Accelerating statistical inferences in astrophysics with Neural Networks and Hamiltonian Monte Carlo

Diego Gonzalez-Hernandez [1] **Molly Wolfson** [1] **Joseph F. Hennawi** [1 2]

Abstract

We present an approach to accelerate statistical inferences in astrophysics by using a combination of neural networks and Hamiltonian Monte Carlo. The neural networks are used to create high-fidelity surrogates of computationally expensive models, while Hamiltonian Monte Carlo accelerates the inferences by more efficiently exploring the parameter space. We demonstrate the potential of this approach by applying it to a realistic model for the Epoch of Reionization.

1. Introduction

One of the main objectives of the natural sciences is to formulate models $m(\boldsymbol{\theta})$ that explain physical phenomena, and to test their validity by comparing these models with observational data. In astrophysics, this is typically done by obtaining observational data $\boldsymbol{d}$, and then constraining the model parameters $\boldsymbol{\theta}$ with Bayes theorem:

$$P(\boldsymbol{\theta}|\boldsymbol{d}) = \frac{\mathcal{L}(\boldsymbol{d}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\boldsymbol{d})} \tag{1}$$

With this framework, the posterior distribution $P(\boldsymbol{\theta}|\boldsymbol{d})$ is calculated given that we assume both a likelihood function $\mathcal{L}(\boldsymbol{d}|\boldsymbol{\theta})$, and a prior distribution $P(\boldsymbol{\theta})$, and that we can calculate the evidence of the model $P(\boldsymbol{d})$. In most applications, however, $P(\boldsymbol{d})$ is intractable. Therefore, sampling algorithms designed to approximate the posterior distribution are usually employed. In general, Markov Chain Monte Carlo (MCMC) and Nested Sampling (NS) methods are used for parameter estimation problems in physics and other sciences. The Metropolis-Hastings algorithm (and its variations) is the most commonly used MCMC algorithm (Lewis & Bridle, 2002; Foreman-Mackey et al., 2013; Karamanis et al., 2022), while multi-modal and sliced-based NS algorithms are also widely employed (Feroz & Hobson, 2008; Feroz et al., 2009; Handley et al., 2015).

Unfortunately, applying standard MCMC and NS algorithms can become too computationally expensive to feasibly perform parameter inference. The most obvious problem is that these algorithms typically require multiple model $m(\boldsymbol{\theta})$ evaluations. Consequently, complex models that incur in large computational costs can significantly slow down parameter estimation. Furthermore, these algorithms can struggle to efficiently explore high-dimensional parameter spaces, reducing the number of parameters that we are feasibly allowed to vary. To overcome these challenges, we propose using a combination of neural emulators and Hamiltonian Monte Carlo. The structure of this article is follows. In Section 2 we explain the approach of using neural networks alongside Hamiltonian Monte Carlo to accelerate parameter inferences. We then show an application of this method in Section 3. We discuss the results of this application in Section 4 and finish with our conclusions in Section 5.

2. The NN + HMC method

2.1. Neural Networks

The main goal of an emulator is to provide a high-fidelity approximation of a computationally expensive model that is fast to compute. There are different approaches for creating emulators (Walther et al., 2019; Heitmann et al., 2014; Euclid Collaboration et al., 2019), but neural networks (NN) have become an increasingly popular choice due to their flexibility and ease of use (Piras & Spurio Mancini, 2023; Gong et al., 2023; Nygaard et al., 2023; Ruiz-Zapatero et al., 2024). The specifics of the neural networks (such as the choice of architecture and hyper-parameters) that each application requires may vary depending on the problem itself (see Section 3 for an example). However, we generally expect the neural networks to take the model parameters $\boldsymbol{\theta}$ as input, and output some model $m(\boldsymbol{\theta})$ dependent observable or summary statistic.

*Equal contribution [1]Department of Physics, University of California, Santa Barbara, CA 93106, USA [2]Leiden Observatory, Leiden University, Niels Bohrweg 2, 2333 CA Leiden, Netherlands. Correspondence to: Diego Gonzalez-Hernandez <dgonzalezhernandez@ucsb.edu>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria.

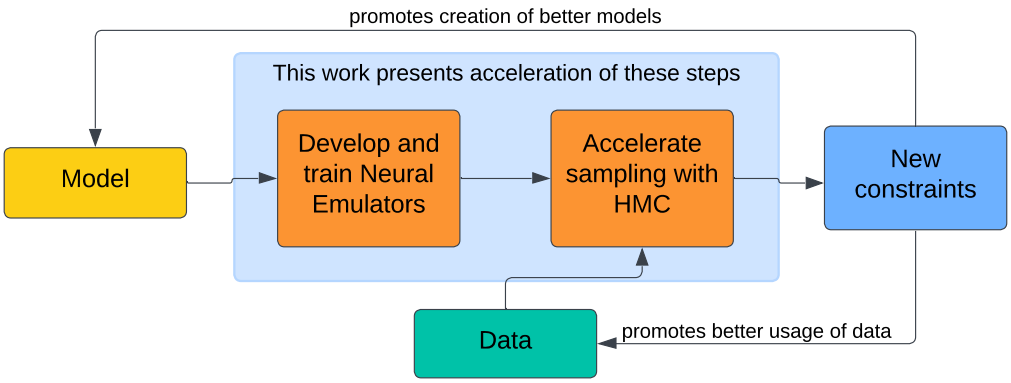

Figure 1. Flow chart that shows the usual workflow of typical inference problems found in astrophysics, emphasizing the challenges that are addressed by employing neural emulators and Hamiltonian Monte Carlo.

2.2. Hamiltonian Monte Carlo

A viable alternative to accelerate sampling is to use gradient-based algorithms. Among these, a popular choice is Hamiltonian Monte Carlo (HMC), a gradient-based modification of the Metropolis-Hastings algorithm. HMC improves the efficiency of the parameter space exploration by proposing samples that have a higher probability of being accepted (Duane et al., 1987). To do so, HMC formulates the problem by creating an analogy between the exploration of the parameter space and the dynamic evolution of a particle living in a phase space defined by the particle's position (chosen to be equal to $\boldsymbol{\theta}$) and momentum $\boldsymbol{p}$. Typically, this particle is assigned the following Hamiltonian $\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p})$:

$$\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^T \mathcal{M}^{-1} \boldsymbol{p} + \boldsymbol{U}(\boldsymbol{\theta}) \tag{2}$$

Where $\mathcal{M}$ is referred as the mass matrix, and $\boldsymbol{U}(\boldsymbol{\theta})$ is the particle's potential energy. The choice of $\boldsymbol{U}(\boldsymbol{\theta})$ can depend on the specific application, but the following is generally used:

$$\boldsymbol{U}(\boldsymbol{\theta}) = -ln\big(\mathcal{L}(\boldsymbol{d}|\boldsymbol{\theta})P(\boldsymbol{\theta})\big) \tag{3}$$

Intuitively, this choice for $\boldsymbol{U}(\boldsymbol{\theta})$ causes the regions of high density in the posterior $P(\boldsymbol{\theta}|\boldsymbol{d})$ to be located in the regions with low potential energy $\boldsymbol{U}(\boldsymbol{\theta})$ (see Equation 1). HMC then increases the efficiency of the standard Metropolis-Hastings algorithm by proposing samples that correspond to low energy levels of $\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{p})$, which have a higher chance of being accepted. Therefore, the momentum $\boldsymbol{p}$ is randomly sampled from a multivariate Gaussian distribution with zero mean and a covariance matrix given by $\mathcal{M}$ at the beginning of every step of the chain, forcing the hypothetical particle to spend more time exploring the parameter space in regions of low $\boldsymbol{U}(\boldsymbol{\theta})$. It is known that for a target distribution with D dimensions, the cost of an independent sample that HMC produces scales as $O(D^{5/4})$, a significant improvement over the scaling of $O(D^2)$ that the standard Metropolis-Hastings algorithm has (Neal, 2011).

Unfortunately, adopting HMC in most problems is unfeasible. This is because the numerical integration that is performed at every step of HMC requires $\mathcal{L}(\boldsymbol{d}|\boldsymbol{\theta})$ (and in turn, the model evaluations) to be fully differentiable within the parameter space that is being explored, which is typically not the case in most inference problems. However, replacing the models with neural emulators solves this problem, because we can evaluate their derivatives via auto-differentiation (Rumelhart et al., 1986; Piras & Spurio Mancini, 2023). This allows us to fully take advantage of HMC.

2.3. Pipeline

Figure 1 summarizes the main points of our approach. Since this approach is problem agnostic, creating a flexible pipeline is possible. To this end, we have started the development of a pipeline that facilitates the application of this approach. To ensure that the auto-differentiation evaluations of the networks are efficient, we opt to use `JAX` as the main backbone for this pipeline. `JAX` is a high-performance Python package that is mainly designed to work as a machine learning framework (Bradbury et al., 2018). Our pipeline is built to perform the following:

Training and hyper-parameter tuning: Given a dataset with pairs of $(\boldsymbol{\theta}, m(\boldsymbol{\theta}))$, the pipeline trains the required emulators and performs their hyper-parameter tuning. We use `Flax` to create the neural networks (Heek et al., 2023), and `Optuna` to perform the hyper-parameter tuning (Akiba et al., 2019).

Parameter inference: Given a set of observations $\boldsymbol{d}$, the pipeline performs the statistical inference for $\boldsymbol{\theta}$ using the trained emulators. For the HMC, we use the implementation of the No-U-Turns algorithm (Hoffman & Gelman, 2011) in the `Numpyro` package (Phan et al., 2019).

Inference test: If the ground truth of $\boldsymbol{\theta}$ is known for $\boldsymbol{d}$ (as is the case with mock observations), then the pipeline is able to produce a coverage plot that evaluates the validity of the statistical setup, and allows for a comparison of the performance of this approach against other methods. See Section 4.3 for more details on this.

To demonstrate the use of this approach, we apply it in the context of parameter inference for models of the Epoch of Reionization that consider fluctuating ultra-violet backgrounds, as explained in the following section.

3. Application: Constraining the EoR

The Epoch of Reionization (EoR) is the period of time in the evolution of the universe during which the neutral hydrogen in the inter-galactic medium (IGM) transitioned from being fully neutral (HI) to fully ionized (HII). A detailed discussion of the EoR and its study is outside the scope of this paper, but its characterization remains an important challenge in modern cosmology (Zaroubi, 2013; Shimabukuro et al., 2023). Among various different probes, the Lyman-alpha forest (LAF) is typically used to study the state of the IGM at the later stages of the EoR, as it is a series of redshifted absorption features caused by the presence of HI along the line of sight of observed quasar spectra.

Unfortunately, modeling the LAF is computationally expensive, which prohibits a proper quantitative comparison of the wide range of reionization scenarios that still need to be considered. In a typical setup, for instance, high-resolution hydrodynamical simulations are used to model the LAF and its observables for the different stages of the EoR (Doughty et al., 2023). Further post-processing of these simulations is usually done to consider different ultra-violet backgrounds (UVB). The high computational cost of these models makes statistical inferences using standard methods unfeasible, a problem that only worsens with more flexible models that have higher-dimensional parameter spaces. As an initial test of our approach, we replicate some of the results from a previous study (Wolfson et al., 2023), where the LAF is modeled for the case of an inhomogeneous UVB. We will refer to this study as W23 for the remainder of this article. To perform the parameter inference, W23 employs a combination of Nearest Grid-Point (NGP) interpolation and the standard Metropolis-Hastings algorithm. For this, a grid of ~ 550 models of both $\boldsymbol{\xi}_m$ and $\boldsymbol{\Sigma}_m$ as a function of $\lambda_{\rm mfp}$ and $\langle F \rangle$ for each redshift z that is being considered is generated, and is used to perform inferences on real data. We refer to this approach as the NGP + MH method for the remainder of this article. Replicating the results of W23 with the NN + HMC method serves as a non-trivial test of this technique.

3.1. Models of EoR

We refer the reader to W23 for a detailed description of the astrophysics and the meaning of the models described in this section. As a brief summary, the model in W23 uses the mean autocorrelation function of the LAF $\boldsymbol{\xi}_m$ as a summary statistic to constrain two physical parameters related to the EoR: the mean-free path of ionizing photons $\lambda_{\rm mfp}$ (responsible for ionizing the HI in the IGM), and the mean flux $\langle F \rangle$ of the flux skewers that are used to calculate $\boldsymbol{\xi}_m$. To obtain $\boldsymbol{\xi}_m$, a series of hydrodynamical simulations are used to model the LAF. Independent fluctuating UVBs are obtained with a semi-numerical method, and are then used to post-process the output from the simulations. For each simulation, a thousand mock Lyman-alpha flux skewers are forward-modeled, which are then used to calculate individual mock LAF autocorrelation functions $\boldsymbol{\xi}_i$. Lastly, the mean autocorrelation function of the LAF $\boldsymbol{\xi}_m$ is obtained by averaging over all the individual mock autocorrelation functions. Additionally, model-dependent covariance matrices $\boldsymbol{\Sigma}_m$ are calculated given the statistical setup that is used (see Section 3.2). These are calculated by using the following:

$$\boldsymbol{\Sigma}_m = \frac{1}{N_{\rm mocks}} \sum_{i=1}^{N_{\rm mocks}} (\boldsymbol{\xi}_i - \boldsymbol{\xi}_m)(\boldsymbol{\xi}_i - \boldsymbol{\xi}_m)^T \tag{4}$$

Where $N_{\rm mocks} = 500000$. In summary, this model allows us to calculate $\boldsymbol{\xi}_m$ as a function of λ_{mfp} and $\langle F \rangle$, as well as their corresponding, model-dependent covariance matrices $\boldsymbol{\Sigma}_m$.

3.2. Parameter inference

To constrain λ_{mfp} and $\langle F \rangle$ from a mock observation $\boldsymbol{\xi}_i$, the following multivariate Gaussian likelihood $\mathcal{L}(\boldsymbol{\xi}_i | \lambda_{mfp}, \langle F \rangle)$ is assumed:

$$\mathcal{L} = \kappa \exp\left(- \frac{(\boldsymbol{\xi}_i - \boldsymbol{\xi}_m))^T \boldsymbol{\Sigma}_m^{-1} (\boldsymbol{\xi}_i - \boldsymbol{\xi}_m)}{2} \right) \tag{5}$$

Where κ is given by:

$$\kappa = \frac{1}{\sqrt{\det(\boldsymbol{\Sigma}_m)(2\pi)^n}} \tag{6}$$

The proper adoption of this Gaussian likelihood requires a careful consideration of the covariance matrices $\boldsymbol{\Sigma}_m$ that are used when evaluating the likelihood. Ideally, the covariance

matrix is obtained directly from the observed data. With this approach, a noisy approximation of $\mathbf{\Sigma}_m$ is calculated by partitioning the data with bootsrapping or jacknifing, and subsequently recombining the samples. This has the advantage that the obtained estimate of $\mathbf{\Sigma}_m$ is agnostic to any underlying assumptions of the model that is being considered, since it is being determined empirically (Shirasaki et al., 2017; Alam et al., 2017; Hamana et al., 2020). Unfortunately, there is not enough high-quality data to do this for the autocorrelation function of the LAF at the redshifts in which we are interested. An alternative is to calculate $\mathbf{\Sigma}_m$ from mock observations (Gruen et al., 2015; Krause et al., 2017; Martinet et al., 2018), as explained in Section 3.1.

3.3. Emulating $\boldsymbol{\xi}_m$

We first need to construct an emulator for $\boldsymbol{\xi}_m$. This emulator needs to take the two input model parameters, and output an emulated mean autocorrelation function of the LAF. We choose to employ a fully connected, feed-forward neural network. For the training, we use the mean absolute percentage error (MAPE) as a loss function. For a single autocorrelation function, the MAPE is calculated by:

$$\text{MAPE} = \frac{1}{n_b} \sum_{i=1}^{n_b} \frac{|\xi_i - \hat{\xi}_i|}{\xi_i} \tag{7}$$

Where ξ is the truth, $\hat{\xi}$ is its emulation, and n_b is the number of velocity bins that an individual autocorrelation function has, and over which the mean error is calculated. Even though the training dataset is normalized before training (as is typically done in ML applications), the loss is calculated with the un-scaled autocorrelation functions. This improves the interpretability of the emulation error, as the calculated loss after training gives us a relative error calculated in terms of the original, astrophysical units.

3.4. Emulating $\mathbf{\Sigma}_m$

Emulating $\mathbf{\Sigma}_m$ requires more careful consideration, as covariance matrices are symmetric. We take advantage of Cholesky decomposition to satisfy this requirement, as it allows us to obtain a lower triangular matrix L (known as a Cholesky factor) for each $\mathbf{\Sigma}_m$, such that $\mathbf{\Sigma}_m = LL^T$. To further simplify the emulation, the Cholesky factors are reordered into 1D arrays that we name the flattened Cholesky factors L_{flat}. We opt to also use a fully connected, feed-forward neural network to emulate L_{flat}, which can then be used to reconstruct an emulated $\mathbf{\Sigma}_m$. For training, we also use an MAPE loss (Eq. 7) calculated on the un-scaled L_{flat} for each covariance matrix in the data set. Figure 5 in Appendix A shows an example of an emulated covariance matrix.

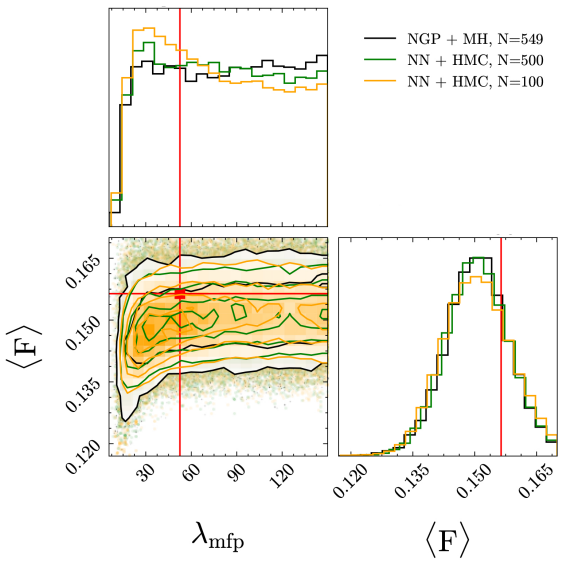

Figure 2. Example of an inference done on a single mock observation. It shows the overlaid corner plot obtained using the three distinct methods: NGP + MH using all 549 models, NN + HMC with $N_{\text{train}} + N_{\text{val}} = 500$, and NN + HMC with $N_{\text{train}} + N_{\text{val}} = 100$. The red line shows the true parameters of this mock observation, which are $\lambda_{\text{mfp}} = 52.5$Mpc and $\langle F \rangle = 0.1564$.

4. Results

For the final test, we decided to use a dataset of the models described in Section 3.1 at a redshift of $z = 5.1$. This dataset includes $N_{\text{total}} = 549$ models, which are split into $N_{\text{train}} = 450$ models for training, $N_{\text{val}} = 50$ for validating, and $N_{\text{test}} = 49$ for testing.

4.1. Emulation performance

Hyper-parameter tuning was performed for both emulators using Optuna (see Section 2.3) to explore different architecture and training parameters. Table 1 in Appendix B shows the results of the hyper-parameter tuning. We used the average MAPE calculated on the test set to evaluate the performance of the two emulators. For the $\boldsymbol{\xi}_m$ emulator, the error was calculated with the un-scaled $\boldsymbol{\xi}_m$ (the same as during training), and was found to be 0.0034. For the $\mathbf{\Sigma}_m$, the error was calculated with the covariance matrices (different from training, see Section 3.4), and was found to be 0.016. This means that both emulators produce sub 2% error on average. Figure 6 in Appendix B shows the distribution of these errors for both emulators.

4.2. Inference on mock observations

For comparison purposes, we performed parameter inference on 500 randomly selected mock observations of the

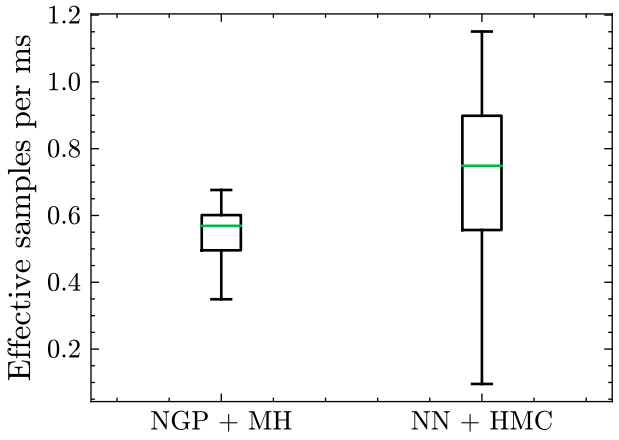

Figure 3. Box plot showing the distribution of the calculated number of effective samples per millisecond (ms). The green line represents the median, the top and bottom edges of the box represent the first quartile (Q1) and third quartile (Q3) of the data respectively, while whiskers extend from the box to the farthest data point lying within 1.5x the inter-quartile range from the box.

autocorrelation function $\boldsymbol{\xi}_i$ using both the NGP + MH and the NN + HMC methods. For each inference, we set the HMC (MH) to have 16 chains (walkers), taking 3500 steps each and removing 500 steps of each chain (walker) as a burn-in. Figure 2 shows an example of the inference results for a typical model using both of these methods, with two variations of the NN + HMC approach (see Section 4.4). An extensive discussion of the inference results is beyond the scope of this paper, and we refer the reader to W23 for further details. Visually inspecting the results of all 500 inferred posterior distributions suggest that both methods produce similar results.

As a way to compare the gain in efficiency that using HMC provides, we calculated the number of effective samples per millisecond n_{eff}/ms using the `ArviZ` python package (Kumar et al., 2019) for all 500 parameter inferences obtained with both methods. Figure 3 shows the results of this calculation, showing that the median of n_{eff}/ms for the NGP + MH and the NN + HMC methods are 0.57 and 0.75 respectively. Such a marginal gain in efficiency is to be expected given that the problem that we have tested it on has a small dimensional space (2 parameters), but we expect this gain to be more significant as the number of parameters increases (see Section 2.2).

4.3. Inference test

To further validate our results, we perform an inference test by creating coverage plots for both methods. Coverage plots are used to test the fidelity of the inference framework that is being employed, and it allows to test the validity of any assumptions that are done during inference (Prangle et al., 2013; Sellentin & Starck, 2019). M23 found that this statistical setup (see Section 3.2) leads to overconfident posteriors, which is likely caused by the assumption of a Gaussian likelihood. Since we performed all the inferences on mock observations (where we know the ground truth), it is possible for us to make a direct comparison of the coverage that our method produces. Figure 4 shows a comparison of the coverage obtained with the NGP + MH and the NN + HMC methods. As we can see, our method is able to reproduce the same coverage, providing strong evidence that the neural emulators and their application with HMC are producing posterior distributions that have similar statistical behavior to the previous method.

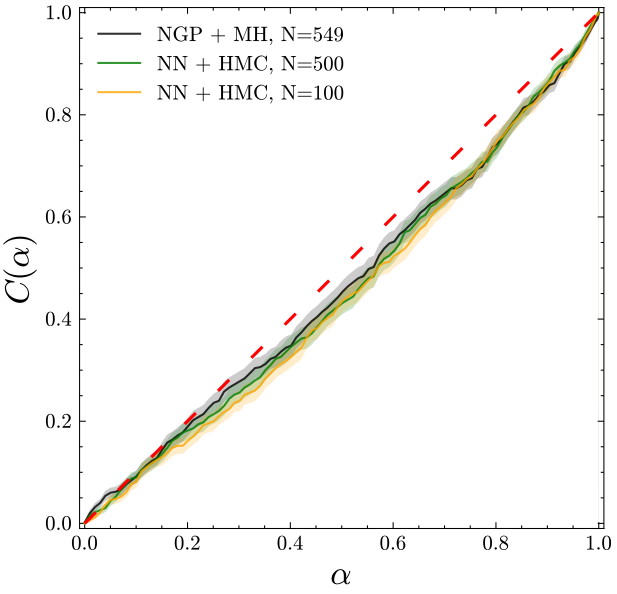

Figure 4. Comparison of coverage plots between different methods. We can see that using the full dataset for training the emulators produces similar results to the original method from M23, and that using a reduced dataset produces competitvely similar results (with a small degradation), at a significant decrease of computational cost.

4.4. Reducing the training set

As a last test to our method, we experimented with decreasing the size of the training and validation data sets to see how the inference would perform. For this, a reduced training and validation sets with $N_{\text{test}} = 65$ and $N_{\text{val}} = 35$ were created by randomly selecting from the original sets. After hyper-parameter tuning, the performance of the emulators was tested with the same testing set as in Section 4.1. Both emulators obtained a sub 2% error on average, with the $\boldsymbol{\xi}_m$

emulator obtaining an MAPE of 0.0046, and the $\mathbf{\Sigma}_m$ an error of 0.014. The inference on the same 500 mocks was then performed using this new emulators for comparison, and a coverage plot was also produced. Figure 2 shows the results of doing the inference with these emulators on a typical mock, and Figure 4 shows the coverage plot. As we can observe, there is a noticeable degradation in performance in the inference, likely caused by having significantly less models in the training and validation sets. However, the results are competitive, and provide evidence on how the NN + HMC method can further reduce the overall computational cost of this inference task by requiring significantly fewer models.

5. Conclusions

We started the development of a pipeline that is able to apply the NN + HMC method and then tested it in the context of cosmological parameter inference. We demonstrated the potential that this method has to accelerate statistical inferences while also reducing the overall computational cost. In future work, we will continue the development of our software, and apply it to a more complex problem with a higher dimensional parameter space that will better show the advantages of this approach.

Acknowledgements

We acknowledge helpful conversations with the ENIGMA group at UC Santa Barbara and Leiden University. JFH acknowledges support from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 885301) and from the National Science Foundation under Grant No. 1816006. This work made use of NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020), sklearn (Pedregosa et al., 2011), Astropy (Astropy Collaboration et al., 2013; 2018; 2022), h5py (Collette, 2013), Matplotlib (Hunter, 2007), corner.py (Foreman-Mackey, 2016), and IPython (Pérez & Granger, 2007).

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Alam, S., Ata, M., Bailey, S., Beutler, F., Bizyaev, D., Blazek, J. A., Bolton, A. S., Brownstein, J. R., Burden, A., Chuang, C.-H., Comparat, J., Cuesta, A. J., Dawson, K. S., Eisenstein, D. J., Escoffier, S., Gil-Marín, H., Grieb, J. N., Hand, N., Ho, S., Kinemuchi, K., Kirkby, D., Kitaura, F., Malanushenko, E., Malanushenko, V., Maraston, C., McBride, C. K., Nichol, R. C., Olmstead, M. D., Oravetz, D., Padmanabhan, N., Palanque-Delabrouille, N., Pan, K., Pellejero-Ibanez, M., Percival, W. J., Petitjean, P., Prada, F., Price-Whelan, A. M., Reid, B. A., Rodríguez-Torres, S. A., Roe, N. A., Ross, A. J., Ross, N. P., Rossi, G., Rubiño-Martín, J. A., Saito, S., Salazar-Albornoz, S., Samushia, L., Sánchez, A. G., Satpathy, S., Schlegel, D. J., Schneider, D. P., Scóccola, C. G., Seo, H.-J., Sheldon, E. S., Simmons, A., Slosar, A., Strauss, M. A., Swanson, M. E. C., Thomas, D., Tinker, J. L., Tojeiro, R., Magaña, M. V., Vazquez, J. A., Verde, L., Wake, D. A., Wang, Y., Weinberg, D. H., White, M., Wood-Vasey, W. M., Yèche, C., Zehavi, I., Zhai, Z., and Zhao, G.-B. The clustering of galaxies in the completed SDSS-III Baryon Oscillation Spectroscopic Survey: cosmological analysis of the DR12 galaxy sample. , 470(3): 2617–2652, September 2017. doi: 10.1093/mnras/stx721.

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P. H., Unther, H. M., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J. E. H., Singer, L., Fox, R., Weaver, B. A., Zabalza, V., Edwards, Z. I., Azalee Bostroem, K., Burke, D. J., Casey, A. R., Crawford, S. M., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lim, P. L., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M., and Streicher, O. Astropy: A community Python package for astronomy. , 558:A33, October 2013. doi: 10.1051/0004-6361/201322068.

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., Günther, H. M., Lim, P. L., Crawford, S. M., Conseil, S., Shupe, D. L., Craig, M. W., Dencheva, N., Ginsburg, A., Vand erPlas, J. T., Bradley, L. D., Pérez-Suárez, D., de Val-Borro, M., Aldcroft, T. L., Cruz, K. L., Robitaille, T. P., Tollerud, E. J., Ardelean, C., Babej, T., Bach, Y. P., Bachetti, M., Bakanov, A. V., Bamford, S. P., Barentsen, G., Barmby, P., Baumbach, A., Berry, K. L., Biscani, F., Boquien, M., Bostroem, K. A., Bouma, L. G., Brammer, G. B., Bray, E. M., Breytenbach, H., Buddelmeijer, H., Burke, D. J., Calderone, G., Cano Rodríguez, J. L., Cara, M., Cardoso, J. V. M., Cheedella, S., Copin, Y., Corrales, L., Crichton, D., D'Avella, D., Deil, C., Depagne, É., Dietrich, J. P., Donath, A., Droettboom, M., Earl, N., Erben, T., Fabbro, S., Ferreira, L. A., Finethy, T., Fox, R. T., Garrison, L. H., Gibbons, S. L. J., Goldstein, D. A., Gommers, R., Greco, J. P., Greenfield, P., Groener, A. M., Grollier, F., Hagen, A., Hirst, P., Homeier, D., Horton, A. J., Hosseinzadeh, G., Hu, L., Hunkeler, J. S., Ivezić, Ž., Jain, A., Jenness, T., Kanarek, G., Kendrew, S., Kern, N. S., Kerzendorf, W. E., Khvalko, A., King, J.,

Kirkby, D., Kulkarni, A. M., Kumar, A., Lee, A., Lenz, D., Littlefair, S. P., Ma, Z., Macleod, D. M., Mastropietro, M., McCully, C., Montagnac, S., Morris, B. M., Mueller, M., Mumford, S. J., Muna, D., Murphy, N. A., Nelson, S., Nguyen, G. H., Ninan, J. P., Nöthe, M., Ogaz, S., Oh, S., Parejko, J. K., Parley, N., Pascual, S., Patil, R., Patil, A. A., Plunkett, A. L., Prochaska, J. X., Rastogi, T., Reddy Janga, V., Sabater, J., Sakurikar, P., Seifert, M., Sherbert, L. E., Sherwood-Taylor, H., Shih, A. Y., Sick, J., Silbiger, M. T., Singanamalla, S., Singer, L. P., Sladen, P. H., Sooley, K. A., Sornarajah, S., Streicher, O., Teuben, P., Thomas, S. W., Tremblay, G. R., Turner, J. E. H., Terrón, V., van Kerkwijk, M. H., de la Vega, A., Watkins, L. L., Weaver, B. A., Whitmore, J. B., Woillez, J., Zabalza, V., and Astropy Contributors. The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package. , 156(3):123, September 2018. doi: 10.3847/1538-3881/aabc4f.

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., Earl, N., Starkman, N., Bradley, L., Shupe, D. L., Patil, A. A., Corrales, L., Brasseur, C. E., N"othe, M., Donath, A., Tollerud, E., Morris, B. M., Ginsburg, A., Vaher, E., Weaver, B. A., Tocknell, J., Jamieson, W., van Kerkwijk, M. H., Robitaille, T. P., Merry, B., Bachetti, M., G"unther, H. M., Aldcroft, T. L., Alvarado-Montes, J. A., Archibald, A. M., B'odi, A., Bapat, S., Barentsen, G., Baz'an, J., Biswas, M., Boquien, M., Burke, D. J., Cara, D., Cara, M., Conroy, K. E., Conseil, S., Craig, M. W., Cross, R. M., Cruz, K. L., D'Eugenio, F., Dencheva, N., Devillepoix, H. A. R., Dietrich, J. P., Eigenbrot, A. D., Erben, T., Ferreira, L., Foreman-Mackey, D., Fox, R., Freij, N., Garg, S., Geda, R., Glattly, L., Gondhalekar, Y., Gordon, K. D., Grant, D., Greenfield, P., Groener, A. M., Guest, S., Gurovich, S., Handberg, R., Hart, A., Hatfield-Dodds, Z., Homeier, D., Hosseinzadeh, G., Jenness, T., Jones, C. K., Joseph, P., Kalmbach, J. B., Karamehmetoglu, E., Kaluszy'nski, M., Kelley, M. S. P., Kern, N., Kerzendorf, W. E., Koch, E. W., Kulumani, S., Lee, A., Ly, C., Ma, Z., MacBride, C., Maljaars, J. M., Muna, D., Murphy, N. A., Norman, H., O'Steen, R., Oman, K. A., Pacifici, C., Pascual, S., Pascual-Granado, J., Patil, R. R., Perren, G. I., Pickering, T. E., Rastogi, T., Roulston, B. R., Ryan, D. F., Rykoff, E. S., Sabater, J., Sakurikar, P., Salgado, J., Sanghi, A., Saunders, N., Savchenko, V., Schwardt, L., Seifert-Eckert, M., Shih, A. Y., Jain, A. S., Shukla, G., Sick, J., Simpson, C., Singanamalla, S., Singer, L. P., Singhal, J., Sinha, M., SipHocz, B. M., Spitler, L. R., Stansby, D., Streicher, O., Sumak, J., Swinbank, J. D., Taranu, D. S., Tewary, N., Tremblay, G. R., Val-Borro, M. d., Van Kooten, S. J., Vasovi'c, Z., Verma, S., de Miranda Cardoso, J. V., Williams, P. K. G., Wilson, T. J., Winkel, B., Wood-Vasey, W. M., Xue, R., Yoachim, P., Zhang, C., Zonca, A., and Astropy Project Contributors. The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package. , 935(2):167, August 2022. doi: 10.3847/1538-4357/ac7c74.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Collette, A. *Python and HDF5*. O'Reilly, 2013.

Doughty, C. C., Hennawi, J. F., Davies, F. B., Lukić, Z., and Oñorbe, J. Convergence of small scale Ly α structure at high-z under different reionization scenarios. , 525 (3):3790–3805, November 2023. doi: 10.1093/mnras/stad2549.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987. ISSN 0370-2693. doi: https://doi.org/10.1016/0370-2693(87)91197-X. URL https://www.sciencedirect.com/science/article/pii/037026938791197X.

Euclid Collaboration, Knabenhans, M., Stadel, J., Marelli, S., Potter, D., Teyssier, R., Legrand, L., Schneider, A., Sudret, B., Blot, L., Awan, S., Burigana, C., Carvalho, C. S., Kurki-Suonio, H., and Sirri, G. Euclid preparation: II. The EUCLIDEMULATOR - a tool to compute the cosmology dependence of the nonlinear matter power spectrum. , 484(4):5509–5529, April 2019. doi: 10.1093/mnras/stz197.

Feroz, F. and Hobson, M. P. Multimodal nested sampling: an efficient and robust alternative to Markov Chain Monte Carlo methods for astronomical data analyses. , 384(2): 449–463, February 2008. doi: 10.1111/j.1365-2966.2007.12353.x.

Feroz, F., Hobson, M. P., and Bridges, M. MULTINEST: an efficient and robust Bayesian inference tool for cosmology and particle physics. , 398(4):1601–1614, October 2009. doi: 10.1111/j.1365-2966.2009.14548.x.

Foreman-Mackey, D. corner.py: Scatterplot matrices in python. *The Journal of Open Source Software*, 1(2): 24, jun 2016. doi: 10.21105/joss.00024. URL https://doi.org/10.21105/joss.00024.

Foreman-Mackey, D., Hogg, D. W., Lang, D., and Goodman, J. emcee: The mcmc hammer. *Publications of the Astronomical Society of the Pacific*, 125(925): 306, feb 2013. doi: 10.1086/670067. URL https://dx.doi.org/10.1086/670067.

Gong, Z., Halder, A., Barreira, A., Seitz, S., and Friedrich, O. Cosmology from the integrated shear 3-point correlation function: simulated likelihood analyses with machine-learning emulators. , 2023(7):040, July 2023. doi: 10.1088/1475-7516/2023/07/040.

Gruen, D., Seitz, S., Becker, M. R., Friedrich, O., and Mana, A. Cosmic variance of the galaxy cluster weak lensing signal. , 449(4):4264–4276, June 2015. doi: 10.1093/mnras/stv532.

Hamana, T., Shirasaki, M., Miyazaki, S., Hikage, C., Oguri, M., More, S., Armstrong, R., Leauthaud, A., Mandelbaum, R., Miyatake, H., Nishizawa, A. J., Simet, M., Takada, M., Aihara, H., Bosch, J., Komiyama, Y., Lupton, R., Murayama, H., Strauss, M. A., and Tanaka, M. Cosmological constraints from cosmic shear two-point correlation functions with HSC survey first-year data. , 72(1):16, February 2020. doi: 10.1093/pasj/psz138.

Handley, W. J., Hobson, M. P., and Lasenby, A. N. polychord: nested sampling for cosmology. , 450:L61–L65, June 2015. doi: 10.1093/mnrasl/slv047.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M. Flax: A neural network library and ecosystem for JAX, 2023. URL http://github.com/google/flax.

Heitmann, K., Lawrence, E., Kwan, J., Habib, S., and Higdon, D. The Coyote Universe Extended: Precision Emulation of the Matter Power Spectrum. , 780(1):111, January 2014. doi: 10.1088/0004-637X/780/1/111.

Hoffman, M. and Gelman, A. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15, 11 2011.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Karamanis, M., Beutler, F., Peacock, J. A., Nabergoj, D., and Seljak, U. Accelerating astronomical and cosmological inference with preconditioned Monte Carlo. *Monthly Notices of the Royal Astronomical Society*, 516(2):1644–1653, 08 2022. ISSN 0035-8711. doi: 10.1093/mnras/stac2272. URL https://doi.org/10.1093/mnras/stac2272.

Krause, E., Eifler, T. F., Zuntz, J., Friedrich, O., Troxel, M. A., Dodelson, S., Blazek, J., Secco, L. F., MacCrann, N., Baxter, E., Chang, C., Chen, N., Crocce, M., DeRose, J., Ferte, A., Kokron, N., Lacasa, F., Miranda, V., Omori, Y., Porredon, A., Rosenfeld, R., Samuroff, S., Wang, M., Wechsler, R. H., Abbott, T. M. C., Abdalla, F. B., Allam, S., Annis, J., Bechtol, K., Benoit-Levy, A., Bernstein, G. M., Brooks, D., Burke, D. L., Capozzi, D., Carrasco Kind, M., Carretero, J., D'Andrea, C. B., da Costa, L. N., Davis, C., DePoy, D. L., Desai, S., Diehl, H. T., Dietrich, J. P., Evrard, A. E., Flaugher, B., Fosalba, P., Frieman, J., Garcia-Bellido, J., Gaztanaga, E., Giannantonio, T., Gruen, D., Gruendl, R. A., Gschwend, J., Gutierrez, G., Honscheid, K., James, D. J., Jeltema, T., Kuehn, K., Kuhlmann, S., Lahav, O., Lima, M., Maia, M. A. G., March, M., Marshall, J. L., Martini, P., Menanteau, F., Miquel, R., Nichol, R. C., Plazas, A. A., Romer, A. K., Rykoff, E. S., Sanchez, E., Scarpine, V., Schindler, R., Schubnell, M., Sevilla-Noarbe, I., Smith, M., Soares-Santos, M., Sobreira, F., Suchyta, E., Swanson, M. E. C., Tarle, G., Tucker, D. L., Vikram, V., Walker, A. R., and Weller, J. Dark Energy Survey Year 1 Results: Multi-Probe Methodology and Simulated Likelihood Analyses. *arXiv e-prints*, art. arXiv:1706.09359, June 2017. doi: 10.48550/arXiv.1706.09359.

Kumar, R., Carroll, C., Hartikainen, A., and Martin, O. Arviz a unified library for exploratory analysis of bayesian models in python. *Journal of Open Source Software*, 4(33):1143, 2019. doi: 10.21105/joss.01143. URL https://doi.org/10.21105/joss.01143.

Lewis, A. and Bridle, S. Cosmological parameters from cmb and other data: A monte carlo approach. *Phys. Rev. D*, 66:103511, Nov 2002. doi: 10.1103/PhysRevD.66.103511. URL https://link.aps.org/doi/10.1103/PhysRevD.66.103511.

Martinet, N., Schneider, P., Hildebrandt, H., Shan, H., Asgari, M., Dietrich, J. P., Harnois-Déraps, J., Erben, T., Grado, A., Heymans, C., Hoekstra, H., Klaes, D., Kuijken, K., Merten, J., and Nakajima, R. KiDS-450: cosmological constraints from weak-lensing peak statistics - II: Inference from shear peaks using N-body simulations. , 474(1):712–730, February 2018. doi: 10.1093/mnras/stx2793.

Neal, R. MCMC Using Hamiltonian Dynamics. In *Handbook of Markov Chain Monte Carlo*, pp. 113–162. 2011. doi: 10.1201/b10905.

Nygaard, A., Holm, E. B., Hannestad, S., and Tram, T. CONNECT: a neural network based framework for emu-

lating cosmological observables and cosmological parameter inference. , 2023(5):025, May 2023. doi: 10.1088/1475-7516/2023/05/025.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pérez, F. and Granger, B. E. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007. ISSN 1521-9615. doi: 10.1109/MCSE.2007.53. URL https://ipython.org.

Phan, D., Pradhan, N., and Jankowiak, M. Composable effects for flexible and accelerated probabilistic programming in numpyro. *arXiv preprint arXiv:1912.11554*, 2019.

Piras, D. and Spurio Mancini, A. Cosmopower-jax: high-dimensional bayesian inference with differentiable cosmological emulators. *The Open Journal of Astrophysics*, 6, 6 2023. doi: 10.21105/astro.2305.06347.

Prangle, D., Blum, M. G. B., Popovic, G., and Sisson, S. A. Diagnostic tools of approximate Bayesian computation using the coverage property. *arXiv e-prints*, art. arXiv:1301.3166, January 2013. doi: 10.48550/arXiv.1301.3166.

Ruiz-Zapatero, J., Alonso, D., García-García, C., Nicola, A., Mootoovaloo, A., Sullivan, J. M., Bonici, M., and Ferreira, P. G. LimberJack.jl: auto-differentiable methods for angular power spectra analyses. *The Open Journal of Astrophysics*, 7:11, February 2024. doi: 10.21105/astro.2310.08306.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Sellentin, E. and Starck, J.-L. Debiasing inference with approximate covariance matrices and other unidentified biases. , 2019(8):021, August 2019. doi: 10.1088/1475-7516/2019/08/021.

Shimabukuro, H., Hasegawa, K., Kuchinomachi, A., Yajima, H., and Yoshiura, S. Exploring the cosmic dawn and epoch of reionization with the 21 cm line. , 75:S1–S32, February 2023. doi: 10.1093/pasj/psac042.

Shirasaki, M., Takada, M., Miyatake, H., Takahashi, R., Hamana, T., Nishimichi, T., and Murata, R. Robust covariance estimation of galaxy-galaxy weak lensing: validation and limitation of jackknife covariance. , 470(3):3476–3496, September 2017. doi: 10.1093/mnras/stx1477.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Walther, M., Oñorbe, J., Hennawi, J. F., and Lukić, Z. New Constraints on IGM Thermal Evolution from the Lyα Forest Power Spectrum. , 872(1):13, February 2019. doi: 10.3847/1538-4357/aafad1.

Wolfson, M., Hennawi, J. F., Davies, F. B., and Oñorbe, J. Forecasting constraints on the mean free path of ionizing photons at z $\geq$ 5.4 from the Lyman-α forest flux autocorrelation function. , 521(3):4056–4073, May 2023. doi: 10.1093/mnras/stad701.

Zaroubi, S. The Epoch of Reionization. In Wiklind, T., Mobasher, B., and Bromm, V. (eds.), *The First Galaxies*, volume 396 of *Astrophysics and Space Science Library*, pp. 45, January 2013. doi: 10.1007/978-3-642-32362-1_2.

A. Example of Covariance Matrix Emulation

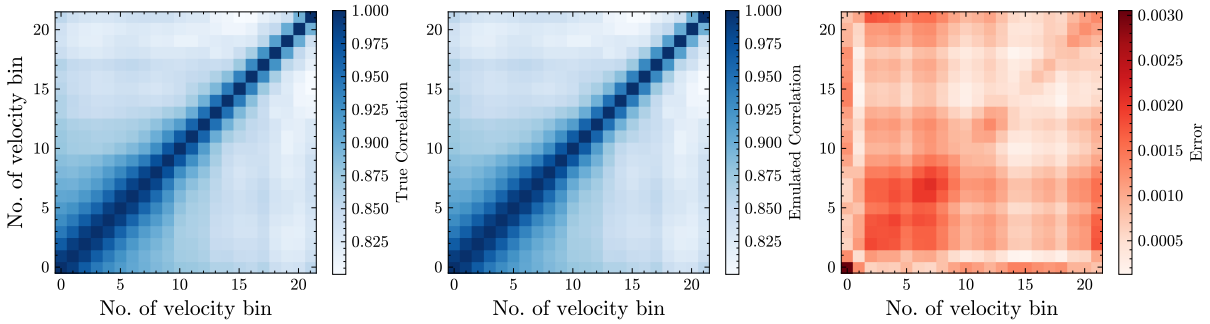

Figure 5. Example of an emulation of a covariance matrix, visualized as correlation matrices. The left correlation matrix represents the true covariance matrix, the middle one shows the corresponding emulated covariance matrix, and the right shows a color map of the relative error.

As explained in Section 3.4, emulating covariance matrices require special attention given that they are symmetric. Taking advantage of Cholesky decomposition and reordering the resulting lower triangular matrices into 1D arrays allows us to employ a simple architecture for the neural network. To visualize a covariance matrix $\mathbf{\Sigma}$, we first convert it into a correlation matrix $\boldsymbol{C}$ using the following:

$$\boldsymbol{C}_{i,j} = \frac{\mathbf{\Sigma}_{i,j}}{\sqrt{\mathbf{\Sigma}_{i,i}\mathbf{\Sigma}_{j,j}}}$$

Where the i and j subscripts refer the the element in the ith and jth element in the covariance matrix. Figure 5 shows an example of a typical covariance matrix from the model explained in Section 3.1, its corresponding emulated covariance matrix, and the emulation error.

B. Performance of emulators

B.1. Hyper-parameter tuning

The hyper-parameter tuning is carried out with `Optuna`, as explained in Section 2.3. For the application in Section 3, the hyper-parameters that were varied for both emulators were the number of hidden layers, the number of perceptrons in each hidden layer, the number of epochs, and the starting learning rate (for this application, both emulators were trained with a cosine scheduler for the learning rate). Table 1 shows the results of the hyper-parameter tuning for both emulators.

Table 1. Results of the hyper parameter-tuning for both emulators.

EMULATOR	LAYERS	PERCEPTRONS	STARTING LEARNING RATE	EPOCHS
$\boldsymbol{\xi}_m$	5	[4, 8, 12, 16, 20]	0.03523	750
$\mathbf{\Sigma}_m$	6	[4, 8, 8, 8, 16, 20]	0.0393	1500

B.2. Distribution of Errors

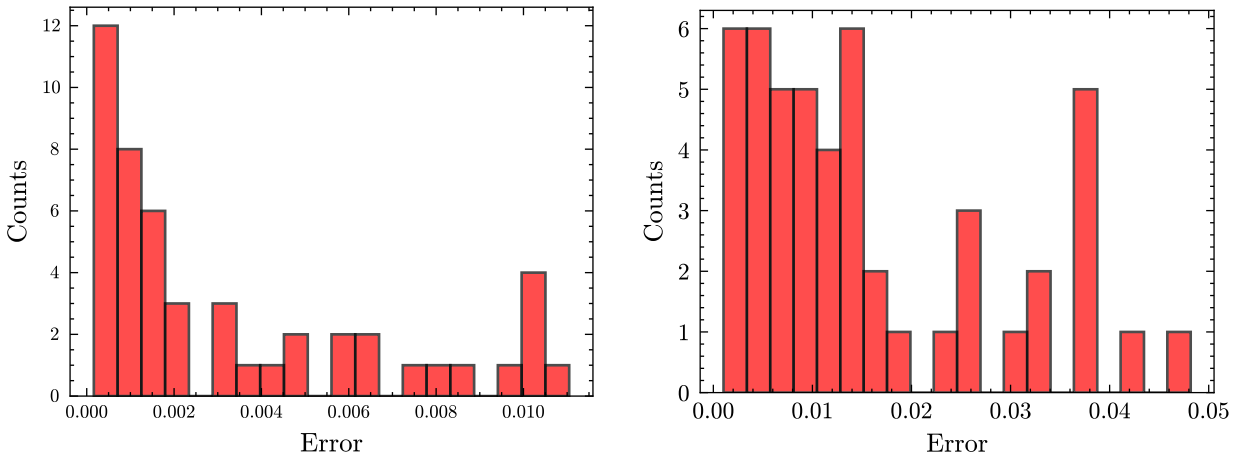

Figure 6. Distribution of errors calculated on the test set ($N_{\text{test}} = 49$) to evaluate the performance of the emulators trained with $N_{train} + N_{val} = 500$. *Left:* Distribution of MAPE calculated with the $\boldsymbol{\xi}_m$ emulator. *Right:* Distribution of MAPE calculated for the $\boldsymbol{\Sigma}_m$ emulator.