Lyapunov-Guided Embedding for Hyperparameter Selection in Recurrent Neural Networks: Supplementary Materials

Ryan Vogt, Yang Zheng and Eli Shlizerman

A Task Implementation and LE Computation Details

A.1 Target Learning with Random RNN

The random four-sine is a combination of four sine waves whose frequencies and amplitudes are integers randomly generated from the range $[1, 6]$ and $[1, 11]$, respectively. We train each network for 15 epochs where each epoch contains a 120s signal with time interval 0.1s, i.e., the signal length is 1200. After training the last epoch, the final recorded validation loss is a mean absolute error (MAE) between the reconstructed and target signal.

In terms of hyperparameter selection, we vary the $\boldsymbol{g}$ value in the network which determines the degree of chaos of the network as larger $\boldsymbol{g}$ means a more chaotic network. All $\boldsymbol{g}$s are selected from range $[1.1, 2.0]$ with an interval of 0.1 and we train 120 networks for each $\boldsymbol{g}$, i.e., 1200 networks in total.

We leave other hyperparameters, such as network size and learning rate constant over all experiments. We use 10 sequences of 200 time steps to calculate the LEs during testing, i.e., 10 random starting points and their following 200 time steps are used for LEs calculation. The final LEs is the average of those 10 results.

A.2 Character Prediction: CharRNN

The character sequences are randomly split into train and validation sets with an 80%/20% split. In order to ensure the independence of the trained networks, each trained network then takes a random sample of 30% of the sequences in the full training set and validation set to form its own training and validation sets.

The network is then trained for 15 epochs on the random training data and the reported validation loss is calculated from the random validation data

after all training. The selection of most hyperparameters for the LSTM, including the dropout, and learning rate, and optimizer, is held constant for each instance. However, the distribution used to initialize the weight parameters of each network instance is a uniform distribution on the interval $[-p, p]$, with p sampled uniformly from the interval $[0.04, 0.40]$. Furthermore, the size of the hidden layer of the LSTM is selected from [64, 128, 256, 512], with 300 networks of each size trained on this task.

We use 10 sequences of length 100 to calculate the LEs. To treat the spectrum for each network size with the same model, we use linear interpolation to increase the dimension of each spectrum to 1024. This gives a total of 1200 spectra of LSTMs on which to train a model.

A.3 Sequential MNIST

Our model is trained with the train dataset for 20 epochs and after all training, the cross entropy loss on test dataset is recorded as the final validation loss. The network size are chosen from $[32, 64, 128, 256]$ with 150 networks of size $32, 128$ and 256, and 350 networks of size 64, i.e., 800 networks, trained on this task. The initial weights of the model is uniformly initialized between $-p$ and p where p is randomly generated from $[2.0, 3.0]$. Other hyperparameters, such as learning rate, number of layers are the same for all experiment. Lyapunov exponents are calculated on one mini-batch of the test dataset which includes 128 different input sequences. The final LEs is the average over those 128 sequences. As we did for the CharRNN task, we make all networks have the same dimension of LEs using linear interpolation.

Table S1 Network training details for RNN tasks

Task	**Network Type**	**Hidden Size(s)**	**Init Param Range**	**Total Networks**	Loss **Type**
Target Learning	Random RNN	512	$1.0 - 2.0$	1200	MAE
CharRNN	LSTM	$64, 128, 256, 512$	$0.04 - 0.40$	1200	Cross Entropy
SMNIST	LSTM	$32, 64, 128, 256$	$2.0 - 3.0$	800	Cross Entropy