Deep Learning Prediction and Uncertainty Quantification of High-Dimensional Time-Series Data

By: Justin Lo Tian Wen
Supervisor: A/P Alexandre Hoang THIERY

01

INTRODUCTION

Motivation & Objectives

02

PREDICTION METHODS

RNN, LSTM, RC
Koopman Autoencoders

03

UNCERTAINTY QUANTIFICATION

Deep Ensembles,
Mean Variance Estimation

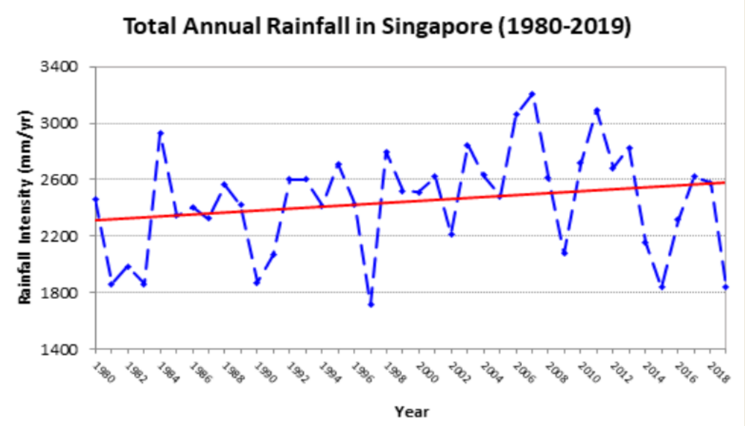
04

DISCUSSION

Conclusion & Future Work

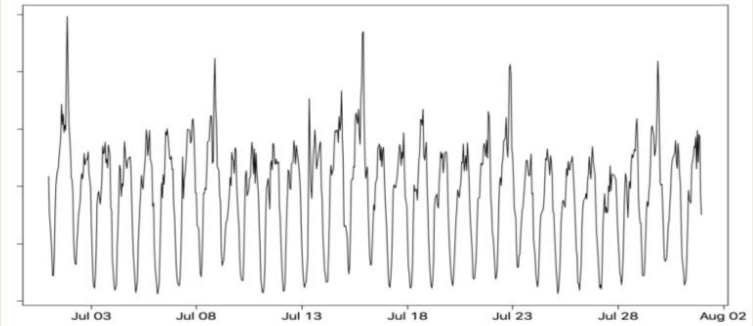
01

Introduction



http://www.weather.gov.sg/climate-past-climate-trends/

The hourly sum of Uber trips in month (2017)



https://eng.uber.com/forecasting-introduction/

COVID-19 cases in USA

TIME-SERIES PREDICTION

TRADITIONAL METHODS

Autoregressive integrated moving average (ARIMA)

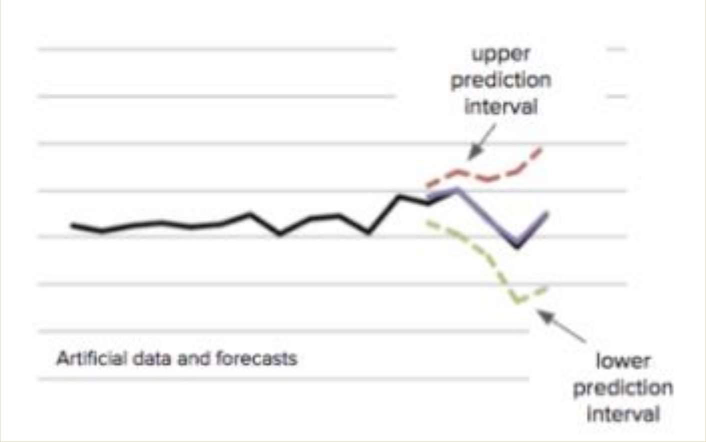
Exponential smoothing methods

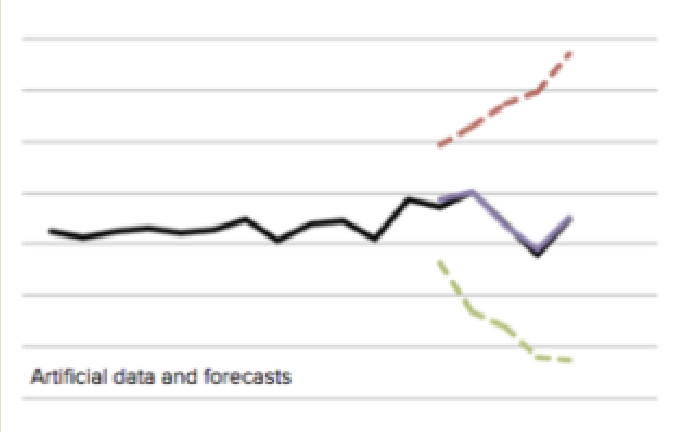
MACHINE LEARNING

Support Vector Regression

Deep Learning

UNCERTAINTY ESTIMATION





https://eng.uber.com/forecasting-introduction/

OBJECTIVES

IMPLEMENT

Implement using JAX to understand the architecture and role of hyperparameters

EVALUATE

Using a common benchmark, compare the effectiveness of prediction methods

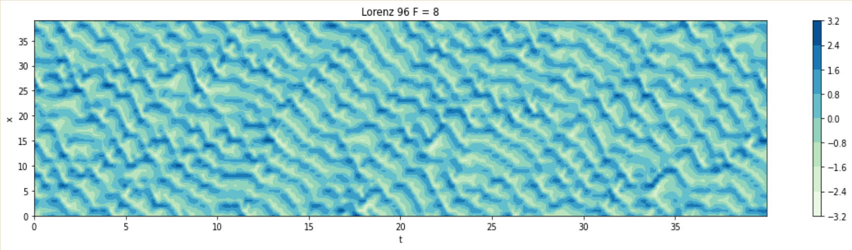
UNCERTAINTY ESTIMATION

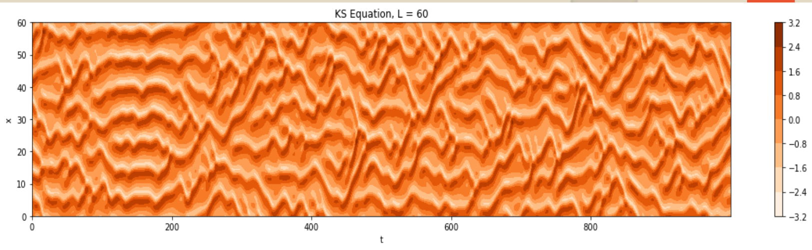
Provide and evaluate effectiveness of uncertainty bounds for predictions

DYNAMICAL SYSTEMS

- Used in applications within time-series forecasting
- Observational data that are described by complex systems
- Evolution of a finite dimensional state, x, across time, t

DYNAMICAL SYSTEMS





LORENZ-96

KS Equation

Continuous-time process: $\frac{dx}{dt} = f(x, t)$

High-dimensional systems

Exhibit chaotic behaviour

LORENZ-96

- Commonly used to study predictability of weather
- System of K coupled ODEs

$$\frac{dX_k}{dt} = -X_{k-2}X_{k-1} + X_{k-1}X_{k+1} - X_k + F$$

$$X_{k-K} = X_{k+K} = X_k$$

KS EQUATION

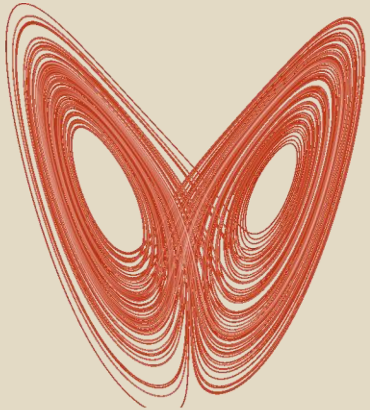
- Kuramoto-Sivashisky Equation
- Describe reaction-diffusion systems
- Describe instability in flame fronts
- Partial differential equation with spatial domain of L

$$u_t + u_{xx} + u_{xxxx} + uu_x = 0$$

$$u(x + L, t) = u(x, t)$$

CHAOTIC BEHAVIOUR

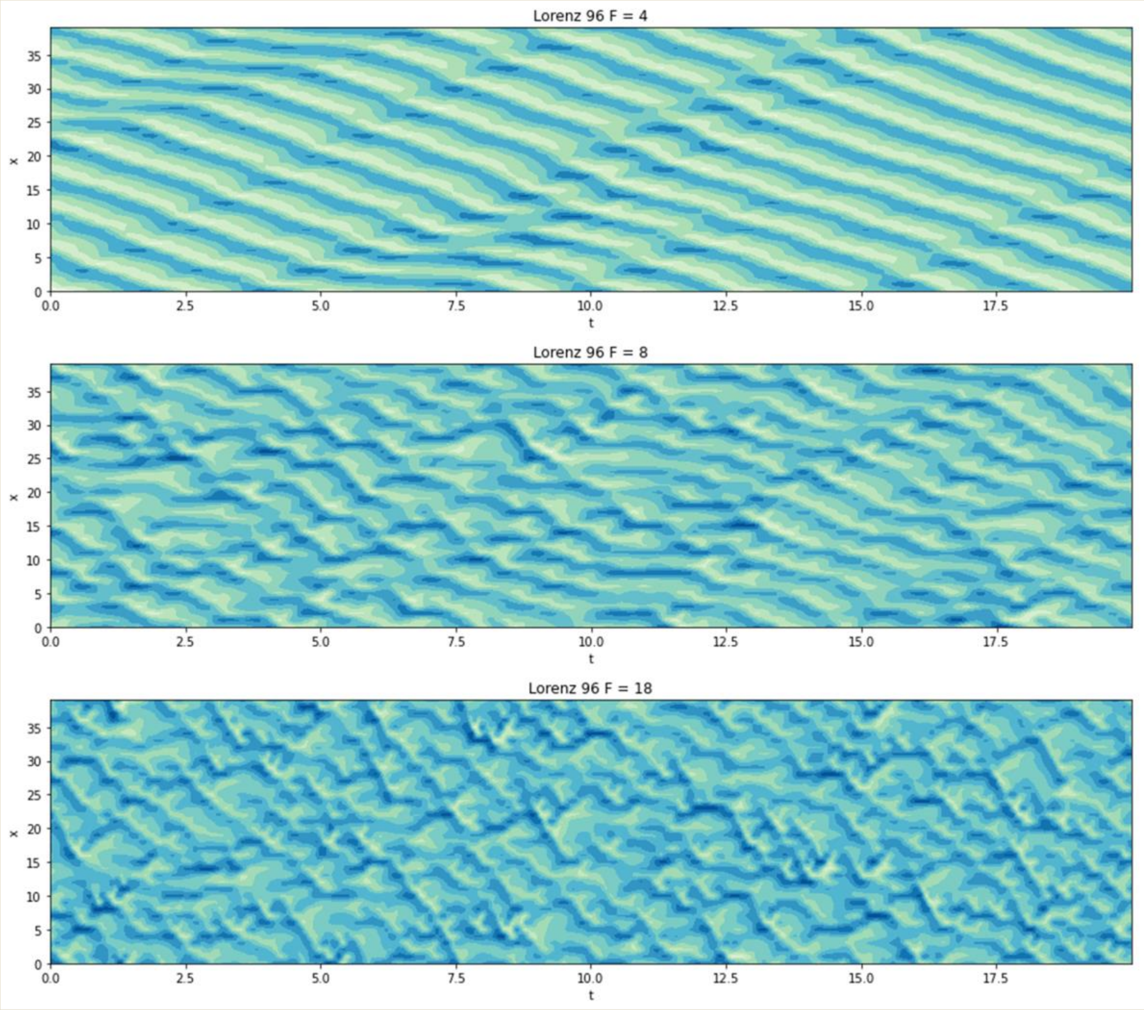
- “Butterfly effect”
- Small change in initial conditions → Exponential differences in neighbouring trajectories



https://en.wikipedia.org/wiki/Chaos_theory

CHAOS (LORENZ-96)

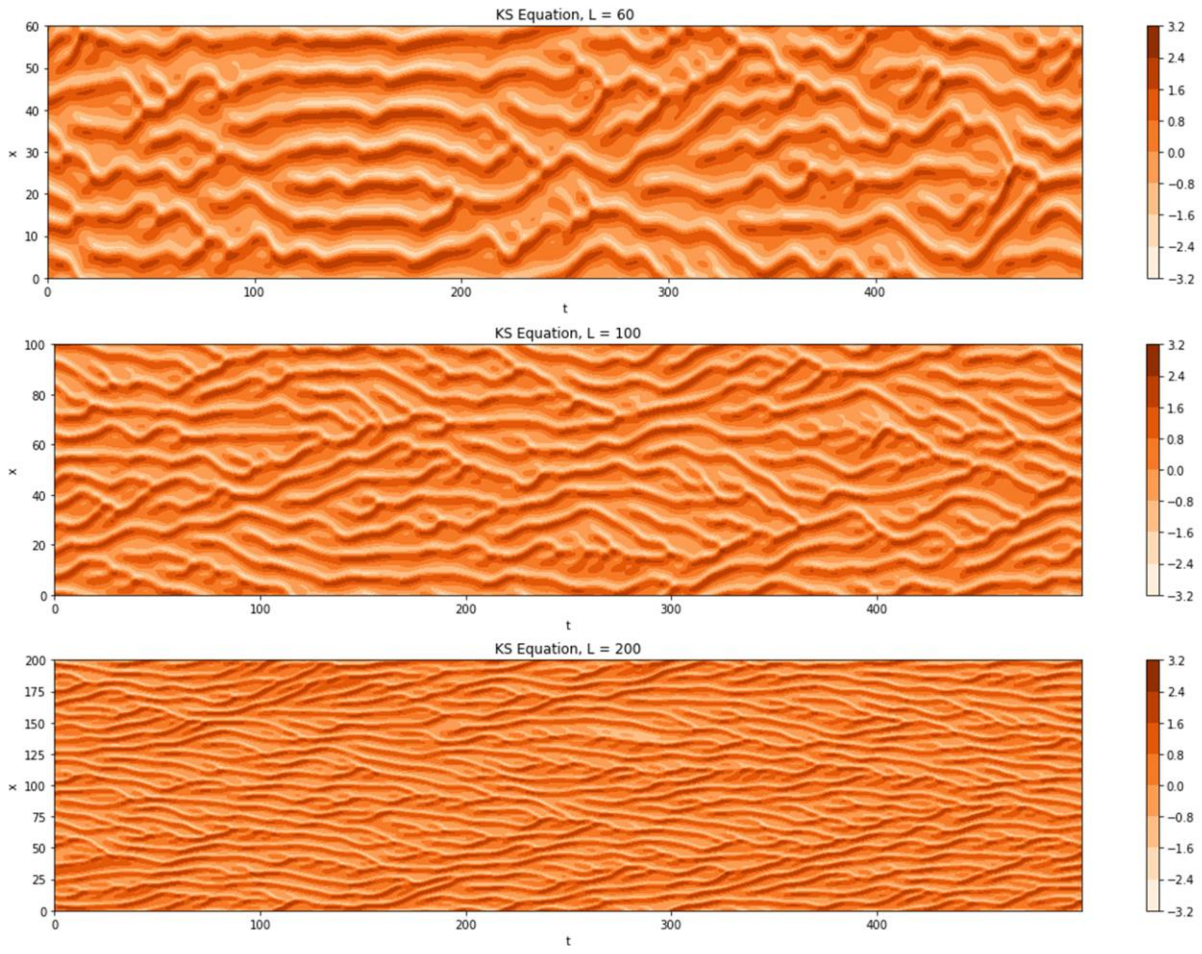
$$\frac{dX_k}{dt} = -X_{k-2}X_{k-1} + X_{k-1}X_{k+1} - X_k + \boldsymbol{F}$$


Lorenz 96 F = 4
Lorenz 96 F = 8
Lorenz 96 F = 18
x
t
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
20
25
30
35

CHAOS (KS EQUATION)

$$u_t + u_{xx} + u_{xxxx} + uu_x = 0$$

$$u(x + L, t) = u(x, t)$$


KS Equation, L = 60
KS Equation, L = 100
KS Equation, L = 200
x
t
3.2
2.4
1.6
0.8
0.0
−0.8
−1.6
−2.4
−3.2

LYAPUNOV TIME

- Quantify level of chaos to allow equal comparisons
- **Lyapunov exponent** Λ measures rate at which neighbouring trajectories diverge
- **Lyapunov time** Λ_{max}^{-1} measures time for a trajectory to diverge by a factor of e

DATA GENERATION

- N = 200 000, 10% discarded
- Remaining data split into 50-50 train-test set
- Lorenz-96
  - F = 8, d = 40
- KS Equation
  - L = 60, spatially discretised into d = 240 points

02

Prediction Methods

EVALUATING PREDICTIONS

- How accurate is a prediction in the regression?
- $NRMSE = \frac{1}{N}\sqrt{\frac{(\hat{y}-y)^2}{\sigma_y^2}}$
- $PH_k = \underset{t}{\operatorname{argmax}}\,(NRMSE(t) < k)$

01

RNN

02

LSTM

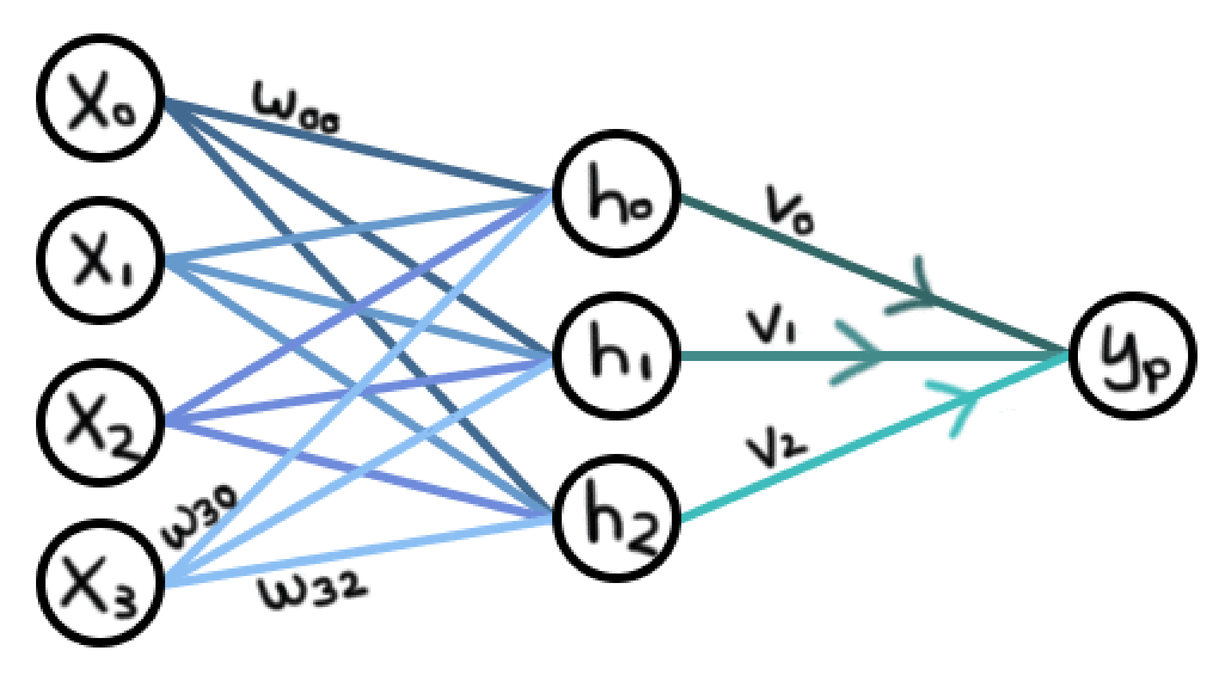
03

RESERVOIR COMPUTING

04

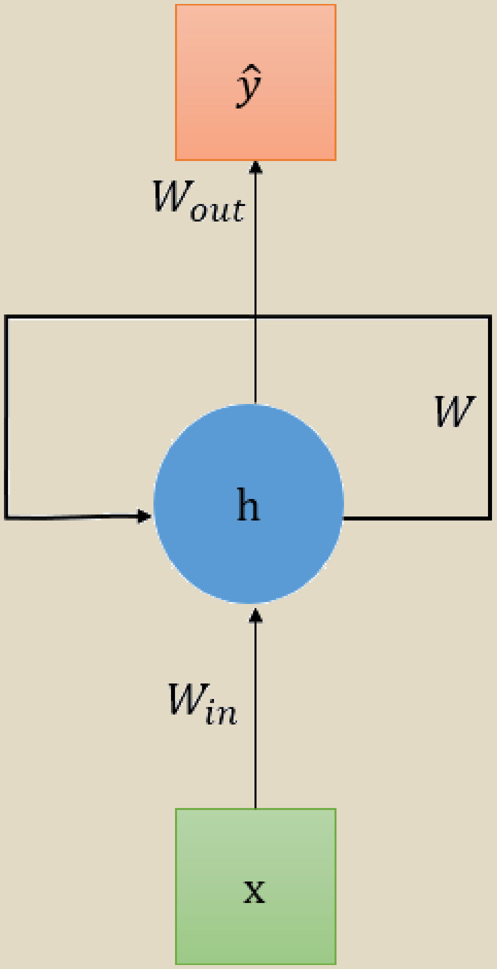
KOOPMAN AUTOENCODERS

VANILLA NN?



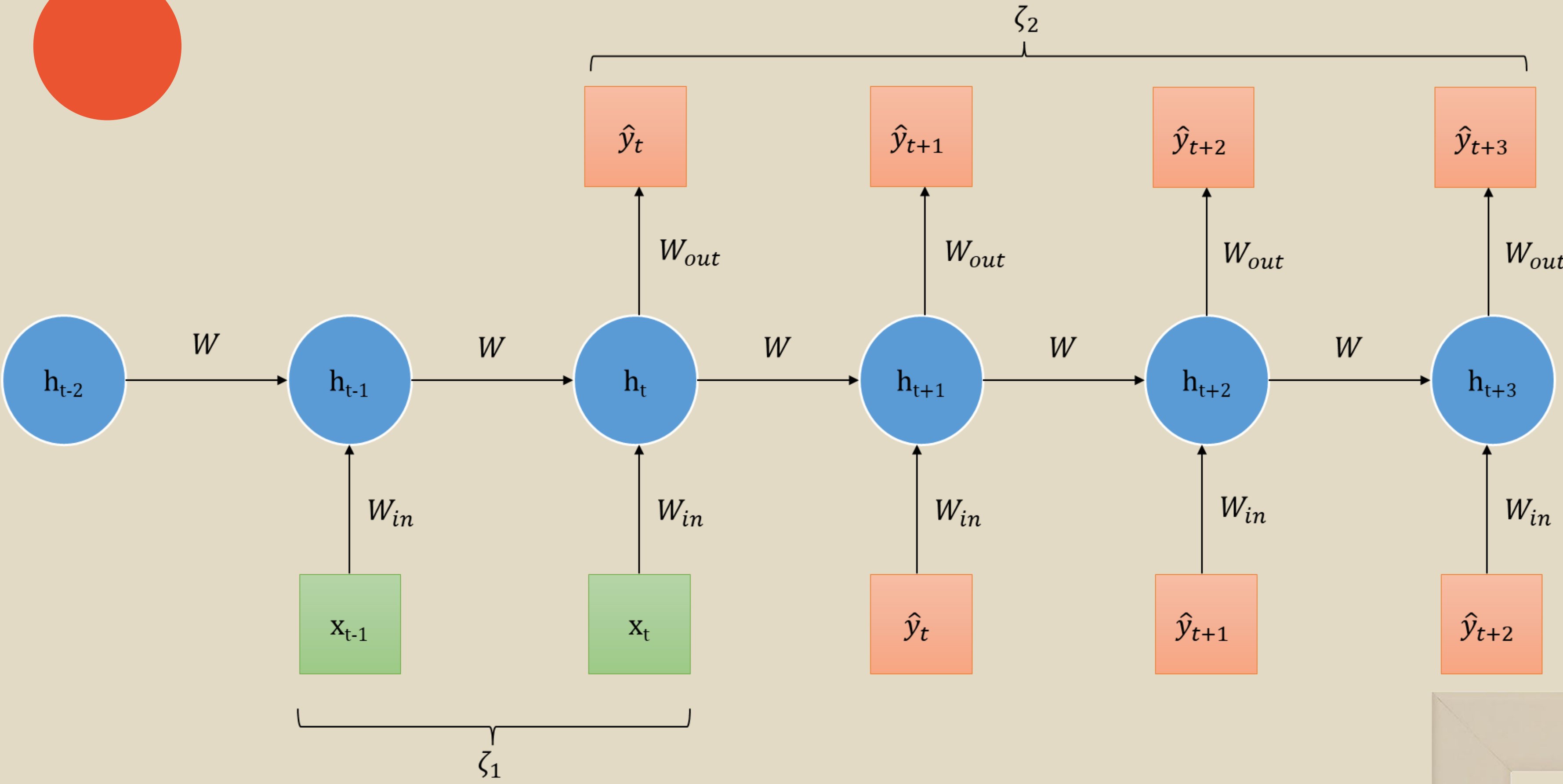
https://blog.insightdatascience.com/a-quick-introduction-to-vanilla-neural-networks-b0998c6216a1

RNN

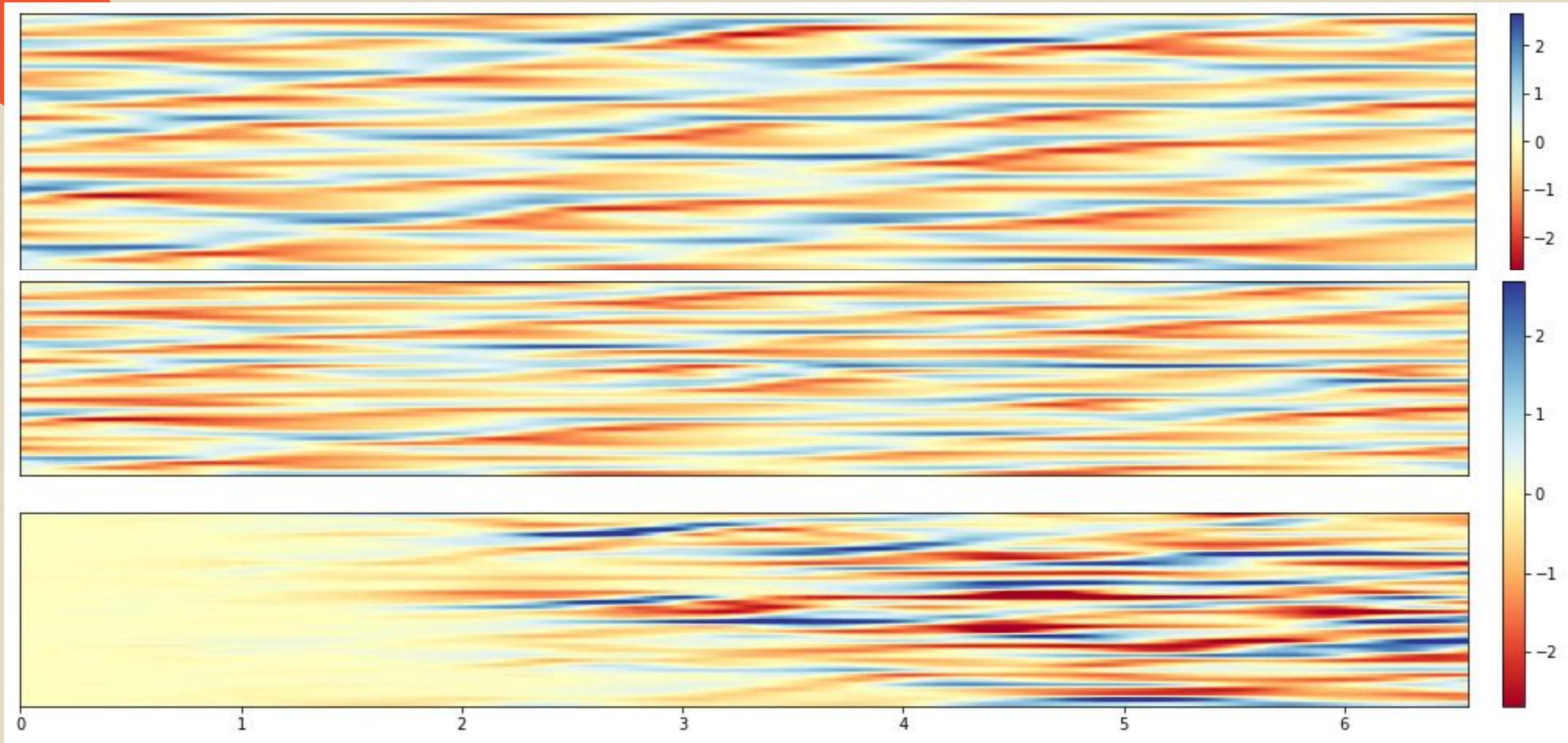


$$\hat{y}_t = W_{out} h_t + b_{out}$$

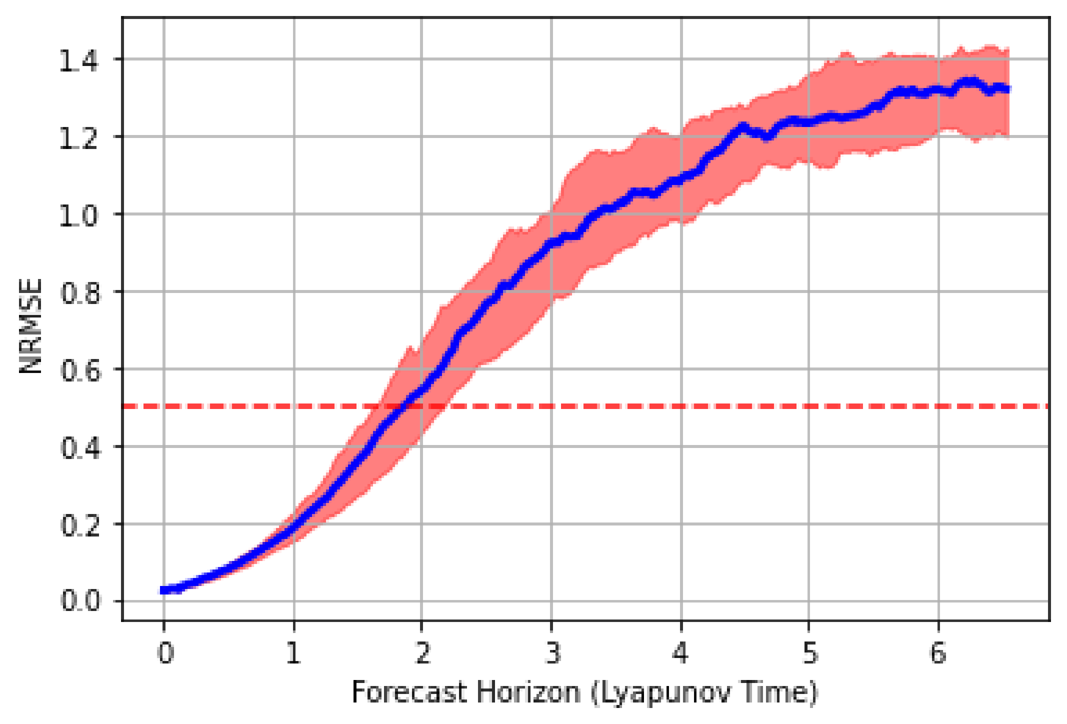
$$h_t = \sigma(W h_{t-1} + W_{in} x(t) + b_h)$$



ζ_2
$\hat{y}_t$
$\hat{y}_{t+1}$
$\hat{y}_{t+2}$
$\hat{y}_{t+3}$
W_{out}
W_{out}
W_{out}
W_{out}
h_{t-2}
W
h_{t-1}
W
h_t
W
h_{t+1}
W
h_{t+2}
W
h_{t+3}
W_{in}
W_{in}
W_{in}
W_{in}
W_{in}
x_{t-1}
x_t
$\hat{y}_t$
$\hat{y}_{t+1}$
$\hat{y}_{t+2}$
ζ_1



(Top) Actual **(Middle)** Predicted **(Bottom)** Error



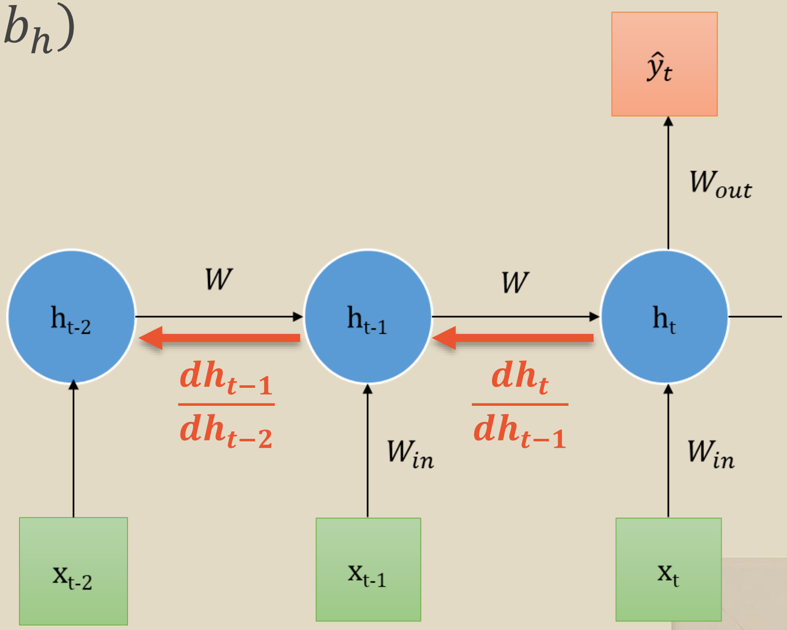
NN size = 500

$\zeta_1 = 8$

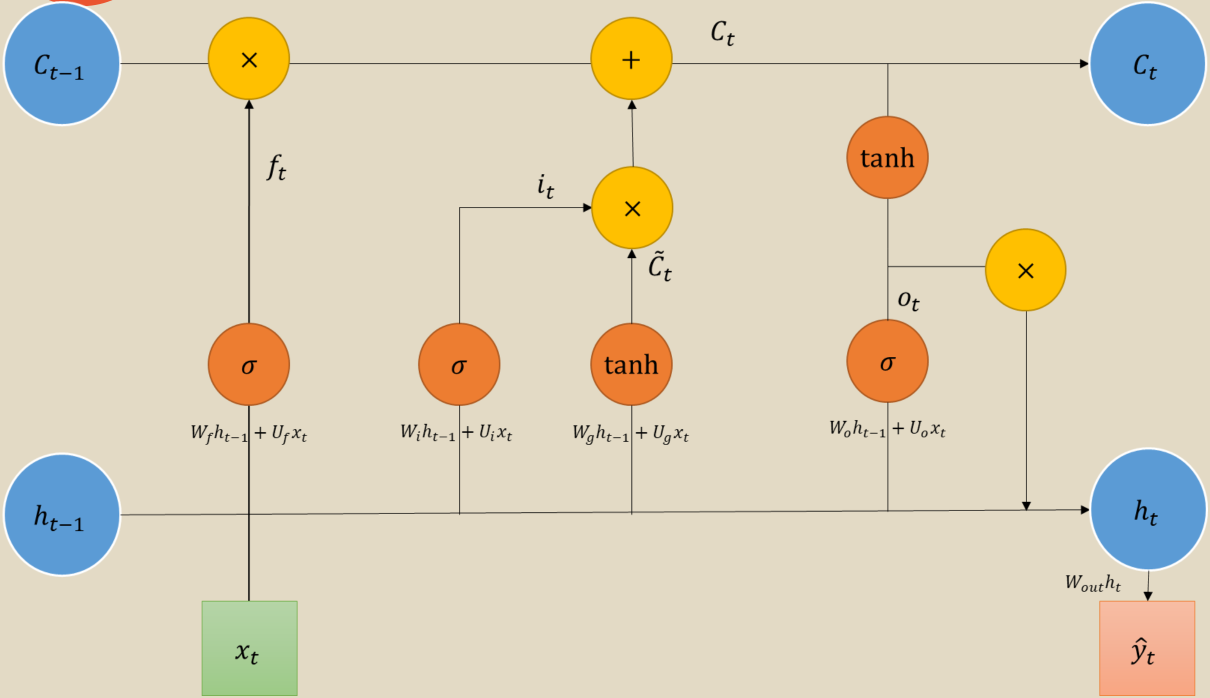
$\zeta_2 = 16$

VANISHING GRADIENT (RNN)

- $h_t = \tanh(W h_{t-1} + W_{in} x(t) + b_h)$
- Derivative of tanh = $1 - \tanh^2 x$
- $\dfrac{dE}{dW} = \dfrac{dE}{d\hat{y}} \dfrac{d\hat{y}}{\mathrm{dh_t}} \dfrac{dh_t}{\mathrm{dh_{t-1}}} \dfrac{dh_{t-1}}{\mathrm{dh_{t-2}}} \dfrac{dh_{t-2}}{dW}$
- $\dfrac{dh}{\mathrm{dh_{t-1}}} = W(1 - \tanh^2(W h_{t-1}))$
- If $W \neq 1$, gradient will vanish/grow exponentially fast



LSTM



$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$

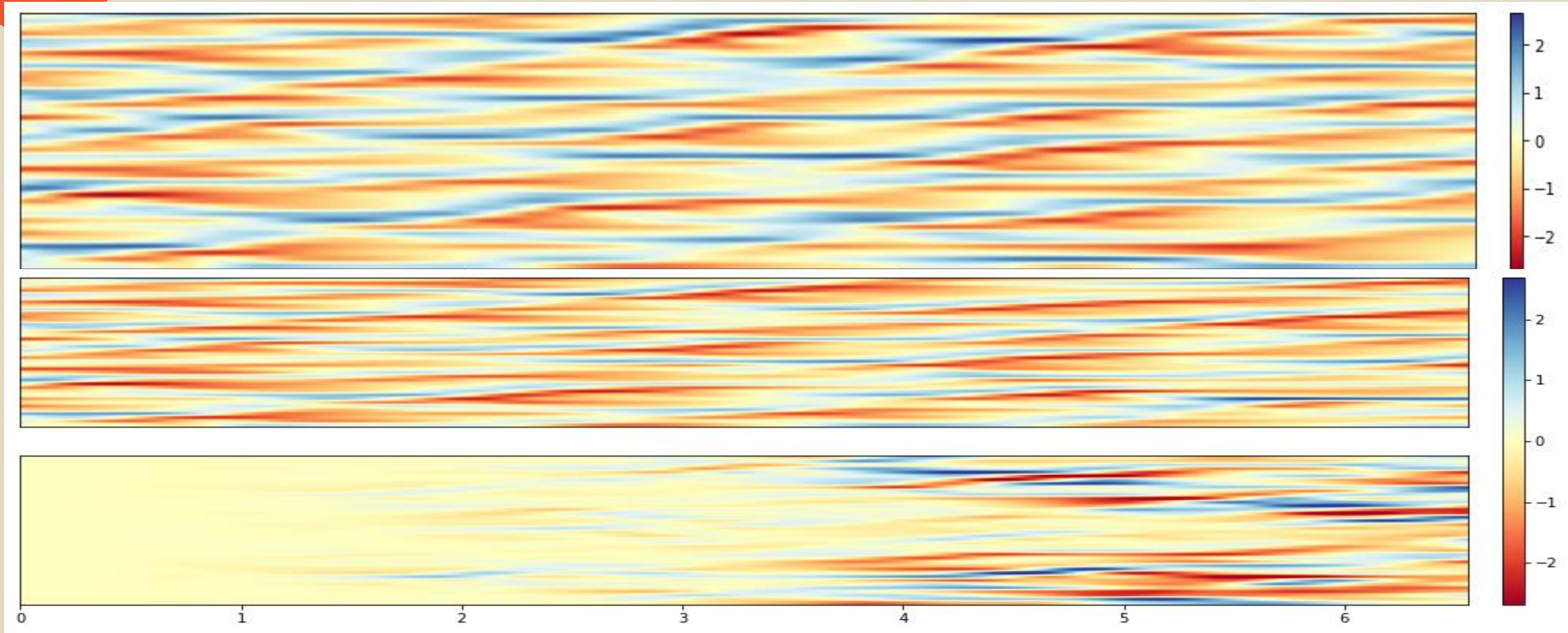
$$\tilde{C}_t = \tanh(W_g h_{t-1} + U_g x_t + b_g)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

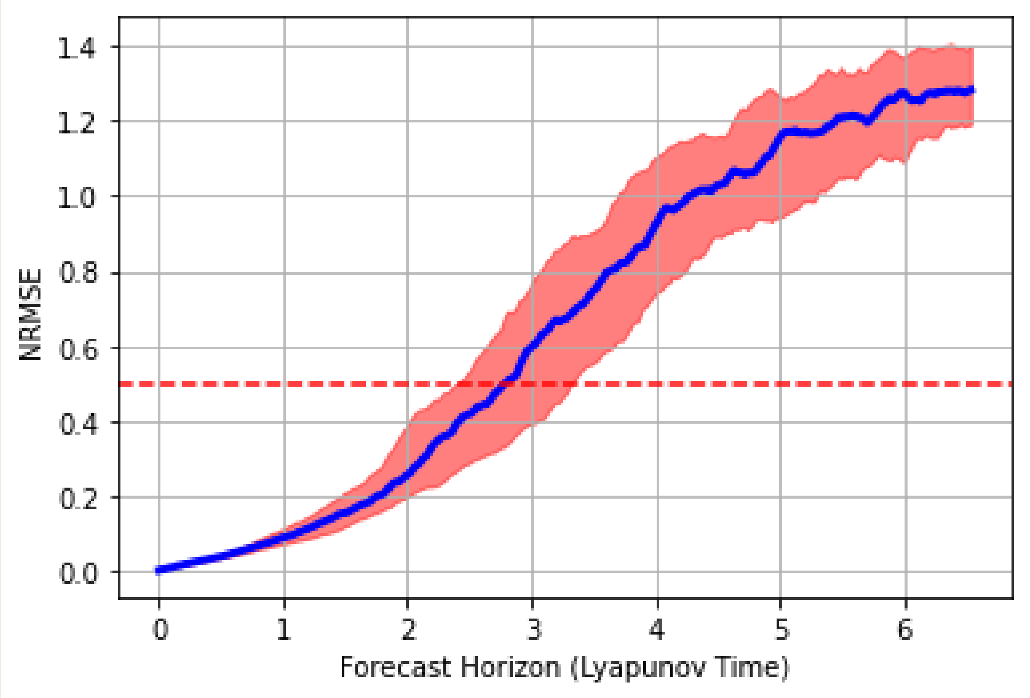
$$h_t = \tanh(C_t) \odot o_t$$

VANISHING GRADIENT? (LSTM)

- $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}t$
- $h_t = \tanh C_t \odot o_t$
- $\frac{dE}{dW} = \frac{dE}{d\hat{y}} \frac{d\hat{y}}{\mathrm{dh_t}} \frac{dh_t}{dC_t} \frac{dC_t}{dC_{t-1}} \frac{dC_{t-1}}{dC_{t-2}} \frac{dC_{t-2}}{dW}$
- $\frac{dh_t}{dC_t} = o_t(1 - \tanh^2 C_t)$
- $\frac{dC_t}{dC_{t-1}} = f_t$



(Top) Actual **(Middle)** Predicted **(Bottom)** Error



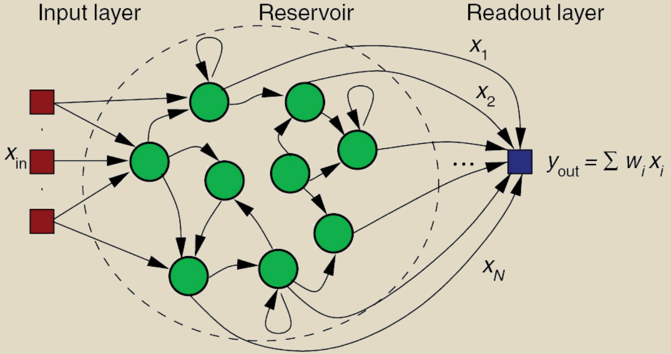
NN size = 500

$\zeta_1 = 4$

$\zeta_2 = 4$

RESERVOIR COMPUTING

RESERVOIR COMPUTING



DYNAMIC RESERVOIR

- Nonlinear
- Time-based function
- Allow diverse representation

READOUT

- Recurrence-free
- Combine signals from reservoir
- Linear & Easy to learn

Image: https://www.degruyter.com/document/doi/10.1515/nanoph-2016-0132/html

DYNAMIC RESERVOIR

- Modelled as a RNN

$$h_t = (1-\alpha)h_{t-1} + \alpha \cdot \sigma(W h_{t-1} + W_{in} x(t) + b_h)$$

- **Fixed** reservoir
    - W_{in}, W, b_h are not trained
    - Emphasis on creating a good reservoir

CREATING A GOOD RESERVOIR

BIG

Plentiful

Size of hidden layer is large

SPARSE

Loosely intercorrelated

Hidden layer weight matrix, W, is sparse

RANDOMLY CONNECTED

Different from one another

Weights of W are generated from a uniform distribution

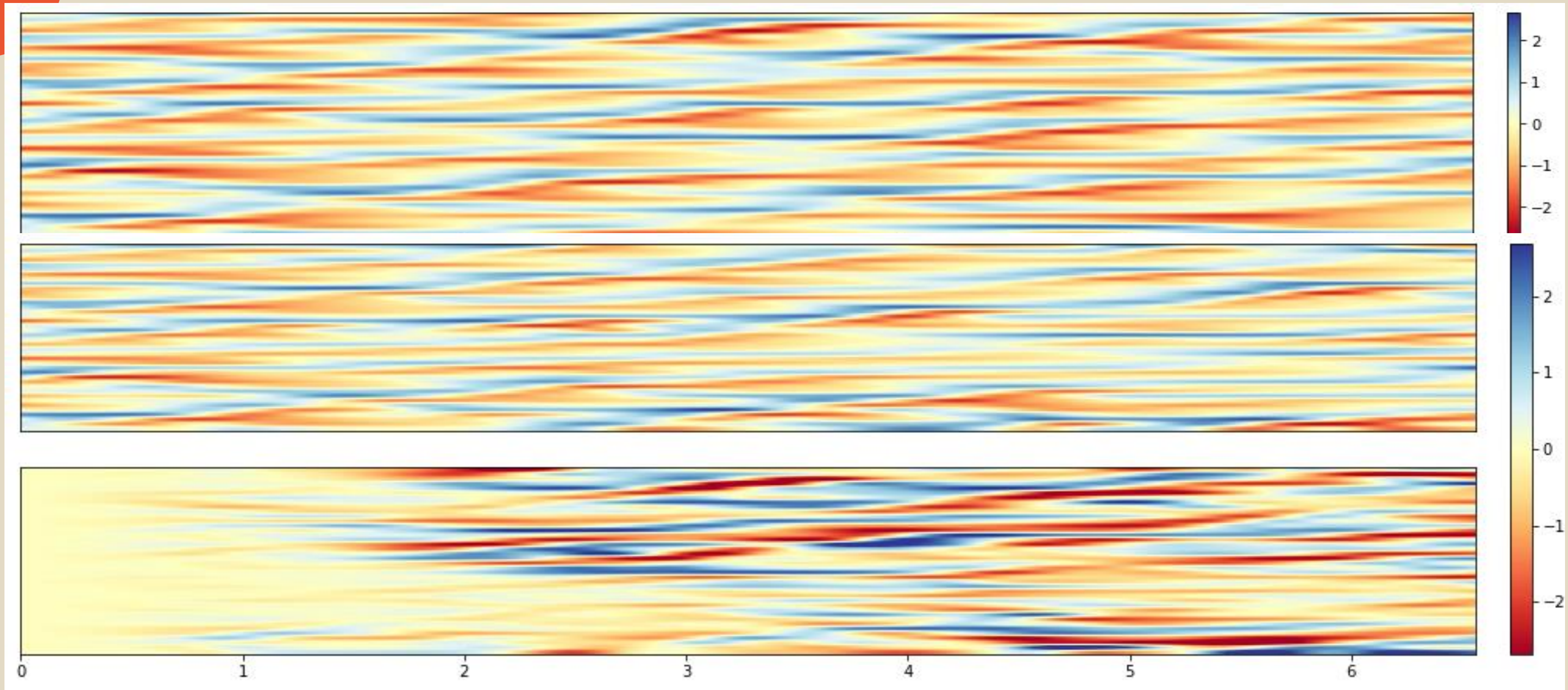
ECHO-STATE PROPERTY

- Ensure previous hidden state and previous output removed gradually within subsequent time-steps
- Spectral radius < 1
- Larger → Longer memory & longer to forget starting state
- Smaller → Useful for tasks where long memory is detrimental

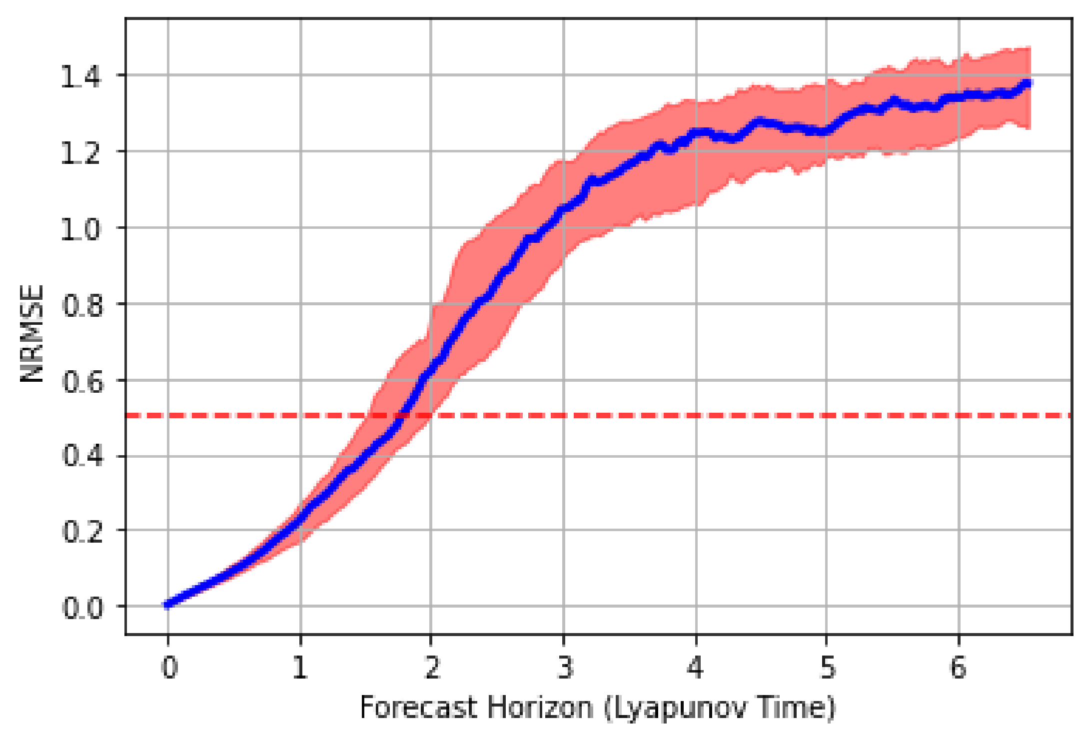
READOUT

$$\hat{y}_t = W_{out}[h_t|h_t^2] + b_{out}$$

- Only train W_{out}, b_{out}
- Linear Regression with Tikhonov Regularization
- Large dimension of $W_{out} \in \mathbb{R}^{d_0 \cdot d_h \cdot 2}$
- Use of Stochastic Gradient Descent



(Top) Actual **(Middle)** Predicted **(Bottom)** Error



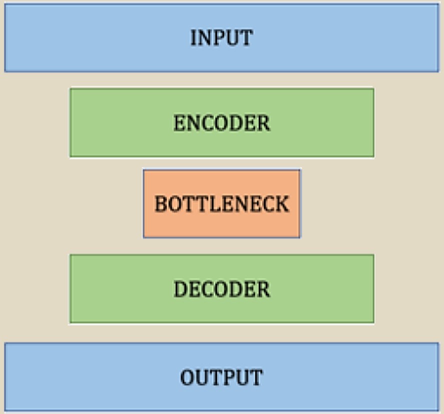
NN size = 12 000
Tikhonov reg = 1e-6
Spectral radius = 0.1
Connectivity = 4

KOOPMAN AUTOENCODER

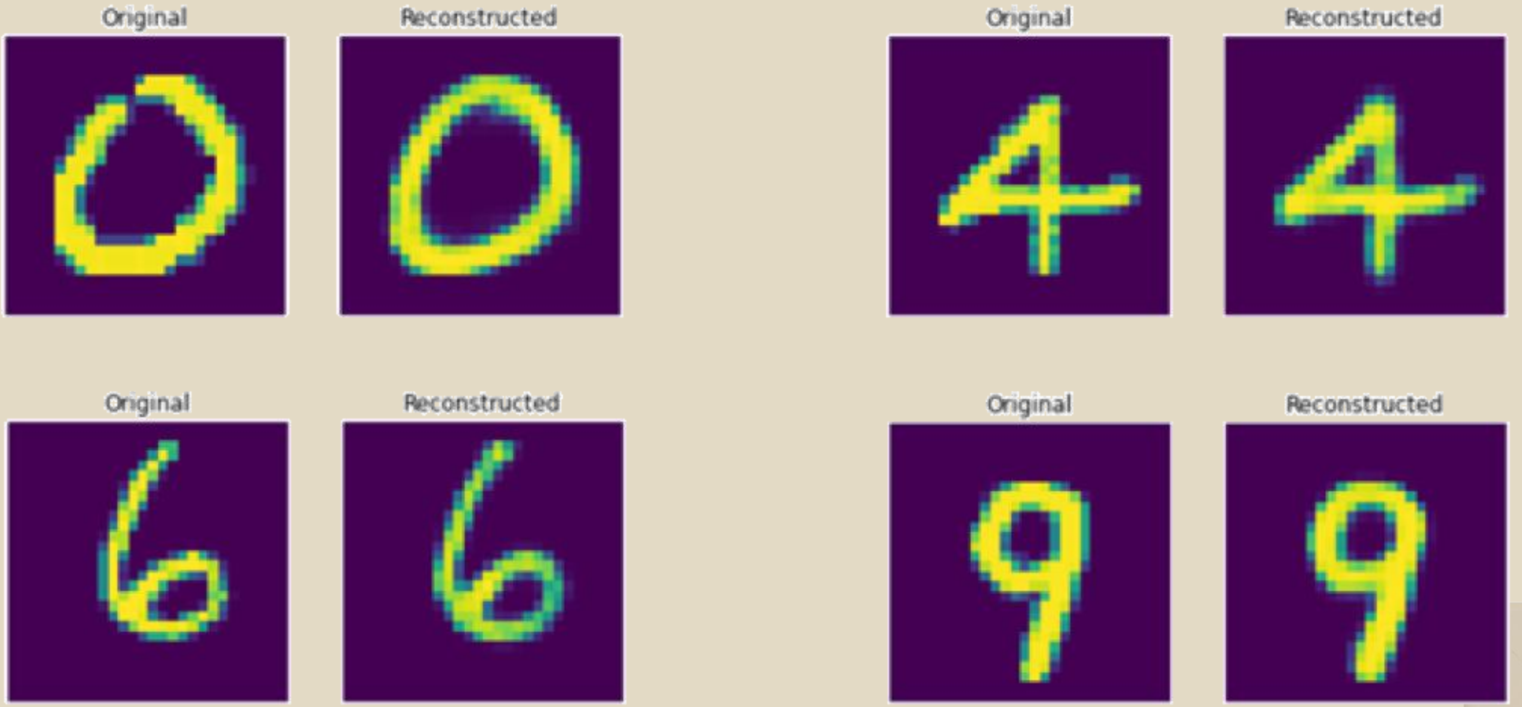
AUTOENCODERS

- Unsupervised learning method
- Extract features & perform dimensionality reduction
- Train with a reconstruction loss $= MSE(input, output)$

MNIST Dataset (28*28)
Flatten → 784
Dense (512)
Dense (128)
Dense(20)
Dense (128)
Dense (512)
Dense (784)



AUTOENCODERS



KOOPMAN AUTOENCODERS

- Koopman theory: **non-linear** dynamical system can be expressed as a **linear operator** transmitted through time
- Exist a g = set of measurement functions derived from state x
- Linear Koopman operator K that advances g forward

KOOPMAN AUTOENCODER

1

ENCODER

Find the set of measurement functions that can encode information of input

2

KOOPMAN OPERATOR

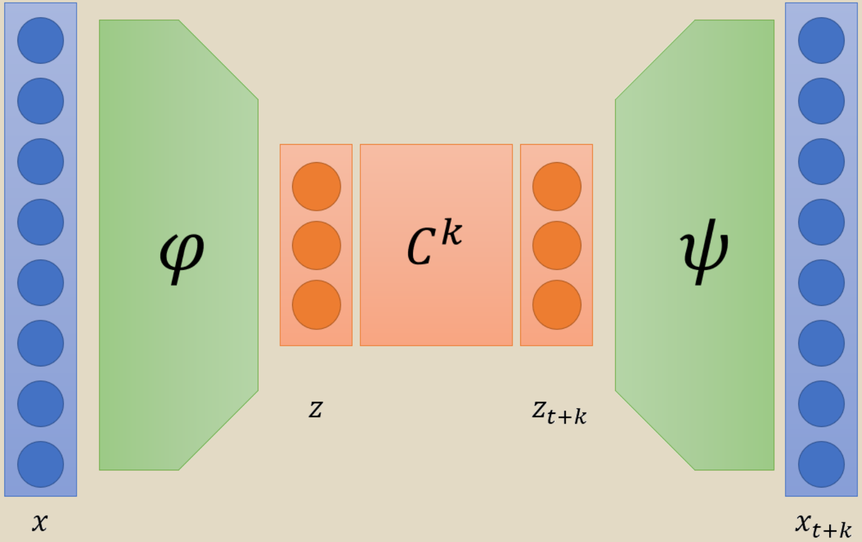
Find the Koopman Operator which learns how measurement function progresses over time

3

DECODER

Create the output based on the evolved measurement function

KOOPMAN AUTOENCODERS



$$\hat{x}_{t+1} = (\varphi \circ C \circ \psi)(x_t)$$

$$\hat{x}_{t+k} = \left(\varphi \circ C^k \circ \psi\right)(x_t)$$

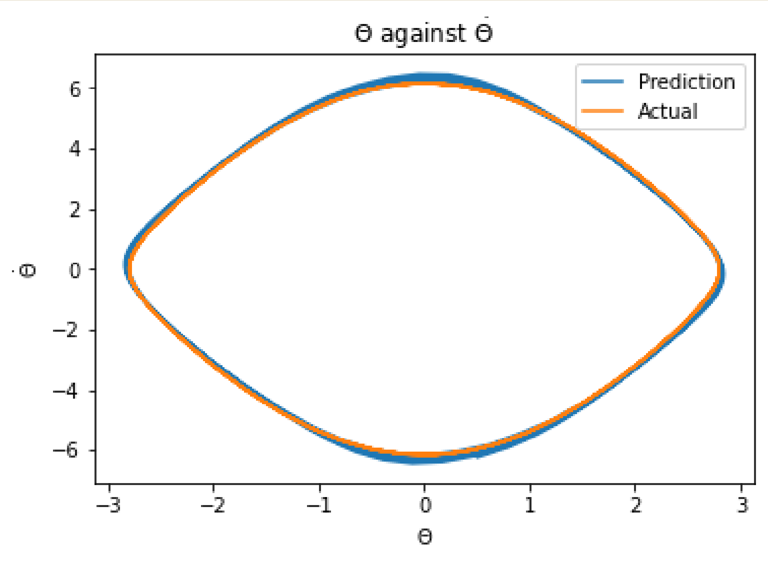
KOOPMAN AUTOENCODERS

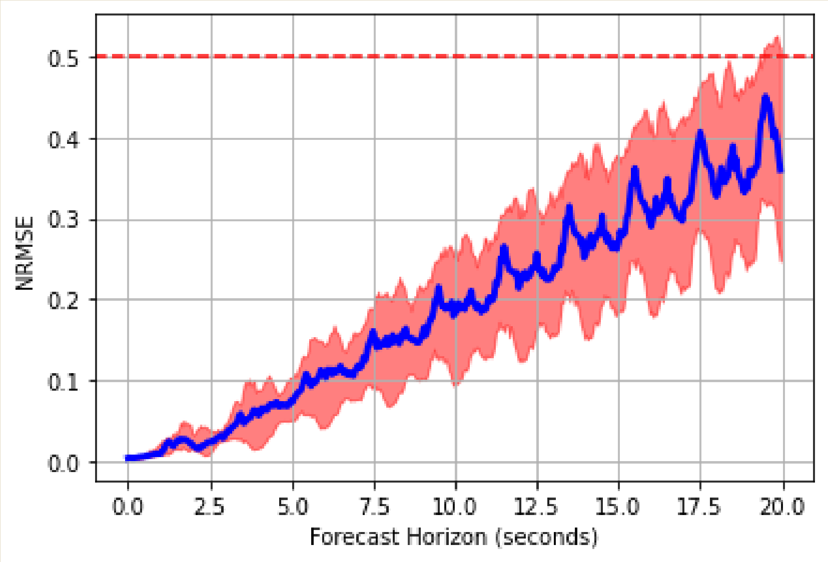
Loss composed of various parts (Azencot, 2020)

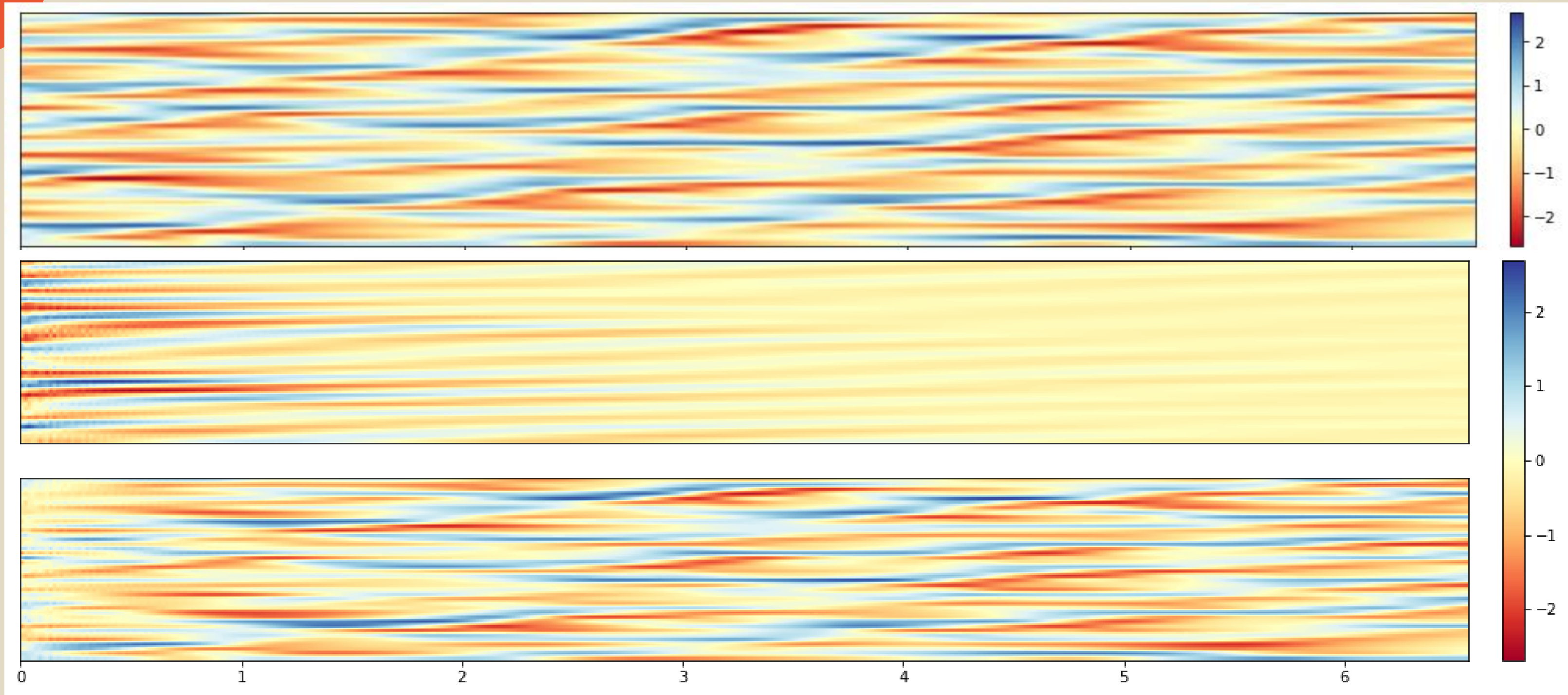
- Reconstruction Loss
- Forward Loss
- Backward Loss
- Consistency Loss

Total loss = $\sum \lambda_l loss$ weighted by different λ for each loss

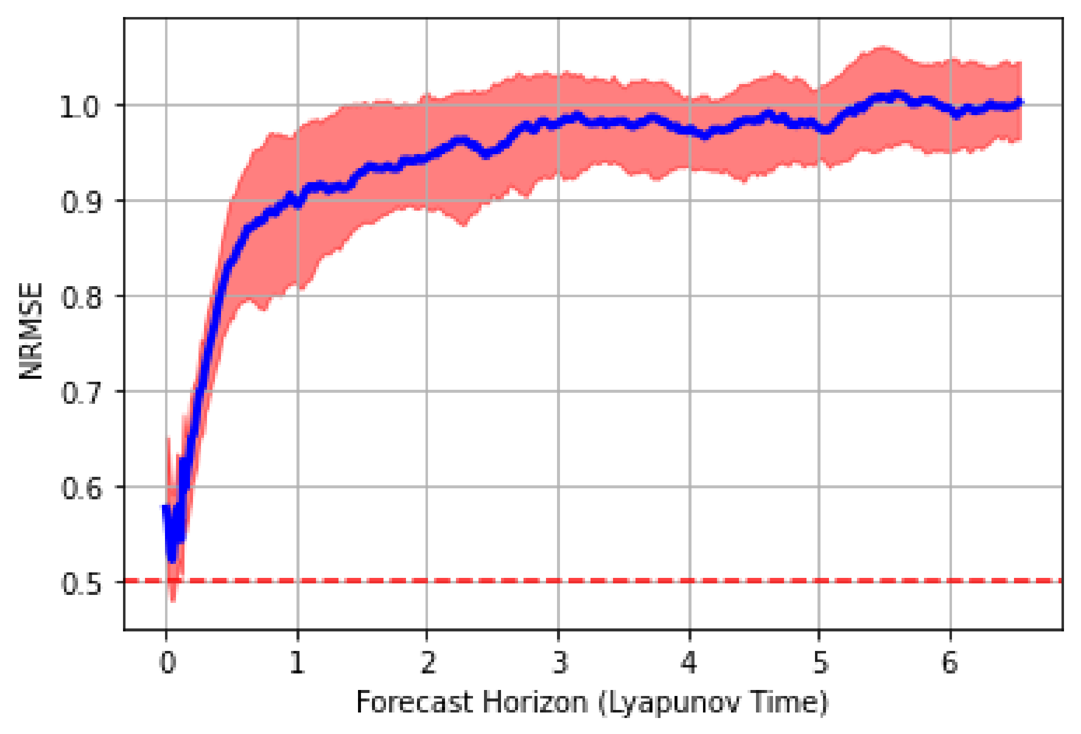
KOOPMAN AUTOENCODER (PENDULUMN)







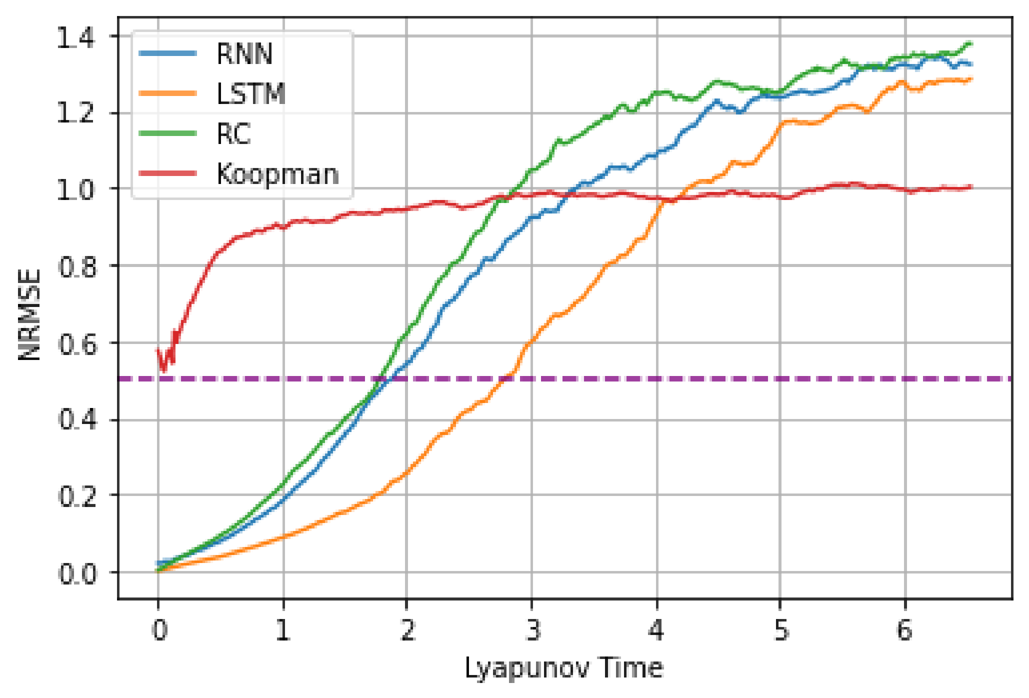
(Top) Actual **(Middle)** Predicted **(Bottom)** Error



NN size = [40, 40, 40, 40]
Time steps (loss) = 8
Loss coeff = [1, 1, 0.1, 0.01]

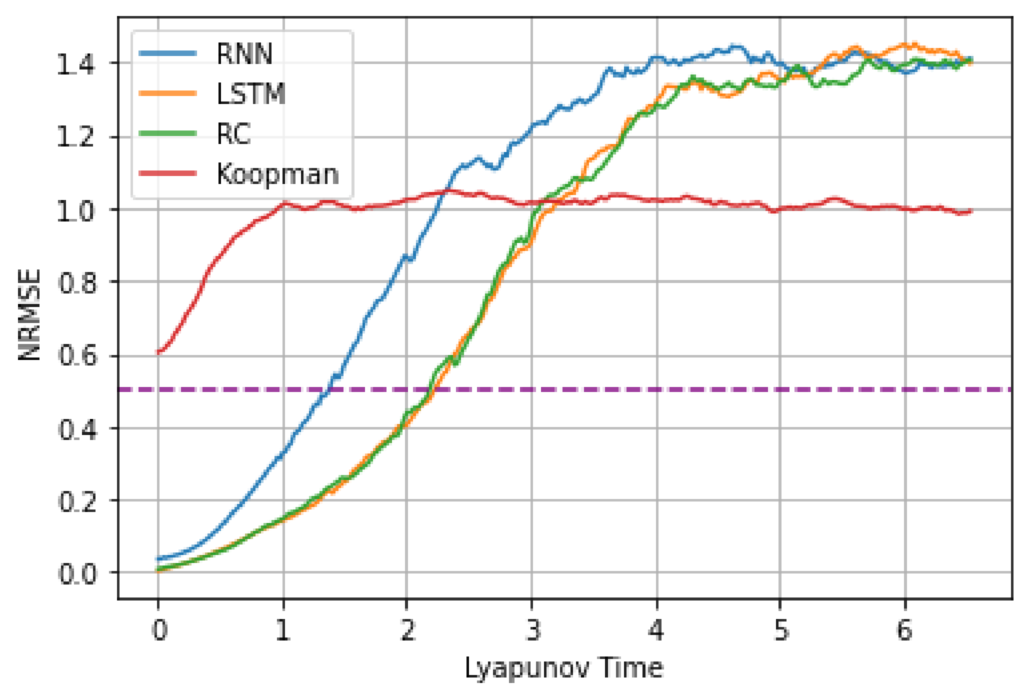
COMPARISON

LORENZ-96



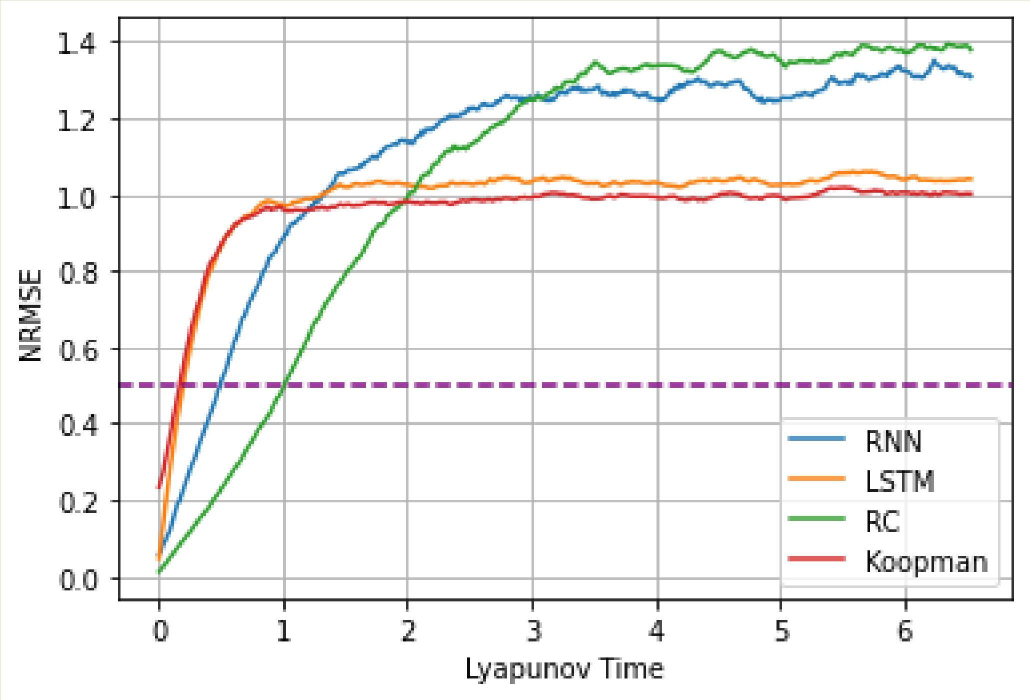
Method	Train Time (s)	Test Time (s)
RNN	1346.14	0.38
LSTM	1274.72	1.04
RC	4551.81	223.66
Koopman AE	388.51	1.58

KS EQUATION



Method	Train Time (s)	Test Time (s)
RNN	2107.61	0.39
LSTM	2762.25	0.83
RC	9498.11	672.32
Koopman AE	2025.50	0.23

LORENZ-96 (SMALL)



Method	Train Time (s)	Test Time (s)
RNN	607.94	0.39
LSTM	741.39	1.02
RC	874.03	240.77
Koopman AE	342.43	0.38

DISCUSSION

- Koopman Autoencoders in chaotic dynamical systems
- Impact of size of dataset
    - RC as a fast estimate for small datasets
- Impact of dimensionality of dataset
    - RC possibly working better at high-dimensional settings

03

Uncertainty Quantification

01

UNCERTAINTY

02

DEEP ENSEMBLES

03

MEAN-VARIANCE ESTIMATION

04

MVE DEEP ENSEMBLES

UNCERTAINTY

- How much to trust a prediction?
- Current literature
  - Low-dimension problems
  - Single-step regression problems
  - Difficult / time-consuming methods
- Goal: Find a **simple** yet **effective** way to quantify uncertainty

UNCERTAINTY

- Epistemic
  - Uncertainty caused by inadequate knowledge
  - Accuracy of the estimate of the true regression
  - Reducible error: More accurate network
- Aleatoric
  - Inherent randomness
  - Irreducible error

QUANTIFYING UNCERTAINTY

- Need to consider if prediction is able to take into account uncertainty
- Negative Log-Likelihood

$$-\log LH = \frac{1}{2}\left(d\ \log 2\pi + d \log \sigma + \frac{|x-\mu|^2}{\sigma^2}\right)$$

- NRMSE

DEEP ENSEMBLES

DEEP ENSEMBLES

- Originally proposed as a bootstrap method (Heskes, 1996)
- NN with different initializations
- Run for n_{run} times

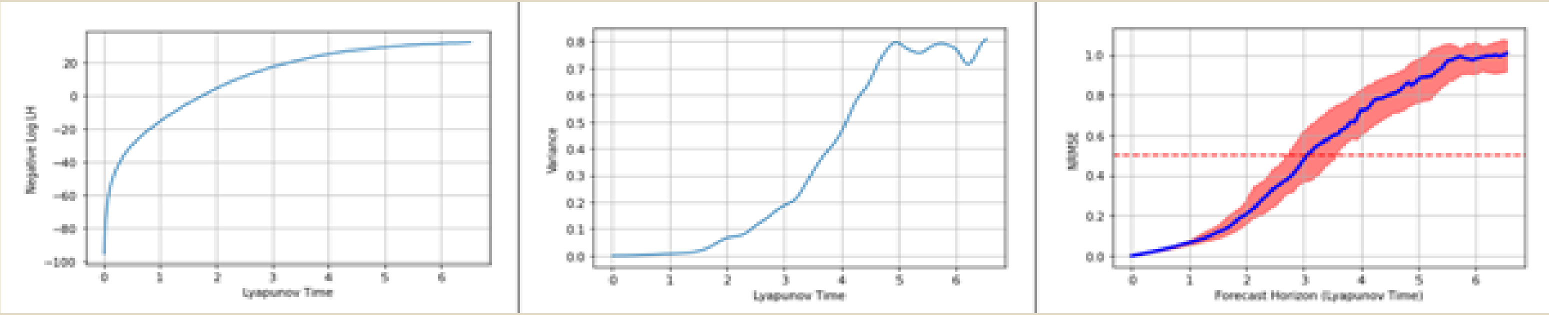
$$\hat{\mu} = \frac{1}{n_{run}} \sum_{i=1}^{n_{run}} \hat{y}_i$$

$$\hat{\sigma}^2 = \frac{1}{n_{run} - 1} \sum_{i=1}^{n_{run}} (\hat{y}_i - \hat{\mu})^2$$

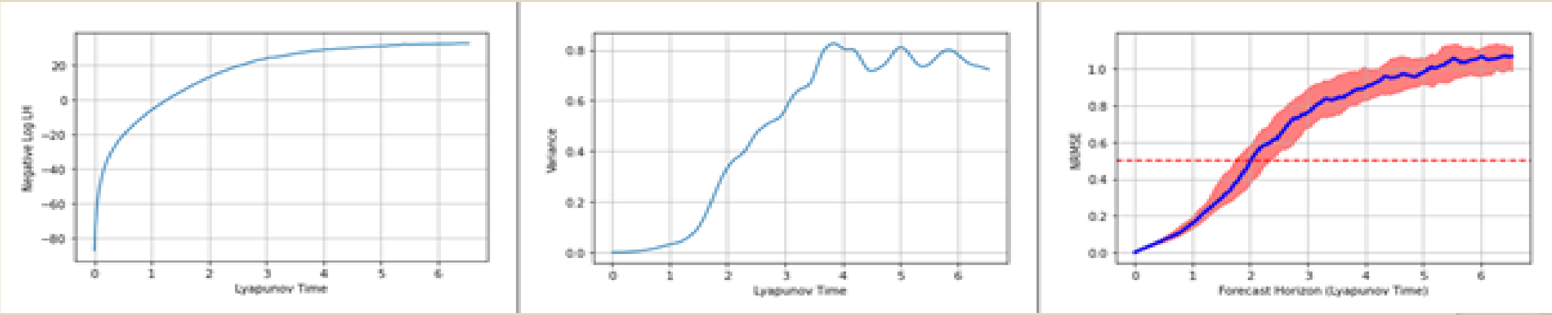
- Target epistemic uncertainty → can be used to provide CI

$$n_{run} = 5$$

LSTM



RC

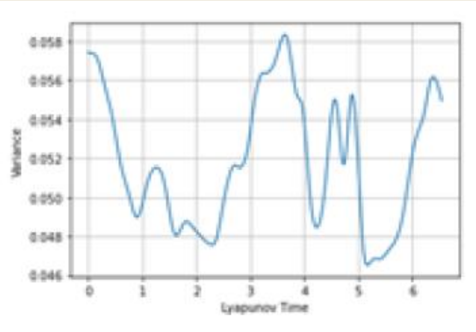


MEAN VARIANCE ESTIMATION

MEAN VARIANCE ESTIMATION

- Traditional regression seeks to optimize for MSE loss
- Goal: Compute a standard deviation, σ
- NN to predict both a μ, σ
- Optimize for Negative Log Likelihood instead

MEAN VARIANCE ESTIMATION

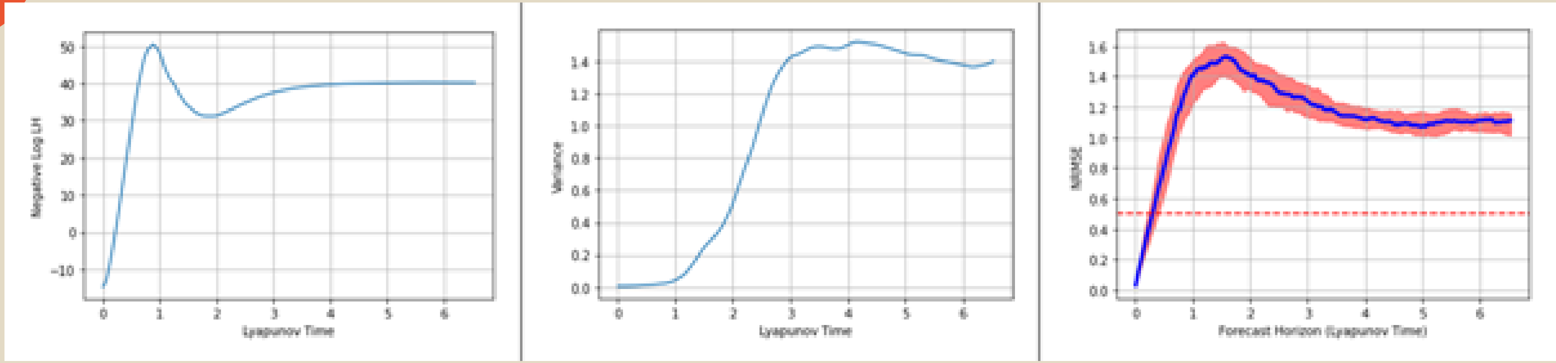


- Problem: Relatively constant variance
- Solution: Sampling
- Use μ_{t0} and σ_{t0}^2 to generate next input
  - $x_{t1} \sim N(\mu_{t0}, \sigma_{t0}^2)$
- Repeat for n_{traj} trajectories → n_{traj} (μ, σ) per time-step
- Mixture of Gaussians

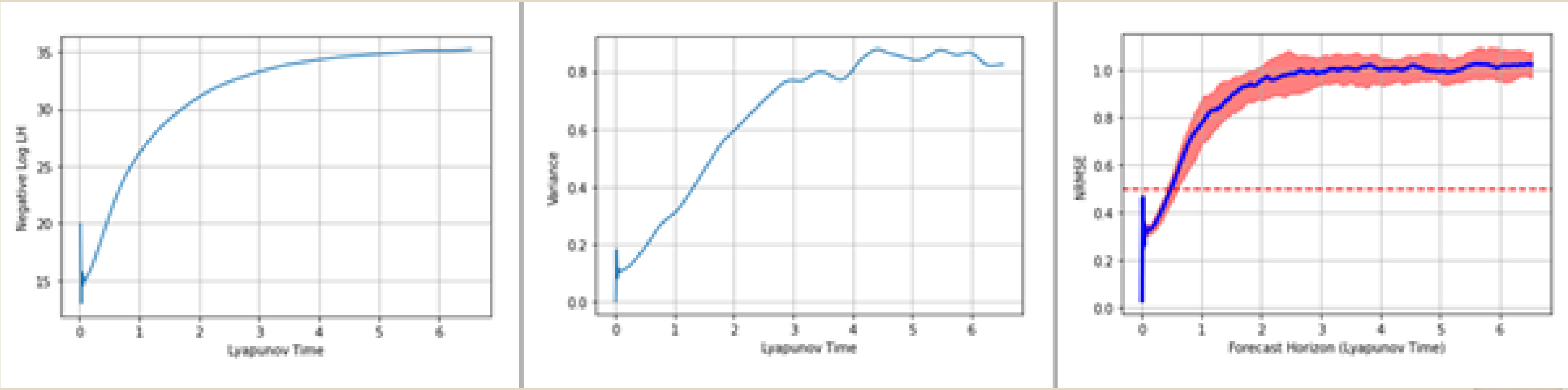
$$\mu_* = \frac{1}{n_{traj}} \sum_{i=1}^{n_{traj}} \mu_i$$

$$\sigma_*^2 = [\frac{1}{n_{traj}} \sum_{i=1}^{n_{traj}} (\mu_i^2 + \sigma_i^2)] - \mu_*^2$$

LSTM $n_{traj} = 100$



RC $n_{traj} = 20$

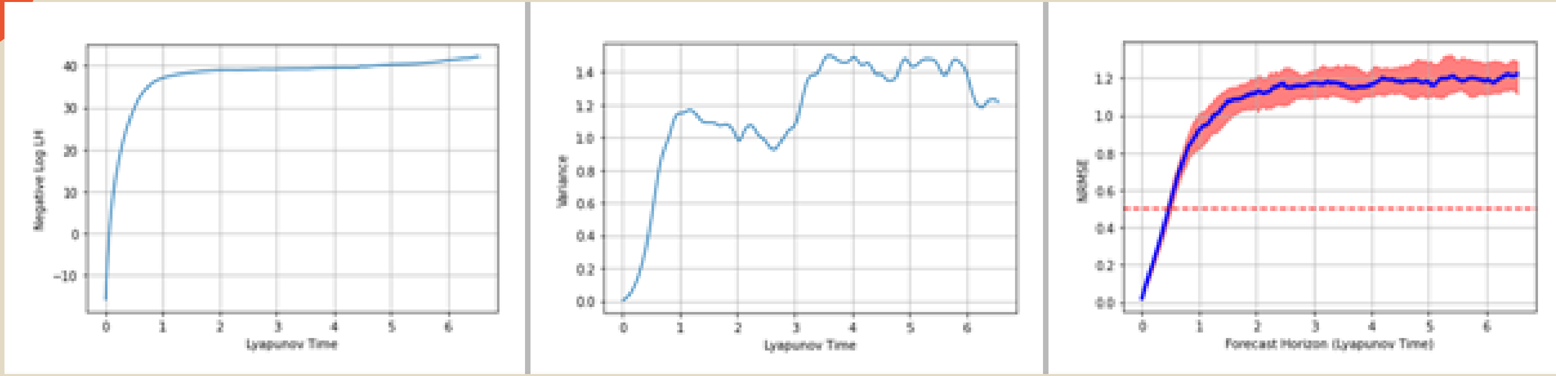


MVE DEEP ENSEMBLES

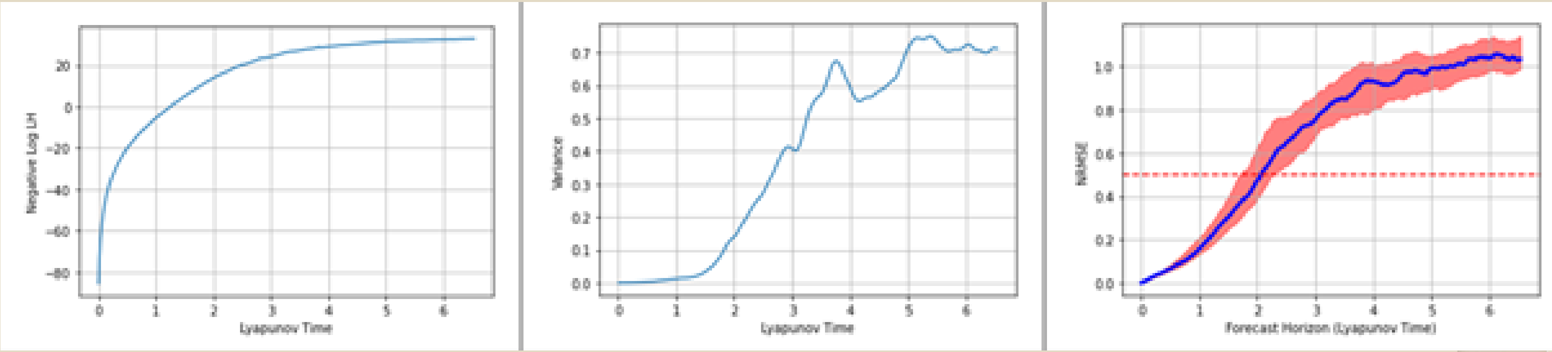
MVE DEEP ENSEMBLES

- Combine the ideas of the Deep Ensembles and MVE
- NN produces μ, σ which is optimized for NLL
- Repeated $n_{run} = 5$ times
- n_{run} (μ, σ^2) for each time-step → Gaussian mixture

LSTM


Negative Log LH
Lyapunov Time
Variance
Lyapunov Time
NRMSE
Forecast Horizon (Lyapunov Time)

RC

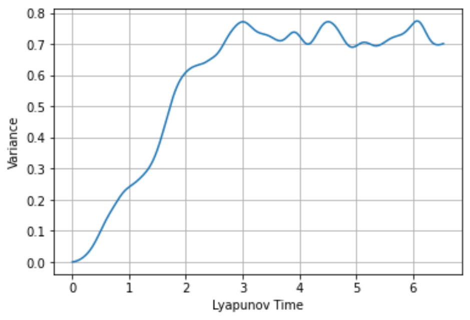

Negative Log LH
Lyapunov Time
Variance
Lyapunov Time
NRMSE
Forecast Horizon (Lyapunov Time)

Method	Deep Ensemble	MVE w/o Sampling	MVE w/ Sampling	MVE Deep Ensemble
LSTM	10.14	406.40	35.91	37.32
RC	15.87	4178810.66	29.71	16.18

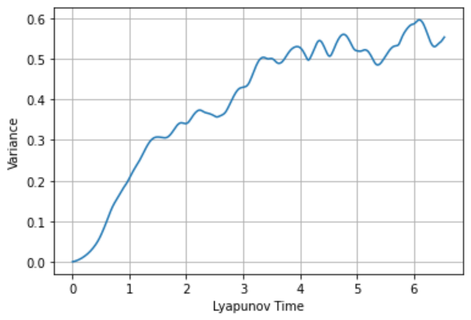
- Deep Ensembles seem to be most effective
  - Time-consuming
- Difficult for LSTM to learn due to overfitting
  - Clean data & No overlapping points
- Adding noise?

Method	Deep Ensemble	MVE w/o Sampling	MVE w/ Sampling	MVE Deep Ensemble
RC	26.21	563.27	31.12	26.44

- Original MVE has lower NLL
- MVE Deep Ensembles smaller PI



Deep Ensemble



Combined

DISCUSSION

- Deep ensembles performed the best
- Overfitting in learning
  - Impose a prior and perform Maximum a Posteriori (MAP) estimation
- Role of noise
  - Potential benefits of MVE – Fast + Size of Prediction Intervals
  - Performing perturbations during training

04

Discussion

CONCLUSION

- Prediction
  - Used common benchmarks of Lorenz-96 and KS Equation
  - LSTMs performed best with abundant data
  - RCs perform better with limited data
  - Koopman Autoencoders unable to predict chaotic systems
- Uncertainty Quantification
  - Compared simple and easily-implementable UQ methods
  - Deep ensembles most effective

FUTURE WORK

- Investigate relationship of data dimensionality
- Use of real time-series data
  - Better investigate parameters related to time-dependencies
  - Presence of noisy data
- Innovating Mean Variance Estimation
  - Prevent overfitting
  - Injecting noise to improve uncertainty bounds

THANK YOU

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, infographics & images by Freepik and illustrations by Storyset