HOW TO TRAIN YOUR RESERVOIR COMPUTER

AUTHOR: BEN JOHNSON

MENTORS: ANDREW POMERANCE, DANIEL CANADAY, & COLIN MCCANN

INTRODUCTION: MAIN OBJECTIVES

01

Using a reservoir computer (RC) to model physical systems represented by differential equation

02

Implementing a hybrid RC (part machine learning/part computational methods)

03

Testing different ways to train an RC (Gaussian averaging, direct input, autoencoder)

INTRODUCTION: MAIN OBJECTIVES

01

Using a reservoir computer (RC) to model physical systems represented by differential equation

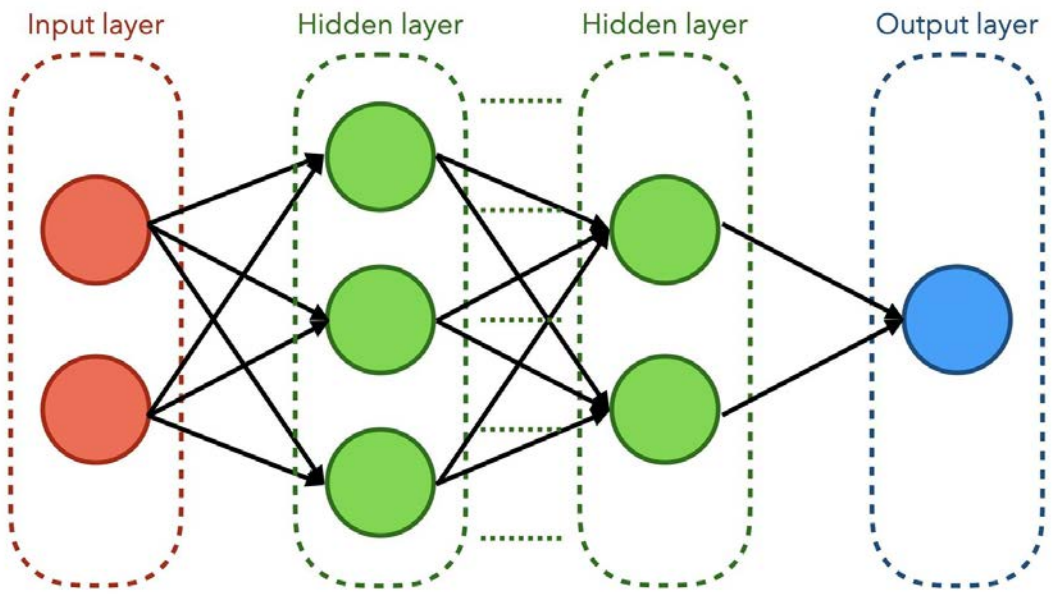
02

Implem hybrid RC (part ine learnin t com tional m ds)

03

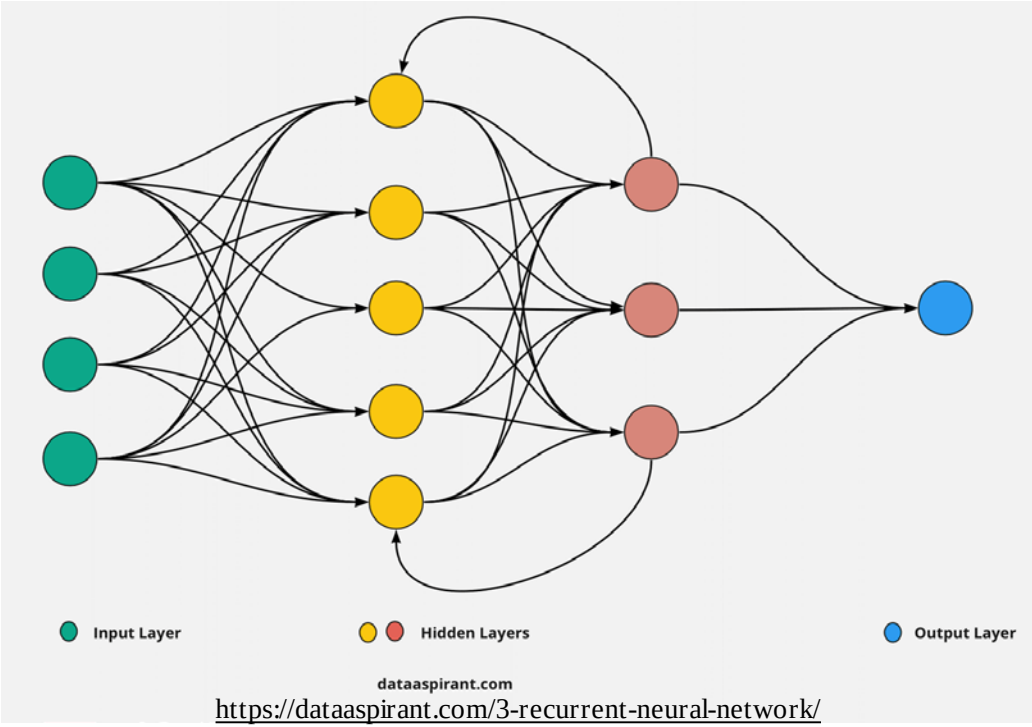
Testing t t ways to train

Feed-Forward



https://towardsdatascience.com/an-illustrated-guide-to-artificial-neural-networks-f149a549ba74

Recurrent



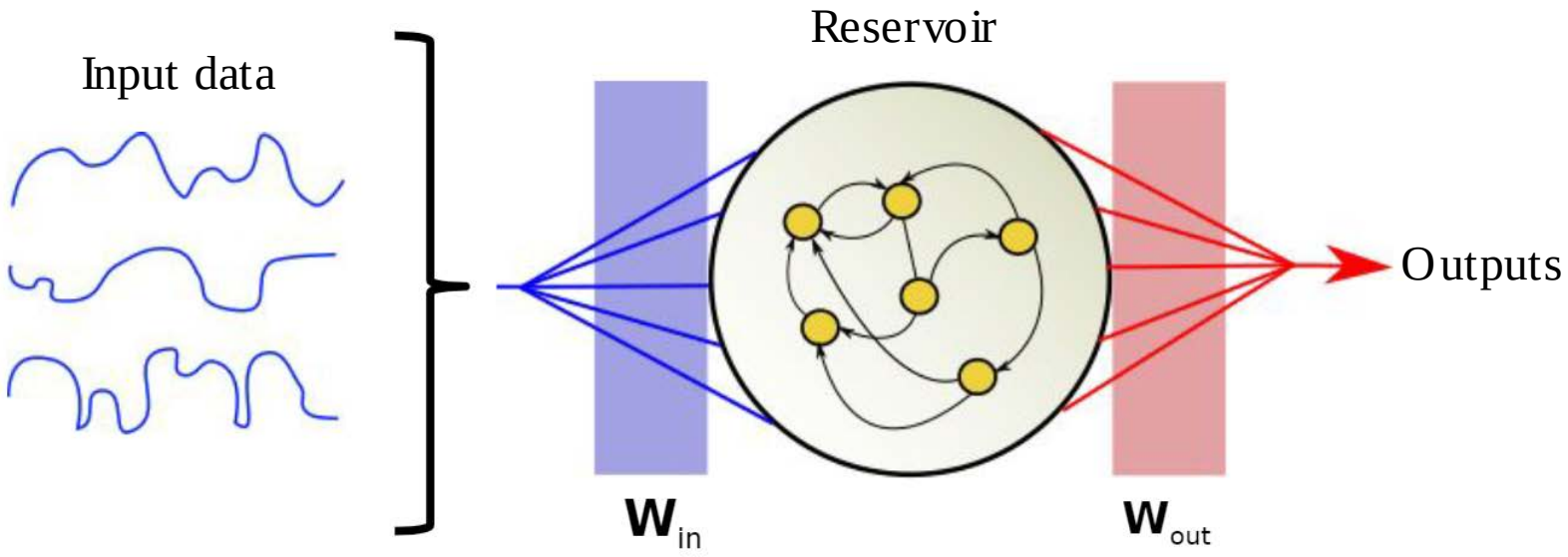
https://dataaspirant.com/3-recurrent-neural-network/

NEURAL NETWORKS

- Neural networks: inspired by neuroscience
- Feed-forward networks: simplest & most commonly used
- Recurrent networks: some advantages, but more difficult to train
- Reservoir computers: have memory like recurrent networks but easier to train

TRAINING A RESERVOIR COMPUTER

- Raw data is passed through input weight matrices and translated to reservoir states
- Prediction is translated from reservoir states to output data by the output weight matrices
- RC's only require training of the output weight matrices (MUCH faster & simpler)

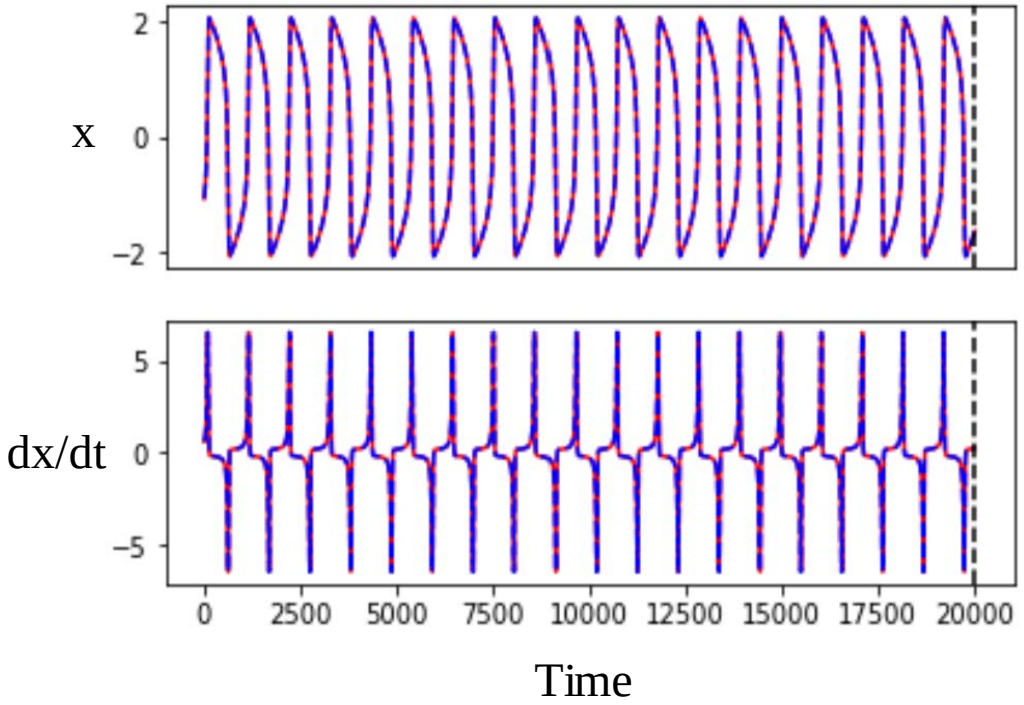


VAN DER POL EQUATION

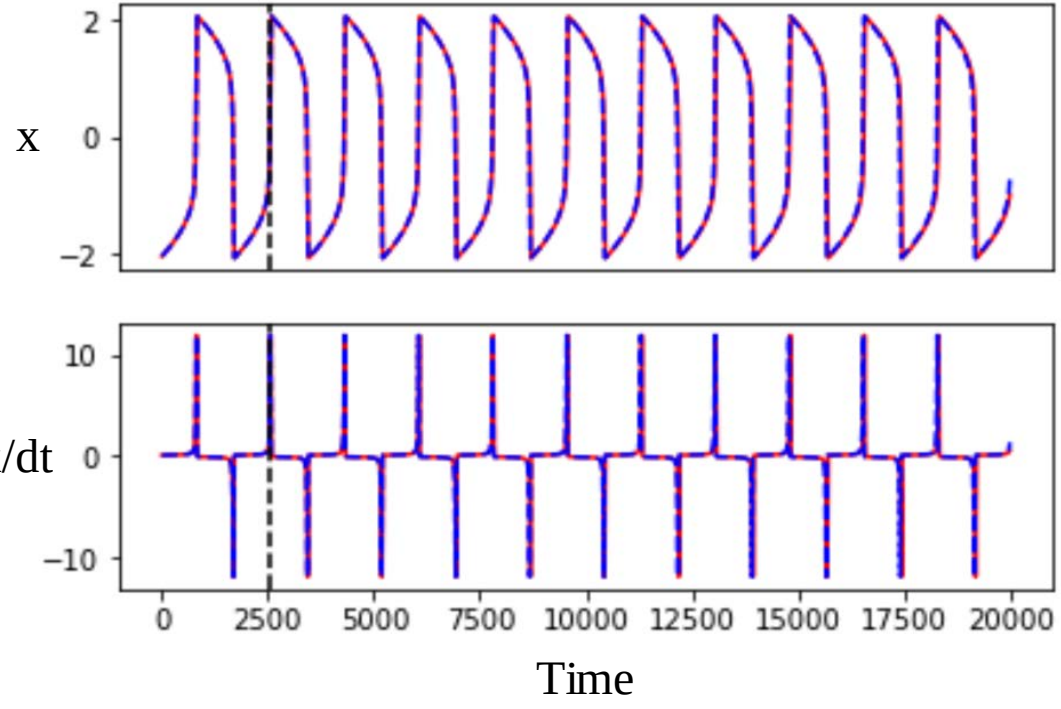
- Started with relatively simple ODE (second-order, one dimensional)
- Models non-linear oscillator

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = 0$$

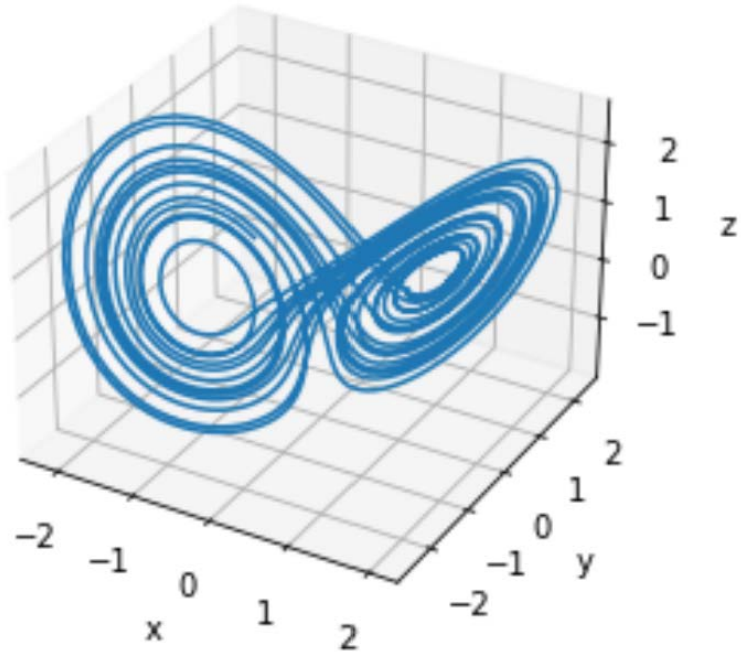
μ = 4



μ = 8



Example Lorenz Solution

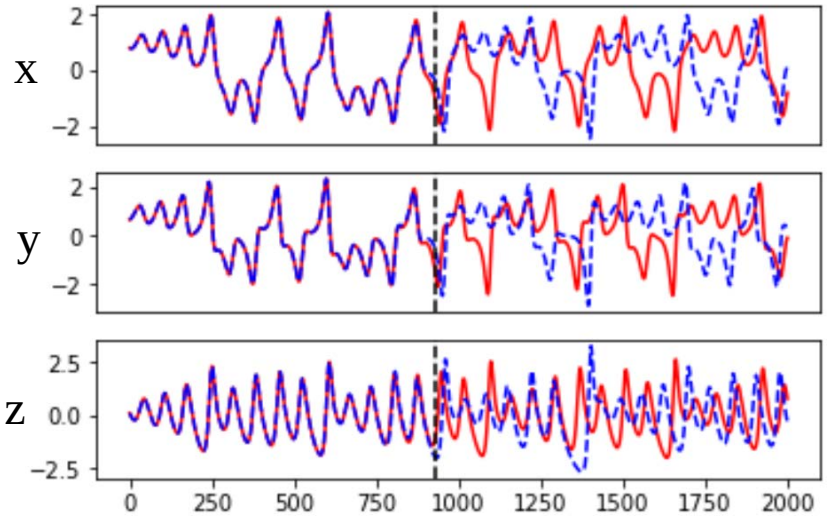


$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x),$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y,$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z.$$

RC Prediction (red) vs. Model (blue)



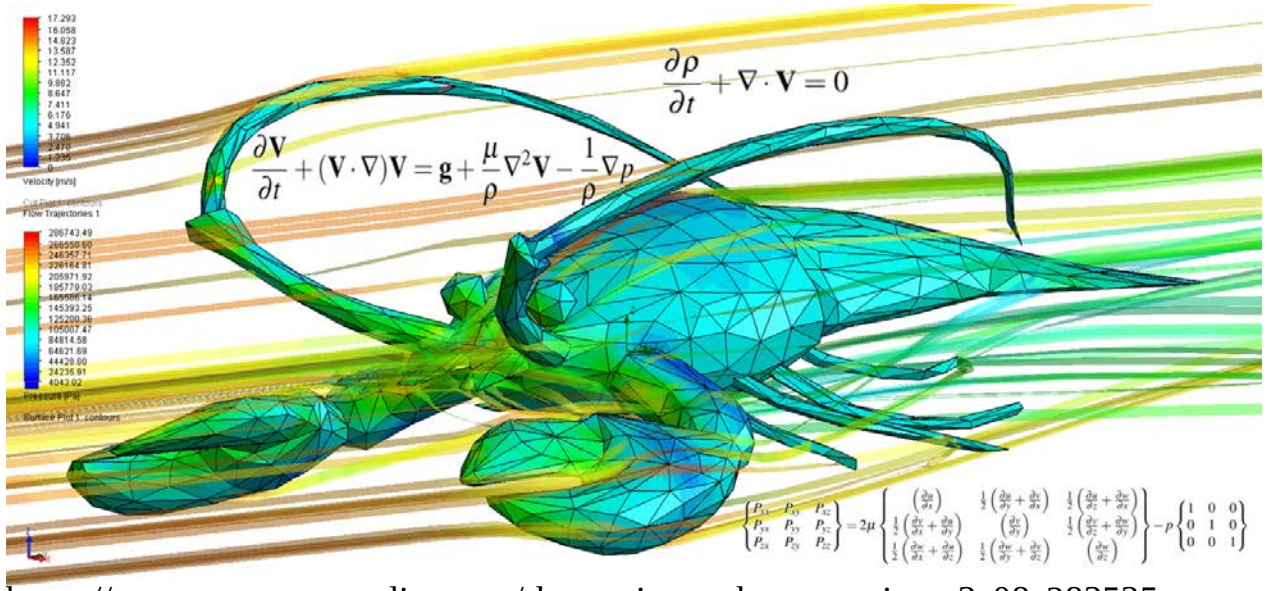
LORENZ SYSTEM

- Moved onto more complicated system of equations
- Simplified model of heat convection in a fluid
- First-order but non-linear, coupled, & chaotic

CONCLUSIONS

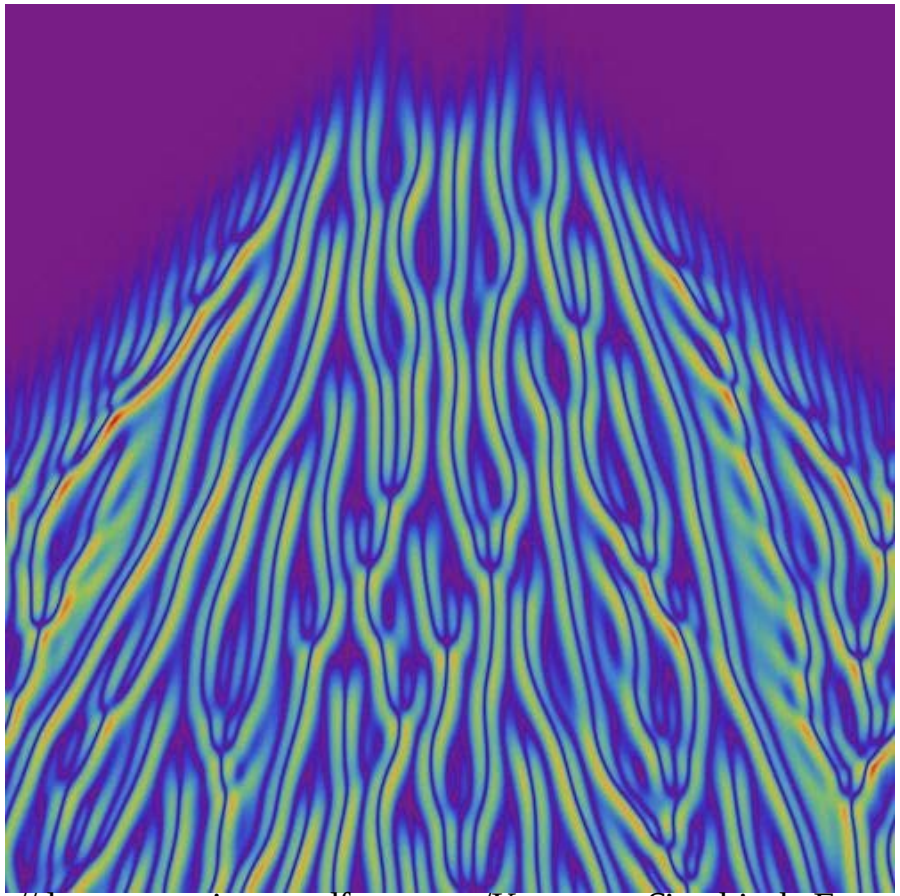
- Successes: Gaussian averaging & direct input methods
- Future work: hybrid RC, tackling more complicated systems (NS/KS)
- Thanks for listening!
- Questions?

Navier-Stokes Equations



https://www.cantorsparadise.com/the-navier-stokes-equation-a3a08c283535

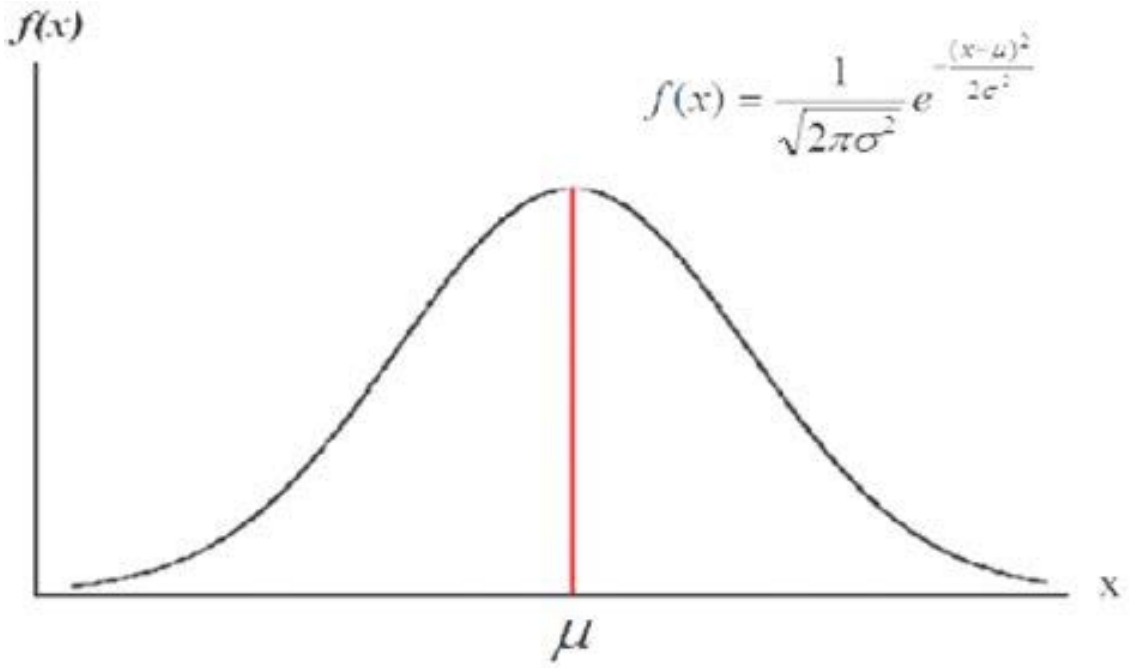
Kuramoto-Sivashinsky Equation



https://demonstrations.wolfram.com/KuramotoSivashinskyEquation

GAUSSIAN AVERAGING METHOD

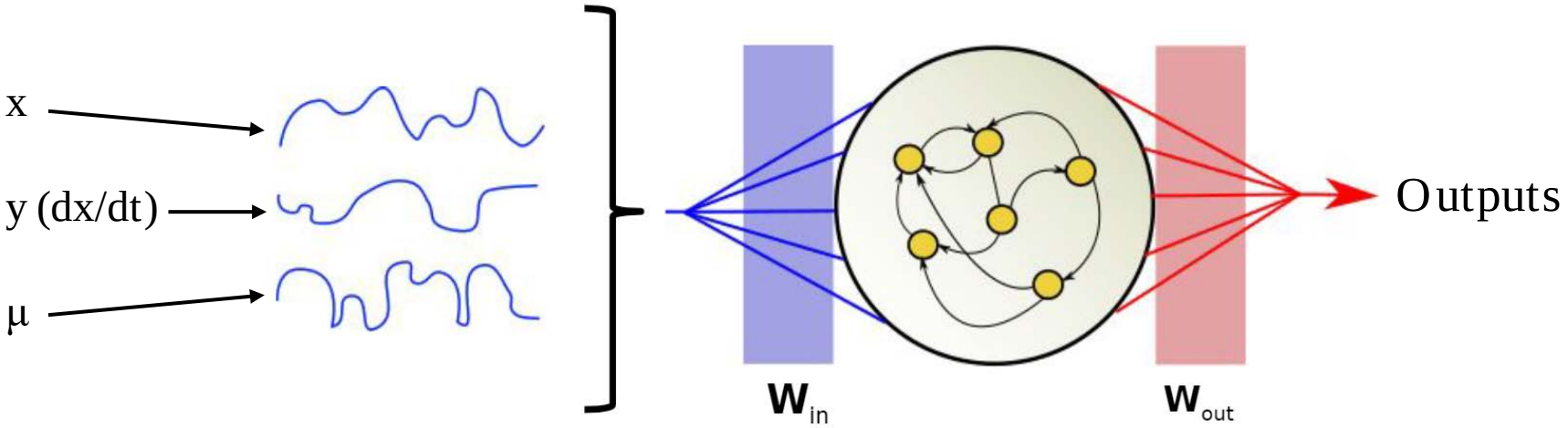
- RC trained over a range of parameter values and allowed to create its own output weight matrices based on target
- For new parameter value, function averages given weight matrices based on distance from their associated parameter values
- This method worked for the Van der Pol equation and Lorenz system well



https://www.researchgate.net/figure/Plot-representing-a-Gaussian-function_fig12_309323775

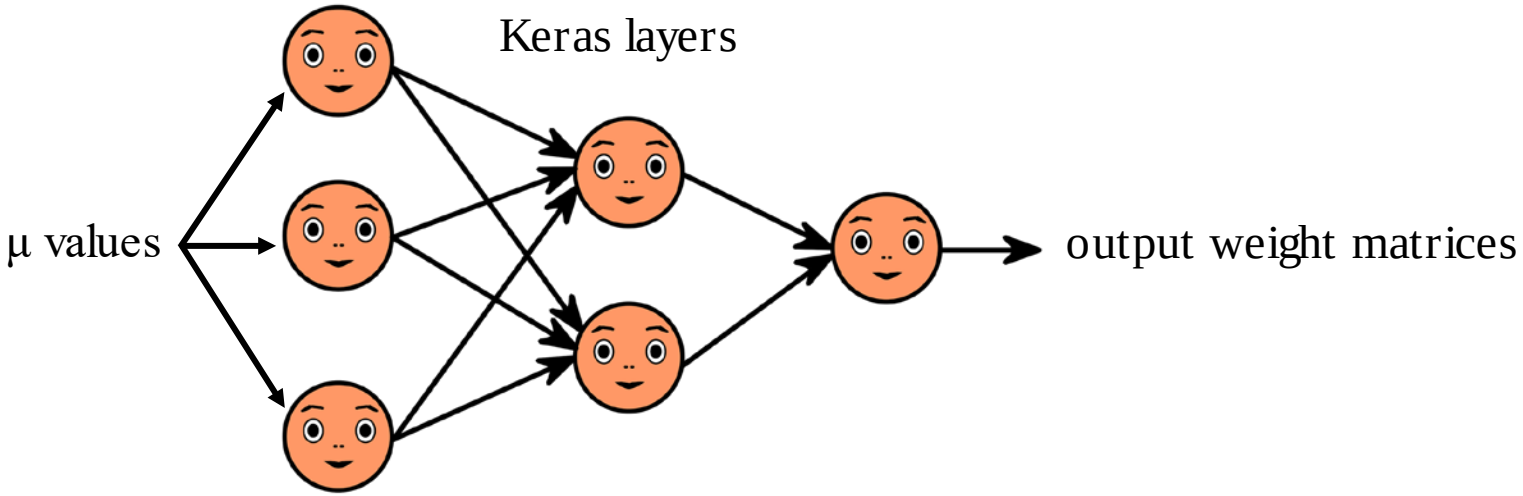
DIRECT INPUT METHOD

- The parameters were passed into the RC directly as an additional dimension
- The RC analyzed the parameter's effect on the system during training
- The RC then determined its own output weight matrix based on the behaviors discovered during training
- Worked for Van der Pol, not yet attempted for Lorenz



FEED-FORWARD METHOD

- After training, Keras machine learning algorithm analyzed the parameters given as inputs to RC and the weight matrices determined during the training process
- This separate network was trained to map directly from parameter values to weight matrices based on RC results
- Wasn't precise enough for Van der Pol, not attempted for Lorenz



https://eduawarenessclub.blogspot.com/2020/09/artificial-neural-network-and-deep.html