Up to two billion times acceleration of scientific simulations with deep neural architecture search

M. F. Kasim,[1, *] D. Watson-Parris,[2] L. Deaconu,[2] S. Oliver,[3] P. Hatfield,[1] D. H. Froula,[4] G. Gregori,[1] M. Jarvis,[5] S. Khatiwala,[3] J. Korenaga,[6] J. Topp-Mugglestone,[1] E. Viezzer,[7, 8] and S. M. Vinko[1]

[1]*Clarendon Laboratory, Department of Physics, University of Oxford, Parks Road, Oxford, UK*
[2]*Atmospheric, Oceanic and Planetary Physics, Department of Physics, University of Oxford, Oxford, UK*
[3]*Department of Earth Sciences, University of Oxford, Oxford, UK*
[4]*Laboratory for Laser Energetics, University of Rochester, New York, USA*
[5]*Denys Wilkinson Building, Department of Physics, University of Oxford, Keble Road, Oxford, UK*
[6]*Department of Geology and Geophysics, Yale University, New Haven, Connecticut, USA*
[7]*Department of Atomic, Molecular and Nuclear Physics, University of Seville, 41012 Seville, Spain*
[8]*Max-Planck-Institut für Plasmaphysik, EURATOM Association, Boltzmannstr. 2, 85748 Garching, Germany*

(Dated: January 23, 2020)

Computer simulations are invaluable tools for scientific discovery. However, accurate simulations are often slow to execute, which limits their applicability to extensive parameter exploration, large-scale data analysis, and uncertainty quantification. A promising route to accelerate simulations by building fast emulators with machine learning requires large training datasets, which can be prohibitively expensive to obtain with slow simulations. Here we present a method based on neural architecture search to build accurate emulators even with a limited number of training data. The method successfully accelerates simulations by up to 2 billion times in 10 scientific cases including astrophysics, climate science, biogeochemistry, high energy density physics, fusion energy, and seismology, using the same super-architecture, algorithm, and hyperparameters. Our approach also inherently provides emulator uncertainty estimation, adding further confidence in their use. We anticipate this work will accelerate research involving expensive simulations, allow more extensive parameters exploration, and enable new, previously unfeasible computational discovery.

I. INTRODUCTION

Finding a general approach to speed up a large class of simulations would enable tasks that are otherwise prohibitively expensive and accelerate scientific research. For example, fast and accurate simulations promise to speed up new materials and drug discovery[1] by allowing rapid screening and ideas testing. Accelerated simulations also open up novel possibilities for online diagnostics for cases like x-ray scattering in plasma physics experiments[2] and to monitor edge-localized modes in magnetic confinement fusion,[3] enabling real-time prediction-based experimental control and optimization. However, for such applications to be successful the simulations need not only be fast but also accurate; achieving both to the level required for advanced applications remains an active objective of current research.

One popular approach to speeding up simulations is to train machine learning models to emulate slow simulations[4–7] and use the emulators instead. The main challenge in constructing emulators with machine learning models is in their need of large amounts of training data to achieve the required accuracy in replicating the outputs of the simulations, which could be prohibitively expensive to generate with slow simulations.

To construct high fidelity emulators with limited training data, the machine learning models need to have a good prior on the simulation models. Most work to date in building emulators using random forests,[4] Gaussian Processes,[5] or other machine learning models[6,7] do not fully capture the correlation among the output points, limiting their accuracy in emulating simulations with one, two, or three-dimensional output signals. On the other hand, convolutional neural network (CNN) have shown to have a good prior on natural signals,[8] making them suitable for processing natural n-dimensional signals. However, as the CNN priors inherently rely on their architectures,[8] one has to find an architecture that gives the suitable prior of a given problem. Manually searching for the right architecture can take a significant amount of time and domain-specific expertise and often produces sub-optimal results.

Here we propose to solve this problem by employing efficient neural architecture search[9] to simultaneously find the neural network architecture that is best suited for a given case and train it. With the efficient neural architecture search and a novel super-architecture presented in this work, the algorithm can find and train fast emulators for a wide range of applications while offering major improvements in terms of accuracy compared with other techniques, even when the training data is limited. We call the presented method Deep Emulator Network SEarch (DENSE).

In DENSE, we start by defining the search space of neural network architectures in a form of super-architecture. A super-architecture consists of multiple nodes where the first node represents the simulation inputs and the last node the predicted simulation outputs. Each pair of nodes is connected by multiple groups of operations. Each group consists of a set of operations, such as 1×1 convolution, 3×3 convolution, or similar. Most of the operations, such as convolution, contain sets

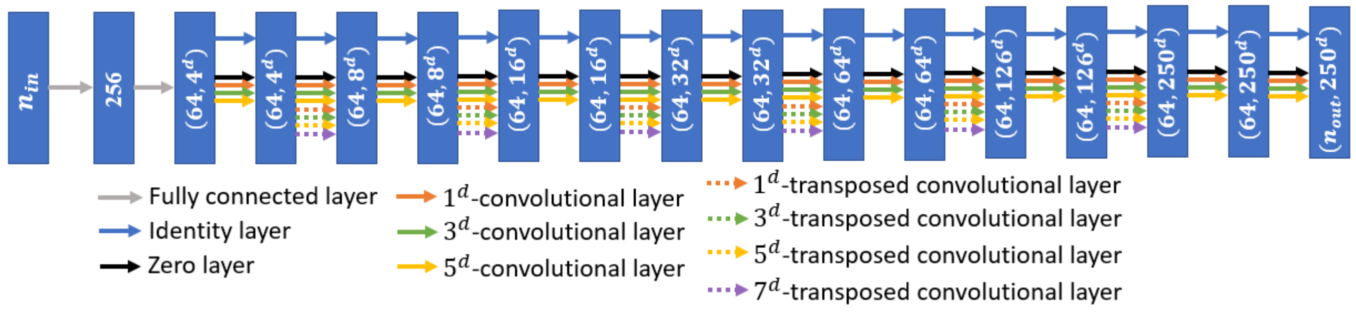

FIG. 1. The super-architecture used in this paper where d is the output signal's dimension. The first numbers in the brackets indicate the number of channels and the last numbers indicate the signal size. For cases where the output signal are not 250^d, then all the sizes in the intermediate nodes are scaled accordingly. Close arrows indicate the operations in the same group. The output of the identity layer and the selected of convolutional layer are added into the destination node.

of trainable values that are commonly known as *weights*. In one forward calculation of the neural network (i.e. predicting a set of outputs given some input), only one operation per group is chosen according to their assigned probability. The probability of an operation being chosen is determined by a trainable value associated with the operation, which we call the *network variable*.

The super-architecture used in this work is shown in Figure 1. In every group in the super-architecture, there are convolutional layers with different kernel sizes and a zero layer that multiplies the input with zero. The option of having a zero layer and multiple convolutional layers enable the algorithm to choose an appropriate architecture complexity for a given problem. The super-architecture also contains skip connections[10] (i.e. identity layers) to make it easier to train.

Training the neural network involves two update steps. In the first step, an operation for each group is chosen according to their probability, forming one possible architecture. The weights, $\mathbf{w}$, of the selected operations are then updated to minimize the expected value of a defined loss function, $\mathcal{L}$, between the predicted simulation output and the actual simulation output,

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha_1 \nabla_{\mathbf{w}} \mathbb{E}_{a\sim\mathcal{A}(\mathbf{b})} \left[\mathcal{L}(\mathbf{w}|\mathbf{X_t}, \mathbf{y_t}, a)\right], \quad (1)$$

where α_1 is the update step size, $\mathbf{X_t}$ and $\mathbf{y_t}$ are the input and output from the training dataset, a is an architecture sampled from the super-architecture $\mathcal{A}(\mathbf{b})$ according to the network variables, $\mathbf{b}$. The loss function in this paper is defined as the Huber loss function[11] to minimize the effect of outlier data and increase robustness.

The second update step involves evaluating the performance of various sampled architectures on the validation dataset, which is different from the training dataset employed in the first step. The performance of an architecture can be evaluated based on the loss function, inference time, power consumption, or some other combination of relevant criteria. The architectures are then ranked based on their performance and they are given rewards according to their rank. The network variables, $\mathbf{b}$, are updated to increase the probability of the high-ranked architectures and decrease the probability of the low-ranked architectures. Formally, the update can be written as,[12]

$$\mathbf{b} \leftarrow \mathbf{b} + \alpha_2 \mathbb{E}_{a\sim\mathcal{A}(\mathbf{b})} \left(\mathcal{R}_a \nabla_{\mathbf{b}} \log\left[\pi(a|\mathbf{b})\right]\right), \quad (2)$$

where α_2 is the update step size, $\mathcal{R}_a$ is the reward value given to the architecture a, and the function $\pi(a|\mathbf{b})$ is the likelihood of the architecture a being chosen given the network variables $\mathbf{b}$.

In this case, we ranked the architectures based on the Huber loss[11] on the validation dataset and gave the rewards to follow the zero-mean ranking function in CMA-ES.[13] The use of zero-mean ranking function reduces the update variance and makes the update step scale-invariant, increasing the robustness of the algorithm.

II. RESULTS

The combined update steps from equations (1) and (2), and the use of a ranking function in assigning rewards, make DENSE a robust algorithm to simultaneously learn the weights and find the right architecture for a given problem. To illustrate this, we apply the method to ten distinct scientific simulation cases: inelastic x-ray Thomson scattering (XRTS) in high-energy-density physics,[2,14] optical Thomson scattering (OTS) in laboratory astrophysics,[15] tokamak edge-localised modes diagnostics (ELMs) in fusion energy science,[3] x-ray emission spectroscopy (XES) in plasmas,[16,17] galaxy halo occupation distribution modelling (Halo) in astrophysics,[18] seismic tomography of the Shatsky Rise oceanic plateau (SeisTomo),[19] global aerosol-climate modelling using a general circulation model (GCM) in climate science,[20] oceanic pelagic stoichiometry modelling (MOPS) in biogeochemistry,[21] and neutron imaging (ICF JAG) and scalar measurements (ICF JAG Scalars) in inertial confinement fusion experiments.[22]

The tested simulations have ranging numbers of input parameters from 3 to 14 and outputs from 0D (scalars) to multiple 3D signals. Datasets for simulations that run in less than 1 CPU-hour were generated by running them

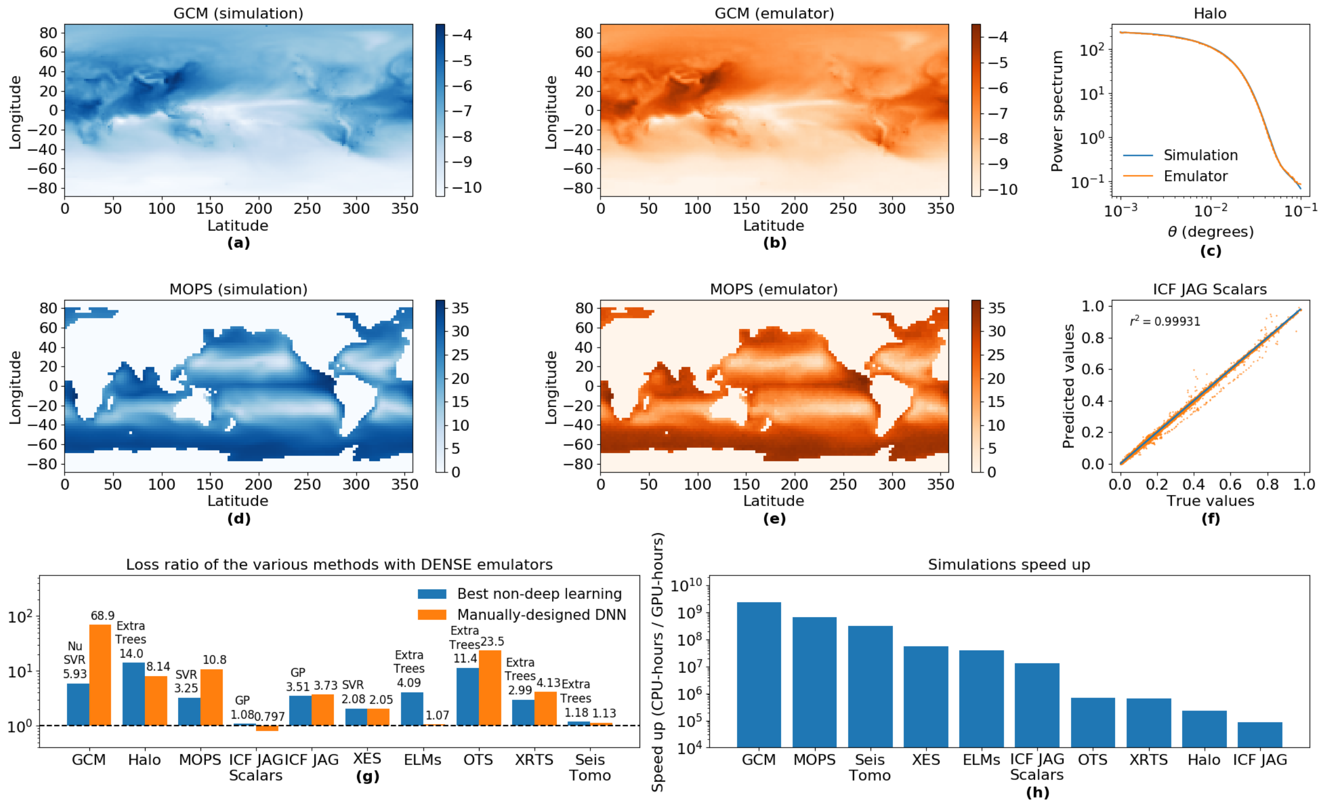

FIG. 2. (a-f) Examples of the emulators output compared to the simulations output taken from the test dataset. Examples of emulator outputs on all test cases can be found on Figure 5. (g) The ratio between the loss function obtained by DENSE emulators and the best loss function found by non-deep learning methods and manually-designed deep neural network. (h) The achieved speed up of the emulators using GPU compared to the original simulations.

14,000 times with random sets of inputs. For simulations that take hundreds to thousands of CPU-hours, such as MOPS and GCM, only 410 and 39 data points are collected, respectively. Each dataset is divided into three parts: 50% is used as the training dataset, 21% for validation, and 29% as the test dataset. The test dataset was used only to present the results in this paper, never to build the models. The hyperparameters were obtained by optimizing the result for OTS with CMA-ES,[13,23] then used for other cases without further tuning.

A. Emulator results

The example output of the trained emulators with DENSE are shown in Figures 2(a-f). We see that the output of the emulators generally matches closely the output of the actual simulations, even in MOPS and GCM where only 410 and 39 data points are available. When only a limited number of data is available, the choice of model architectures that give the right priors become important. Complex model architectures with bad priors could still fit the sampled training data, but likely to overfit, giving bad accuracy on the out-of-samples data.

With DENSE, the model architecture that gives a good prior on the problem is automatically searched for by preferring models that can fit well the out-of-samples data (i.e. the validation dataset). Moreover, randomly choosing an operation in every layer acts as a regularizer in updating the weights during the training to minimize overfitting. These two advantages make DENSE suitable for learning to emulate a wide range of simulations even for expensive simulations where only a limited number of datasets can be generated.

While the simulations presented typically run in minutes to days, the DENSE emulators can process multiple sets of input parameters in milliseconds to a few seconds with one CPU core, or even faster when using a Titan X GPU card. For the GCM simulation which takes about 1150 CPU-hours to run, the emulator speedup is a factor of 110 million on a like-for-like basis, and over 2 billion with a GPU card. The speed up achieved by DENSE emulators for each test case is shown in Figure 2(h).

Compared with other non-deep learning techniques usually employed in building emulators,[24] the models found and trained by DENSE achieved the best results in all tested cases, and in most cases by a significant margin. As seen in Figures 2(g), the emulators built by DENSE achieved a loss function up to 14 times lower than the best performing non-deep learning model.

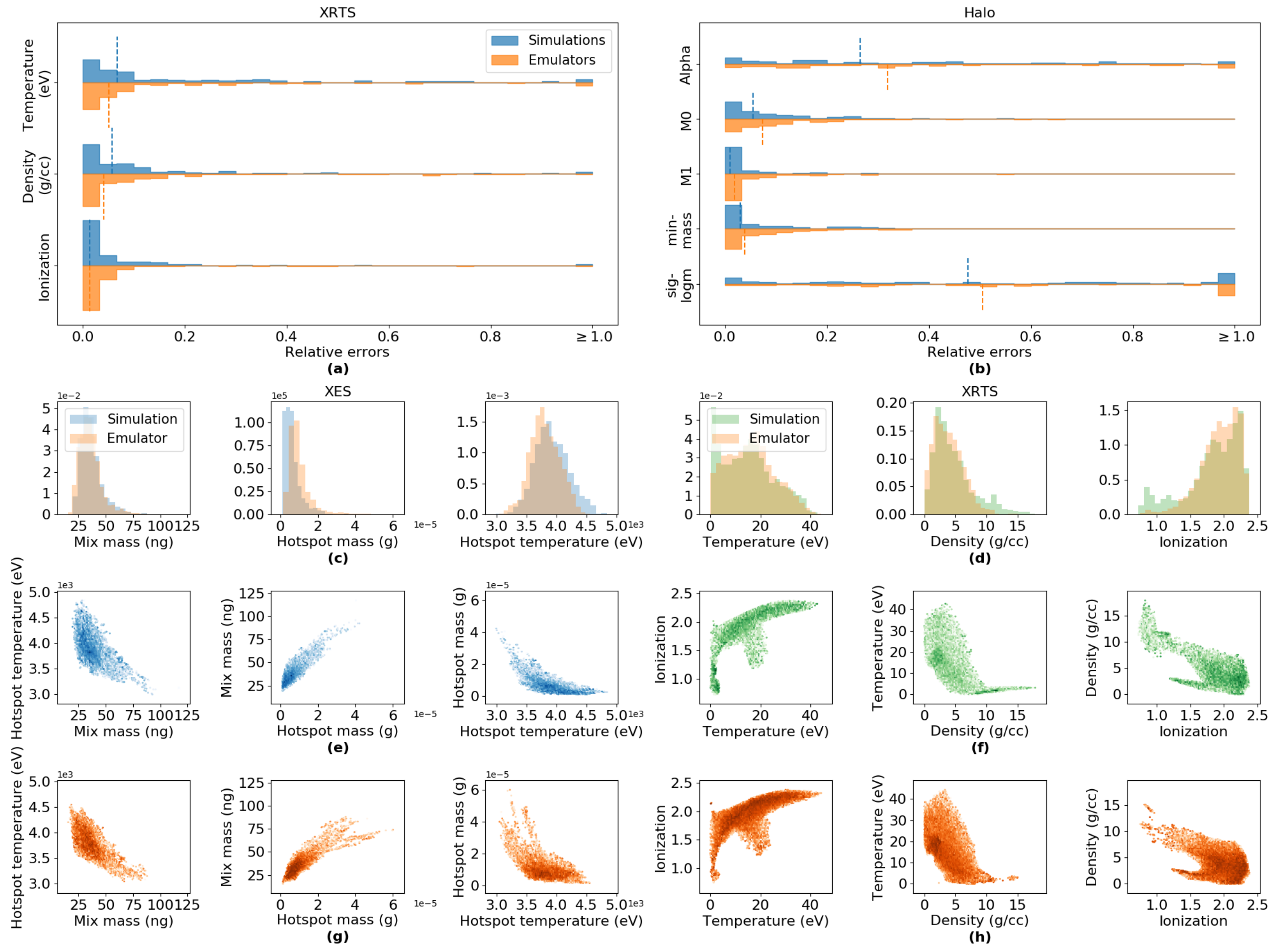

FIG. 3. (a,b) Histograms of the relative errors between the true parameters and the retrieved parameters in (a) XRTS and (b) Halo cases. The dashed lines indicate the median of the distributions. (c-h) Bayesian posterior sampling results using the emulators compared to the actual simulations for XES and XRTS cases. (c,d) The sampled histograms of the posterior distribution of parameters for both cases. (e,f) The scatter plots of the parameters posterior distribution obtained using the actual simulations and (g,h) the DENSE emulators.

We also compared DENSE with a manually-designed deep neural network model by an architecture from super-architecture in Figure 1 where all the convolutional layers have size 3. The use of kernel size 3 and skip connections follows the idea of ResNet.[10]

As we show in Figure 2(g), the DENSE emulators obtained better results than the manually-designed deep neural network in all cases except for ICF JAG Scalars where the results are comparable. It is worth noting that in some particularly challenging cases (GCM and OTS) the manually-designed deep neural network failed to learn the simulations at all, while DENSE achieved good results across the board and surpassed other methods in all cases but one. This illustrates the wide applicability of DENSE to build emulators for various cases.

B. Emulators for inverse problems

The high fidelity emulators built by DENSE are sufficiently accurate to allow us to substitute simulations even for more advanced tasks such as solving the inverse problem. To illustrate this, we took a simulated output signal randomly from the test dataset where the actual parameters are known. A small noise ($\sim 1\%$) was added to the chosen signal to closely mimic a real observed signal from an experiment. Using this signal, we use an optimization algorithm[25] to retrieve the input parameters by minimizing the error between the sample signal and the output of the emulators.

The results of the parameter retrieval using the emulators are compared with the retrieval using directly the simulations in Figures 3(a,b), where we plot the relative error histograms for the two cases. We observe that

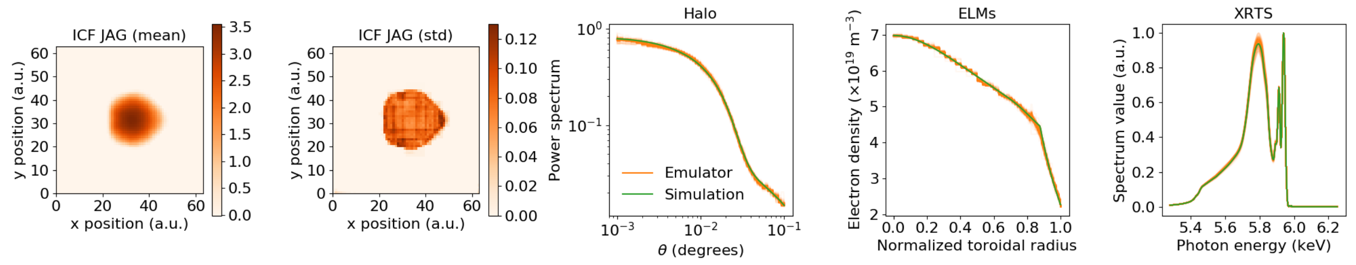

FIG. 4. Samples of the DENSE emulators output and the actual output from the simulations. From the samples, one can calculate the mean and uncertainty of the emulators output.

the relative errors from the emulators are very similar to those from the simulations.

Without losing much accuracy, the parameter retrieval with the emulators only takes about 800 ms with a single GPU card. This is to be compared with using the actual simulations which could take up to 2 days (XES) even when using 32 CPU cores. As the parameters can be retrieved in less than one second rather than in hours or days, one can envisage employing this technique for online diagnostics, real-time data interpretation, or even on-the-fly intelligent automation with an accuracy comparable to high-fidelity simulations that are by far too computationally expensive to be used directly. The use of DENSE emulators also enables parameters retrieval with resource-intensive simulations, such as MOPS and GCM, that were too expensive before.

In addition to interpreting signals and parameter retrieval, the emulators can also be used to significantly speed up modelling uncertainty. Uncertainty quantification is usually done with Bayesian posterior sampling by employing Markov Chain Monte Carlo (MCMC) algorithms. However, the cost of running MCMC to collect sufficient samples for an uncertainty analysis is typically much larger than the cost for parameter retrieval, and is often intractable in practice. Here we perform the Bayesian posterior sampling using an ensemble MCMC algorithm[26] with the same conditions as in ref.[27] In short, we collect all parameters sets that produces spectra that lie in a certain band around a central spectrum.

Figure 3(c-h) compares the results of sampling the parameter space using simulations and emulators in two cases to interpret scattering and spectroscopy data. The sampling histograms by the emulators are very similar to those by actual simulations, and we see that the emulators are well-suited to capture the correlations between parameters. However, note that while collecting 200,000 XES samples via simulations takes over 22 days, the sampling process with the emulators was completed in just a few seconds. Interestingly, building the emulator for XES from scratch only needs some 14,000 samples and 8 hours for training, so building the emulator and using it for MCMC is still considerably faster than directly collecting 200,000 samples from the original simulation.

C. Prediction uncertainty

A final important advantage of building emulators with DENSE is the availability of an intrinsic estimator of emulator uncertainty for free. The randomization of network architectures from the super-architecture can be seen as a special case of dropout.[28] Thus, by adapting the theory of prediction uncertainty with dropout by ref.,[29] we can show that DENSE emulators can produce the uncertainty of their outputs. The expected value and variance of a DENSE emulator prediction can be obtained by

$$\begin{aligned}\mathbb{E}(\mathbf{y}|\mathbf{x}) &= \mathbb{E}_{a\sim\mathcal{A}(\mathbf{b})}\left(\mathbf{y}|\mathbf{x},a\right)\\ \mathrm{Var}(\mathbf{y}|\mathbf{x}) &= \mathrm{Var}_{a\sim\mathcal{A}(\mathbf{b})}\left(\mathbf{y}|\mathbf{x},a\right),\end{aligned} \tag{3}$$

where a is the architecture sampled from the super-architecture $\mathcal{A}$ based on the final values of the network parameters, $\mathbf{b}$. Figure 4 shows the predicted uncertainties of the DENSE emulators, illustrating regions where they are either uncertain or confident in their predictions.

III. DISCUSSIONS

We have shown that Deep Emulator Network SEarch (DENSE), a method based on neural architecture search, can be used to robustly build fast and accurate emulators for various types of scientific simulations even with limited number of data. The ability of DENSE to accurately emulate simulations with limited number of data makes the acceleration of very expensive simulations possible. With the achieved acceleration of up to 2 billion times, DENSE emulators enable tasks that were impossible before, such as real-time simulation-based diagnostics, uncertainty quantification, and extensive parameters exploration. This large acceleration in solving inverse problems removes the barriers of using high fidelity simulations in real-time measurements, opening up new types of online diagnostics in the future. The wide range of successful test cases presented here shows the generality of the method in speeding up simulations, enabling rapid ideas testing and accelerating new discovery across the sciences and engineering.

IV. METHODS

A. Test cases

Here we provide a description of the test cases employed in the paper. A summary of the test case parameters is given in Table I.

X-ray Thomson scattering (XRTS): XRTS is a technique widely used in high-energy-density physics to extract plasma temperatures and densities by measuring the spectrum of an inelastically scattered x-ray pulse.[2,30] The spectrum of the scattered light can be calculated from a set of plasma conditions and the scattering geometry;[14] this forms the simulation on which our emulator is based.

In this paper we consider the specific experimental case presented in ref.[2] where three parameters (temperature, ionization, and density) are to be retrieved from a spectrum of x-rays scattered at a 90-degree angle from a shock-compressed Beryllium plasma. The high-speed emulator for XRTS enables fast solutions to the inverse problem and access to statistical information on the intrinsic uncertainty of the experiment, allowing better control of the experimental optimization and information extraction.

Optical Thomson scattering (OTS): OTS is conceptually similar to XRTS except that it uses optical light instead of x-rays. Optical Thomson scattering is used in measuring electrons and ions temperatures and densities, as well as the flow speed of the plasma using the Doppler shift.[15]

Here we considered retrieving five physical parameters (electron and ion temperatures, electron density, ionization, and flow speed) from a normalized scattered spectrum. The impact of building an emulator for OTS is similar to XRTS as it enables access to real-time data interpretation and to uncertainty quantification.

X-ray emission spectroscopy (XES): X-ray emission spectroscopy is a general technique to probe a system by measuring the emitted spectrum and matching it with simulations or theoretical models. In this paper, we consider the diagnostic case of a laser-driven implosion experiment at the National Ignition Facility,[16] using the spectroscopic model based on the CRETIN atomic kinetics code described in detail in ref.[17]

Edge-Localized Modes (ELMs) diagnostics: Edge-localized modes are magnetohydrodynamic instabilities that occur in magnetically confined fusion plasmas with high confinement.[31] ELMs are explosive events and cause detrimental heat and particle loads on the plasma facing components of a tokamak. Various diagnostics are implemented to track ELMs.[32] Here, we compare the emulator to the predictive model[33] for the temporal evolution of the electron density profile using the transport code ASTRA.[34]

The 14 input parameters in this case describe the diffusion, convective velocity, and particle source profiles[35] as a function of toroidal radial position and time. The output observable is the time-dependent electron density as a function of toroidal radial position.

Galaxy halo occupation distribution modelling (Halo): Here we considered simulations of the angular-scale correlation of a galaxy population. The simulation software Halomod[36] was used to calculate the correlation function, angular scales, redshifts and the cosmological model as described in ref.[18] Fast parameter retrieval is of particular interest here as often researchers are interested in extracting parameters for multiple different galaxy populations.

JAG model for Inertial Confinement Fusion (ICF JAG): JAG model simulates the observables from an inertial confinement fusion experiment.[22] There are 5 input parameters in the considered case. One simulation with 5 input parameters produces four two-dimensional images and 15 scalar values. Constructing fast and accurate emulators of the model allows for a more efficient exploration of the parameters space, and to obtain optimum sets of parameters more efficiently.

Shatsky Rise seismic tomography (SeisTomo): The case considered here is seismic tomographic inversion problem of Shatsky Rise oceanic plateau.[19] Given the input parameters that describe the initial velocity profile and regularization in the optimization, the software solves for the velocity structure and the crustal thicknesses as a function of position in the Shatsky Rise that matches the seismic reflection data. Performing uncertainty quantification of the tomographic inversion would require the execution of the software hundreds of thousand times which is very expensive without a fast emulator model.

Global aerosol-climate modelling (GCM): The model considered here is ECHAM-HAM[20] which calculates the distribution and evolution of both internally and externally mixed aerosol species in the atmosphere and their affect on both radiation and cloud processes. The model simulates the aerosol absorption optical depth as the observable. The cost of running the model for one year (including three months of spin-up) is about 1150 CPU-hours which is prohibitively expensive when generating thousands of training data points. However, we have shown that an accurate emulator over three parameters can be built with as few as 39 data points.

The Model of Oceanic Pelagic Stoichiometry (MOPS): The Model of Oceanic Pelagic Stoichiometry (MOPS) is a global ocean biogeochemical model[37] that simulates the cycling of nutrients (i.e., phosphorus, nitrogen), phytoplankton, zooplankton, dissolved oxygen and dissolved inorganic carbon. MOPS is coupled to the Transport Matrix Method (TMM), a computational framework for efficient advective-diffusive transport of ocean biogeochemical tracers.[21,38] In this study we use monthly mean transport matrices derived from a configuration of MITgcm[39] with a horizontal resolution of 2.8° and 15 vertical levels. There are 6 MOPS input parameters considered in this case, whose definitions and ranges are described in an optimization study by ref.[40]

No.	Test case	# Inputs	# Outputs	Output type	# Dataset	Est. running time
1	XRTS	3	1	1D (250 points)	14,000	15 seconds
2	OTS	5	1	1D (250 points)	14,000	15 seconds
3	XES	10	1	1D (250 points)	14,000	20 minutes
4	ELMs	14	10	1D (250 points)	14,000	15 minutes
5	Halo	5	1	1D (250 points)	14,000	5 seconds
6	ICF JAG	5	4	2D (64 × 64)	10,000	30 seconds
7	ICF JAG Scalars	5	15	0D (scalar)	10,000	30 seconds
8	SeisTomo	13	1	1D (250 points)	6,100	2 hours
9	MOPS	6	45	2D (128 × 64)	410	144 CPU-hours
10	GCM	3	12	2D (192 × 96)	39	1150 CPU-hours

TABLE I. Summary of test cases considered in this paper

Each simulation involves integrating the model for 3000 years to a steady state starting from a uniform spatial distribution of tracers. Annual mean 3D fields of oxygen, phosphorus, and nitrate at the end of the simulation are used for training. All code and relevant data used for the simulations are freely available.[38]

---

* muhammad.kasim@physics.ox.ac.uk

[1] Jeff Greeley, Thomas F Jaramillo, Jacob Bonde, IB Chorkendorff, and Jens K Nørskov. Computational high-throughput screening of electrocatalytic materials for hydrogen evolution. *Nature materials*, 5(11):909–913, 2006.

[2] HJ Lee, P Neumayer, J Castor, T Döppner, RW Falcone, C Fortmann, BA Hammel, AL Kritcher, OL Landen, RW Lee, et al. X-ray thomson-scattering measurements of density and temperature in shock-compressed beryllium. *Physical review letters*, 102(11):115001, 2009.

[3] J Galdon-Quiroga, Manuel Garcia-Munoz, KG McClements, M Nocente, M Hoelzl, AS Jacobsen, F Orain, JF Rivero-Rodriguez, Mirko Salewski, L Sanchis-Sanchez, et al. Beam-ion acceleration during edge localized modes in the asdex upgrade tokamak. *Physical review letters*, 121(2):025002, 2018.

[4] JL Peterson, KD Humbird, JE Field, ST Brandon, SH Langer, RC Nora, BK Spears, and PT Springer. Zonal flow generation in inertial confinement fusion implosions. *Physics of Plasmas*, 24(3):032702, 2017.

[5] Juliana Kwan, Katrin Heitmann, Salman Habib, Nikhil Padmanabhan, Earl Lawrence, Hal Finkel, Nicholas Frontiere, and Adrian Pope. Cosmic emulation: fast predictions for the galaxy power spectrum. *The Astrophysical Journal*, 810(1):35, 2015.

[6] Felix Brockherde, Leslie Vogt, Li Li, Mark E Tuckerman, Kieron Burke, and Klaus-Robert Müller. Bypassing the kohn-sham equations with machine learning. *Nature communications*, 8(1):872, 2017.

[7] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

[8] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[9] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[12] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[13] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[14] G Gregori, Siegfried H Glenzer, W Rozmus, RW Lee, and OL Landen. Theoretical model of x-ray scattering as a dense matter probe. *Physical Review E*, 67(2):026412, 2003.

[15] P Tzeferacos, A Rigby, AFA Bott, AR Bell, R Bingham, A Casner, F Cattaneo, EM Churazov, J Emig, F Fiuza, et al. Laboratory evidence of dynamo amplification of magnetic fields in a turbulent plasma. *Nature communications*, 9(1):591, 2018.

[16] SP Regan, R Epstein, BA Hammel, LJ Suter, HA Scott, MA Barrios, DK Bradley, DA Callahan, C Cerjan, GW Collins, et al. Hot-spot mix in ignition-scale inertial confinement fusion targets. *Physical review letters*, 111(4):045001, 2013.

[17] Orlando Ciricosta, H Scott, P Durey, BA Hammel, R Epstein, TR Preston, SP Regan, SM Vinko, NC Woolsey, and JS Wark. Simultaneous diagnosis of radial profiles and mix in nif ignition-scale implosions via x-ray spectroscopy. *Physics of Plasmas*, 24(11):112703, 2017.

[18] PW Hatfield, SN Lindsay, MJ Jarvis, B Häußler, M Vaccari, and A Verma. The galaxy–halo connection in the video survey at 0.5¡ z¡ 1.7. *Monthly Notices of the Royal Astronomical Society*, 459(3):2618–2631, 2016.

[19] J Korenaga and WW Sager. Seismic tomography of shatsky rise by adaptive importance sampling. *Journal of Geophysical Research: Solid Earth*, 117(B8), 2012.

[20] Ina Tegen, David Neubauer, Sylvaine Ferrachat, Siegenthaler-Le Drian, Isabelle Bey, Nick Schutgens,

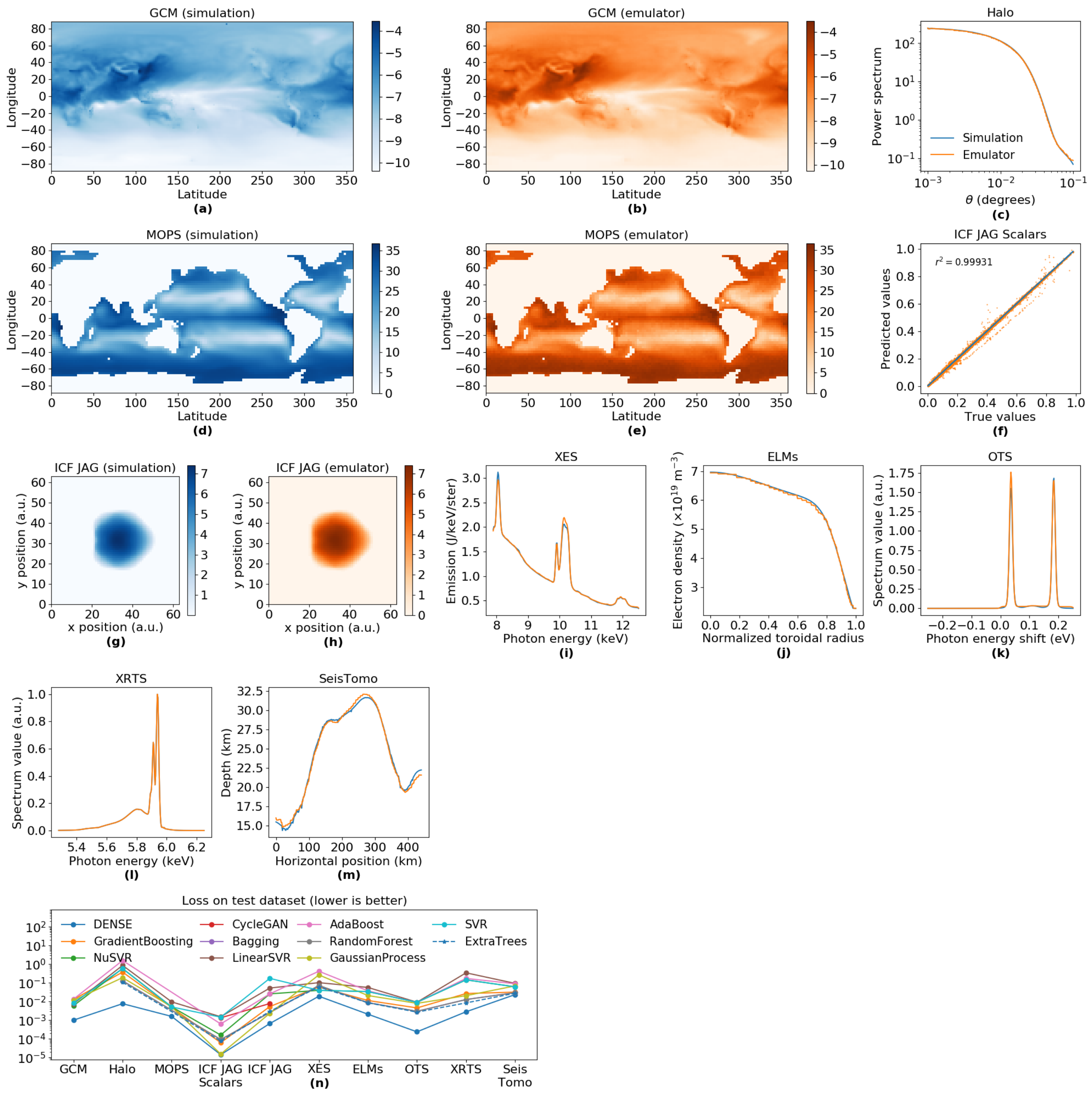

FIG. 5. (a-m) Examples of emulator outputs for all test cases. (n) The results of emulators built by DENSE and other non-deep learning methods.

Philip Stier, Duncan Watson-Parris, Tanja Stanelle, Hauke Schmidt, et al. The global aerosol-climate model echam6. 3-ham2. 3-part 1: Aerosol evaluation. *Geoscientific Model Development*, 12(4):1643–1677, 2019.

[21] Samar Khatiwala. A computational framework for simulation of biogeochemical tracers in the ocean. *Global Biogeochemical Cycles*, 21(3), 2007.

[22] Rushil Anirudh, Peer-Timo Bremer, Jayaraman Jayaraman Thiagrarjan, and USDOE National Nuclear Security Administration. Cycle consistent surrogate for inertial confinement fusion. 2 2019.

[23] Ilya Loshchilov and Frank Hutter. Cma-es for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*, 2016.

[24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[25] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen

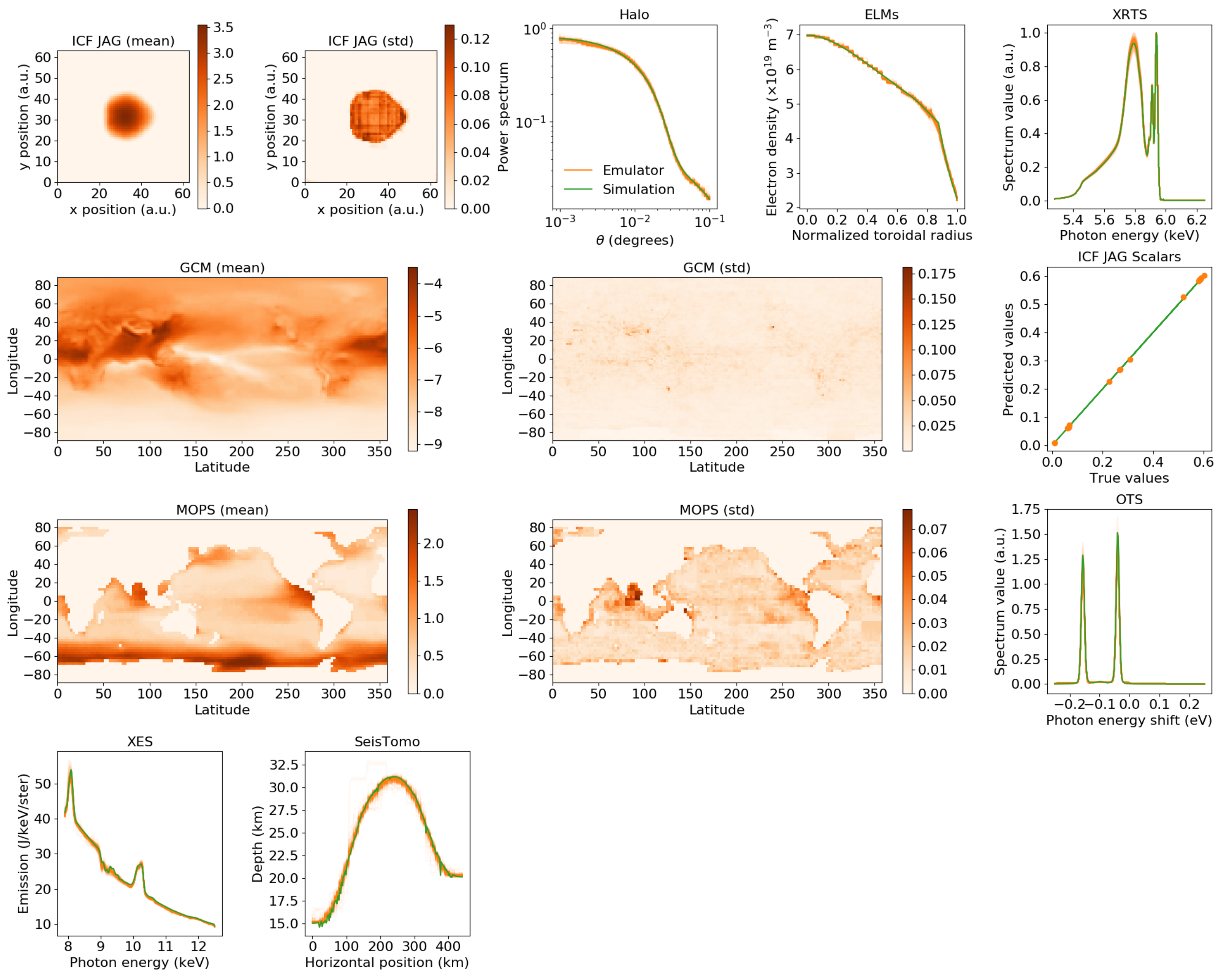

FIG. 6. Examples of emulator uncertainties for all test cases.

Schmidhuber. Natural evolution strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3381–3387. IEEE, 2008.

[26] Jonathan Goodman and Jonathan Weare. Ensemble samplers with affine invariance. *Communications in applied mathematics and computational science*, 5(1):65–80, 2010.

[27] MF Kasim, TP Galligan, J Topp-Mugglestone, G Gregori, and SM Vinko. Inverse problem instabilities in large-scale modeling of matter in extreme conditions. *Physics of Plasmas*, 26(11):112706, 2019.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[29] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[30] Andrea L Kritcher, Paul Neumayer, John Castor, Tilo Döppner, Roger W Falcone, Otto L Landen, Hae Ja Lee, Richard W Lee, Edward C Morse, Andrew Ng, et al. Ultrafast x-ray thomson scattering of shock-compressed matter. *Science*, 322(5898):69–71, 2008.

[31] Hartmut Zohm. Edge localized modes (elms). *Plasma Physics and Controlled Fusion*, 38(2):105, 1996.

[32] M Cavedon, T Pütterich, Eleonora Viezzer, FM Laggner, A Burckhart, M Dunne, R Fischer, A Lebschy, F Mink, U Stroth, et al. Pedestal and e r profile evolution during an edge localized mode cycle at asdex upgrade. *Plasma Physics and Controlled Fusion*, 59(10):105007, 2017.

[33] E Viezzer, M Cavedon, E Fable, FM Laggner, RM McDermott, J Galdon-Quiroga, MG Dunne, A Kappatou, C Angioni, P Cano-Megias, et al. Ion heat transport dynamics during edge localized mode cycles at asdex upgrade. *Nuclear Fusion*, 58(2):026031, 2018.

[34] E Fable, C Angioni, FJ Casson, D Told, AA Ivanov, F Jenko, RM McDermott, S Yu Medvedev, GV Pereverzev, F Ryter, et al. Novel free-boundary equilibrium and transport solver with theory-based models and its validation

against asdex upgrade current ramp scenarios. *Plasma Physics and Controlled Fusion*, 55(12):124028, 2013.

[35] M Willensdorfer, E Fable, E Wolfrum, Leena Aho-Mantila, F Aumayr, R Fischer, F Reimold, F Ryter, et al. Particle transport analysis of the density build-up after the l–h transition in asdex upgrade. *Nuclear Fusion*, 53(9):093020, 2013.

[36] Steven Murray. halomod: Python package for dealing with the Halo Model, June 2017.

[37] Iris Kriest and Andreas Oschlies. Mops-1.0: modelling the regulation of the global oceanic nitrogen budget by marine biogeochemical processes. *Geoscientific Model Development*, 8:2929–2957, 2015.

[38] samarkhatiwala. samarkhatiwala/tmm: Version 2.0 of the transport matrix method software, May 2018.

[39] John Marshall, Alistair Adcroft, Chris Hill, Lev Perelman, and Curt Heisey. A finite-volume, incompressible navier stokes model for studies of the ocean on parallel computers. *Journal of Geophysical Research: Oceans*, 102(C3):5753–5766, 1997.

[40] Iris Kriest, Volkmar Sauerland, Samar Khatiwala, Anand Srivastav, and Andreas Oschlies. Calibrating a global three-dimensional biogeochemical ocean model (mops-1.0). *Geoscientific Model Development*, 10:127–154, 2017.