Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **487,** 104–122 (2019) doi:10.1093/mnras/stz972
Advance Access publication 2019 April 8

On the dissection of degenerate cosmologies with machine learning

Julian Merten,[1★] Carlo Giocoli,[1,2,3,4] Marco Baldi,[1,3,4] Massimo Meneghetti,[1,3,4] Austin Peel,[5] Florian Lalande,[5,6] Jean-Luc Starck[5] and Valeria Pettorino[5]

[1]*INAF–Osservatorio di Astrofisica e Scienza dello Spazio di Bologna, Via Gobetti 93/3, I-40129 Bologna, Italy*
[2]*Dipartimento di Fisica e Scienze della Terra, Università degli Studi di Ferrara, via Saragat 1, I-44122 Ferrara, Italy*
[3]*Dipartimento di Fisica e Astronomia, Alma Mater Studiorum Università di Bologna, via Gobetti 93/2, I-40129 Bologna, Italy*
[4]*INFN–Sezione di Bologna, viale Berti Pichat 6/2, I-40127 Bologna, Italy*
[5]*AIM, CEA, CNRS, Université Paris-Saclay, Université Paris Diderot, Sorbonne Paris Cité, F-91191 Gif-sur-Yvette, France*
[6]*ENSAI, rue Blaise Pascal, F-35170 Bruz, France*

Accepted 2019 March 26. Received 2019 March 26; in original form 2018 October 26

ABSTRACT

Based on the DUSTGRAIN-*pathfinder* suite of simulations, we investigate observational degeneracies between nine models of modified gravity and massive neutrinos. Three types of machine learning techniques are tested for their ability to discriminate lensing convergence maps by extracting dimensional reduced representations of the data. Classical map descriptors such as the power spectrum, peak counts, and Minkowski functionals are combined into a joint feature vector and compared to the descriptors and statistics that are common to the field of digital image processing. To learn new features directly from the data, we use a convolutional neural network (CNN). For the mapping between feature vectors and the predictions of their underlying model, we implement two different classifiers; one based on a nearest-neighbour search and one that is based on a fully connected neural network. We find that the neural network provides a much more robust classification than the nearest-neighbour approach and that the CNN provides the most discriminating representation of the data. It achieves the cleanest separation between the different models and the highest classification success rate of 59 per cent for a single source redshift. Once we perform a tomographic CNN analysis, the total classification accuracy increases significantly to 76 per cent with no observational degeneracies remaining. Visualizing the filter responses of the CNN at different network depths provides us with the unique opportunity to learn from very complex models and to understand better why they perform so well.

Key words: gravitation – neutrinos – methods: numerical – large-scale structure of Universe.

1 INTRODUCTION

The standard Lambda cold dark matter (ΛCDM) cosmological model – based on a cosmological constant as the source of the observed accelerated cosmic expansion (Riess et al. 1998; Schmidt et al. 1998; Perlmutter et al. 1999) and on cold dark matter particles as the bulk of the clustering mass in the Universe (White & Rees 1978; White 1993, 1996; Springel et al. 2005) – has survived the past two decades of cosmological observations targeted to a wide range of independent probes. This includes the statistical properties of cosmic microwave background anisotropies (Bennett et al. 2013; Planck Collaboration VI 2018), the large-scale distribution and dynamics of visible galaxies (Parkinson et al. 2012; Alam et al. 2017; Pezzotta et al. 2017), weak gravitational lensing signals (Fu et al. 2008; Hildebrandt et al. 2017; Joudaki et al. 2017; Hikage et al. 2018; Troxel et al. 2018), the abundance of galaxy clusters, as well as its time evolution (Vikhlinin et al. 2009; Planck Collaboration XXIV 2016b).

Despite this astonishing success, the fundamental nature of the two main ingredients of the ΛCDM model – summing up to about 95 per cent of the total energy density of the Universe – remains unknown. On one side, the energy scale associated with the cosmological constant does not find any reasonable explanation in the context of fundamental physics, with predictions based on the standard model of particle physics failing by tens of orders of magnitude. On the other hand, no clear detection – direct or indirect – of any new fundamental particle that may be associated with cold dark matter has been made despite a longstanding chase through astrophysical observations (Aartsen et al. 2013; Ackermann et al. 2017; Albert et al. 2017) and laboratory experiments (see e.g. ATLAS Collaboration 2013; CMS Collaboration 2016; Bernabei et al. 2018).

This leaves the next generation of cosmological observations with the arduous challenge of clarifying the fundamental nature

★ E-mail: julian.merten@inaf.it

of the dark sector by systematically scrutinizing the huge wealth of high-quality data that will be made available in the near future by several wide-field surveys (such as Ivezic et al. 2008; Laureijs et al. 2011; Benitez et al. 2014; Spergel et al. 2015). As a matter of fact, any possible insights from future data sets must come in the form of very small deviations from the expectations of the ΛCDM model, otherwise past observations would have already detected them. This suggests that either the fundamental physics behind dark energy and dark matter is indeed extremely close to that of general relativity (GR) with a cosmological constant and heavy fundamental particles with negligible thermal velocities, respectively, or that a more radical shift from this standard paradigm is hidden and masked by other effects such as an observational degeneracy with some not yet fully constrained cosmological parameter. The latter scenario may result in a severe limitation of the discriminating power of observations, thereby providing a particularly challenging test bed for the next generation of cosmological surveys.

A typical example of such a possible intriguing situation is given by the well-known degeneracy between some modified gravity (MG; see e.g. Amendola et al. 2018, for a recent review on a wide range of MG scenarios) theories and the yet unknown value of the neutrino mass. It is now generally accepted (He 2013; Motohashi, Starobinsky & Yokoyama 2013; Baldi et al. 2014; Wright, Winther & Koyama 2017) that MG theories such as $f(R)$ gravity (Buchdahl 1970) are strongly observationally degenerate with the effects of massive neutrinos on structure formation (see Baldi et al. 2014). Some commonly adopted statistics such as the matter autopower spectrum (Giocoli, Baldi & Moscardini 2018a), the lensing convergence power spectrum (Peel et al. 2018a), and the halo mass function (Hagstotz et al. 2018) may hardly distinguish standard ΛCDM expectations from some specific combinations of the $f(R)$ gravity parameters and the total neutrino mass.

As such degeneracies extend down to the non-linear regime of structure formation, the use of full numerical simulations currently represents the only viable method to explore these scenarios, even though alternative approaches based on approximate methods (see e.g. Wright et al. 2017) have been developed in the last years and are being tested and calibrated against simulations. In this work, we will explore the prospects of using machine learning techniques applied to numerical simulations of both MG and ΛCDM cosmologies that are highly observationally degenerate through standard observational statistics.

Several variants of higher order statistics have been applied in the past to characterize cosmological data sensitive to the late-time evolution of structure in the Universe. Recent analyses of the weak lensing (WL; Bartelmann & Schneider 2001) data from CFHTLens (Heymans et al. 2012) used either higher order (>2) moments of the convergence field (Van Waerbeke et al. 2013), or Minkowski functionals (Petri et al. 2015) to draw cosmological inference from a data description that goes beyond two-point statistics. Martinet et al. (2018) and Shan et al. (2018) applied peak count statistics (Dietrich & Hartlap 2010; Kratochvil, Haiman & May 2010) to shear and convergence fields from KiDS (Hildebrandt et al. 2017), and Gruen et al. (2018) used counts-in-cells (Friedrich et al. 2018) to extract information from the DES (Abbott et al. 2018) catalogues. A new set of techniques based on deep learning (LeCun, Bengio & Hinton 2015) currently has gained momentum in many scientific fields, including astrophysics. The extremely complex models that can be constructed through a modular building-block concept (e.g. Chollet 2017) have been very successful for tasks like language translation (e.g. Johnson et al. 2016; Wu et al. 2016), text and handwriting recognition (e.g. Graves 2013), as well as for the classification of images (starting with the seminal work of Krizhevsky, Sutskever & Hinton 2012). In cosmology, deep learning is used for the extraction of information from N-body simulations (Ravanbakhsh et al. 2017), to learn the connection between initial conditions and the final shape of structure (Lucie-Smith et al. 2018), for the characterization of point spread functions (Herbel et al. 2018) or the measurement of shear for WL (Springer et al. 2018), the characterization of non-Gaussian structure in mass maps (Gupta et al. 2018), the determination of galaxy cluster X-ray masses (Ntampaka et al. 2018), and the fast creation of simulated data using generative adversarial networks (Rodriguez et al. 2018). In this work, we will use such techniques to break the degeneracies between models of MG in the presence of massive neutrinos.

The text is organized as follows. Section 2 gives an overview of the numerical simulations and the creation of the mass maps that constitute our main data set. In Section 3, we introduce the different characterization and classification techniques that we apply to the mass map data and show the results that they produce in Section 4. We present our conclusions in Section 5. Two appendices provide more details on certain technical aspects of the computer vision (Appendix A) and deep neural network (Appendix B) methods we are using.

2 NUMERICAL SIMULATIONS

We perform our analysis on a set of WL maps extracted from a suite of cosmological dark matter-only simulations called the DUSTGRAIN-*pathfinder* runs. These simulations represent a preliminary calibration sample for the DUSTGRAIN (Dark Universe Simulations to Test GRAvity In the presence of Neutrinos) project, an ongoing numerical effort aimed at investigating cosmological models characterized by a modification of the laws of gravity from their standard GR form and by a non-negligible fraction of the cosmic matter density being made of standard massive neutrinos.

2.1 DUSTGRAIN-*pathfinder*

The modification of gravity considered in the DUSTGRAIN project consists in an $f(R)$ model defined by the action (Buchdahl 1970)

$$S = \int \mathrm{d}^4x \sqrt{-g} \left(\frac{R + f(R)}{16\pi G} + \mathcal{L}_\mathrm{m} \right). \tag{1}$$

We assume a specific analytical form for the $f(R)$ function (Hu & Sawicki 2007)

$$f(R) = -m^2 \frac{c_1 \left(\frac{R}{m^2}\right)^n}{c_2 \left(\frac{R}{m^2}\right)^n + 1}, \tag{2}$$

where R is the Ricci scalar curvature and $m^2 \equiv H_0^2 \Omega_\mathrm{M}$ is a mass scale, while $\{c_1, c_2, n\} \geq 0$ are free parameters of the model. Such a form is particularly popular and widely investigated as it allows one to recover with arbitrary precision a ΛCDM background expansion history by choosing $c_1/c_2 = 6\Omega_\Lambda/\Omega_\mathrm{M}$. Here, Ω_Λ and Ω_M are the vacuum and matter energy density, respectively, under the condition $c_2(R/m^2)^n \gg 1$, so that the scalar field f_R takes the approximate form

$$f_R \approx -n \frac{c_1}{c_2^2} \left(\frac{m^2}{R} \right)^{n+1}. \tag{3}$$

By restricting to the case $n = 1$, the only remaining free parameter of the model can be written as

$$f_{R0} \equiv -\frac{1}{c_2} \frac{6\Omega_\Lambda}{\Omega_\mathrm{M}} \left(\frac{m^2}{R_0} \right)^2 \tag{4}$$

Table 1. The subset of the DUSTGRAIN-*pathfinder* simulations considered in this work with their specific parameters. f_{R0} represents the MG parameter, m_ν and $m_\nu^{\rm p}$ the neutrino mass in electron volts and in $M_\odot/h$ as implemented in the simulation, $m_{\rm CDM}^{\rm p}$ cold dark matter particle mass, and $\Omega_{\rm CDM}$ and Ω_ν the CDM and neutrino density parameters, respectively.

Simulation name	Gravity type	f_{R0}	m_ν (eV)	$\Omega_{\rm CDM}$	Ω_ν	$m_{\rm CDM}^{\rm p}$ ($M_\odot\,h^{-1}$)	$m_\nu^{\rm p}$ ($M_\odot\,h^{-1}$)
ΛCDM	GR	–	0	0.313 45	0	8.1×10^{10}	0
f_4	$f(R)$	-1×10^{-4}	0	0.313 45	0	8.1×10^{10}	0
f_5	$f(R)$	-1×10^{-5}	0	0.313 45	0	8.1×10^{10}	0
f_6	$f(R)$	-1×10^{-6}	0	0.313 45	0	8.1×10^{10}	0
$f_4^{0.3}$	$f(R)$	-1×10^{-4}	0.3	0.306 30	0.007 15	7.92×10^{10}	1.85×10^{9}
$f_5^{0.15}$	$f(R)$	-1×10^{-5}	0.15	0.309 87	0.003 58	8.01×10^{10}	9.25×10^{8}
$f_5^{0.1}$	$f(R)$	-1×10^{-5}	0.1	0.311 07	0.002 38	8.04×10^{10}	6.16×10^{8}
$f_6^{0.1}$	$f(R)$	-1×10^{-6}	0.1	0.311 07	0.002 38	8.04×10^{10}	6.16×10^{8}
$f_6^{0.06}$	$f(R)$	-1×10^{-6}	0.06	0.312 02	0.001 43	8.07×10^{10}	3.7×10^{8}

and its absolute value $|f_{R0}|$ will quantify how much the model departs from GR.

The DUSTGRAIN-*pathfinder* simulations have been devised to sample the $\{f_{R0},\ m_\nu\}$ parameter space and to identify highly degenerate combinations of parameters. Some analyses of the corresponding WL signal have been presented by Giocoli et al. (2018a) and Peel et al. (2018a), while Hagstotz et al. (2018) have used the simulations to calibrate a theoretical modelling of the halo mass function in $f(R)$ gravity with and without the contribution of massive neutrinos. In this further paper, we will use machine learning techniques to tackle the issue of observational degeneracy in these combined models based on the WL reconstruction described in Giocoli et al. (2018a). A similar approach, focused on a subset of particularly degenerate models, is presented in Peel et al. (2018b).

From a technical point of view, the DUSTGRAIN-*pathfinder* runs are cosmological collisionless simulations including 768^3 dark matter particles of mass $m_{\rm CDM} = 8.1 \times 10^{10}\ M_\odot\,h^{-1}$ (for the case of $m_\nu = 0$) and as many neutrino particles (for the case of $m_\nu > 0$) in a $(750\,{\rm Mpc}/h)^3$ cosmological volume with periodic boundary conditions evolving under the effect of a gravitational interaction defined by equation (1). The simulations have been performed with the MG-GADGET code (see Puchwein, Baldi & Springel 2013), a modified version of the GADGET code (Springel 2005) that implements all the modifications that characterize $f(R)$ gravity (see Puchwein et al. 2013, for more details on the algorithm). MG-GADGET has been extensively tested (see e.g. the MG code comparison project described in Winther et al. 2015) and employed recently for a wide variety of applications (Arnold, Puchwein & Springel 2014; Arnold, Puchwein & Springel 2015; Arnold, Springel & Puchwein 2016; Arnold et al. 2019; Baldi & Villaescusa-Navarro 2018; Naik et al. 2018; Roncarelli, Baldi & Villaescusa-Navarro 2018). For the DUSTGRAIN-*pathfinder* simulations, as was already done in Baldi et al. (2014), we have combined the MG-GADGET solver with the particle-based implementation of massive neutrinos developed by Viel, Haehnelt & Springel (2010). This allowed us to include massive neutrinos in the simulations as an independent family of particles with its own initial transfer function and velocity distribution. Initial conditions have been generated following the approach of, e.g. Zennaro et al. (2017) and Villaescusa-Navarro et al. (2018) at the starting redshift of the simulation $z_{\rm i} = 99$ with thermal neutrino velocities added on top of the gravitational velocities by random sampling the neutrino momentum distribution at the initial redshift.

Standard cosmological parameters are set to be consistent with the Planck 2015 constraints (Planck Collaboration XIII 2016a). Concerning non-standard parameters, the DUSTGRAIN-*pathfinder* simulations spanned the range $-1 \times 10^{-4} \leq f_{R0} \leq -1 \times 10^{-6}$ for the scalar amplitude and $0\ {\rm eV} \leq m_\nu \leq 0.3\ {\rm eV}$ for the neutrino mass, for a total of 20 simulations. In this work, we will consider a subset of the full DUSTGRAIN-*pathfinder* suite consisting of nine simulations whose specifications are summarized in Table 1.

2.2 Lensing light cones

For all simulations, we stored 34 snapshots at different redshifts that allow us to construct lensing light cones up to a source redshift $z_{\rm s} = 4$ without gaps. Different methods have been developed to produce lensing light cones from large cosmological N-body simulations. Recent works have employed post-processing reconstructions based on the slicing of a set of comoving particle snapshots (as e.g. in Hilbert et al. 2008, 2009; Giocoli et al. 2016; Shirasaki et al. 2017), as well as on-the-fly algorithms capable of storing only the projected matter density on a given field of view without resorting on the flat-sky approximation (see e.g. Barreira et al. 2016; Arnold et al. 2019). In this work, we use the MAPSIM routine (Giocoli et al. 2014; Tessore et al. 2015; Castro et al. 2018), which is based on the former strategy. We use the particles stored in 21 different snapshots to construct our continuous past light cones up to $z = 4$, building 27 lens planes of the projected matter density distribution, considering a square sky coverage of 5 deg on a side. For each cosmological model, we construct 256 different light-cone realizations by randomizing the various comoving cosmological boxes (Giocoli et al. 2018a; Peel et al. 2018a).

2.3 Convergence maps

The MAPSIM pipeline is composed of two algorithms. The first one – called i-MAPSIM – constructs lensing planes from the different simulation snapshots, saving for each plane l and on each pixel with coordinate indices (i, j) the particle surface mass density Σ

$$\Sigma_l(i,\,j) = \frac{\sum_k m_k}{A_l}\,. \tag{5}$$

A_l represents the comoving pixel area of the lens plane l and $\sum_k m_k$ the sum over all particle masses associated with the given pixel. The second algorithm named ray-MAPSIM projects the matter density distribution along the line of sight by weighing the lens planes with the lensing kernel in the Born approximation regime (Bartelmann & Schneider 2001; Schäfer et al. 2012; Giocoli et al. 2016; Petri 2016; Giocoli et al. 2017, 2018b; Petri, Haiman & May 2017; Castro et al. 2018). From Σ_l, we can derive the convergence κ at a given source

redshift z_s as

$$\kappa = \sum_l \frac{\Sigma_l}{\Sigma_{\rm crit,l,s}}, \tag{6}$$

where l varies over the different lens planes with the lens redshift z_l smaller than z_s, and $\Sigma_{\rm crit,l,s}$ represents the critical surface density at the lens plane z_l for sources at redshift z_s

$$\Sigma_{\rm crit,l,s} \equiv \frac{c^2}{4\pi G}\frac{D_l}{D_s D_{ls}}. \tag{7}$$

Here, c is the speed of light, G is the Newton's constant, and D_l, D_s, and D_{ls} are the angular diameter distances between observer lens, observer source, and source lens, respectively. The final κ maps cover the 25 square deg field of view with 2048^2 pixels, resulting in a map resolution of ∼8.8 arcsec.

3 METHODOLOGY

A variety of machine learning techniques is applied to the DUSTGRAIN-*pathfinder* convergence maps. It was shown by Peel et al. (2018a) that summary statistics up to second order do not reliably separate such mass maps. Higher order statistics, especially peak counts (e.g. Peel et al. 2017; Lin & Kilbinger 2018; Martinet et al. 2018; Shan et al. 2018), do a better job but still leave room for improvement when distinguishing between a large number of models and in the presence of noise. Most commonly used methods to characterize observational data are naturally based on physical models. In the following, we present an agnostic approach, which also applies techniques and algorithms found in the fields of computer science and specifically digital image processing.

We distinguish two subsequent steps in the process of mass map classification. The first is to find a feature extraction function Θ, which takes a high-dimensional data vector x as input and finds a general, dimensional reduced representation of it in the form of a feature vector F

$$\Theta(x; w_f) = F. \tag{8}$$

The feature extraction function can have several parameters that are stored in the feature weight vector w_f. In order to arrange the data vector x in a meaningful way, we introduce an index notation x_{ijc}. The first two indices reflect a spatial ordering of the 2D data along the coordinate axes. This means that all elements with $i = 1$ are located in the first row of the pixelized image and all elements with, e.g. $j = 10$ are located in the tenth column of the image. This notation also includes 1D data, ordered or not, by setting $i = 1\ \forall j$, c. The third index c – commonly dubbed as a channel – allows us to collect multiple aspects of the same entity represented by x. For the example of an RGB image, $c = 1$ would be the red channel of the image, $c = 2$ the green, and $c = 3$ the blue channel. Finally, we define the shape of a data vector with a bracket notation. The shape of our input convergence maps is $\#x = (2048, 2048, 6)$ since we have 2048 × 2048 pixel maps with convergence values κ at six different source redshift channels and where in the above we have introduced the shape operator # that returns the shape of a data vector.

The second step classifies F into a set of target classes. The classification function ζ, which can again depend on a set of parameters w_m, should not only output a single class prediction, but rather a prediction vector P of shape $\#P = (1, 1, n)$ with probabilities to belong to each of n target classes

$$\zeta(F, w_m) = P. \tag{9}$$

It must hold that $P_n \in [0, 1]$, $\sum P_n = 1$ and in our case $n \in (1, ..., 9)$.

In the following, we explore different choices for the feature extraction and classification functions and find ways to optimize their parameters to achieve an optimal classification. We do so with the help of training sets, which are data vector–label pairs (x, y^l), meaning mass maps for which we a priori know the underlying cosmological model. Specifically, the label y^l is an indicator function for the class $l \in (1, ..., 9)$ with elements y^l_k for which

$$y^l_k = \begin{cases} 1 & \text{if} \quad k = l \\ 0 & \text{else.} \end{cases} \tag{10}$$

3.1 Definition of data sets

Our full data set consists of 256 convergence maps of shape (2048,2048,6) for each of the nine cosmological models. We split each map further into 64 smaller patches to define our main data vectors with $\#x = (256, 256, 6)$. 75 per cent of those maps (12 289) are used as a training set in order to optimize the parameters of our models. We use 15 per cent of the maps (2457) as a validation set where the correct labels y are known to us, but not to the optimization algorithm. Performing a classification on the validation data serves as quality control and helps us to decide if an optimization is successful and when to stop it. Another 10 per cent of the maps (1638) are used as a test set, where the labels are not known to us a priori and to which our trained and validated algorithms are applied to blindly. The success rates on those test sets will be the main result of this work. We provide examples of the actual data in the left-hand panel of Fig. 1, which shows example convergence maps, chosen at random from the test set, for four instructive models. This includes the ΛCDM reference, f_4 that deviates most from ΛCDM, $f_6^{0.06}$ that is observationally most degenerate with ΛCDM, and a sample map of $f_5^{0.1}$ that is between the two extremes. The source redshift for all the maps shown is $z_s = 1.0$.

3.2 Mass map feature extraction

Two important subclasses for Θ are possible. In the first, the parameters w_f are free and can be optimized during a training phase. In the second, they are fixed. We want to highlight that we do not perform any initial transformations of the data, which have proven to be useful for the analysis of lensing mass maps. It was shown in, e.g. Peel et al. (2018a) that an aperture mass transformation (Schneider 1996; Schneider et al. 1998) can largely improve the discrimination power of certain statistics, but we want to stay as general and agnostic as possible at this stage and use the raw pixel data of the convergence maps as the initial data vector.

3.2.1 Standard mass map descriptors

Examples of fixed feature extraction are the mass map descriptors that are commonly used in the cosmological community to describe convergence or shear catalogues. For the purposes of this work, such descriptors serve as the reference for other techniques that we apply. We combine a number of mass map features, which we extract with the `LensTools` package[1] by Petri (2016) into a feature vector of shape (1,1,99). The first four entries in this vector are the mean, variance, skewness, and kurtosis of the convergence

[1] https://github.com/apetri/LensTools

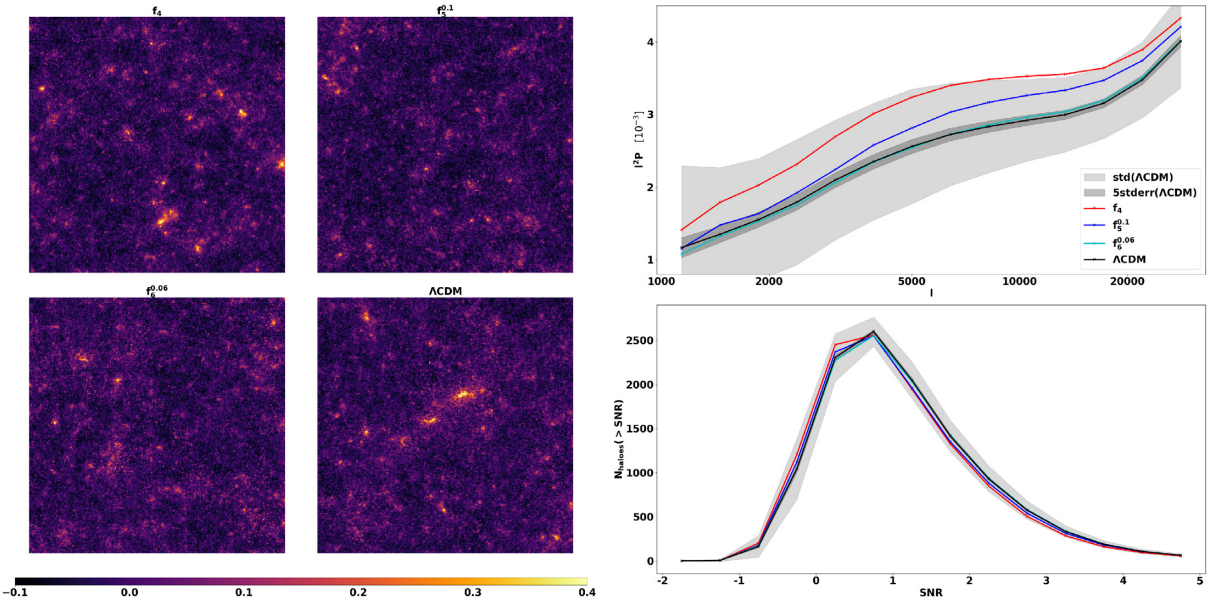

Figure 1. The *left-hand panel* shows randomly chosen convergence maps from the test set in four specifically chosen models of structure formation. Those models span the range of observational degeneracy from the ΛCDM reference, with f_4 being the most distinct, $f_6^{0.06}$ the most similar, and $f_5^{0.1}$ the middle ground between the extremes. The *right-hand panel* shows, for the same models, the average power spectrum over all maps in the test set on the *top* panel and the peak counts as a function of signal-to-noise ratio on the *bottom panel*. The coloured lines indicate the results for the four different models, while the grey shaded areas indicate typical variations within the test set sample for the case of ΛCDM. The source redshift in all cases is $z_s = 1.0$.

maps. This is followed by 11 percentiles between the 0th percentile (the minimum) and the 100th percentile (the maximum) in steps of 10 per cent. The normalized histogram of the convergence values in each map is sorted with 14 bins and the value for each bin is appended to the feature vector. Next, we calculate the power spectrum in 14 logarithmically spaced bins between $l = 1000$ and 32 000, which cover the angular size and resolution of our mass maps. Finally, we use the standard deviation of each map to define 14 signal-to-noise bins between -2 and 5. For each such bin, we calculate the peak counts, as well as the first three Minkowski functionals (e.g. Kratochvil et al. 2012; Petri et al. 2015, and references therein), which concludes our collection of 99 features. The right-hand panel of Fig. 1 shows examples for the variation between models for such classical features. Presented there are the average power spectrum and peak count for all maps in the test set and for the four instructive models we chose in Section 3.1 for data visualization purposes.

3.2.2 *Classical computer vision*

We know from Peel et al. (2018a) that at least some of the standard descriptors above are not optimally suited for the task at hand and it is, at this point, not entirely obvious how to define better ones. This is why we now aim to derive as many fixed features as possible. The publicly available `wnd-charm` algorithm (Orlov et al. 2008; Shamir et al. 2008; Shamir et al. 2010) was designed for the classification of microscopy images and derives a particularly large feature vector of shape (1,1,2919). This includes most of the common statistics and descriptors known to digital image processing. Many features are thereby not only calculated from the raw image, but also from some of its alternative representations like the Fourier, Wavelet, Chebyshev, or Edge transformation. Moreover, some features are also extracted from transformations of transformations. While we did state earlier that we do not want to vet our data with transformations, we want to point out that the listed transformations are by no means inspired by the mechanisms of lensing or structure formation. We refer the interested reader to Orlov et al. (2008) for the full description of the algorithm and the description of the full feature vector, but we do provide a short summary in appendix A and a compact overview in Table A1.

3.2.3 *Convolutional neural networks (CNNs)*

As the class of feature extraction functions that are able to change their shape during the training process, we chose multilayered neural networks (LeCun et al. 2015; Goodfellow, Bengio & Courville 2016). The input data vector x is manipulated and eventually reduced in dimension by a long – deep – chain of simple layers θ, which implement a specific mathematical operation. The output of one layer becomes the input of the following layer and contains its own set of parameters w_i. The set of all layer parameters becomes the feature parameter vector w_f.

$$\Theta(x, w_\mathrm{f}) = \theta_n \circ \theta_{n-1} \circ ... \circ \theta_1 \tag{11}$$

$$\theta_i \circ \theta_{i-1}(\cdot) \equiv \theta_i\,(\theta_{i-1}(\cdot, w_{i-1}), w_i) \tag{12}$$

$$w_\mathrm{f} \equiv \{w_i\}_{i=1}^{n} \tag{13}$$

Deep neural networks source their performance from the sheer number of layers they are comprised of and have gained much popularity in recent years. This is mainly due to the advancements in numerical performance by, e.g. exploiting many-core architectures.[2] This allows for the construction of particularly deep and complex

[2]General Purpose Graphics Processing Units (GPGPUs) are a popular example of a many-core architecture.

networks with hundreds of millions of parameters. The functional forms of the layers that are used in a deep neural network depend on the problem at hand. For image classification, CNNs have proven to be particularly useful (Krizhevsky et al. 2012; Simonyan & Zisserman 2014; Szegedy et al. 2014; He et al. 2015; Lin et al. 2017) and hence we chose this class of models for our purposes. The main functionality of a CNN is provided by a convolutional layer Conv(n, m, Δi, Δj, p, C) that applies a number of C convolutions with kernel size (n, m) to a 2D input vector I_{ijc} with $\#I_{ijc} = (X, Y, l)$. The stride parameters Δi and Δj allow one to implement dimensional reduction and the parameter p controls if the input data are padded ($p = v$) or unaltered ($p = s$). We provide a thorough mathematical definition of all deep neural network layers used in this work, including the convolutional one, in Appendix B1.

Convolution layers are often followed by pooling layers for dimensional reduction. We implement average pooling layers AvgPool(n, m, Δi, Δj, p) that average entries of the 2D data vector within a window of size (n, m), apply a stride defined by Δi and Δj, and follow the same padding scheme that was introduced earlier. Maximum pooling layers MaxPool(n, m, Δi, Δj, p) work in a similar manner but instead of the average they return the maximum within a given window. Both pooling layers exist also as global versions, indicated by GlobalMaxPool and GlobalAvgPool, where all entries per channel are considered for either the maximum or averaging operation.

Up to this point, we only allowed for layers to be placed strictly sequential. In order to implement a horizontal layout, we connect several layers to the same input and combine their results I_{ijc_1}, ..., I_{ijc_n} with the help of a concatenation layer Concatenate(I_{ijc_1}, ..., I_{ijc_n}). This concept of performing not only one operation at a given depth of the network but several has proven very successful for image classification as, e.g. shown in Szegedy et al. (2014), who dubbed such horizontal layers as inception modules.

The output of a layer can be followed by a non-linear activation function. For convolution and pooling layers, we mainly deploy rectangular linear units (ReLUs) and we give the full detail about the activation functions used in this work in Appendix B2. To avoid the network from overfitting, the so-called dropout layers are introduced as a regularization. In there, a given percentage of the elements of an input vector is chosen at random and is subsequently discarded from the output (Srivastava et al. 2014). Finally, to compensate for fluctuations in the amplitudes of input vectors at different network depths, Ioffe & Szegedy (2015) introduced the concept of batch normalization that we also use after each convolutional layer. The output of the last layer in the CNN, the feature vector F, is used for classification in a final section of the network, which is commonly referred to as top. The concrete architecture of the CNN that we use in this work is provided in Section 4.3 and Appendix B3.

3.3 Feature-based classification

We now turn our attention to the classification function $\zeta(F; w_\mathrm{m})$. We investigate two different approaches to classification. The first one is a nearest-neighbour-search scheme based on distances in feature space. The other approach, based again on a class of neural networks, uses regression through a training set to find the optimal mapping between features and labels.

3.3.1 *Feature-space distances*

In the following, we denote with T all those feature vectors that belong to a sample from the training set and with T^n the subset that belongs only to class n of the training set. We calculate a Fisher discriminant (e.g. Bishop 2006) to find suited classification weights w_m for each individual feature T_i.

$$(w_\mathrm{m})_i = \frac{\sum_{n=1}^{N} \left(\langle T_i \rangle - \langle T_i^n \rangle\right)^2}{\sum_{n=1}^{N} (\sigma_{T_i}^n)^2} \frac{N}{N-1} \tag{14}$$

Here, N is the total number of classes and $(\sigma_{T_i}^n)^2$ is the variance of the feature i within class n.

Once we found the weights w_m, we can define a weighted nearest-neighbour distance (WNN) of any feature vector F to all the classes n in the training set.

$$d_\mathrm{WNN}^n = \min_{T \in T^n} \sum_{i=1}^{M} (w_\mathrm{m})_i \, (F_i - T_i)^2 \,, \tag{15}$$

where $M = |F|$ is the length of the feature vector. The problem with this WNN distance is the fact that it is based only on a single element in the training set, the one that minimizes the sum in equation (15). To remedy this, Orlov et al. (2006) introduced a weighted neighbour distance (WND), which takes into account the distance to all elements in the training set, but largely penalizes large distances through the free parameter b

$$d_\mathrm{WND}^n = \frac{\sum_{T \in T_c} \left[\sum_{i=1}^{M} (w_c)_i \, (F_i - T_i)^2\right]^b}{|T^c|} . \tag{16}$$

Orlov et al. (2006) found that the results do not strongly depend on b once $b > 2$ and that $b = 5$ is a generally good, numerically stable, choice. The final step in order to make predictions P is to define a similarity using a distance of choice, e.g. WNN or WND, and by normalizing appropriately

$$P_n = \left(d_n \sum_{i=1}^{N} (d_i)^{-1}\right)^{-1} . \tag{17}$$

3.3.2 *Fully connected neural networks*

A different approach to the classification task is another form of neural network (equation 11). The main layer in such a neural network is a fully connected – sometimes called affine – layer FC(n), which implements a linear mapping between the input vector of length m and the output vector of length n using a matrix of nm free parameters and an additional bias parameter (see Appendix B1).

Such layers are again chained together and the last layer produces an output vector of the same length as the number of classes N. As before, in between those layers one may use dropout, activation, and normalization layers. The top of the network is followed by a specific activation function called a softmax (see Appendix B2) that provides the desired predictions P_n.

Since the optimal weights w_m are found by a regression, we define a loss function L, which in the case of this classification problem is a categorical cross entropy

$$L(x; w_\mathrm{m}) = -\sum_{n=1}^{N} y_n \log P_n(x; w_\mathrm{m}). \tag{18}$$

y are the labels for the elements in the training data x and $P_n(w_\mathrm{m})$ their class predictions given a current set of parameters w_m. In order to minimize the loss, while continuously looping over the training data, we use a specific implementation of stochastic gradient

descent called ADAM (Kingma & Ba 2014). The gradients of our model $\frac{d\zeta}{dw_m}$ are thereby calculated via a back-propagation algorithm (Rumelhart, Hinton & Williams 1986). We end this description of our methodology by noting that for a full feature extraction and classification chain $P = \zeta[\Theta(x; w_f); w_m]$, with a CNN as Θ and a neural network as classifier ζ, the classification and feature extraction weights can be optimized at the same time.

3.4 Numerical set-up

As mentioned earlier, we use the Python package lenstools[3] (Petri 2016) for the extraction of the standard map descriptors from Section 3.2.1. For the computer vision fixed features from Section 3.2.2, we slightly adapted the publicly available version of wnd-charm.[4] We altered the C++ version of the feature extraction algorithm to accept FITS files (Hanisch et al. 2001) as an input image container with pixel values as double precision floating-point numbers. We then use the feature output files of wnd-charm as an input for our own distance-based classification pipeline written in Python. We make these routines publicly available in this repository.[5] All deep learning elements of our analysis stack use the widely used tensorflow[6] framework, which uses NVIDIA's CUDNN (Chetlur et al. 2014) library to carry out tensor operations on GPUs. We pair a tensorflow backend with the high-level deep learning Python interface keras[7] as a frontend. The network training was carried out on two NVIDIA Titan Xp GPUs. All convergence maps and Jupyter[8] notebooks used to produce the results in this work are either linked to or publicly available in the aforementioned repository. In there, we refer the reader to the 'reproducible_science' folder.

4 RESULTS

Section 3 introduced a number of methods to perform mass map characterization and classification. We now present the results obtained by applying those methods and provide details on their training process with the help of the validation sets. For the most successful method, we investigate the dependence of the results on the convergence map source redshift and we end this section with a closer look at the most relevant features, which are extracted by the different methods. If not stated otherwise, the results in this section are based on training, validation, and test set maps at a source redshift $z_s = 1.0$.

4.1 Classification based on feature distance

For the case of the distance-based classifier from Section 3.3.1, the training process is just the derivation of the Fisher weights shown in equation (14). We calculate them using the training set and present the 20 top-ranked features for the classical descriptors in Table 2 and for the wnd-charm features in Table 3. For the first case, we see quite a mix of features in the top, with the power spectrum and peak counts being the most important ones. This result is nicely confirmed by the right-hand panel of Fig. 1, which shows that the bins with $5000 < l < 15\,000$ of the power spectrum indeed show a clear separation between the more degenerate models. For the wnd-charm features however, the ranking is completely dominated by Zernike coefficients on transformations of the image, with a few contributions of Haralick textures. One should keep in mind though that we extract a total of 2919 features, out of which 51 have a weight >0.1, 868 have a weight >0.01, and only 193 features have a vanishing weight. It is the combination of all the non-zero weights that will lead to the distance-based classification later on.

For the 99 standard features, we find a total classification success rate of 22 per cent, meaning that out of 14 742 samples in the test set, only 3243 were classified correctly. For some specific classes,

Table 2. The top-ranked classical mass map features according to their Fisher score (equation 14). The meaning of each feature and the explanation of its index can be found in Section 3.2.1.

Rank	Name	Index	Weight
1	Power spectrum	11	0.106
2	'	10	0.104
3	'	9	0.092
4	'	12	0.084
5	Peak counts	13	0.083
6	Power spectrum	8	0.078
7	Peak counts	12	0.078
8	Power spectrum	7	0.065
9	Peak counts	5	0.064
10	Skewness	–	0.059
11	Peak counts	14	0.055
12	Power spectrum	6	0.051
13	Percentile	100	0.051
14	Minkowski functional 1	14	0.049
15	Percentile	0	0.049
16	Power spectrum	13	0.046
17	Minkowski functional 2	14	0.045
18	Peak counts	11	0.044
19	Power spectrum	5	0.042
20	Kurtosis	–	0.042

Table 3. Same as Table 2 but for the wnd-charm features. We refer the reader to Appendix A for the definition of each feature and the exact meaning of the feature index and the transform column.

Rank	Transform	Name	Index	Weight
1	F	Zernike coefficients	20	0.285
2	F	'	42	0.270
3	F(W)	'	50	0.255
4	F(E)	'	52	0.242
5	F(W)	'	21	0.236
6	F(E)	'	39	0.214
7	F	'	12	0.205
8	F(W)	'	22	0.204
9	F(E)	'	37	0.196
10	F(W)	'	56	0.183
11	F(E)	'	5	0.174
12	F(E)	'	28	0.170
13	W	Haralick textures	5	0.169
14	F	Zernike coefficients	17	0.166
15	F(E)	'	34	0.166
16	F(E)	Haralick textures	0	0.164
17	F(E)	'	14	0.161
18	F(E)	Zernike coefficients	24	0.159
19	F	'	60	0.154
20	–	Edge features	0	0.152

[3] https://github.com/apetri/LensTools
[4] https://github.com/wnd-charm/wnd-charm
[5] https://bitbucket.org/jmerten82/mydnn
[6] https://www.tensorflow.org/
[7] https://keras.io/
[8] http://jupyter.org/

Table 4. The sequence of layers used in the neural network to classify fixed mass map features. The output shape notation follows the convention introduced in Section 3. The description of all layers can be found in Sections 3.2.3 and 3.3.2, their formal definition in Appendices B1 and B2. The numbers in square brackets refer to the case where the `wnd-charm` feature vector is used instead of the smaller vector of classical features.

Index	Layer	Free parameters	Output shape
1	Input	0	(1,1,99[2919])
2	FC(32)	3200 [93 440]	(1,1,32)
3	leakyReLU(0.03)	0	(1,1,32)
4	FC(9)	+297	(1,1,9)
5	Softmax	0	(1,1,9)
	Output	= 3497 [93 737]	9

the classification success rate is barely above the success rate for a random guess (11 per cent). The important ΛCDM class, for example, shows a success rate of 13 per cent. The picture improves marginally when using the 2919 `wnd-charm` features instead. The total classification success over all classes rises mildly to 25 per cent. While especially the three f_6 models still show success rates around or even below 11 per cent, at least some classes, including ΛCDM, are now significantly above the 20 per cent level. We do not show more details[9] on the distance-based classification since it is clear from those results already that this classification method does not qualify for a successful discrimination of our models.

4.2 Classification based on neural network

We now use the same set of fixed features but feed them into a fully connected neural network for classification. For the case of the 99 standard features, we show the very simple topology of the classification network in Table 4. The same network is used to classify the 2919 `wnd-charm` features but due the larger input vector, the number of free parameters is larger, which we indicate by a square bracket notation in the same table. The regression to find the optimal parameters of the main fully connected layers is based on the training set. In total, we train with 110 601 feature vectors of shape (1,1,99) or (1,1,2919) and where one iteration over all those elements during the regression is commonly called an epoch. Gradient evaluations and corresponding changes to the network parameters are made after a subset of an epoch, usually called a batch. The batch size in this case was set to 128. After each epoch, we evaluate the current performance of the network with the 22 113 (2457 per class) feature vectors in the validation set. Fig. 2 shows for both feature sets the evolution of the loss function for the training and validation data as a function of training epoch. For the larger feature vector, the validation loss starts to saturate around epoch 70 while the training loss keeps declining. This indicates that the network starts to overfit, meaning that it learns training-set specific features that are of no use to characterize the validation set or any data unknown to the model. This is where we stop the training and save the model parameters that produced the smallest validation loss.

The neural network classification yields significantly better results compared to the classification based on feature-space distances. In the case of the 99 standard features, the total classification rate rises to 39 per cent and to 35 per cent in the case of the `wnd-charm` features. Most interestingly, the smaller vector of 99 classical features produces better results than the much larger

[9]A full success rate analysis is provided in the repository (https://bitbucket.org/jmerten82/mydnn) associated with this article.

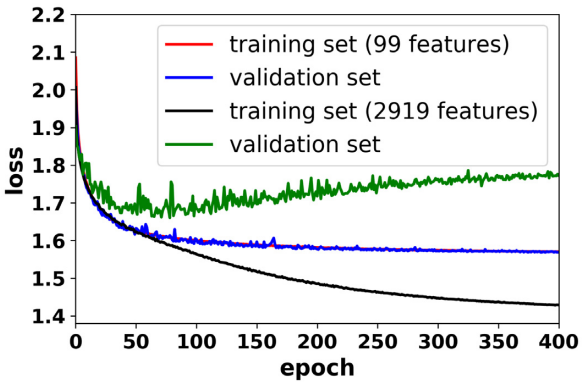

Figure 2. The evolution of the loss as a function of epoch for the training of the neural network. Shown are both cases where either a smaller vector of standard is the input for the network, or a larger set of `wnd-charm` features.

feature vector provided by `wnd-charm`. Some of the most discriminative features from the computer vision method shown in Table 3 are certainly describing the data well and should be used in future analyses; however, once the information from all standard descriptors such as the binned power spectrum, peak counts, and Minkowski functionals is combined in an optimal way by a neural network, there is no advantage in using features that are inspired by only computer vision. Table 5 shows the classification success matrix for the standard features, where each row refers to a subset of the test data comprising only maps from that true class labelled by the first column. The first number in each block of four shows how many times the 1638 members of this subset have been sorted into the respective predicted class, which is indicated by the label in the very first row. The second number is the percentage of predictions with respect to the total number of maps in the class. The third and fourth numbers are the mean and its standard error on the prediction probability for all maps in the subset given by the row and for the class predictions indicated by the label of the column. For an optimal classification, only the diagonal of this matrix (those fields typeset in boldface) would show non-zero values.

While Table 5 gives a good indication of what to expect from a classification of single maps, only the mean and its standard error on the class predictions give an idea on how well the full ensemble of test set maps is classified. We therefore further evaluate the statistics of the prediction vectors for each true test set class. Fig. 3 shows nine panels of box plots, each of which represents the statistics for one such subset. The black box in each panel represents the correct predictions, equivalent to the bold diagonal of Table 5. The horizontal line spanning each panel is the median for all the true class predictions and the error band shows the scatter of medians derived from 1000[10] bootstrap samples. The upper and lower ends of each box show the 75th and 25th percentiles, respectively, and the whiskers show the outlier cleaned minimum and maximum value of the class predictions. Whenever a box that is not the true label is shown in green, it means that the median and its errors, indicated by the notches of each box, are lower than the one of the correct prediction box and do not overlap with its horizontal error band. If those criteria are not met, the respective box is shown in red.

When looking at the results in Table 5 and Fig. 3, the following observations catch the eye. Although the overall classification

[10]This number is of course arbitrary but is close to the sample size and we also checked that the bootstrap-derived error does not depend significantly on the number of bootstraps.

Table 5. The classification success matrix for the neural-network-based classification of the classical features. Each row represents a different subset of the test data indicated by the first column. The first number in each block of four in a column is the number of samples in the subset that was assigned to the predicted class indicated by the column label on the top. The second number is the relative classification success rate for the subset. The third number is the mean of all predictions in the subset and the fourth number is its standard error. The success rates indicate that only the two f_4 models and to a lesser degree f_5, $f_5^{0.15}$, and ΛCDM are well separated from the other models with correct classification rates of 40 per cent or above and false classification rates for other models of 17 per cent or less. $f_5^{0.1}$ and the two f_6 models with non-vanishing neutrino mass are basically undistinguished from other models and f_6 (success rate 31 per cent) shows still a large degeneracy with ΛCDM (15 per cent misclassification rate).

	f_4	$f_4^{0.3}$	f_5	$f_5^{0.15}$	$f_5^{0.1}$	f_6	$f_6^{0.06}$	$f_6^{0.1}$	ΛCDM
f_4	**958**	80	157	103	97	137	26	24	56
	58 per cent	5 per cent	10 per cent	6 per cent	6 per cent	8 per cent	2 per cent	1 per cent	3 per cent
	0.376	0.052	0.122	0.083	0.11	0.087	0.068	0.056	0.046
	±0.007	±0.003	±0.003	±0.002	±0.002	±0.002	±0.002	±0.001	±0.002
$f_4^{0.3}$	70	**1135**	1	72	10	18	31	116	185
	4 per cent	**69 per cent**	0 per cent	4 per cent	1 per cent	1 per cent	2 per cent	7 per cent	11 per cent
	0.05	**0.468**	0.006	0.052	0.018	0.058	0.091	0.12	0.137
	±0.002	**±0.007**	±0.001	±0.002	±0.001	±0.001	±0.002	±0.002	±0.003
f_5	232	6	**857**	158	275	84	10	5	11
	14 per cent	0 per cent	**52 per cent**	10 per cent	17 per cent	5 per cent	1 per cent	0 per cent	1 per cent
	0.129	0.008	**0.354**	0.129	0.246	0.055	0.036	0.026	0.018
	±0.004	±0.001	**±0.005**	±0.003	±0.002	±0.002	±0.001	±0.001	±0.001
$f_5^{0.15}$	149	90	173	**651**	254	120	38	52	111
	9 per cent	5 per cent	11 per cent	**40 per cent**	16 per cent	7 per cent	2 per cent	3 per cent	7 per cent
	0.082	0.052	0.129	**0.229**	0.182	0.087	0.086	0.08	0.073
	±0.003	±0.003	±0.003	**±0.003**	±0.003	±0.002	±0.002	±0.002	±0.002
$f_5^{0.1}$	223	22	485	333	**413**	98	18	14	32
	14 per cent	1 per cent	30 per cent	20 per cent	**25 per cent**	6 per cent	1 per cent	1 per cent	2 per cent
	0.113	0.02	0.249	0.18	**0.243**	0.067	0.054	0.043	0.031
	±0.003	±0.001	±0.005	±0.003	**±0.003**	±0.002	±0.001	±0.001	±0.001
f_6	193	103	44	181	65	**512**	126	171	243
	12 per cent	6 per cent	3 per cent	11 per cent	4 per cent	**31 per cent**	8 per cent	10 per cent	15 per cent
	0.091	0.061	0.048	0.094	0.065	**0.192**	0.17	0.155	0.123
	±0.003	±0.003	±0.002	±0.002	±0.002	**±0.003**	±0.002	±0.002	±0.003
$f_6^{0.06}$	128	194	30	163	39	377	**147**	257	303
	8 per cent	12 per cent	2 per cent	10 per cent	2 per cent	23 per cent	**9 per cent**	16 per cent	18 per cent
	0.07	0.095	0.032	0.087	0.05	0.169	**0.176**	0.176	0.146
	±0.003	±0.003	±0.002	±0.002	±0.002	±0.002	**±0.002**	±0.002	±0.003
$f_6^{0.1}$	71	262	18	164	39	279	109	**345**	351
	4 per cent	16 per cent	1 per cent	10 per cent	2 per cent	17 per cent	7 per cent	**21 per cent**	21 per cent
	0.049	0.12	0.023	0.086	0.042	0.154	0.176	**0.189**	0.162
	±0.002	±0.004	±0.001	±0.002	±0.002	±0.002	±0.002	**±0.002**	±0.003
ΛCDM	69	265	5	135	9	158	60	166	**771**
	4 per cent	16 per cent	0 per cent	8 per cent	1 per cent	10 per cent	4 per cent	10 per cent	**47 per cent**
	0.043	0.14	0.014	0.074	0.028	0.12	0.146	0.164	**0.271**
	±0.002	±0.004	±0.001	±0.002	±0.001	±0.002	±0.002	±0.002	**±0.004**

success rate is only 39 per cent, none of the classes is classified incorrectly as an ensemble. In the case of the two f_4 models, we see a clear separation between the correct predictions from the other classes. This is confirmed by the classification matrix, which shows no substantial overlap (> 11 per cent) with any other model. This however changes for the three f_5 and three f_6 models. Although the median for the correct predictions is the highest for all of the models,[11] the degeneracies within the same model of gravity are strong in those cases as one can see from the basically equal heights of the centres of the boxes in Fig. 3 and from the classification matrix, which lists a large number of misclassifications up to 30 per cent in the case of $f_5^{0.1}$ misclassified as f_5. A lot more severe is the case of the three f_6 models. For them, we find substantial overlap of up to 21 per cent with ΛCDM. Even the predictions for ΛCDM itself are not completely separate from the three f_6 models and $f_4^{0.3}$ with an overlap of up to 16 per cent.

[11] The mean is not in the case of $f_5^{0.1}$.

4.3 CNN

The CNN extracts the characterizing features directly from the pixel data of the training mass maps. We have experimented with a number of architectures, including classic topologies that

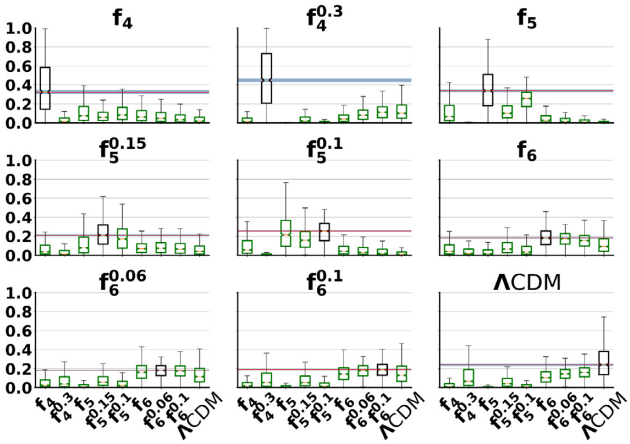

Figure 3. The prediction statistics of the classical feature vector classified by a neural network. Each labelled panel represents all predictions for one true class of the test set. In every panel, each box summarizes the statistics of the model predictions indicated by the bottom labels. The median and its bootstrap error for the correct prediction are shown by the red line with error band. For this method, only the two f_4 models are clearly distinguished from the other models. The f_5 and f_6 models, especially, remain largely degenerate within themselves, but also with ΛCDM.

Table 6. The sequential structure of the CNN used in this work. All layers marked by '∗' are batch normalized. More complicated inception layers are shown in the respective figure.

	Layer	Free parameters	Output shape
1	Input	0	(256,256,1)
2	Conv(3,3,2,2,v,32)*	288	(127,127,32)
3	lReLU(0.03)	0	(127,127,32)
4	Conv(3,3,1,1,v,32)*	+9216	(125,125,32)
5	lReLU(0.03)	0	(125,125,32)
6	Conv(3,3,1,1,s,64)*	+9216	(125,125,32)
7	StemInception* (Fig. B1)	+555 008	(29,29,384)
8	InceptionA* (Fig. B2)	+316 416	(29,29,384)
9	ReductionA* (Fig. B3)	+2304 000	(14,14,1024)
10	InceptionB* (Fig. B4)	+2931 712	(14,14,1024)
11	ReductionB* (Fig. B5)	+2744 320	(6,6,1536)
12	InceptionC* (Fig. B6)	+4546 560	(6,6,1536)
13	GlobalAvgPool	0	(1,1,1536)
14	Dropout(0.33)	0	(1,1,1536)
15	FC(9)	+13 833	(1,1,9)
16	Softmax	0	(1,1,9)
17	Output	= 13 469 865	9

implement a large number of 3 × 3 convolutions inspired by VGG-net (Simonyan & Zisserman 2014), as well as architectures presented in Ravanbakhsh et al. (2017) and Gupta et al. (2018). The model that worked best for our purposes is almost exclusively based on the inception layers first presented in Szegedy et al. (2014). Here, we adopt one of its latest iterations, version 4 introduced in Szegedy, Ioffe & Vanhoucke (2016). The global linear structure of our CNN is shown in Table 6 and we describe in detail the different elements of this network and their purpose in Appendix B3. We visualize the evolution of the network's loss during training in Fig. 4.

The total classification success rate of the CNN is 52 per cent and its classification success matrix is shown in Table 7. Compared to the fixed feature results in Table 5, we find much larger true prediction values for many models. Exceptions are $f_5^{0.1}$ and $f_6^{0.06}$.

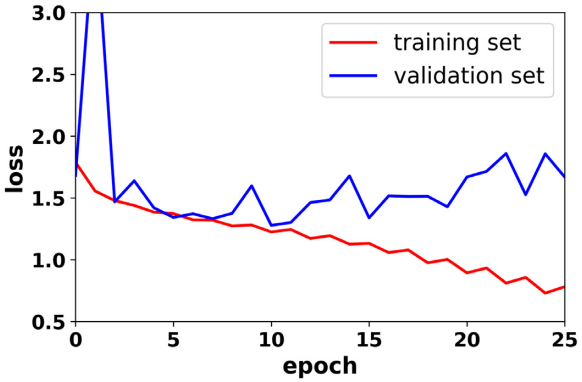

Figure 4. The evolution of the loss as a function of epoch for the training of the CNN.

Fig. 5 shows the statistics of the predictions for all classes in the test set and reveals that the $f_6^{0.06}$ and $f_6^{0.1}$ models, even as an ensemble, cannot be classified correctly by the CNN since the error bars on the medians of the predictions in their samples overlap with other f_6 models. However, the degeneracy with ΛCDM is now broken for all models and the CNN robustly discriminates most of the nine models from each other.

4.4 Dependence on redshift

A source redshift of $z_s = 1$ is realistic for future space- and ground-based surveys but it is certainly optimistic for current ground-based surveys. On the other hand, it also does not test the full potential of our classification methods since one would expect a better classification accuracy for larger source redshifts. We therefore repeat training and classification for one lower ($z_s = 0.5$) and one higher ($z_s = 2$) source redshift. For simplicity, we restrict this analysis to the CNN, which delivered the best results.

For a source redshift $z_s = 0.5$, the overall accuracy drops significantly from 52 to 44 per cent. When comparing the prediction statistics of the full set at this redshift in Fig. 6 with the reference at $z_s = 1$ in Fig. 5, one can see that the decrease in the overall accuracy mainly stems from a weaker separation of the two f_4 models, f_5, and ΛCDM. The known issue of degeneracies between the three neutrino masses for f_5 and f_6 are already present and more prominent. The issue of model misclassification for f_6 gravity gets worse with now two misclassifications. The improvements when going from $z_s = 1$ to 2 are highlighted by Fig. 7. For $z_s = 2$, the network's ability to distinguish between the base models increases and the overall classification accuracy is now 59 per cent. The discrimination accuracy for massive neutrinos within each gravity model increases for the f_5 models, and only the two f_6 models with massive neutrinos show significant overlap. Those models are also the only ones that show residual, but insignificant overlap with ΛCDM. Given the fact that the ensemble of $f_6^{0.1}$ maps also gets misidentified as $f_6^{0.06}$, it is clear that the discrimination within the f_6 models remains an issue even at a larger source redshift.

As a last analysis using the CNN, we perform a tomographic classification. For each line-of-sight realization, we are not using a single mass map at a specific source redshift but we feed data vectors of shape $\#x = (256, 256, 4)$ into the CNN where the four channels refer to $z_s = 0.5$, 1, 1.5, and 2, respectively. The classification success matrix for this analysis is shown in Table 8, and Fig. 8 shows the familiar box-plot representation of the prediction-vector statistics.

Table 7. The classification success matrix for the CNN. The general structure of the table is the same as in Table 5. We see successful classifications of the two f_4 models, f_5, f_6, and ΛCDM. However, large degeneracies remain within the neutrino mass variants of f_5 and f_6 gravity, respectively. In some cases, the wrong predictions can outnumber the correct ones as is the case for $f_5^{0.1}$ and $f_6^{0.06}$.

	f_4	$f_4^{0.3}$	f_5	$f_5^{0.15}$	$f_5^{0.1}$	f_6	$f_6^{0.06}$	$f_6^{0.1}$	ΛCDM
f_4	**1307** **80 per cent** **0.646** ±**0.008**	116 7 per cent 0.085 ±0.004	36 2 per cent 0.045 ±0.002	21 1 per cent 0.035 ±0.002	31 2 per cent 0.05 ±0.002	88 5 per cent 0.049 ±0.002	12 1 per cent 0.036 ±0.002	5 0 per cent 0.025 ±0.001	22 1 per cent 0.028 ±0.002
$f_4^{0.3}$	51 3 per cent 0.046 ±0.003	**1298** **79 per cent** **0.658** ±**0.007**	0 0 per cent 0.001 ±0.0	11 1 per cent 0.014 ±0.001	0 0 per cent 0.004 ±0.0	7 0 per cent 0.02 ±0.001	8 0 per cent 0.031 ±0.001	27 2 per cent 0.037 ±0.002	236 14 per cent 0.19 ±0.005
f_5	105 6 per cent 0.064 ±0.003	1 0 per cent 0.002 ±0.0	**1065** **65 per cent** **0.444** ±**0.005**	90 5 per cent 0.114 ±0.003	320 20 per cent 0.316 ±0.002	48 3 per cent 0.028 ±0.002	2 0 per cent 0.016 ±0.001	1 0 per cent 0.011 ±0.001	6 0 per cent 0.006 ±0.001
$f_5^{0.15}$	103 6 per cent 0.059 ±0.003	37 2 per cent 0.031 ±0.002	161 10 per cent 0.153 ±0.004	**721** **44 per cent** **0.3** ±**0.005**	347 21 per cent 0.235 ±0.003	78 5 per cent 0.048 ±0.002	14 1 per cent 0.044 ±0.002	31 2 per cent 0.041 ±0.002	146 9 per cent 0.088 ±0.004
$f_5^{0.1}$	122 7 per cent 0.071 ±0.004	5 0 per cent 0.007 ±0.001	624 38 per cent 0.323 ±0.005	271 17 per cent 0.187 ±0.004	**514** **31 per cent** **0.315** ±**0.003**	70 4 per cent 0.036 ±0.002	5 0 per cent 0.025 ±0.001	7 0 per cent 0.019 ±0.001	20 1 per cent 0.018 ±0.001
f_6	51 3 per cent 0.035 ±0.002	27 2 per cent 0.022 ±0.002	11 1 per cent 0.023 ±0.001	42 3 per cent 0.034 ±0.002	44 3 per cent 0.036 ±0.002	**968** **59 per cent** **0.326** ±**0.004**	74 5 per cent 0.235 ±0.002	307 19 per cent 0.218 ±0.002	114 7 per cent 0.072 ±0.003
$f_6^{0.06}$	36 2 per cent 0.026 ±0.002	46 3 per cent 0.036 ±0.002	11 1 per cent 0.017 ±0.001	40 2 per cent 0.034 ±0.002	35 2 per cent 0.029 ±0.002	713 44 per cent 0.271 ±0.003	**95** **6 per cent** **0.235** ±**0.002**	458 28 per cent 0.24 ±0.002	204 12 per cent 0.112 ±0.004
$f_6^{0.1}$	20 1 per cent 0.018 ±0.001	79 5 per cent 0.05 ±0.003	5 0 per cent 0.01 ±0.001	35 2 per cent 0.03 ±0.001	17 1 per cent 0.02 ±0.001	558 34 per cent 0.24 ±0.003	64 4 per cent 0.23 ±0.002	**565** **34 per cent** **0.253** ±**0.002**	295 18 per cent 0.149 ±0.004
ΛCDM	41 3 per cent 0.027 ±0.002	179 11 per cent 0.141 ±0.005	0 0 per cent 0.005 ±0.0	43 3 per cent 0.04 ±0.002	3 0 per cent 0.014 ±0.001	99 6 per cent 0.087 ±0.002	43 3 per cent 0.111 ±0.002	144 9 per cent 0.129 ±0.002	**1086** **66 per cent** **0.445** ±**0.006**

The overall classification success rate rises to 76 per cent and all models besides $f_6^{0.06}$ and $f_6^{0.1}$ now show correct classification rates of 74 per cent or clearly above. The probabilities of correctly classifying a single map in those two models are only 38 or 50 per cent, respectively; however, a look at Fig. 8 reveals that they are correctly classified as an ensemble and at high significance. Finally, it is worth noting that none of the models shows any degeneracy with ΛCDM, which is larger than 4 per cent according to Table 8.

4.5 Remarks on extracted features

After presenting the raw classification results for different methods, we now briefly investigate what insight can be gathered into the actual meaning and importance of specific features that drive the classification success of different methods. To do so, we take a closer look at the training process. The first important observation is strikingly highlighted in Table 3, which shows that almost all of the most discriminating `wnd-charm` features are Zernike coefficients derived from the Fourier transform of the raw image or from the Fourier transform of the edge- or wavelet-processed image. This is interesting since Zernike polynomials were originally introduced to describe the effects of certain optical elements such as lenses or reflecting surfaces in optical imaging (Zernike 1934). This suggests that a decomposition of mass maps into a function set that has a well-defined physical meaning does indeed lead to a good general representation of our data. In addition, all those features are derived from transformations of the raw mass map, which shows the power of filtering the input data as, e.g. shown by Peel et al. (2018a). The ranking of the standard features shown in Table 2 is less dominated by a single class, although the power spectrum and peak counts seem most relevant. The good results with a neural network as classifier show that the optimal combination of such classical

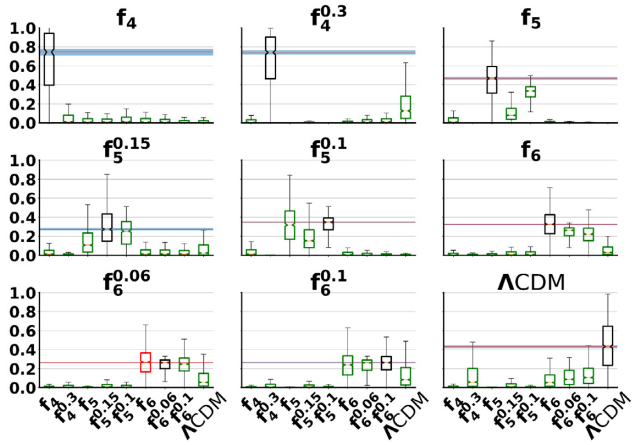

Figure 5. Prediction statistics for the CNN at source redshift $z_s = 1$. The structure of the figure is the same as in Fig. 3. The CNN discriminates more clearly between the models since both f_4 realisations, f_5 and ΛCDM are now clearly distinguished. Problems remain for the different neutrino mass realizations within f_6 and f_5 gravity. $f_6^{0.06}$ is incorrectly classified as $f_6^{0.1}$.

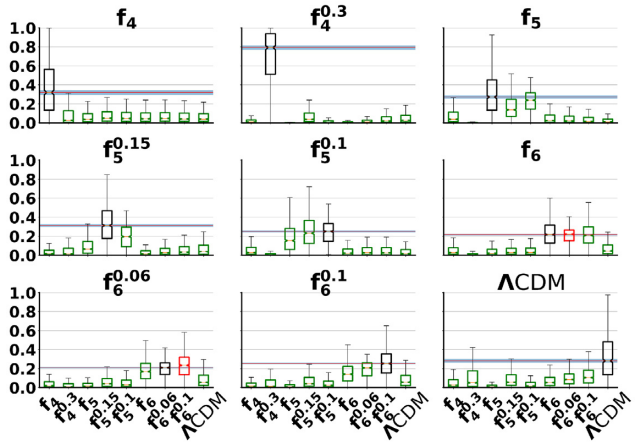

Figure 6. Prediction statistics for the CNN and a source redshift of $z_s = 0.5$. Compared to Fig. 5, the separation between the models becomes washed out. Two misclassifications occur: f_6 is incorrectly classified as $f_6^{0.06}$ and $f_6^{0.06}$ is misclassified as $f_6^{0.1}$. Also, the $f_5^{0.1}$ samples cannot be distinguished as an ensemble from the $f_5^{0.15}$ ones since their prediction medians overlap within the error bars.

features leads to a good classification even without the need for additional descriptors.

CNNs often deliver superior results compared to other methods for certain tasks, but it is often believed that they are harder to understand and interpret. We are attempting to counteract this trend by applying visualization techniques for the different filters linked together in a deep neural network (Girshick et al. 2013; Szegedy et al. 2013; Zeiler & Fergus 2013; Springenberg et al. 2014) and in order to reveal the inner workings of the complex model. We follow the approach of Simonyan, Vedaldi & Zisserman (2013) to extract our filter responses.[12] Starting from an image of random numbers with the same shape as our mass maps, we retrieve the output of every convolutional layer in the network and perform a gradient ascent in order to maximize the response of those layers.

[12]Also see https://github.com/keras-team/keras/blob/master/examples/conv_filter_visualization.py.

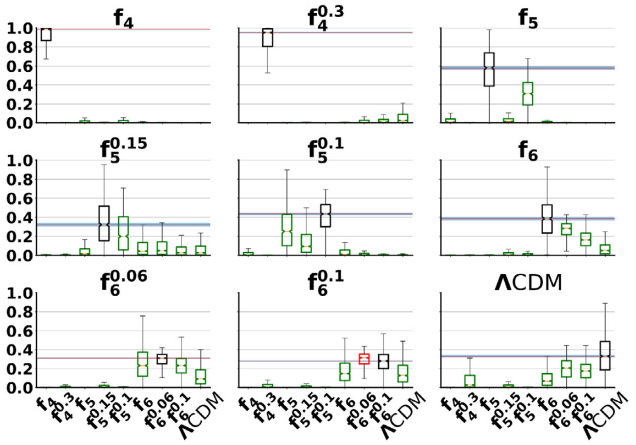

Figure 7. Prediction statistics for the CNN and a source redshift of $z_s = 2$. Only the $f_6^{0.1}$ model remains degenerate given the error bar on the median of its sample predictions. In fact, it is misclassified as $f_6^{0.06}$ by the CNN.

While this is of course not a unique solution, the result of the final iteration of the ascent represents an example that triggered a strong response at a particular depth in the network. In Fig. 9, we show a few examples. The top row shows the four channels that had the strongest loss compared to the initial random image in CNN layer one, that is the 3×3 convolution marked with index 2 in Table 6. The second row shows the top four channel responses of the 3×3 convolution with stride two just above the input layer in Fig. B1. The row marked with InceptionA shows the most responsive channels among all four convolutions just below the concatenation layer in Fig. B1 and equivalently for the figure rows marked InceptionB and C. As is typical for CNNs (Zeiler & Fergus 2013), the very first level extracts very regular horizontal and vertical stripe patterns from the image. The stripes turn into a grid pattern deeper into the network and once arriving at the end of the InceptionA layer we can identify patterns of peaks and troughs that are either grouped regularly or along larger structures. It is not surprising that the earlier layers of the network, up to InceptionA, perform a global filtering of the map that highlights structure as long as the image still consists of a relatively large number of pixels. It is just from the finer InceptionB layers onwards that more specific structures, like objects that look like individual clusters or voids, are picked up. Such detailed analyses of the inner structure of trained CNNs will lead to a deeper understanding why those networks work so well. This can potentially lead to the development of more specific algorithms at lower numerical cost but with similar or better classification performance.

5 CONCLUSIONS

We studied the ability of different kinds of machine learning techniques to discriminate between highly degenerate cosmological models, which combine the effects of MG and massive neutrinos on structure formation. For this purpose, we used a subset of the DUSTGRAIN-*pathfinder* simulation suite that consists of ΛCDM and eight $f(R)$ models of gravity in the range of $-1 \times 10^{-4} \leq f_{R0} \leq -1 \times 10^{-6}$. The neutrino masses in the simulations span 0 eV $\leq m_\nu \leq$ 0.3 eV. Lensing convergence maps produced from these simulations provided the input for the different classification methods.

In order to characterize the mass maps, we used three different approaches to feature extraction. Commonly used statistics in

Table 8. The classification success matrix for the tomographic analysis using the CNN. The general structure of the table is the same as in Table 5. We see good classification rates above 79 per cent and typically above 90 per cent for all models but the f_6 family. Also, f_6 with vanishing neutrino mass is correctly classified 74 per cent of the time. The remaining degeneracies are limited to $f_6^{0.06}$ and $f_6^{0.1}$ with 38 and 50 per cent classification accuracy, respectively, but given the error bars on the prediction mean the degeneracy is not significant for the ensemble of mass maps in the test set.

	f_4	$f_4^{0.3}$	f_5	$f_5^{0.15}$	$f_5^{0.1}$	f_6	$f_6^{0.06}$	$f_6^{0.1}$	ΛCDM
f_4	**1618**	0	7	0	10	3	0	0	0
	99 per cent	0 per cent	0 per cent	0 per cent	1 per cent	0 per cent	0 per cent	0 per cent	0 per cent
	0.985	0.0	0.007	0.0	0.006	0.002	0.0	0.0	0.0
	±0.002	±0.0	±0.002	±0.0	±0.001	±0.001	±0.0	±0.0	±0.0
$f_4^{0.3}$	0	**1501**	0	1	0	0	1	20	115
	0 per cent	**92 per cent**	0 per cent	0 per cent	0 per cent	0 per cent	0 per cent	1 per cent	7 per cent
	0.0	**0.91**	0.0	0.001	0.0	0.0	0.002	0.015	0.072
	±0.0	**±0.006**	±0.0	±0.0	±0.0	±0.0	±0.001	±0.002	±0.005
f_5	3	0	**1257**	2	375	1	0	0	0
	0 per cent	0 per cent	**77 per cent**	0 per cent	23 per cent	0 per cent	0 per cent	0 per cent	0 per cent
	0.001	0.0	**0.748**	0.002	0.247	0.001	0.0	0.0	0.0
	±0.001	±0.0	**±0.008**	±0.001	±0.008	±0.001	±0.0	±0.0	±0.0
$f_5^{0.15}$	0	0	1	**1470**	96	22	7	2	40
	0 per cent	0 per cent	0 per cent	**90 per cent**	6 per cent	1 per cent	0 per cent	0 per cent	2 per cent
	0.0	0.0	0.001	**0.873**	0.071	0.015	0.009	0.004	0.027
	±0.0	±0.0	±0.001	**±0.007**	±0.005	±0.002	±0.001	±0.001	±0.003
$f_5^{0.1}$	0	0	130	207	**1289**	12	0	0	0
	0 per cent	0 per cent	8 per cent	13 per cent	**79 per cent**	1 per cent	0 per cent	0 per cent	0 per cent
	0.001	0.0	0.104	0.148	**0.74**	0.008	0.0	0.0	0.0
	±0.0	±0.0	±0.005	±0.007	**±0.008**	±0.002	±0.0	±0.0	±0.0
f_6	0	0	0	15	1	**1206**	275	30	111
	0 per cent	0 per cent	0 per cent	1 per cent	0 per cent	**74 per cent**	17 per cent	2 per cent	7 per cent
	0.0	0.0	0.0	0.01	0.001	**0.676**	0.194	0.046	0.073
	±0.0	±0.0	±0.0	±0.002	±0.0	**±0.008**	±0.005	±0.003	±0.005
$f_6^{0.06}$	0	0	0	7	0	363	**627**	319	322
	0 per cent	0 per cent	0 per cent	0 per cent	0 per cent	22 per cent	**38 per cent**	19 per cent	20 per cent
	0.0	0.0	0.0	0.005	0.0	0.226	**0.344**	0.233	0.191
	±0.0	±0.0	±0.0	±0.001	±0.0	±0.007	**±0.005**	±0.006	±0.007
$f_6^{0.1}$	0	3	0	3	0	77	385	**827**	343
	0 per cent	0 per cent	0 per cent	0 per cent	0 per cent	5 per cent	24 per cent	**50 per cent**	21 per cent
	0.0	0.002	0.0	0.002	0.0	0.064	0.272	**0.455**	0.204
	±0.0	±0.001	±0.0	±0.001	±0.0	±0.004	±0.005	**±0.008**	±0.008
ΛCDM	0	1	0	6	0	57	69	37	**1468**
	0 per cent	0 per cent	0 per cent	0 per cent	0 per cent	3 per cent	4 per cent	2 per cent	**90 per cent**
	0.0	0.001	0.0	0.006	0.0	0.039	0.059	0.044	**0.85**
	±0.0	±0.001	±0.0	±0.001	±0.0	±0.003	±0.003	±0.003	**±0.007**

astrophysics such as, and among others, the power spectrum, peak counts, and Minkowski functionals were combined into a single feature vector. In order to probe features that are more common to the field of computer vision and digital image processing, we used the publicly available `wnd-charm` algorithm that produces a large feature vector that combines a variety of common and more exotic descriptors and statistics. As the most flexible method of feature extraction, we used a CNN. For classification, we tested a nearest-neighbour method in feature space and a fully connected neural network.

We provide an overview of the classification results from Section 4 in Table 9 and our results can be summarized as follows:

(i) Nearest-neighbour classifiers based on distances in feature space are not delivering robust results. No matter if a small classical feature vector is used or a longer version based on computer vision, the total classification accuracy stays below 25 per cent. Eight, out of the nine tested models, remain observationally degenerate.[13]

(ii) With the same classical or computer vision feature vectors, a neural network delivers a much more robust classification than the nearest-neighbour method. The total success rate for the classical feature vector is 39 per cent and the number of degenerate models reduces to three.

(iii) The longer feature vector containing 2919 features inspired by computer vision delivers a slightly worse classification of our models than the shorter vector with 99 classical descriptors. The total classification success rate is 3 per cent lower and the method produces one additional degenerate model. Some

[13]We declare a model as degenerate if the median and its error for the predictions of a true test set class overlap with the median and its error of the predictions for any other class.

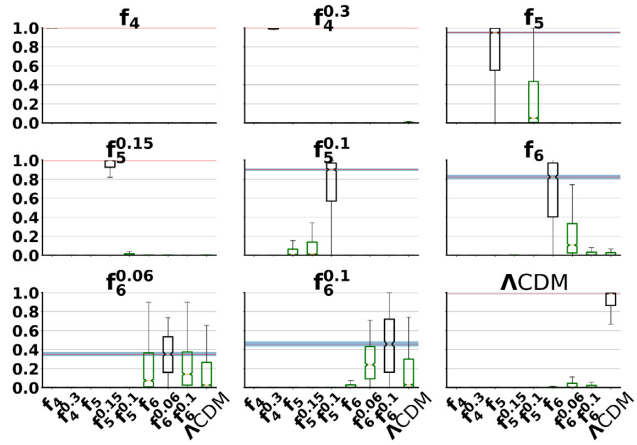

Figure 8. Prediction statistics on the tomographic analysis with the CNN. For many classes, the classification is so good that the prediction samples cluster around the optimal value of 1. All models are correctly classified and within the error bars of the prediction medians, no model remains observationally degenerate. Only small similarities remain between the three f_6 models of varying neutrino mass.

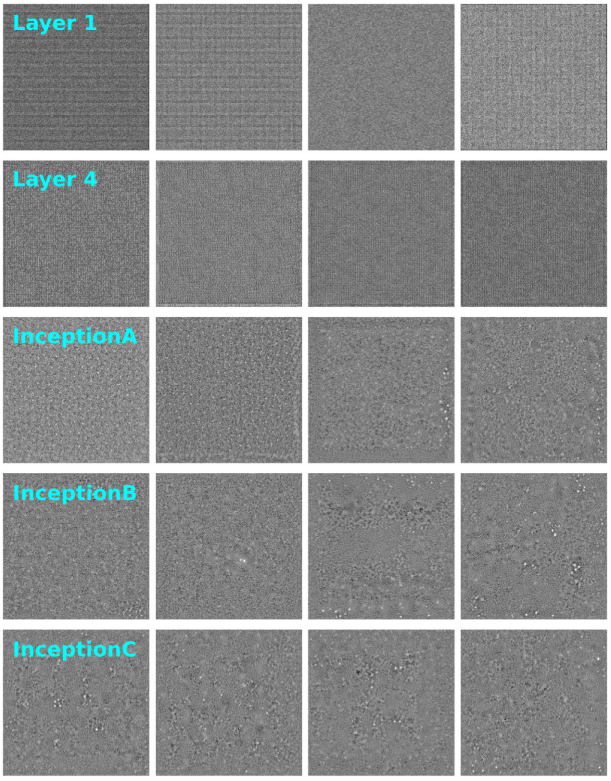

Figure 9. Visualizations of the convolutional filters applied by the CNN at different depths of the network.

of the computer vision feature may very well be useful, but currently we see no advantage of using features inspired by digital image processing compared to features well-established in cosmology.

(iv) A CNN delivers the best classification results with 52 per cent correct classifications at source redshift $z_s = 1.0$. The number of degenerate models reduces to two, both of which are part of the same f_6 model of gravity.

(v) Classification success is clearly a function of mass map source redshift. While going from $z_s = 1$ to 0.5 the success rate of the CNN decreases by 8 per cent and the number of degenerate models increases by one. When going from $z_s = 1$ to 2 the accuracy increases by 7 per cent and the number of degenerate models reduces by one. This increase of success rate with increasing redshift is not surprising since more information relevant to structure formation can be picked up along a deeper line of sight.

(vi) When using a CNN in a tomographic analysis of four different mass map source redshifts along the same line of sight, all observational degeneracies are fully broken. The total classification success rate increases to 76 per cent.

A number of improvements to our methodology come to mind and we reserve them for future work. First, the flexible features derived by a CNN can be combined with fixed features that are known to contribute to a successful classification of degenerate models. Secondly, instead of working on the raw image data, a clever transformation can be applied to the input data to enhance features that allow for the desired discrimination. We attempt such an approach in the context of machine learning in Peel et al. (2018b, PRL submitted). In fact, the CNN used in this work applies such transformations as we discussed in Section 4.5. A careful analysis of the filtering process of a CNN at the early levels of its filter chain can provide useful insights into the most powerful image transformation for a given classification task. Furthermore, the careful analysis of

Table 9. A summary of the performance of the different methods used in this analysis. Θ indicates the feature extraction function as described in Section 3.2. ζ is the classification function introduced in Section 3.3 and z_s is the convergence map source redshift. 'Degenerate classes' represents the number of all models for which the median and its error for the predictions of the true test set class overlap with the median and its error of the predictions for any other class. The table also lists the performance for the particularly important ΛCDM class and shows the classification accuracy of each method for this model as well as the largest misclassification rate and the associated model. A reference to the detailed results of each model is given in the last column, where the reference 'repository' points to the online repository mentioned in Section 3.4.

Θ	ζ	z_s	Total accuracy	Degenerate classes	ΛCDM performance	Reference
classic	nearest neighbour	1.0	22 per cent	8	14 per cent/15 per cent $\left(f_4^{0.3}\right)$	repository
`wnd-charm`	nearest neighbour	1.0	25 per cent	7	24 per cent/24 per cent $\left(f_4^{0.3}\right)$	repository
classic	neural network	1.0	39 per cent	3	47 per cent/16 per cent $\left(f_4^{0.3}\right)$	Table 5, Fig. 3
`wnd-charm`	neural network	1.0	36 per cent	4	42 per cent/24 per cent $\left(f_4^{0.3}\right)$	repository
CNN	neural network	0.5	44 per cent	3	52 per cent/15 per cent $\left(f_4^{0.3}\right)$	Fig. 6
CNN	neural network	1.0	52 per cent	2	66 per cent/11 per cent $\left(f_4^{0.3}\right)$	Table 7, Fig. 5
CNN	neural network	2.0	59 per cent	1	53 per cent/12 per cent $\left(f_4^{0.3}\right)$	Fig. 7
CNN	neural network	0.5,1.0,1.5,2.0	76 per cent	0	90 per cent/4 per cent $\left(f_6^{0.06}\right)$	Table 8, Fig. 8

the filters at a much deeper level of the network might actually lead to more insights on structure formation in different models, since it is at this deeper level where individual structure is characterized and isolated by the algorithm.

Much work is left to be done before this machine learning approach to the classification of mass maps in different cosmological models can be applied to real data. In this work, we limited ourselves to optimal noise-free maps in order to see how different methodologies compare under optimal conditions. The influence of pixel shot noise, observational systematics, and practical issues like masking and image artefacts needs to be studied in detail. Furthermore, since the currently most successful methods use a supervised training process with labelled data based on numerical simulations, it needs to be carefully investigated how closely those simulated maps resemble a real observation. Without this important sanity check, even the best machine learning technique is useless since it learns the wrong data.

ACKNOWLEDGEMENTS

We would like to thank Ofer Springer for useful discussions about deep learning. JM has received funding from the European Union's Horizon 2020 research and Innovation programme under the Marie Skłodowska-Curie grant agreement no. 664931. AP acknowledges support from an Enhanced Eurotalents Fellowship, a Marie Skłodowska-Curie Actions Programme co-funded by the European Commission and Commissariat à l'énergie atomique et aux énergies alternatives (CEA). CG and MB acknowledge support from the Italian Ministry for Education, University and Research (MIUR) through the SIR individual grant SIMCODE (project number RBSI14P4IH). CG and MM acknowledge support from the Italian Ministry of Foreign Affairs and International Cooperation, Directorate General for Country Promotion (Project 'Crack the lens'). We also acknowledge the support from the grant MIUR PRIN 2015 'Cosmology and Fundamental Physics: illuminating the Dark Universe with Euclid', and the financial contribution from the agreement ASI n.I/023/12/0 'Attività relative alla fase B2/C per la missione Euclid'. The DUSTGRAIN-*pathfinder* simulations analysed in this work have been performed on the Marconi supercomputing machine at Cineca, thanks to the PRACE project SIMCODE1 (grant no. 2016153604), and on the computing facilities of the Computational Center for Particle and Astrophysics (C2PAP) and of the Leibniz Supercomputer Center (LRZ) under the project ID pr94ji. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the two Titan Xp GPUs used for this research.

REFERENCES

Aartsen M. G. et al., 2013, Phys. Rev. Lett., 110, 131302
Abbott T. M. C. et al., 2018, Phys. Rev. D, 98, 043526
Abraham R. G., van den Bergh S., Nair P., 2003, ApJ, 588, 218
Ackermann M. et al., 2017, ApJ, 840, 43
Alam S. et al., 2017, MNRAS, 470, 2617
Albert A. et al., 2017, ApJ, 834, 110
Amendola L. et al., 2018, Living Rev. Relativ., 21, 2
Arnold C., Fosalba P., Springel V., Puchwein E., Blot L., 2019, MNRAS, 483, 790
Arnold C., Puchwein E., Springel V., 2014, MNRAS, 440, 833
Arnold C., Puchwein E., Springel V., 2015, MNRAS, 448, 2275
Arnold C., Springel V., Puchwein E., 2016, MNRAS, 462, 1530
ATLAS Collaboration, 2014, Phys. Rev. Lett., 112, 041802
Baldi M., Villaescusa-Navarro F., 2018, MNRAS, 473, 3226
Baldi M., Villaescusa-Navarro F., Viel M., Puchwein E., Springel V., Moscardini L., 2014, MNRAS, 440, 75
Barreira A., Llinares C., Bose S., Li B., 2016, JCAP, 5, 001
Bartelmann M., Schneider P., 2001, Phys. Rep., 340, 291
Benitez N. et al., 2014, preprint (arXiv:1403.5237)
Bennett C. L. et al., 2013, ApJS, 208, 20
Bernabei R. et al., 2018, J. Nucl. Phys. At. Energy, 19, 307
Bishop C. M., 2006, Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA
Buchdahl H. A., 1970, MNRAS, 150, 1
Castro T., Quartin M., Giocoli C., Borgani S., Dolag K., 2018, MNRAS, 478, 1305
Chetlur S., Woolley C., Vandermersch P., Cohen J., Tran J., Catanzaro B., Shelhamer E., 2014, preprint (arXiv:1410.0759)
Chollet F., 2017, Deep Learning with Python. 1st edn., Manning Publications Co., Greenwich, CT, USA
CMS Collaboration, 2016, JHEP, 12, 88
Dietrich J. P., Hartlap J., 2010, MNRAS, 402, 1049
Fogel I., Sagi D., 1989, Biol. Cybern., 61, 103
Friedrich O. et al., 2018, Phys. Rev. D, 98, 023508
Fu L. et al., 2008, A&A, 479, 9
Giocoli C., Baldi M., Moscardini L., 2018a , MNRAS, 481, 2813
Giocoli C., Meneghetti M., Metcalf R. B., Ettori S., Moscardini L., 2014, MNRAS, 440, 1899
Giocoli C., Moscardini L., Baldi M., Meneghetti M., Metcalf R. B., 2018b, MNRAS, 478, 5436
Giocoli C. et al., 2016, MNRAS, 461, 209
Giocoli C. et al., 2017, MNRAS, 470, 3574
Girshick R., Donahue J., Darrell T., Malik J., 2013, preprint (arXiv:1311.2524)
Goodfellow I., Bengio Y., Courville A., 2016, Deep Learning. The MIT Press , Cambridge, Masachusetts
Graves A., 2013, preprint (arXiv:1308.0850)
Gruen D. et al., 2018, Phys. Rev. D, 98, 023507
Gupta A., Matilla J. M. Z., Hsu D., Haiman Z., 2018, Phys. Rev. D, 97, 103515
Hagstotz S., Costanzi M., Baldi M., Weller J., 2019 , MNRAS, 486, 3927
Hanisch R. J., Farris A., Greisen E. W., Pence W. D., Schlesinger B. M., Teuben P. J., Thompson R. W., Warnock A., III, 2001, A&A, 376, 359
Haralick R. M., Shanmugam K., Dinstein I., 1973, IEEE Trans. Syst. Man Cybern. (SMC-3), 6, 610
He J.-h., 2013, Phys. Rev. D, 88, 103523
He K., Zhang X., Ren S., Sun J., 2015, preprint (arXiv:1512.03385)
Herbel J., Kacprzak T., Amara A., Refregier A., Lucchi A., 2018, JCAP, 2018, 054
Heymans C. et al., 2012, MNRAS, 427, 146
Hikage C., Mandelbaum R., Leauthaud A., Rozo E., Rykoff E. S., 2018, MNRAS, 480, 2689
Hilbert S., Hartlap J., White S. D. M., Schneider P., 2009, A&A, 499, 31
Hilbert S., White S. D. M., Hartlap J., Schneider P., 2008, MNRAS, 386, 1845
Hildebrandt H. et al., 2017, MNRAS, 465, 1454
Hu W., Sawicki I., 2007, Phys. Rev. D, 76, 064004
Ioffe S., Szegedy C., 2015, preprint (arXiv:1502.03167)
Ivezic Z. et al., 2008, ApJ, 873, 111
Johnson M. et al., 2016, preprint (arXiv:1611.04558)
Joudaki S. et al., 2017, MNRAS, 465, 2033
Kingma D. P., Ba J., 2014, preprint (arXiv:1412.6980)
Kratochvil J. M., Haiman Z., May M., 2010, Phys. Rev. D, 81, 043519
Kratochvil J. M., Lim E. A., Wang S., Haiman Z., May M., Huffenberger K., 2012, Phys. Rev. D, 85, 103513
Krizhevsky A., Sutskever I., Hinton G. E., 2012, in Pereira F., Burges C. J. C., Bottou L., Weinberger K. Q., eds, Advances in Neural Information Processing Systems 25. Curran Associates, Inc., Red Hook, New York, p. 1097
Laureijs R. et al., 2011, preprint (arXiv: 1110.3193)

Lecun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436
Lin C.-A., Kilbinger M., 2018, A&A, 614, A36
Lin T., Goyal P., Girshick R. B., He K., Dollár P., 2017, preprint (arXiv:1708.02002)
Lucie-Smith L., Peiris H. V., Pontzen A., Lochner M., 2018, MNRAS, 479, 3405
Martinet N. et al., 2018, MNRAS, 474, 712
Motohashi H., Starobinsky A. A., Yokoyama J., 2013, Phys. Rev. Lett., 110, 121302
Naik A. P., Puchwein E., Davis A.-C., Arnold C., 2018, MNRAS, 480, 5211
Ntampaka M. et al., 2018, preprint (arXiv:1810.07703)
Orlov N., Johnston J., Macura T., Wolkow C., Goldberg I., 2006, 3rd IEEE Int. Symp. Biomedical Imaging: Nano to Macro, 2006, 1152
Orlov N., Shamir L., Macura T., Johnston J., Eckley D. M., Goldberg I. G., 2008, Pattern Recognit. Lett., 29, 1684
Otsu N., 1979, IEEE Trans. Syst. Man Cybern., 9, 62
Parkinson D. et al., 2012, Phys. Rev. D, 86, 103518
Peel A., Lalande F., Starck J.-L., Pettorino V., Merten J., Giocoli C., Meneghetti M., Baldi M., 2018b, preprint (arXiv:1810.11030)
Peel A., Lin C.-A., Lanusse F., Leonard A., Starck J.-L., Kilbinger M., 2017, A&A, 599, A79
Peel A., Pettorino V., Giocoli C., Starck J.-L., Baldi M., 2018a, A&A, 619, A38
Perlmutter S. et al., 1999, ApJ, 517, 565
Petri A., 2016, Astron. Comput., 17, 73
Petri A., Haiman Z., May M., 2017, Phys. Rev. D, 95, 123503
Petri A., Liu J., Haiman Z., May M., Hui L., Kratochvil J. M., 2015, Phys. Rev. D, 91, 103511
Pezzotta A. et al., 2017, A&A, 604, A33
Planck Collaboration VI, 2018, preprint (arXiv:1807.06209)
Planck Collaboration XIII, 2016a, A&A, 594, A13
Planck Collaboration XXIV, 2016b, A&A, 594, A24
Prewitt J., 1970, Picture Processing and Psychopictorics. Academic Press, New York
Puchwein E., Baldi M., Springel V., 2013, MNRAS, 436, 348
Radon J., 1917, in Berichte über die Verhandlungen der Königlich-Sächsischen Akademie der Wissenschaften zu Leipzig. Teubner, p. 262
Ravanbakhsh S., Oliva J., Fromenteau S., Price L. C., Ho S., Schneider J., Poczos B., 2017, preprint (arXiv:1711.02033)
Riess A. G. et al., 1998, AJ, 116, 1009
Rodríguez A. C., Kacprzak T., Lucchi A., Amara A., Sgier R., Fluri J., Hofmann T., Réfrégier A., 2018, Comput. Astrophys. Cosmology, 5, 4
Roncarelli M., Baldi M., Villaescusa-Navarro F., 2018, MNRAS, 481, 2497
Rumelhart D. E., Hinton G. E., Williams R. J., 1986, Nature, 323, 533
Schmidt B. P. et al., 1998, ApJ, 507, 46
Schneider P., 1996, MNRAS, 283, 837
Schneider P., van Waerbeke L., Jain B., Kruse G., 1998, MNRAS, 296, 873
Schäfer B. M., Heisenberg L., Kalovidouris A. F., Bacon D. J., 2012, MNRAS, 420, 455
Shamir L., Delaney J. D., Orlov N., Eckley D. M., Goldberg I. G., 2010, PLoS Comput. Biol., 6, e1000974
Shamir L., Orlov N., Eckley D. M., Macura T., Johnston J., Goldberg I., 2008, Source Code Biol. Med., 3, 13
Shan H. et al., 2018, MNRAS, 474, 1116
Shirasaki M., Nishimichi T., Li B., Higuchi Y., 2017, MNRAS, 466, 2402
Simonyan K., Vedaldi A., Zisserman A., 2013, preprint (arXiv:1312.6034)
Simonyan K., Zisserman A., 2014, preprint (arXiv:1409.1556)
Spergel D. et al., 2015, preprint (arXiv:1503.03757)
Springel V., 2005, MNRAS, 364, 1105
Springel V. et al., 2005, Nature, 435, 629
Springenberg J. T., Dosovitskiy A., Brox T., Riedmiller M., 2014, preprint (arXiv:1412.6806)
Springer O. M., Ofek E. O., Weiss Y., Merten J., 2018, preprint (arXiv:1808.07491)
Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., 2014, J. Mach. Learn. Res., 15, 1929
Szegedy C., Ioffe S., Vanhoucke V., 2016, preprint (arXiv:1602.07261)
Szegedy C., Zaremba W., Sutskever I., Bruna J., Erhan D., Goodfellow I., Fergus R., 2013, preprint (arXiv:1312.6199)
Szegedy C. et al., 2014, preprint (arXiv:1409.4842)
Tamura H., Mori S., Yamawaki T., 1978, IEEE Trans. Syst. Man Cybern., 8, 460
Teague M. R., 1980, J. Opt. Soc. Am., 70, 920
Tessore N., Winther H. A., Metcalf R. B., Ferreira P. G., Giocoli C., 2015, JCAP, 2015, 036
Troxel M. A. et al., 2018, Phys. Rev. D, 98, 043528
Van Waerbeke L. et al., 2013, MNRAS, 433, 3373
Viel M., Haehnelt M. G., Springel V., 2010, JCAP, 2010, 015
Vikhlinin A. et al., 2009, ApJ, 692, 1060
Villaescusa-Navarro F., Banerjee A., Dalal N., Castorina E., Scoccimarro R., Angulo R., Spergel D. N., 2018, ApJ, 861, 53
White S. D. M., 1993, in Gleiser R. J., Kozameh C. N., Moreschi O. M., eds, General Relativity and Gravitation 1992, CRC Press , Boca Raton, Florida, p. 331
White S. D. M., 1996, in Schaeffer R., Silk J., Spiro M., Zinn-Justin J., eds, Cosmology and Large Scale Structure, Elsevier, Amsterdam, p. 349
White S. D. M., Rees M. J., 1978, MNRAS, 183, 341
Winther H. A. et al., 2015, MNRAS, 454, 4208
Wright B. S., Winther H. A., Koyama K., 2017, JCAP, 2017, 054
Wu C.-M., Chen Y.-C., Hsieh K.-S., 1992, IEEE Trans. Med. Imaging, 11, 141
Wu Y. et al., 2016, p reprint (arXiv:1609.08144)
Zeiler M. D., Fergus R., 2013, preprint (arXiv:1311.2901)
Zennaro M., Bel J., Villaescusa-Navarro F., Carbone C., Sefusatti E., Guzzo L., 2017, MNRAS, 466, 3244
Zernike von F., 1934, Physica, 1, 689

APPENDIX A: WND-CHARM FEATURES

The total length of the `wnd-charm` feature vector entails 2919 descriptors, which can be divided into five families. We provide an overview of the features and their respective families in Table A1. The algorithm does not only work on the image itself (raw), but also on its Fourier (F), Wavelet (W), Chebyshev (C), or Edge transformation (E) as indicated by the 'Input' column of Table A1. Transformations of transformations are considered by the bracket notation. While Fourier and Chebyshev transforms are implemented using common algorithms and methodologies, the Wavelet transformation is performed with a one-level filter pass with a fifth-order symlet (Orlov et al. 2008) and the Edge transformation is carried out using a Prewitt operator (Prewitt 1970) to approximate the image gradient.

The pixel statistics family is made out of four different subclasses, with the simplest being the intensity statistics consisting of mean, median, standard deviation, minimum, and maximum. The multi-scale histograms are calculated by using three, five, seven, or nine bins to order the pixel amplitudes. The counts in each of those bins make up the 24 features in this subclass. The combined moments are mean, standard deviation, skewness, and kurtosis, which are calculated in a horizontal stripe through the image centre and with a width that is half the total image width. The stripe is then rotated by 45, 90, and 135 deg and the measurement is repeated. Those 16 numbers are sampled into 3 bins each, providing a total of 48 features. The Gini coefficient (Abraham, van den Bergh & Nair 2003) is a measure of how equal the spectrum of pixel intensities is distributed within the image.

Table A1. `wnd-charm` image features used in this analysis.

Family	Class	Features	Input	Reference
Pixel statistics	Combined moments	48	raw, F, W, C, C(F), W(F) F(W), F(C), C(W), E, F(E), W(E)	–
	Gini coefficient	1	raw, F, W, C, C(F), W(F) F(W), F(C), C(W), E, F(E), W(E)	Abraham et al. (2003)
	Multiscale histograms	24	raw, F, W, C, C(F), W(F) F(W), F(C), C(W), E, F(E), W(E)	–
	Pixel intensity statistics	5	raw, F, W, C, C(F), W(F) F(W), F(C), C(W), E, F(E), W(E)	–
Polynomial decomposition	Chebyshev coefficients	32	raw, F, W, C, F(W), E, F(E), W(E)	–
	Chebyshev–Fourier coefficients	32	raw, F, W, C, F(W), E, F(E), W(E)	Orlov et al. (2006)
	Radon coefficients	12	raw, F, W, C, C(F), W(F) F(W), F(C), C(W), E, F(E), W(E)	Radon (1917)
	Zernike coefficients	72	raw, F, W, C, F(W), E, F(E), W(E)	Teague (1980))
Textures	Fractal analysis	20	raw, F, W, C, C(F), W(F) F(W), F(C), C(W), E, F(E), W(E)	Wu et al. (1992)
	Gabor	7	raw	Fogel & Sagi (1989)
	Haralick	28	raw, F, W, C, C(F), W(F) F(W), F(C), C(W), E, F(E), W(E)	Haralick et al. (1973)
	Tamura	6	raw, F, W, C, C(F), W(F) F(W), F(C), C(W), E, F(E), W(E)	Tamura et al. (1978)
Objects	Edge features	28	raw	Prewitt (1970)
	Otsu object features	34	raw	Otsu (1979)
	Inverse Otsu object features	34	raw	Otsu (1979)

The second feature family is comprised of polynomial decompositions. The coefficients of an order 20 Chebyshev and an order 23 Chebyshev–Fourier (Orlov et al. 2006) transformation are sorted into 32 bin histograms. Radon transformations are carried out along lines with an inclination angle of 0, 45, 90, and 135 deg with respect to the image horizontal (Radon 1917) and ordered in 3 bin histograms. The class of Zernike coefficients is derived from a 2D Zernike decomposition of the image (Teague 1980) and the first 72 of those coefficients contribute to the feature vector.

The use of textures is common in image processing and is a way of describing spatial correlations of intensity values. We extract seven Gabor filters (e.g. Fogel & Sagi 1989) using Gaussian harmonic functions and define their image occupation area as a feature. Tamura textures are described in detail in Tamura, Mori & Yamawaki (1978) and `wnd-charm` uses contrast, directionality, coarseness sum, and coarseness binned into a 3 sample histogram. The 28 Haralick textures are specific properties of the grey-level dependence matrix of the image and are described in Haralick, Shanmugam & Dinstein (1973). The fractal analysis is based on a Brownian motion model of the image following Wu, Chen & Hsieh (1992) and `wnd-charm` uses the first 20 parameters of this analysis as features.

Object statistics are only derived from the raw image data. The starting point is an edge transform using a Prewitt filter and mean, median, variance, and 8 bin histogram of both image gradient and its directionality add up to 22 features, which are supplemented by the total number of edge pixels, their genus, and the differences between the directionality bins. Otsu features and their inverse are calculated after the application of an Otsu threshold (Otsu 1979). Finally, for all objects the algorithm calculates minimum, maximum, mean, median, variance, and 10 bin histogram for area and image-centre distance of all Otsu objects in the image.

APPENDIX B: DEEP NEURAL NETWORKS

In this appendix, we collect some more detailed information about deep neural networks. The first section formally defines all network layers used in this work and the second section deals with activation functions. The third section provides a thorough description about the architecture of the CNN that we use in our analyses.

B1 Layers

Given a 2D input vector I_{ijc} with $\#I_{ijc} = (X, Y, l)$, a convolution layer applies the following operation to produce an output O_{ijc}

$$\mathrm{Conv}(n, m, \Delta i, \Delta j, p, C) I_{i'j'c'} = O_{ijc} \tag{B1}$$

$$O_{ijc} = B_c + \sum_{i'=1}^{n} \sum_{j'=1}^{m} \sum_{c'=1}^{l} W^{c}_{i'j'c'} I_{(i\Delta i+i')(j\Delta j+j')c'} \tag{B2}$$

$$w_{\mathrm{conv}} = \left\{ B_c, W_{ijc} \right\} \quad \forall c \tag{B3}$$

$$\#O_{ijc} = \left(\frac{X}{\Delta i}, \frac{Y}{\Delta j}, l \right) \quad \text{for} \quad p = \mathrm{s} \tag{B4}$$

$$\#O_{xyc} = \left(\frac{X}{\Delta i} - \frac{n}{2}, \frac{Y}{\Delta j} - \frac{m}{2}, C \right) \quad \text{for} \quad p = \mathrm{v}. \tag{B5}$$

The stride parameters Δi and Δj allow one to implement dimensional reduction. The parameter p indicates if the input data are padded, which means that additional rows and columns are added in order to produce an output that has exactly the same spatial shape as the input, at least in the absence of stride. This is known as *same*

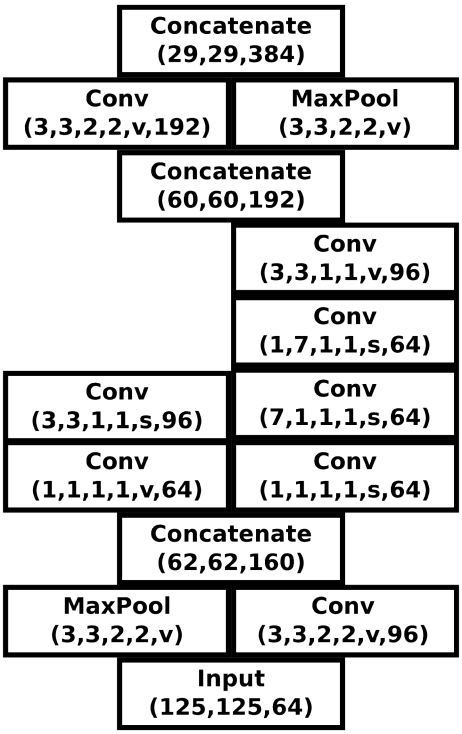

Figure B1. The internal structure of the stem Inception layer. The layout is identical to Szegedy et al. (2016), but with the image dimensions of our mass maps.

padding $p = \mathrm{s}$. Alternatively, the data can stay unaltered, or *valid* $p = \mathrm{v}$, which means that the spatial dimensions of the data vector are slightly reduced, since every convolution must be fully contained within the 2D image domain.

We use four different kinds of pooling layers. Their main functionality is either an averaging

$$\mathrm{AvgPool}(n, m, \Delta i, \Delta j, p) I_{i'j'c} = O_{ijc} \tag{B6}$$

$$O_{ijc} = \frac{1}{nm} \sum_{i'=1}^{n} \sum_{j'=1}^{m} I_{(i\Delta i+i')(j\Delta j+j')c} \tag{B7}$$

$$\#O_{ijc} = \left(\frac{X}{\Delta i}, \frac{Y}{\Delta j}, C \right) \quad \text{for} \quad p = \mathrm{s} \tag{B8}$$

$$\#O_{ijc} = \left(\frac{X}{\Delta i} - \frac{n}{2}, \frac{Y}{\Delta j} - \frac{m}{2}, l \right) \quad \text{for} \quad p = \mathrm{v}, \tag{B9}$$

or a maximum selection operation

$$\mathrm{MaxPool}(n, m) I_{i'j'c} = O_{ijc} \tag{B10}$$

$$O_{ijc} = \max \left\{ I_{(i\Delta i+i')(j\Delta j+j')c} \right\}_{i'=1, j'=1}^{n,m} \tag{B11}$$

$$\#O_{ijc} = \left(\frac{X}{\Delta i}, \frac{Y}{\Delta j}, l \right) \quad \text{for} \quad p = \mathrm{s} \tag{B12}$$

$$\#O_{ijc} = \left(\frac{X}{\Delta i} - \frac{n}{2}, \frac{Y}{\Delta j} - \frac{m}{2}, l \right) \quad \text{for} \quad p = \mathrm{v}. \tag{B13}$$

Both pooling layers exist also as global versions, indicated by GlobalMaxPool and GlobalAvgPool, where all entries in a channel are considered for either the maximum or averaging operation. In this case, the shape of the output reduces to $(1, 1, l)$.

A concatenation layer performs a stacking operation along the c-axis, which means that the spatial dimensionality of each input I_{ijc} must be the same (X, Y).

$$\mathrm{Concatenate}(I_{ijc_1}, ..., I_{ijc_n}) = O_{ijc} \tag{B14}$$

$$O_{ijc} = I_{ijc_1} \oplus ... \oplus I_{ijc_n} \tag{B15}$$

$$\#O_{ijc} = (X, Y, C_1 + ... + C_n), \tag{B16}$$

where the $\oplus$ operator implements the channel stacking. The respective number of input channels is C_1, ..., C_n for a concatenation of n layers.

Fully connected, sometimes called affine, layers create a linear mapping between the input and the output

$$\mathrm{FC}(n) I_{ijc'} = O_{ijc} \tag{B17}$$

$$O_{ijc'} = B_{ij} + \sum_{k=1}^{X} A_{c'k} I_{ijk} \tag{B18}$$

$$w_{\mathrm{FC}} = \left\{ B_{ij}, A_{c'k} \right\} \tag{B19}$$

$$\#O_{ij} = (1, 1, n). \tag{B20}$$

Here, we assume that the input layer has a simple 1D shape (1,1,X).

B2 Activation functions

We use three kinds of activation functions. Feature extraction layers such as convolution and pooling layers are often followed by ReLUs or its generalization that is commonly called a leaky ReLU

$$\mathrm{ReLU}(x) = \max(0, x) \tag{B21}$$

$$\mathrm{leakyReLU}(x; \alpha) = \begin{cases} x & x \geq 0 \\ \alpha x & \text{otherwise.} \end{cases} \tag{B22}$$

The last fully connected layer in a neural network that is used for classification is often followed by softmax function in order to produce predictions in the final output of the

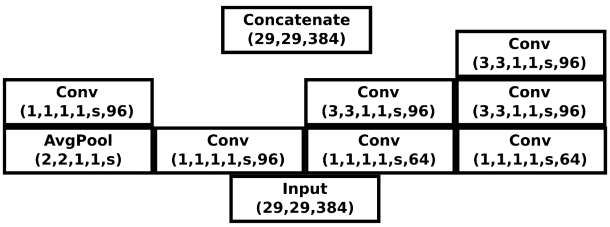

Figure B2. InceptionA layer of our CNN, based on Szegedy et al. (2016).

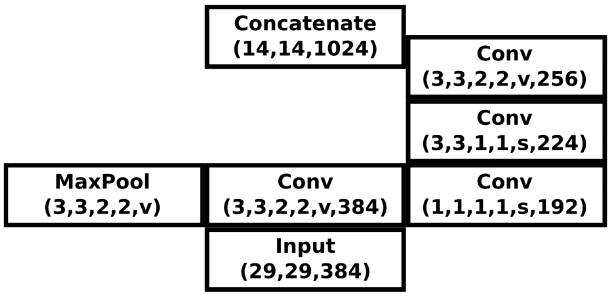

Figure B3. ReductionA layer of our CNN, based on Szegedy et al. (2016).

network

$$\mathrm{Softmax}(x)_n = \frac{\exp x_n}{\sum_{j=1}^{N} \exp x_j} \quad \text{for} \quad n = 1, ..., N. \tag{B23}$$

B3 CNN architecture

In Section 4.3, we described the global structure of our CNN, which is largely based on Szegedy et al. (2016). Here, we describe in detail the purpose of each of the functional elements that are shown in Table 6. After three conventional 3×3 convolutions for initial feature extraction and dimensional reduction, we enter the StemInception layer, which is shown in detail in Fig. B1. In our CNN, the purpose of the stem layer is twofold. First, it further reduces the data vector from 125×125 pixels down to 29×29 pixels, which is a computationally manageable size for applying a large number of convolution channels. Secondly, it already applies a more refined combination of 3×3, 7×7, and 1×1 convolutions. The latter only have the purpose of channel reduction as explained in Szegedy et al. (2016). The stem layer is followed by the three main inception layers A, B, and C. The main purpose of those layers is feature extraction, with a particularly large number of convolutions of varying kernel size. Between the main feature extraction layers, we insert reduction layers, breaking up the image further into smaller postage stamps and allowing the application of a larger number of convolution channels within acceptable runtimes and within the memory constraints of the hardware we deploy. The very last concatenation layer of InceptionC is followed by a global averaging layer and a single fully connected layer for classification.

The ReductionA layer, shown in Fig. B3, consists of a relatively simple combination of 3×3 convolutions and a MaxPooling layer. The purpose of this network module is to reduce the spatial dimension of the images from 29×29 pixels down to 14×14 in order to allow for large convolutions in the following InceptionB layer, which is shown in Fig. B4. This module consists of larger 7×7 convolutions, split into perpendicular stripes for runtime reasons and hence makes an important contribution in the feature extraction process. Reduction layer B, shown in Fig. B5, reduces the image dimensionality further from 14×14 to 6×16 with a rather complicated combination of convolutions. It is followed by the final InceptionC layer shown in Fig. B6, which naturally applies only small convolution but using a particularly large amount of channels.

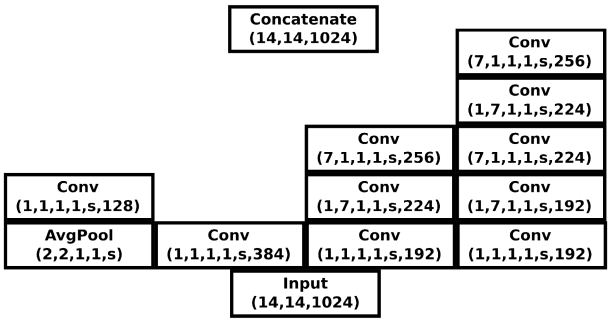

Figure B4. InceptionB layer of our CNN, based on Szegedy et al. (2016).

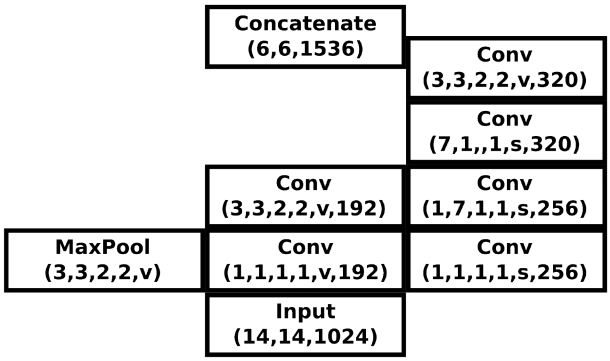

Figure B5. ReductionB layer of our CNN, based on Szegedy et al. (2016).

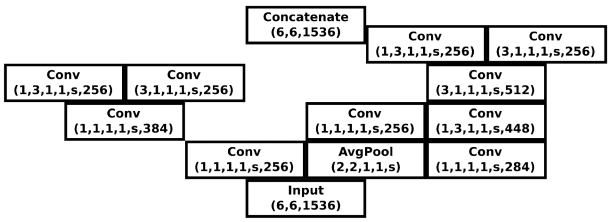

Figure B6. InceptionC layer of our CNN, based on Szegedy et al. (2016).

This paper has been typeset from a TEX/LATEX file prepared by the author.