# GPU Acceleration for Informative Ensemble Kalman Learning at Scale

Thelonious Cooper

We present a novel method for the GPU acceleration of joint neural architecture search (NAS) and optimization based on an Informative Ensemble Kalman Learner (IEKL) which represents a significant improvement over state-of-the-art for many models. This approach constructs empirical distributions over parameters via a Gaussian Process formulation. The mean and covariance of parallel forward passes form an accurate estimate of the model Jacobian with no need to track gradients or backpropagate, all while quantifying uncertainty. The construction of empirical distributions, combined with the assumption of local gaussianity enable a new paradigm of Informative Learning. In this paradigm the approximate mutual information of parameters with respect to model error can be readily computed and used to identify dead weights. Further, information gain can be used to inductively build parsimonious models from data. We implement the methods in PyTorch for easy portability and gpu-acceleration. The method is numeric-format agnostic and can be extended to small integer formats in order to reduce the amount of memory needed for important models to that of common microprocessors.

## I. INTRODUCTION

When making a data-driven mathematical model it is important to first understand what you want from it. If you ask an undergraduate what they want from such a model, they will say "accuracy". If you ask a graduate student the same question they may also say "precision" If prodded further they may also add niceties such as "computational tractability" or "mechanistic interpretability." If you ask a professor, they will likely go on a diatribe about "generalizability" that could not be summarized in the page limit of this article.

Unfortunately, the majority of machine learning models provide only a small subset of all the things we'd like from them. Despite continual progress in modeling capabilities, many of the limitations of modern techniques in application are driven not by the nature of the models, but by how they are trained.

Linear and readily linearizable models have long been the cornerstone of modeling and control due to their simplicity and tractability. The current best practice in nonlinear optimization employs backpropagation [8] or one of its more sophisticated cousins such as Adam[3] or, more recently, Lion[1]. These procedures involve a forward pass and a backward pass. The forward pass runs a model on a set of training data, storing gradients at each step. The backward pass then uses these gradients to simplify the model and minimize it around the operating point. But when confronted with nondifferentiable functions, the classical approach faces difficulties.

In general, "learning" can be seen as a two-point boundary value problem, where one attempts to find a path in parameter space starting with an initial guess towards a path such that the model outputs align with known data. These types of problems crop up across many domains of science and engineering. As has been discussed in our group's prior work [9] [5], Backprop is but one of many solutions to this problem that engineers and scientists have developed.

Backpropagation has many desirable qualities, but amongst its greatest failings is its inability to quantify uncertainty in its parameter estimates. In important high-risk scenarios, a model with no measure of its own confidence is almost immediately disqualified. When making decisions based on model predictions, it is crucial to understand how uncertainties in one's model structure and parameters propagate to uncertainty in forecasts is crucial.

Strides made by our research group in information theory and stochastic methods allow us to admit a stochastic formulation of learning and reap all the benefits that come from it. The method we present in this paper can learn in black-box settings, with no constraints on differentiability or even continuity. By bypassing these constraints, we admit learning to whole new classes of mathematical objects such as jump processes and stochastic differential equations which have powerful modeling abilities. This is achieved by examining nonlinear relationships from an information-theoretic perspective.

By employing our method, models gain many desirable traits for "free." Simply by replacing 3 lines of code in a python implementation, structured sparsity, uncertainty quantification, and robust optimization follow shortly. Uncertainty quantification offers a method for accurately assessing the precision of models. Structured sparsity can increase the computational tractibility of a large model. The robustness of the optimization also increases the model's ability to climb out of local minima and generalize to unseen or out-of-distribution data more readily.

Beyond mathematical elegance, our method is of great interest to ML practitioners for its ability to promote sparsity in models, drastically reducing the computational resources needed to employ them. By estimating the information gains associated with changes in parameters we can identify and eliminate unnecessary weights in both large and small models, reducing their complexity. This approach is more methodologically sound than traditional magnitude-based pruning methods[10].

Our new approach uses an ensemble of models to make forward predictions. These models can be perturbed in terms of state, parameter, or control input, depending on

the problem at hand. The spread of model prediction trajectories, along with the precision of goal attainment or measurement noise in estimation cases, provides the gain terms needed for backward calculations. This eliminates the need for explicit calculation of the model adjoint.

The Ensemble Kalman Filter and Smoother technique [2], which forms a Gaussian process implemented in reduced-rank square-root form, has been applied to numerous problems with nonlinear forward system dynamics. It is particularly popular in the geosciences where it is often used for data assimilation from multiple sensor sources [4]. Its benefits extend beyond estimation to control, autonomy, learning, and inference. In prior work our group has developed quadcopter controllers based on this technique [11].

For inference problems, we have extending the ensemble approach to handle fully non-Gaussian posterior uncertainties. This involves using information-theoretic cost functions and closely related Variational Bayes methods. These extensions are being applied to seismic monitoring problems, such as locating small earthquakes and nuclear explosions.

From a computational perspective, we see promise in developing new architectures that enable additional optimizations of the ensemble approach. Beyond current work, we are also excited that this technique applies to the variational information-theoretic non-Gaussian setting. By creating a platform for fast ensemble filtering models, the project will have extensions to many problem domains and could become a general tool for future projects.

We believe that the availability of GPU-enabled embedded processors enables the development of a new computational paradigm for ensemble-based estimation, control, and learning. This can be applied in various applications, including but not limited to autonomy. Although Informative Ensemble Kalman Learning (IEKL) has been tested in simulated experiments [11] [9] [7], this paper represents the first development of a scalable and open-source implementation.

## II. METHODS

### A. Ensemble Kalman Learning

Starting from a neural model with Gaussian label noise and parameter perturbation term

$$\hat{y} = N(x; \alpha + \tilde{\alpha}) \tag{1}$$

$$\hat{y} = N(x; \alpha) + \mathcal{G}\tilde{\alpha} \tag{2}$$

we extract the estimate $\hat{y}$ into label variation and mean to obtain

$$\mathcal{G}\tilde{\alpha} = \tilde{\hat{y}} + \bar{\hat{y}} - N(x; \alpha) \tag{3}$$

$\mathcal{G}$ can then be estimated as

$$\mathcal{G}\mathbb{E}[\tilde{\alpha}\tilde{\alpha}^T] = \mathbb{E}[(\tilde{\hat{y}} + \delta\hat{y})\tilde{\alpha}^T] \tag{4}$$

Because the label noise and parameter perturbations are uncorrelated the expectations are the associated cross covariances. Thus the forward Guassian process is

$$\mathcal{G} = C_{\alpha\hat{y}} C_{\alpha\alpha}^{-1} \tag{5}$$

Note that the computation of $C_{\alpha\hat{y}}$ involves the forward pass of the model and projection into the observation domain. The adjoint Guassian process is then

$$\mathcal{G} = C_{\alpha\hat{y}} {C_{\hat{y}\hat{y}}}^{-1} \tag{6}$$

Considering the performance objective

$$J(\alpha|y) = \frac{1}{2}(\delta\hat{y}^T {C_{zz}}^{-1}\delta\hat{y} + \delta\hat{\alpha}^T {C_{\alpha\alpha}}^{-1}\delta\hat{\alpha}) \tag{7}$$

The solution to the minimization problem

$$\alpha^+ = \arg\min_{\alpha} J(\alpha|y) \tag{8}$$

is as follows [11]

$$\hat{\alpha}^+ = \hat{\alpha} + C_{\hat{\alpha}\hat{y}}(C_{\hat{y}\hat{y}} + C_{yy})^{-1}(y - \hat{y}) \tag{9}$$

Using the ensemble approximation with $E$ ensemble members, we can write this in matrix form where each realization of the random vector is a column. We adopt the notation that $\tilde{A}$ is $A - \mathbb{E}[A]$. Under this scheme the cross covariance of two ensembles is

$$C_{AB} = \frac{\tilde{A}\tilde{B}^T}{E-1} \tag{10}$$

Next we can compute the singular value decomposition of the prediction deviations $U\Sigma V^T = \tilde{\hat{Y}}$ and carry out the ensemble update in square-root form without explicitly constructing the cross covariance matrix via the following equation.

$$\hat{\alpha}^+ = \hat{\alpha}[\mathbb{I}_E + V\Sigma(\Sigma^2 + \rho^2\mathbb{I}_E)^{-1}U^T(\hat{Y} - y)] \tag{11}$$

Avoiding explicitly storing the cross covariance matrix is crucial because it allows us to tackle very high dimensional problems at their true rank, as opposed to at the dimensionality of the observation. We expect that even in very high dimensional systems, there are only few significant components. By identifying and optimizing over these high-impact factors, we can get more efficiency in the learner or controller system. This notion of greedy informative optimization / green control as implemented by this method is discussed at length in prior work [11].

### B. Observational Noise Scheduling

A principle challenge with ensemble methods is the possibility of *ensemble collapse*. As the optimizer reaches convergence the standard deviation of the parameter ensemble can become quite small. If the deviation of the resulting forward passes also becomes small, the observation ensemble matrix can become ill-conditioned, causing the singular value decomposition to fail. In order to compensate for this we introduce zero-mean observational noise with a deviation corresponding to the performance target variable $\rho$. As training progresses, we scale down the performance target by a fixed annealing factor every time the maximum of the parameter variances dips below the value of the performance target. This annealing rate is important, as having slow annealing rates limits convergence, and having fast annealing rates can lead to spontaneous collapse. In our experiments we use an annealing factor of 0.75 across all experiments.

Another strategy to combat ensemble collapse is dynamic weight resampling. For parameters with a variance across the ensemble too low for the numeric precision of our algorithm, we can identify the ensemble mean and variance and resample the distribution from a normal distribution with the same mean and higher variance, enforcing a diverse ensemble.

### C. Informative Structural Adaptation

A major benefit of the ensemble method is that it lends itself to information theoretic approaches. By assuming the conditional expectation of the error with respect to the parameters is Gaussian, we can obtain simple linear algebraic equations to quantify mutual information between ensembles. For structure adaptation we adopt a greedy informative approach. Given parameter ensemble $\hat{\alpha}$ we can compute the mutual information with the model error $e$ via the following formula:

$$I(\hat{\alpha}; e) = -\frac{1}{2} \log \left[1 - \left(\frac{C\hat{\alpha}e}{\Sigma_{\hat{\alpha}} \Sigma_e^T}\right)^2\right] \tag{12}$$

This gives us the mutual information between each error dimension and each parameter. We take the $l_1$ norm with respect to the error dimensions so we have a 1-to-1 correspondence between parameters and information. We then set some number of the least informative parameters to 0 and fine tune the remaining parameters until convergence. By repeating this process we can achieve high sparsity rates in models. For fully connected models we can go one step further, pruning "dead" nodes that have no incoming weights and propagating their biases to the biases of the next layer. This process is demonstrated in experiment 2**??**

As alluded to at earlier at the end of subsection [**?** ], informative optimization provides a positive feedback loop which increases efficiency. By identifying unimportant e.g. low mutual information components of the optimization space, the space can be traversed more efficiently. This then makes it less computationally intensive to identify further components of minimal contributions. This cycle can be repeated until only the parameters contributing to the true rank of the system are left as demonstrated in experiment 2 **??**.

## III. IMPLEMENTATION OF GPU ACCELERATION

Much of the engineering challenge of the implementation comes in the form of organizing available GPUs in order to execute $E$ forward passes of the model in as little time as possible. Ideally all ensemble member forward calculations would take place in parallel, but due to VRAM constraints it is necessary to limit the number of executions happening to the number of models whose parameters can fit on a set of GPUs. In order to maximize the throughput of the forward passes we used a pipelined multi-gpu approach.

Taking advantage of PyTorch's [6] asynchronous GPU execution model and vectorized mapping ability, our implementation dispatches computation requests to each GPU in chunks. Each GPU is given the next chunk of inputs and parameters as soon as it completes its previous chunk without waiting for other GPUs to halt. Extensive use of PyTorch's VMAP function with statically allocated buffers is used to take full advantage of each GPU by running multiple forward models per GPU call dispatch. The complete architecture is depicted in Figure 1 1 Care was taken to ensure support for many types of models at various scales. Our implementation supports training on single-GPU, or N-GPUs. The only constraint is that at least one model in its entirety must fit on a single host device. This is a limitation of the implementation, however, not the method.

In order to reduce the computational burden and memory consumption of the ensemble update step, we split the weights into sections which each get an ensemble update with respect to only the weights within the section. This assumes that the covariance matrix can be block partitioned without losing significant rank. We find this to be a practical assumption which implies that the weights at the start of a network have little mutual information with those at the end of the network. This is the empirical observation that underlies foundation models with arbitrary fine-tuned heads.

## IV. EXPERIMENTS

### A. Convergence Comparison

To test our methodology we start small with a 600k parameter LeNet model. In 2 we compare AdamW, the Ensemble Kalman approach, and a hyrbid alternating

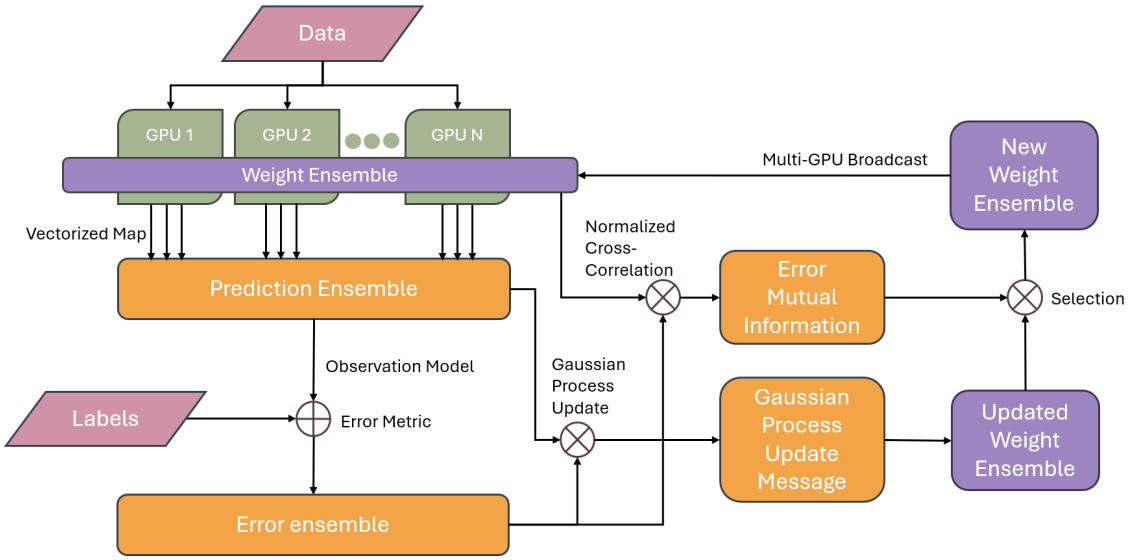



Figure 1: Software architecture of IEKL update procedure

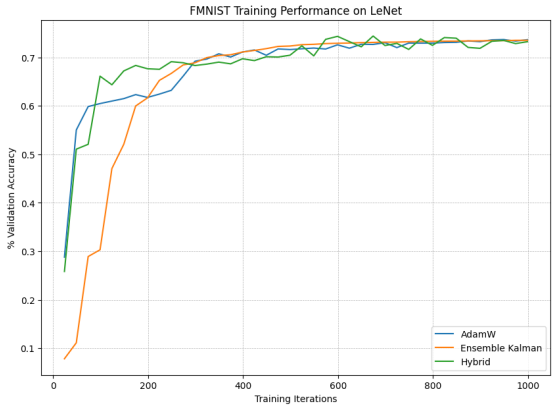



Figure 2: Training LeNet on FashionMNIST with AdamW, IEKL, and a hybrid approach alternating AdamW and IEKL updates

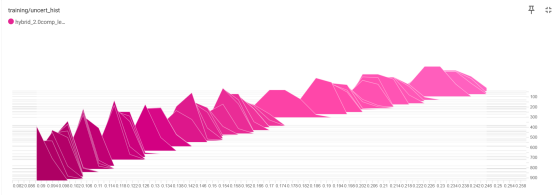


Figure 3: Histograms of parameter variances shrinking over training iterations as uncertainty decreases.

methodology. We then visualize the reduction in model uncertainty over the learning process in3.

Default hyperparameters for each optimizer were used.

### B. Unstructured Model Sparsification

By training LeNet on Fashion-MNIST with multiple levels of target informative sparsification, we breach the Pareto frontier of model parameter efficiency in CNNs.

### C. Reduced Order Modeling

In **??** we demonstrate that iteratively training and pruning a much larger student model against a small teacher model can yield a smaller model that has indistinguishable outputs from the teacher.

## V. CONCLUSIONS AND FUTURE WORK

We develop a scalable implementation of the Ensemble Kalman Filter technique for machine learning optimization applications. We see this approach as having many important advantages over current methods such as Adam due to its ability to quantify uncertainty and promote sparsity. With the exception of models so large that they cannot fit on a single GPU, our methods represent the most effective and accessible way to jointly optimize structure and parameters.

Informative Ensemble Kalman Learning is a recent technique with wide applications. In our group alone, we apply the technique to seismic nuclear test monitoring, geothermal energy systems optimization, learning polynomial ordinary differential equations for hurricane trajectories, and training models for remote sensing of groundwater and salinity from satellite data. We can't wait to see new challenges we and others can face with this novel technique and implementation.

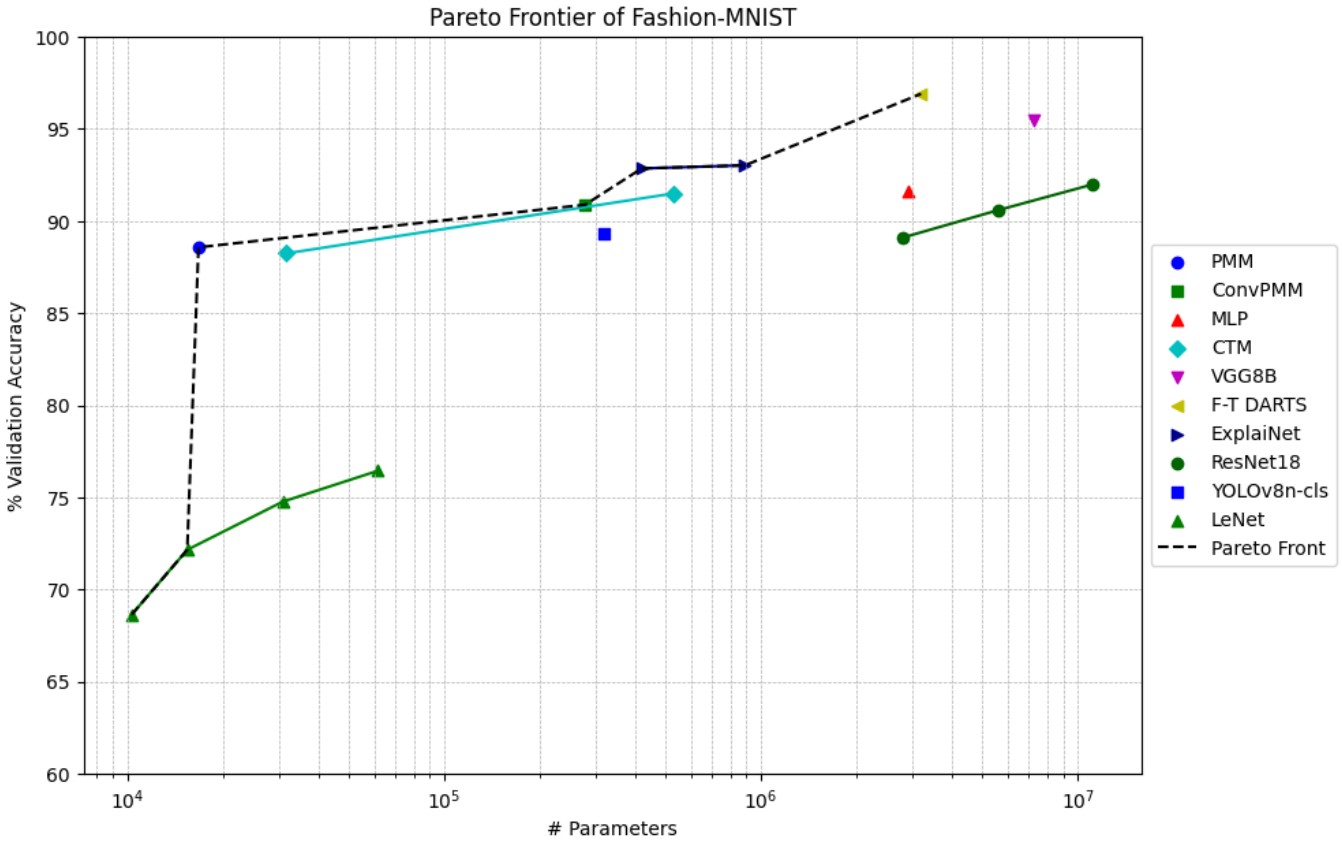



Figure 4: Despite having high performance, ResNet18 is clearly larger than necessary for the FasionMNIST. ResNet18 sparsified via Hyrbid IEKL loses performance as it shrinks, but does not approach the efficiency of natively smaller models.

In future work we seek to develop computational architectures to accelerate this technique and apply flexible computing media such as FPGAs to perform rapid inference and learning, perhaps even in real-time. The development of sparse-matrix operation accelerators provides an exciting insight into the future of computation. By exploiting information-theoretic measures of sparsity in large nonlinear models, we look towards a future where even models as large as ChatGPT can be securely and privately executed on embedded devices. The compression of model's nonzero-parameters can allow for us to scale to larger and deeper models without being bound by hardware memory and compute constraints.

## VI. CONTRIBUTIONS AND ACKNOWLEDGMENT

We thank the entire Earth Signals and Systems Group for their support and express appreciation to the MIT Climate Grand Challenges Fund for financial contributions.

[1] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, 2023.

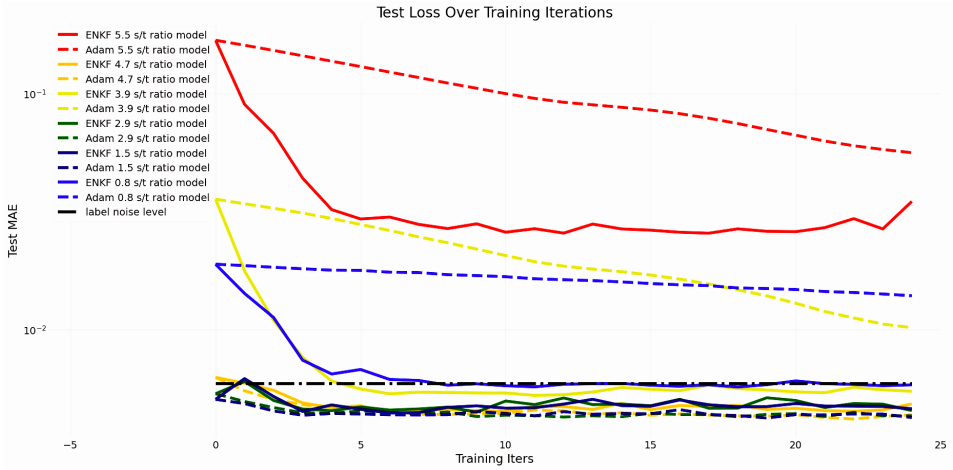
[2] Geir Evensen. The ensemble kalman filter: theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367, 2003.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego*, 2014.

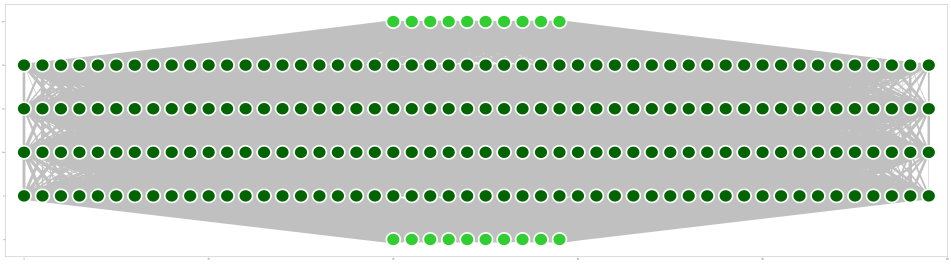
[4] Liangping Li, Ryan Puzel, and Arden Davis. Data assimilation in groundwater modelling: ensemble kalman filter versus ensemble smoothers. *Hydrological Processes*, 32(13):2020–2029, 2018.

[5] Ziwei Li and Sai Ravela. Neural networks as geometric chaotic maps. *IEEE Transactions on Neural Networks and Learning Systems*, 34(1):527–533, January 2023.

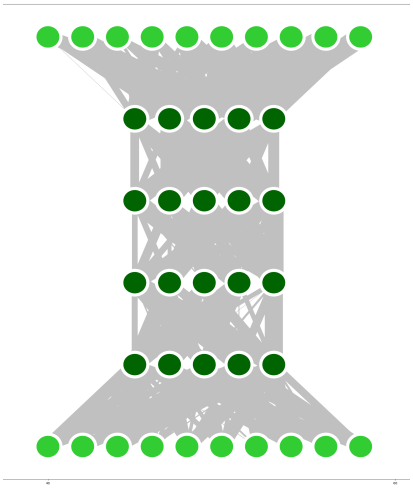
[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Des-



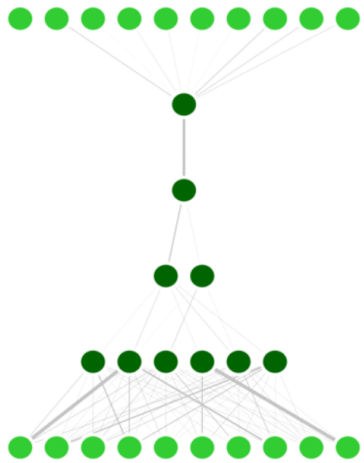

**(a) Comparison of Adam to ENKF optimizer after each structural adaptation**



**(b) Student**



**(c) Teacher**



**(d) Learned Model**

Figure 5: The white noise response of the teacher network is used to informatively optimize and sparsify the student network until convergence is reached. Subfigure (a) compares the performance of the current state-of-the-art optimizer, Adam, against ENKF at several stages of structural optimization. The student, teacher, and learned networks are visualized in subfigures (b), (c), and (d), respectively.

maison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019.

[7] S. Chandu Ravela and Dennis B. McLaughlin. Fast ensemble smoothing. *Ocean Dynamics*, 57:123–134, 2006.

[8] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[9] Margaret Trautner, Gabriel Margolis, and Sai Ravela. Informative neural ensemble kalman learning, 2020.

[10] Sunil Vadera and Salem Ameen. Methods for pruning deep neural networks, 2021.

[11] Erina Yamaguchi and Sai Ravela. Multirotor ensemble model predictive control i: Simulation experiments, 2023.