

# Machine Learning for Forecasting and Data Assimilation

Troy Arcomano, Sarthak Chandra, Rebeckah Fussell,  
Michelle Girvan, **Brian Hunt**, Daniel Lathrop,  
Zhixin Lu, Edward Ott, Jaideep Pathak,  
Artur Perevalov, Ruben Rojas, Itamar Shani,  
Istvan Szunyogh, Alexander Wikner  
**University of Maryland** and **Texas A&M**

Partially funded by ARO, DARPA, and LTS

# Machine Learning

**Machine-learning** algorithms are **trained** with data to perform a particular task, such as **classification**.

Starting from **generic** input-output equations, training means choosing **parameter values** that minimize the difference between the **actual** outputs and the **desired** outputs (“supervised” learning).

In most applications, the task is **static training data** is a set of input-output pairs.

**Reservoir computing** is a type of machine learning well suited to **dynamic** tasks, mapping an input **time series** to an output time series.

# Forecasting and Attractor Reconstruction

We seek to use **reservoir computing** to create an **ad hoc** forecast model, based **only** on a finite-time sample (**training data**) from a dynamical system.

We are interested in two tasks for our **forecast model**:

Forecast “**weather**”: Predict future measurements (only feasible in short-term for chaotic systems).

**Learn “climate**”: Reproduce long-term properties of a chaotic attractor, such as Lyapunov exponents.

# Reservoir Computing

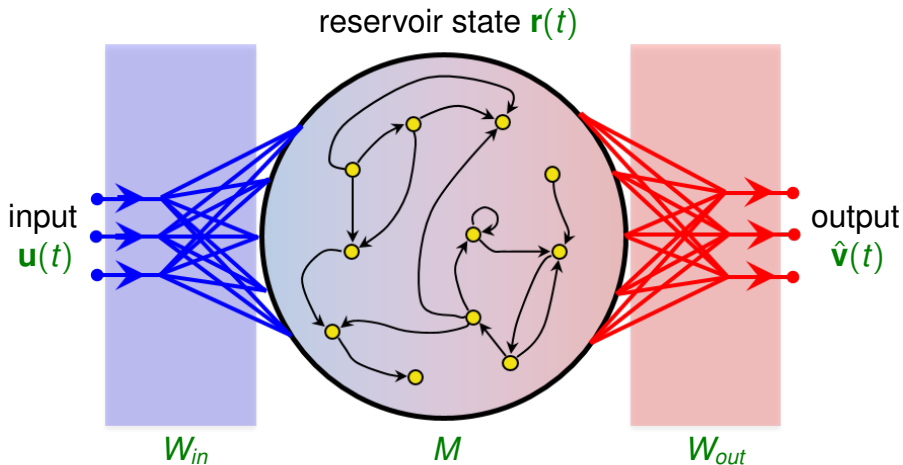
A **reservoir** is a **driven dynamical system** whose internal parameters are **not** adjusted to fit the training data only a linear post-processor is trained.

**Train** the reservoir by driving it with an input time series  $\mathbf{u}(t)$  and fitting a linear function of the reservoir state  $\mathbf{r}(t)$  to a desired output time series  $\mathbf{v}(t)$ .

This approach was proposed as Echo State Networks (Jaeger 2001) and Liquid State Machines (Maass, Natschlaeger, Markram 2002) see [http://www.scholarpedia.org/article/Echo\\_state\\_network](http://www.scholarpedia.org/article/Echo_state_network)

A reservoir can be a (super-fast) hardware device.

# Reservoir During Training



Matrices  $W_{in}$  and  $M$  are chosen randomly in advance

# Continuous-Time Jaeger ESN

**Listen:**  $\beta d\mathbf{r}/dt = \mathbf{r}(t) + \tanh[M\mathbf{r}(t) + W_{in}\mathbf{u}(t) + \mathbf{c}]$   
(in software, solve with Euler time step  $\tau$ ).

We choose  $M$  and  $W_{in}$  at random, scaling  $M$  to have spectral radius close to 1 and scaling  $W_{in}$  so that  $W_{in}\mathbf{u}$  is of order 1.

We choose  $\beta$  commensurate with the input time scale and we listen for  $0 \leq t \leq T$ , where  $T$  is the duration of the training time series  $\mathbf{u}(t)$ .

**Fit:** Find the matrix  $W_{out}$  such that  $\hat{\mathbf{v}}(t) = W_{out}\mathbf{r}(t)$  least-squares minimizes the residuals  $\hat{\mathbf{v}}(t) - \mathbf{v}(t)$  for  $0 \leq t \leq T$ .

## Inference Task (DA w/o Model)

Suppose that we can inexpensively measure some state variables of a dynamical system, but other variables of interest are difficult to measure.

Let  $\mathbf{u}(t)$  consist of state variables that can be measured for all time, and  $\mathbf{v}(t)$  consists of state variables that are only measured for  $0 \leq t \leq T$ .

After training, we continue to evolve the **listening equation** for  $t \leq T$ , attempting to **infer**  $\mathbf{v}(t)$  from  $\mathbf{u}(t)$ .

The estimated value of  $\mathbf{v}(t)$  is  $\hat{\mathbf{v}}(t) = W_{out}\mathbf{r}(t)$ .

## Example: Lorenz System

We ran the Lorenz system with time step  $\tau = 0.05$ :

$$\frac{dx}{dt} = 10(y - x), \quad \frac{dy}{dt} = x(28 - z) - y, \quad \frac{dz}{dt} = xy + 8z/3.$$

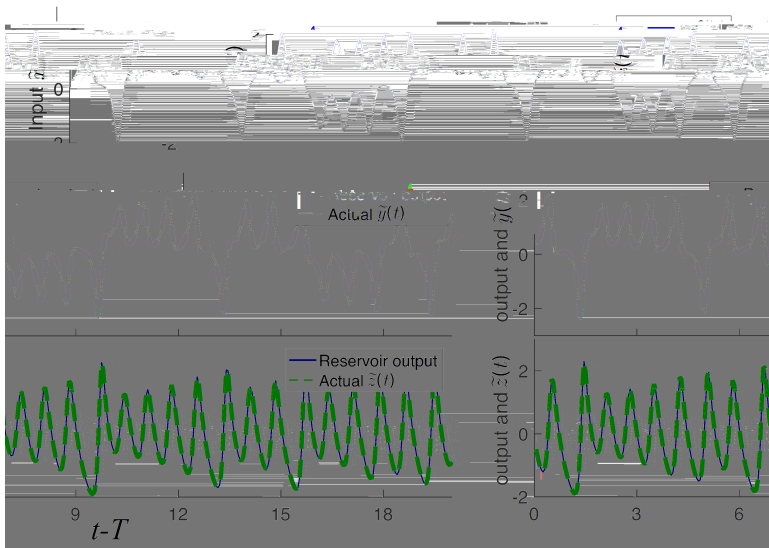
For training, we used  $\mathbf{u}(t) = [x(t)]$  and  $\mathbf{v}(t) = [y(t), z(t)]^T$ .

We trained a 400-node reservoir (meaning that the dimension of  $\mathbf{r}(t)$  is 400) for training time  $T = 200$ .

Details are in (Lu et al., Chaos, 2017).



# Inferring Lorenz $y(t)$ and $z(t)$ from $x(t)$



# Forecasting with Feedback

Suppose training data is sampled at time interval  $\tau$ , but we want to forecast farther than  $\tau$  into the future.

One option is to train with lead time  $n\tau$  [that is, set  $\mathbf{v}(t) = \mathbf{u}(t + n\tau)$ ] for the value(s) of  $n$  of interest.

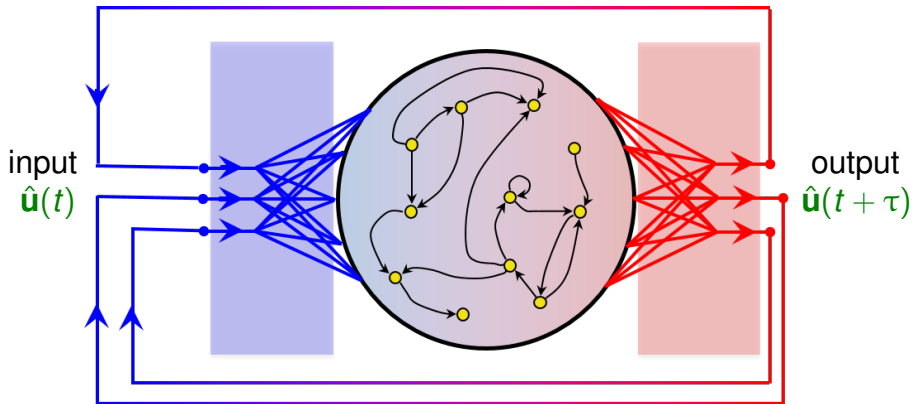
We've gotten better results by training with lead time  $\tau$  and **iterating** the trained time- $\tau$  forecast  $n$  times.

We train with desired output  $\mathbf{v}(t) = \mathbf{u}(t + \tau)$  and then forecast with  $\mathbf{u}(t)$  replaced with  $\hat{\mathbf{u}}(t)$ .

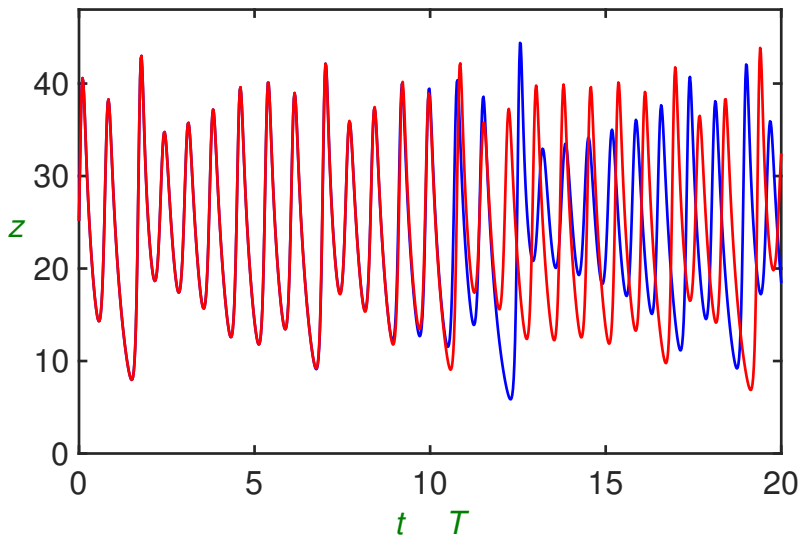
This **feedback** approach (Jaeger & Haas 2004) can be used with other machine-learning methods.

## Forecast Model for $t > T$

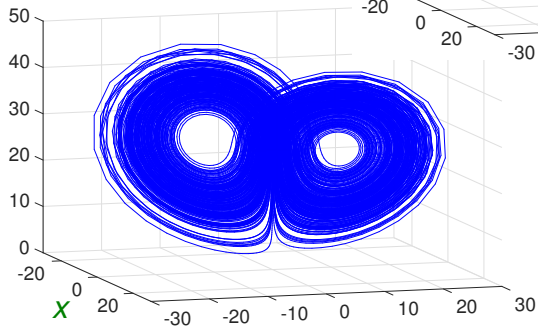
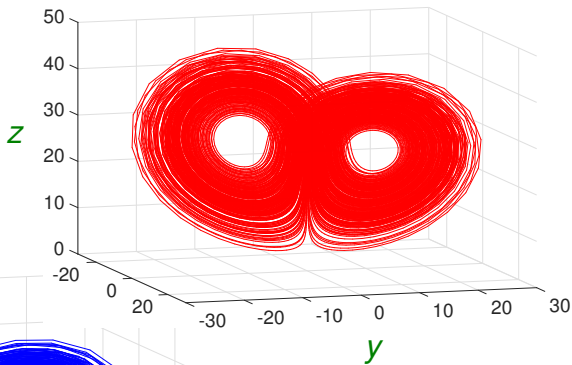
Predict:  $\beta d\mathbf{r}/dt = \mathbf{r}(t) + \tanh[M\mathbf{r}(t) + W_{in}\hat{\mathbf{u}}(t)]$   
 $\hat{\mathbf{u}}(t) = W_{out}\mathbf{r}(t - \tau)$



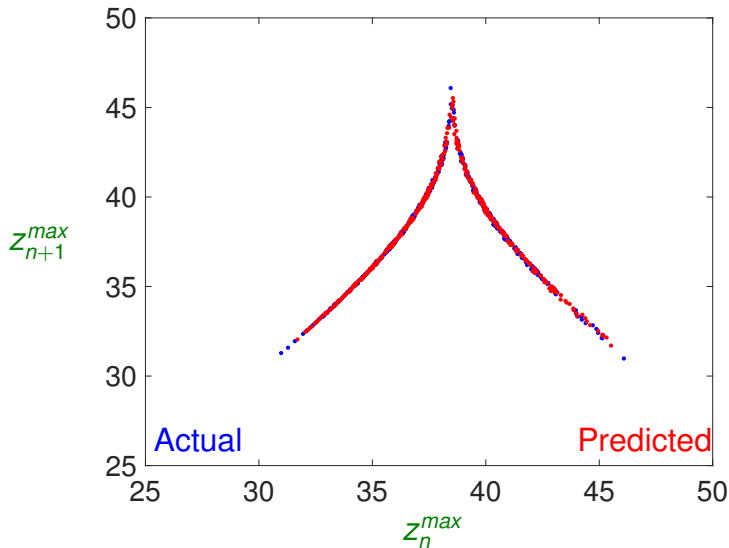
# Lorenz system: Actual and Predicted $z(t)$



# Actual and Predicted Attractors



# Poincaré Section of Successive $z(t)$ Maxima



## Questions Raised

How can a reservoir “learn” these dynamics? Is there an approximate copy of the Lorenz attractor in the high-dimensional  $\mathbf{r}(t)$  system that we chose independently of the training data?

No – the feedback term in our forecast model depends on the matrix  $W_{out}$  determined from training.

We have found an approximation to the Lorenz system in a family parametrized by  $W_{out}$ , which has more than 1000 entries.

But how is it feasible to find appropriate parameter values in practice? Partial theory to follow.

# Theoretical Framework

During training/listening, the reservoir and its input form a “skew product” or “drive-response” system:

**Input Dynamics (Drive):**  $\mathbf{u}(t + \tau) = \mathbf{f}[\mathbf{u}(t)]$

**Listening (Response):**  $\mathbf{r}(t + \tau) = \mathbf{g}[\mathbf{r}(t), \mathbf{u}(t)]$

Assume that  $\mathbf{f}$  and  $\mathbf{g}$  are continuous on Euclidean spaces, that  $\mathbf{f}$  is invertible, and that  $\mathbf{u}$  lies in a compact attracting set  $A$ .

If  $\mathbf{g}$  is uniformly contracting w.r.t.  $\mathbf{r}$ , then as  $t \rightarrow \infty$ , the reservoir state  $\mathbf{r}(t)$  becomes independent of its initial state (roughly, Jaeger’s “echo state property”).



# Generalized Synchronization

Furthermore, we get **generalized synchronization**: there is a continuous function  $\phi$  on  $A$  such that  $\mathbf{r}(t) = \phi(\mathbf{u}(t)) + \mathbf{0}$  as  $t \rightarrow \infty$  [short proof in (Stark 1997)].

Asymptotically, the reservoir state  $\mathbf{r}(t)$  is a function of the current input  $\mathbf{u}(t)$  only, **not the entire history** of  $\mathbf{u}$ . However, we don't know the function  $\phi$  in practice.

Uniform contraction can be guaranteed by choice of  $\mathbf{g}$ , but strong contraction may inhibit extraction of  $\mathbf{u}$  from  $\mathbf{r}$  [extreme case: if  $\mathbf{g}$  is identically  $\mathbf{0}$ , then so is  $\mathbf{r}$ ].

# Inverting the Synchronization Function

If  $\mathbf{r}$  is much higher-dimensional than the attractor  $A$ , then embedding theory (Sauer-Yorke-Casdagli 1991) suggests that  $\phi$  is likely to be one-to-one on  $A$ .

If so, the inverse of  $\phi$  on the set  $\phi(A)$  can be extended to  $\mathbf{r}$ -space in many ways.

**Fitting:** In training, we attempt to find a linear function  $W_{out}$  that approximately inverts  $\phi$  on  $\phi(A)$ :

$$\mathbf{u}(t) \approx W_{out} \mathbf{r}(t) \approx W_{out} \phi(\mathbf{u}(t)).$$

This does **not** require  $\phi$  to be approximately linear. It requires only that we can approximate the nonlinear function  $\phi^{-1}$  on a low-dimensional set by a linear function on a high-dimensional space.

# Attractor Reconstruction and Stability

If training is successful, then our **forecast model**

$$\mathbf{r}(t + \tau) = \mathbf{g}[\mathbf{r}(t), W_{out}\mathbf{r}(t)] \quad (1)$$

approximates [on  $\phi(A)$ ] the **idealized model**

$$\mathbf{r}(t + \tau) = \mathbf{g}[\mathbf{r}(t), \phi^{-1}(\mathbf{r}(t))]. \quad (2)$$

Generalized synchronization implies that system (2) is conjugate to true dynamics  $\mathbf{u}(t + \tau) = \mathbf{f}(\mathbf{u}(t))$  on  $A$ .

To reproduce the climate of  $A$  **in practice**, we need system (1) to approximate an extension of system (2) that makes  $\phi(A)$  attracting.

More “theory” in (Lu et al., Chaos 2018).

# Kuramoto-Sivashinsky PDE

We tested our methods on the spatiotemporally chaotic KS system

$$u_t = -uu_x - u_{xx} - u_{xxxx}$$

with periodic boundary condition  $u(x + L, t) = u(x, t)$ .

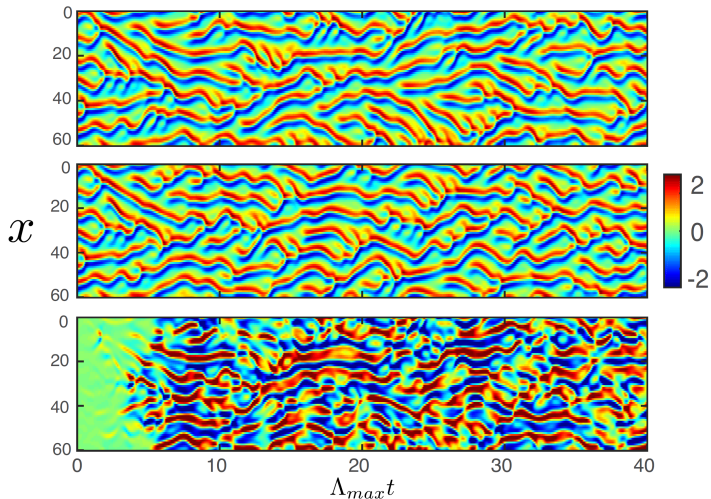
With system size  $L = 60$ , we trained a 9000-node reservoir for time  $T = 20000$  to predict the (numerical) KS solution.

The largest Lyapunov exponent is  $\Lambda_{max} = 0.1$ , so we trained for roughly 2000 Lyapunov times.

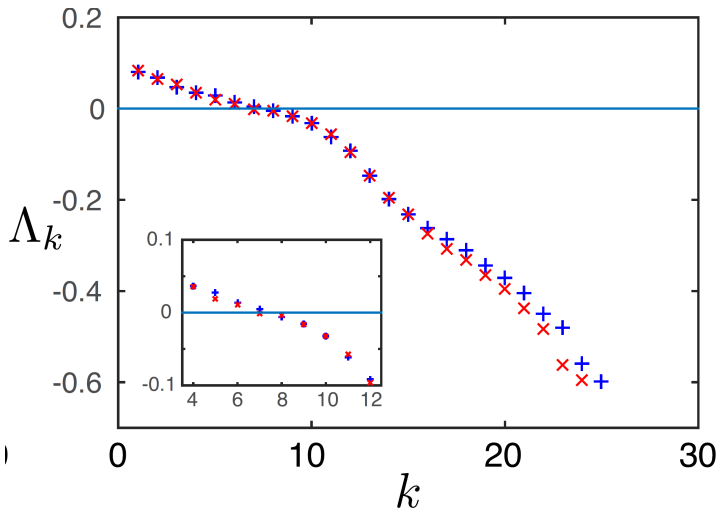
Details are in (Pathak et al., Chaos 2017).

# Kuramoto-Sivashinsky Forecast

Top: "Truth"    Middle: Reservoir    Bottom: Error



# Estimation of Lyapunov Exponents



Actual exponents

Reservoir exponents

# Hybrid Forecasting

Suppose we have an **imperfect knowledge-based** forecast model for a physical system.

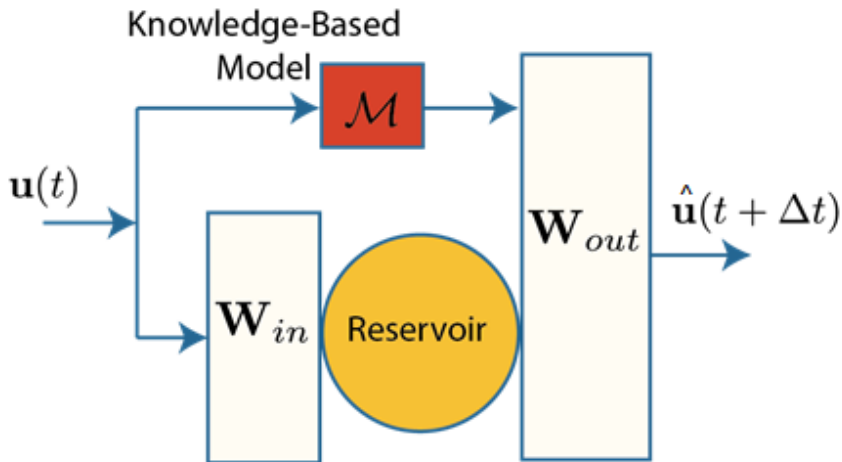
A **hybrid** method uses machine learning to **improve** (rather than **replace**) the model.

We train by feeding the same input to the **model** and the **reservoir**, and optimize a linear combination of the model output and the reservoir output.

We tested on the KS system with  $L = 35$ , using an imperfect model that replaces  $u_{xx}$  with  $(1 + \varepsilon)u_{xx}$ .

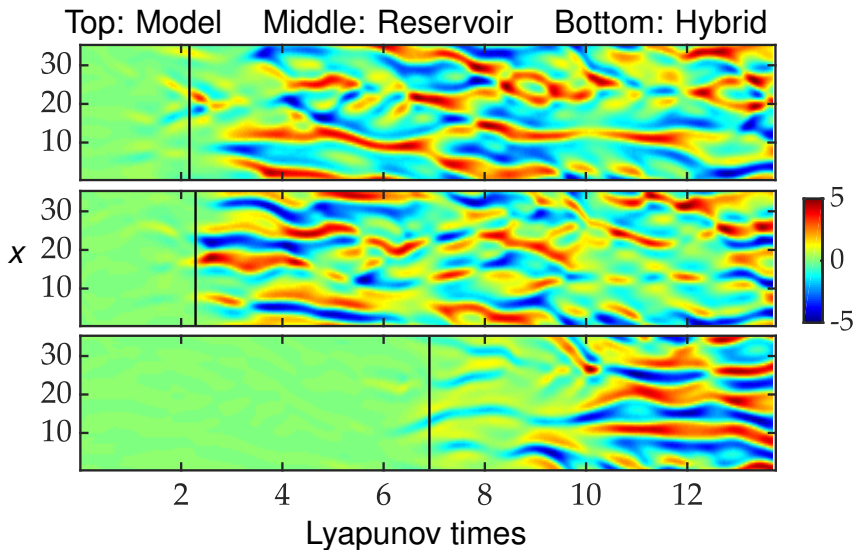
Details in **(Pathak et al., Chaos 2018)**.

# Hybrid Architecture



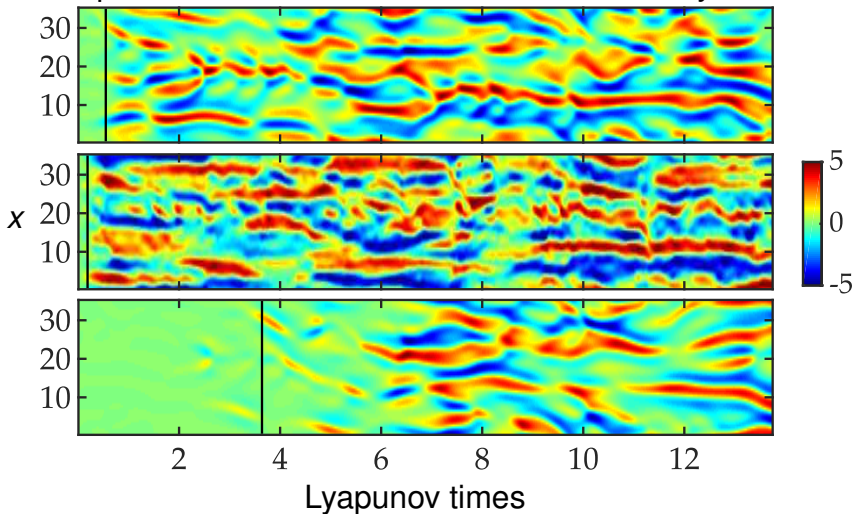


# Forecast Errors: 8000 nodes, $\varepsilon = 0.01$



# Forecast Errors: 500 nodes, $\varepsilon = 0.1$

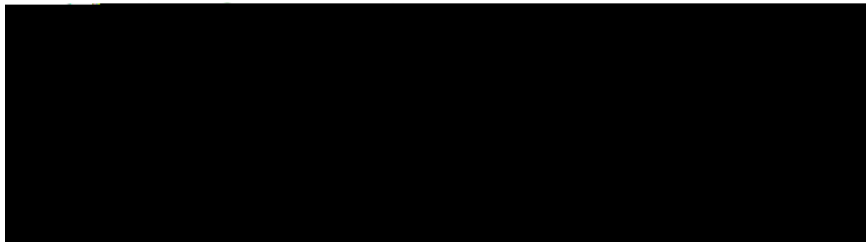
Top: Model      Middle: Reservoir      Bottom: Hybrid



# Parallel Reservoir Architecture

We developed a method for forecasting high-dimensional, spatially extended systems with multiple reservoirs that can process in parallel (Pathak et al., PRL 2018).

Each reservoir forecasts on a local region based on input from its own and neighboring regions.



# Preliminary Results with a GCM

We are currently developing with T. Arcomano and I. Szunyogh (Texas A&M) a **parallel hybrid** code for the SPEEDY model (Molteni 2003) from ICTP.

Model grid is T30 (96 48) with 8 vertical levels.

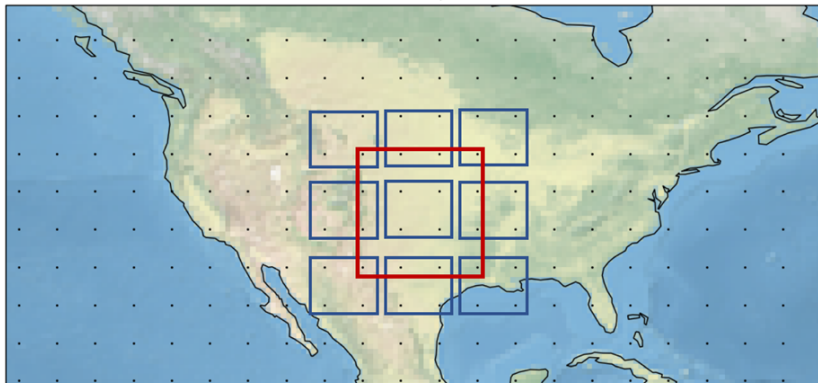
We interpolated the ERA5 reanalysis (ECMWF) to the SPEEDY grid for training and verification data.

**Preliminary** results are from a **parallel-only** code (no hybrid yet) using 9 years of hourly data for training.

We added noise to the reservoir input during training to improve stability.

# Parallel Regions on SPEEDY Grid

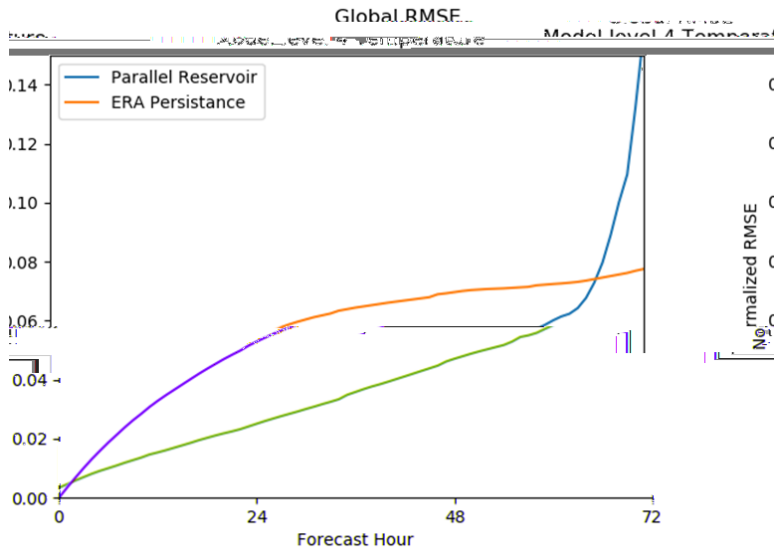
Domain Decomposition



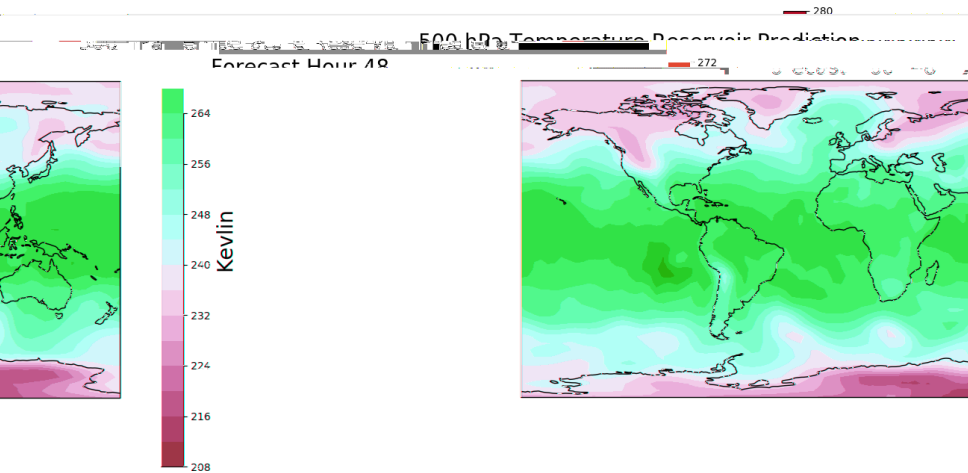
Red: Input Region

Blue: Output Region

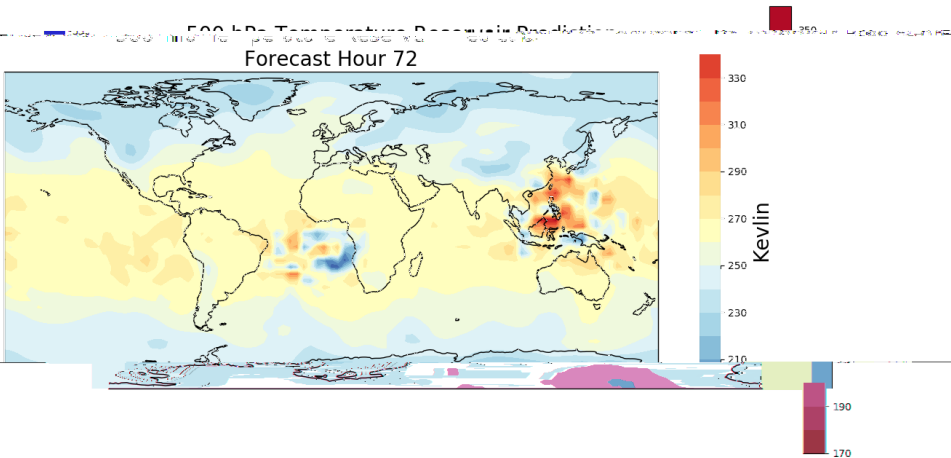
# Comparison with Persistence Forecast



# 48-hour Machine-Learning Forecast



# 72-hour Machine-Learning Forecast





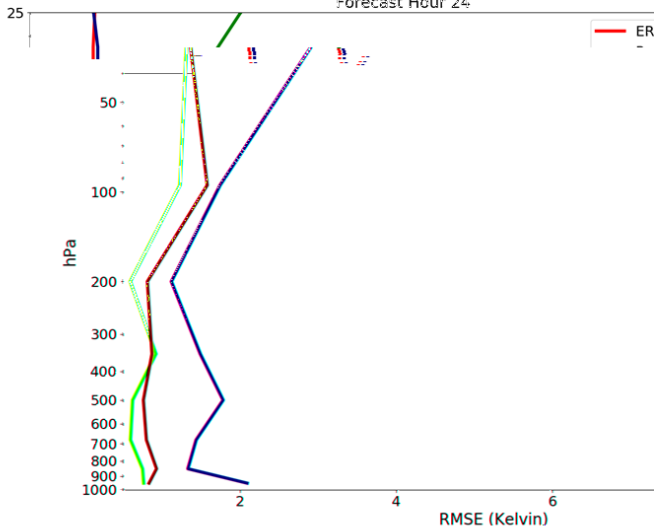
# 24-hour Forecasts: Tropics

A Persistence

Parallel Reservoir  
SPEEDY

Tropical Temperature RMSE

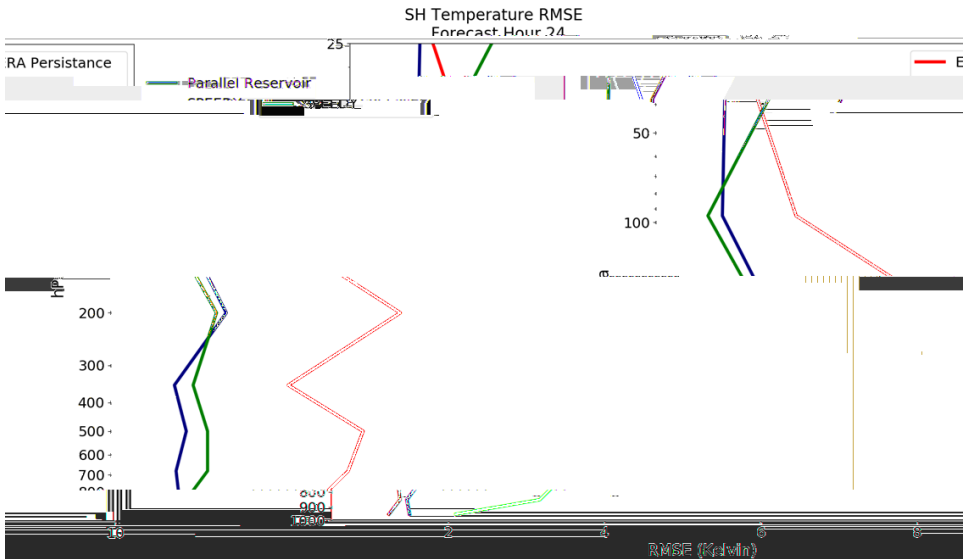
Forecast Hour 24



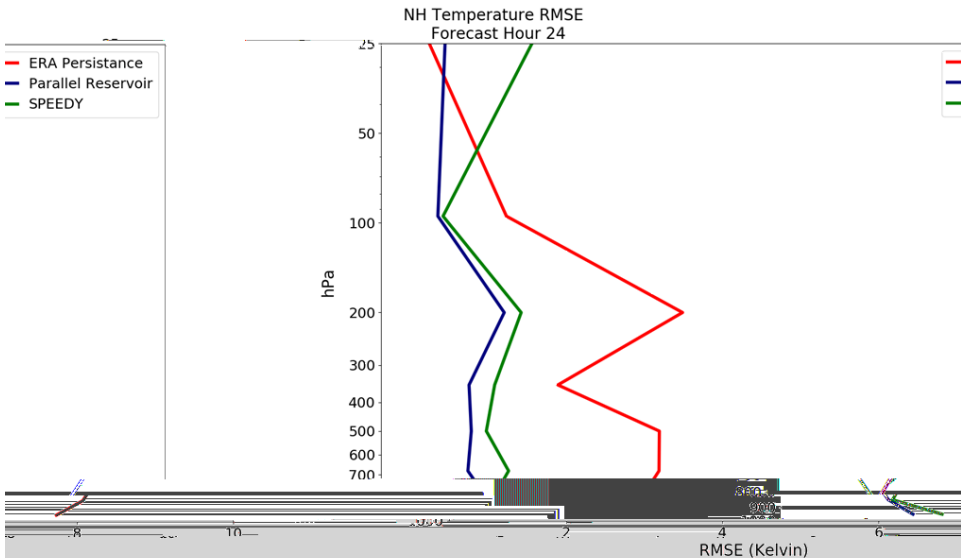
8 10

RMSE (Kelvin)

# 24-hour Forecasts: SH



# 24-hour Forecasts: NH



## Further directions

We are developing and testing various approaches to combine our hybrid method with **data assimilation**.

Training directly on observations rather than on reanalysis is a challenge.

One goal is to perform adaptive (“online”) training of the machine-learning component as part of the data assimilation cycle.

Our colleagues Dan Gauthier (Ohio State) and Dan Lathrop (Maryland) are using FPGAs and ASICs to create hardware reservoirs that are vastly faster than software implementations.

# Concluding Remarks

Reservoir forecasting with a **feedback loop** is relatively simple to implement and capable of learning the dynamics behind chaotic time series from a modest amount of training data.

**Hybrid** approach uses machine learning to improve (not replace) an imperfect knowledge-based model.

**Parallel** method scales to high-dimensional spatiotemporal systems by considering only local interactions.

Simplified training relative to other machine-learning methods makes reservoir computing attractive for **hardware** implementations.

# Data from Magnetohydrodynamic Experiment

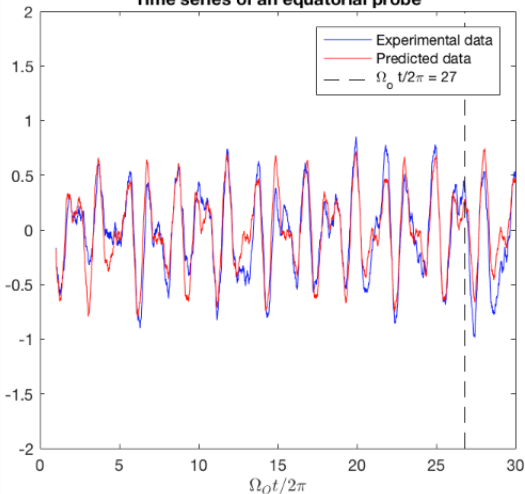
Dan Lathrop's lab at U.Md. has a 3-meter diameter rotating sphere filled with liquid sodium, with an internal counter-rotating sphere and an externally applied magnetic field.

We attempted to predict time series data from 33 sensors, 31 of which are on the outer sphere.

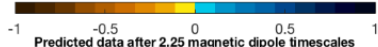
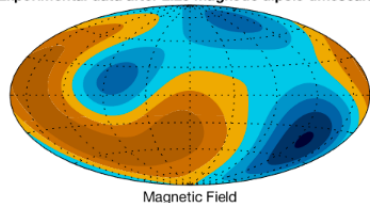
The duration of the training data was about 150 rotations of the sphere.

# Prediction of Experimental Data

Time series of an equatorial probe



Experimental data after 2.25 magnetic dipole timescales



Predicted data after 2.25 magnetic dipole timescales

