction for each input realization. The architecture is described in more detail in appendix A.4.

The dataset used to train the CVNNs is the same as described in the previous section, i.e., $\mathcal{C}_{\ell m \ell' m'}$ data with $\ell \in [2, 20]$. However, we use the complex array itself as input to the network. Due to memory constraints, we do not apply any data augmentation via extra rotation when training the CVNN (i.e., we use the rotated and unrotated data samples with 40,000 realizations in each case). The data set is split into 80% for training, 10% for validation, and 10% for test. The network is trained for 120 epochs using *stochastic gradient descent* with a learning rate of 0.002 (see appendix A.4 for a further discussion of the chosen training parameters).

4 Results

The obtained results indicate that the classification accuracy depends on several factors, including the range of multipoles included in the dataset, the data ordering, and, most importantly, whether or not the realizations have been randomly rotated. Similarly, the results depend strongly on the topology scale. In this section, we summarize the main findings for each of the algorithms considered for realizations with topology scale $L < L_\mathrm{LSS}$. In addition, we present a set of results for realizations with $L \gtrsim L_\mathrm{LSS}$.

4.1 E_1 realizations with $L < L_\mathrm{LSS}$

The results obtained for decision-tree-based algorithms, i.e., random forests and `XGBoost`, indicate a classification accuracy of 99.8%, when calculated on unseen unrotated test data. In the case where the training and the test datasets are randomly rotated, the combined test accuracy (over all dataset classes) decreases to 91%–94%, depending on the algorithm (see table 1). The performance on the studied topology classes is summarized in the confusion matrices presented in figure 5. The values along the diagonals list the accuracy for each topology class, while the off-diagonal values specify the percentage of misclassified realizations.

We find generally that for both datasets the two smallest classes $L = 0.05 \times L_\mathrm{LSS}$ and $L = 0.1 \times L_\mathrm{LSS}$ are classified with 100% accuracy. For the randomly rotated realizations, both algorithms perform somewhat worse when distinguishing the $L = 0.5 \times L_\mathrm{LSS}$ and L_∞ classes. For these particular classes the `XGBoost` algorithm achieves significantly higher accuracies, i.e., 86%–91% of realizations are classified correctly, compared to 78%–88% for the random forest algorithm. When the realizations are misclassified, we generally find that the $L = 0.5 \times L_\mathrm{LSS}$ class is confused for the covering space realizations and vice versa.

JCAP09(2024)057

	Random forests	`XGBoost`
Training (unrotated)	100%	100%
Training (rotated)	100%	100%
Test (unrotated)	99.8%	99.8%
Test (rotated)	91.4%	94.2%

Table 1. Summary of the random forest and `XGBoost` results for different datasets. The percentages refer to the combined classification accuracy (over all the considered classes, i.e., $\{0.05, 0.1, 0.5, \infty\} \times L_{\rm LSS}$). Unrotated and rotated refer to the two cases where the algorithms are trained and tested on unrotated and randomly rotated datasets, respectively. The accuracy for each individual class is listed in the confusion matrices of figure 5. In each case, the algorithms are trained on data with $\ell \in [2, 100]$.

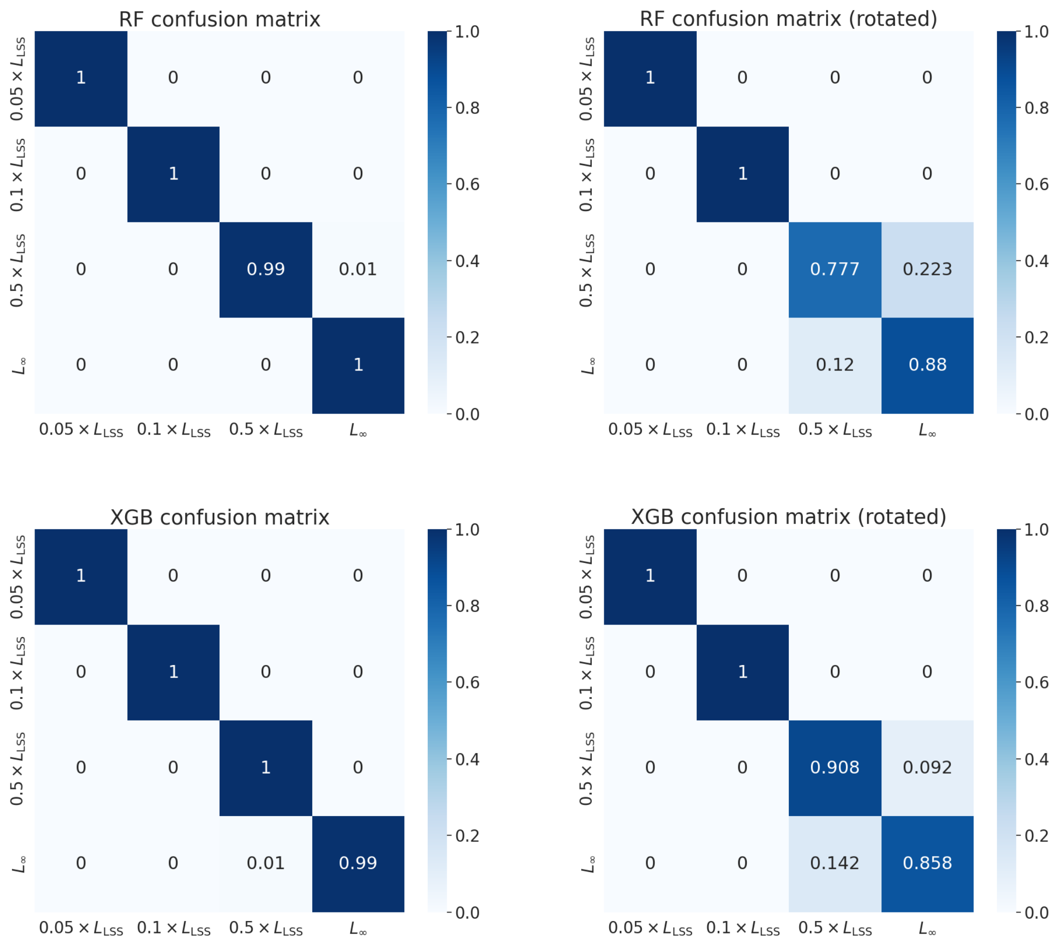

Figure 5. Normalized confusion matrices for the rotated and unrotated test datasets and for the random forest (RF) and `XGBoost` (XGB) algorithms. The left panels show the obtained classification accuracies for the unrotated test datasets, while the right panels show the accuracies for the randomly Wigner-rotated test data. In each confusion matrix the rows represent the true class, while the columns represent the predicted class.

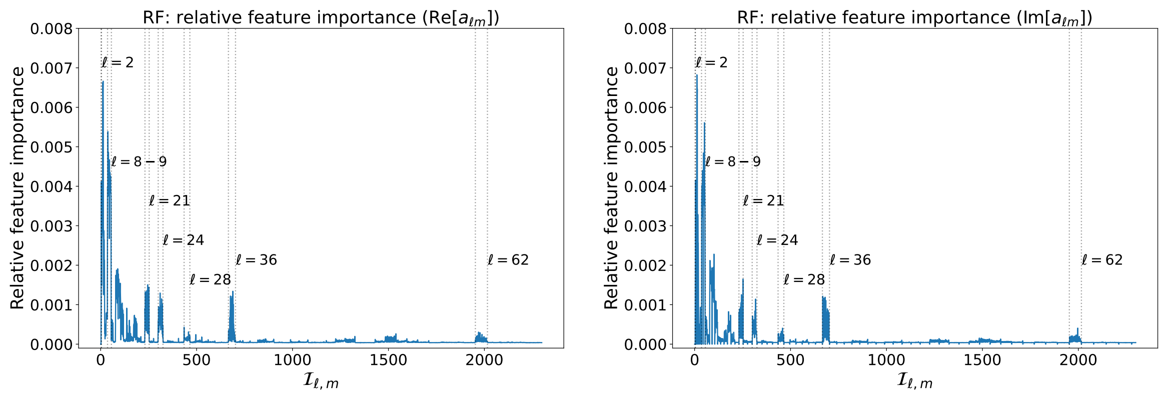

Figure 6. Relative feature importance plots for the random forest (RF) algorithm. The feature importance for the real parts of the $a_{\ell m}$ are shown on the left and the imaginary parts are shown on the right. The different spikes show the $a_{\ell m}$ coefficients that are particularly important for distinguishing the randomly rotated $\{0.05, 0.1, 0.5, \infty\} \times L_{\rm LSS}$ topology classes. We have marked certain values of ℓ with dotted lines for reference. The feature importance for the $||a_{\ell m}||^2$ (the non-zero eigenvalue) that is also appended to the training data is not shown here. The $||a_{\ell m}||^2$ feature importance is equal to 0.04.

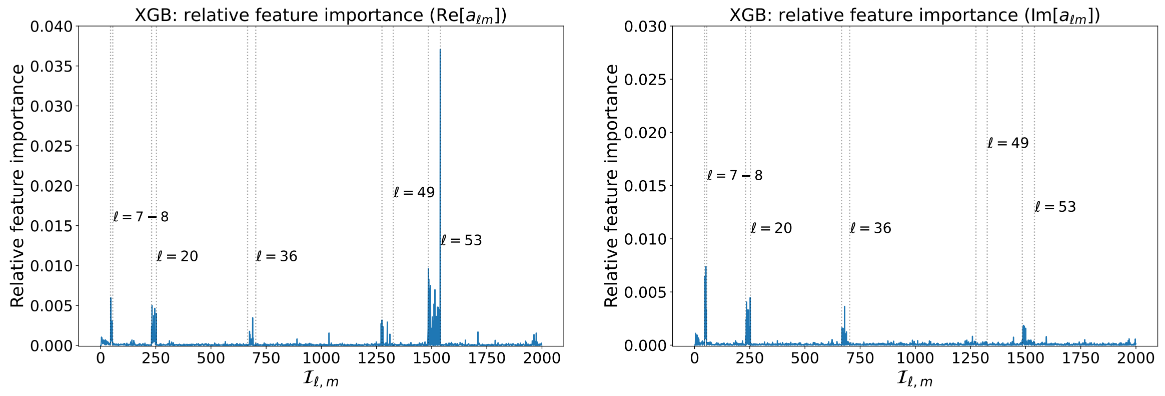

Figure 7. As in figure 6, but for the `XGBoost` (XGB) algorithm. The $||a_{\ell m}||^2$ feature importance (not shown here) has the value of 0.026.

This aligns with the expectation that smaller manifolds should be easier to classify, as they exhibit stronger correlations.

One of the particular advantages of the decision-tree-based algorithms is the ability to calculate the relative feature importance. In our case, this specifically refers to the subset of $a_{\ell m}$ coefficients that are of particular importance for distinguishing the different topology classes. Figure 6 and 7 are feature importance plots for the random forest and `XGBoost` classification results. The different spikes mark the most important $a_{\ell m}$ coefficients. The features are split into the real and imaginary values.

For the random forests, we observe that the $||a_{\ell m}||^2$ feature (the non-zero eigenvalue) has the largest feature-importance score, likely due to being rotationally invariant. We find that the $a_{\ell m}$ with small ℓ values are considerably more important than those with large ℓ values. In the case of the `XGBoost` algorithm, however, the subsets of the $a_{\ell m}$ with $\ell \in \{7, 8, 20, 36, 49, 53\}$

	1D CNN	**2D CNN**	**2D CVNN**
Training (unrotated)	100%	99.9%	100%
Training (rotated)	98.8%	98.4%	98.6%
Test (unrotated)	99.8%	99.4%	99.3%
Test (rotated)	93.7%	92.5%	91.8%

Table 2. As in table 1, but for the neural-network-based results. 1D CNN refers to the one-dimensional convolutional neural network trained directly on $a_{\ell m}$ data, 2D CNN refers to the `ResNet-50` architecture trained on two-dimensional $\mathcal{C}_{\ell m \ell' m'}$ data, and 2D CVNN refers to the two-dimensional complex convolutional architecture trained on $\mathcal{C}_{\ell m \ell' m'}$ data. The accuracy for each individual class is listed in the confusion matrices of figure 8.

seem to be particularly important, comparable in importance to the $||a_{\ell m}||^2$ feature. We also observe some minor differences between the real and imaginary features, but it is difficult to tell whether these point to the real features being more important that the imaginary ones, or that the observed differences are due to inherent randomness of the algorithm. Specifically, training the algorithms several times will result in the tallest peaks appearing in the same locations, but their relative importances will fluctuate due to dataset randomization and the inherent semi-randomness when choosing the features from which to build the decision trees. Another minor difference between the real and imaginary features is that a significant subset of imaginary features have a feature importance of 0. This specifically refers to the subset of $\{a_{\ell 0}\}$, which are purely real and hence their imaginary parts are not important for classification. Finally, in the random forest case, we also observe several important features corresponding to larger values of ℓ, e.g., $\ell = 62$. The origin of the listed feature-importance spikes is unclear, but may be related to the inherent symmetries of the E_1 topology.

Table 2 is a summary of the results for the neural-network-based algorithms for the rotated and unrotated datasets. Specifically, we summarize the results for the 1D convolutional neural networks trained directly on the $a_{\ell m}$ data, as well as the 2D convolutional neural network results trained on the $\mathcal{C}_{\ell m \ell' m'}$ correlation data. The trend displayed in these results is generally the same as that shown for decision-tree-based algorithms — we are able to accurately classify all the topology classes in the unrotated cases, while in the randomly rotated cases, classifying the largest E_1 realizations remains challenging. When classifying unrotated realizations, both the 1D CNN and 2D CNN (`ResNet-50` and CVNN) algorithms obtain nearly perfect accuracy scores in the range of 99%–100% for both the test and the training datasets. For the randomly rotated data, the algorithms obtain accuracies in the range 91.8% – 93.7%, when calculated on unseen test data. In this regard, the test results are slightly worse than those obtained by the `XGBoost` algorithm, however, any comparison between the algorithms should be done with caution, as different methodologies and different subsets of the dataset have been used in each case. For example, we do not observe the CVNN model outperforming the 2D (real-valued) CNN and the `RestNet-50` model. Specifically, the CVNN model, in its current implementation, does not work with many of the commonly used layers employed by many of the real-valued networks described in this work (e.g., skip connections), making a direct comparison of the classification results difficult.

JCAP09(2024)057

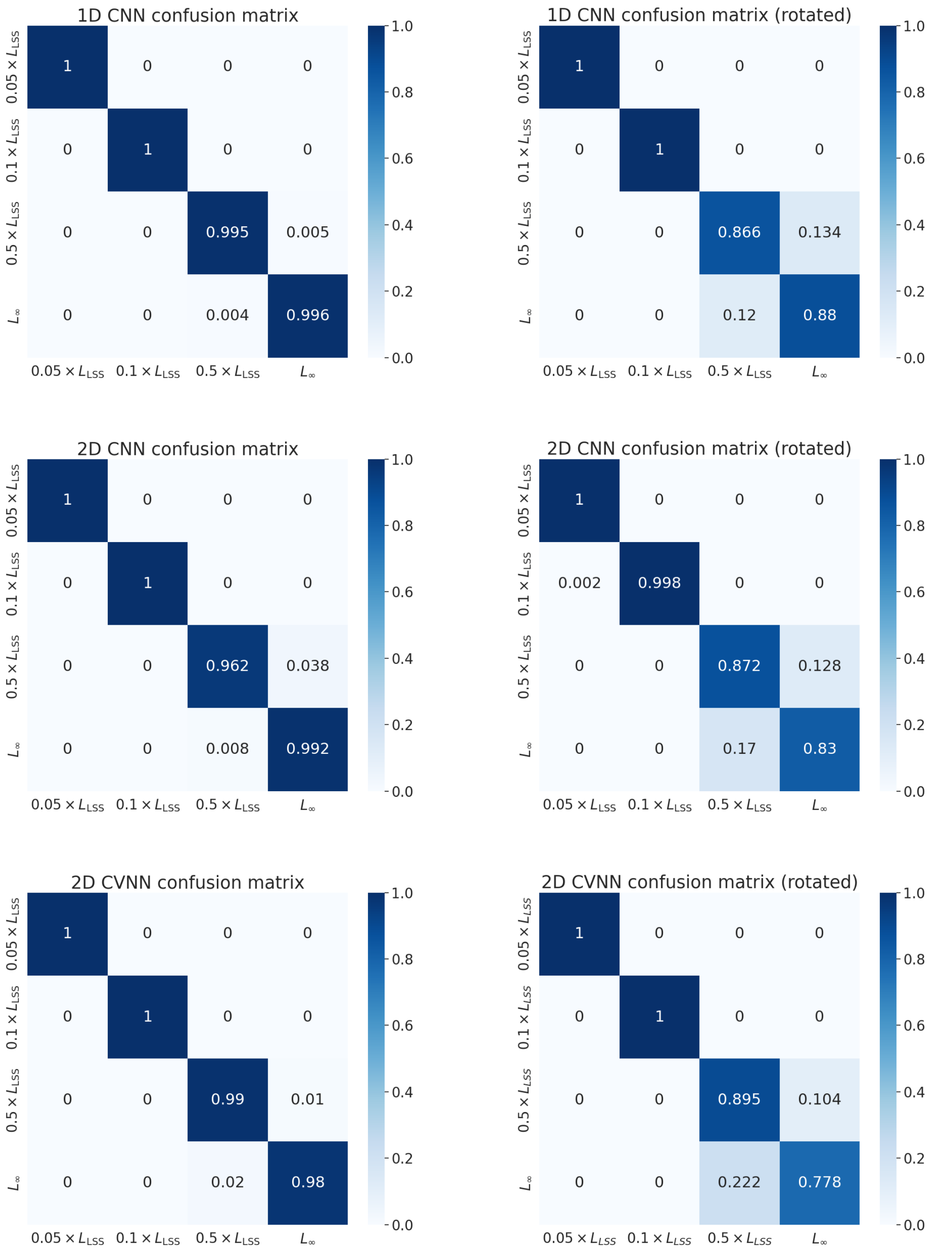

Figure 8. As in figure 5, but for the neural-network-based algorithms 1D CNN, 2D CNN (`ResNet-50`), and CVNN.

As before, the confusion matrices for each dataset and each algorithm illustrate the individual topology class performance, as shown in figure 8. Here we find that the algorithms classify the smaller E_1 topology classes, i.e., $0.05 \times L_{\rm LSS}$ and $0.1 \times L_{\rm LSS}$, nearly perfectly for all datasets (rotated and unrotated). When trained on randomly rotated realizations, the smaller classes are still classified with near-perfect accuracy, while classifying the larger ones, i.e., $0.5 \times L_{\rm LSS}$ and the covering space classes, is more challenging.

It is also important to note that there are methods for calculating different feature importance statistics for CNNs (see, e.g., refs. [53–55]). This includes techniques such as activation maximization, layer-wise relevance propagation, and saliency maps. However, calculating these statistics is non-trivial and often requires a specific training procedure and sometimes specific layers incorporated in the architecture of the CNN. Similarly, while generally one can determine which input pixels were relatively more important, converting that to an interpretable statistic is difficult. For these reasons, we do not consider these methods in this work and leave it to be explored in future publications.

4.2 E_1 realizations with $L \gtrsim L_{\mathrm{LSS}}$

Previous work in the literature indicates that the KL divergence related to the signal of non-trivial topology in the CMB temperature anisotropies decreases with the size of the manifold. Specifically, there is a sharp decrease when the topology scale becomes larger than the diameter of the last-scattering surface, L_{LSS} (see figure 4 of ref. [14]). In part this is related to the fact that we do not expect to see matched circle pairs in the CMB for manifolds of such size. These matched circle pairs are composed of tightly correlated pixel pairs. When $L > L_{\mathrm{LSS}}$, these tight pixel-pixel correlations are absent, and we may expect a qualitative shift in the nature of the correlations — certainly the KL divergence between a compact manifold and the covering space declines. Thus, a pressing question is whether we can use the same machine learning methods that are described in this work in order to classify realizations of E_1 larger than L_{LSS}. Here, as a test, we present the results for a random forest classifier trained on $a_{\ell m}$ realizations falling into two classes: E_1 topology with $L = 1.01 \times L_{\mathrm{LSS}}$ and the covering space. The results are shown for different dataset sizes ranging between 200 and 200,000 realizations, generated via the Cholesky decomposition method (see appendix B for technical details).

The results are summarized in figure 9. For unrotated realizations, the performance of the random forest classifier depends significantly on the total dataset size. In order to obtain accuracies of $\sim 90\%$, around 30,000 realizations are needed. For a dataset of 200,000 realizations the algorithm obtains a test dataset accuracy of 98%, i.e., comparable to that obtained for realizations with $L < L_{\mathrm{LSS}}$. This is an encouraging result as it shows that, even for large unrotated E_1 realizations, the classification accuracy is limited primarily by the dataset size. If sufficient number of realizations is available, high classification accuracies can be obtained.

For rotated realizations, however, the results are less optimistic. In particular, increasing the dataset size from 20,000 realizations to 200,000 realizations has almost no effect on the test classification accuracy, which is a dismal 52%. Note that a similar decrease in accuracy when training on the randomly rotated dataset (albeit to a much lesser extent) is observed for the results in section 4.1. These results likely indicate that the issues related to rotated realizations cannot simply be resolved by adding more data. New techniques are required.

5 Discussion and conclusions

In this work, we have explored the effectiveness of two categories of machine learning algorithms, decision-tree-based random forests and `XGBoost` classifier, along with 1D and

JCAP09(2024)057

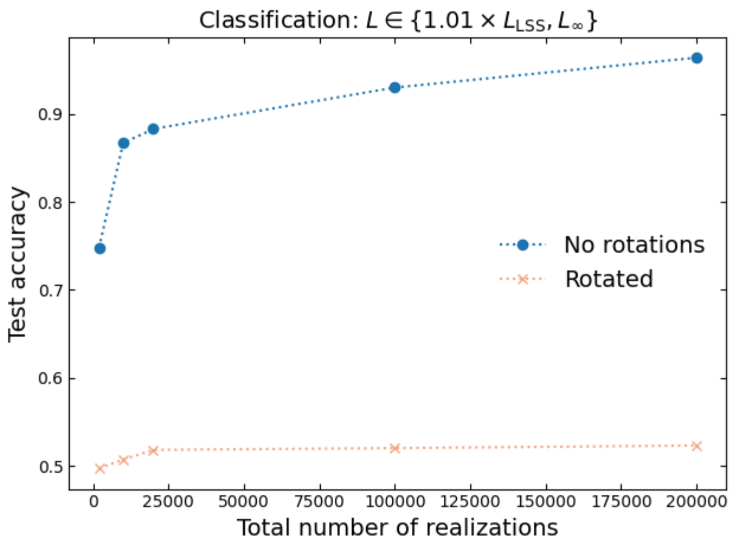

Figure 9. Classification results for E_1 realizations with $L = 1.01 \times L_{\rm LSS}$ versus the covering space realizations. All realizations are generated with $\ell_{\rm max} = 30$ using the Cholesky decomposition method. The test dataset accuracy is calculated on unseen data, which is equal in size to the 20% of the total dataset size.

2D convolutional neural networks, for classification of harmonic-space realizations of CMB full-sky temperature maps for classes of cosmic topology. We have tested the outlined algorithms on the specific case of cubic E_1 (or 3-torus) topology of four different size classes, $\{0.05, 0.1, 0.5, \infty\} \times L_{\rm LSS}$. In addition, we have assessed the feasibility of distinguishing $a_{\ell m}$ realizations of cubic E_1 with the size scales $L \gtrsim L_{\rm LSS}$ from the $a_{\ell m}$ realizations of the covering space. We have generated two sets of results in each case: one for the case where the orientation of each realization is aligned with the coordinate frame of the cubic E_1 manifold and one where each realization is randomly rotated. Even though our focus in this work has been on manifolds with small size scales, which are excluded by current observational data through, e.g., circles-in-the-sky searches, the obtained results identify the prospects and the main challenges for developing machine learning methods that are capable of accurately classifying observationally viable topologies.

All the machine learning algorithms we have tested can distinguish realizations of E_1 topology with a cubic fundamental domain (and of the covering space) with high accuracy when the realizations in the dataset are unrotated relative to a coordinate system aligned with the edges of the cube. Although we have not presented these, the same conclusion applies when all the realizations in all the data sets are rotated by the exact same arbitrary rotation. In other words, if the coordinate system of the observer is oriented relative to the coordinate axes in the same exact way for each and every realization, the nearly perfect results observed for the unrotated realization are recovered.

For datasets where each realization is randomly rotated relative to one another, the classification results are generally worse for the $L = 0.5 \times L_{\rm LSS}$ and the covering space classes. To some extent, this is not surprising, as Wigner rotations scramble the m's for

JCAP09(2024)057

each $a_{\ell m}$ in the dataset, and do so in a different way for each ℓ, making it difficult for the algorithms to learn the features encoded in the correlations between the different m and ℓ values. Conceptually, this problem can be compared to classic CNN architectures being generally bad at training on rotated image data. In fact, a more accurate analogy would be training a CNN on a set of images, where the pixels of each image have been randomly shuffled. More generally, standard CNN architectures do not allow for learning rotationally invariant and equivariant features. There are, however, custom CNN architectures that are capable of learning rotationally invariant features, which does allow dealing with rotated image data more easily and accurately [56–59]. Hence, one possible direction of future work is exploring such more complex neural network architectures. Given that we can obtain an accuracy greater than $\sim 90\%$ for realizations with $L < 0.5 \times L_{\rm LSS}$ without imposing rotational invariance, it is reasonable to expect that using one of the rotationally invariant architectures mentioned above would only further increase the classification accuracy.

The importance of the orientation of the coordinate system when looking for signal of non-trivial topology has been previously considered in ref. [28]. Specifically, the authors concluded that the likelihood of detecting a signal of non-trivial topology is maximized when the orientation of the harmonic-space realizations is aligned with the orientation of the covariance matrix that is used for comparison (figure 1 in ref. [28]). This result, in particular, offers some hints on how an analytic covariance matrix for a specific topology class could be used to align a given randomly rotated $a_{\ell m}$ realization. Specifically, one could calculate rotation curves akin to those shown in figure 1 of ref. [28], and then use the maximum value of the correlation coefficient or the likelihood to further rotate a given realization to one of the optimal orientations. Alternatively, such rotation curves could be used as an extra input to our algorithms. Based on the findings of ref. [28], such an approach is promising; however, it is beyond the scope of this work and is left for future publications.

Our investigation indicates that machine learning techniques described in this work are also capable of correctly classifying harmonic-space realizations for non-trivial topology manifolds of $L \gtrsim L_{\rm LSS}$. Specifically, in section 4.2 we have calculated the classification accuracy for significantly larger datasets generated using a different method (Cholesky decomposition). The obtained results indicate that random forests trained on a large dataset with $\sim 10^5$ harmonic-space realizations can be classified with accuracies comparable to those presented in table 1. Increasing the size of the training dataset results in a continuous increase of the accuracy scores on the test data. However, that is only the case when trained on unrotated harmonic-space realization data. Increasing the total dataset size seems to have almost no effect when training on randomly rotated realizations. This shows that the issues related to training on rotated realizations cannot be solved simply with a larger training dataset. Nonetheless, if these problems can be addressed, our results indicate that machine learning offers a set of powerful techniques to detect the signal of non-trivial topology.

Note that in this work we have only explored a small region of the parameter space of the E_1 topology, i.e., cubic manifolds with $L < L_{\rm LSS}$ and $L \gtrsim L_{\rm LSS}$. In future work this will be extended to non-cubic and possibly tilted (see, e.g., section 3.1 of ref. [16]) manifolds. Previous work also indicates that different classes of topology can have different KL divergence values for the same topology scale (e.g., see figure 4 of ref. [14]). What is more, even if a given topology has the same value of KL divergence, it does not necessarily mean that the features encoded in harmonic-space realizations will be encoded in the same way. Hence, another natural extension of this work is to explore a larger set of topologies. Cubic 3-torus topology has significant accidental rotational symmetries (i.e., the symmetry of a cube), whereas generic topologically non-trivial manifolds lack those symmetries. It therefore seems likely that more general E_1 and other topologies will be easier to classify. Our preliminary tests indicate that the classification accuracies for E_2–E_6 are generally larger than those obtained in this work. However, these results require further analysis, which is left for future publications.

Furthermore, we note that a spherical harmonic expansion is not the only possible representation of the CMB sky. An alternative representation, the multipole vector formalism [60], has been used extensively in studies of the CMB anomalies [60–62]. A distinguishing feature of multipole vectors is that rotations of the coordinate system correspond to a rigid rotation of the set of multipole vectors. It is therefore easy to define rotationally invariant quantities, such as dot products of multipole vectors. These features of the multipole vector formalism warrant further investigation using machine learning, which is left for future work.

Techniques presented in this work deal specifically with the harmonic realizations of the full sky maps. Looking forward, an interesting question is whether techniques like this could be applied to real observational data. This raises a number of important questions, such as what would be the effects of working with the harmonic realizations of partial or masked sky maps. Presumably, effects such as these would break isotropy and introduce extra harmonic space correlations, which would be challenging to distinguish from the correlations induced by non-trivial topology. Similarly, effects such as foreground residuals and beam asymmetries would have to be accounted for. While studying these systematics is beyond the scope of this work, it is clear that any successful method applied to real CMB data would have to take into account these observational effects. Hence, correctly accounting for such effects is a natural direction for future work.

The problem of distinguishing harmonic space realizations in non-trivial topology from the corresponding covering space realizations has previously been explored using Bayesian and frequentist approaches in ref. [28]. While we did not attempt to use likelihood-based methods in this work, the techniques described in ref. [28] offer a valuable point of comparison. Namely, the authors of ref. [28] have calculated the likelihood function using 10^4 simulated harmonic space realizations of E_1 topology of different sizes. The likelihood calculations have then been used to distinguish between two cases of special interest: one of an infinite universe (covering space) and the other corresponding to a universe with an E_1 topology characterized by the relevant correlation matrix. The statistical significance with which the two scenarios can be distinguished has been quantified by the signal-to-noise ratio statistic (eq. (12) in ref. [28]). For instance, a universe with $L \approx 0.66 \times L_{\mathrm{LSS}}$ (or $T[4,4,4]$ in the units of ref. [28]) can be distinguished from the covering space with $5\sigma - 9.3\sigma$ significance depending on the value of $\ell_{\max}$ and the details of the normalization. In our case, for instance with the 1D CNN algorithm, we can distinguish the $L = 0.5 \times L_{\mathrm{LSS}}$ class from the covering space with 86.6%–88% accuracy for the rotated datasets and 99.5%–99.6% accuracy for the non-rotated datasets. However, it is important to note that the correlation matrices used in ref. [28] do not include the integrated Sachs-Wolfe effect (ISW) contribution. Properly accounting for the ISW effect should in principle significantly lower the detection power of the outlined

JCAP09(2024)057

likelihood approach, as the authors have pointed out in their section VIII C. In our work, the used correlation matrices do include the ISW and the Doppler shift contributions.

Nonetheless, the results quoted above indicate that likelihood-based methods can generally distinguish E_1 from the covering space with statistical significance higher than that of the machine learning methods described in this work. However, it is important to point out the limitations of the likelihood-based approaches. Firstly, the obtained results are very sensitive to the relative orientation of a given $\{a_{\ell m}\}$ set and the corresponding correlation matrix used in the likelihood calculation (this is fundamentally the same issue of dealing with rotated realizations that we have described in this work). The likelihood generally needs to be calculated for over 10^6 orientations to obtain the results quoted in ref. [28]. Furthermore, a full likelihood search would require considering 17 non-trivial topologies, each with up to 6 real parameters specifying the manifold, up to 3 parameters specifying the location of the observer, and up to 3 further parameters specifying the orientation. Machine learning methods offer a computationally cheaper and more efficient alternative: after a training stage, which generally takes several hours, predictions can be generated for thousands of realizations (and over thousands of orientations) in a matter of seconds.

Another important aspect to consider is that in the likelihood-based approaches it is difficult to know what particular features in harmonic space (or pixel space) carry the information that allows distinguishing non-trivial topology from the covering space. In other words, given a CMB map of non-trivial topology, what information specifically distinguishes it from a corresponding covering space map? Machine learning approaches offer a number of techniques to extract this information. As illustrated in our work, the feature importance analysis provides information on which multipoles (or which angular scales in the sky maps) carry the information that is particularly useful for distinguishing that given realization from the covering space realizations. Such information is particularly valuable when considering $a_{\ell m}$ realizations with $L > L_{\mathrm{LSS}}$, as in this regime there are no matched-circle pairs, and hence, it is not trivial to deduce how the information that allows distinguishing between different topologies is encoded in pixel space. In future work we hope to combine the outlined feature importance analysis with other methods such as convolutional neural network layer-wise relevance propagation or saliency analysis. Our hope is to also use autoencoders and information-maximizing neural networks (IMNN's) [63] to extract features related to non-trivial topology and do likelihood-free inference [64].

In summary, we have demonstrated that machine learning techniques are promising for using harmonic space realizations to distinguish between classes of topological manifolds, including distinguishing non-trivial topology from the covering space. Machine learning methods presented in this work also offer a computationally efficient way of extracting features related to non-trivial topology. Nonetheless, our freedom to rotate our coordinate system is the current principal challenge.

Acknowledgments

We thank Roger French for valuable conversations. This work has made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University. This work has also made use of the High Perfor-

JCAP09(2024)057

mance Computing Resources available as a part of the Pitzer Cluster provided by the Ohio Supercomputer Center.

A.T. is supported by the Richard S. Morrison Fellowship. F.C.G is supported by the Presidential Society of STEM Postdoctoral Fellowship at Case Western Reserve University. Y.A. acknowledges support by the Spanish Research Agency (Agencia Estatal de Investigación)'s grant RYC2020-030193-I/AEI/10.13039/501100011033, by the European Social Fund (Fondo Social Europeo) through the Ramón y Cajal program within the State Plan for Scientific and Technical Research and Innovation (Plan Estatal de Investigación Científica y Técnica y de Innovación) 2017–2020, and by the Spanish Research Agency through the grant IFT Centro de Excelencia Severo Ochoa No. CEX2020-001007-S funded by MCIN/AEI/10.13039/501100011033. J.R.E. acknowledges support from the European Research Council under the Horizon 2020 Research and Innovation Programme (Grant agreement No. 819478). T.S.P acknowledges financial support from the Brazilian National Council for Scientific and Technological Development (CNPq) under grants 312869/2021-5 and 88881.709790/2022-01. C.J.C., A.K., D.P.M., and G.D.S. acknowledge partial support from NASA ATP grant RES240737; G.D.S. from DOE grant DESC0009946; Y.A., O.G., G.D.S., S.S., and Q.T. from the Simons Foundation; Y.A., A.H.J., and G.D.S. from the Royal Society (U.K.); and A.H.J. from STFC in the U.K.

Certain images in this publication have been obtained by the author(s) from the Wikipedia/Wikimedia website, where they were made available under a Creative Commons licence or stated to be in the public domain. Please see individual figure captions in this publication for details. To the extent that the law allows, IOP Publishing and SISSA disclaim any liability that any person may suffer as a result of accessing, using or forwarding the image(s). Any reuse rights should be checked and permission should be sought if necessary from Wikipedia/Wikimedia and/or the copyright owner (as appropriate) before using or forwarding the image(s).

A Training procedures and settings used for machine learning algorithms

A.1 Random forests and extreme gradient boosting classifier

Our analysis shows that the most important parameters for the random forest classifier are the number of estimators (`n_estimators`) and the maximum depth of the trees (`max_depth`). After performing a grid search on a small sample dataset to determine the optimal set of these parameters, we find that the number of estimators should be set to around 30%–35% of the number of features, i.e., `n_estimators` $\approx$ 3000–3500 for a full dataset with $\ell_{\rm max} = 100$. Similarly, not adding a limit to the maximum tree depth results in the highest test dataset accuracies.

For the `XGBoost` classifier, we use the same exact dataset with the same data ordering. Similarly, we find that the key parameters for obtaining high classification accuracy are the number of boosted trees (`n_estimators`), maximum three depth for the base learners (`max_depth`), and the boosting learning rate parameter (`learning_rate`). After performing a grid search for the optimal number of boosted decision trees, we find the optimal value to be 2000.

Here we summarize the parameter values used by both classifiers:

1. **Random forest classifier:**
   - Number of tree estimators: `n_estimators = 3500`
   - Maximum tree depth: `max_depth = None`
   - Function to measure the quality of splits: `gini`
   - Number of features to consider when looking for optimal splits: `auto`
   - All the other settings are set to their default values listed in ref. [65].
2. **`XGBoost` classifier:**
   - Number of tree estimators: `n_estimators = 2000`
   - Maximum tree depth: `None`
   - Boosting learning rate parameter: `learning_rate = 0.1`
   - Type of the used booster: `booster = gbtree`
   - Feature importance calculation setting: `importance_type = gain`
   - All the other settings are set to their default values listed in ref. [45].

A.2 1D convolutional neural networks

While the results of the decision-tree-based algorithms do not depend on the $a_{\ell m}$ realization data ordering, this is not generally the case for CNNs. To investigate this for each dataset, we test the two (m, ℓ) and (ℓ, m) orderings of $a_{\ell m}$ values. To asses these orderings we run the 1D CNN training procedure for different subsets of the total dataset with $\ell_{\max} = \{10, 20, 50, 100\}$. In each case, the classification accuracies are then compared for the two different orderings. We generally find that the ordering does not have a significant effect on the classification accuracy, and hence, we choose the (ℓ, m) ordering, which is easier to interpret.

Similarly, we test different ways of representing real and imaginary parts of the harmonic-space realizations. Namely, we test the representation of the real and imaginary parts as different neural network channels rather than simply appending the imaginary part to the real values. However, we do not find it to have a significant effect on the classification accuracy. A similar question is that of the ordering of the $a_{\ell m}$ real and imaginary parts, i.e., one can simply append the array of the real parts to that of the imaginary parts for each realization, resulting in: $[\{a^{\rm Re}_{\ell m}\}, \{a^{\rm Im}_{\ell m}\}]$. Alternatively, one can consider appending the two components for each ℓ, resulting in $[a^{\rm Re}_{00}, a^{\rm Im}_{00}, a^{\rm Re}_{10}, a^{\rm Im}_{10}, \ldots]$. Here we find that the change in the classification accuracy is not generally larger than the usual fluctuations in the accuracy due to the inherent randomness of the training procedure (e.g., randomly choosing a subset of the training/test data, randomizing the initial weights of the network, etc.). We therefore choose to work with the simple approach of appending the real values to the imaginary values for the full range of ℓ's rather than for each ℓ in the dataset.

One key difference we find is that 1D CNNs are much more sensitive to the number of features in the training dataset. In other words, we generally find that including only a subset of the available ℓ values, e.g., $\ell \in [2, \ell_{\max} = 50]$, results in higher test dataset accuracy score. Particularly, we repeat the training procedure for different values of $\ell_{\max} \in \{10, 20, 30, 50, 100\}$ and find that values of $\ell \leq \ell_{\max} = 50$ result in a similar test dataset accuracy, with $\ell_{\max} = 50$ giving the highest value. On the contrary, $\ell_{\max} = 100$ results in worse performance, likely due to feature over-abundance compared to the size of the available dataset.

JCAP09(2024)057

	Activation	Output shape	Parameters
Input map	—	(None, 2652, 32)	128
`Conv1D`	`LReLU`	(None, 2652, 64)	10304
`Conv1D`	`LReLU`	(None, 2652, 128)	57472
`Conv1D`	`LReLU`	(None, 2652, 256)	295 K
`Max Pooling 1D`	—	(None, 1326, 256)	—
`Dropout`	—	(None, 1326, 256)	—
`Flatten`	—	(None, 339456)	—
`Dense`	`LReLU`	(None, 512)	173 M
`Dense`	`LReLU`	(None, 256)	131 K
`Dense`	`LReLU`	(None, 128)	32 K
`Output layer`	`Softmax`	4	516
Total trainable parameters:			**174 M**

Table 3. Summary of the 1D CNN architecture used to train on the $a_{\ell m}$ data. The model is compiled with the `Adam` optimizer with the learning rate of 10^{-5} and *sparse categorical crossentropy* loss function. The dropout rate is set to 0.3.

For the model architecture, we choose to work with multiple 1D convolutional layers with a varying number of filters of different sizes. This is motivated by the fact that we expect the different pairs of multipoles to be correlated in non-trivial topologies (e.g., the *checkerboard* pattern in figure 4). Given the chosen data ordering, this implies that we expect non-local correlations between the different features in our dataset, and having multiple convolutional layers with different kernel sizes is a known method for extracting non-local features and has been considered in architectures used for audio and music genre classification, electroencephalography (EEG) classification, and transiting exoplanet signal detection [66–69]. The convolutional layers are then appended by 1D max pooling and dropout layers in order to assist the feature extraction and to reduce overfitting. The final layers in the architecture are the four dense layers with `LeakyRelu` activation functions. The architecture is summarized in table 3.

We compile the model with the `Adam` optimizer along with the default value for the learning rate and the *sparse categorical crossentropy* loss function. The model is pre-trained for 10 epochs with a batch size of 32 samples and then trained for further 60 epochs. We save the model weights corresponding to the highest validation dataset accuracy. The training is performed on the Case Western Reserve University HPC Pioneer facilities using the GPU cores powered by the Nvidia Tesla V100-SXM2-32GB V100 Graphics Accelerator Card.

A.3 2D convolutional neural networks

In order to directly capture the correlations between the different multipoles, we train a 2D convolutional neural network directly on the correlation data, i.e., $\mathcal{C}_{\ell m \ell' m'}$, rather than on the $a_{\ell m}$ for each realization. This imposes tighter memory constraints, which limit us to $\ell_{\max} = 20$ for each realization. The data is then prepared by extracting the $a_{\ell m}$ corresponding to $\ell \in [2, 20]$ for each realization and then calculating the corresponding $\mathcal{C}_{\ell m \ell' m'} = a_{\ell m} a^*_{\ell' m'}$. We disregard the first two multipole values as we want our data to be comparable with the analytic correlation matrices (figure 4) that are calculated without taking into account $\ell = 0$ and $\ell = 1$ data. Similarly, this is done to correctly rescale the data by $\left(C_\ell^{\Lambda\mathrm{CDM}} C_{\ell'}^{\Lambda\mathrm{CDM}}\right)^{1/2}$, which is calculated without taking the first two multipole values into account. Since our used 2D CNN cannot natively deal with complex values, we split the data into components, which are stacked into 3 channels corresponding to the absolute values, the real part, and the imaginary part, i.e., $\mathcal{C}^i_{\ell m \ell' m'} = [\mathrm{Abs}(\mathcal{C}^i_{\ell m \ell' m'}), \mathrm{Re}(\mathcal{C}^i_{\ell m \ell' m'}), \mathrm{Im}(\mathcal{C}^i_{\ell m \ell' m'})]$.

For this particular dataset, we choose to work with the `ResNet-50` architecture [48, 70]. `ResNet-50` is a deep neural network architecture consisting of a total of 50 layers, originally designed to tackle the vanishing gradient problem. As a core feature, it uses the so-called residual blocks, each of which contains skip connections introduced to avoid the gradient values getting reduced to zero, which effectively halts the training procedure. The bulk of the `ResNet-50` architecture consists of convolutional blocks followed by layers that perform identity mapping (where the input to a layer is directly added to the output). These identity mapping layers, or identity blocks, are a type of residual blocks that combine multiple convolutional layers, followed by batch normalization layers with the `ReLU` activation. The mentioned features along with the different filter sizes and pooling operations in the different layers of the architecture lead to `ResNet-50` being a promising choice for extracting features that are correlated at different scales in the training data (e.g., like the *checkerboard* pattern observed in figure 4). While originally developed for image classification, the `ResNet`-based architectures have been used when working with much more abstract data, e.g., weak lensing convergence and mass maps [71], strong gravitational lensing data [72], and astronomical target classification [73]. The key components of the `ResNet-50` architecture are summarized in figure 10.

`ResNet-50` architecture is usually trained in one of the two ways: via transfer learning, or by fully retraining the model weights. As an initial approach, we test the transfer learning approach by freezing the bulk of the model weights to their optimal values (i.e., model weights deduced by training a `Keras ResNet-50` model on the `ImageNet` dataset [75]) and then appending two fully-connected layers (followed by batch normalization and droupout layers), which are meant to fine-tune the pre-trained model weights. This results in a sub-optimal performance, hence we choose the second approach, which is to fully re-train the `ResNet-50` model weights. We train the model on a total of 40,000 randomly rotated realizations, as well as on the same number of non-rotated data samples. Following a procedure similar to the 1D CNN model, we pre-train the `ResNet-50` model for 30 epochs followed by further 40 epochs with a batch size of 32 data samples. We save the best model during the second stage of the training procedure based on the maximum value of the validation dataset accuracy. The model is trained by using the Case Western Reserve University HPC Pioneer GPU nodes equipped with a Nvidia Tesla V100-SXM2-32GB V100 Graphics Accelerator Card. Additional tests and optimization are done on the Pitzer Cluster provided by the Ohio Supercomputer Center using the high memory nodes with no GPU support [76].

JCAP09(2024)057

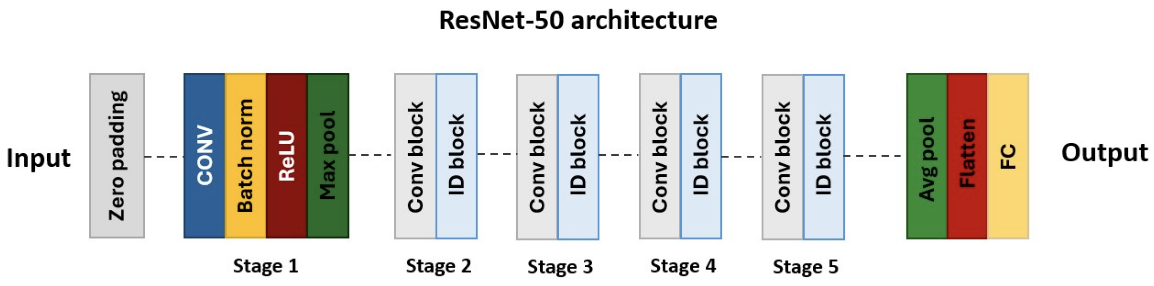

Figure 10. Main components of the `ResNet-50` architecture. The convolutional blocks refer to a series of convolutional layers followed by batch normalization and `ReLU` activation. The final convolutional layer in the convolutional block is generally appended with a shortcut/skip connection. The identity blocks consist of series of convolutional layers with 1×1 and 3×3 kernels followed by batch normalization and `ReLU` activations. In addition, the identity blocks are appended by shortcut/skip connections and an addition operation, which combines the original input with the learnt features from the convolutional layers in the block. The final block that is appended to the base `ResNet-50` architecture ("FC") contains two *Dense* layers with 64 and 32 neurons followed by dropout layers with dropout rates of 0.5. Full technical details on the `ResNet` architecture are provided in ref. [48]. Figure adapted (with permission from the original author) from ref. [74]. Reproduced from [74]. CC BY 4.0. This ResNet50 image has been obtained by the author(s) from the Wikimedia website where it was made available by Gorlapraveen123 under a CC BY-SA 4.0 licence. It is included within this article on that basis. It is attributed to Gorlapraveen123.

A.4 Complex convolutional neural networks

Complex-valued convolutional neural networks allow us to use complex datasets as an input and classify them while maintaining the power of convolutional layers, i.e., studying the correlations between different multipoles. Here, instead of feeding 3 channels of the covariance matrix as described in appendix A.3, we can directly use the complex matrix as the input. As before, to satisfy the memory constraints, the dataset is formed by calculating $\mathcal{C}_{\ell m\ell' m'}$ from the $a_{\ell m}$ corresponding to $\ell \in [2, 20]$ for each realization.

As illustrated in figure 3, cosmic variance introduces noise in the covariance matrix, smoothing away the expected patterns observed in the analytic matrices (figure 4). For this reason we choose to work with a particular CVNN architecture inspired by the one used in ref. [52], which has been shown to be effective when working with noisy data. The network is formed by 15 complex convolutional layers, organized in 5 sets followed by a complex average pooling layer (two sets of 2, two sets of 3, and one set of 5 convolutional layers). The output is flattened and then passed through to dense layers that are responsible for outputting the classification results (see table 4). We train the model on a total of 40,000 randomly rotated realizations, as well as the same number of non-rotated data samples, using 80% for training, 10% for validating, and 10% for testing. The network is pretrained for 40 epochs followed by further 80 with a batch size of 32 data samples. We save the best model during the second stage of the training procedure based on the maximum value of the validation dataset accuracy. The model is trained by using the Nvidia Tesla V100-SXM2-32GB V100 Graphics Accelerator Card provided by the Pioneer HPC facilities at Case Western Reserve University.

	Activation	**Output shape**	**Parameters**
Input map	—	(207,207,1,32)	—
`ComplexConv2D`	`cart_relu`	(None, 205, 205, 32)	640
`ComplexConv2D`	`cart_relu`	(None, 203, 203, 32)	18 K
`ComplexAvgPooling2D`	—	(None, 101, 101, 32)	—
`ComplexConv2D`	`cart_relu`	(None, 99, 99, 64)	37 K
`ComplexConv2D`	`cart_relu`	(None, 98, 98, 64)	33 K
`ComplexAvgPooling2D`	—	(None, 49, 49, 64)	—
`ComplexConv2D`	`cart_relu`	(None, 47, 47, 128)	148 K
`ComplexConv2D`	`cart_relu`	(None, 47, 47, 64)	17 K
`ComplexConv2D`	`cart_relu`	(None, 45, 45, 128)	148K
`ComplexAvgPooling2D`	—	(None, 22, 22, 128)	—
`ComplexConv2D`	`cart_relu`	(None, 20, 20, 256)	590 K
`ComplexConv2D`	`cart_relu`	(None, 20, 20, 128)	66 K
`ComplexConv2D`	`cart_relu`	(None, 18, 18, 256)	590K
`ComplexAvgPooling2D`	—	(None, 9, 9, 256)	—
`ComplexConv2D`	`cart_relu`	(None, 7, 7, 512)	2.4 M
`ComplexConv2D`	`cart_relu`	(None, 7, 7, 256)	263 K
`ComplexConv2D`	`cart_relu`	(None, 5, 5, 512)	2.4 M
`ComplexConv2D`	`cart_relu`	(None, 5, 5, 256)	263 K
`ComplexConv2D`	`cart_relu`	(None, 3, 3, 512)	2.4 M
`ComplexAvgPooling2D`	—	(None, 1, 1, 512)	—
`ComplexFlatten`	—	(None, 512)	—
`ComplexDense`	`cart_relu`	(None, 64)	66 K
`ComplexDense`	`softmax_real_with_abs`	4	520
Total trainable parameters:			**590 K**

Table 4. Summary of the CVNN architecture used for classification of complex $\mathcal{C}_{\ell m \ell' m'}$ data. The used complex activation functions and the complex layers are explained in full detail in ref. [77]. The architecture is based on the one described in ref. [52].

JCAP09(2024)057

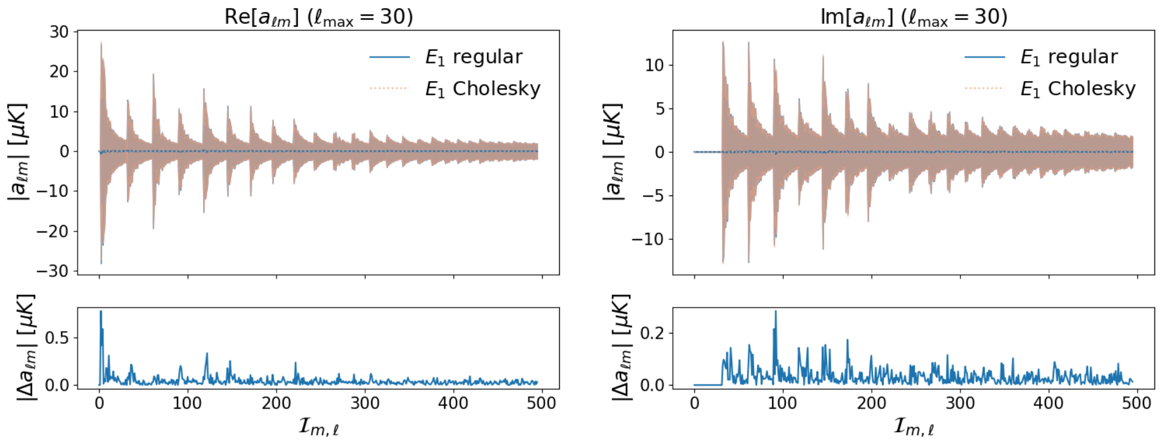

Figure 11. A comparison of the real (left) and imaginary (right) values for an ensemble of 10,000 realizations generated via Cholesky decomposition and by numerically evaluating (2.4). The solid and the dashed lines show the mean values for each m and ℓ, while the color bands correspond to the standard deviations. The residuals, i.e., the absolute values of the differences between the two mean value lines are shown as $\Delta a_{\ell m}$.

B Generating realizations with $L \gtrsim L_{\rm LSS}$ via Cholesky decomposition

In order to test the effectiveness of the algorithms outlined in this work for classifying large realizations of the E_1 topology, a large dataset is required. A key challenge for obtaining large datasets for realizations with $L \gtrsim L_{\rm LSS}$ by numerically evaluating (2.4) is that of memory constraints. Therefore, we present here a different method for generating large E_1 realizations. Specifically, we follow the numerical approach described in ref. [16] to evaluate the covariance matrix for the considered topology class. The covariance matrix can then be used to generate novel $a_{\ell m}$ realizations via Cholesky decomposition (see refs. [16, 36, 37]). The main advantage of this method when compared to numerically evaluating (2.4) is that we can generate realizations significantly faster, with the caveat that the range of multipoles that we consider in our dataset, $\ell \in [2, \ell_{\max}]$, is limited to $\ell_{\max} = 10$–30. In other words, the Cholesky decomposition method allows generating realizations with significantly higher L values at a cost of having significantly smaller $\ell_{\max}$.

Since we are generating this particular set of realizations using a different method, a natural question is how similar are these realizations when compared to those obtained by numerically evaluating (2.4)? To investigate this, we generate two sets of E_1 topology realizations with the same $\ell_{\max}$ and L values and perform a thorough comparison. Namely, we generate a set of 100,000 realizations using the two methods, and then calculate the mean and the standard deviation for the real and imaginary values for each ℓ and m. Similarly, we compute the mean and the standard deviation values for the power spectrum. The obtained ensemble level quantities show only minor differences in the power spectrum and almost no discernible difference between the ensemble values of the Re[$a_{\ell m}$] and Im[$a_{\ell m}$] components. The outlined comparison is illustrated in figures 11 and 12. Finally, to perform a comparison at an individual realization level, we train the random forest and 1D CNN classifiers to distinguish $a_{\ell m}$ realizations generated using the two distinct methods. The obtained results

JCAP09(2024)057

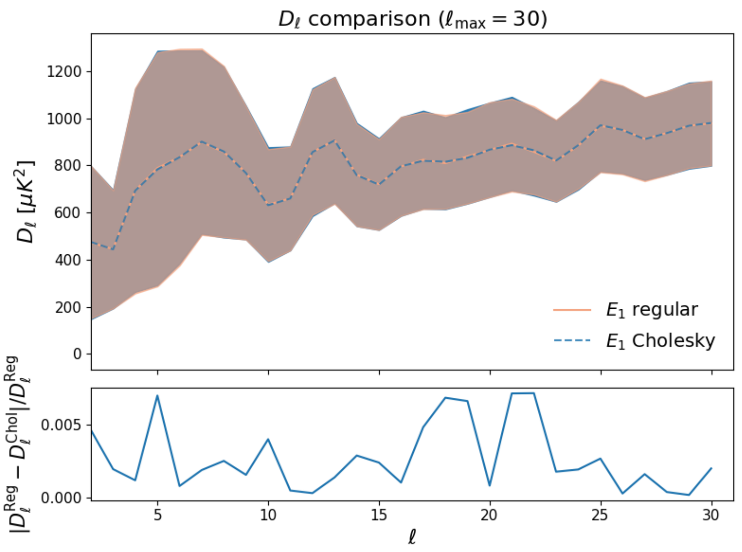

Figure 12. A comparison between the rescaled power spectra of the 10,000 realizations shown in figure 11. Note that $D_\ell \equiv \ell(\ell+1)C_\ell/2\pi$. The solid and the dashed lines show the mean values for each ℓ, while the color bands correspond to the standard deviations. The residual ratio is shown in the bottom section of the figure.

show that neither random forests nor the 1D CNN is capable of distinguishing the two sets of realizations (i.e., the classification accuracy is $\sim 50\%$).

References

[1] A. Einstein, *Cosmological Considerations in the General Theory of Relativity*, *Sitzungsber. Preuss. Akad. Wiss. Berlin* **1917** (1917) 142 [INSPIRE].

[2] P.J.E. Peebles, *Principles of Physical Cosmology*, Princeton University Press (2020) [INSPIRE].

[3] G.D. Starkman, *Topology and cosmology*, *Class. Quant. Grav.* **15** (1998) 2529 [INSPIRE].

[4] M. Lachieze-Rey and J.-P. Luminet, *Cosmic topology*, *Phys. Rep.* **254** (1995) 135 [gr-qc/9605010] [INSPIRE].

[5] J.-P. Luminet and B.F. Roukema, *Topology of the universe: Theory and observation*, in the proceedings of the *NATO Advanced Study Institute: Summer School on Theoretical and Observational Cosmology*, Cargese, France, 17–29 August 1998, *NATO Science Series C: Mathematical and Physical Sciences* **541**, M. Lachièze-Rey ed., Kluwer Academic (1999) [astro-ph/9901364] [INSPIRE].

[6] R. Lehoucq, M. Lachieze-Rey and J.-P. Luminet, *Cosmic crystallography*, *Astron. Astrophys.* **313** (1996) 339 [gr-qc/9604050] [INSPIRE].

[7] B. Mota, M.J. Reboucas and R. Tavakol, *Circles-in-the-sky searches and observable cosmic topology in a flat Universe*, *Phys. Rev. D* **81** (2010) 103516 [arXiv:1002.0834] [INSPIRE].

[8] H. Fujii and Y. Yoshii, *An improved cosmic crystallography method to detect holonomies in flat spaces*, *Astron. Astrophys.* **529** (2011) A121 [arXiv:1103.1466] [INSPIRE].

[9] N.J. Cornish, D.N. Spergel and G.D. Starkman, *Circles in the Sky: Finding Topology with the Microwave Background Radiation*, gr-qc/9602039 [INSPIRE].

JCAP09(2024)057

[10] N.J. Cornish, D.N. Spergel and G.D. Starkman, *Circles in the sky: Finding topology with the microwave background radiation*, *Class. Quant. Grav.* **15** (1998) 2657 [astro-ph/9801212] [INSPIRE].

[11] N.J. Cornish, D. Spergel and G. Starkman, *Can COBE see the shape of the universe?*, *Phys. Rev. D* **57** (1998) 5982 [astro-ph/9708225] [INSPIRE].

[12] N.J. Cornish, D.N. Spergel and G.D. Starkman, *Measuring the topology of the universe*, *Proc. Natl. Acad. Sci. U.S.A.* **95** (1998) 82 [astro-ph/9708083] [INSPIRE].

[13] A. Riazuelo, S. Caillerie, M. Lachieze-Rey, R. Lehoucq and J.-P. Luminet, *Constraining cosmic topology with CMB polarization*, astro-ph/0601433 [INSPIRE].

[14] COMPACT collaboration, *Promise of Future Searches for Cosmic Topology*, *Phys. Rev. Lett.* **132** (2024) 171501 [arXiv:2210.11426] [INSPIRE].

[15] COMPACT collaboration, *Cosmic topology. Part I. Limits on orientable Euclidean manifolds from circle searches*, *JCAP* **01** (2023) 030 [*Erratum ibid.* **04** (2024) E01] [arXiv:2211.02603] [INSPIRE].

[16] COMPACT collaboration, *Cosmic topology. Part IIa. Eigenmodes, correlation matrices, and detectability of orientable Euclidean manifolds*, *JCAP* **03** (2024) 036 [arXiv:2306.17112] [INSPIRE].

[17] N.J. Cornish, D.N. Spergel, G.D. Starkman and E. Komatsu, *Constraining the topology of the universe*, *Phys. Rev. Lett.* **92** (2004) 201302 [astro-ph/0310233] [INSPIRE].

[18] J. Shapiro Key, N.J. Cornish, D.N. Spergel and G.D. Starkman, *Extending the WMAP Bound on the Size of the Universe*, *Phys. Rev. D* **75** (2007) 084034 [astro-ph/0604616] [INSPIRE].

[19] P.M. Vaudrevange, G.D. Starkman, N.J. Cornish and D.N. Spergel, *Constraints on the Topology of the Universe: Extension to General Geometries*, *Phys. Rev. D* **86** (2012) 083526 [arXiv:1206.2939] [INSPIRE].

[20] WMAP collaboration, *First year Wilkinson Microwave Anisotropy Probe (WMAP) observations: Determination of cosmological parameters*, *Astrophys. J. Suppl.* **148** (2003) 175 [astro-ph/0302209] [INSPIRE].

[21] WMAP collaboration, *Nine-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Cosmological Parameter Results*, *Astrophys. J. Suppl.* **208** (2013) 19 [arXiv:1212.5226] [INSPIRE].

[22] PLANCK collaboration, *Planck 2013 results. Part XXVI. Background geometry and topology of the Universe*, *Astron. Astrophys.* **571** (2014) A26 [arXiv:1303.5086] [INSPIRE].

[23] PLANCK collaboration, *Planck 2015 results. Part XVIII. Background geometry and topology of the Universe*, *Astron. Astrophys.* **594** (2016) A18 [arXiv:1502.01593] [INSPIRE].

[24] T. Souradeep, D. Pogosian and J.R. Bond, *Probing cosmic topology using CMB anisotropy*, in the proceedings of the *33rd Rencontres de Moriond: Fundamental Parameters in Cosmology*, Les Arcs, France, 17–24 January 1998, astro-ph/9804042 [INSPIRE].

[25] M. Hitchman, *Geometry with an Introduction to Cosmic Topology*, Jones and Bartlett Publishers (2009).

[26] A. Riazuelo, J. Weeks, J.-P. Uzan, R. Lehoucq and J.-P. Luminet, *Cosmic microwave background anisotropies in multi-connected flat spaces*, *Phys. Rev. D* **69** (2004) 103518 [astro-ph/0311314] [INSPIRE].

[27] W. Thurston and S. Levy, *Three-Dimensional Geometry and Topology. Volume 1*, in *Princeton Mathematical Series* **35**, Princeton University Press (2014).

[28] M. Kunz, N. Aghanim, L. Cayon, O. Forni, A. Riazuelo and J.P. Uzan, *Constraining topology in harmonic space*, *Phys. Rev. D* **73** (2006) 023511 [astro-ph/0510164] [INSPIRE].

[29] R. Aurich and F. Steiner, *Betti Functionals as Probes for Cosmic Topology*, *Universe* **10** (2024) 190 [arXiv:2403.09221] [INSPIRE].

[30] A. Lewis, A. Challinor and A. Lasenby, *Efficient computation of CMB anisotropies in closed FRW models*, *Astrophys. J.* **538** (2000) 473 [astro-ph/9911177] [INSPIRE].

[31] A. Lewis and A. Challinor, *CAMB: Code for Anisotropies in the Microwave Background*, Astrophysics Source Code Library, ascl:1102.026 (2011).

[32] Planck collaboration, *Planck 2018 results. Part VI. Cosmological parameters*, *Astron. Astrophys.* **641** (2020) A6 [*Erratum ibid.* **652** (2021) C4] [arXiv:1807.06209] [INSPIRE].

[33] A. Zonca et al., *healpy: equal area pixelization and spherical harmonics transforms for data on the sphere in Python*, *J. Open Source Softw.* **4** (2019) 1298 [INSPIRE].

[34] K.M. Górski et al., *HEALPix — A Framework for high resolution discretization, and fast analysis of data distributed on the sphere*, *Astrophys. J.* **622** (2005) 759 [astro-ph/0409513] [INSPIRE].

[35] E. Wigner and H. Massey, *Group Theory: And Its Application to the Quantum Mechanics of Atomic Spectra*, Elsevier Science (2013).

[36] D. Watkins, *Fundamentals of Matrix Computations*, in *Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts*, Wiley (2004).

[37] S. Mukherjee and T. Souradeep, *Statistically anisotropic Gaussian simulations of the CMB temperature field*, *Phys. Rev. D* **89** (2014) 063013 [arXiv:1311.5837] [INSPIRE].

[38] O. Fabre, S. Prunet and J.-P. Uzan, *Topology beyond the horizon: how far can it be probed?*, *Phys. Rev. D* **92** (2015) 043003 [arXiv:1311.3509] [INSPIRE].

[39] M. Defferrard, M. Milani, F. Gusset and N. Perraudin, *DeepSphere: a graph-based spherical CNN*, arXiv:2012.15000.

[40] G. Biau, *Analysis of a Random Forests Model*, arXiv:1005.0208.

[41] R. Genuer, J.-M. Poggi, C. Tuleau-Malot and N. Villa-Vialaneix, *Random Forests for Big Data*, arXiv:1511.08327.

[42] G. Biau and E. Scornet, *A Random Forest Guided Tour*, arXiv:1511.05741.

[43] F. Pedregosa et al., *Scikit-learn: Machine Learning in Python*, *J. Mach. Learn. Res.* **12** (2011) 2825 [arXiv:1201.0490] [INSPIRE].

[44] T. Chen and C. Guestrin, *XGBoost: A Scalable Tree Boosting System*, arXiv:1603.02754 [DOI:10.1145/2939672.2939785] [INSPIRE].

[45] xgboost developers, *Xgboost classifier: Python api reference*, (2022), https://xgboost.readthedocs.io/en/stable/python/python_api.html.

[46] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, (2015), https://www.tensorflow.org/.

[47] F. Chollet et al., *Keras*, (2015), https://keras.io.

[48] K. He, X. Zhang, S. Ren and J. Sun, *Deep Residual Learning for Image Recognition*, arXiv:1512.03385 [DOI:10.1109/CVPR.2016.90] [INSPIRE].

[49] J.A. Barrachina, *NEGU93/cvnn: Fixed Max Pooling* 2*D* Zenodo (2021) [DOI:10.5281/zenodo.4452131].

[50] A. Hirose and S. Yoshida, *Generalization Characteristics of Complex-Valued Feedforward Neural Networks in Relation to Signal Coherence*, *IEEE Trans. Neural Networks Learn. Syst.* **23** (2012) 541.

[51] J.A. Barrachina, C. Ren, C. Morisseau, G. Vieillard and J.-P. Ovarlez, *Complex-Valued vs. Real-Valued Neural Networks for Classification Perspectives: An Example on Non-Circular Data*, arXiv:2009.08340.

[52] D. Ribli, B.A. Pataki, J.M. Zorrilla Matilla, D. Hsu, Z. Haiman and I. Csabai, *Weak lensing cosmology with convolutional neural networks on noisy data*, Mon. Not. Roy. Astron. Soc. **490** (2019) 1843 [arXiv:1902.03663] [INSPIRE].

[53] I. Ahern, A. Noack, L. Guzman-Nateras, D. Dou, B. Li and J. Huan, *NormLime: A New Feature Importance Metric for Explaining Deep Neural Networks*, arXiv:1909.04200.

[54] M. Wojtas and K. Chen, *Feature Importance Ranking for Deep Learning*, arXiv:2010.08973.

[55] K.H. Lee, C. Park, J. Oh and N. Kwak, *LFI-CAM: Learning Feature Importance for Better Visual Explanation*, arXiv:2105.00937.

[56] D. Marcos, M. Volpi and D. Tuia, *Learning rotation invariant convolutional filters for texture classification*, arXiv:1604.06720 [DOI:10.1109/ICPR.2016.7899932].

[57] J. Kim, W. Jung, H. Kim and J. Lee, *CyCNN: A Rotation Invariant CNN using Polar Mapping and Cylindrical Convolution Layers*, arXiv:2007.10588.

[58] T. Alt, K. Schrader, J. Weickert, P. Peter and M. Augustin, *Designing Rotationally Invariant Neural Networks from PDEs and Variational Methods*, arXiv:2108.13993.

[59] H. Mo and G. Zhao, *RIC-CNN: Rotation-Invariant Coordinate Convolutional Neural Network*, arXiv:2211.11812.

[60] C.J. Copi, D. Huterer and G.D. Starkman, *Multipole vectors — A New representation of the CMB sky and evidence for statistical anisotropy or non-Gaussianity at $2 \leq \ell \leq 8$*, Phys. Rev. D **70** (2004) 043515 [astro-ph/0310511] [INSPIRE].

[61] P. Bielewicz and A. Riazuelo, *The study of topology of the universe using multipole vectors*, Mon. Not. Roy. Astron. Soc. **396** (2009) 609 [arXiv:0804.2437] [INSPIRE].

[62] R.A. Oliveira, T.S. Pereira and M. Quartin, *CMB statistical isotropy confirmation at all scales using multipole vectors*, Phys. Dark Univ. **30** (2020) 100608 [arXiv:1812.02654] [INSPIRE].

[63] T. Charnock, G. Lavaux and B.D. Wandelt, *Automatic physical inference with information maximizing neural networks*, Phys. Rev. D **97** (2018) 083004 [arXiv:1802.03537] [INSPIRE].

[64] J. Alsing, B. Wandelt and S. Feeney, *Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology*, Mon. Not. Roy. Astron. Soc. **477** (2018) 2874 [arXiv:1801.01497] [INSPIRE].

[65] scikit-learn developers, *Scikit-learn random forest classifier*, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[66] S. Allamy and A.L. Koerich, *1D CNN Architectures for Music Genre Classification*, arXiv:2105.07302.

[67] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj and D.J. Inman, *1D convolutional neural networks and applications: A survey*, Mech. Syst. Signal Process. **151** (2021) 107398.

[68] F. Mattioli, C. Porcaro and G. Baldassarre, *A 1D CNN for high accuracy classification and transfer learning in motor imagery EEG-based brain-computer interface*, J. Neural Eng. **18** (2021) 066053.

[69] S. Iglesias Álvarez, E. Díez Alonso, M.L. Sánchez, J. Rodríguez Rodríguez, F. Sánchez Lasheras and F.J. de Cos Juez, *One-dimensional Convolutional Neural Networks for Detecting Transiting Exoplanets*, arXiv:2312.07161 [DOI:10.3390/axioms12040348].

[70] I. Bello et al., *Revisiting ResNets: Improved Training and Scaling Strategies*, arXiv:2103.07579.

[71] S.E. Hong, S. Park, M.J. Jee, D. Bak and S. Cha, *Weak-lensing Mass Reconstruction of Galaxy Clusters with a Convolutional Neural Network*, Astrophys. J. **923** (2021) 266 [arXiv:2102.05403] [INSPIRE].

[72] C. Liu, Z. Zhang, J. Li, Y. Li and Z. Zou, *Recognition of Astronomical Strong Gravitational Lens System Based on Deep Learning*, in the proceedings of the *2021 9th International Conference on Intelligent Computing and Wireless Optical Communications (ICWOC)*, Chongqing, China, 4–7 June 2021, pp. 58–63 [DOI:10.1109/icwoc52624.2021.9529967].

[73] P. Jia, Q. Liu and Y. Sun, *Detection and Classification of Astronomical Targets with Deep Neural Networks in Wide-field Small Aperture Telescopes*, Astron. J. **159** (2020) 212 [arXiv:2002.09211].

[74] P. Gorla, *ResNet-50*, Wikimedia Commons (2021), https://commons.wikimedia.org/wiki/File:ResNet50.png.

[75] O. Russakovsky et al., *ImageNet Large Scale Visual Recognition Challenge*, arXiv:1409.0575 [INSPIRE].

[76] Ohio Supercomputer Center, Columbus OH, U.S.A. (1987), http://osc.edu/ark:/19495/f5s1ph73

[77] J.A. Barrachina, *Complex-Valued Neural Network (CVNN)*, version 1.2.22 (2022), https://complex-valued-neural-networks.readthedocs.io/en/latest/.