Improving convolutional neural networks for cosmological fields with random permutation

Kunhao Zhong,* Marco Gatti, and Bhuvnesh Jain
Department of Physics and Astronomy, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA

(Received 18 March 2024; accepted 25 July 2024; published 29 August 2024)

Convolutional neural networks (CNNs) have recently been applied to cosmological fields—weak lensing mass maps and Galaxy maps. However, cosmological maps differ in several ways from the vast majority of images that CNNs have been tested on: they are stochastic, typically low signal-to-noise per pixel, and with correlations on all scales. Further, the cosmology goal is a regression problem aimed at inferring posteriors on parameters that must be unbiased. We explore simple CNN architectures and present a novel approach of regularization and data augmentation to improve its performance for lensing mass maps. We find robust improvement by using a mixture of pooling and shuffling of the pixels in the deep layers. The random permutation regularizes the network in the low signal-to-noise regime and effectively augments the existing data. We use simulation-based inference to show that the model outperforms CNN designs in the literature. Including systematic uncertainties such as intrinsic alignments, we find a 30% improvement over unoptimized CNNs and power spectrum in the constraints of the S_8 parameter for simulated Stage-III surveys. We explore various statistical errors corresponding to next-generation surveys and find comparable improvements. We expect that our approach will have applications to other cosmological fields as well, such as Galaxy maps or 21-cm maps.

DOI: 10.1103/PhysRevD.110.043535

I. INTRODUCTION

The large scale structure (LSS) of the Universe contains crucial information about its late-time growth history and fundamental physics. Over the past few decades, a diverse array of probes has been employed to study these structures, each contributing significantly to our understanding of the Universe. Prominent among these are the Stage-III LSS surveys, which include the Dark Energy Survey (DES) [1–3], the Kilo-Degree Survey [4–6], the Hyper Suprime-Cam Subaru Strategic Program [7–9], and the Baryon Oscillation Spectroscopic Survey [10–12].

The upcoming Stage-IV LSS surveys, including Rubin Observatory's Legacy Survey of Space and Time (LSST) [13], the Roman Space Telescope [14], and the recently launched Euclid mission [15], in combination with the Dark Energy Spectroscopic Instrument [16], are or will be gathering data soon. All these efforts will shed light on two of the most mysterious problems in physics: dark matter and dark energy.

One of the most important analytical tools used in these studies is the two-point correlation function. This function links observational measurements with theoretical predictions derived from perturbation theory. However, the two-point correlation function only captures the Gaussian information.

A variety of non-Gaussian statistics have thus been proposed to study cosmological fields. These include the three-point function and bispectrum [17–19], peaks and voids [20–25], Minkowski functionals [26–28], Betti numbers and persistent homology [29–32], k-th nearest neighbor (k-NN) and cumulative distribution function (CDF) [33,34], and wavelet transform based statistics [35–40]. See Ref. [41] for a forecast with the settings of the Euclid Mission.

Most recently, machine learning has been used to extract cosmological information at the field level, particularly with convolutional neural networks (CNN) [42–52], graph neural networks (GNN) [53–56], and generative models [57–61].

Neural networks generally surpass traditional Gaussian statistics in constraining cosmologies, such as the power spectrum. However, the extent of the improvement can vary significantly depending on the realism of the simulations and the scales considered. When compared to other non-Gaussian statistics, the situation is more complicated. For instance, several studies [42,44] have suggested that CNNs outperform peak count methods, whereas others have pointed out that their enhancement over Minkowski functionals is only moderate [47]. Moreover, there are scenarios where CNNs appear not to perform as effectively as other methods, such as the scattering wavelet transform [35], or even the power spectrum with the presence of more realistic

*Contact author: kunhaoz@sas.upenn.edu

systematics [62]. In other words, even though the CNN often contains thousands of free parameters to approximate the underlying function, it might not be suitable to directly apply the commonly used classification-oriented CNN in cosmology. Such insights are crucial for guiding future research and methodology choices in the field of cosmology, particularly in the integration and application of machine learning techniques.

In this paper, we explore modifications to the architecture of CNNs to enhance their performance in cosmological tasks. Cosmological maps such as weak lensing mass maps differ from images used to develop CNNs in that they are stochastic and low signal to noise. Moreover, we use them for cosmological inference, which is a regression task instead of classification as is typically done in industry. So we explore a variety of regularization and data augmentation schemes in applying CNNs on lensing maps. Our preliminary findings suggest that employing a combination of maximum and average pooling improves the network's ability to extract relevant information. We also introduce a novel regularization scheme rooted in our understanding of cosmological fields: the incorporation of random permutation layers designed to effectively augment the training data by sacrificing the information of large-scale correlation. We demonstrate the effectiveness of this straightforward technique for statistical noise levels expected in current and upcoming weak lensing surveys. We also include a number of sources of systematic uncertainty.

Additionally, we conduct comparative analyses with conventional regularization methods and alternative strategies that disrupt large-scale correlation. Our initial results indicate that the permutation operation directly applied to the latent space of the neural network consistently boosts accuracy in both the simplified and realistic simulation cases considered in this work. This suggests that by selectively filtering out large-scale information in the deeper layers, the neural network exhibits enhanced performance in extracting information from more critical scales. We anticipate that this simple regularization technique holds promise for improving not only weak lensing fields but also other stochastic fields, such as Galaxy maps or 21-cm maps. We do not assert that our architecture represents the optimal model as we have not carried out a systematic study for all possible use cases.

The structure of this paper is as follows: In Sec. II we describe the simulations we used for this work, the preprocessing of the data, and the simulation-based inference setup. In Sec. III we discuss how the CNN for cosmological fields should be different from the standard design for image classification. In Sec. IV we present the random permutation layer as a regularization scheme that helps model generalization. We summarize the findings and outlook for future work in Sec. V, and we perform extensive tests comparing to other models in the Appendixes.

II. METHODOLOGY

A. Simulation of weak lensing convergence field

For this work, we use weak lensing mass maps (i.e., maps of the weak lensing convergence field) created from the `DarkGridV1` N-body simulation suite [63,64]. The `DarkGridV1` suite consists of ΛCDM-only simulations that vary two parameters, Ω_m and σ_8, exploring 57 different cosmologies.[1] Each cosmology is represented by five independent full-sky simulations. The simulations have been produced using the PKDGRAV3 code [65]; the code produces particle number counts at different redshifts (100 redshifts from $z = 49$ to $z = 0.0$), provided as HEALPIX [66] maps. For this work, we downsample the resolution of the original maps to NSIDE = 512, corresponding to a pixel resolution ≈6.9 arcmin. To create weak lensing mass maps, we follow a procedure similar to that in [67,68]. First, noiseless convergence maps are produced from the particle counts for each redshift shell assuming the Born approximation [69]. Noiseless shear maps, for each redshift shell, are obtained from the convergence maps using a generalization of the Kaiser-Squires algorithm [70,71]. Then, we consider two cases.

1. Simulation case 1—no observational systematics or tomography

In case 1, we simplify the map-making procedure, and we choose to not include observational systematics in the modeling. We first generate integrated DES-Y3-like shear maps, by weighting the shear maps as a function of redshift by the DES-Y3 redshift distributions [72]. To simplify and reduce the number of maps generated, we focus on just one of the four DES redshift bins, specifically the third bin. Next, we introduce Gaussian noise to the shear maps. This noise is assumed to have a per-pixel value of $\sigma = \sigma_e/\sqrt{N_{\rm eff}}$, where $\sigma_e = 0.26$ represents the shape noise, and $N_{\rm eff} = 5.6 \times A_{\rm pixel}$ is the effective number of galaxies per pixel. This is equivalent to adding $\sigma = \sigma_e/\sqrt{2N_{\rm eff}}$ to the convergence (mass) map as used in Ref. [43]. Note that here we have used the number density of the full DES-Y3 shear sample, not just that of the third bin. This approach is intended to yield cosmological constraints similar to the entire DES-Y3 survey, even though we are using only one tomographic redshift bin. Next we cut 12 nonoverlapping square patches from the full-sky mass maps; each patch is 512×512 pixels (see Fig. 1), and is chosen to be centered on the center of a HEALPIX pixel with a resolution NSIDE = 1. For the 285 full-sky simulations we generated, we have $285 \times 12 = 3420$ patches in total.

[1]The samples are chosen to follow lines of approximately constant S_8. See Fig. 2 of Ref. [63] for the grid spanning the Ω_m–σ_8 plane.

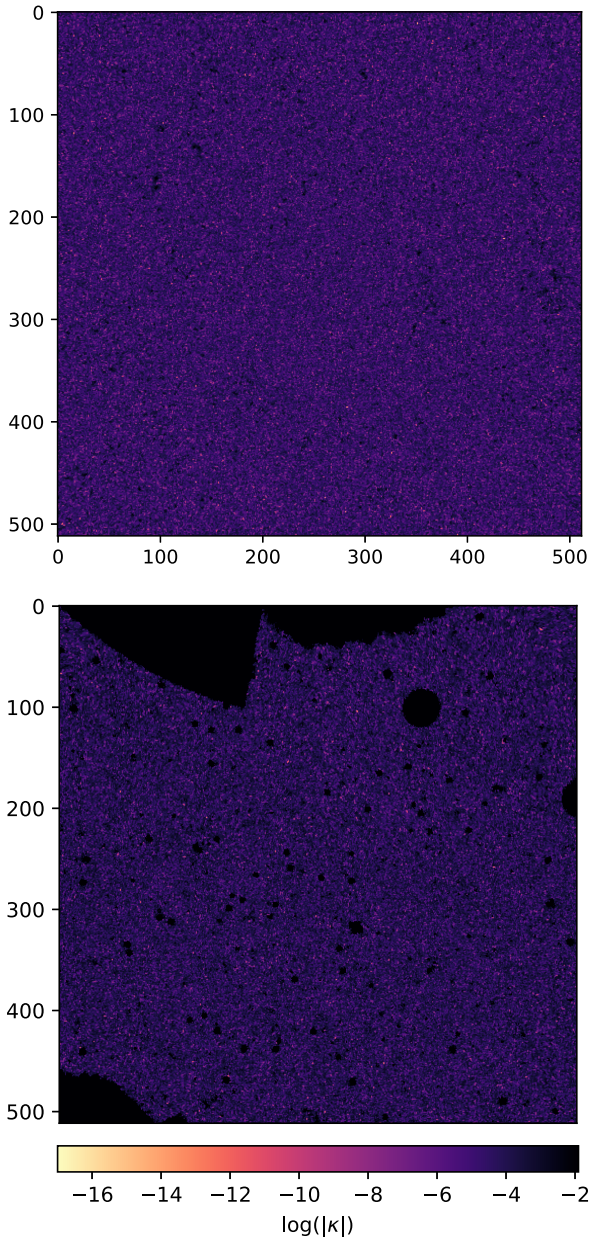

FIG. 1. Upper panel: example patch of the weak lensing mass maps for simulation case 1, in which no mask is applied and only one tomographic bin is used. The side length is 58 degrees and is given in pixel units. Lower panel: example patch for simulation case 2, which includes a mask that follows the DES footprint. The figure shows the third of the four tomographic bins. For visualization purposes, we show the kappa map in log scale, and both figures share the same color range. For the rest of this work when training the neural network and other comparisons, we keep the maps in the physical linear scale and do not normalize them in training.

2. Simulation case 2—full map-making procedure

In case 2, we produce weak lensing maps using the full procedure implemented in [67,68]. The procedure produces four maps, one for each of the four DES tomographic bins. Moreover, a DES footprint (fixed mask for bright galaxies; see Fig. 1) is applied. Each map also includes observational and astrophysical systematics, namely redshift uncertainties, shear biases, intrinsic alignment [in the form of the nonlinear alignment model], and source clustering. These are managed using various nuisance parameters. Practically, for each map, we forward model these effects by randomly selecting the nuisance parameters that control them from their respective priors—see [68] for details. In this case, we only train using a single patch of the same galaxy mask, approximately centered at the center of the DES footprint (see Fig. 1). For each full-sky simulation with the same cosmology, we shift and rotate the DES footprint four times to generate four nonoverlapping maps. For the 2280 full-sky simulations we generated, we thus have $2280 \times 4 = 9120$ patches in total.

In both cases, the training set is images of size 512×512 with a resolution of 6.9 arcmin, which corresponds to 3367 degree^2. In case 1 images have one channel, while in case 2 images have four channels, one for each tomographic bin. These maps are significantly larger than previous works [42–44,47,49,50,73] allowing us to use one patch to represent the survey area. Using a larger patch also has the benefit of keeping more modes. We tested the six-layer CNN model in comparison to cutting to smaller patches of size 256×256, or 128×128. Although the total training set is larger with smaller patches, we observed a lower accuracy of the CNN, especially in the case of galaxy masks. Therefore, we use the large 512×512 image in this work. Note that the large sky area of a single projection breaks the flat-sky approximation. This in theory would not produce a biased result since it is part of the data compression for simulation-based inference as discussed in Sec. II C. However, in the future we will extend the work to the curved sky, e.g., with structures as presented in [53].

B. Data augmentation and training

Prior to training, we add random Gaussian noise to each pixel as a data augmentation technique, particularly effective for noisy images [74]. In our methodology, we apply Gaussian noise with a mean equal to the image's mean value and a standard deviation equivalent to 10% of the original image's standard deviation. In this work, we add this to all images (training, validation, and test). The noise level remains almost unchanged since it increases by $\sqrt{1+0.1^2}$. Note that since the additional noise is added according to each image and each channel, it is different from the global shape noise where the variance is fixed for all maps. While Ref. [43] demonstrates the efficacy of augmenting data through random rotation and flipping, we observe no significant difference with these methods. The difference may be attributed to our approach of extracting patches from nonoverlapping regions of the full sky. Although our data augmentation strategy differs from previous studies, we emphasize its importance as it

improves the test error by ∼25%. This approach is also applied to the entire masked map, ensuring that the augmented map assigns nonzero values to the masked regions. This procedure can be optimized for future exercise. For example, using a dynamic approach such that different noise is added while training, the strengths of the noise can also decay away as the model is trained for more epochs.

We train separate CNNs for inferring S_8 and Ω_m, as the training time is short (15 mins for simulation case 1, and 1 hour for simulation case 2 on one A100 GPU). This approach also mitigates the issue of degeneracy when inferring two parameters simultaneously in a single network. In this work, we chose the mean square error (MSE) as the loss function, and we report both the root mean square error (RMSE) and the coefficient of determination (R squared) in the test results, which provides complementary information of goodness of fit. Although MSE is widely adopted, alternative loss functions can significantly influence results. For instance, had we noticed a tendency for biased predictions at extreme input values, adopting a log function could have been beneficial to more heavily penalize such discrepancies. Since we did not have such biases for the prediction, we kept MSE in this work. The dataset is randomly divided into 80% for training, 10% for validation, and 10% for testing. The last 10% test set data is then used for the simulation-based inference. We note that in [43] a more careful division method based on the initial conditions of simulations is utilized. Since our simulation only has 58 different cosmologies, we expect this not to be an issue as the validation set effectively covers the full sampling regions.

The CNN is implemented using PYTORCH [75] and trained using the ADAMW [76] optimizer. We use a batch size of 16 and a total epoch of 200. A learning rate scheduler, REDUCELRONPLATEAU, is used with a reduction factor of 0.3 (default is 0.1). For all other hyperparameters not explicitly mentioned, we use PYTORCH-V1.13's default settings. Due to computational resource limits, we did not show every test with averaging over different random seeds. The random seeds in training affect the weight initialization and also the training-validation split, and thus affect the final results. However, we tested in a few cases with different random seeds and found the variance of RMSE to be as small as $\Delta\mathrm{RMSE} \sim 0.01\%$, which is partially because we trained for long enough epochs. We hold hyperparameters with respect to training the same for all tests in this paper. We observe effective convergence with this substantial number of epochs. However, it is noteworthy that quasi-Newtonian optimization methods, such as L-BFGS [77] or the Levenberg–Marquardt algorithm [78], are theoretically more suitable for parameter inference where accuracy is more important. Yet, their application often demands substantial memory due to the Jacobian computation. An effective quasi-Newtonian optimizer should be helpful for precision cosmology.

C. Simulation-based inference

We do not use neural networks to directly infer posteriors; for that one usually needs to assume a Gaussian likelihood. Instead, we use the neural density estimator (NDE) package PYDELFI introduced in [79]. Our CNN thus serves as a field-level data compressor similar to the DEEPCOMPRESSOR in [48]. PYDELFI provides different flow models for neural density estimation. In this paper, we use masked autoregressive flows (MAF) [80], which is a stacked version of masked autoencoders for distribution estimation [81] with normalizing flow. The network of the NDE is trained to minimize the Kullback-Leibler (KL) divergence [82] between the true probability distribution and the proposed probability distribution. The true likelihood is of course unknown, but it is reduced to a constant in calculating the expectation value of the KL divergence over the implicit prior (the prior set by the distribution of the point cloud of parameter values used for the training simulations). For details, see [48,79,83].

In this work, we stack two MAFs with two and three hidden layers of length 50. Since one patch corresponds to approximately 3367 square degrees of the sky, we infer the posterior using only one image. Note that the simulations used for NDE training are the *test set* that is not used for CNN training or validation. After the isolikelihood surface is learned, we use the affine Markov chain Monte Carlo (MCMC) sampler EMCEE [84] to get the final posterior with a flat prior of $S_8 \in [0.45, 1.1]$ and $\Omega_m \in [0.1, 0.5]$. For the purpose of forecasting, we average over nine predicted values of the same underlying cosmology as the target vector (in our case S_8 and Ω_m). This way the posterior is more centered and minimizes the effect of the prior, which allows us to make a more direct comparison of the constraining power.

We follow previous work on the validation for simulation-based inference (SBI) via the empirical coverage test [85–87], which plots the expected coverage probability (usually from the highest posterior density) vs the credible region. For a calibrated posterior with enough samples of the coverage test, one should find that the true parameter is contained within the $X\%$ credible region for $X\%$ times of the tests. This plot, often called the p-p plot, is not unique to SBI as it is used for comparing two probability distributions in general. In this work, we use a test of accuracy with random points (TARP) [87], which is a more computationally efficient way of performing the coverage test. It is important to note that the coverage test aims to verify that the posterior learned from NDE is neither overconstrained nor underconstrained. A failed result would suggest that the NDE network is not suitable for the given problem, e.g., leading to overfitting. As shown in Sec. IV, the results we obtain are well calibrated.

SBI, also called likelihood-free inference, is an emerging field and has been applied to cosmology in several recent studies [83,88–90]. The results for the inferred likelihood

have been tested for summary statistics that have an explicit form of the likelihood such as the power spectrum [40,48,91]. However, a robust convergence criterion and calibration of misspecification are missing. Various other robustness tests have been proposed to check that the results are not biased or overestimated [92,93]. Since the main topic of this paper is to improve the CNN for data compression, we do not further examine the robustness of SBI in the presence of model misspecification.

III. CNN FOR COSMOLOGICAL FIELDS

Unlike the prevalent use of CNNs in image recognition tasks, their application in cosmology typically addresses regression problems rather than image classification. In this context, the precision of the output is most important. Moreover, the input data in cosmology, in particular weak lensing fields, significantly differ from conventional images, such as those of cats and boats and others that are part of ImageNet [94], the long-standing industry benchmark for image classification algorithms. As depicted in Fig. 1, the weak lensing field is stochastic, low signal to noise, and has correlations on all scales. We seek a CNN approach that is simple and generalizable, rather than adopting complex architectures like ResNet [95] or VGGNet [96] that have been optimized for classification tasks on images in other domains. Prior research [42–44,47,49,50,73] employing CNNs for analyzing weak lensing fields has yielded promising results using relatively simple architectures.

Our six-layer CNN model, as shown in Fig. 2, incorporates a notable departure from previous designs [42–51], where either the maximum pooling or the average pooling is used. Instead, we have combined the use of maximum pooling and average pooling layers. One motivation for us to use maximum pooling in the initial layers is driven by the observation that positive extreme values in our pixels correspond to galaxy clusters which are known to carry valuable cosmological information. Indeed, a study of CNNs using saliency maps for lensing mass maps [47] found that clusters are more informative than voids in the presence of realistic noise. Conversely, deeper layers, correlating distant sky regions of the original map, may require equitable consideration. Here, our choice to switch to average pooling facilitates better generalization in the subsequent multilayer perceptron (MLP) layers. We test these choices as discussed below and in Appendix. A.

We tested both simulation cases 1 and 2 and also increased the effective galaxy density (lower noise) to approximately represent future LSST and Roman weak lensing surveys. As shown in Table VI we reduced the noise by assuming 2× and 4× the effective number density of source galaxies, which roughly span the range of expectations for LSST Year 1-10 data (we do not attempt to simulate the redshift coverage of those surveys). However, the best pooling option does not appear to correlate with the noise level. In fact, the average pooling is similar to or better than the mix pooling. When training with average pooling on $\Omega_{\rm m}$ though, the network fails to converge. We also tested extreme pooling with the largest absolute value (which then includes underdense extrema) and found no difference in the final results. For this reason, we use the mixed pooling as our baseline choice. One caveat is that the extreme valued pixels could also be the ones most affected by noise or limitations in the simulations or unmodeled effects; this becomes more of a concern for smaller pixel values than ours. Adopting pooling layers in a Bayesian way could potentially overcome this problem [97].

It is important to note that this architecture is not optimal for every scenario, such as when dealing with patches of different sizes or different choices of systematic uncertainties. Furthermore, we only optimize the architecture for the inference of S_8 while a better design for $\Omega_{\rm m}$ is likely different and merits follow-up work.

Nonetheless, our findings below show that simple, physics-inspired modifications to a CNN can significantly improve accuracy. This encourages the pursuit of tailored models for specific problems in cosmology, rather than relying on existing architectures primarily designed for image classification.

IV. CNN WITH RANDOM PERMUTATION LAYERS

Adding randomness is a common way in machine learning to improve the generalization accuracy. In this section, we introduce such a technique: random permutation of the pixels in the deeper layers of the network that correspond to large scales.

Figure 2 illustrates our design and the positions where we replace the normal design with the random permutation layers. The network is designed similarly to a VGG net [96]. In this work, we use $n_{\rm ch} = 8$ channels in the first layer, with $n_{\rm ch}$ increasing by a factor of 2 for successive deeper layers. The three "options" shown involve replacing the deeper layers with a random permutation layer followed by convolution and activation at different depths as indicated (note that the arrows do not indicate skip connections). There is always an extra convolutional layer with an activation function followed by the random permutation layer to make the shuffling effective and efficient. The random permutation layer shuffles (randomizes) the position of each pixel of the feature map.

As an example, if we add the permutation layer before convolution layer 6 (option 1), the input is the feature map of size 8 by 8 for 128 channels. The random permutation operation shuffles these 8×8 pixels randomly, and we perform different shuffles for each 128 channel. Each pixel in this feature map approximately corresponds to $512/8 \times 6.8 \approx 435.2$ arcmin. At this scale, the two-point

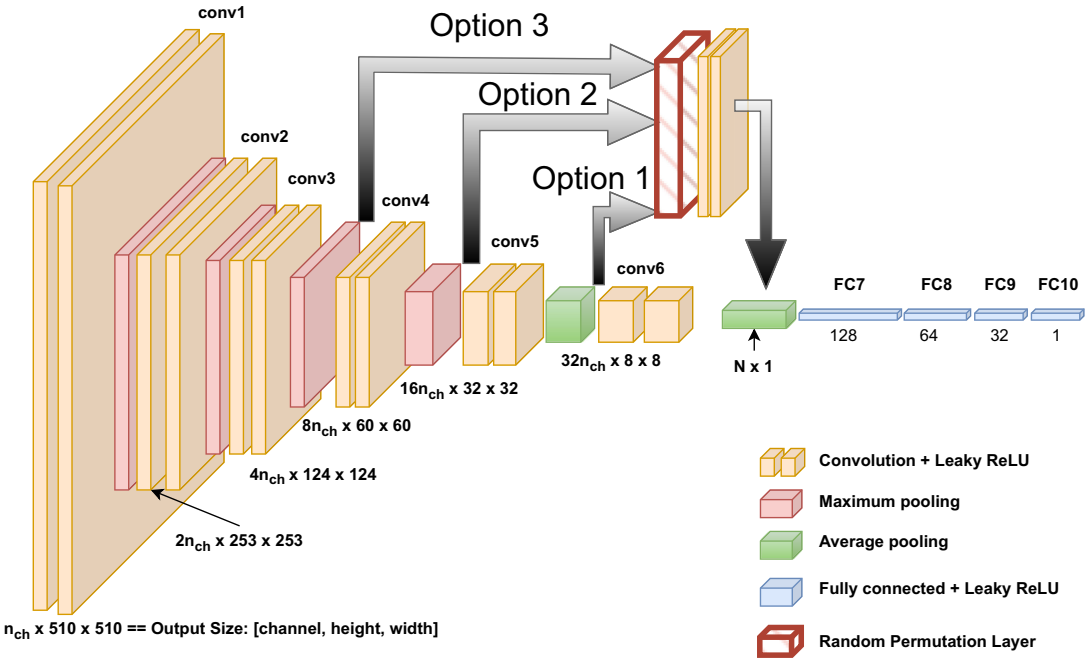

FIG. 2. The six-layer CNN design used in our study, including the optional random permutation blocks. The network is designed similarly to a VGG net [96]. In this work, we use $n_{\rm ch} = 8$ channels in the first layer, with $n_{\rm ch}$ increasing by a factor of 2 for successive deeper layers. The three "options" shown involve replacing the deeper layers with a random permutation layer followed by convolution and activation at different depths as indicated (note that the arrows do not indicate skip connections). In option 1, one extra random permutation layer is added without any change to the six-layer CNN design. For option 2 and option 3, the random permutation is performed at earlier layers. There is always an extra convolutional layer with an activation function followed by the random permutation layer to make the shuffling effective and efficient. See Sec. IV for more detailed discussions. Note that we also mix average and maximum pooling layers: see Sec. IV for details. The blocks shown in this figure are not to scale.

correlation has a large sample variance. We find that by removing the information at large scales in the deeper layers, the neural network optimizes for the scales that are more relevant for cosmology. Moreover, it appears to extract some of the large-scale information in the earlier layers, though as discussed below we do not yet understand exactly how it does that. The shuffling operation can be inserted at different positions in the network, depending on the scale of which noise dominates over the signal. Note that this way of calculating scales is only approximate.

In this work, the shuffle is different for each batch, each channel, and changes every epoch. Note that the four tomographic bins have the same random permutation (shuffle) applied.[2] The convolution operation makes this shuffle effective. Otherwise, the data would be the same with the global pooling. One can also flatten the deep layer pixels, but that would increase the MLP size, and we show in Table III that it decreases accuracy. We perform additional tests in Sec. IV A.

A pseudocode in PYTORCH is displayed in Algorithm 1. Only the lines associated with the random permutation operation are shown. Note that the SELF_TRAINING variable might not be effective depending on the version of PYTORCH. In that case, a customized variable should be used. This is also optional as in some cases random shuffling in both training and validation could boost the overall accuracy. Similar to this, we can also easily define the model to only shuffle the pixels after every N epoch.

We test the positions of the shuffling operation for simulation cases 1 and 2. As shown in Fig. 3, the accuracy depends on where we perform the shuffling. For simulation case 1 it is the second to the last layer, and for simulation case 2 it is the third to the last layer. The simplification effectively makes them five-layer and four-layer models respectively. In general, we expect the results to depend on the underlying variation of cosmologies, the noise level, tomographic binning, patch size, training set size, and other details. In particular, it will be interesting to see how increasing sample variance at large scales impacts the choice of the network [98]. To ensure a fair comparison, we also varied the depth of the nonshuffled CNN and found the six-layer one works the best (see Appendix B). For comparison with traditional Gaussian 2-point statistics, we include the results of the power spectrum as well. We present the details of training 1D vectors in Appendix F. The errors on the test set for both simulation cases are

[2]This is due to the fact that we use the Conv2d function implemented in Pytorch https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html.

ALGORITHM 1. Pytorch pseudocode of CNN with random permutation.

```
class ShuffledCNN:
  def __init__(self):
    //Initialize CNN layers here

  def randomize_images(self, tensor, image_size):
    x_flat = tensor.view(-1, image_size*image_size)
    idx = torch.stack([torch.randperm(image_size*image_size) for _ in range(x_flat.size(0))])
    if tensor.is_cuda:
      idx = idx.cuda()
    x_randomized_flat = torch.gather(x_flat, 1, idx)
    x_randomized_flat.view(tensor.shape[0], tensor.shape[1], image_size, image_size)

  def forward(self, x):
    // Implement the initial forward pass here
    // Randomly permute the feature maps at corresponding position
    if self.training: // This is optional
      x = self.randomize_images(x, x.shape[-1])
    x = self.conv6(x)
    x = self.LeakyReLU(x)
    x = self.pool6(x)
    // Implement the rest forward pass here
    return x
```

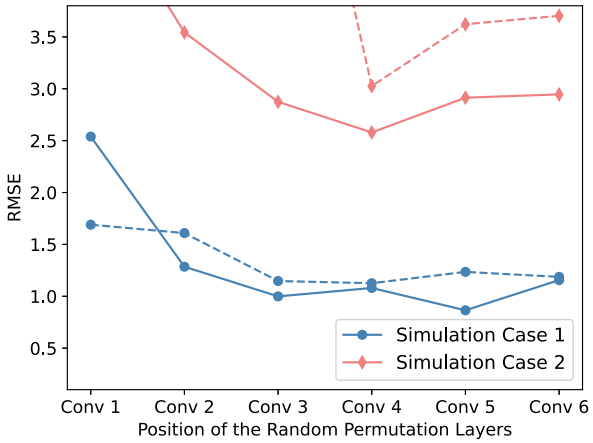

FIG. 3. The test errors for two simulation cases are shown for different positions of the random permutation layers. Solid lines mean the shuffling case and dashed lines for the corresponding network without shuffling. "Conv 6" denotes adding the shuffling operation before the final convolution, namely option 1 in Fig. 2. On the other hand, "Conv 1" refers to the first layer, so every pixel of the input image is randomly permuted (which leads to a higher loss as expected). While it is clear that shuffling is effective only in the deeper layers, the best shuffling position varies for the two cases and is not necessarily the final layers—possibly because either option 2 or 3 leads to a simplified MLP structure (see Fig. 2). We also notice that the training fails to give meaningful results if the network is too shallow and without shuffling for regularization.

shown in Fig. 4 and Fig. 5 respectively. The posteriors for the two parameters are shown in Fig. 6. We found that the CNN with shuffling the standard deviation of S_8 is 30% smaller than the nonshuffle case. When compared to the power spectrum, the unoptimized CNN does not give more constraining power by itself, which is similarly found in the recent study that uses similar simulations and map-making processes [62]. We also show the results that combine the CNN and power spectrum. We concatenate the two compressed data vectors and retrain the NDE to learn the likelihood surface. The nonshuffled CNN benefits a lot from the combination with the power spectrum, giving a 20% smaller standard deviation in S_8, whereas the shuffled-CNN only improves by 8%. Comparing the CNN + PS results to the power spectrum alone, we found 27% improvement for nonshuffled CNN and 36% for shuffled CNN.

Note that the CNN for the two cases is optimized separately, and indeed it is also different for S_8 and Ω_m. In particular, it is the six-layer model for nonshuffle for S_8, the four-layer model for nonshuffle for Ω_m, and four-layer model for shuffle CNN (option 3). To make the comparison for the same CNN model with the same weight size, we use shuffle option 1 for S_8 as well and get a 17% improvement on the standard deviation of S_8. However, it is important to note that this change of layer depths only helps the shuffle CNN, while the nonshuffle CNN suffers from insufficient depth in our case.

We applied the TARP coverage test with 58 simulations (as there are only 58 independent cosmologies for the S_8-Ω_m plane), and found the posterior is well calibrated as

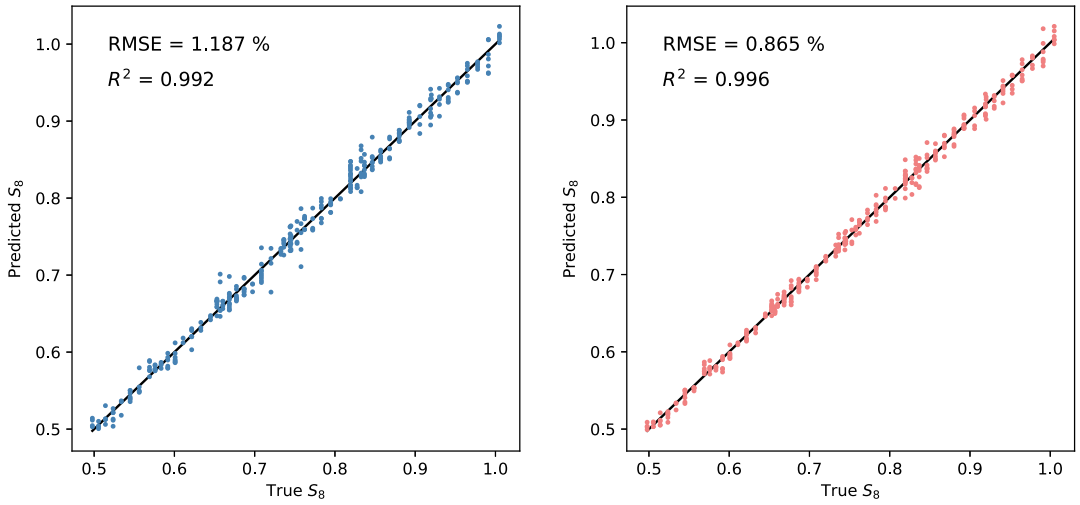

FIG. 4. Comparison of the test error in S_8 with and without shuffling (random permutation) for simulation case 1 (case 2 is shown in Fig. 5). Left panel: six-layer CNN with no shuffle. Right panel: shuffled CNN (option 2 in Fig. 2 which corresponds to a five-layer CNN), which corresponds to removing correlations for scales larger than 109 arcmin. Note the significant improvement due to shuffling for the full range of values of S_8. The two panels in this figure represent the best results we found for nonshuffle and with shuffle.

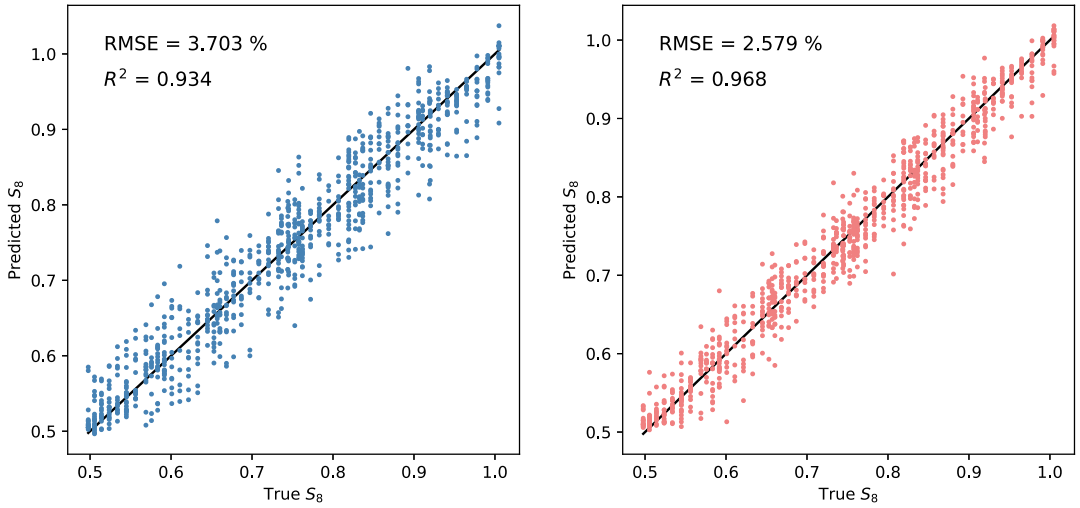

FIG. 5. Comparison of the test error in S_8 with and without shuffling for simulation case 2 (with mask and tomography). Left panel: six-layer CNN with no shuffle. Right panel: shuffled CNN (option 3 in Fig. 2 which corresponds to a four-layer CNN), which removes correlations for scales larger than 56 arcmin. The two panels in this figure represent the best results we found for nonshuffle and with shuffle. (If we shuffle at deeper layers, the performance is degraded to RMSE $\approx 3.0\%$ but is still better than the nonshuffle case).

shown in Fig. 7. As an additional sanity check, we applied our CNN on Gaussian random fields and tested that the CNN does not give a more constraining posterior than using the power spectrum as the summary statistics.

A. Experiments to understand the improvement

To get insights into the source of these improvements, we conducted a series of experiments, varying the models and altering the training datasets. We plot in Fig. 8 the loss function decay for two models with and without shuffling. The six-layer CNN nonshuffle model can fit the underlying data very well as suggested by the training loss, but it does not generalize well to the validation or test set. The four-layer CNN with shuffling, however, can generalize very well, even though it does not fit the training set as well. This demonstrates that the shuffling technique is effective in preventing overfitting. One hypothesis is that the six-layer nonshuffle model suffers from coadaptation—a scenario where certain neurons become highly interdependent, leading to overfitting with new test inputs [99,100]. The shuffling prevents any large-scale outliers from significantly affecting the results.

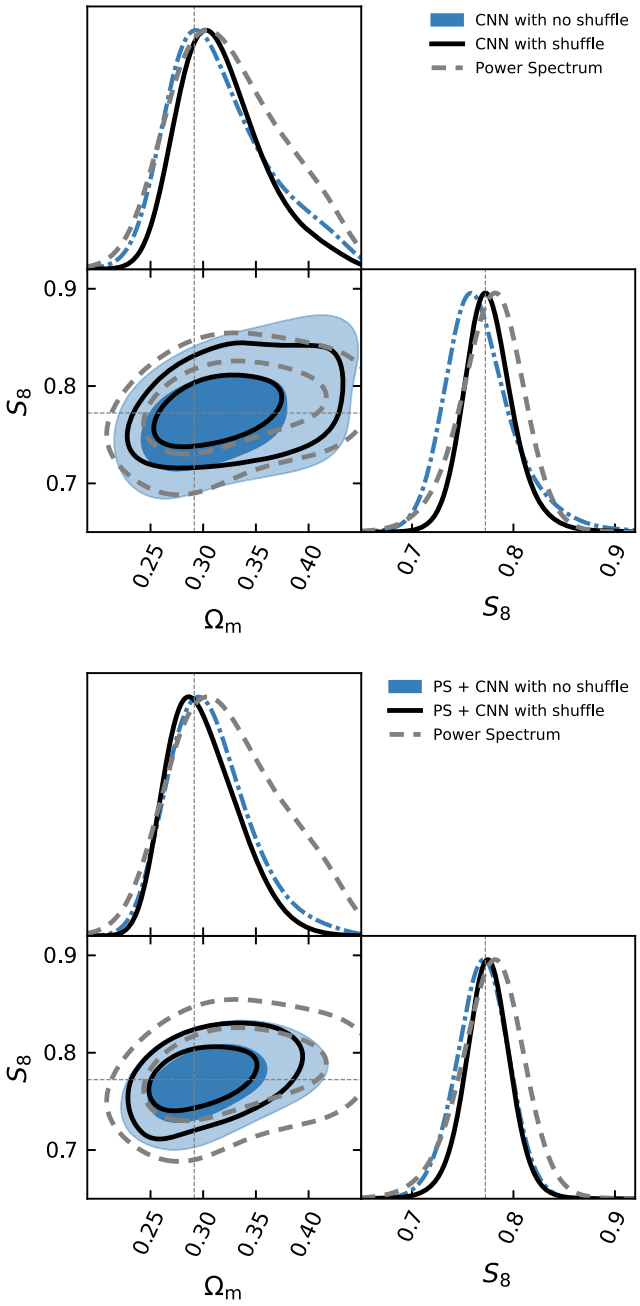

FIG. 6. The posterior inferred using simulation-based inference for simulation case 2. Upper panel: comparison between using two CNNs and the power spectrum. Comparing the optimized CNN with shuffle regularization, we found the improvement in the standard deviation of marginalized S_8 to be 30% over the nonshuffled CNN and power spectrum. The Figure of Merit $= 1/\sqrt{\det \mathrm{Cov}(\Omega_m, S_8)}$ is 50% larger than the nonshuffle CNN and power spectrum, although this may not be the best metric as the posterior is very non-Gaussian. Lower panel: the posterior inferred when combining CNNs and the power spectrum. For the nonshuffled CNN with the power spectrum, we find 20% tighter constraints on S_8 than nonshuffled CNN alone. For the shuffled CNN with the power spectrum, we found 8% tighter constraints on S_8 than shuffled CNN alone. These combinations provided 36% and 27% better constrains on S_8 than the power spectrum, respectively. See Sec. IV for detailed discussion.

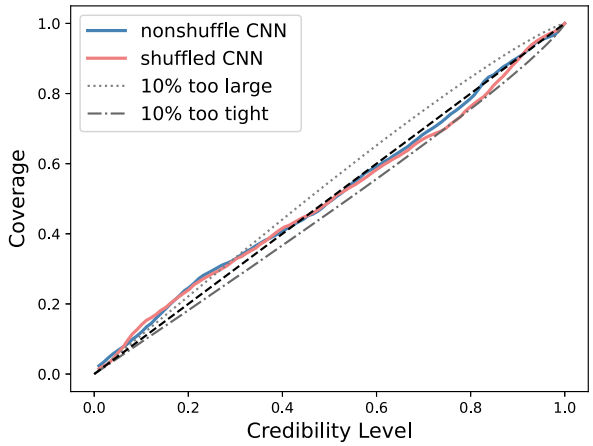

FIG. 7. Empirical coverage vs credibility level. A well-calibrated posterior closely aligns with the identity line, indicating that the CNN compression is neither overconstrained nor underconstrained. The lines are generated with TARP [87], as discussed in Sec. II C.

One case of special interest is when we directly shuffle the original images. As shown in Fig. 9, the result worsens but by less than a factor of 2, which is not as bad as one might expect in the context of standard image analysis, since no structure of the image is retained. However, in cosmology, it is known that the 1-point probability distribution function (PDF) of the image pixels has useful information as it is the PDF of the projected density. In particular, it includes its variance, the smoothed two-point function. The pixel scale in our original image corresponds

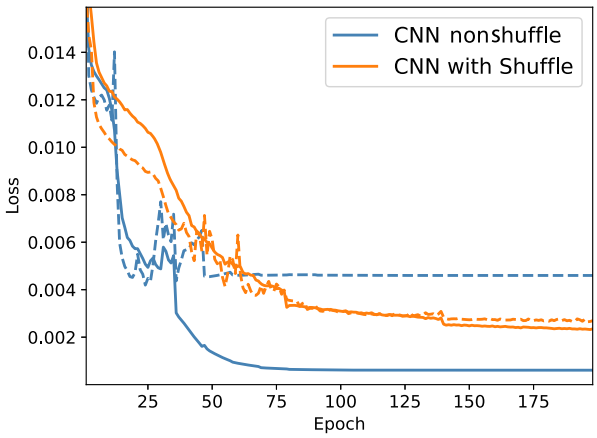

FIG. 8. The MSE loss vs training epoch. The solid line shows the training loss while the dashed line shows the validation loss. The blue lines are for the CNN without shuffle, and the orange lines are for the shuffled CNN (option 3 in Fig. 2). The six-layer CNN without shuffle can fit the training set very well, but the generalization performance is not good as is evident from the higher validation loss. This gives an accuracy boost for the models with random shuffling. The loss decay curve shows some differences from case to case, but this is a typical case of how shuffling improves the CNN.

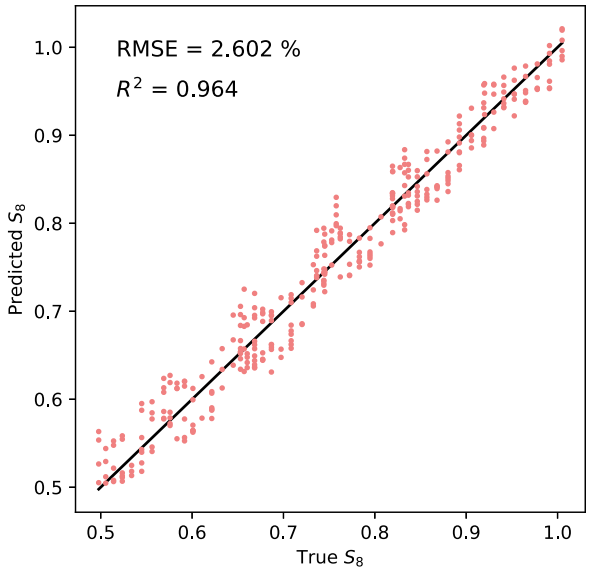

FIG. 9. Test error for simulation case 1 with the random permutation as the first layers, followed by a $[1 \to 8]$ convolution layer (eight different kernels to learn). The image is entirely disrupted. The maximum information in each channel is thus the PDF of pixel values. However, if we directly train an MLP using mean, variance, or the full PDF, we get RMSE $\sim 7\%$ at best. See Appendix F for details. This example shows that the convolution operation is a computationally efficient way of summarizing different information, even when the input of the convolution has no spatial correlation.

to a few Mpc at the lens redshift. We test that if we simply measure the average of the entire image, or even use the PDF as summary statistics, the RMSE is ∼7%, which is significantly worse. This example shows that even if all the information on scales larger than the single pixel value is destroyed, the convolutional layer (plus subsequent nonlinear operations) can extract more information than just PDFs. We do not pursue this point further in this study.

Next, we test cases where we removed the large-scale correlation directly from the map. We do this by setting the spherical harmonic coefficients $a_{lm} = 0$ for $\ell < \ell_{\rm min}$ when making the map for simulation case 2. The results are summarized in Table I. On removing the large-scale correlations, the relative improvement from shuffling is decreased, which makes sense as sample variance in the large-scale modes is also removed by setting $a_{lm} = 0$ (so the CNN was not learning the noise in the nonshuffle case). For different shuffling options, the variance is larger but still follows the pattern in Fig. 3.

We note that if the six-layer CNN model (no shuffling in training) undergoes a *post hoc* permutation in the final layer during testing, the error only marginally increases from 1.1% to 1.7%. This suggests that, typically, the last convolution layer is not optimized to capture spatial correlation but instead functions more as the global average and optimizes performance by its weighting of the different

TABLE I. This table summarizes the results for the S_8 predictions when we remove the large-scale correlation from the maps by setting the spherical harmonic coefficients $a_{lm} = 0$ for $\ell < \ell_{\rm min}$. The goal is to understand the shuffle CNN better as discussed in the text. $\ell_{\rm min} = 76$ approximately corresponds to 108 arcmin, and $\ell_{\rm min} = 148$ approximately corresponds to 56 arcmin. The column is the percentage improvement over the CNN model without shuffle. In both cases, the overall accuracy is decreased because information is removed from the maps. The relative improvement of shuffling option 1 is also decreased because shuffling at large scales becomes less relevant.

Models	RMSE (%)	R^2	ΔRMSE
Case 2 with NO $\ell_{\rm min}$			
Six-layer CNN (no shuffle)	3.703	0.934	···
CNN with shuffle option 1	2.946	0.958	20.4%
CNN with shuffle option 2	2.914	0.959	21.3%
CNN with shuffle option 3	2.579	0.968	30.5%
Case 2 with $\ell_{\rm min} = 76$			
Six-layer CNN (no shuffle)	4.047	0.923	···
CNN with shuffle option 1	3.837	0.930	5.2%
CNN with shuffle option 2	3.261	0.950	19.4%
CNN with shuffle option 3	2.953	0.959	27.1%
Case 2 with $\ell_{\rm min} = 148$			
Six-layer CNN (no shuffle)	4.476	0.903	···
CNN with shuffle option 1	4.095	0.918	8.5%
CNN with shuffle option 2	3.549	0.939	20.7%
CNN with shuffle option 3	3.240	0.949	27.6%

channels. We note that the tests discussed in this section are exploratory; a detailed analysis of regularization methods in cosmology is interesting for future research.

B. Other tests, symmetries, and permutation invariance

We perform various tests of regularization in comparison with the random permutation layers. In Appendix B, we show seven additional models that share the same goal as shuffling—to impose permutation invariance at certain layers. For example, we replaced deep layers with an adaptive average pooling layer that simply takes the average value of the 32×32 feature maps to one value. We also tested using other permutation invariant quantities like variance, maximum, and minimum. The results suggest that they are not as effective as the shuffling option. In Appendix C we compare with two other regularization schemes, DROPOUT and batch normalization. We showed that they negatively affect the accuracy in our case. As a supplementary test, we show in Appendix E the results for different noise levels and training set sizes. Across various test scenarios, the addition of random permutations consistently elevates the generalization accuracy.

Constraining neural networks with physical symmetries has been shown to help the training and the accuracy of the

model. In cosmology particularly, translation and rotation symmetries are key ingredients to improve performance. Previous work that imposes these symmetries in normalizing flows [57] and a graph neural network [55] demonstrate great potential for both generative models and models used for parameter inference. The scattering wavelet transform [101] is powerful for cosmological fields partially due to its translation invariance construction. The convolution operation is translation equivariant by definition. However, note that the max-pooling operation breaks the translation equivariance so the CNN is not rigorously translation invariant. Investing in architectures suitable for cosmology with translational [102] and rotational symmetry [103,104] is a promising future direction—we leave such investigations for future work.

Permutation invariance, although less relevant in CNN, is very common for GNN [105]. JANOSSY POOLING [106] employs a random sample of a subset of possible permutations as an alternative pooling operation. Such tweaks to the training process can also be found in CNNs. In particular, in SHUFFLENET [107] a channel shuffle preceding a group convolution is employed. A more extreme case can be found in [108], where the authors disturb the loss layer by directly giving wrong labels, and they show it boosts the neural network's ability to learn more general features.

In this work, we shuffle the feature map for every epoch but only for training. Other variations of random permutation are also worth exploring—for example, we tested shuffle every 10 epochs, which allows the convolution layer to learn the same map within these 10 epochs. The test error is similar to that of shuffling every epoch. One can also introduce a probability distribution where say only 10% of the time the feature map is shuffled. We expect that shuffling every N epoch to be a more general solution that can be optimized for other cases.

V. DISCUSSIONS AND CONCLUSIONS

CNNs are the most extensively developed deep learning approach for image analysis. In recent years they have also been applied for inferring cosmological parameters from weak lensing and other cosmological maps. We show that for the characteristics of cosmological fields like lensing mass maps, variations of the standard CNN architecture can lead to improved performance. We have presented a set of regularization and data augmentation methods and quantified the performance improvement for simulated lensing surveys. The simulated surveys mimic the Dark Energy Survey parameters in both statistical and systematic uncertainties, but we also consider the lower shape noise levels corresponding to Stage-IV surveys. The explorations in this paper are not definitive as the detailed implementation should be adapted to the specific applications.

We use random permutations (shuffling operation) in the deep layers of CNN as a novel regularization and data augmentation technique. We also use a mix of maximum and average pooling for varying layer depths. These simple modifications can enhance the performance compared to traditional CNN designs that have been adapted from image classification. Figure 6 demonstrates the improvement in parameter inference. We also show in Fig. 8 that it can be expected to have better generalization accuracy. While the design requires optimization for specific problems, incorporating a random permutation layer emerges as a promising approach for cosmological fields. An open question that we have addressed only partially is whether CNNs like the ones we have tested lose some large-scale information. It is possible that the early layers of the CNN, prior to shuffling, have extracted the available signal. We find some evidence for this when considering the power spectrum—adding the power spectrum, including the large scales (small angular wave numbers), to the MLP does not improve the performance further. We leave a detailed investigation for future work.

With the upcoming Stage-IV surveys (Euclid, Rubin-LSST, and Roman), we expect a substantial improvement in the quality and size of lensing mass maps. Deep learning approaches along with higher-order statistics and field-level inference can extract the vast amount of information from these maps that goes beyond standard 2-point statistics. While CNNs may not capture all the information contained in cosmological maps, they offer some clear advantages. As a model-specific data compressor, CNNs are flexible for a lot of different settings. For example, if the data have multiple tomographic bins as in current weak lensing surveys, CNNs can take them as different channels and capture the cross-channel information. If one prefers to analyze data with a shear field instead of the convergence field, one can also treat the two components as different channels and avoid the complexity of reconstructing the convergence map. Hence exploring improved performance with CNNs is likely to be of value in cosmological applications.

VI. CAVEATS AND FUTURE WORK

This paper has focused on testing a set of regularization and data augmentation schemes with a simple CNN design. Given that this approach is inspired by our understanding that cosmological fields are noisy on scales approaching the survey size, we anticipate that the random permutation layer, when applied at appropriate scales, could be synergized with more complex architectures. For instance, similar shuffling operations can be easily tested with Vision Transformer [109] or Graph-based Neural Networks [53].

It is also interesting to test if this regularization and data augmentation method can be effective in other cosmological fields such as the overdensity field [110–113], 21-cm maps [114–119], and secondary anisotropies in cosmic microwave background (CMB) temperature or polarization maps [120,121].

A useful direction for future research is the optimization of the shuffling operation. One possibility is to introduce a hyperparameter that modulates the probability of performing the shuffle, which is a technique used in [122]. Our limited explorations are conservative in the sense that we did not tune hyperparameters, so there is room for improvement by tuning all the hyperparameters for different variations of the noise level. Additionally, exploring other randomness-incorporating structures, such as random shifting [123] or randomly wired CNNs [124], could also offer valuable insights.

Another important direction for future research is in the interpretability of CNNs when applied to cosmological data. Despite their proven efficacy in regression problems, a persistent challenge with CNNs is the "black box" nature of their decision-making processes. Understanding how these networks derive their conclusions is important, not just for validating and improving accuracy, but also for gaining deeper insights into the underlying physics. Previous studies [47,125] show intriguing results by using saliency maps [126,127] (see also a recent study [128] using a "sum-of-parts" approach to interpretability). Architectures designed to relate the neural network results to physical qualities like the N-point correlation function are also promising [129,130]. It will be interesting to see how the results change with the presence of random permutation layers.

ACKNOWLEDGMENTS

We thank Shubh Agrawal, Pratik Chaudhari, Cyrille Doux, Rafael Gomes, Mike Jarvis, Amrut Nadgir, Shivam Pandey, Helen Qu, Dimitrios Tanoglidis, and Eric Wong for useful discussions. We are especially grateful to Tomek Kacprzak and his collaborators for generating and making available the DARKGRID simulations used in this work. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231. B. J. and M. G. are supported in part by the U.S. Department of Energy Grant No. DE-SC0007901.

APPENDIX A: TEST POOLING CHOICES

The experiments involving various pooling options are detailed in Table II. The outcomes vary between simulation cases 1 and 2 and with the level of shape noise but not in a straightforward way. This quantitative comparison is restricted to S_8. We selected mixed pooling as the baseline since the all-average approach is consistently ineffective for training on Ω_m. Although this investigation is far from comprehensive due to computational resource limitations, it shows that different choices in pooling can impact the outcomes.

TABLE II. Summary of comparing pooling choices in different simulation cases when applied to S_8. Mixed pooling stands for the 4 max +2avg as shown in Fig. 2. Mixed pooling-v2 denotes 2 max +4avg. Note that for simulation case 2 the all avg is slightly better than mixed pooling, but the all avg does not work at all for inferring Ω_m. Hence, we use mixed pooling (4 max +2avg) as the baseline choice for this work. The bolded font indicates the best performance.

Models	RMSE (%)	R^2
Simulation case 1		
All max pooling	2.174	0.975
All avg pooling	1.308	0.991
Mixed pooling	**1.187**	0.992
Simulation case 2		
All max pooling	4.764	0.890
All avg pooling	**3.61**	0.937
Mixed pooling	3.703	0.934
Simulation case 2 with $2n_{\rm eff}$		
All max pooling	4.5820	0.898
All avg pooling	**2.520**	0.969
Mixed pooling	3.079	0.954
Mixed pooling-v2	2.699	0.964
Simulation case 2 with $4n_{\rm eff}$		
All max pooling	3.526	0.939
All avg pooling	2.042	0.980
Mixed pooling	2.380	0.972
Mixed pooling-v2	**1.683**	0.986

APPENDIX B: TESTING ALTERNATIVES TO RANDOM PERMUTATION LAYER

Since introducing random permutation layers improves performance, we consider other ways to impose permutation invariance. A large average pooling over the entire feature map, for example, is permutation invariant. One can also add other permutation-invariant summary statistics such as the variance, maximum, or minimum. This is more common in graph neural networks, where permutation invariance is crucial due to the data structure. For example, in [131], the input of the MLP layers is the collection of sum, mean, max, and min.

Here, we test the following possibilities:

(i) *Test Model 1-3. Large Avg* These test models replace the random permutation layer and the subsequent convolution layer with a large average pooling. Test model 1-3 corresponds to models that have five, four, and three convolution layers. Note that the following MLP also changes in size because the channel number changes.

(ii) *Test Model 4.* Mean + Variance This model employs a large average pooling layer and additionally calculates the variance of the 8×8 pixel blocks for

TABLE III. Different models that we test as alternatives to random permutation layers, in comparison with the best model with shuffle (option 2). The description for each model is listed in Appendix B. None of these models performs better than even the six-layer CNN model without shuffling. This test shows that the effect of random permutation layers cannot be replaced by such permutation invariant statistics.

Models	RMSE (%)	R^2
Best model with shuffle	0.865	0.996
Test model 1	1.235	0.992
Test model 2	1.126	0.993
Test model 3	1.147	0.993
Test model 4	1.263	0.991
Test model 5	1.419	0.989
Test model 6	2.316	0.971
Test model 7	1.84	0.982

each channel. These are then concatenated as the input for the MLP.

(iii) *Test Model 5.* Mean + Sum + Max + Min This model is an extension of Test Model 4, using the mean (avg), sum, maximum, and minimum of the current channel as the inputs of the MLP.

(iv) *Test Model 6. Large Flat Layer* In this model, the final average pooling is omitted. Instead, a flattening operation is used to compile all the information. The input size of the MLP is increased to $128 \times 8 \times 8 = 8192$. We also tested other channel numbers with flattening in the end and found similar results.

(v) *Test Model 7. Patchify into smaller CNNs* For this model, we patchify the original image to 8×8 smaller images of size 64×64. We forward these smaller patches to the same weight-sharing convolutional layers and then take the average as the input of the MLP layers.

Note that all of the models above do not involve random permutation. Models 1 to 5 are strictly permutation invariant at corresponding layers. The test errors are summarized in Table III, and they do not help the six-layer CNN model as the random permutation layers. This is a model-specific result. Following the idea of random shuffling at large scales, we anticipate some of the choices above could increase accuracy in certain simulation settings. However, the random permutation layer remains a possibility when designing neural networks for cosmological fields.

For Model 7, we patchify the original image into smaller patches, and we tried various combinations of patch sizes, convolutional layers, and MLP sizes. This choice shares the same idea of making the corresponding scales permutation invariant. However, the modes between each patch are not captured by the convolution operations, and thus much less information is available. The RMSE is 1.84, significantly larger than the four-layer shuffled model or the six-layer nonshuffle model. Note that this experiment is conducted with simulation case 1, and we expect this method to work worse for simulation case 2 in the presence of masks.

APPENDIX C: COMPARISON WITH STANDARD REGULARIZATION

1. DROPOUT

DROPOUT is a widely recognized regularization technique for mitigating overfitting in machine learning. Essentially, DROPOUT combats coadaptation by randomly deactivating a portion of the neurons. Despite its effectiveness, DROPOUT is not universally applicable, especially in more recent and complex computer vision models like ResNet [95].

We show in Table IV three different use of DROPOUT:

(i) DROPOUT-v1 One DROPOUT layer before the MLP of the six-layer CNN model. This test is to see if masking some of the embedding representations can prevent overfitting of MLP.

(ii) DROPOUT-v2 Three DROPOUT layers after each linear + activation function in MLP, namely fc 7-9 in Fig. 2. This is the more traditional use of DROPOUT, where all the representations from the convolution layer are kept, but the use of DROPOUT directly prevents the overfitting of the MLP regression.

(iii) DROPOUT-v3 A 2D DROPOUT layer before convolution 6, where the random permutation layer is applied. This test is to make a direct comparison to the permutation layer, but instead of shuffling the positions, some patches are masked out during training.

For all three tests, we use a DROPOUT rate of 0.2. We found no improvement over the six-layer CNN model.

2. BATCHNORM

Batch Normalization (BATCHNORM) [132] is another very powerful regularization scheme that is commonly used in machine learning and has been adopted in previous weak lensing analyses such as [43,49,50], but not in [42]. BATCHNORM essentially whitens the output activation function and thus alleviates the internal covariance shift problem,[3] which is common in most of the neural network design.

However, BATCHNORM is not a once and for all solution. As a simple example, putting BATCHNORM before or after the activation function is nontrivial, with either choice differently affecting the goal of whitening.[4] The issue is

[3]See https://joelouismarino.github.io/posts/2017/08/statistical_whitening/ for a detailed explanation on statistical whitening and data normalization.

[4]See http://torch.ch/blog/2016/02/04/resnets.html for a discussion on different effects of BATCHNORM before or after the activation.

TABLE IV. Test models with two other regularization methods DROPOUT and BATCHNORM when applied to simulation case 1 and S_8. We also tested applying BATCHNORM before and after each activation function. See Appendix C for the details of where these regularization layers are used. As in Table III, none of these methods do as well as the shuffle CNN. BN stands for BatchNorm.

Models	RMSE (%)	R^2
Best model with shuffle	0.865	0.996
DROPOUT		
All avg w/ DROPOUT-v1	1.214	0.992
Mixed avg/max w/ DROPOUT-v1	1.115	0.993
Mixed avg/max w/ DROPOUT-v2	1.563	0.987
Mixed avg/max w/ DROPOUT-v3	1.165	0.993
BATCHNORM before activation		
All avg w/ BN	1.664	0.985
Mixed avg/max w/ BN	2.780	0.959
BATCHNORM after activation		
All avg w/ BN	2.169	0.975
Mixed avg/max w/ BN	1.890	0.981

TABLE V. The architecture used in [49], except for the difference in the input and output size. The RMSE for simulation case 2 is 3.776%, which is similar to our nonshuffle CNN model as shown in Fig. 5.

Layer	Kernel size	Stride	Output dimensions
(Input)			$(4 \times 512 \times 512)$
Convolution	5×5	2	$(n_{\rm ch}/2) \times 254 \times 254$
Convolution	5×5	2	$n_{\rm ch} \times 125 \times 125$
Residual block			
⋮	⋮	⋮	⋮
Residual block			
Pooling	2×2	2	$n_{\rm ch} \times 62 \times 62$
Convolution	3×3	1	$(2n_{\rm ch}) \times 60 \times 60$
Pooling	2×2	2	$(2n_{\rm ch}) \times 30 \times 30$
Convolution	3×3	1	$(4n_{\rm ch}) \times 28 \times 28$
Pooling	2×2	2	$(4n_{\rm ch}) \times 14 \times 14$
Convolution	3×3	1	$(8n_{\rm ch}) \times 12 \times 12$
Pooling	2×2	2	$(8n_{\rm ch}) \times 6 \times 6$
Convolution	3×3	1	$(16n_{\rm ch}) \times 4 \times 4$
Pooling	4×4	⋯	$(16n_{\rm ch}) \times 1 \times 1$
Linear	⋯	⋯	256
ReLU	⋯	⋯	256
Linear	⋯	⋯	1

further complicated with regularization schemes such as DROPOUT [133]. In fact, a large number of architectures find that when combining the two most powerful regularization schemes DROPOUT and BATCHNORM, the accuracy often decreases.

In this work, we did not adopt any BATCHNORM in our architecture, and we show in Table IV that it negatively affects the results. Note that we use the default setting that enables affine parameters, which are the only two learnable parameters. However, this complicates the training in the minibatch setting because they have a large effect on the whitened output.

Again, this choice is only specific to optimizing our problem, which can be different if the data and other assumptions change. For instance, if one uses data augmentation with less Gaussian noise injection level than the 10% used in this work, or studies the case with all cosmological parameters varying, the internal large variance of the input data can make BATCHNORM necessary. We do, nevertheless, expect the permutation layer to improve the generalization performance in the presence of BATCHNORM.

APPENDIX D: COMPARISON WITH RESNET USED IN PREVIOUS WORK

We also test an architecture with a residual connection. The architecture is summarized in Table V, which is the same as in [49,50] except the input channel is 4 for our four tomography bin and the output number is 1 since we are training one parameter per network. The residual block consists of two convolution layers and a skip connection, similarly in [95]. We use ten residual blocks and $n_{\rm ch} = 10$ as suggested in [49]. We find no difference between the ResNet and our six-layer CNN model without shuffle. For simplicity, we did not further optimize this design or test with random permutation layers with this structure. This test suggests that the residual connection might not be necessary for our simulations. We stress again that the comparison is not exact because our simulations are very different. Even though the input sizes are both 512×512, the resolution is very different, with those in [95] being 0.87 arcmin. However, this result suggests that the proposed random permutation layer is optimizing the neural network for cosmological fields in a different way.

APPENDIX E: ADDITIONAL TESTS WITH DIFFERENT NOISE LEVEL AND SIZE OF TRAINING SET

We provide some additional tests varying the noise level and training data size. As summarized in Table VI, the shuffling operation consistently boosts the generalization accuracy of the model. Note that the six-layer CNN should be compared with CNN with shuffle option 1 as they have the same structure and trainable parameters, but only differ by one random permutation layer. In these tests, the other option is not always better than option 1, but CNN with shuffle option 1 is always better than the six-layer CNN model.

TABLE VI. Additional tests varying noise level and size of the training set. The reported RMSE is for S_8. The shuffling operation consistently improves the accuracy when comparing nonshuffle six-layer CNN and shuffle option 1, which only differs by one random permutation layer. Note that the best position of the shuffling operation changes from the fiducial test of the paper.

Models	RMSE (%)	R^2
Simulation case 1 with $2n_{\rm eff}$		
Nonshuffle six-layer CNN	0.974	0.996
CNN with shuffle option 1	0.740	0.998
CNN with shuffle option 2	**0.684**	0.998
Simulation case 1 with $0.5n_{\rm eff}$		
Nonshuffle six-layer CNN	1.537	0.988
CNN with shuffle option 1	**1.275**	0.991
CNN with shuffle option 2	1.833	0.982
Simulation case 1 with no shape noise		
Nonshuffle six-layer CNN	0.561	0.998
CNN with shuffle option 1	**0.424**	0.999
CNN with shuffle option 2	0.565	0.998
Simulation case 2 with half training data		
Nonshuffle six-layer CNN	4.097	0.912
CNN with shuffle option 1	3.594	0.932
CNN with shuffle option 2	3.478	0.936
CNN with shuffle option 3	**2.921**	0.955
Simulation case 2 with $2n_{\rm eff}$		
Nonshuffle six-layer CNN (mixed pooling)	3.079	0.954
Nonshuffle six-layer CNN (avg pooling)	2.520	0.969
CNN with shuffle option 1	2.520	0.969
CNN with shuffle option 2	2.179	0.977
CNN with shuffle option 3	**2.013**	0.980
Simulation case 2 with $4n_{\rm eff}$		
Nonshuffle six-layer CNN (mixed pooling-v2)	1.683	0.9886
Nonshuffle six-layer CNN (avg pooling)	2.042	0.980
CNN with shuffle option 1	1.994	0.981
CNN with shuffle option 2	1.748	0.985
CNN with shuffle option 3	**1.410**	0.990

TABLE VII. The architecture we used for performing the data compression for 1D data vectors used in this work (power spectrum and PDF). We varied the number of layers and hidden layer size and observed negligible changes in the compression accuracy. We also tried other architectures such as MLP with residual connections and 1D-CNN. We found no difference in the results.

Layer	Output dimensions
Dense + ReLu	256
Dense + ReLu	128
Dense + ReLu	64
Dense + ReLu	32
Dense + ReLu	1

APPENDIX F: DETAILS FOR TRAINING 1D DATA VECTORS: POWER SPECTRUM AND PDF

Here we present the details of training the summary statistics used in this paper: power spectrum as shown in Fig. 6, and PDF for the comparison in Sec. IV A.

For the architecture, we use a rather simple network as shown in Table VII. We found the compression accuracy insensitive to the choice of architectures for the 1D data vector. We experimented with varying hidden layer sizes, the number of layers, and other architectures, including ResMLP and 1D-CNN. This suggests that the compression converges for the simulations we have.

When making the data vectors, we found binning the full-length data vector to a smaller number helps training, especially for $\Omega_{\rm m}$ in the case of the power spectrum. We use a logarithm binning of the number 16 for each tomographic bin, similar to Ref. [62].

When using PDFs as the summary statistics, we smoothed the maps to get information across scales. We applied a Gaussian filter with a standard deviation from 1 to 8 and concatenated the PDFs to get the final data vector. We tested various bin sizes and found no difference in the final result. They all gave around 9% RMSE, significantly larger than the 2.6% shown in Fig. 9. This suggests that binning the PDFs may not be the most efficient way to extract nonspatial information.

[1] T. Abbott, F. B. Abdalla, J. Aleksić, S. Allam, A. Amara, D. Bacon, E. Balbinot, M. Banerji, K. Bechtol *et al.* (Dark Energy Survey Collaboration), Mon. Not. R. Astron. Soc. **460**, 1270 (2016).

[2] M. A. Troxel, N. MacCrann, J. Zuntz, T. F. Eifler, E. Krause, S. Dodelson, D. Gruen, J. Blazek, O. Friedrich, S. Samuroff *et al.*, Phys. Rev. D **98**, 043528 (2018).

[3] A. Amon, D. Gruen, M. A. Troxel, N. MacCrann, S. Dodelson, A. Choi, C. Doux, L. F. Secco, S. Samuroff, E. Krause *et al.*, Phys. Rev. D **105**, 023514 (2022).

[4] K. Kuijken, C. Heymans, H. Hildebrandt, R. Nakajima, T. Erben, J. T. A. de Jong, M. Viola, A. Choi, H. Hoekstra, L. Miller *et al.*, Mon. Not. R. Astron. Soc. **454**, 3500 (2015).

[5] C. Heymans, T. Tröster, M. Asgari, C. Blake, H. Hildebrandt, B. Joachimi, K. Kuijken, C.-A. Lin, A. G. Sánchez, J. L. van den Busch *et al.*, Astron. Astrophys. **646**, A140 (2021).

[6] M. Asgari, C.-A. Lin, B. Joachimi, B. Giblin, C. Heymans, H. Hildebrandt, A. Kannawadi, B. Stölzner, T. Tröster, J. L. van den Busch *et al.*, Astron. Astrophys. **645**, A104 (2021).

[7] R. Dalal, X. Li, A. Nicola, J. Zuntz, M. A. Strauss, S. Sugiyama, T. Zhang, M. M. Rau, R. Mandelbaum, M. Takada *et al.*, Phys. Rev. D **108**, 123519 (2023).

[8] X. Li, T. Zhang, S. Sugiyama, R. Dalal, M. M. Rau, R. Mandelbaum, M. Takada, S. More, M. A. Strauss, H. Miyatake *et al.*, Phys. Rev. D **108**, 123518 (2023).

[9] S. Sugiyama, H. Miyatake, S. More, X. Li, M. Shirasaki, M. Takada, Y. Kobayashi, R. Takahashi, T. Nishimichi, A. J. Nishizawa *et al.*, Phys. Rev. D **108**, 123521 (2023).

[10] S. A. Smee, J. E. Gunn, A. Uomoto, N. Roe, D. Schlegel, C. M. Rockosi, M. A. Carr, F. Leger, K. S. Dawson, M. D. Olmstead *et al.*, Astron. J. **146**, 32 (2013).

[11] C. Zhao, A. Variu, M. He, D. Forero-Sánchez, A. Tamone, C.-H. Chuang, F.-S. Kitaura, C. Tao, J. Yu, J.-P. Kneib *et al.*, Mon. Not. R. Astron. Soc. **511**, 5492 (2022).

[12] S. Alam, M. Aubert, S. Avila, C. Balland, J. E. Bautista, M. A. Bershady, D. Bizyaev, M. R. Blanton, A. S. Bolton, J. Bovy *et al.*, Phys. Rev. D **103**, 083533 (2021).

[13] Ž. Ivezić, S. M. Kahn, J. A. Tyson, B. Abel, E. Acosta, R. Allsman, D. Alonso, Y. AlSayyad, S. F. Anderson, J. Andrew *et al.*, Astrophys. J. **873**, 111 (2019).

[14] O. Dore, C. Hirata, Y. Wang, D. Weinberg, T. Eifler, R. J. Foley, C. H. Heinrich, E. Krause, S. Perlmutter, A. Pisani *et al.*, Bull. Am. Astron. Soc. **51**, 341 (2019).

[15] R. Laureijs, J. Amiaux, S. Arduini, J. L. Augueres, J. Brinchmann, R. Cole, M. Cropper, C. Dabin, L. Duvet, A. Ealet *et al.*, arXiv:1110.3193.

[16] M. Levi, L. E. Allen, A. Raichoor, C. Baltay, S. BenZvi, F. Beutler, A. Bolton, F. J. Castander, C.-H. Chuang, A. Cooper *et al.*, Bull. Am. Astron. Soc. **51**, 57 (2019).

[17] M. Takada and B. Jain, Mon. Not. R. Astron. Soc. **344**, 857 (2003).

[18] A. Halder, Z. Gong, A. Barreira, O. Friedrich, S. Seitz, and D. Gruen, J. Cosmol. Astropart. Phys. 10 (2023) 028.

[19] M. Takada and B. Jain, Mon. Not. R. Astron. Soc. **348**, 897 (2004).

[20] B. Jain and L. Van Waerbeke, Astrophys. J. Lett. **530**, L1 (2000).

[21] L. Marian, R. E. Smith, and G. M. Bernstein, Astrophys. J. Lett. **698**, L33 (2009).

[22] N. Martinet, P. Schneider, H. Hildebrandt, H. Shan, M. Asgari, J. P. Dietrich, J. Harnois-Déraps, T. Erben, A. Grado, C. Heymans *et al.*, Mon. Not. R. Astron. Soc. **474**, 712 (2018).

[23] J. Harnois-Déraps, N. Martinet, T. Castro, K. Dolag, B. Giblin, C. Heymans, H. Hildebrandt, and Q. Xia, Mon. Not. R. Astron. Soc. **506**, 1623 (2021).

[24] D. Zürcher, J. Fluri, R. Sgier, T. Kacprzak, M. Gatti, C. Doux, L. Whiteway, A. Réfrégier, C. Chang, N. Jeffrey *et al.*, Mon. Not. R. Astron. Soc. **511**, 2075 (2022).

[25] G. A. Marques, J. Liu, M. Shirasaki, L. Thiele, D. Grandón, K. M. Huffenberger, S. Cheng, J. Harnois-Déraps, K. Osato, and W. R. Coulton, Mon. Not. R. Astron. Soc. **528**, 4513 (2024).

[26] D. Munshi, L. van Waerbeke, J. Smidt, and P. Coles, Mon. Not. R. Astron. Soc. **419**, 536 (2012).

[27] J. M. Kratochvil, E. A. Lim, S. Wang, Z. Haiman, M. May, and K. Huffenberger, Phys. Rev. D **85**, 103513 (2012).

[28] N. Grewal, J. Zuntz, T. Tröster, and A. Amon, Open J. Astrophys. **5**, 13 (2022).

[29] J. Feldbrugge, M. van Engelen, R. van de Weygaert, P. Pranav, and G. Vegter, J. Cosmol. Astropart. Phys. 09 (2019) 052.

[30] C. Parroni, É. Tollet, V. F. Cardone, R. Maoli, and R. Scaramella, Astron. Astrophys. **645**, A123 (2021).

[31] S. Heydenreich, B. Brück, and J. Harnois-Déraps, Astron. Astrophys. **648**, A74 (2021).

[32] S. Heydenreich, B. Brück, P. Burger, J. Harnois-Déraps, S. Unruh, T. Castro, K. Dolag, and N. Martinet, Astron. Astrophys. **667**, A125 (2022).

[33] A. Banerjee and T. Abel, Mon. Not. R. Astron. Soc. **519**, 4856 (2023).

[34] D. Anbajagane, C. Chang, A. Banerjee, T. Abel, M. Gatti, V. Ajani, A. Alarcon, A. Amon, E. J. Baxter, K. Bechtol *et al.*, Mon. Not. R. Astron. Soc. **526**, 5530 (2023).

[35] S. Cheng, Y.-S. Ting, B. Ménard, and J. Bruna, Mon. Not. R. Astron. Soc. **499**, 5902 (2020).

[36] S. Cheng and B. Ménard, arXiv:2112.01288.

[37] G. Valogiannis and C. Dvorkin, Phys. Rev. D **106**, 103509 (2022).

[38] I. Hothi, E. Allys, B. Semelin, and F. Boulanger, Astron. Astrophys. **686**, A212 (2024).

[39] C. Pedersen, S. Ho, and M. Eickenberg, in Machine Learning for Astrophysics (2022), p. 40, arXiv:2307.14362.

[40] M. Gatti, N. Jeffrey, L. Whiteway, J. Williamson, B. Jain, V. Ajani, D. Anbajagane, G. Giannini, C. Zhou, A. Porredon *et al.*, arXiv:2310.17557.

[41] V. Ajani, M. Baldi, A. Barthelemy, A. Boyle, P. Burger, V. F. Cardone, S. Cheng, S. Codis, C. Giocoli *et al.* (Euclid Collaboration), Astron. Astrophys. **675**, A120 (2023).

[42] J. Fluri, T. Kacprzak, A. Refregier, A. Amara, A. Lucchi, and T. Hofmann, Phys. Rev. D **98**, 123518 (2018).

[43] D. Ribli, B. Á. Pataki, J. M. Zorrilla Matilla, D. Hsu, Z. Haiman, and I. Csabai, Mon. Not. R. Astron. Soc. **490**, 1843 (2019).

[44] A. Gupta, J. M. Zorrilla Matilla, D. Hsu, and Z. Haiman, Phys. Rev. D **97**, 103515 (2018).

[45] J. Fluri, T. Kacprzak, A. Lucchi, A. Refregier, A. Amara, T. Hofmann, and A. Schneider, Phys. Rev. D **100**, 063514 (2019).
[46] D. Ribli, B. Á. Pataki, and I. Csabai, Nat. Astron. **3**, 93 (2019).
[47] J. M. Z. Matilla, M. Sharma, D. Hsu, and Z. Haiman, Phys. Rev. D **102**, 123506 (2020).
[48] N. Jeffrey, J. Alsing, and F. Lanusse, Mon. Not. R. Astron. Soc. **501**, 954 (2021).
[49] T. Lu, Z. Haiman, and J. M. Zorrilla Matilla, Mon. Not. R. Astron. Soc. **511**, 1518 (2022).
[50] T. Lu, Z. Haiman, and X. Li, Mon. Not. R. Astron. Soc. **521**, 2050 (2023).
[51] H. J. Hortúa, R. Volpi, D. Marinelli, and L. Malagò, Phys. Rev. D **102**, 103509 (2020).
[52] A. Akhmetzhanova, S. Mishra-Sharma, and C. Dvorkin, Mon. Not. R. Astron. Soc. **527**, 7459 (2024).
[53] N. Perraudin, M. Defferrard, T. Kacprzak, and R. Sgier, Astron. Comput. **27**, 130 (2019).
[54] J. Fluri, T. Kacprzak, A. Lucchi, A. Schneider, A. Refregier, and T. Hofmann, Phys. Rev. D **105**, 083518 (2022).
[55] T. L. Makinen, T. Charnock, P. Lemos, N. Porqueres, A. F. Heavens, and B. D. Wandelt, Open J. Astrophys. **5**, 18 (2022).
[56] S. Mishra-Sharma, Mach. Learn. **3**, 01LT03 (2022).
[57] B. Dai and U. Seljak, Mon. Not. R. Astron. Soc. **516**, 2363 (2022).
[58] M. Ullmo, A. Decelle, and N. Aghanim, Astron. Astrophys. **651**, A46 (2021).
[59] T. W. H. Yiu, J. Fluri, and T. Kacprzak, J. Cosmol. Astropart. Phys. 12 (2022) 013.
[60] F. Villaescusa-Navarro *et al.* (CAMELS Collaboration), Astrophys. J. **915**, 71 (2021).
[61] B. Dai and U. Seljak, Proc. Natl. Acad. Sci. U.S. A. **121**, e2309624121 (2024).
[62] N. Jeffrey *et al.* (DES Collaboration), arXiv:2403.02314.
[63] D. Zürcher, J. Fluri, R. Sgier, T. Kacprzak, and A. Refregier, J. Cosmol. Astropart. Phys. 01 (2021) 028.
[64] D. Zürcher, J. Fluri, R. Sgier, T. Kacprzak, M. Gatti, C. Doux, L. Whiteway, A. Réfrégier, C. Chang, N. Jeffrey *et al.*, Mon. Not. R. Astron. Soc. **511**, 2075 (2022).
[65] D. Potter, J. Stadel, and R. Teyssier, Comput. Astrophys. Cosmol. **4**, 2 (2017).
[66] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann, Astrophys. J. **622**, 759 (2005).
[67] M. Gatti, N. Jeffrey, L. Whiteway, V. Ajani, T. Kacprzak, D. Zürcher, C. Chang, B. Jain, J. Blazek, E. Krause *et al.*, Mon. Not. R. Astron. Soc. **527**, L115 (2024).
[68] M. Gatti, N. Jeffrey, L. Whiteway, J. Williamson, B. Jain, V. Ajani, D. Anbajagane, G. Giannini, C. Zhou, A. Porredon *et al.*, Phys. Rev. D **109**, 063534 (2024).
[69] P. Fosalba, E. Gaztañaga, F. J. Castander, and M. Crocce, Mon. Not. R. Astron. Soc. **447**, 1319 (2015).
[70] N. Jeffrey, M. Gatti, C. Chang, L. Whiteway, U. Demirbozan, A. Kovacs, G. Pollina, D. Bacon, N. Hamaus, T. Kacprzak *et al.*, Mon. Not. R. Astron. Soc. **505**, 4626 (2021).
[71] N. Kaiser and G. Squires, Astrophys. J. **404**, 441 (1993).
[72] J. Myles, A. Alarcon, A. Amon, C. Sánchez, S. Everett, J. DeRose, J. McCullough, D. Gruen, G. M. Bernstein, M. A. Troxel *et al.*, Mon. Not. R. Astron. Soc. **505**, 4249 (2021).
[73] N. Jeffrey, F. Lanusse, O. Lahav, and J.-L. Starck, Mon. Not. R. Astron. Soc. **492**, 5023 (2020).
[74] C. Shorten and T. M. Khoshgoftaar, J. Big Data **6**, 60 (2019).
[75] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, arXiv:1912.01703.
[76] I. Loshchilov and F. Hutter, arXiv:1711.05101.
[77] G. Andrew and J. Gao, in International Conference on Machine Learning (2007), 10.1145/1273496.1273501.
[78] K. Levenberg, Q. Appl. Math. **2**, 164 (1944).
[79] T. Charnock, G. Lavaux, and B. D. Wandelt, Phys. Rev. D **97**, 083004 (2018).
[80] G. Papamakarios, T. Pavlakou, and I. Murray, arXiv:1705.07057.
[81] M. Germain, K. Gregor, I. Murray, and H. Larochelle, *Proceedings of the 32nd International Conference on Machine Learning* (2015), arXiv:1502.03509.
[82] S. Kullback and R. A. Leibler, Ann. Math. Stat. **22**, 79 (1951).
[83] K. Cranmer, J. Brehmer, and G. Louppe, Proc. Natl. Acad. Sci. U.S.A. **117**, 30055 (2020).
[84] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, Publ. Astron. Soc. Pac. **125**, 306 (2013).
[85] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, arXiv:1706.04599.
[86] J. Hermans, A. Delaunoy, F. Rozet, A. Wehenkel, V. Begy, and G. Louppe, arXiv:2110.06581.
[87] P. Lemos, A. Coogan, Y. Hezaveh, and L. Perreault-Levasseur, *Proceedings of the 40th International Conference on Machine Learning* (2023), 202, 19256, arXiv:2302.03026.
[88] J.-M. Lueckmann, J. Boelts, D. S. Greenberg, P. J. Gonçalves, and J. H. Macke, arXiv:2101.04653.
[89] C. Hahn *et al.*, arXiv:2310.15246.
[90] P. Lemos *et al.*, Phys. Rev. D **109**, 083536 (2024).
[91] P. Lemos, M. Cranmer, M. Abidi, C. Hahn, M. Eickenberg, E. Massara, D. Yallup, and S. Ho, Mach. Learn. **4**, 01LT01 (2023).
[92] J. Linhart, A. Gramfort, and P. L. C. Rodrigues, arXiv:2211.09602.
[93] H. Jia, arXiv:2401.02413.
[94] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, in *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255, 10.1109/CVPR.2009.5206848.
[95] K. He, X. Zhang, S. Ren, and J. Sun, arXiv:1512.03385.
[96] K. Simonyan and A. Zisserman, arXiv:1409.1556.
[97] C. R. Njieutcheu Tassi, in Pattern Recognition and Artificial Intelligence, edited by C. Djeddi, A. Jamil, and I. Siddiqi (Springer International Publishing, Cham, 2020), pp. 118–132.
[98] C. Doux, B. Jain, D. Zeurcher, J. Lee, X. Fang, R. Rosenfeld, A. Amon, H. Camacho, A. Choi, L. F. Secco *et al.*, Mon. Not. R. Astron. Soc. **515**, 1942 (2022).

[99] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, arXiv:1207.0580.
[100] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, J. Mach. Learn. Res. **15**, 1929 (2014).
[101] S. Mallat, Commun. Pure Appl. Math. **65**, 1331 (2012).
[102] R. Zhang, arXiv:1904.11486.
[103] T. S. Cohen and M. Welling, arXiv:1602.07576.
[104] M. Weiler and G. Cesa, arXiv:1911.08251.
[105] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, arXiv:1901.00596.
[106] R. L. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, arXiv:1811.01900.
[107] X. Zhang, X. Zhou, M. Lin, and J. Sun, arXiv:1707.01083.
[108] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, arXiv:1605.00055.
[109] S. Y. Hwang, C. G. Sabiu, I. Park, and S. E. Hong, J. Cosmol. Astropart. Phys. 11 (2023) 075.
[110] F. Villaescusa-Navarro, B. D. Wandelt, D. Anglés-Alcázar, S. Genel, J. M. Zorrilla Mantilla, S. Ho, and D. N. Spergel, Astrophys. J. **928**, 44 (2022).
[111] F. Villaescusa-Navarro *et al.*, arXiv:2109.09747.
[112] S. Ravanbakhsh, J. Oliva, S. Fromenteau, L. C. Price, S. Ho, J. Schneider, and B. Poczos, arXiv:1711.02033.
[113] T. Flöss and P. D. Meerburg, J. Cosmol. Astropart. Phys. 02 (2024) 031.
[114] S. Gagnon-Hartman, Y. Cui, A. Liu, and S. Ravanbakhsh, Mon. Not. R. Astron. Soc. **504**, 4716 (2021).
[115] M. Bianco, S. K. Giri, I. T. Iliev, and G. Mellema, Mon. Not. R. Astron. Soc. **505**, 3982 (2021).
[116] D. Prelogović, A. Mesinger, S. Murray, G. Fiameni, and N. Gillet, Mon. Not. R. Astron. Soc. **509**, 3852 (2022).
[117] C. P. Novaes, E. J. de Mericia, F. B. Abdalla, C. A. Wuensche, L. Santos, J. Delabrouille, M. Remazeilles, and V. Liccardo, Mon. Not. R. Astron. Soc. **528**, 2078 (2023).
[118] M. Bianco, S. K. Giri, D. Prelogović, T. Chen, F. G. Mertens, E. Tolley, A. Mesinger, and J.-P. Kneib, Mon. Not. R. Astron. Soc. **528**, 5212 (2024).
[119] J. Kennedy, J. C. Carr, S. Gagnon-Hartman, A. Liu, J. Mirocha, and Y. Cui, Mon. Not. R. Astron. Soc. **529**, 3684 (2024).
[120] J. M. Casas, L. Bonavera, J. González-Nuevo, C. Baccigalupi, M. M. Cueli, D. Crespo, E. Goitia, J. D. Santos, M. L. Sánchez, and F. J. de Cos, Astron. Astrophys. **666**, A89 (2022).
[121] M. Münchmeyer and K. M. Smith, arXiv:1905.05846.
[122] B. Swiderski, S. Osowski, G. Gwardys, J. Kurek, M. Slowinska, and I. Lugowska, EURASIP J. Image Video Process. **2022**, 3 (2022).
[123] G. Zhao, J. Wang, and Z. Zhang, in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17* (2017), pp. 3476–3482, 10.24963/ijcai.2017/486.
[124] S. Xie, A. Kirillov, R. Girshick, and K. He, arXiv:1904.01569.
[125] M. Ntampaka and A. Vikhlinin, Astrophys. J. **926**, 45 (2022).
[126] K. Simonyan, A. Vedaldi, and A. Zisserman, arXiv:1312.6034.
[127] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, arXiv:1412.6806.
[128] W. You, H. Qu, M. Gatti, B. Jain, and E. Wong, arXiv:2310.16316.
[129] C. Miles, A. Bohrdt, R. Wu, C. Chiu, M. Xu, G. Ji, M. Greiner, K. Q. Weinberger, E. Demler, and E.-A. Kim, Nat. Commun. **12**, 3905 (2021).
[130] Z. Gong, A. Halder, A. Bohrdt, S. Seitz, and D. Gebauer, arXiv:2402.09526.
[131] B. Y. Wang, A. Pisani, F. Villaescusa-Navarro, and B. D. Wandelt, Astrophys. J. **955**, 131 (2023).
[132] S. Ioffe and C. Szegedy, arXiv:1502.03167.
[133] X. Li, S. Chen, X. Hu, and J. Yang, arXiv:1801.05134.