MACHINE LEARNING STRATEGIES FOR PREDICTING EXTREME EVENTS AND ANOMALOUS STATISTICS WITH UNOBSERVED DYNAMICS

We study the data-driven prediction strategies to recover dynamical systems with unobserved hidden dynamics using unambiguous test models. The main features to recover are extreme events and the related anomalous statistics which are fascinating phenomena observed in a wide class of complex systems. The main difficulties in accurate quantification and prediction of such extreme events include the lack of a complete understanding of the background physical process, and the expensive computational overload to directly run such complex dimensional systems. Practically, the available data for training is often limited with partial observation and polluted with measurement errors. Recurrent neural networks (RNNs) keep track of a hidden state, that encodes information about the history of the high dimensional spatio-temporal dynamical systems.

1. Topographic Barotropic Model with Interacting Large Mean Flow and Small-Scale Feedback

- The topographic barotropic model with complex interaction between a large-scale zonal flow U and small-scale fluctuations

$$\frac{\partial q}{\partial t}+\nabla^{\perp}\psi\cdot\nabla q+U\frac{\partial q}{\partial x}+\beta\frac{\partial \psi}{\partial x} = \mathcal{D}(\Delta)\psi+F_q, \tag{1a}$$

$$\frac{dU}{dt}+\oint\frac{\partial h}{\partial x}\psi = -d_U U+\sigma_U\dot{W}_0, \tag{1b}$$

  with $q=\omega+h=\nabla^2\psi+h$.
- The passive tracer field $T=\alpha y+T'$ with a mean gradient α along the y direction and a simplified advection flow field $\mathbf{v}=(U(t),v(x,t))$

$$\frac{\partial T'}{\partial t}+U\frac{\partial T'}{\partial x}=-d_T T'+\kappa\frac{\partial^2 T'}{\partial x^2}-\alpha v(x,t). \tag{2}$$

  Above the cross-sweep U and shear flow v can be modeled by the topographic barotropic model solution (1).

An interesting special case for the topographic model (1) is to assume a layered topography and stream function in the expansion form

$$h(x,y)=\sum_k \hat{h}_k e^{ik\mathbf{l}\cdot\mathbf{x}},\quad \omega(x,y,t)=\sum_k \hat{\omega}_k(t)e^{ik\mathbf{l}\cdot\mathbf{x}}, \tag{3}$$

along one characteristic wavenumber unit direction $\mathbf{l}$ with $|\mathbf{l}|=1$ and $\hat{\omega}_k=-k^2\hat{\psi}_k$. The corresponding full velocity field can be found combining the zonal mean flow U and the small-scale fluctuations

$$\mathbf{u}=(U+u',v')=\left(U+il_y\sum_k k^{-1}\hat{\omega}_k e^{ik\mathbf{l}\cdot\mathbf{x}},-il_x\sum_k k^{-1}\hat{\omega}_k e^{ik\mathbf{l}\cdot\mathbf{x}}\right). \tag{4}$$

Then nonlinear coupling term, $\nabla^{\perp}\psi\cdot\nabla q$, vanishes under the above layered topography expansion. The original equations (1) can be reformulated in the following form for each Fourier spectral mode

$$\begin{aligned}\frac{d\hat{\omega}_k}{dt}+il_x k\left(U-k^{-2}\beta\right)\hat{\omega}_k+il_x k\hat{h}_k U&=-d_k\hat{\omega}_k+\sigma_k\dot{W}_k,\\ \frac{dU}{dt}-l_x\sum_k k^{-1}\mathfrak{Im}\left(h_k^*\hat{\omega}_k\right)&=-d_U U+\sigma_U\dot{W}_0.\end{aligned} \tag{5}$$

In this model (5), the large-scale mean flow U is driven by the topographic stress from h combining all the small-scale feedbacks, while the mean flow also inversely contributes to each small-scale spectral mode through the nonlinear advection and topographic effect. The layered topographic model enables us to focus on the coupling effect between small and large scales through the topographic stress. On the right hand side of (5), the states are subject to linear

damping and random forcing as a white noise process. Further, given the realization of the large-scale process U, the first equation (5) has the conditional Gaussian structure as described in the general formulation.

Similarly, we can introduce the passive tracer model based on the same formulation in the previous section (2). Here the zonal cross-sweep U and the small-scale fluctuations v can be adopted from the topographic barotropic model solution in (5). The same layered structure (3) can be also assumed for the tracer state so that we consider the tracer solution in the following structure.

$$T(x,y,t) = \sum_k \hat{T}_k(t)\, e^{ik\mathbf{l}\cdot\mathbf{x}}.$$

Then the small-scale velocity field $\mathbf{v} = \nabla^{\perp}\psi$ will give no contribution to the tracer advection term, and the tracer field is following the same dynamics as in (2)

$$d\hat{T}_k = \left(-\gamma_{T,k} + i\omega_{T,k}(t)\right)\hat{T}_k dt - \alpha\hat{v}_k dt, \tag{6}$$

with the tracer dissipation and dispersion coefficients for each mode

$$\gamma_{T,k} = d_T + \kappa k^2, \quad \omega_{T,k} = -kU(t).$$

The zonal mean flow U acting as the zonal cross-sweep and $\hat{v}_k = -il_x k^{-1}\hat{\omega}_k$ as the shear flow are both from the direct solution of the topographic model (5).

2. Machine learning strategies for capturing the presentative statistical features and extreme events

We summarize the general plan to be carried out by using the various machine learning strategies to learn the unresolved model dynamics and predict the solutions among various statistical regimes. The main task here is to develop effective machine learning techniques to learn the complex dynamics with multiscale interacting spatiotemporal scales, from partial data (only the large-scale flow U) and the various hidden unobserved small-scale processes. We can consider two approaches for using machine learning techniques to predict model solutions and statistics:

- **i):** A neural network to learn the unresolved small-scale dynamics, and directly predict the single trajectory solutions;
- **ii):** A combination of the neural network and conditional Gaussian framework to recover the crucial statistical structures and statistics with skewed PDFs.

2.1. General data-driven model construction for capturing extreme events.

In the model construction, we assume that: i) the deterministic mean flow dynamics with all small-scale feedback $\sum_k h_k^*\hat{v}_k$ is know; ii) the unresolved small-scale dynamics for $\hat{v}_k$ are directly learned from proper neural network; iii) in addition, the random white noise $\sigma_U\dot{W}$ can be treated as additional model parameter to be recovered in the training process.

2.1.1. *Formulation of the machine learning problem.* We consider the two-mode barotropic topographic model (5) decomposed into the following observed (known) large-scale zonal flow equation with (unknown) white noise forcing and the unresolved (unknown) small-scale fluctuation processes with topographic feedback to the mean flow equation:

- The observed zonal mean state U follows the dynamical equation

$$\frac{dU}{dt} = \sum_k h_k^*\hat{v}_k - d_U U + \sigma_U\dot{W}_U. \tag{7}$$

  The equation is also subject to the white noise forcing process. It can be also treated as an additional noise added to the observed data. We assume the deterministic structures $\sum_k h_k^*\hat{v}_k - d_U U$ known, and treat the white noise $\sigma_U\dot{W}_U$ also as one unknown process to be learnt from the data;
- The unobserved process $\hat{v}_k$ from small scale fluctuations follows the equation:

$$\frac{d\hat{v}_k}{dt} = F_{v,k}(U(\cdot)) = i\omega_{v,k}(U)\hat{v}_k - \hat{h}_k U - d_{v,k}\hat{\omega}_k + \sigma_{v,k}\dot{W}_k, \tag{8}$$

  The detailed dynamics $F_{v,k}$ for each wavenumber k is to be directly learned from the data.

Above, we always treat the large scale mean flow U as the 'observed' process with known dynamics, while the detailed structures in the small scale shear flow mode $\hat{v}_k$ (as well as the passive tracer T) are the 'hidden' processes to be learnt from the deep neural network .

From the LSTM network to shown next, the observed process U can be treated as the input data $\mathbf{x}_t$, while the unobserved shear flow and tracer field can be predicted as the hidden output of the network $\mathbf{h}_t$. In the learning method for the future state of the observed process U, two unknown processes need to be recovered to solve the equation (7): i) the unresolved small scale modes $\hat{v}_k$; and ii) the stochastic white noise $\dot{W}_U$. In the following learning procedure, the dynamics for the unresolved processes $F_{v,k}$ are learnt from the neural network; and the random noise W_U can be treated as additional unknown model parameters to be learned through the training process.

2.1.2. *Data-driven model framework for prediction.* In the first direct approach, deep neural network structures can be designed to learn the functional $F_{v,k}$, that is,

$$\vec{v}^M(t) = [\hat{v}_1, \cdots, \hat{v}_\Lambda] = \mathrm{NN}_v\left([U_{t-m}, U_{t-m+1}, \cdots, U_{t-1}]\right), \tag{9}$$

with NN_v the data-driven approximation for the unresolved discrete-time flow dynamics F_v. The input data is the time-series from the history of large-scale mean flow U, with the time lag m. The history of the input data represented by the time lag m can be determined determined by the decorrelation time T_{corr}, or it can simply become another hyperparameter of the network. And the white noise forcing is treated as additional unknown parameters in the model

$$\vec{W}^M(1:t) = \left[\xi_1^M, \cdots, \xi_n^M\right]. \tag{10}$$

The output $\vec{v}^M = \left\{\hat{v}_k^M\right\}$ is the vector of spectral modes $k = 1, \cdots, \Lambda$ for all the small-scale state at the next time step t. Therefore, the entire unresolved model dynamics in (8) is to be recovered entirely from the neural network. With the prediction of all the shear flow mode $\hat{v}_k^M(t)$, the zonal mean flow solution at the next time instant can be updated directly from the model dynamical equation

$$U_{n+1}^M = U_n^M + \Delta t \sum_k h_k^* \mathrm{NN}_{v,k}\left(U_{n-m:n}^M\right) - d_U U^M + \sigma_U \xi_n^M,$$

where the unresolved small scales $\hat{v}_k^M$ are learned from the above neural network (9). Further, we may consider higher-order time integration scheme for the updating of U.

Finally, with the time-series of the solutions in U^M and $\hat{v}_k^M$ at $t-m, \cdots, t$, we can use the neural network to learn the dynamics of the passive tracer F_T, that is,

$$\hat{T}_k^M(t) = \mathrm{NN}_T\left([U_{t-m}, U_{t-m+1}, \cdots, U_t]; [\hat{v}_{t-m}, \hat{v}_{t-m+1}, \cdots, \hat{v}_t]\right). \tag{11}$$

Again from the previous analysis, we see that the solution of the tracer should be fully determined by the time-series in the zonal cross-sweep and shear flow trajectories. The question is whether the neural network can learn the typical tracer dynamics purely from data. In the regime with occurrence of strong extreme events, it is interesting to check the improvement by adopting the relative entropy cost function as well as the partition function structure.

2.2. **Detailed structures in the deep learning networks.** Recurrent neural networks (RNNs) offer the desirable structure to incorporate temporal processes of sequential data, and keep tracking of hidden processes. The *long short-time memory* (LSTM) network is a special RNN that is useful to recover the time-series including very long time correlations. The LSTM designed to learn the multi-scale temporal structures overcoming the problem of vanishing gradients. In the computational cell of the LSTM network, it consists of the basic building cell as

$$\begin{aligned}
f_t =& \sigma_g\left(W_f x_t + U_f h_{t-1} + V_f c_{t-1} + b_f\right),\\
i_t =& \sigma_g\left(W_i x_t + U_i h_{t-1} + V_i c_{t-1} + b_i\right),\\
c_t =& f_t \otimes c_{t-1} + i_t \otimes \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right),\\
o_t =& \sigma_g\left(W_o x_t + U_o h_{t-1} + V_o c_t + b_o\right),\\
h_t =& o_t \otimes \tanh\left(c_t\right).
\end{aligned} \tag{12}$$

Above, the $\sigma_g = \frac{1}{1+e^{-x}}$ is the sigmoid activation function, and $\otimes$ represents the element-wise product. The model cell includes forget, input, and output gates f_t, i_t, o_t, and the cell state c_t. The hidden process $\{h_{t-m}, \cdots, h_{t-1}, h_t\}$ represents the time-series of the unresolved process. The final output data is given by a final linear layer $y_t = W_x h_t$ applying on the final state of the hidden process.

2.2.1. *Connections in the neural network structure.* Different structures (fully connected or convolutional) can be exploited for the detailed neural networks for the LSTM cell (12). The above neural networks enjoy the advantages of universal approximation ability, and skill to capture both temporal and spatial dependencies. Thus they become especially fit for the data-driven predictions of complex turbulent systems. The neural networks can be designed in the predictions of different components of the test models:

- The LSTM network can be used to learn the unresolved or missing processes in the conditional Gaussian framework. A series of LSTM cells can be linked linearly for the time-series prediction. This can be tested on the multiscale test model with scale separation;
- The MS-D structure can be used to improve the performance of the neural network model. The neural network structures in the cell can include all the previous history information through the MS-D structure to add more history information to produce the prediction in the next step. Different weights for the cells can be assigned according to the the decorrelation time.

2.2.2. *Loss function and optimization process.* In the training process, the model parameters are achieved through the optimization for the proper loss functions (or cost). The direct choice of the loss function is by using the square L_2 norm

$$\mathcal{L}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{m} w_i \left\| F_{\boldsymbol{\theta}}^i(\mathbf{x}) - \mathbf{y}^i \right\|_2^2 + \gamma \|\mathbf{W}\|^2 + \beta \|\boldsymbol{\theta}\|^2 . \tag{13}$$

Above F^i is the push forward operator from the model for the time i steps ahead. The weight w_i can be determined by the autocorrelation functions of the corresponding mode with a proper decay rate. The second term is the control on the additional white noise also as parameters, and the third term is a penalty for all the model parameters $\boldsymbol{\theta}$. It appears that penalizing the parameter weights of the neural network avoids overfitting and is useful for larger neural networks.

Further, in the regimes with stronger extreme events and many small-scale fluctuations, it is useful to use the *Relative entropy loss* distance to replace the L_2 norm in ()13

$$\mathcal{L}_{\mathrm{KL}}(\mathbf{x},\mathbf{y}) = \frac{1}{M}\sum_{j=1}^{M} L_j, \quad L_j\left(\mathbf{x}^j,\mathbf{y}^j\right) = \sum_i \tilde{y}_i^{(j)} \log \frac{\tilde{y}_i^{(j)}}{\tilde{x}_i^{(j)}}, \tag{14}$$

where the superscript j represents the mini-batch members and the subscript i goes through the dimensions of the variable. The following two sets of positive and negative temperatures are added to rescale the data from the partition functions

$$\tilde{x}_i^+ = \frac{\exp(x_i/T_+)}{\sum_i \exp(x_i/T_+)}, \quad \tilde{x}_i^- = \frac{\exp(-x_i/T_-)}{\sum_i \exp(-x_i/T_-)},$$

where $T_+ > 0, T_- > 0$ are the positive and negative temperatures weighting the importance of extreme events in the scaled measure.

2.3. **Prediction of the PDFs using the conditional Gaussian framework.** From another approach to predict the final equilibrium PDFs with highly non-Gaussian structures, we can adopt the neural networks to learn the conditional Gaussian dynamics. In the previous section, we see that the final PDF of the barotropic topographic model can be computed from the conditional Gaussian framework

$$p([U(t),\hat{v}(t)]) = \frac{1}{L}\sum_{i=1}^{L} K_H\left(U(t) - U^i(t)\right) \sum_k p\left(\hat{v}_k \mid U^i(s \le t)\right). \tag{15}$$

In this way, the loss function for the training process can use the relative entropy directly from the above conditional Gaussian distributions, then

$$\mathcal{L}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{L} w_i \mathcal{P}\left(p^M\left(\mathbf{x} \mid U^i\right), p\left(\mathbf{y} \mid U^i\right)\right), \tag{16}$$

with $p\left(\mathbf{y} \mid U^i\right)$ the true conditional distribution, and $p^M\left(\mathbf{x} \mid U^i\right)$ from the trained neural network prediction

The density function for the large scale mean flow is achieved through a particle method for L trajectories U^i with a Gaussian kernel K_H

$$\frac{dU^i}{dt} = \sum_k h_k^* \hat{v}_k^i \; - d_U U^i + \sigma_U \dot{W}_0^i.$$

two-mode topo. h	β	H_1	H_2	dt	d_U	d_k	σ_U	d_T	κ
$H_1(\cos x+\sin x)+$ $H_2(\cos 2x+\sin 2x)$	2	1	0.5	1×10^{-2}	0.0125	0.0125	$\frac{1}{2\sqrt{2}}$	0.1	0.001

TABLE 1. Standard model parameters for the two-mode topographic model simulations.

And given each realization U^i for $s\le t$, the small scale modes are approximated by the conditional Gaussian structure $p\left(\hat{v}_k \mid U^i(\cdot)\right)\sim\mathcal{N}\left(\bar{v}_k,\sqrt{r_{v,k}}\right)$ with the conditional mean and variance solved by the equations

$$\begin{aligned}\frac{d\bar{\omega}_k^i}{dt}=&-ik\hat{h}_kU^i+\left[-d_k+ik\left(k^{-2}\beta-U^i\right)\right]\bar{\omega}_k^i-r_{k^i}\frac{i\hat{h}_k}{\sigma_U^2}\left[-i\sum_k k^{-1}h_k^*\omega_k^{i\prime}+\sigma_U\dot{W}_0^i\right],\\ \frac{dr_k^i}{dt}=&-\sigma_U^{-1}\left|\hat{h}_k\right|^2r_k^{i2}-2d_kr_k^i+\sigma_k^2.\end{aligned}$$

As we can see the above conditional Gaussian dynamics is usually the most expensive part to compute with strong nonlinearity during the conditional Gaussian framework solution. Here instead of solving directly the conditional mean and variance equations, we can again seek proper deep neural network model to predict the solutions in the conditional mean and variance such as

$$\begin{aligned}\bar{v}_k^M&=\mathrm{NN}_{\mathrm{mean}}\left(\left[U_{t-m},U_{t-m+1},\cdots,U_t\right],r_k^M\right),\\ r_k^M&=\mathrm{NN}_{\mathrm{var}}\left(\left[U_{t-m},U_{t-m+1},\cdots,U_t\right]\right).\end{aligned}\tag{17}$$

Additional information can be incorporated into the neural network structure depending on the statistical regime with either near-Gaussian or highly skewed statistics. With the optimized neural network (17) together with explicit dynamics for the mean flow U, it is interesting to check the skill of such neural network structure to learn and predict the various extreme events and skewed PDFs in the various statistical solutions.

3. Learning the Unresolved Dynamics from Partially Observed Data with Noise Forcing

Starting from the two-mode topographic barotropic model, we aim to get the trajectory and statistical prediction of the following coupled system

$$\frac{d\hat{\omega}_k}{dt}-i\left(k^{-1}\beta-kU\right)\hat{\omega}_k+ik\hat{h}_kU=-d_k\hat{\omega}_k+\sigma_k\dot{W}_k,\tag{18}$$

$$\frac{dU}{dt}-\sum_k k^{-1}\mathfrak{Im}\left(h_k^*\hat{\omega}_k\right)=-d_UU+\sigma_U\dot{W}_0,\tag{19}$$

together with the associated passive tracer equation

$$\frac{d\hat{T}_k}{dt}=\left(-\gamma_{T,k}+i\omega_{T,k}(t)\right)\hat{T}_k-\alpha\hat{v}_k,\quad \gamma_{T,k}=d_T+\kappa k^2,\ \omega_{T,k}=-kU(t).\tag{20}$$

Especially, we refer the large-scale zonal mean flow U as the 'observed state' that is measure at the time internal Δt (notice this is usually much longer than the integration time step $dt=0.01$ in the numerical scheme); and the small-scale vortical modes $\hat{\omega}_k$ with important feedbacks to the mean flow are treated as the unresolved states. The neural network is designed to predict all the unresolved small-scale vortical modes $\hat{\omega}_k$ without pre-knowledge of the original dynamical model (18). And together with the neural network output, the equation (19) can be used to update the solution of the large scale U with different levels of noises at the future state. The standard model parameters in the simulations to generate the true data is listed in Table 1.

The typical trajectory solutions with the model statistics of autocorrelation functions (ACFs) and probability density functions (PDFs) from the model setup listed in Table 1 are shown in Figure 3.1. The direct simulation solution provides the dataset for training in the neural network model. In this test regime, we observe strong intermittent dynamics and extreme events appearing in both the small-scale vortical modes $\hat{\omega}_1,\hat{\omega}_2$ as well as in the two passive tracer modes $\hat{T}_1,\hat{T}_2$. Fat-tails are created for both the large and small scale states. And multi time scales are observed between the large-scale mean flow U and the small-scale vortical modes $\hat{\omega}_1,\hat{\omega}_2$. This provides a desirable testbed containing the sequence of interesting features to be captured with proper neural network architecture.

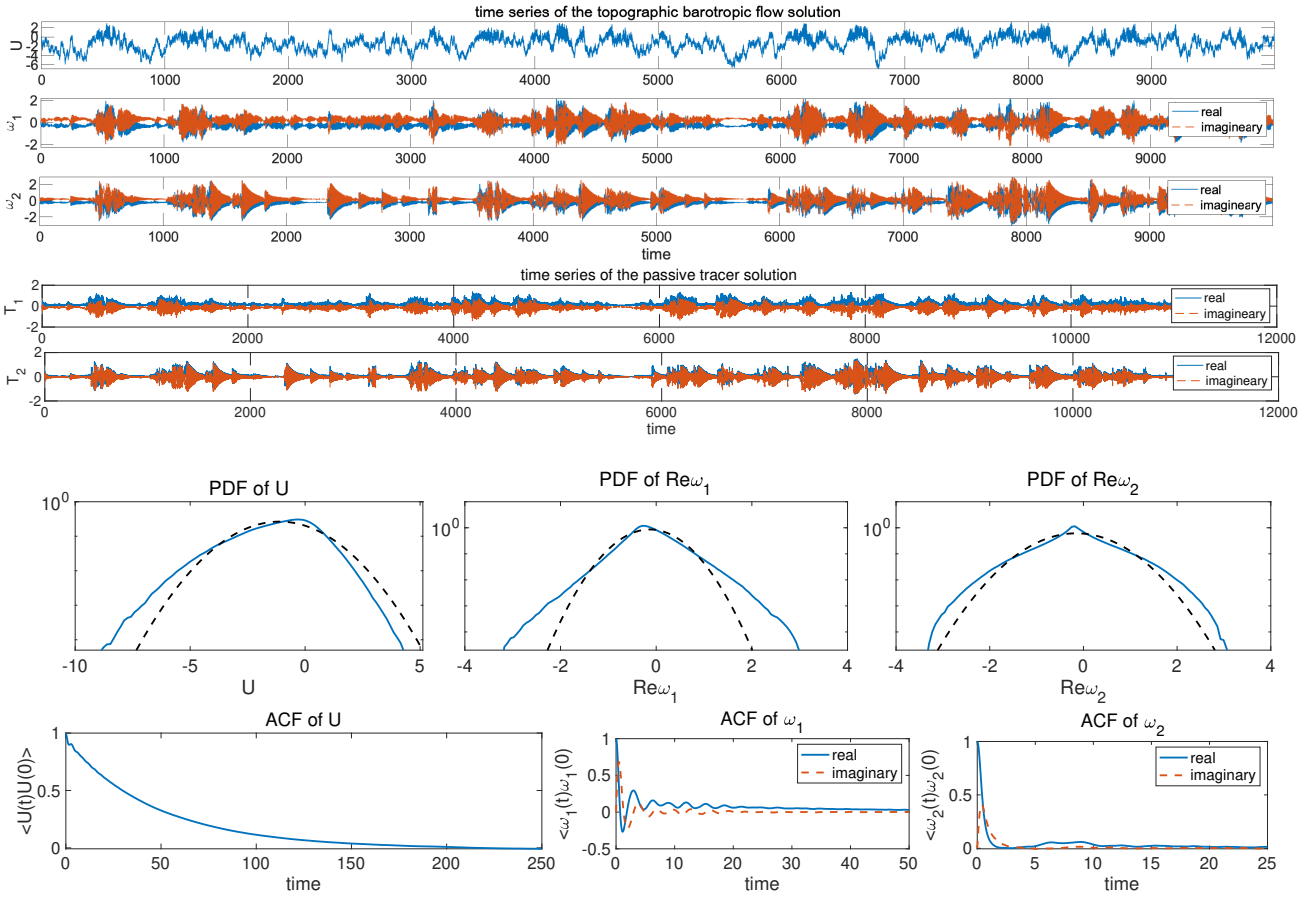


FIGURE 3.1. Time-series, probability density functions and autocorrelation functions of the large-scale state U and the two small-scale modes $\hat{\omega}_1, \hat{\omega}_2$ with large extreme events. The solutions will be used to produce the training data for the neural network

Remark. The white noise forcing, $\sigma_k \dot{W}_k$ and $\sigma_U \dot{W}_U$, in (18) and (19) have independent time increments, thus these *white noise uncertainty* are impossible to be learned by the neural network from data for trajectory predictions with new white noise forcing realization. Thus they are better to be treated as additional noises in the model. This becomes increasingly challenging for the neural network model to learn from data with larger noise level from $\sigma_U \dot{W}$, which adds overwhelming uncertainty into the model states.

To evaluate the prediction accuracy in the neural network models, we can also adopt two approaches by examining both the deterministic trajectory solution and the statistical solution:

- *Trajectory prediction*: the neural network is used for the pointwise prediction for one trajectory solution from particular initial state with uncertainties from input value and white noise forcing. We cannot expect accurate prediction of the trajectory very far beyond the decorrelation time due to the turbulent natural of the system especially with strong white noise forcing.
- *Statistical prediction*: the neural network is used to recover from data the statistical features such as the autocorrelation functions and fat-tailed probability density functions. Instead of focusing on the pointwise solution, it is often more useful with practical importance to learn the representative statistical structures directly from (partial) data.

3.1. **Training model with different total number of epochs.** First of all, we check the different number of epochs to improve the trained model performance. During the training process, we generate 10000 samples from a long time series, and train each batch with batch size 100 for 100 iterations. Then the model is trained for 50, 100, and 200 epochs. The learning rate starts with lr = 0.005, and is reduced to 0.5 of the previous value twice. In

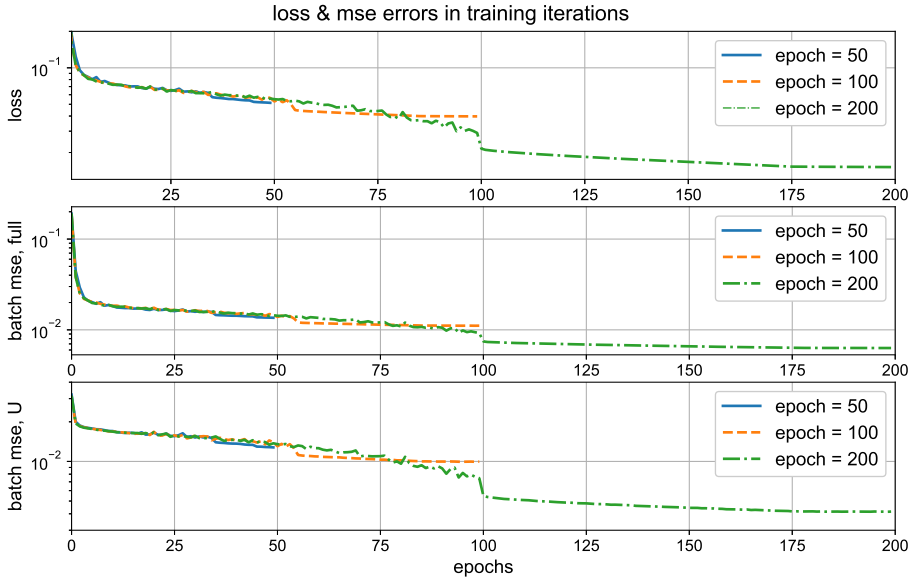


FIGURE 3.2. Loss function and batch averaged relative mean square error (BMSE) during the training iterations with different total number of epochs. The BMSEs are compared among the full predicted states, $(U, \hat{\omega}_1, \hat{\omega}_1)$; and the BMSEs in the large-scale mean state U. Note the logarithmic scale along y coordinate.

this first test case, we pick the model hyperparameters as: input sequence length $m = 100$, observation time step $\Delta t = 0.5$, the size of hidden layer $h = 50$.

The training performance with different number of epochs 50, 100, and 200 are compared in Figure 3.2. It shows that the loss function is effectively minimized during the training process as well as the RMSEs. The loss function and the BMSEs drop sharply during the first few iterations. The error keeps decreasing for a long time with more training iterations. More refined results can be achieved with the largest number of epochs 200. Still, the improvement from even larger number of epochs is minimal and will increase the danger of overfitting. As seen from Figure 3.3, the model trained with 100 epochs gives the best prediction results with larger number of time iterations. In the following tests, we will training the neural networks with 100 epochs as the standard setup.

Figure 3.3 shows the prediction results using the three trained models with different training epochs. The errors are normalized by the variance in the equilibrium state. Here we compute the batch prediction with $N = 200$ samples. The trained neural network model is used recursively to update the solution for longer time prediction (that is, up to $T = 100$ with $n = 200$ iterations). The error accumulates during the model iterations, and saturates with large iterations. Still we see the prediction results stay accurate for a long time far beyond the decorrelation time.

For a qualitative illustration of the predicted solution trajectory structures, Figure 3.4 plots one trajectory recovery from the trained neural network model compared with the true solution from solving the dynamical system. It can be seen that the intermittent structures can be captured from the neural network prediction. This implies that the neural network has the skill to learn the essential dynamical structures directly from data. The large scale mean state U gives the best most accurate prediction, while the first small-scale mode $\hat{\omega}_1$ is less accurate but with good agreement. Still, it shows that the second mode $\hat{\omega}_2$ with a shorter mixing time is more difficult to capture with accuracy in the neural network.

3.2. **Detail comparison of model performance with different hyper parameters.** Next in this subsection, we carry out a detailed extensive analysis for the effects from various neural network structures. The major hyperparameters to test here include: i) the size of the hidden state size h; ii) the number of time lags s used to update the next stage state; iii) the available data time frequency Δt between two adjacent measurements; and iv) the LSTM chain length m for the input data length.

In general from intuition, a higher dimensional phase space for the hidden state provides a larger group of candidate configurations to generate the final prediction states. Thus it will provide more accurate training and

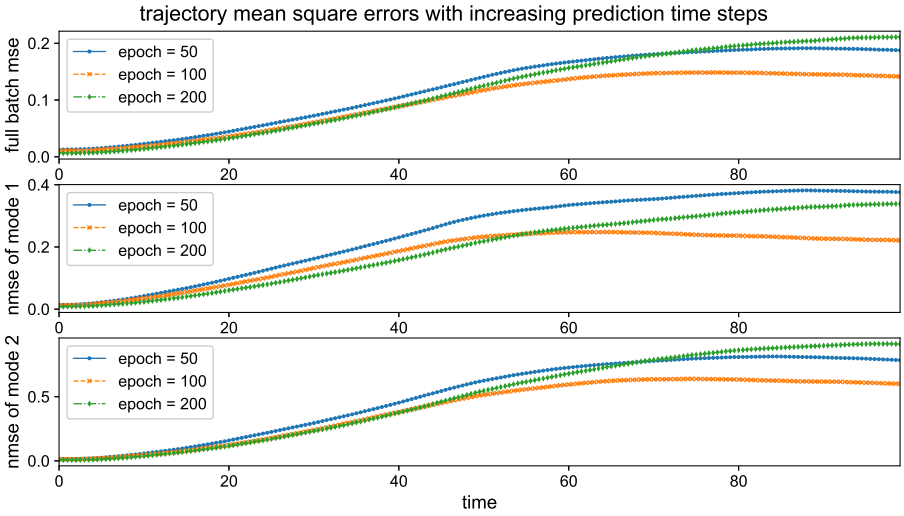


FIGURE 3.3. Prediction mean square errors by recursively using the optimized neural network model trained with different numbers of training epochs. The first row shows the full batch averaged MSE with $N = 200$ samples, and the detailed normalized MSEs with the variance are compared in the second and third rows.

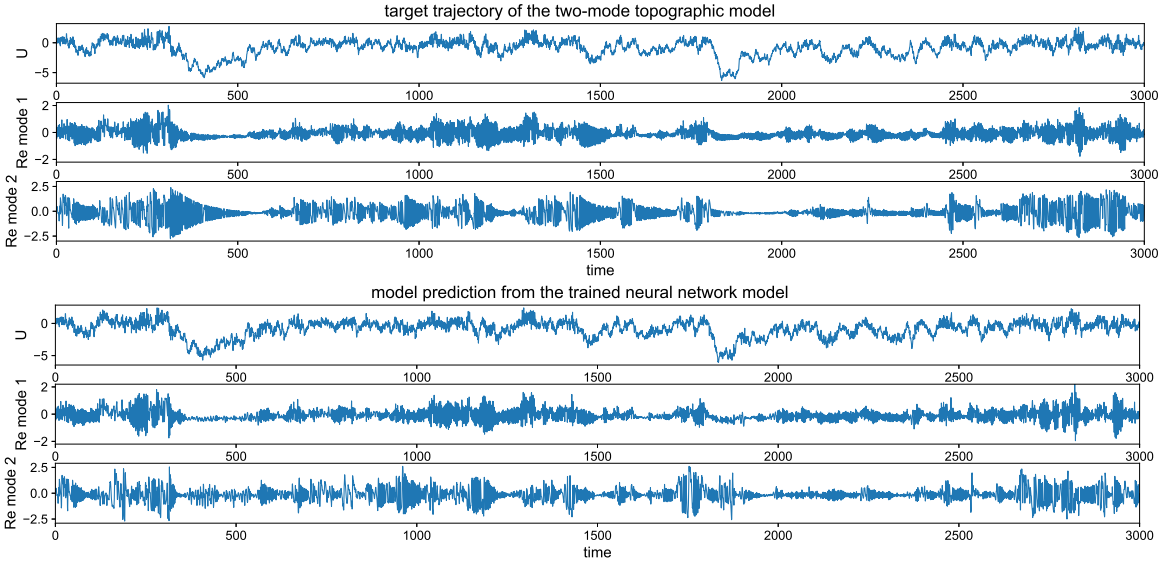


FIGURE 3.4. Trajectory prediction using the trained model with 100 epochs. The true solution from directly solving the dynamical equation is compared in the upper row.

prediction results. At the same time, it will become more expensive to train the neural network. The time lag size s represents the number of the previous states along the LSTM chain used to update the next state, that is, $y_n = f(x_{n-1}, \cdots, x_{n-s})$. In the standard LSTM net, only the current model output is used, that is, $s = 1$. Here as a further improvement designed for the dynamical system prediction, the previous history information is also incorporated to include the history of the system more explicitly. The final training loss function values and batch MSEs are listed in Table 2.

3.2.1. *Effects from different hidden state sizes and time lag sizes.* Figure 3.5 illustrates the training processes with different parameters in the LSTM connections. Different hidden state sizes $h = 200, 100, 50, 10$ are compared with lag size $s = 5$; and different lag sizes $s = 1, 2, 5, 10$ are compared with hidden state size $h = 100$. First, we see that

h	200	100	50	10	s	1	2	5	10
loss	0.0020	0.0045	0.0138	0.0708	loss	0.0233	0.0069	0.0045	0.0033
MSE	0.0029	0.0031	0.0051	0.0173	MSE	0.0062	0.0028	0.0031	0.0042
Δt	0.1	0.2	0.5	0.8	m	200	100	50	10
loss	0.0122	0.0120	0.0045	0.0025	loss	0.0041	0.0045	0.0049	0.0161
MSE	0.0028	0.0035	0.0031	0.0045	MSE	0.0020	0.0031	0.0055	0.0307

TABLE 2. Final training loss value and BMSE with different model hyperparameters. The standard model parameter for comparison are set as hidden state size $h = 100$, time lag for next update $s = 5$, data measurement time $\Delta t = 0.5$, and input data length $m = 100$. The mixed loss function is always used in the optimization.

a larger number of hidden state size h indeed improves the final optimized state accuracy. The smallest case $h = 10$ is not large enough to capture all the required state configurations to provide an accurate prediction. On the other hand, with around $h = 100$, the training accuracy is already high enough, while a large hidden state size $h = 200$ further improves the accuracy, while just gives a small improvement but large computational cost.

In the tests with different time lags for update, as in the previous illustration, we observe that adding more history state information indeed improve the training results with higher accuracy, compared with the standard case $s = 1$ using only the current network output. This confirms that it is meaningful to introduce the model modification with inclusion of longer previous information for the next update. Still we also observe that a too large lag size (such as $s = 10$) is also not desirable, and in fact produced larger errors in the final state. At the same time it is also observed that the larger lag sizes $s = 10$ gives faster decay of the error in the first 100 iterations. This implies that the model including more time lags for updating is easier to train, but might suffer a less overall accuracy. This can be useful when there is no enough data for training or too expensive to train for too many epochs.

Then we can compare the prediction skill with increasing time iterations using the optimized models trained from different parameters. Figure compares the NMSEs using different hidden state sizes h and different time lags s in the recurrent updates. Consistent with the training results, larger hidden state sizes h gives better overall prediction accuracy. In long time prediction, the small state size $h = 10$ saturated at much larger error amplitude, while the large state sizes $h = 100, 200$ gives similar final errors. Still, the larger state size case $h = 200$ give slower growth in the error from the starting time, meaning more robust to the accumulated prediction errors from the previous steps. Similar observation can be found depending on the size of time lags s of previous information for the update of the next state. A longer lag size help to improve the prediction accuracy. Also, it is observed that the first state $\hat{\omega}_1$ is easier to prediction than the second mode $\hat{\omega}_2$ with a faster mixing rate. The larger time lag size s can effectively improve the prediction accuracy in the fast variable $\hat{\omega}_2$.

3.2.2. *Effects from data measurement time and LSTM chain length.* Next, we study the effects from the data time frequency Δt between two observed states and the length of the LSTM chain l for the input variables in training. Figure 3.7 shows the training results. First, different time steps $\Delta t = 0.05, 0.1, 0.2, 0.5, 0.8$ and different input data length $l = 10, 50, 100, 200$ are compared. A sparser data measurement time Δt leads to larger final training errors, and requires longer iterations to train. Second, it shows that sufficient long LSTM chain length is required to reach accurate model training results. Notice that the initial value of the hidden state is always set to be zero in the first LSTM cell. Thus, a long chain is necessary to make sure the hidden state reaches a proper saturated state.

The corresponding prediction results with the different structures are compared in Figure 3.8. First, with a smaller (and more frequent) data observation time Δt, the error starts in smaller values in the first few iterations. But the error gets accumulated fast since larger number of iterations is required to reach the final prediction time T. As a result, the shorter observation time Δt actually leads to larger errors in the final prediction with more error emerged during the iterations. Next, we see a relatively long input data size l is necessary to lead to accurate prediction result. The longer LSTM chain contains more previous information to update the hidden state as well as the final prediction. It leads to the smallest final prediction error and the slowest growth during the process with more robustness to the previous output model errors.

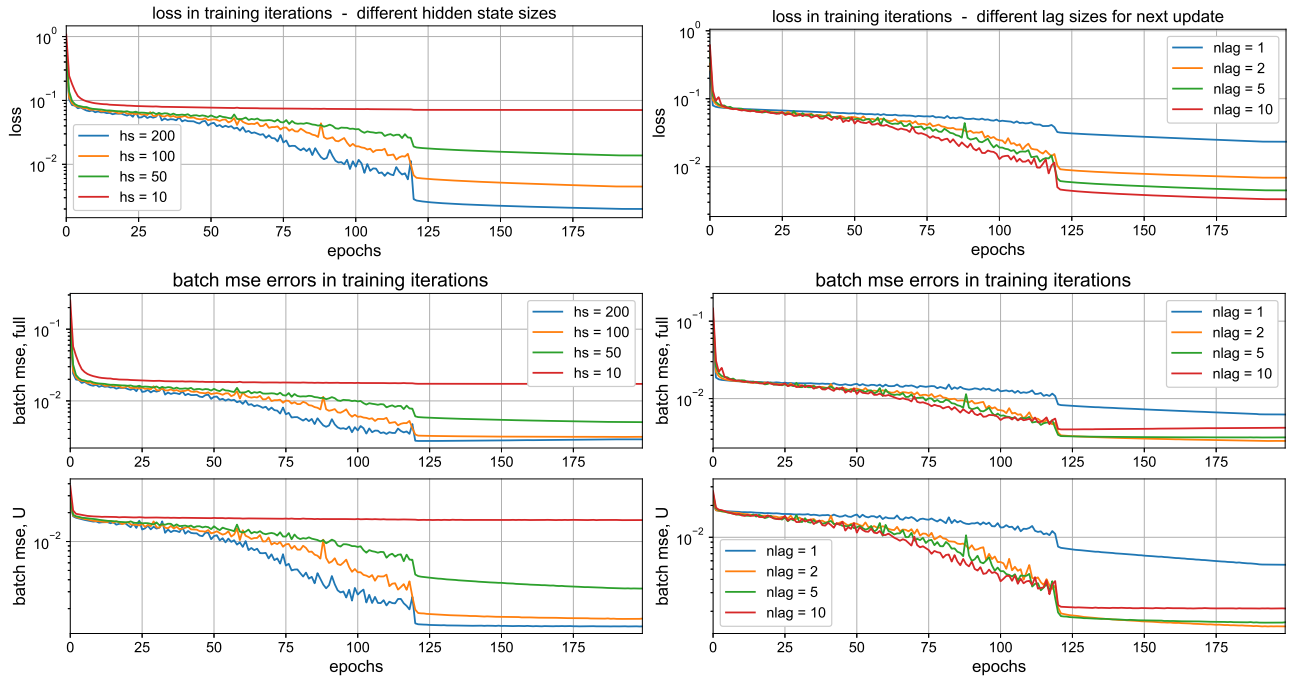
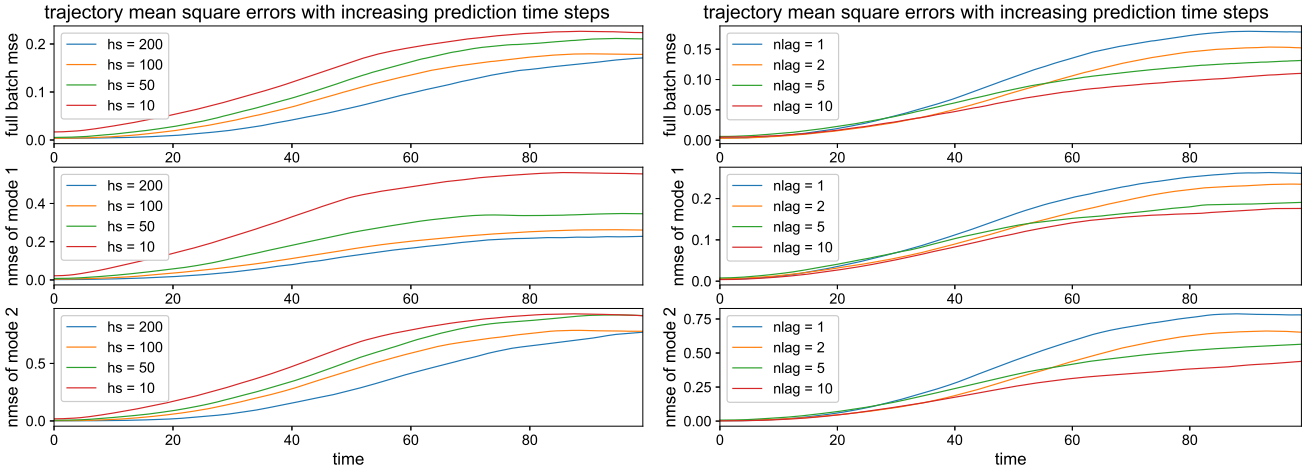


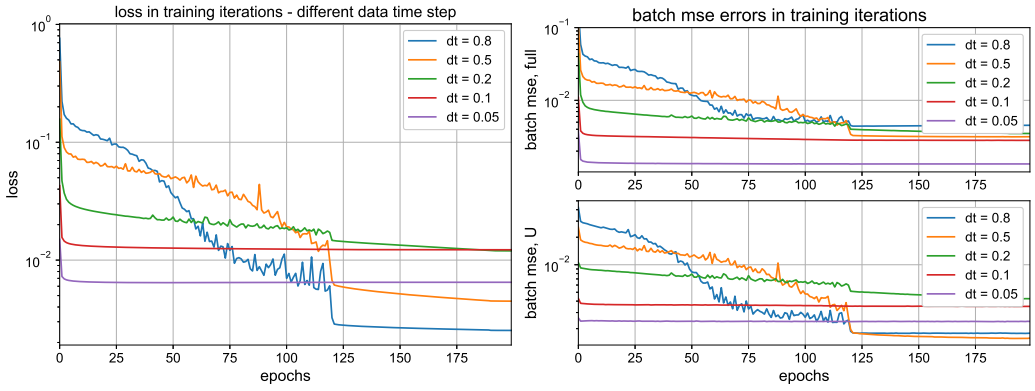
FIGURE 3.5. Training with different model hyperparameters, the hidden state size h (left), and the number of lags s for next update (right). The first row plots the time evolution of the loss function (using the mixed KLD + L_2 loss), and the next two rows compare the batch averaged MSEs in the full prediction state (middle row) and in the large-scale mean flow U (bottom row).



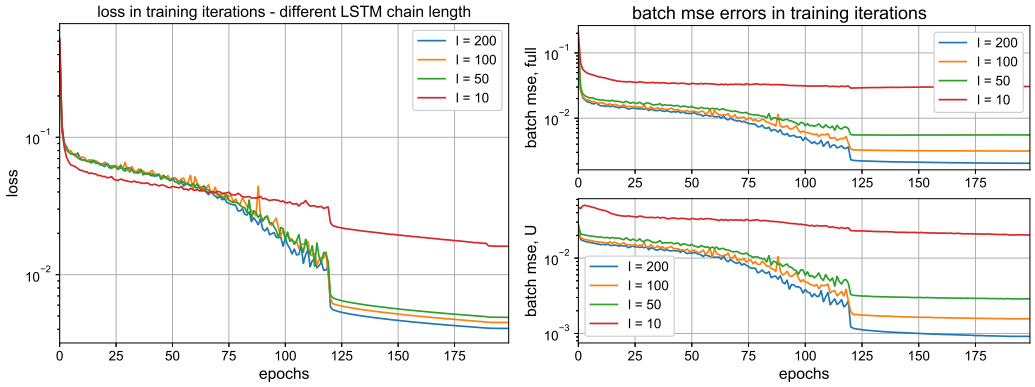
(A) different hidden state size h

(B) different lag size s for updating

FIGURE 3.6. Prediction mean square errors by recursively using the optimized neural network model trained with different hidden state sizes h (left) and number of lags s (right).

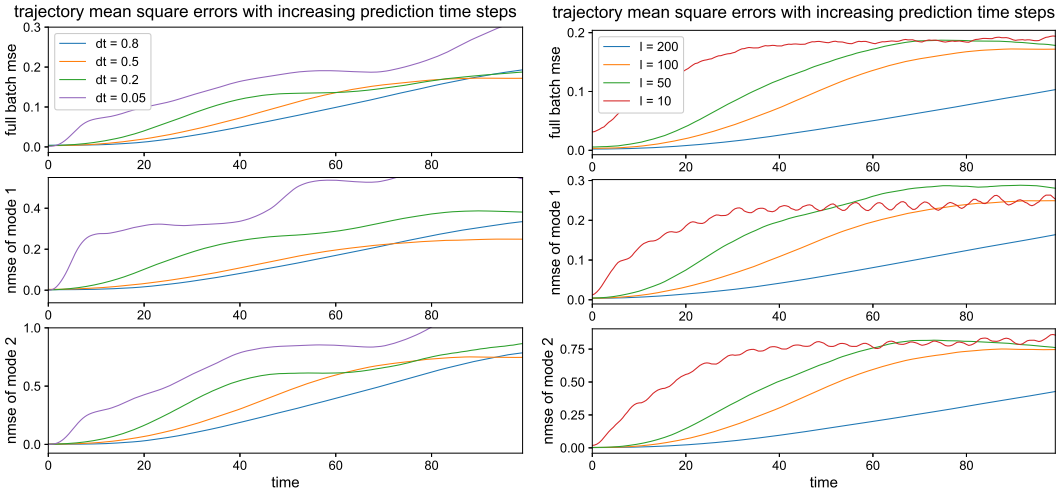


(A) training with different data observation step size



(B) training with different LSTM chain size

FIGURE 3.7. Training the neural network model with different data observation time Δt (upper) and LSTM chain length l (lower). The left panel plots the time evolution of the loss function, and the right panel compares the batch averaged MSEs in the full prediction state (middle row) and in the large-scale mean flow U (bottom row).



(A) different data observation step size

(B) different LSTM chain size

FIGURE 3.8. Prediction mean square errors by recursively using the optimized neural network model trained with different data time step (left) and LSTM chain size (right).