ARTICLE Communicated by Eric Mjolsness

Machine Learning: Deepest Learning as Statistical Data Assimilation Problems

Henry D. I. Abarbanel
habarbanel@ucsd.edu
Marine Physical Laboratory, Scripps Institution of Oceanography, and Department of Physics, University of California, San Diego, La Jolla, CA 92093-0374, U.S.A.

Paul J. Rozdeba
paul.rozdeba@uni-potsdam.de
Sasha Shirman
ashirman@physics.ucsd.edu
Department of Physics, University of California, San Diego, La Jolla, CA 92093-0374, U.S.A.

We formulate an equivalence between machine learning and the formulation of statistical data assimilation as used widely in physical and biological sciences. The correspondence is that layer number in a feedforward artificial network setting is the analog of time in the data assimilation setting. This connection has been noted in the machine learning literature. We add a perspective that expands on how methods from statistical physics and aspects of Lagrangian and Hamiltonian dynamics play a role in how networks can be trained and designed. Within the discussion of this equivalence, we show that adding more layers (making the network deeper) is analogous to adding temporal resolution in a data assimilation framework. Extending this equivalence to recurrent networks is also discussed.

We explore how one can find a candidate for the global minimum of the cost functions in the machine learning context using a method from data assimilation. Calculations on simple models from both sides of the equivalence are reported.

Also discussed is a framework in which the time or layer label is taken to be continuous, providing a differential equation, the Euler-Lagrange equation and its boundary conditions, as a necessary condition for a minimum of the cost function. This shows that the problem being solved is a two-point boundary value problem familiar in the discussion of variational methods. The use of continuous layers is denoted "deepest learning."

These problems respect a symplectic symmetry in continuous layer phase space. Both Lagrangian versions and Hamiltonian versions of these problems are presented. Their well-studied implementation in a discrete

Neural Computation **30, 2025–2055** (2018)
doi:10.1162/neco_a_01094

time/layer, while respecting the symplectic structure, is addressed. The Hamiltonian version provides a direct rationale for backpropagation as a solution method for a certain two-point boundary value problem.

1 Introduction

Through the use of enhanced computational capability, two seemingly unrelated inverse problems have flourished over the past decade. One is machine learning (ML) in the realm of artificial intelligence (Potember, 2017; Goodfellow, Bengio, & Courville, 2016; LeCun, Bengio, & Hinton, 2015) with developments that often go under the name "deep learning." The other is data assimilation (DA) in the physical and life sciences. This describes the transfer of information from observations to models of the processes producing those observations (Bennett, 1992; Evensen, 2009; Abarbanel, 2013).

This article demonstrates that these two areas of investigation are the same at a fundamental level. Each is a statistical physics problem where methods used in one may prove valuable for the other. The main goal of the article is to point out that many developments in DA can be used in the arena of ML. We also suggest that innovative methods from ML may be valuable in DA.

A connection between data assimilation, machine learning, and optimal control theory has been discussed in the literature (Doya, 1992; Haber, Ruthotto, Holtham, & Jun, 2017; Rey, 2017). Here we explore further the implications of this recognition that once one has a variational principle underlying the selection of an optimally trained network or DA model, the connection of such variational principles with Lagrangian and Hamiltonian formulations of classical mechanics provides additional opportunities for understanding the design and performance of ML problems.

Two areas of focus are attended to here: (1) a variational annealing (VA) method for the action (cost function) for ML or statistical DA that permits the location of the apparent global minimum of that cost function to be found and (2) the notion of analyzing each problem in continuous time or layer, which we call `deepest learning`, wherein it is clear that one is addressing a two-point boundary value problem (Ye, Rey et al., 2015; Gelfand & Fomin, 1963) with an underlying symplectic structure. Methods abound for solving such two-point boundary value problems (Press, Teukolsky, Vetterling, & Flannery, 2007) and for ensuring that symplectic structures are respected when time (or layer) is discretized. These may be quite fruitful in ML.

This article primarily discusses multilayer perceptrons or feedforward networks (Goodfellow et al., 2016), though it also makes it clear that the discussion carries over to recurrent networks as well (Jordan, 1986; Elman, 1990; Parlos, Chong, & Atiya, 1994).

2 Background

2.1 Machine Learning; Standard Feedforward Neural Nets. We begin with a brief characterization of simple architectures in feedforward neural networks (Potember, 2017; Goodfellow et al., 2016; LeCun et al., 2015). The network we describe is composed of an input layer l_0 and an output layer l_F and hidden layers $l_1, l_2, \ldots, l_F - 1$. Within each layer we have N active units, called neurons. For our purposes here, the "neurons" in each layer are assigned the same structure as a nonlinear map. This can be generalized to different numbers and different types of neurons in each layer at the cost of a notation explosion.

Data are available at layer l_0 and at layer l_F in many pairs of L-dimensional input, at l_0, and output, at l_F. These are pairs of L-dimensional vectors: $\{y_r^{(k)}(l_0), y_r^{(k)}(l_F)\}$ where $k = 1, 2, \ldots$ identifies the pairs, and r is an index on the L-dimensional data $r = 1, 2, 3, \ldots, L \leq N$.

The activity of the units in each hidden layer l, $x_r^{(k)}(l)$; $l_0 < l < l_F$, is determined by the activity in the previous layer. This connection is described by the nonlinear function $f_r(\bullet)$ via

$$x_r^{(k)}(l) = f_r(\mathbf{x}^{(k)}(l-1), l) = f_r\left(\sum_{q=1}^{N} W_{rq}(l) x_q^{(k)}(l-1)\right). \tag{2.1}$$

The summation over weights $W_{rq}(l)$ determines how the activities in layer $l-1$ are combined before allowing $f_r(\bullet)$ to act, yielding the activities at layer l. There are numerous choices for the manner in which the weight functions act, as well as numerous choices for the nonlinear functions, and we direct readers to the References for discussion of the virtues of those choices (Potember, 2017; Goodfellow et al., 2016; LeCun et al., 2015).

At the input and the output layers l_0, l_F, the network activities are compared to the M pairs of observations, and the network performance is assessed using an error metric, often a least squares criterion, or cost function,

$$C_M(\mathbf{x}^{(k)}(l), \mathbf{y}^{(k)}(l)) = \frac{1}{M}\sum_{k=1}^{M}\frac{1}{2L}\sum_{r=1}^{L} R_m(r,l)[x_r^{(k)}(l) - y_r^{(k)}(l)]^2, \tag{2.2}$$

$R_m(r,l)$ is nonzero only at $l = \{l_0, l_F\}$.

Minimization of this cost function over all $x_r^{(k)}(l)$ and weights $W_{rq}(l)$, subject to the network model, equation 2.1, is used to determine the weights, the variables $x_r^{(k)}(l)$ in all layers, and any other parameters appearing in the architecture of the network.

We wish to find the global minimum of the cost function, equation 2.2, which is a nonlinear function of the neuron activities, the weights, and any

other parameters in the functions at each layer. This is an NP-complete problem (Murty & Kabadi, 1987) and suggests one cannot find a global minimum of the ML problem, as set, unless there is a special circumstance. We will identify just such a circumstance in a DA problem equivalent to ML tasks.

The ML problem as described here assumes there is no error in the model itself, so that the minimization of the cost function, equation 2.2, is subject to strong equality constraints through the model. This results in the outputs at layer l_F, $\mathbf{x}^{(k)}(l_F)$, being very complicated functions of the parameters in the model and the activities at layers $l \leq l_F$. This is likely connected to the many local minima associated with the NP-complete nature of the search problem.

2.2 ML with Model Error. The ML problem as described here (Potember, 2017; Goodfellow et al., 2016) assumes there is no error in the model, equation 2.1. This results in the outputs at layer l_F, $\mathbf{x}^{(k)}(l_F)$, being very complicated functions of the parameters in the model and the activities at layers $l \leq l_F$. We relax the equality constraint, equation 2.1, by adding it as a penalty function to the cost function, defining the ML action $A_{ML}(\mathbf{X})$:

$$A_{ML}(\mathbf{X}) = \sum_{l=l_0}^{l_F} C_M(\mathbf{x}^{(k)}(l), \mathbf{y}^{(k)}(l)) + \frac{R_f}{2} \sum_{k=1}^{M} \sum_{l=l_0}^{l_F-1} \sum_{j=1}^{N} \left[x_j^{(k)}(l+1) - f_j\left(\sum_{i=1}^{N} W_{j,i}(l) x_i^{(k)}(l) \right) \right]^2. \quad (2.3)$$

In the limit $R_f \to \infty$, the equality constraint is restored. Another viewpoint sees the layer-to-layer rule as stochastic with additive gaussian noise and a diagonal precision matrix R_f.

2.3 Standard Statistical Data Assimilation. In the same spirit, we now briefly describe the formulation of a statistical DA problem.

DA uses observations of a sparse set of dynamical variables, associated with a model of the processes producing the observations. This will allow the estimation of parameters in the model and of the unobserved state variables of the model. We denote the state variables of a model as $\mathbf{x}(t) = \{x_1(t), x_2(t), \ldots, x_D(t)\}$, including fixed parameters in the model as state variables that do not change in time. This follows (Press et al., 2007) in spirit and simplifies our notation in many places.

The goal is to estimate any unknown parameters in the model and, because not all of the dynamical state variables in the model may be observed, estimate those unmeasured state variables as well. After a certain time window $[t_0, t_F]$ in which information is transferred to the model, we have an

estimate of the full model. This provides a new initial condition at $t = t_F$ $x_a(t_F)$; $a = 1, 2, \ldots, D$ for all state variables and yields estimated parameters. One may now make predictions with the completed model, and these may be compared to new observations. This validation by prediction is essentially the same as the question of generalization as addressed in ML (Goodfellow et al., 2016).

In DA, one has a window in time $[t_0, t_F]$ during which observations $\mathbf{y}^{(k)}(\tau_s)$ are made at times $t = \{\tau_1, \tau_2, \ldots, \tau_F\}$ that lie in $[t_0 \leq \tau_s \leq t_F]$; $s = 1, 2, \ldots, F$. At each observation time L, measurements $y_l(\tau_s)$; $l = 1, 2, \ldots, L$ are made, $L \leq D$. Through knowledge of the measurement instruments, the observations are related to the state variables of the model via measurement functions $h_l(\mathbf{x})$: $y_l(\tau_k) = h_l(\mathbf{x}(\tau_k))$; $l = 1, 2, \ldots, L$.

Using what knowledge we have of the processes producing the observations, we develop a dynamical model for the state variables. These satisfy a set of D dynamical differential equations:

$$\frac{dx_a(t)}{dt} = F_a(\mathbf{x}(t), t); \ a = 1, 2, \ldots, D. \tag{2.4}$$

The time dependence of the vector field $\mathbf{F}(\mathbf{x}, t)$ may come from external forcing functions driving the dynamics.

This set of differential equations will necessarily be represented in discrete time when solving them numerically, resulting in a map $x_a(t_k) \rightarrow x_a(t_{k+1}) = f_a(\mathbf{x}(t_k), t_k)$ in which the discrete time vector field $f_a(\bullet)$ is related to $F_a(\mathbf{x}, t)$ via the integration procedure one chooses for equation 2.4.

Starting from an initial condition $x_a(t_0)$, we use the discrete time model $x_a(t_{k+1}) = f_a(\mathbf{x}(t_k), t_k)$ to move forward to the first observation time τ_1, then to τ_2, . . . eventually reaching the end of the observation window at t_F. Altogether by making NI model integration steps in each of the intervals $[\tau_n, \tau_{n+1}]$, we make $(F + 1)NI$ time steps: $t_0 \rightarrow \tau_1 \rightarrow \tau_2 \ldots \rightarrow \tau_F \rightarrow t_F$. There are many reasons why we would want to have many model integration steps between observations. One might be the desire for higher temporal resolution in moving the model forward; another might be stability of the nonlinear dynamical processes involved.

The measurements are noisy and the models have errors, so this is a statistical problem. Our goal is to construct the conditional probability distribution, $P(\mathbf{X}|\mathbf{Y})$, of the model states $\mathbf{X}(t_F) = \{\mathbf{x}(t_0), \mathbf{x}(t_1), \ldots, \mathbf{x}(t_N), \ldots, \mathbf{x}(t_F)\}$, conditioned on the LF measurements $\mathbf{Y}(\tau_F) = \{\mathbf{y}(\tau_1), \mathbf{y}(\tau_2), \ldots, \mathbf{y}(\tau_k), \ldots, \mathbf{y}(\tau_F)\}$.

Assuming the transition to the model state at time t_{k+1} depends only on the model state at time t_k (i.e., the dynamics in $x_a(t_k) \rightarrow x_a(t_{k+1}) = f_a(\mathbf{x}(t_k), t_k)$ is Markov) and using identities on conditional probabilities (Abarbanel, 2013), one may write the action $A(\mathbf{X}) = -\log[P(\mathbf{X}|\mathbf{Y})]$ (suppressing the $\mathbf{Y}$ dependence of the action) as

$$A(\mathbf{X}) = -\sum_{n=1}^{F} CMI[\mathbf{X}(\tau_n), \mathbf{y}(\tau_n)|\mathbf{Y}(\tau_{n-1})] - \sum_{n=0}^{NI(F+1)-1} \log[P(\mathbf{x}(t_{n+1})|\mathbf{x}(t_n))] - \log[P(\mathbf{x}(t_0))]. \tag{2.5}$$

The conditional mutual information is given as (Fano, 1961) $CMI(a, b|c) = \log\left[\frac{P(a,b|c)}{P(a|c)\,P(b|c)}\right]$. If the model is error free, the transition probability, $P(\mathbf{x}(t_{n+1})|\mathbf{x}(t_n))$, is a delta function: $P(\mathbf{x}(t_{n+1})|\mathbf{x}(t_n)) = \delta^D(\mathbf{x}(t_{n+1}) - \mathbf{f}(\mathbf{x}(t_n), t_n))$.

With a representation of $P(\mathbf{X}|\mathbf{Y})$, we may evaluate conditional expected values of functions $G(\mathbf{X})$ on the path $\mathbf{X} = \mathbf{X}(t_F)$ of the model through the observation window $[t_0, t_F]$ as

$$E[G(\mathbf{X})|\mathbf{Y}] = \langle G(\mathbf{X})\rangle = \frac{\int d\mathbf{X}\, G(\mathbf{X}) \exp[-A(\mathbf{X})]}{\int d\mathbf{X} \exp[-A(\mathbf{X})]}, \tag{2.6}$$

in which

$$A(\mathbf{X}) = -\sum_{n=1}^{F} \log[P(\mathbf{y}(\tau_n)|\mathbf{X}(\tau_n), \mathbf{Y}(\tau_{n-1})] - \sum_{n=0}^{N(F+1)-1} \log P[(\mathbf{x}(t_{n+1})|\mathbf{x}(t_n))] - \log[P(\mathbf{x}(t_0))], \tag{2.7}$$

and terms depending only on the observations were canceled between numerator and denominator in the expected value, equation 2.6.

If the observations at the times τ_k are independent and the measurement function is the identity $y_r(\tau_k) = h_r(\mathbf{x}(\tau_k)) = x_r(\tau_k)$, and if the noise in the measurements is gaussian, with a diagonal inverse covariance matrix $R_m(r, \tau_k)$, the first term in the action $A(\mathbf{X})$, the `measurement error term`, takes the form

$$\sum_{n=1}^{F} \sum_{r=1}^{L} \frac{R_m(r, \tau_n)}{2} \Big(x_r(\tau_n) - y_r(\tau_n)\Big)^2. \tag{2.8}$$

If no measurement is made at τ_k, $R_m(\mathbf{x}, \tau_k) = 0$.

If the error in the model $x_a(t_{k+1}) = f_a(\mathbf{x}(t_k), t_k)$ is taken as additive and gaussian with diagonal precision matrix (inverse covariance matrix) $R_f(a)$, the second term in $A(\mathbf{X})$, the `model error term`, becomes

$$\sum_{n=0}^{NI(F+1)-1} \sum_{a=1}^{D} \frac{R_f(a)}{2} \left(x_a(t_{n+1}) - f_a(\mathbf{x}(t_n), t_n) \right)^2. \tag{2.9}$$

In each term, constants having to do with normalizations of the gaussians cancel in the expected value. If we keep these constants and take the limit $R_f(a) \to \infty$, we would restore the delta function and thus the strong constraints in the dynamics.

Finally, if one accepts ignorance of the distribution of initial conditions $P(\mathbf{x}(t_0))$ and selects it as uniform over the dynamical range of the model variables, $\langle G(\mathbf{X}) \rangle$ is evaluated with

$$A_0(\mathbf{X}) = \sum_{n=1}^{F} \sum_{r=1}^{L} \frac{R_m(r, \tau_n)}{2} \left(x_r(\tau_n) - y_r(\tau_n) \right)^2 + \sum_{n=0}^{NI(F+1)-1} \sum_{a=1}^{D} \frac{R_f(a)}{2} \left(x_a(t_{n+1}) - f_a(\mathbf{x}(t_n), t_n) \right)^2. \tag{2.10}$$

$\langle G(\mathbf{X}) \rangle$ is then

$$\langle G(\mathbf{X}) \rangle = \frac{\int d\mathbf{X}\, G(\mathbf{X}) \exp[-A_0(\mathbf{X})]}{\int d\mathbf{X} \exp[-A_0(\mathbf{X})]}. \tag{2.11}$$

This is the desired connection between the ML formulation (with model error) and the statistical DA formulation: identify layer labels as time $l \Leftrightarrow t$. One needs to compare equation 2.10 with equation 2.3 to see that with the recognition that DA receives information input at many times τ_k while the ML formulation receives information input only at the input and output layers, the problems are the same in their formulation. As we noted before, there has been some discussion of this connection (Doya, 1992; Rey, 2017; Haber et al., 2017), and we provide further analysis of the opportunities provided to train and design and validate ML problems.

Following its role in statistical DA, we call $A_0(\mathbf{X})$, equation 2.10, the standard model.

One critical suggestion here, relative to standard practice in ML (Potember, 2017; Goodfellow et al., 2016; LeCun et al., 2015), is that by allowing R_f to be finite from the outset—so acknowledging model error—we may add a tool for exploration in ML environments where typically no account for model error is introduced. Further, the hyperparameter R_f serves as a regulating device for the complexity of the surface in path space in which the estimation of states and parameters occurs.

3 Data Assimilation Developments of Use in Machine Learning

3.1 Finding the Global Minimum. The key to establishing estimates for unobserved state variables ($L < D$) and unknown parameters in the model is to perform, approximately of course, the integral, equation 2.11. The integral is not gaussian. If it were, we would just do it. As the functions $f_a(\bullet)$ are nonlinear, we must perform a numerical evaluation of equation 2.6. One can do this using Monte Carlo methods (Landau, Paez, & Bordeianu, 2010) or by the method of Laplace (Laplace, 1774, 1986). In the Laplace method one seeks minima of the action $A_0(\mathbf{X})$, equation 2.10.

The Laplace method approximates the integral with contributions from the lowest minima of the action if one can find them. Minima associated with paths $\mathbf{X}$ having larger action give exponentially smaller contributions to expected values, equation 2.11, than paths with smaller action. Viewing the Laplace method as an estimation of an expected value integral turns our attention to networks, trained with given data, where the smallest action minimum is quite a bit smaller than other action minima. This turns our goal away from seeking a convex $A(\mathbf{X})$ and to identifying when the information in the available data, combined with the architecture of the network, together provide a very good approximation to desired properties of the conditional probability distribution $P(\mathbf{X}|\mathbf{Y})$. We elaborate on this view in the examples worked out in later sections for both a DA problem and an ML problem.

We have developed a variational annealing (VA) approach (Ye, Kadakia, Rozdeba, Abarbanel, & Quinn, 2015; Ye, Rey, et al., 2015) to finding the path with the smallest value of the action. While we have no mathematical proof that the global minimum is found, our numerical results indicate this may be the case. The VA method produces a set of minima of the action giving a numerical clue as to the roughness of the surface in path $\mathbf{X}$ space. It also finds low-action minima with much higher rates of success than starting directly with large R_f.

In DA, the surface depends, among other items, on the number of measurements L at each observation time τ_k, on the hyperparameter R_f; and on the number of model time steps between measurement times τ_n. This translates directly to the analogous ML problem with time $\rightarrow$ layer. As the number of model time steps between measurement times increases, the number of hidden layers increases and the model architecture deepens.

VA proceeds by a kind of numerical continuation (Allgower & Georg, 1990) in R_f of the requirement that varying over all $\mathbf{X}$ and all parameters in $A_0(\mathbf{X})$ minimizes $A_0(\mathbf{X})$. The procedure begins by taking $R_f \rightarrow 0$, namely, the complete opposite of the value found in usual ML where $R_f \rightarrow \infty$ (deterministic, error free layer to layer maps) from the outset. In the $R_f = 0$ limit, the action is just a quadratic function of the model variables $\mathbf{x}(\tau_k)$ at the times measurements are made, and the minimization is simple: $x_r(\tau_k) = y_r(\tau_k)$ for the $r = 1, 2, \ldots, L \leq D$ data presented at the input and output

layers. The minimum can be degenerate as we know only $L \leq D$ values for the state variables.

At the first step of VA, we choose as a solution to the optimization problem $x_r(\tau_k) = y_r(\tau_k)$ and select the other $D - L$ states as drawn from a uniform distribution with ranges known from the dynamical range of the state variables. One can learn that well enough by solving the underlying model forward for various initial conditions. We make this draw K times and now have K paths $\mathbf{X}^0$ as candidates for the VA procedure.

Now we select a small value for R_f; call it R_{f0}. Using the K paths $\mathbf{X}^0$ as initial choices in our minimization procedure, we arrive at K new paths $\mathbf{X}^1$. This gives us K values of the action $A_0(\mathbf{X}^1)$ associated with the new paths $\mathbf{X}^1$.

Next we increase the value of R_f to $R_f = R_{f0}\alpha$ where $\alpha > 1$. (We have found values of α in the range 1.1 to 2 to be good choices, but a choice here is problem dependent.) For this new value of R_f, we perform the minimization of the action starting with the K initial paths $\mathbf{X}^1$ from the previous step to arrive at K new paths $\mathbf{X}^2$. Evaluating the action on these paths $A_0(\mathbf{X}^2)$ now gives us an ordered set of actions that are no longer as degenerate. Many of the paths $\mathbf{X}^2$ may give the same numerical value of the action. However, typically the "degeneracy" lies within the noise level of the data $\approx (1/\sqrt{R_m})$.

This procedure is continued until R_f is large enough, which is indicated by at least one of the action levels becoming substantially independent of R_f. As a check on the calculation, we observe that if the action $A_0(\mathbf{X})$ is independent of R_f, its expected value is that of the measurement error term. As the measurement errors were taken to be gaussian, this term in the action is distributed as χ^2, and its expected value is readily evaluated. If the action levels are at this expected value of χ^2 for large R_f, the procedure is consistent, and no further increases in R_f are required.

Effectively VA starts with a problem ($R_f = 0$) where the global minimum is apparent and systematically tracks it and many other paths through increases in R_f. In doing the tracking of the global minimum, one must check that the selected value of α is not too large lest one leave the global minimum and land in another minimum. Checking the result using smaller α is worthwhile.

It is important to note that simply starting with a large value of R_f, $R_f \approx 1$ or larger places one in the undesirable situation of the action $A_0(\mathbf{X})$ having multiple local minima into which any optimization procedure is quite likely to fall.

In the dynamical problems we have examined, one typically finds that as the number of measurements L at each τ_k is increased, fewer and fewer minima of the action remain, and when L is large enough, there is one minimum. This we attribute to the additional information from the augmented set of measurements, and this will be manifest in the discussion that follows where the additional information effectively controls unstable directions in the phase space.

3.2 Smallest Minimum; Not Necessarily a Convex Action. As our goal is to provide accurate estimations of the conditional expected value of functions $G(\mathbf{X})$ where $\mathbf{X}$, a path in model space, is distributed as $\exp[-A(\mathbf{X})]$, we actually do not require convexity of $A(\mathbf{X})$ as a function in path space. From the point of view of accurately estimating expected values, it is sufficient that the lowest action level be much smaller than the second-lowest action level. If the action value at the lowest level $A(\mathbf{X}_{\text{lowest}})$ is much smaller than the action value at the next minimum $A(\mathbf{X}_{\text{second lowest}})$, then by a factor $\exp[-\{A(\mathbf{X}_{\text{lowest}}) - A(\mathbf{X}_{\text{second lowest}})\}]$, the lowest path, $\mathbf{X}_{\text{lowest}}$, dominates the integral to be done and provides a sensible choice for the path at which to evaluate the integral.

We will see in the examples that when the VA procedure is used, we may encounter situations where the action is apparently not convex. However, it may have a distinct smallest action level, much smaller in magnitude than the next-lowest action level. That lowest level is expected to give a path that provides an accurate estimation to the expected value of functions $G(\mathbf{X})$. This may occur in cases where sufficient information from the data has been transferred to the model (in either ML or DA), and this can indicate the size model adequate for the problem posed.

4 An Example from DA and an Example from a Feedforward Neural Network

In this section, we examine one example from multilayer perceptrons and one example from statistical DA. The latter uses a differential equation model introduced by Lorenz in 1996 (Lorenz, 2006), which permits one to easily increase the number of dimensions of the phase space, easily select the number of observations within a designated measurement window, and easily choose the number of model evaluations between measurement times. The last is analogous to increasing the number of layers in a multilayer perceptron.

In each case, we perform a twin experiment, a designation taken from meteorological applications of DA. They are often used to test new DA methods before applying them to field or laboratory data. In a twin experiment, one uses a model to generate solutions from some initial conditions. These solutions, when gaussian noise is added to them, become our library of noisy data. With the noisy data, we use VA to estimate the unobserved state variables (hidden layer variables) and parameters/weights.

4.1 Data Assimilation for the Lorenz96 Model

4.1.1 Action Level Plots; $A(\mathbf{X})$ versus R_f. We begin by examining the D-dimensional dynamical equations introduced by Lorenz (2006):

$$\frac{dx_a(t)}{dt} = x_{a-1}(t)(x_{a+1}(t) - x_{a-2}(t)) - x_a(t) + \nu, \tag{4.1}$$

where $a = 1, 2, \ldots, D$; $x_{-1}(t) = x_{D-1}(t)$; $x_0(t) = x_D(t)$; $x_{D+1}(t) = x_1(t)$. ν is a fixed parameter, which we take to be 10.0 at which the solutions to the dynamical equations are chaotic (Kostuk, 2012). The equations for the states $x_a(t)$ are meant to describe weather stations on a periodic spatial lattice. This model is widely used in atmospheric science as a test bed for the exploration of innovative DA ideas.

Our example selects D = 11, and we display in Figure 1 the action level plot for L = 2, 4, 5, and 6 observations at each measurement time within the window $[t_0, t_F]$. We perform a twin experiment wherein we generate D time series $\{x_a(t);\ a = 1, 2, \ldots, D\}$ for equation 4.1 using a standard adaptive fourth-order Runge-Kutta algorithm with a time step $\Delta t = 0.025$ and an initial condition $\mathbf{x}(t_0)$ drawn from a uniform distribution over the range of the variables $\mathbf{x}(t)$, namely, $[-10, +10]$. To these solutions of equation 4.1, we add gaussian noise with mean zero and variance $\sigma^2 = 0.2$ to each time series $x_a(t)$. These noisy versions of our model time series constitute our data $\{y_a(t)\}$. L of these D time series are presented to the model at each observation time τ_n; $t_0 \le \tau_n \le t_F$; $n = 1, 2, \ldots, F$.

The measurement window is from $t_0 = 0$ to $t_F = 4.125$. $L = 2, 4, 5, 6$ measurements are made at each time step; these are the $\mathbf{y}(\tau_n)$. This gives us 165 measurements in the window [0,4.125]. The measurement error matrix $\mathbf{R}_m$ is taken to have diagonal elements at each measurement time τ_n and is zero at other times. Its magnitude is taken as $R_m = 1/\sigma^2 = 5$.

The model error precision matrix $R_f(a)$ is also taken as diagonal, with elements along the diagonal $R_f = R_{f0}2^{\beta}$, in performing the VA procedure, and we take $\beta = 0, 1, 2, \ldots$. R_{f0} was chosen 0.01.

The minimizations of nonlinear objective functions in the example using the Lorenz96 model were performed using the public domain software IPOPT (Wachter & Biegler, 2006) with a front end script written in Python.

In Figure 1, we display action-level plots for $L = 2, 3, 4$, and 6 observations at each measurement time. As we can see in the top left panel, where $L = 2$, there are numerous local minima in the action $A_0(\mathbf{X})$ for all values of $R_f \ge R_{f0}$, and these action levels remain as R_f becomes very large, indicating there are multiple local minima for the action $A(\mathbf{X})$. None of these minima are very far separated from the paths with the smallest minimum, so that the evaluation of the expected value integrals, equation 2.11, would require contributions from many maxima of the conditional probability distribution.

When $L = 4$ (see the top right panel), we begin to see an isolated action level whose contribution to the expected value integral is overwhelmingly larger than the contribution from the path giving rise to the next largest action level. In this case, the contribution of the second smallest action level is a bit less than 10^{-4} of the contribution of the smallest action level. As the

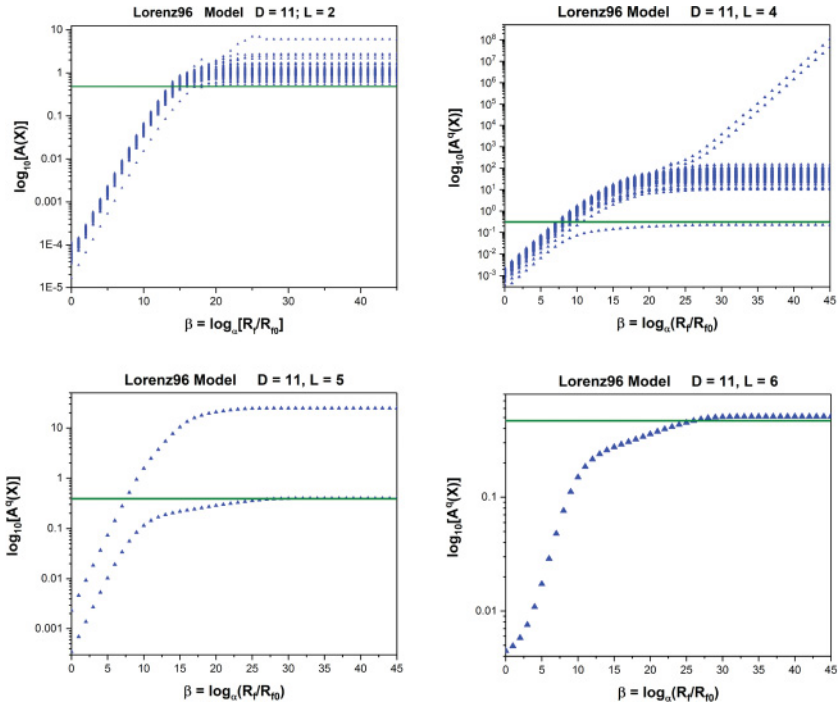

Figure 1: Action-level plots for the Lorenz96 model, equation 4.1, with $D = 11$ and $\nu = 10.0$. The variational annealing procedure is performed with 100 different initial conditions for the minimization of the action at each value of $R_f = R_{f0}\alpha^\beta$; $\beta = 0, 1, \ldots$; $\alpha = 2$. (Top left) $L = 2$ measurements at each measurement time. At $L = 2$, there are many minima, but none so much smaller than the others that it dominates the expected value integral, equation 2.11. (Top right) $L = 4$ measurements at each measurement time. At $L = 4$, the action in path space X has numerous local minima. The lowest minimum has an action value much smaller than the action values from the other minima, and this dominates the expected value integral, equation 2.11. (Bottom left) $L = 5$ measurements at each measurement time. At $L = 5$, the number of minima found is only two, and again the lowest minimum dominates the expected value integral. (Bottom right) $L = 6$ measurements at each measurement time. At $L = 6$, there is only one minimum of the action. The solid green line is the expected value of the measurement error term. This is distributed as χ^2. As the action becomes independent of R_f, its expected value should equal this value.

action has many minima, it is likely not convex, and that criterion may, for purposes of practical calculation, be somewhat unimportant. The value $L = 4$ is consistent with the observation in Kostuk (2012) that around $L \approx 0.4D$,

the instabilities in the Lorenz96 state space appear to be controlled by the DA process.

At $L = 5$ or 6 (see the bottom panels), we see that the dominance of the lowest action level is even clearer. The horizontal olive-colored line is the expected value of the measurement error term in the action. This is a sign of the consistency of the DA calculations.

The lesson from these calculations is that although there is one remaining action level for the cases when $L \geq 6$, indicating that sufficient information has been provided to the model from the observations to guarantee a kind of convexity, it is not necessary to measure this many quantities at each measurement time τ_n if our goal is accurate evaluation of the expected value integrals of interest. This tells us when we can stop increasing the number of observations in what might be an expensive measurement program.

4.1.2 Increasing the Number of Steps for the Dynamics. In Figure 2, we explore another aspect of the action-level plots. We still use $D = 11$, and now we hold $L = 6$ fixed. We increase the time between observations to 28 within the window $[t_0, t_F]$, so $\delta\tau \approx 0.147$. We choose now to move the model forward between observations 0, 2, 5, or 11 times. This is to provide an analogy to how many layers are present in an equivalent ML example. Our example here differs by having many entry points in the measurement window while the ML example has only one.

We display in the top left panel the action-level plots for the selected number of model evaluation steps. As one can see for 0 and 2 intermediate steps, we have many persisting minima of the action. At 5 and 11 intermediate steps, there is only a single minimum that is found, and for large R_f, it comes to the same action level as with 2 intermediate steps. All are consistent with the expected value of the measurement error term. This calculation, performed in an ML context, provides information on how many hidden layers are required to achieve a desired accuracy.

In the top right panel, we display the accuracy of the estimation of the single parameter in the Lorenz96 model. It has been set at $\nu = 10.0$ in producing the data, and that value is clearly selected for 5 or 11 intermediate model evaluations, while it is not so clearly selected for 2 intermediate steps and with zero intermediate steps there is a consistent 6% to 8% error.

In the bottom panel, we blow up the section of the top right panel where some of the results are obscured by overlay of the symbols. We also removed the solid olive green line at $\nu = 10.0$ for clarity.

We see, in this collection of calculations, as noted earlier (Ye, Kadakia et al., 2015; Ye, Rey et al., 2015), the ability to identify that the dominant minimum of the action depends on the number of measurements presented during the statistical DA procedure embodying transfer of information from the data to the model. In DA, this is associated with the number of positive conditional Lyapunov exponents (Kostuk, 2012) of the model. In the ML instantiation, it may play the same role when the number of data

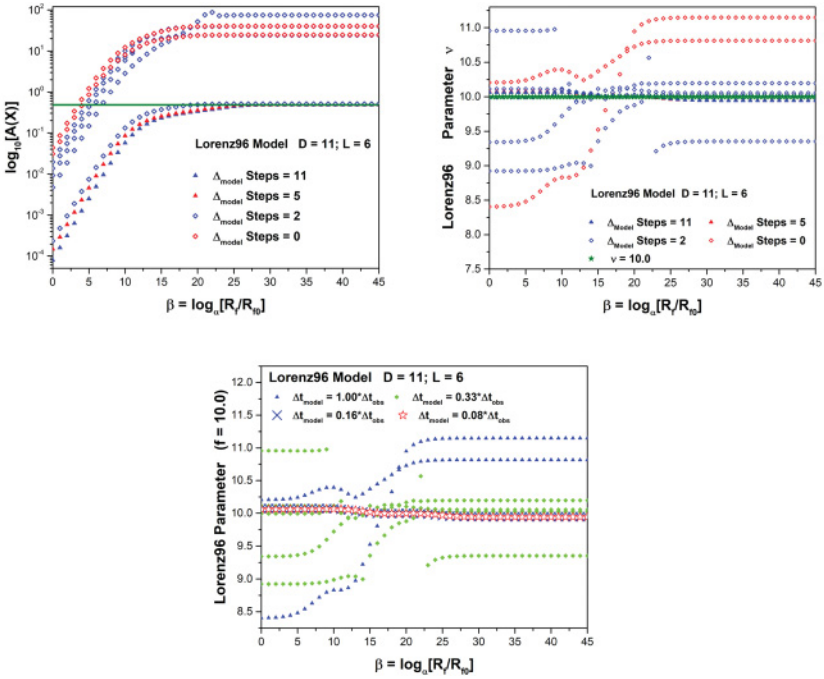

Figure 2: Parameter estimation and action-level results for the Lorenz96 model, $D = 11$, $L = 6$. $\nu = 10.0$. Observations were made every $\Delta t_{obs} = 0.15$, and $L = 6$ measurements were made at each observation time. (Top left) `Action-level estimates`: These are the action levels when $L = 6$ observations are made at each measurement time and with the choice of 0, 2, 5, and 11 model evaluation steps between measurement times. The horizontal olive green line indicates the expected action level for large R_f. (Top right) `Parameter estimates`: Between the observations, the model was evaluated 0, 2, 5, and 11 times leading to Δt_{model} = 1.0, 0.33, 0.16, and 0.08 $*\Delta t_{obs}$. The parameter estimates are quite accurate for 5 and for 11 model time steps between observations. They are less well defined or accurate for 0 or 2 model steps between observations. The red triangles denoting 5 model steps are nearly directly beneath the blue triangles denoting 11 model steps. Improvement of parameter estimation with increasing model steps between observations begins to saturate. One can associate the idea of increasing the number of model steps between observations as equivalent to deepening the hidden (unobserved) layers. The horizontal olive green line is the parameter value, $\nu = 10.0$, used in generating the data. (Bottom) `Blowup for parameter estimates`: The previous panel is blown up so that each Δt_{model} is now visible. The horizontal olive green line at $\nu = 10$ is removed.

presented at the output layer is not sufficient to determine the parameters and hidden states in each layer.

We also see the analog of deepening the network produces higher accuracy estimates of conditional expected values.

4.2 A Multilayer Perceptron; Feedforward Network

4.2.1 Creating Data with a Known Network. We built a feedforward network with l_F layers: one input layer at l_0 and one output layer at l_F. This network has $l_F - 2$ hidden layers. A network with 100 layers was constructed with weights drawn from a uniform distribution $U[-0.1, 0.1]$. The network has 10 neurons in each layer. The activity in neuron j in layer l, $x_j(l)$, is related to $x_j(l-1)$ in the previous layer as

$$x_j(l) = g(\mathbf{W}(l)\mathbf{x}(l-1)); \quad g(z) = 0.5\left[1 + \tanh\left(\frac{z}{2}\right)\right]. \tag{4.2}$$

Data, consisting of input-output pairs $\{x_i^{(k)}(l_0), x_i^{(k)}(l_F)\}$; $k = 1, 2, \ldots$, were constructed by presenting $x_i^{(k)}(l_0)$ at layer l_0 and passing it through the network to generate $x_i^{(k)}(l_F)$ at the output layer l_F. To these input-output pairs, we added gaussian noise with zero mean and variance 0.0025 to create our noisy library of $\{\mathbf{y}^{(k)}(l_0), \mathbf{y}^{(k)}(l_F)\}$ pairs. These noisy data were now stored for further use.

4.2.2 Selecting a Model to Represent the $\{\mathbf{y}^{(k)}(l_0), \mathbf{y}^{(k)}(l_F)\}$ *Pairs.* We are now presented a subset of this library of input-output pairs and choose a multilayer perceptron with l_F layers and N neurons per layer to represent the input-output relationship: $\{y_i^{(k)}(l_0), y_i^{(k)}(l_F)\}$; $k = 1, 2, \ldots, M$. This selected network had $N = 10$ neurons in each of $l_F = 10$ layers at first. At the input and output layers, 10 noisy values $\{y_i^{(k)}(l_0), y_i^{(k)}(l_F)\}$; $i = 1, \ldots, N = 10$ are presented for various values of M. This corresponds to $L = 10$ in the notation used before. As M is increased, more and more of the complexity of the relationship of the pairs is explored and more and more is demanded of our selected network.

We purposely chose to explore the relationship of our noisy data set pairs $\{\mathbf{y}^{(k)}(l_0), \mathbf{y}^{(k)}(l_F)\}$; $\mathbf{y} = \{y_1, y_2, \ldots, y_L\}$ with a wrong model. We tried to place ourselves in the situation where one receives a library of input-output pairs and is asked to develop a multilayer perceptron to learn the representation of the information in the pairing. This is more like the usual situation in which one wishes to perform a machine learning task. We call the selected model with $l_F = 50$ "wrong" because in this twin experiment, we know the model that generated the data. Our goal is to see where the incorrectness of the model displays itself, and then to augment the original model choice to improve its ability to predict outputs from inputs it had not seen before.

M input-output pairs are presented to the model with $L = 10$ inputs $\mathbf{y}^{(k)}(l_0)$ at layer l_0 and 10 data outputs $\mathbf{y}^{(k)}(l_F)$ at layer l_F. We investigated $M = 1, 2, 10$ and 100.

In each case, we minimized the action over all the weights and the states $x_a^{(k)}(l)$ at all layers of the model:

$$A_M(\mathbf{X}) = \frac{1}{M}\sum_{k=1}^{M}\left\{\frac{R_m}{2L}\sum_{i=1}^{N}\left[(x_i^{(k)}(l_0) - y_i^{(k)}(l_0))^2 + (x_i^{(k)}(l_F) - y_i^{(k)}(l_F))^2\right]\right.$$
$$\left. + \frac{R_f}{Nl_F}\sum_{l=l_0}^{l_F-1}\sum_{a=1}^{N}\left[x_a^{(k)}(l+1) - g_a(\mathbf{W}(l)\mathbf{x}^{(k)}(l))\right]^2\right\}. \tag{4.3}$$

We use the variational annealing procedure already described to identify the action levels for various paths through the network. The initial value of R_{f0}/R_m is taken to be 10^{-8}, and this is incremented via $R_f/R_m = R_{f0}\alpha^\beta$ with $\alpha = 1.1$ and $\beta = 0, 1, \ldots,$. In the numerical optimizations for the ML example, we used L-BFGS-B (Byrd, Lu, Nocedal, & Zhu, 1995; Zhu, Byrd, & Nocedal, 1997).

To present more information to our selected model, we could increase the number of training pairs available to the network at l_0 and l_F; this is our number M. M can be chosen as large as the user wishes. To augment the ability of the model to represent the complexity of the data set as M increases, we could also increase N or l_F or both.

In the calculations we present here, we did not explore the importance of presenting fewer than N data elements at the input or output layer; $L < N$. We have done that for other examples not reported here, and as one increases L, the ability of the model to represent the data does improve. We also do not report here on using nonlinear neuron functions such as $g(x) = \log(1 + e^x)$, a ReLU-like nonlinearity (Goodfellow et al., 2016; LeCun et al., 2015). The results are not substantially different from what we report here.

4.2.3 Prediction with the Selected Model. Once we have used the VA procedure to select the path $\mathbf{X}$, comprising $x_j^{(k)}(l)$ and the $W_{ij}(l)$ for all neurons and weights at each layer $l_0 \le l \le l_F$, giving the minimum action $A(\mathbf{X})$, we have a new model that can be used to predict the output from presenting a new input. From the original data set of noisy pairs $\{y_j(l_0), y_i(l_F)\}$, we now select M_P pairs for testing the model. The inputs $y_j^{(r)}(l_0)$; $r = 1, 2, \ldots, M_P$; $j = 1, \ldots, N$ are presented to the input layer of the estimated model. The outputs from the operation of the model network $x_j^{(r)}(l_F)$ are compared to the known outputs $y_j^{(r)}(l_F)$ using the averaged square error:

$$E(l_F)^2 = \frac{1}{NM_P} \sum_{r=1}^{M_P} \sum_{j=1}^{N} (x_j^{(r)}(l_F) - y_j^{(r)}(l_F))^2. \tag{4.4}$$

4.2.4 Models to Represent the Pairs $\{\mathbf{y}^{(k)}(l_0), \mathbf{y}^{(k)}(l_F)\}$*: Results.* All models have 10 neurons in each layer, and we examine different values of l_F and M, the number of training input-output pairs. In each calculation, we used $K = 100$ initial paths in the VA procedure. In each graphic, therefore, there are 100 action levels possible. As the number of training pairs is increased, the number of action levels decreases as a function of R_f/R_m, so many, and then all, of the K = 100 initial conditions go to the same minimum of the action.

Our first model selected $l_F = 10$; the results are in Figure 3. In the upper left panel, $M = 1$ data pair, there is not enough information at $M = 1$ to produce an action level that dominates the integral, equation 2.6. This is indicated in this graphic as well as others we will show by the many action levels (up to $K = 100$ here) that do not separate well.

$M = 2$ data pairs. Here we see a few action levels separate from the others, and a number of remaining levels are close as R_f/R_m grows. The apparent smallest action level in Figure 3 is suspect because its action is too low. When the action levels as a function of R_f/R_m become nearly independent of R_f, the action level should be approximately one in magnitude. In the lower left panel, $M = 10$ data pairs. At large R_f/R_m, only one action level remains. Now the remaining action level at large R_f/R_m is near one as we anticipated by the normalization chosen in the action for the measurement error term. In the lower right panel is a blowup of the lower left panel to show that only one action level remains for large enough R_f/R_m.

The essential message from using the VA procedure is that when enough information is presented to the model, the action levels show a clear global minimum with much higher valued (lower probability) action minima. Finally, when the information presented is enough, only a single (global) minimum remains. The path associated with this action level is then expected to produce good predictions when new input-output pairs are presented to the model. We will see how this works out in the $l_F = 50$ example below.

Our second model selected $l_F = 20$, and the results are in Figure 4. In each calculation, we display 100 action levels at each value of R_f/R_m in the variational annealing protocol described in the text. The action levels, equation 4.3, are displayed when we increase R_f/R_m from a quite small value, $R_{f0}/R_m = 10^{-8}$, to large values: $R_f/R_m = (R_{f0}/R_m)\alpha^\beta$ with $\alpha = 1.1$ and $\beta = 0, 1, \ldots$, until $R_f/R_m \approx 10^{10}$.

In the upper left panel, $M = 1$ data pair. There is not enough information at $M = 1$ to produce an action level that dominates the integral, equation 2.6. In the upper right panel, $M = 2$ data pairs, and in the lower left panel, $M = 5$ data pairs. At large R_f/R_m only one action level remains: lower right

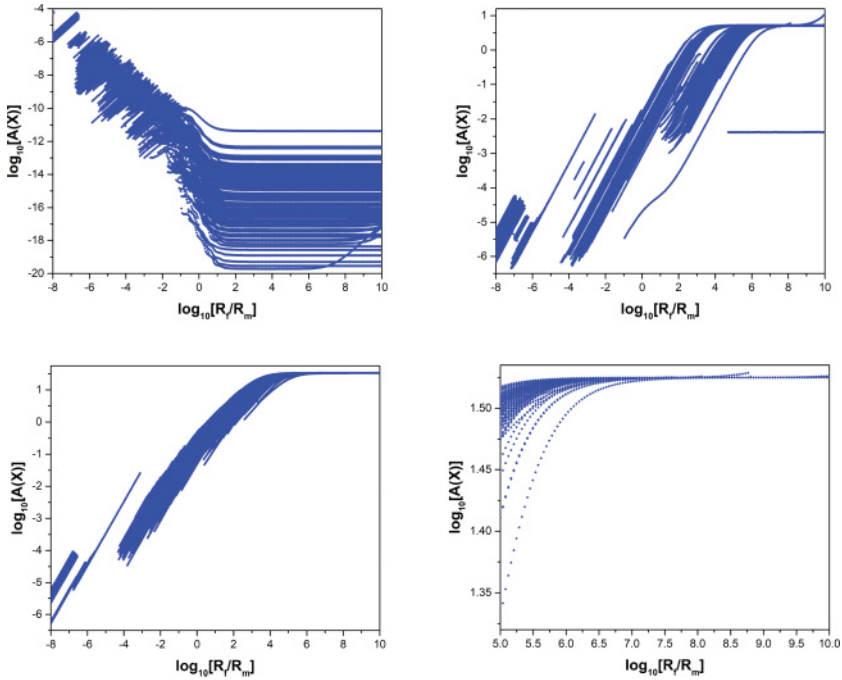

Figure 3: Action levels from training, estimating parameters (weights), and unobserved state variables, in a network with $l_F = 10$ layers and $N = 10$ neurons per layer. The data input-output pairs were generated with $l_F = 100$, $N = 10$, and chosen parameters. In each calculation, we display 100 action levels at each value of R_f/R_m in the variational annealing protocol described in the text. The action levels, equation 4.3, are displayed when we increase R_f/R_m from a quite small value $R_{f0}/R_m = 10^{-8}$ to large values $R_f/R_m = (R_{f0}/R_m)\alpha^\beta$ with $\alpha = 1.1$ and $\beta = 0, 1, \ldots$, until $R_f/R_m \approx 10^{10}$. (Upper left) $M = 1$ data pair. There is not enough information at $M = 1$ to produce an action level that dominates the integral, equation 2.6. (Upper right) $M = 2$ data pairs. (Lower left) $M = 10$ data pairs. At large R_f/R_m, only one action level remains. (Lower right) Blowup of the lower left panel to show that only one action level remains for large enough R_f/R_m. The input-output pairs constituting the data were generated by a network with $N = 10$ neurons in each of $l_F = 100$ layers.

panel with $M = 10$ data pairs. At large R_f/R_m, only one action level remains. The input-output pairs constituting the data were generated by a network with $N = 10$ neurons in each of $l_F = 100$ layers.

The remarks on this set of results are essentially the same as for Figure 3. Here we also include the calculation of $M = 5$. Again, if we start with the data from our library of input-output pairs, we would conclude that

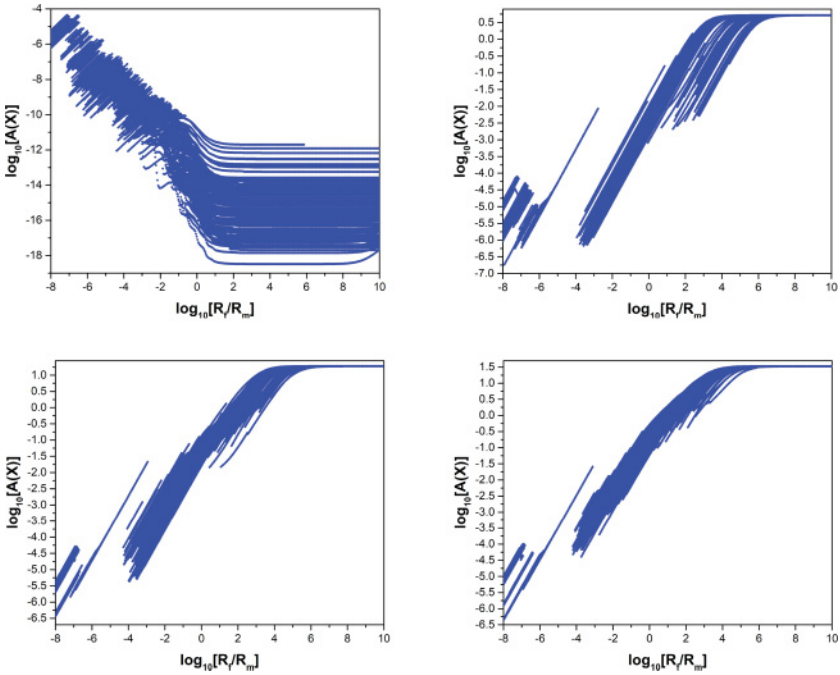

Figure 4: Action levels from training, estimating parameters (weights), and unobserved state variables in a network with $l_F = 20$ layers and $N = 10$ neurons per layer. The data input-output pairs were generated with $l_F = 100, N = 10$, and chosen parameters. In each calculation, we display 100 action levels at each value of R_f/R_m in the variational annealing protocol described in the text. The action levels, equation 4.3, are displayed when we increase R_f/R_m from a quite small value $R_{f0}/R_m = 10^{-8}$ to large values $R_f/R_m = (R_{f0}/R_m)\alpha^\beta$ with $\alpha = 1.1$ and $\beta = 0, 1, \ldots,$ until $R_f/R_m \approx 10^{10}$. (Upper left) $M = 1$ data pair. There is not enough information at $M = 1$ to produce an action level that dominates the integral, equation 2.6. (Upper right) $M = 2$ data pairs. (Lower left) $M = 5$ data pairs. At large R_f/R_m, only one action level remains. (Lower right) $M = 10$ data pairs. At large R_f/R_m, only one action level remains. The input-output pairs constituting the data were generated by a network with $N = 10$ neurons in each of $l_F = 100$ layers.

an accurate model would be achieved with each of the model training by $M = 2, 5,$ and 10. In our next example, we will also compare the models trained with different M via their predictions output for new inputs.

Our last model presented here has $l_F = 50$ and displays action levels for $M = 1, 2, 10$ in Figure 5. The action levels, equation 4.3, are displayed when we increase R_f/R_m from a quite small value, $R_{f0}/R_m = 10^{-8}$, to large values, $R_f/R_m = (R_{f0}/R_m)\alpha^\beta$, with $\alpha = 1.1$ and $\beta = 0, 1, \ldots,$ until $R_f/R_m \approx 10^{10}$.

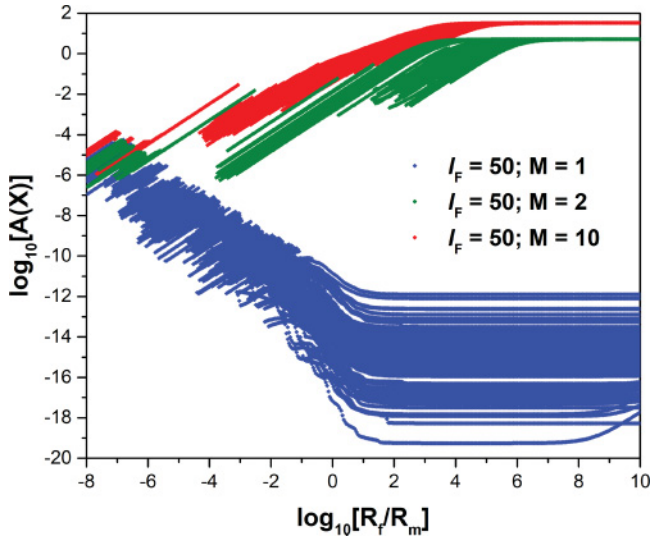

Figure 5: Action levels from training, estimating parameters (weights), and unobserved state variables, in a network with $l_F = 50$ layers and $N = 10$ neurons per layer. The data input-output pairs were generated with $l_F = 100, N = 10$, and chosen parameters. In each calculation, we display 100 action levels at each value of R_f/R_m in the variational annealing protocol described in the text. The action levels, equation 4.3, are displayed when we increase R_f/R_m from a quite small value $R_{f0}/R_m = 10^{-8}$ to large values $R_f/R_m = (R_{f0}/R_m)\alpha^\beta$ with $\alpha = 1.1$ and $\beta = 0, 1, \ldots$, until $R_f/R_m \approx 10^{10}$. We display action levels for $M = 1, 2$, and 10 input-output pairs presented to the model network.

We display action levels for $M = 1, 2$, and 10 input-output pairs presented to the model network.

The results here are quite similar to those for smaller networks, $l_F = 10, 20$, and our remarks on the action-level display are much the same.

Finally, we evaluate the quality of predictions in the cases with $l_F = 10, 20, 50$ with $M = 1, 2, 5, 10$ for training each model. The prediction errors are shown in Figure 6. Looking, for example, at the outcome when $L_F = 50$, we see that increasing the number of training pairs leads to a decreasing prediction error. Depending on the information in the library of training pairs, we may find networks of different sizes in l_F (layer size or machine depth here), and different numbers of training pairs to achieve a desired level of prediction accuracy.

The VA procedure allows one to select a model among the choices one makes in the model design to accomplish the quality of prediction error desired. In the example data set we created and used here, many fewer layers were required than the number used ($l_F = 100$) to create the data set, and

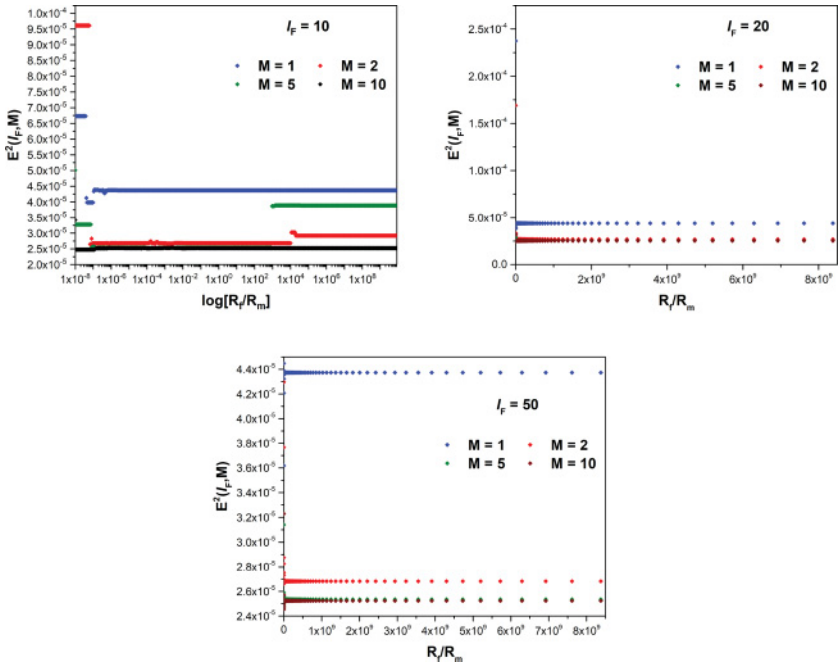

Figure 6: Prediction errors, equation 4.4, for proposed networks with $l_F = 10, 20, 50$ layers and $M = 1, 2, 5, 10$ training pairs. In each case, the path, hidden-layer neuron activities and weights, producing the lowest minimum in the action levels determined the trained network. The number of test pairs presented to the trained network was $M_P = 100$. The training data input-output pairs, as well as the testing input-output pairs, were generated from a network with $l_F = 100$ layers.

with the presentation of quite a small number of input-output pairs, we could achieve small prediction errors. It may go without saying that while we lack tools to simply examine a set of data pairs and know which model (l_F, N) and architecture one would require and how many training pairs would also be required, VA provides a constructive path to addressing the question based on the prediction accuracy the user desires.

4.3 Recurrent Networks. In a recurrent network architecture, one allows both interactions among neurons from one layer to another layer as well as interactions among neurons within a single layer (Jordan, 1986; Elman, 1990). The activity $x_j(l)$ of neuron j, $j = 1, 2, \ldots, N$ in layer l $\{l_0, l_1, \ldots, l_F\}$ is given by $x_j(l) = f[\sum_i w_{ji}(l)x_i(l-1)]$ in a feedforward, `layer goes to the next layer`, network.

We can add interactions within a layer in the same fashion; and to give some dynamics to this within-layer activity, we introduce a sequential label σ to the activity of neuron j in layer l: $x_j(l, \sigma)$. The mapping from layer to layer and within a layer can be summarized by

$$x_j(l, \sigma) = f\left[\sum_i W_{ji}(l) x_i(l-1, \sigma) + \sum_i w_{ji}(l) x_j(l, \sigma - 1)\right]. \tag{4.5}$$

Another version of this allows the nonlinear function to be different for layer-to-layer connections and within-layer connections, so

$$x_j(l, \sigma) = f\left[\sum_i W_{ji}(l) x_i(l-1, \sigma)\right] + g\left[\sum_i w_{ji}(l) x_j(l, \sigma - 1)\right], \tag{4.6}$$

where $f(x)$ and $g(x)$ can be different nonlinear functions.

We can translate these expressions into the DA structure by recognizing that $x_j(l)$ is the model variable in the layer-to-layer function, while in the recurrent network, the state variables become $x_j(l, \sigma)$. It seems natural that as dimensions of connectivity are added—here going from solely feedforward to that plus within-layer connections—that additional independent variables would be aspects of the neuron state variables' representation.

In adding connections among the neurons within a layer, we have another independent variable—we called it σ—and the point neurons depending on layer alone become `fields` $x_j(l, \sigma)$. In the ML/AI networks, we have no restrictions on the number of independent variables. This may lead to the investigation of neural fields $\phi_j(\mathbf{v})$, where $\mathbf{v}$ is a collection of independent variables indicating which layers are involved in the progression of the field from an input to an output layer.

However many independent variables and however many neurons we use in the architecture of our model network, the overall goal of identifying the conditional probability distribution $P(\mathbf{X}|\mathbf{Y})$ and estimating the moments or expected values of interest still comes to one form or another of approximating integrals such as equation 2.6.

4.4 Making Time Continuous; Continuous Layers: Deepest Learning

4.4.1 Euler-Lagrange Equations for DA and ML; Lagrangian Formulation. There is much to learn about the DA or ML problem as the number of layers or, equivalently, the number of time points within an epoch become very large. The limit of the action where the number of layers becomes a continuous variable is, in DA notation (Kadakia, Rey, Ye, & Abarbanel, 2017),

$$A_0(\mathbf{x}(t), \dot{\mathbf{x}}(t)) = \int_{t_0}^{t_F} dt\, L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t), \tag{4.7}$$

where

$$\begin{aligned}
&L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) \\
&= \sum_{r=1}^{L} \frac{R_m(r,t)}{2}\left(x_r(t) - y_r(t)\right)^2 + \sum_{a=1}^{D} \frac{R_f(a)}{2}\left(\dot{x}_a(t) - F_a(\mathbf{x}(t))\right)^2. \\
&= \chi(\mathbf{x}(t) - \mathbf{y}(t)) + \sum_{a=1}^{D} \frac{R_f(a)}{2}\left(\dot{x}_a(t) - F_a(\mathbf{x}(t))\right)^2.
\end{aligned} \tag{4.8}$$

In this formulation, the quantity $R_m(r,t)$ is nonzero only near the times $t \approx \tau_k$. Within the ML context, we call this "deepest learning" as the number of layers goes to infinity in a useful manner.

To find the minima of the action, we must require the necessary condition that changes the action vanish, under small variations in $x_a(t) \to x_a(t) + \delta x_a(t)$ and $\dot{x}_a(t) \to \dot{x}_a(t) + \delta \dot{x}_a(t)$. This means

$$\begin{aligned}
&\delta A_0(\mathbf{x}(t), \dot{\mathbf{x}}(t)) \\
&= \int_{t_0}^{t_F} dt \left\{ \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial x_a(t)} \delta x_a(t) + \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \dot{x}_a(t)} \delta \dot{x}_a(t) \right\} \\
&= \int_{t_0}^{t_F} dt \left\{ \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial x_a(t)} - \frac{d}{dt} \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \dot{x}_a(t)} \right\} \delta x_a(t) \\
&\quad + \delta x_a(t) \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \dot{x}_a(t)}\bigg|_{t=t_F} - \delta x_a(t) \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \dot{x}_a(t)}\bigg|_{t=t_0} \\
&= 0.
\end{aligned} \tag{4.9}$$

The minimization of the action now requires that the paths $\mathbf{x}(t)$ in $\{\mathbf{x}(t), \dot{\mathbf{x}}(t)\}$ space satisfy the Euler-Lagrange equation:

$$\frac{d}{dt}\left[\frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \dot{x}_a(t)}\right] = \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial x_a(t)}. \tag{4.10}$$

The solutions must also satisfy the boundary conditions $\sum_{a=1}^{D} \delta x_a(t_0)$ $p_a(t_0) = 0$, $\sum_{a=1}^{D} \delta x_a(t_F) p_a(t_F) = 0$, where $p_a(t) = \partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)/\partial \dot{x}_a(t)$ is the canonical momentum.

For the standard model, the Euler-Lagrange (E-L) equations take the form

$$R_f\left[\frac{d}{dt}\delta_{ab}+DF_{ab}(\mathbf{x}(t))\right]\left[\frac{dx_b(t)}{dt}-F_b(\mathbf{x}(t))\right]=\frac{\partial\chi(\mathbf{x}(t)-\mathbf{y}(t))}{\partial x_a(t)}$$
$$\frac{d^2x_a(t)}{dt^2}-\sum_{b=1}^{o}\Omega_{ab}(\mathbf{x}(t))\dot{x}_b(t)=\frac{\partial\left[\frac{\chi(\mathbf{x}(t)-\mathbf{y}(t))}{R_f}+\frac{\mathbf{F}(\mathbf{x}(t))^2}{2}\right]}{\partial x_a(t)}+\frac{\partial F_a(\mathbf{x}(t),t)}{\partial t}, \tag{4.11}$$

where we have

$$DF_{ab}(\mathbf{x})=\partial F_a(\mathbf{x})/\partial x_b;\quad \Omega_{ab}=DF_{ab}(\mathbf{x}(t))-DF_{ba}(\mathbf{x}(t));$$
$$\chi(\mathbf{x}(t)-\mathbf{y}(t))=\sum_{r=1}^{L}\frac{R_m(l,t)}{2}\Big(x_r(t)-y_r(t)\Big)^2. \tag{4.12}$$

The E-L equations show how errors represented on the right-hand side of the E-L equation drive the model variables at all layers to move $\mathbf{x}(l)\to\mathbf{y}(l)$ where data are available.

In the integral for $\langle G(\mathbf{X})\rangle$, the coordinates $\mathbf{x}(t_0)$ and $\mathbf{x}(t_F)$ are not restricted, so we have the "natural" boundary conditions (Gelfand & Fomin, 1963; Kot, 2014; Liberzon, 2012) $p_a(t_0)=0$ and $p_a(t_F)=0$.

This shows quite clearly that the minimization problem requires a solution of a two-point boundary value problem in $\{\mathbf{x}(t),\mathbf{v}(t)=\dot{\mathbf{x}}(t)\}$ space. One way to address two-point boundary value problems is to start at one end, t_0, with a value of $\mathbf{x}(t_0)$, and proceed from t_F with a value of $\mathbf{x}(t_F)$ and integrate both ways requiring a match (Press et al., 2007). Furthermore, the residual of the measurement error term on the right-hand side of equation 4.11 nudges the solution in $\mathbf{x}(t)$ to the desired output.

If one were to specify $\mathbf{x}(t_0)$ but not $\mathbf{x}(t_F)$, then the boundary conditions for the Euler-Lagrange equation are the given $\mathbf{x}(t_0)$ ($\delta\mathbf{x}(t_0)=0$) and require the canonical momentum $p_a(t_F)=0$. Examining the Hamiltonian dynamics for this problem then suggests integrating the $\mathbf{x}(t)$ equation forward from t_0 and the canonical momentum equation backward from t_F. This is backpropagation.

The skew-symmetric matrix $\Omega_{ab}(\mathbf{x}(t))=\partial F_a(\mathbf{x}(t)/\partial x_b(t)-\partial F_b(\mathbf{x}(t)/\partial x_a(t)$ generates a local rotation. The potential expansion or contraction of orbits of $\mathbf{x}(t)$ is under control because of the compact structure of the rotations so generated. In the Hamiltonian formulation where backprop is employed, this balancing aspect of the Jacobians, $\partial F_a(\mathbf{x}(t)/\partial x_b(t)$, is split between the coordinate equation and the canonical momentum equation and may lead to unstable or numerically quite difficult issues in its implementation. The solution in Lagrangian coordinates $\{\mathbf{x}(t),\dot{\mathbf{x}}(t)\}$ avoids this and retains the symplectic nature of the solutions (Marsden & West, 2001; Kadakia et al., 2017).

It could be that making backpropagation explicitly symplectic (Hairer, Lubich, & Wanner, 2006) could address this issue.

4.4.2 Gauge Transformations. The Euler-Lagrange equation resembles the motion of a charged object in a D-dimensional magnetic field and an electric field in D-dimensions. The term analogous to the magnetic field is $\Omega_{ab} = DF_{ab}(\mathbf{x}(t)) - DF_{ba}(\mathbf{x}(t))$, as it is perpendicular to the velocity. The analog electric field is

$$\mathcal{E}(\mathbf{x}, t) = \frac{\partial \left[\frac{\chi(\mathbf{x}(t) - \mathbf{y}(t))}{R_f} + \frac{\mathbf{F}(\mathbf{x}(t))^2}{2} \right]}{\partial x_a(t)} + \frac{\partial F_a(\mathbf{x}(t), t)}{\partial t}, \tag{4.13}$$

recalling $-E_a(\mathbf{x}, t) = \frac{\partial \phi(\mathbf{x},t)}{\partial x_a} + \frac{\partial A_a(\mathbf{x},t)}{\partial t}$ in electrodynamics. The analogy is that the scalar potential $\phi(\mathbf{x}, t)$ is $\chi(\mathbf{x}(t) - \mathbf{y}(t)) + \mathbf{F}(\mathbf{x}, t)^2/2$, and the vector potential is $A_a(\mathbf{x}, t) = F_a(\mathbf{x}, t)$.

If we make the "gauge" transformation $F_a(\mathbf{x}, t) \to F_a(\mathbf{x}, t) + \partial_a \psi(\mathbf{x}, t)$, then Ω_{ab} is unchanged. The "electric field" becomes

$$\mathcal{E}(\mathbf{x}, t) \to \mathcal{E}(\mathbf{x}, t) + \nabla_a \left[\mathbf{F}(\mathbf{x}, t) \cdot \nabla \psi(\mathbf{x}, t) + \frac{\nabla \psi(\mathbf{x}, t))^2}{2} + \frac{\partial \psi(\mathbf{x}, t)}{\partial t} \right], \tag{4.14}$$

and this "electric field" and the equations of motion are unchanged if $\psi(\mathbf{x}, t)$ satisfies

$$\left(\frac{\partial \psi(\mathbf{x}, t)}{\partial t} + \mathbf{F}(\mathbf{x}, t) \cdot \nabla \psi(\mathbf{x}, t) \right) + \frac{(\nabla \psi(\mathbf{x}, t))^2}{2} = 0. \tag{4.15}$$

Along with an invariance such as this is a conserved current. In this case, it happens to be local in $\mathbf{x}$ space conservation of number of particles. Were we discussing recurrent networks, more structure would be involved.

4.4.3 Hamiltonian Dynamics Realization; Backpropagation. If one moves from the Lagrangian realization of the variational problem to a Hamiltonian version by trading in the Lagrangian phase space from $\{\mathbf{x}(t), \mathbf{v}(t)\}$ to canonical coordinates $\{\mathbf{x}(t), \mathbf{p}(t)\}$, then the Hamiltonian $H(\mathbf{x}, \mathbf{p})$ for the standard model reads

$$\begin{aligned} H(\mathbf{x}, \mathbf{p}, t) = & \sum_{a=1}^{D} \left\{ \frac{p_a(t) p_a(t)}{2R_f(a)} + p_a(t) F_a(\mathbf{x}(t)) \right\} \\ & - \sum_{r=1}^{L} \frac{R_m(r, t)}{2} (x_r(t) - y_r(t))^2. \end{aligned} \tag{4.16}$$

In these coordinates, the equations of motion are then given by Hamilton's equations:

$$\frac{dp_a(t)}{dt} = -p_b(t)\frac{\partial F_b(\mathbf{x}(t))}{\partial x_a(t)} + \delta_{ar}R_m(r,t)(x_r(t) - y_r(t)),$$
$$\frac{dx_a(t)}{dt} = F_a(\mathbf{x}(t)) + \frac{p_a(t)}{R_f(a)}. \tag{4.17}$$

Returning from this to discrete time (or layers), we see that if the variational principle is carried out in $\{\mathbf{x}, \mathbf{p}\}$ space, the boundary conditions $p_a(t_0) = p_a(t_F) = 0$ are quite easy to impose while the other variables, all the $x_a(t_k)$ and the $p_a(t_k)$; $k \neq 0, F$, are varied. Going forward in $\mathbf{x}$ and backward in $\mathbf{p}$ is neither required nor suggested by this formulation. It is worth noting that in either $\{\mathbf{x}, \mathbf{v}\}$ space or $\{\mathbf{x}, \mathbf{p}\}$ space, the continuous time (layer) formulation has a symplectic symmetry (Gelfand & Fomin, 1963; Kadakia et al., 2017). This not automatically maintained when the discrete time (layer) problem is reinstated (Marsden & West, 2001; Wendlandt & Marsden, 1997); however, many choices of integration procedure in which time/layer becomes discrete and the symplectic symmetry is maintained are known (Marsden & West, 2001; Wendlandt & Marsden, 1997; Hairer et al., 2006).

In a detailed analysis (Kadakia et al., 2017; Ye, Rey et al., 2015) of the variational problem in Lagrangian and Hamiltonian formulations, it appears that the direct Lagrangian version in which the state variables $\mathbf{x}(t_n)$ or $\mathbf{x}(l_n)$ are varied, the symplectic structure can be maintained and the boundary conditions on the canonical momentum respected (Marsden & West, 2001; Wendlandt & Marsden, 1997).

In practice, this means that the direct variational methods suggested for the ML problems taking into account model error ($R_f \neq \infty$) may skirt issues associated with backpropagation. This issue may be seen a bit more directly by comparing how one moves in $\{\mathbf{x}(t), \dot{\mathbf{x}}(t)\}$ space organized by equation 4.11 with the motion in $\{\mathbf{x}(t), \mathbf{p}(t)\}$ space guided by equation 4.17. These are equivalent motions of the model in time/layer, connected by a Legendre transformation from $\{\mathbf{x}(t), \dot{\mathbf{x}}(t)\} \to \{\mathbf{x}(t), \mathbf{p}(t)\}$.

In the Hamiltonian form, where $R_f \to \infty$ is the limit where one usually works, moving in regions where $\mathbf{DF}(\mathbf{x})$ may have saddle points may "slow down" the progression in the canonical momentum $\mathbf{p}(t)$. This may occur at a maximum, at a minimum, or at a saddle point of $\mathbf{DF}(\mathbf{x})$. At any of these, the observation in LeCun et al. (2015), "The analysis seems to show that saddle points with only a few downward curving directions are present in very large numbers, but almost all of them have very similar values of the objective function. Hence, it does not much matter which of these saddle points the algorithm gets stuck at," may apply. In the Lagrangian formulation, equation 4.11, the manner in which $\mathbf{DF}(\mathbf{x})$ enters the motion is quite

different and may well avoid this confounding property. We have pointed out that equation 4.17 is backprop. The use of the Lagrangian variational principle (Marsden & West, 2001; Wendlandt & Marsden, 1997) solves the same problem, so may have an unexpected virtue.

5 Summary and Discussion

This article has drawn a direct analogy between the formulation of a much utilized class of ML problems and a set of equivalent problems in DA as encountered in many physical, biological, and geoscience problems. Many engineering analyses where data-driven model development and testing is a goal are also in this category. The fundamental equivalence of the two inquiries is the core of this article.

The analogy allows us to identify methods developed in DA as potentially quite useful in ML contexts. In particular, the possibility of using variational annealing to produce the global minimum of the action (cost function) of the standard model of DA with both observation error and model error appears potentially of value.

The idea of making time continuous for purposes of exploring properties of DA suggests a similar tactic in ML. The ML step of making layers continuous we have called "deepest learning," as deep learning appears to result from increasing the number of layers. In the continuous layer (time) formulation, we see clearly that the problem to be solved is a two-point boundary value problem. This may lead to the construction and solution of tractable models that may helpfully illuminate how deep learning networks operate successfully and expand the possibilities of utilizing them employing additional methods for numerical calculations and interpretation.

In the formulation of the statistical DA problem at the general level expressed in equation 2.5, we see that the measurement error term, which is where information from data is passed to the model, it is explicitly information through Shannon's conditional mutual information that is being passed from observations to the model. This suggests that the idea that deep learning works because of major increases in computing power, as well as in having large data sets may have a sound basis; however, the attribute of the data sets is not so much that they are large but that they possess information, in a precise manner, that can be utilized to learn about models. The conjunction of information transfer and state and parameter estimation is embodied in the work of Rissanen (1989, 2007), where he identifies the cost of estimating a parameter or a state variable at some time. The arguments we have presented suggest evaluating how much information in a data set is available to inform a model is of greater utility than just the size of the data set itself.

One point not made explicit in the main text but worth noting is that once we have formulated the DA or ML problems as accurately performing high-dimensional integrals such as equation 2.6, the Laplace approximation

method, namely, the usual variational principle, permits the investigation of corrections through further terms in the expansion of the action about the path leading to the global minimum (Abarbanel, 2013). Ye, Rey et al. (2015) showed that corrections to this first approximation are small as R_f becomes large when analyzing the standard model. This need not be the case for other choices of noise distributions in the measurement error or model error terms in the action.

Another item of interest is the argument noted in LeCun et al. (2015) that as the dimension of a model increases, one may find fewer and fewer local minima confounding the search in path space for a global minimum, and in that situation many more unstable saddle points in path space will arise (Dauphin et al., 2014; Choromanska, Henaff, Mathieu, Arous, & LeCun, 2015).

In the case of a chaotic system such as the Lorenz96 model, the evidence is that however large the dimension of the model itself and the paths over which one may search, there are multiple local minima until the number of measurements at any observation time is large enough and the information transferred to the model is sufficient. The role of the number of model evaluations between observations, suggested in some of the arguments here, also play a significant part in establishing whether the action surface has many local minima.

The view of a deep network as moving from a few hidden layers to many may also be illuminated by our arguments. One idea is that by increasing the number of hidden layers, one is increasing the resolution in the analog of time in DA. When one does that in DA, we see it as probing the variation of the underlying model as it evolves from an initial condition through "layer" = "time." Missing the higher-frequency variations in time by employing a coarse grid in discrete time should have its counterpart role in the feedforward networks discussed here.

It is recognized that the "neuron" models widely used in ML applications have little in common with properties of biological neurons; the construction and implementation of large networks that have successful functions within ML may prove a useful guide for the construction and implementation of functional natural neural networks.

Finally, it is important to comment that while the analogy drawn and used here may improve the testing and validation of models supported by observational data, it does not assist in the selection of the models and their formulation. That is still a task to be addressed by the user.

Acknowledgments

Partial support from the MURI Program (N00014-13-1-0205), sponsored by the Office of Naval Research, is acknowledged, as is support for A.S. from the ARCS Foundation.

References

Abarbanel, H. D. I. (2013). *Predicting the future: Completing models of observed complex systems*. New York: Springer.

Allgower, E. L., & Georg, K. (1990). *Numerical continuation methods: An introduction*. Berlin: Springer. doi:10.1007/978-3-642-61257-2

Bennett, A. F. (1992). *Inverse methods in physical oceanography*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511600807

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, *16*(5), 1190–1208. doi:10.1137/0916069

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015). The loss surfaces of multilayer networks. In G. Lebanon & S. V. N. Vishwanathan (Eds.), *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics* (vol. 38, pp. 192–204). San Diego, CA: PMLR. http://proceedings.mlr.press/v38/choromanska15.html

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information Processing Systems*, *27* (pp. 2933–2941). Red Hook, NY: Curran.

Doya, K. (1992, May). Bifurcations in the learning of recurrent neural networks. In *Proceedings of the 1992 IEEE International symposium on Circuits and Systems* (vol. 6, pp. 2777–2780). Piscataway, NJ: IEEE. http://ieeexplore.ieee.org/abstract/document/230622/

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211. doi:10.1207/s15516709cog1402_1

Evensen, G. (2009). *Data assimilation: The ensemble Kalman filter*. Berlin: Springer. doi:10.1007/978-3-642-03711-5

Fano, R. M. (1961). *Transmission of information: A statistical theory of communication*. Cambridge, MA: MIT Press.

Gelfand, I. M., & Fomin, S. V. (1963). *Calculus of variations* (R. A. Silverman, Trans.). Mineola, NY: Dover.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, MA: MIT Press.

Haber, E., Ruthotto, L., Holtham, E., & Jun, S.-H. (2017). *Learning across scales: A multiscale method for convolution neural networks*. arXiv:1703.02009v2 [cs.NE].

Hairer, E., Lubich, C., & Wanner, G. (2006). *Geometric numerical integration: Structure-preserving algorithms for ordinary differential equations*. Berlin: Springer.

Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 531–546). Hillsdale, NJ: Erlbaum.

Kadakia, N., Rey, D., Ye, J., & Abarbanel, H. D. I. (2017). Symplectic structure of statistical variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, *143*(703), 756–771. doi:10.1002/qj.2962

Kostuk, M. (2012). *Synchronization and statistical methods for the data assimilation of HVc neuron models*. Ph.D. diss., University of California, San Diego. http://escholarship.org/uc/item/2fh4d086

Kot, M. (2014). *A first course in the calculus of variations*. Providence, RI: American Mathematical Society.

Landau, R. H., Paez, M. J., & Bordeianu, C. C. (2010). *A survey of computational physics: Introductory computational science*. Princeton, NJ: Princeton University Press.

Laplace, P. S. (1774). Memoir on the probability of causes of events. *Mémoires de Mathématique et de Physique, Tome Sixième* (pp. 621–656).

Laplace, P. S. (1986). Memoir on the probability of the causes of events (S. M. Stigler, Trans.). *Statistical Science*, *1*(3), 364–378.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444. doi:10.1038/nature14539

Liberzon, D. (2012). *Calculus of variations and optimal control theory*. Princeton, NJ: Princeton University Press.

Lorenz, E. N. (2006). Predictability: A problem partly solved. In T. Palmer & R. Hagedorn (Eds.), *Predictability of weather and climate.* Cambridge: Cambridge University Press.

Marsden, J. E., & West, M. (2001). Discrete mechanics and variational integrators. *Acta Numerica*, *10*, 357–514. doi:10.1017/S096249290100006X

Murty, K. G., & Kabadi, S. N. (1987). Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, *39*(2), 117–129. doi:10.1007/BF02592948

Parlos, A. G., Chong, K. T., & Atiya, A. F. (1994). Application of the recurrent multilayer perceptron in modeling complex process dynamics. *IEEE Transactions on Neural Networks*, *5*(2), 255–266. doi:10.1109/72.279189

Potember, R. (2017, January). *Perspectives on research in artificial intelligence and artificial general intelligence relevant to DoD* (Tech. Rep. JSR-16-Task-003). McLean, VA: JASON, MITRE Corporation. http://www.dtic.mil/docs/citations/AD1024432

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing* (3rd ed.). Cambridge: Cambridge University Press.

Rey, D. (2017). *Chaos, observability and symplectic structure in optimal estimation*. Ph.D. diss., University of California, San Diego. https://escholarship.org/uc/item/3049w2dh

Rissanen, J. (1989). *Stochastic complexity in statistical inquiry theory*. River Edge, NJ: World Scientific Publishing.

Rissanen, J. (2007). *Information and complexity in statistical modeling*. New York: Springer. doi:10.1007/978-0-387-68812-1

Wachter, A. R., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57. doi:10.1007/s10107-004-0559-y

Wendlandt, J. M., & Marsden, J. E. (1997). Mechanical integrators derived from a discrete variational principle. *Physica D: Nonlinear Phenomena*, *106*(3–4), 223–246. doi:10.1016/S0167-2789(97)00051-1

Ye, J., Kadakia, N., Rozdeba, P. J., Abarbanel, H. D. I., & Quinn, J. C. (2015). Improved variational methods in statistical data assimilation. *Nonlinear Processes in Geophysics*, *22*(2), 205–213. doi:10.5194/npg-22-205-2015

Ye, J., Rey, D., Kadakia, N., Eldridge, M., Morone, U., Rozdeba, P., . . . Quinn, J. C. (2015). Systematic variational method for statistical nonlinear state and parameter estimation. *Physical Review E*, *92*(5), 052901. doi:10.1103/PhysRevE.92.052901

Zhu, C., Byrd, R. H., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B: FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, *23*(4), 550–560.

Received October 2, 2017; accepted February 9, 2018.