Tackling climate change problems with machine learning methodologies: prospects for climate extremes, geological and geo-coastal modeling

Dr. Rohitash Chandra

Centre for Translational Data Science
http://sydney.edu.au/data-science

Seminar hosted by:
EarthByte Group
School of Geosciences
www.earthbyte.org

THE UNIVERSITY OF SYDNEY

Abstract

Machine learning methods such as neural networks and deep learning are increasingly becoming popular with innovations in social media, e-commerce, and autonomous driving systems. This motivates for its application for climate change-related problems that involve time series data and models that try to model biological or geophysical processes, for instance, predicting the trend in the growth of corals and landscape dynamics. This seminar will introduce basics of machine learning and neural networks to ecologists and geoscientists. It will begin with some recent applications of neural networks for cyclone track and wind intensity prediction. It will also present uncertainty quantification in neural networks via Bayesian learning for tackling climate change problems. Moreover, it will provide a roadmap for using ensemble, multi-task and transfer learning methodologies via neural networks for cyclone prediction and related applications that deal with multiple sources of data. The prospects of machine learning and Bayesian optimization in Basin and Landscape Dynamics (Badlands) and PyReef software that predicts reef evolution will also be discussed.

Overview

1. Introduction to machine learning and neural networks
2. Neural networks for climate extremes: cyclones
   - Encoding cyclone track prediction problem in coevolutionary RNNs
   - Minimal timespan problem for cyclone wind-intensity prediction
   - Rapid intensification identification in tropical cyclones
   - Transfer and Multi-task learning
   - Multi-task learning for wind intensity prediction
   - Multi-task learning for multi-step-ahead prediction
3. Uncertainty quantification with Bayesian methods
4. Landscape dynamics and geocoastal modeling
5. Prospects for machine learning in climate modeling

Introduction to machine learning and neural networks

Machine learning

- Machine learning is the processes of learning from a data set using a computer program.
- The process of learning involves adjusting some variables so that the data can be explained by the model. The process of learning can also be seen an optimisation problem.
- Machine learning algorithms are used to learn from the data for problem which involve *classification*, *prediction*, *clustering* and *control*.

Brain vs Machine Learning

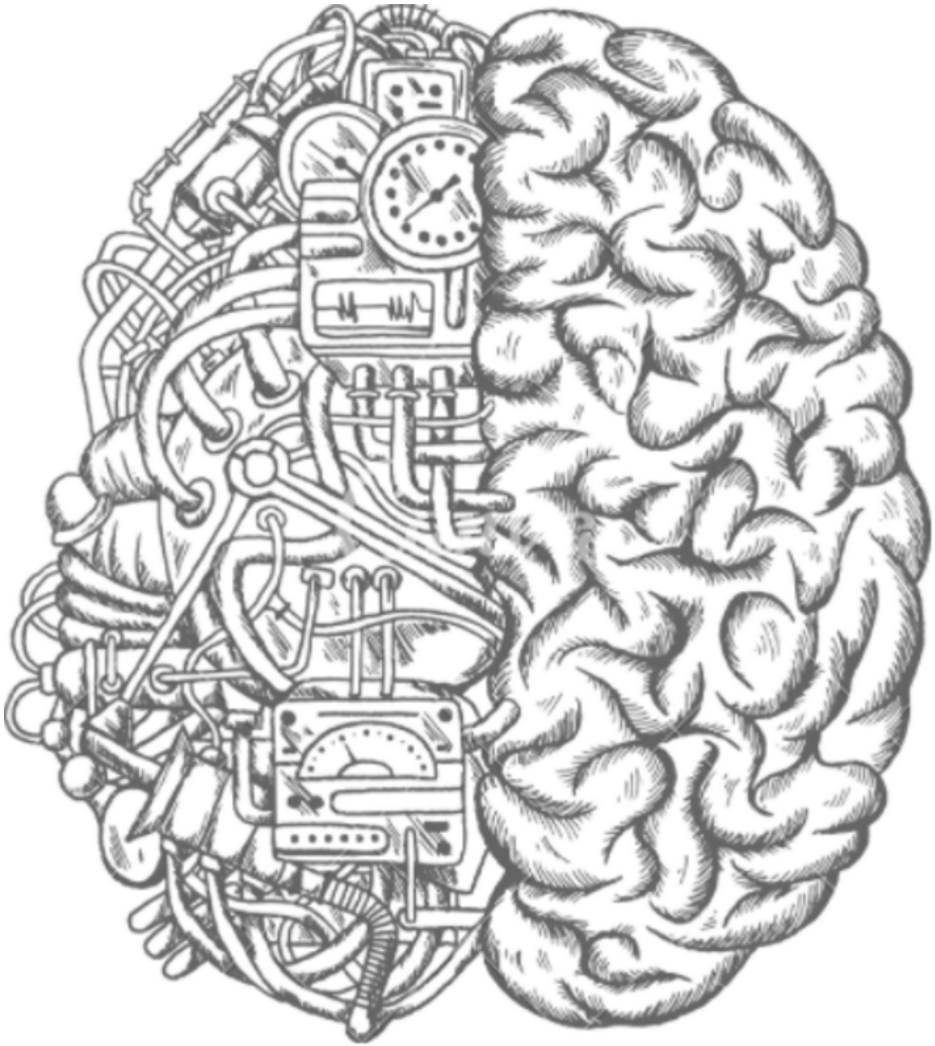


Figure: An artists interpretation

Neural Networks

- Neural networks are nature inspired computational methods that are loosely modelled after biological neural systems. A neural network consists of a group of interconnected units called *neurons* that have adaptive properties that provide learning.
- A *neuron* is a single processing unit that computes the weighted sum of its inputs. The interconnections between neurons are called *synapses* or *weights*. Neural networks learn by training on data using an algorithm that modifies the interconnection weights as directed by a learning objective. Prominent neural networks architectures are characterized into *feedforward* and *recurrent* networks.

Synapses in biological neural network

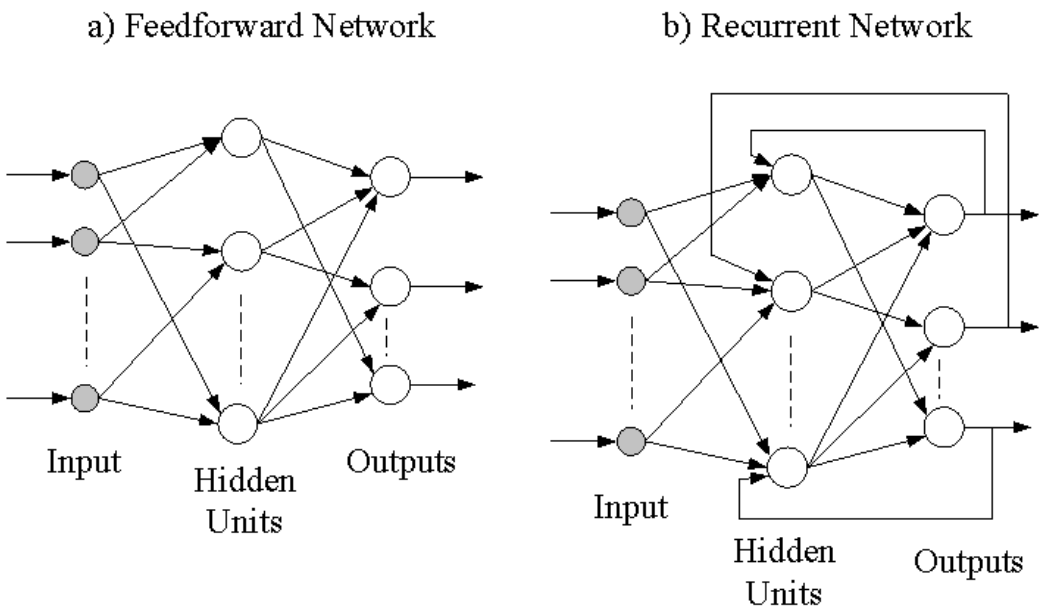
Figure: Synapses that motivate weight connections (weights)

Feedforward Neural Networks

The weight w_{ij} is defined from the input signal x_j to neuron i. The linear combiner computes the *weighted sum* of input signals. The dynamics of a feedforward network is described by Equation (1), where the total net input activation value y_i of neuron i is given for N input connections.

$$y_i = \sum_{j=1}^{N} (w_{ij} x_j) + \theta_i \tag{1}$$

where θ_i represents the bias.



Feedforward Neural Networks

The transfer function $f(y_i)$ computes the output o_i of the unit. Some common transfer functions are threshold, sigmoid, hyperbolic tangent and piece-wise linear functions. The sigmoid transfer function is given in Equation (2).

$$f(y_i) = \frac{1}{1 + e^{-y_i}} \tag{2}$$

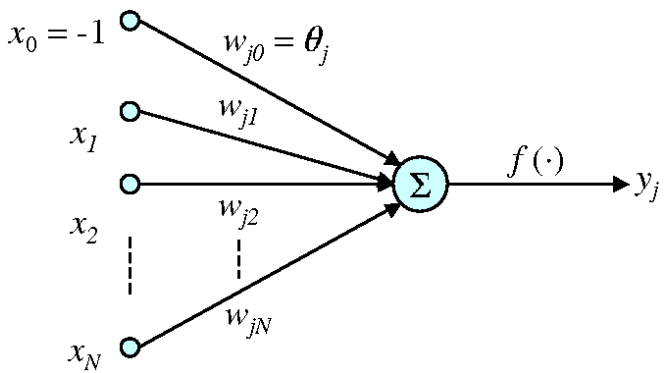


Figure: Neuron

Training (Learning) Neural Networks

- The inputs x_j and weights w_{ij} with $j = 1, \ldots, N$ can be represented as the vectors $\mathbf{x} = [x_0, x_1, \ldots, x_N]^t$ and $\mathbf{w}_i = [w_{i0}, w_{i1}, \ldots, w_{iN}]^t$, respectively. The activation value for the ith neuron can be written as $y_i = \mathbf{w}_i \cdot \mathbf{x}$ and thus its output is $o_i = f(\mathbf{w}_i \cdot \mathbf{x})$.
- Rumulhart *et al.* introduced the *backpropagation* algorithm which employs gradient descent for training feedforward networks. Gradient descent in its purest form is a technique for function optimisation, neural network training can be seen as an optimisation problem.
- Backpropagation requires the transfer function that governs the neurons to be differentiable. The backpropagation algorithm adjusts the connection weights in the neural network in a two-phase process which consists of a *forward* and a *backward pass*.

Learning

- The backpropagation algorithm adjusts the connection weights in the neural network in a two-phase process which consists of a *forward* and a *backward pass*. In the forward pass, information is propagated from the input to the hidden and output layer, and the error of the network is calculated using a cost function.
- The cost function is usually the *sum-squared-error* which is calculated with the actual y_k and desired or target output t_k of the respective neurons in the output layer as given in Equation 3.

$$E = \frac{1}{2} \sum_{k=1}^{N} (y_k - t_k)^2 \tag{3}$$

where N is the number of neurons in the output layer and E is the sum-squared-error. The gradient $\frac{\delta E}{\delta w}$ of each weight w can be computed by propagating an error backwards through the network in a manner analogues to the way activations are propagated forward.

Application: Conservation of Endangered Species

- Feedforward neural networks was used for Renosterveld conservation in South Africa. Knowledge based neural networks was used for environmental management application.
- The approach takes into account local expert knowledge together with collected field data about plant habitats in order to identify areas which show potential for conserving thriving areas of Renosterveld vegetation and areas that are best suited for agriculture.
- The available data is limited and cannot be adequately explained by expert knowledge alone. The paradigm combines expert knowledge about the local conditions with the collected ground truth in a knowledge-based neural network. [1]

[1] R. Chandra, C. W. Omlin, Renosterveld Conservation in South Africa: A Case Study for Handling Uncertainty in Knowledge Based Neural Networks for Environmental Management.Journal of Environmental Informatics 13(1):56-65

Renosterveld

Expert knowledge

The expert knowledge about the presence of the Renos- terveld vegetation is summarized in the following rules:

- If the sand is shale or granite and the rainfall is between 300 to 700 mm, then the probability of occurrence of Renosterveld is high.
- If the sand is of Quartzitic type and the rainfall is below 300 mm, and if the altitude is between 200 m to 400 m, and the slope is 0 to 10 degrees, then the chances of the presence of Renosterveld is high.
- If the sand is of Quartzitic type and the rainfall is be- tween 300 mm to 700 mm, and if the altitude is between 200 m to 400 m, and the slope is 0 to 10 degrees and faces a sou- therly direction, then the chances of occurrences is high

Knowledge-based neural networks

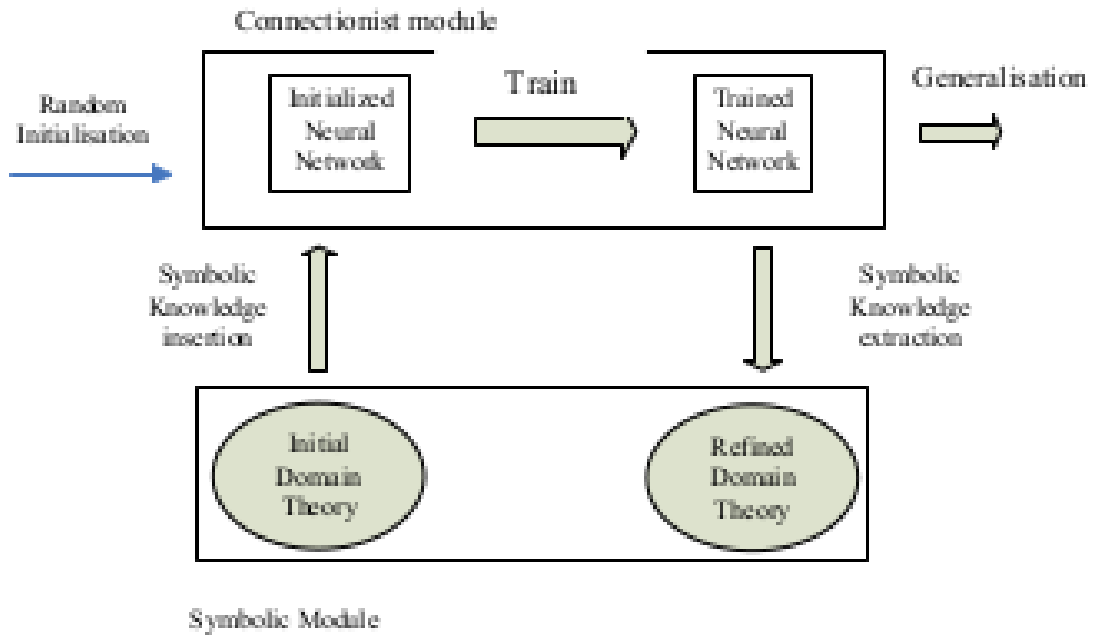


Figure 3. Knowledge-based neurocomputing paradigm.

Knowledge-based neural networks

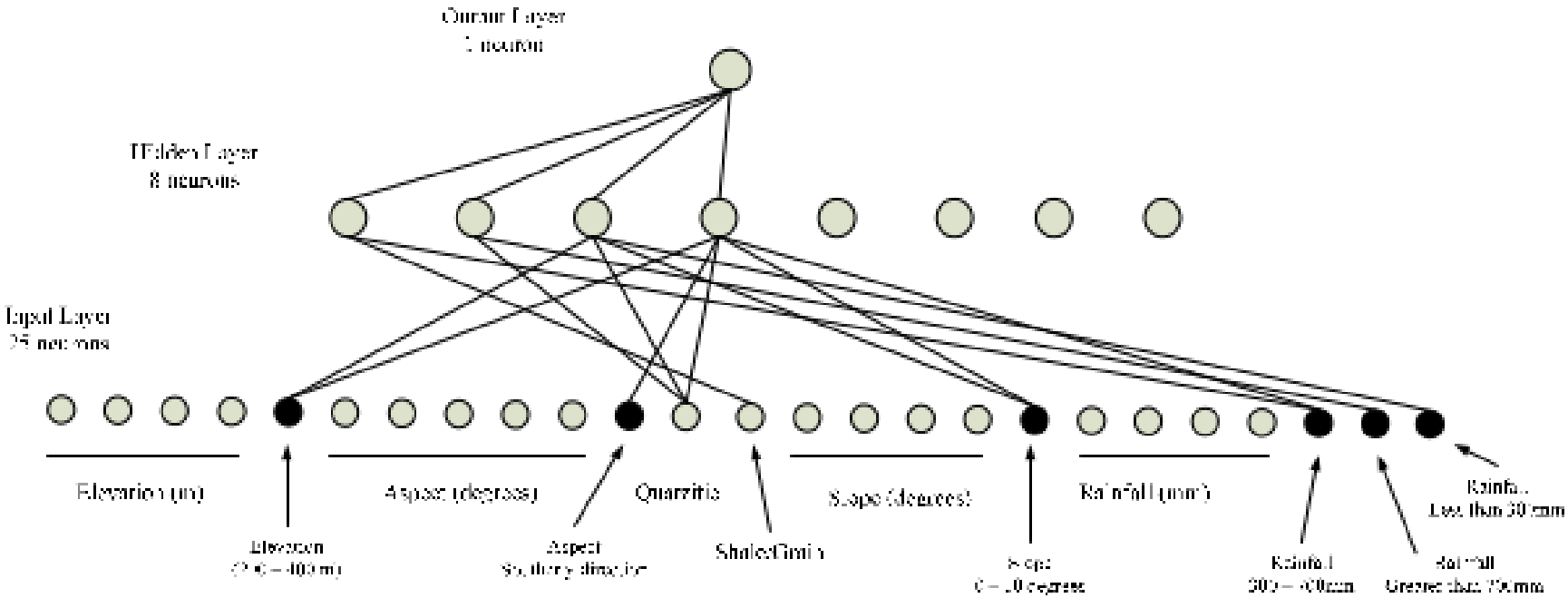


Figure 4. Knowledge insertion.

Neural Networks for Time Series Prediction

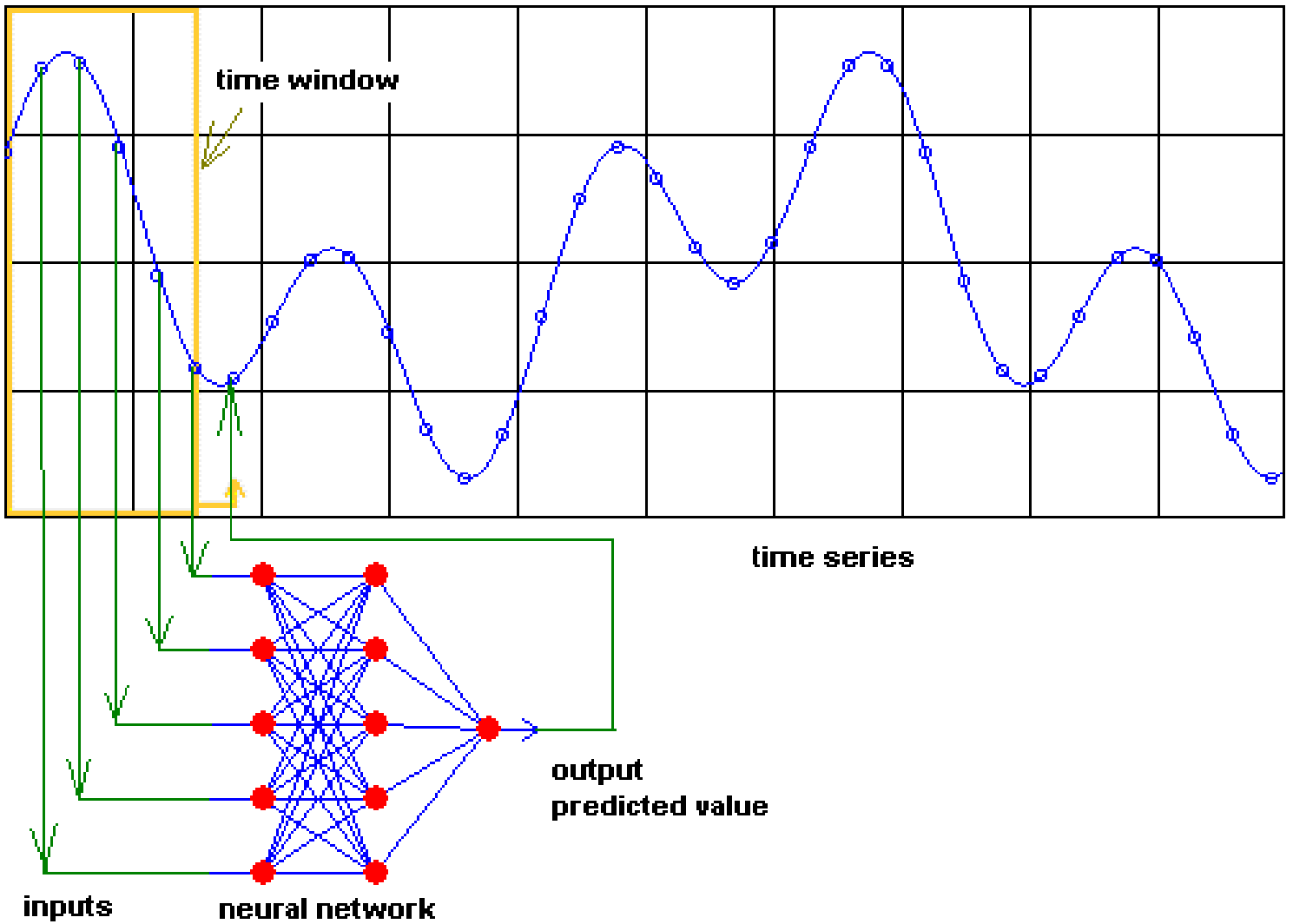


Figure: Time Series Prediction (Forecasting)

Deep Learning

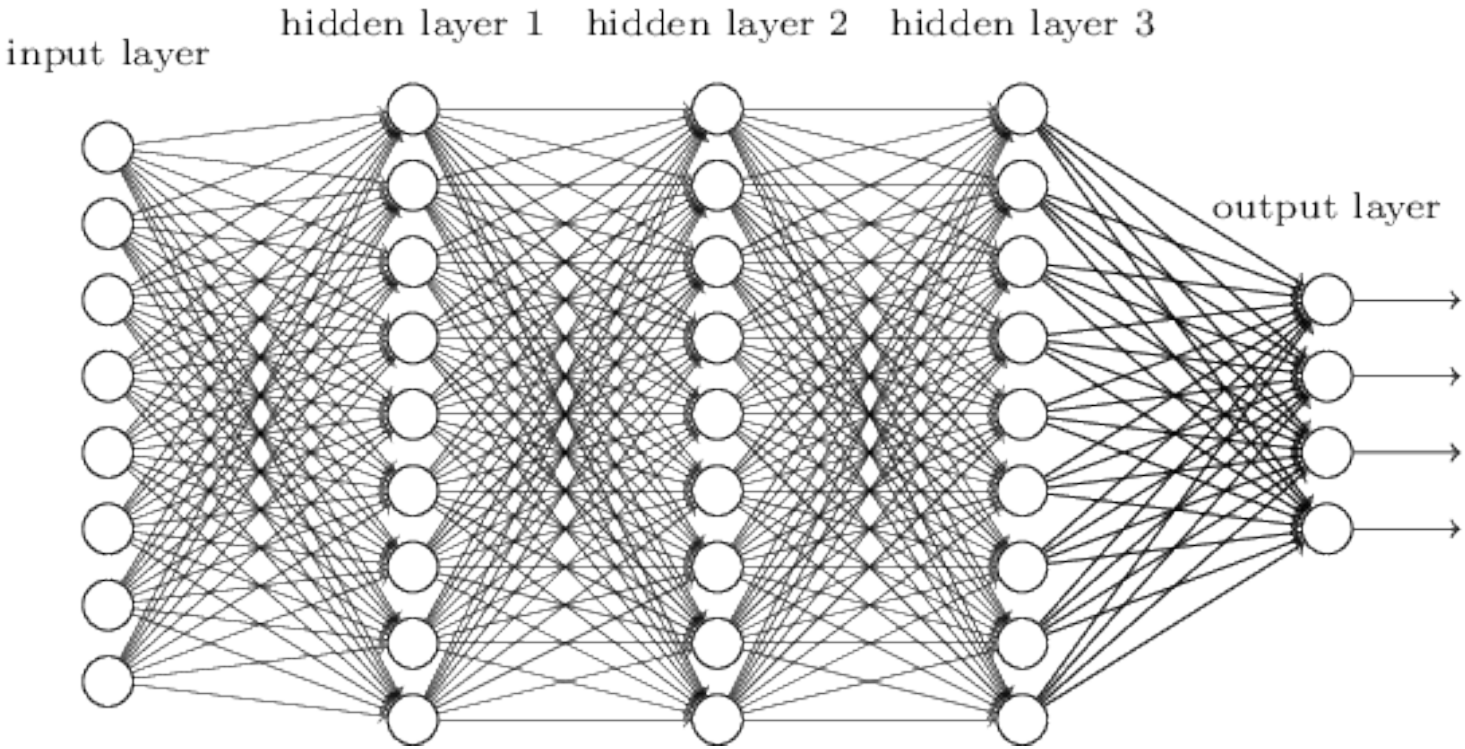


Figure: CNNs

Deep Learning: Convolutional Neural Networks

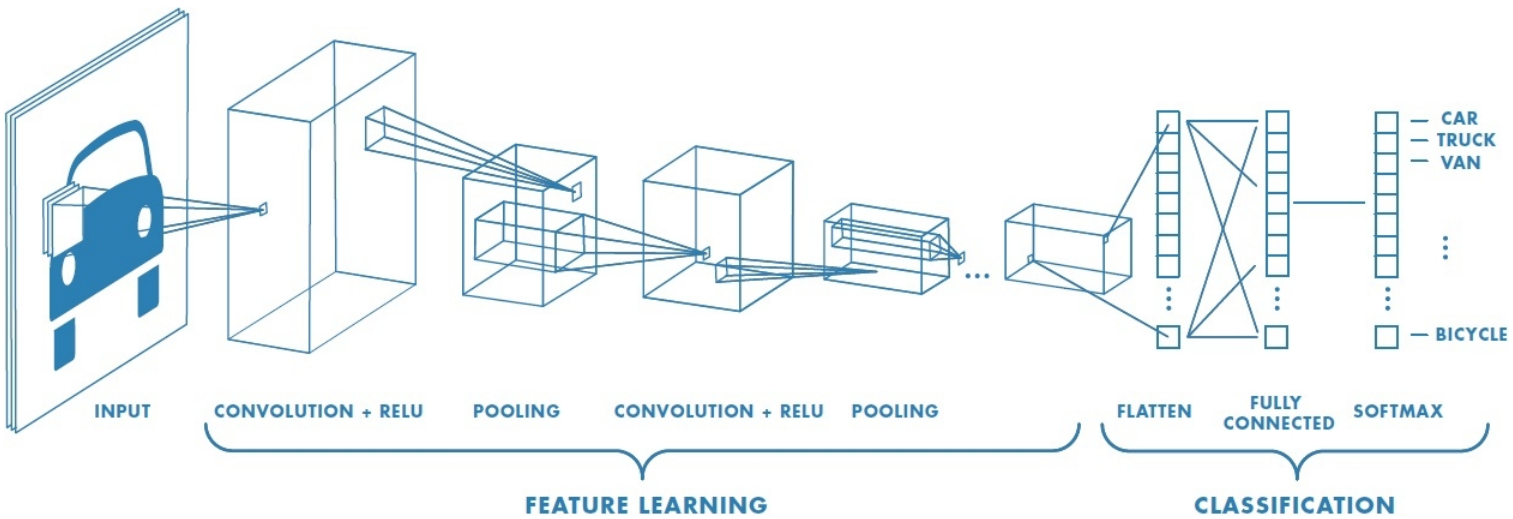


Figure: CNNs

Deep Learning: Convolutional Neural Networks

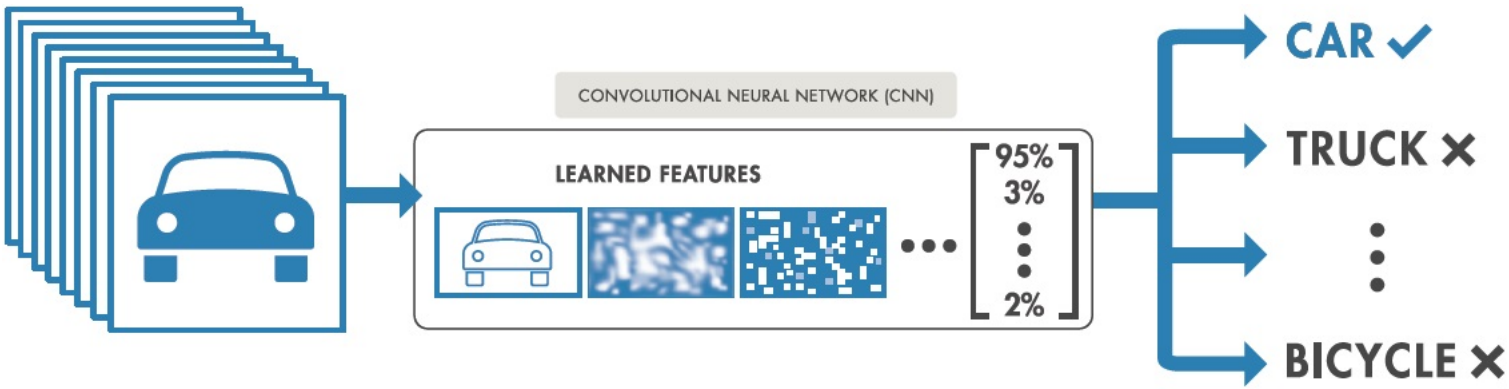


Figure: CNNs

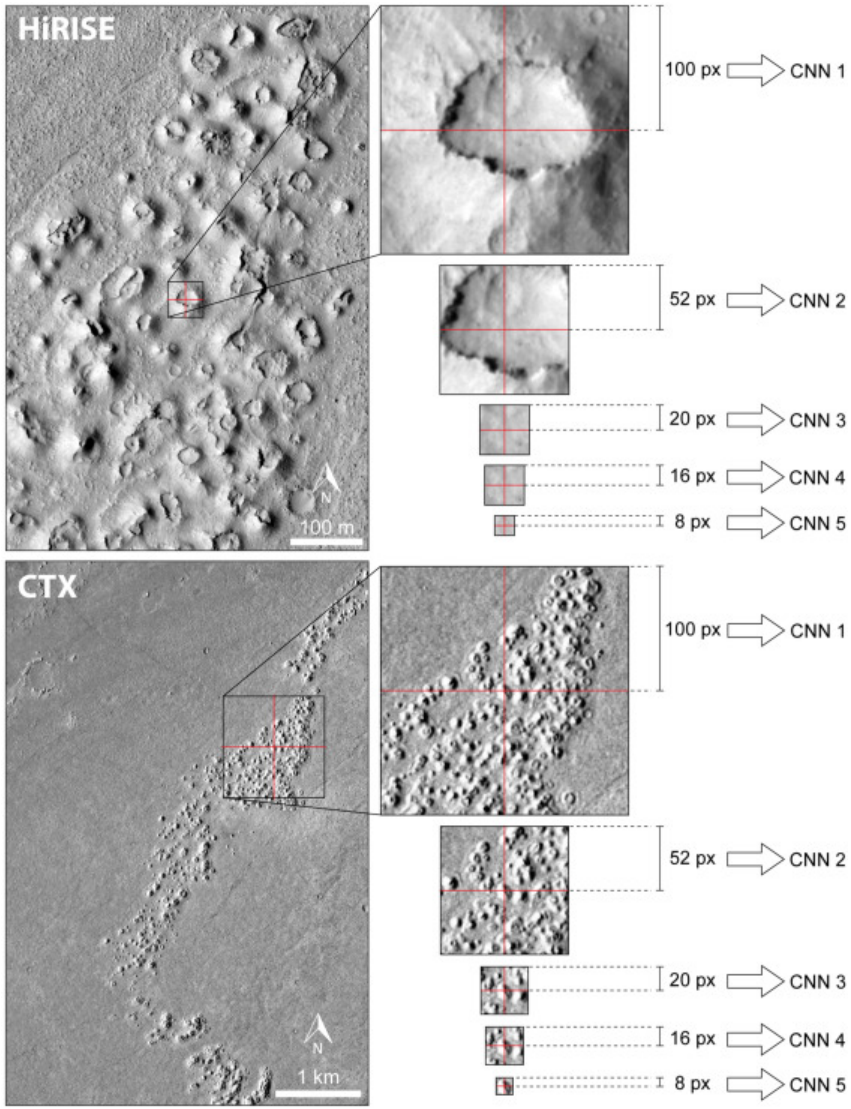
CNN Applications

- Computer vision application, speech recognition applications, autonomous driving systems, face, gesture and object recognition and others [2] .
- Automated detection of geological landforms on Mars. Detect both volcanic rootless cones and transverse aeolian ridges. The system consists of five networks, each of which is trained to detect landforms of different sizes. [3]

[2] Wikipedia page on CNNs

[3] Leon et. al, Automated detection of geological landforms on Mars using Convolutional Neural Networks, Computers and Geosciences, Volume 101, April 2017, Pages 48-56

Mars application



Recurrent Neural Networks

- Recurrent neural networks (RNNs) use context units to store the output of the state neurons from computation of the previous time steps. The applications include Speech recognition, gesture recognition, financial prediction, signature verification and robotics control
- They are considered dynamical systems whose next state and output depend on the present network state and input. The change of hidden (state) neuron activation's in a Elman RNN is given by Equation (4).

$$y_i(t) = f\left(\sum_{k=1}^{K} v_{ik}\ y_k(t-1) + \sum_{j=1}^{J} w_{ij}\ x_j(t-1)\right) \tag{4}$$

where, $y_k(t)$ and $x_j(t)$ represent the output of the context state neuron and input neurons, respectively. v_{ik} and w_{ij} represent their corresponding weights; i represents the number of input neurons while j and k represents the number of hidden and context layer neurons, respectively. $f(.)$ is a sigmoid transfer function.

Unfolded Recurrent Neural Network

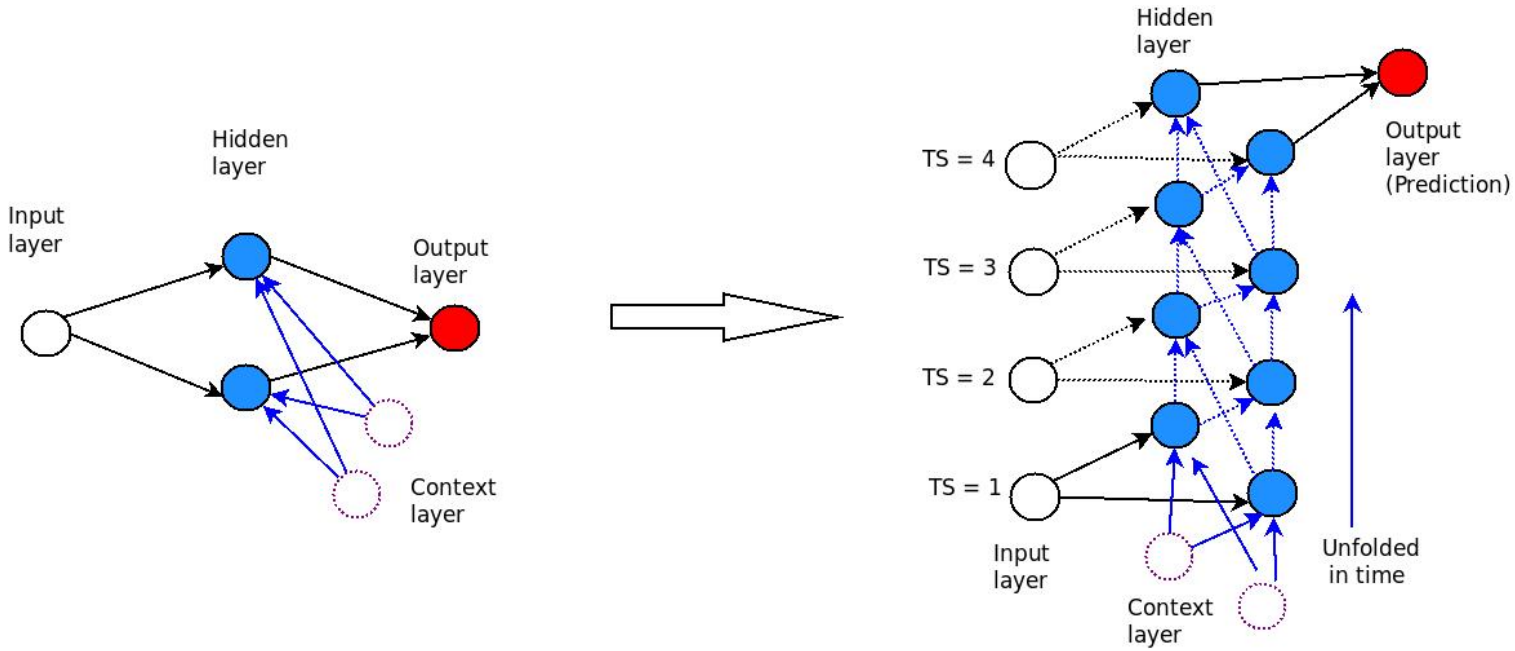


Figure: Elman RNN having 2 hidden neurons unfolded for 4 time steps

RNN Training Algorithms

- Backpropgation-through-time: Employs gradient descent based algorithms. Stocastic gradient descent is used in cases when dataset is large. Challenges include local convergence and vanishing error problem for problems with long-term dependencies. Long-Short-Term-Memory (LSTM) networks were first used to address long term dependencies. They have become popular recently in the deep learning revolution!
- Neuro-evolution employs evolutionary algorithms which are known as global optimisation methods. They are easily deployed in any neural network training problem, without being constrained to a particular network architecture. The limitations and challenges lie in longer optimisation time which are not appropriate for big-data related problems.

Training Algorithms: Backpropagation-Through-Time

Algorithm : Backpropagation Through-Time for Training Elman RNNs

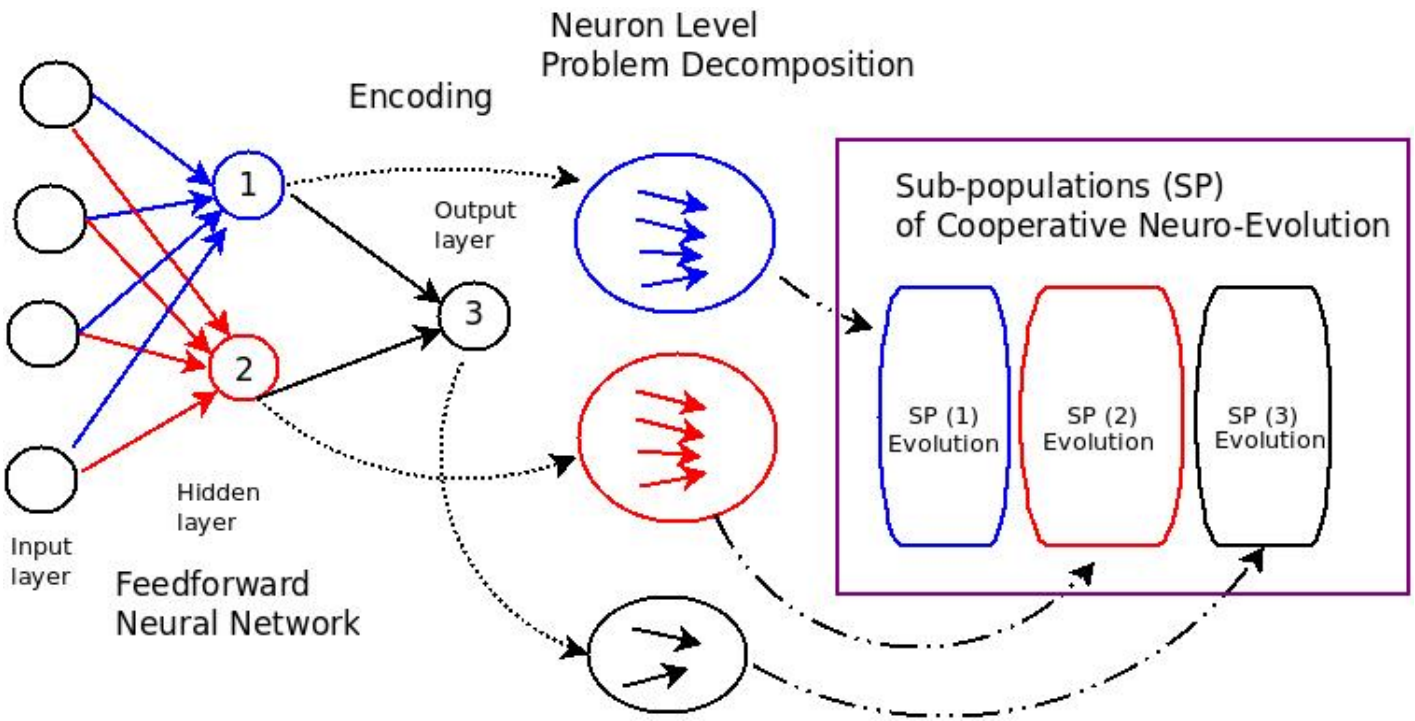
Step 1: Prepare Training and Testing dataset
Step 3: Initialize the RNN weights with small random numbers in range [-0.5, 0.5]
for each *Epoch* until termination **do**
  for each Sample **do**
    for n Time-Steps **do**
      Forward Propagate
    end for
    for n Time-Steps **do**
      i) Backpropagate Errors using Gradient Descent
      ii) Weight update
    end for
  end for
end for

Neuro-evolution

- Cooperative coevolution (CC) is an evolutionary computation method that decomposes a problem into subcomponents and employs standard evolutionary algorithms in order to gradually solve the bigger problem. The subcomponents are also known as species or modules and are represented as subpopulations.
- The subpopulations are evolved separately and the co-operation only takes place for fitness evaluation for the respective individuals in each subpopulation.
- Applications: training feedforward neural networks, recurrent neural networks. Cooperative Coevolution methods have shown promising performance in time series prediction problems [4].

[4] R. Chandra: Competition and Collaboration in Cooperative Coevolution of Elman Recurrent Neural Networks for Time-Series Prediction. IEEE Trans. Neural Netw. Learning Syst. 26(12): 3123-3136 (2015)

Problem decomposition and training via cooperative coevolution



Training Algorithm: Cooperative Coevolution

Algorithm : Cooperative Coevolutionary Training of Elman Recurrent Networks

Step 1: Decompose the problem into k subcomponents according to the number of Hidden, State, and Output neurons

Step 2: Encode each subcomponent in a sub-population in the following order:

i) Hidden layer sub-populations

ii) State (recurrent) neuron sub-populations

iii) Output layer sub-populations

Step 3: Initialize and cooperatively evaluate each sub-population

for each *cycle* until termination **do**

  for each Sub-population **do**

    for n Generations **do**

      i) Select and create new offspring

      ii) Cooperatively evaluate the new offspring

      iii) Add the new offspring to the sub-population

    end for

  end for

end for

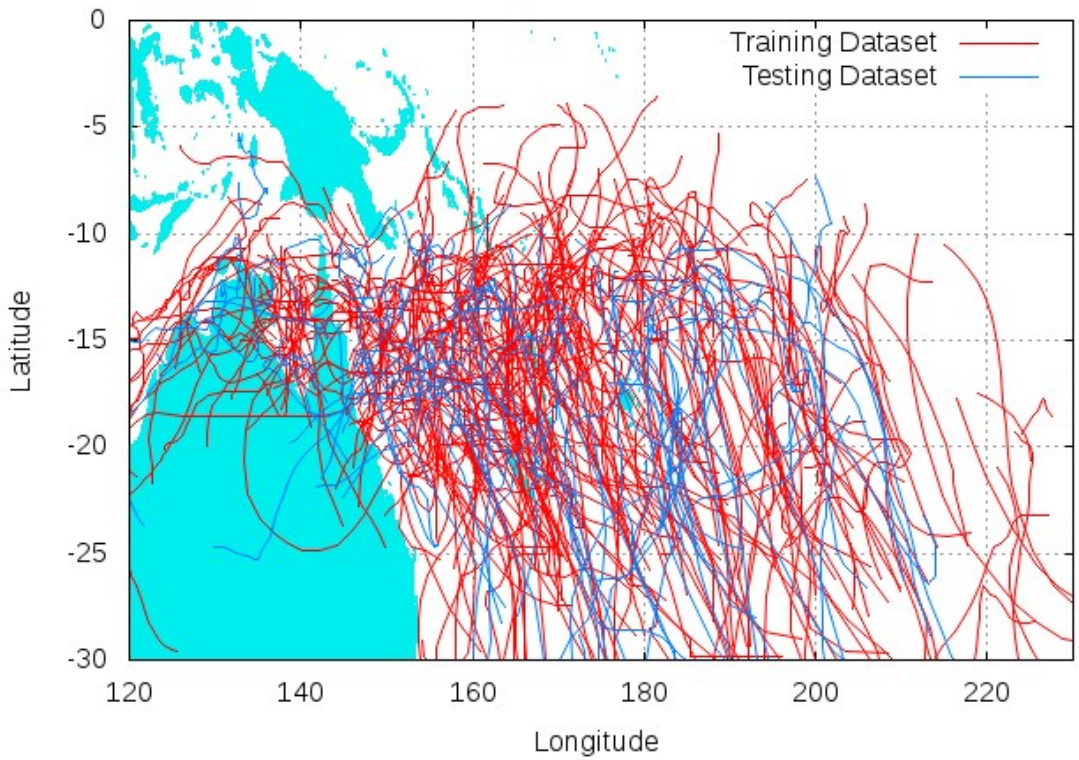
Neural networks for climate extremes: cyclones

Encoding cyclone track prediction problem in coevolutionary RNNs

- Cyclone track prediction is a two dimensional time series prediction problem that involves latitudes and longitudes which define the position of a cyclone.
- Coevolutionary recurrent neural networks have been used for time series prediction and also applied to cyclone track prediction.
- We present an architecture for encoding two dimensional time series problem into Elman recurrent neural networks composed of a single input neuron. We use cooperative coevolution and back-propagation through-time algorithms for training.

Tropical cyclones in the South Pacific ocean

Data used contained 6000 points in the training set (Tropical Cyclones from 1985 - 2005). There were 2000 points in the test set (Tropical Cyclones from 2006 - 2013) taken from JTWC data set.



Data Preprocessing and Reconstruction

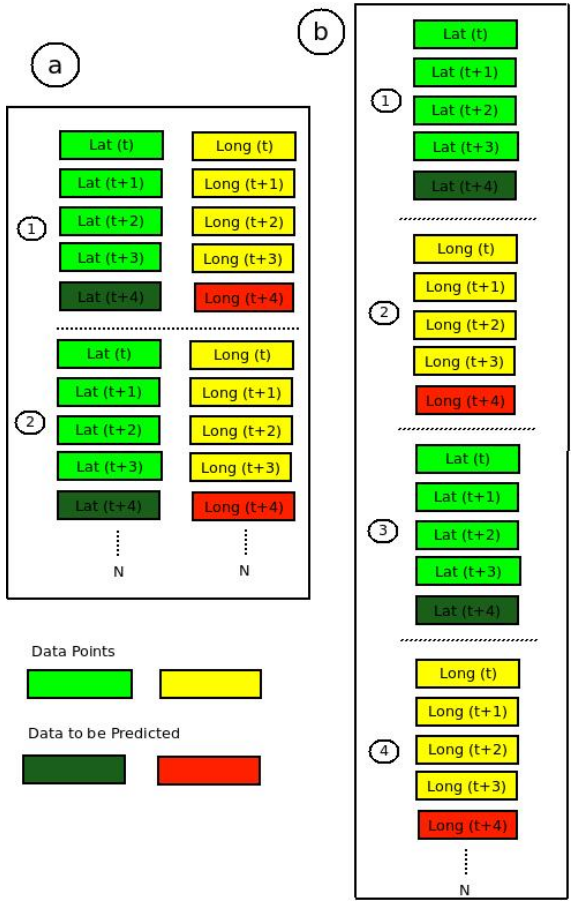


Figure: cyclone track data embedded and reconstructed using Taken's theorem.

Application to Cyclone Track Prediction

The main idea is to have a RNN model that combines the two dimensions of latitude and longitude into a single data stream variable in an attempt to represent the direct relationship between the dimensions.

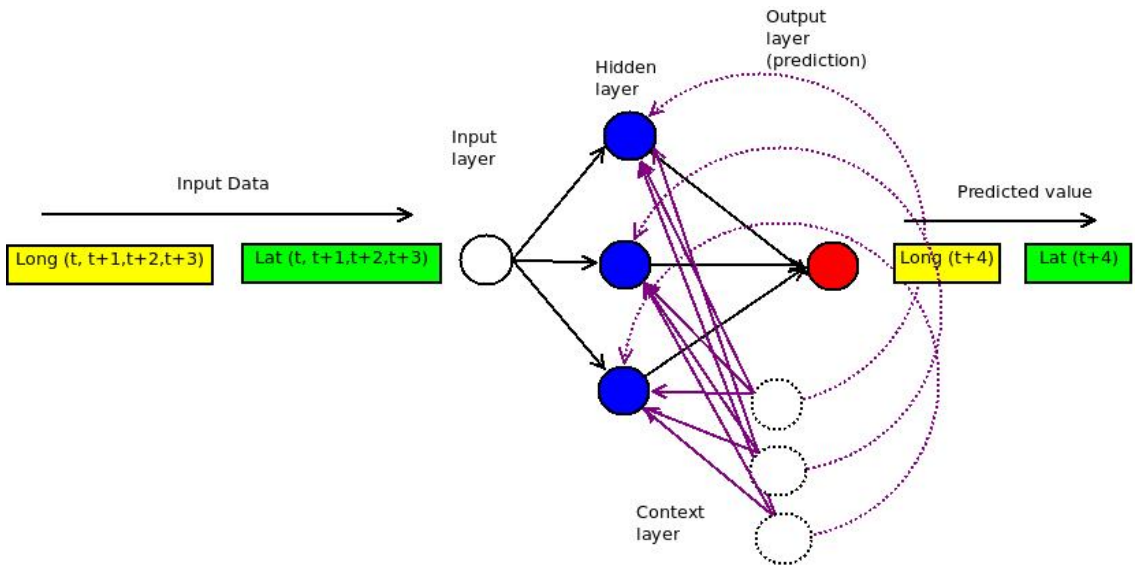


Figure: RNN for track prediction

Experimental Setup

- We experimented with different number of hidden neurons in the RNN that employed sigmoid units in the hidden and output layer.
- The CC-SIORNN as well as BPTT-SIORNN were both trained for and their predictions tested for 24-hour and 30-hour advance warning.
- The termination condition was set at 50,000 function evaluations for CC and 2000 epochs for BPTT. The following configuration was used: $D = 4$ and $T = 2$, reconstructed dataset contains 3417 samples in training set and 1298 samples in test set.
- The root mean squared error (RMSE) was used to evaluate the performance of the two architectures for cyclone track prediction. y_i and $\hat{y}_i$ are the observed and predicted data, respectively. N is the length of the observed data. $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$

Results

Table: Results

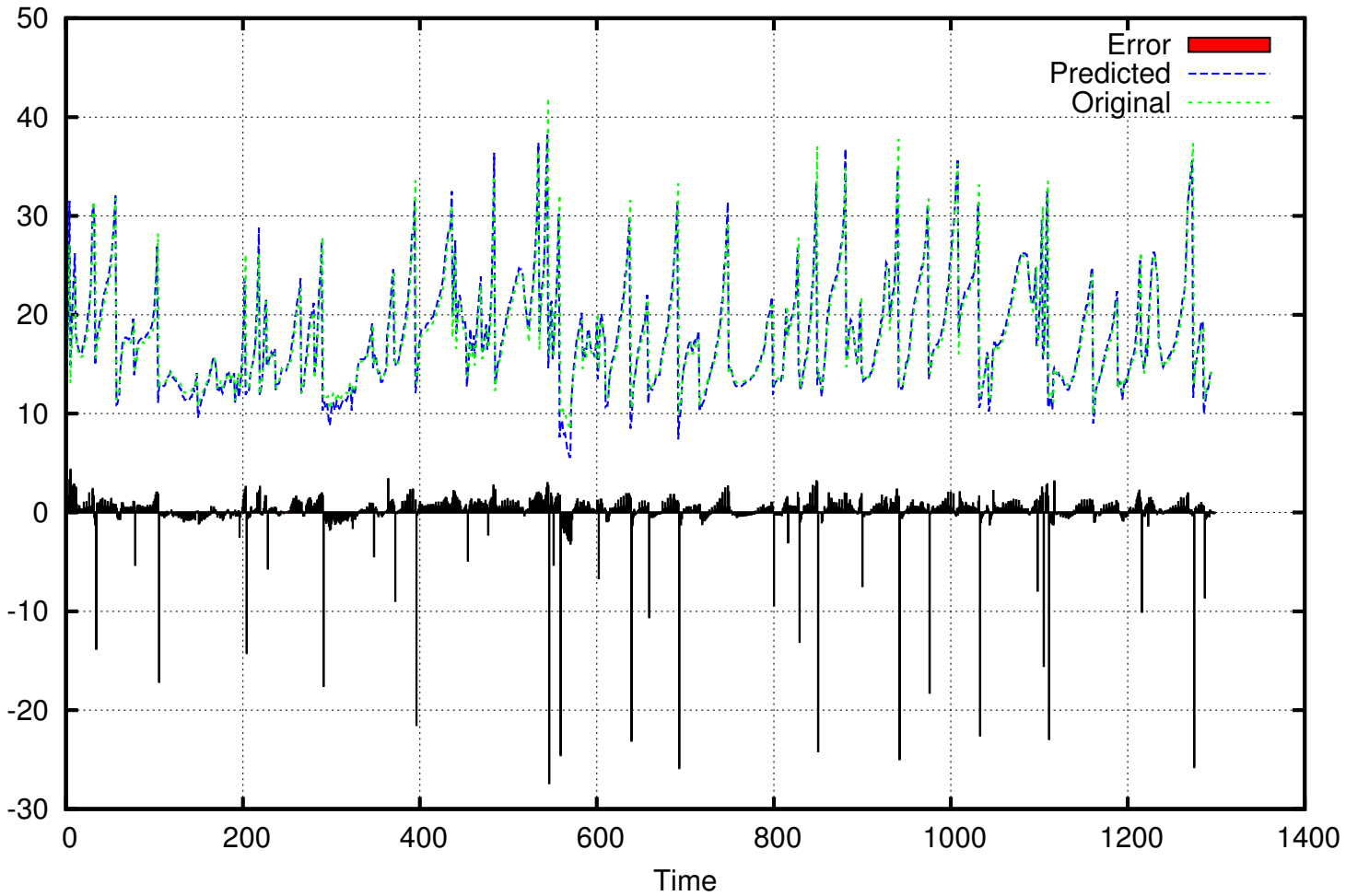
Model	Hidden	RMSE (Train)	RMSE (Test)	Best
CCRNN	3	0.0508 ± 0.0010	0.0484 ± 0.0010	0.0455
CCRNN	5	0.0493 ± 0.0006	0.0471 ± 0.0006	0.0447
CCRNN	7	0.0492 ± 0.0007	**0.0471± 0.0006**	**0.0448**
CC-SIORNN	3	0.0252 ± 0.0003	**0.0244 ± 0.0002**	0.0238
CC-SIORNN	5	0.0252 ± 0.0003	0.0245 ± 0.0003	0.0238
CC-SIORNN	7	0.0266 ± 0.0033	0.0260 ± 0.0034	**0.0237**
BPTT-SIORNN	3	0.0265 ± 0.0002	0.0257 ± 0.0002	0.0245
BPTT-SIORNN	5	0.0260 ± 0.0003	0.0254 ± 0.0002	0.0245
BPTT-SIORNN	7	0.0256 ± 0.0003	**0.0251 ± 0.0003**	**0.0242**

Reference [5]

[5] R. Chandra, R. Deo, C. W. Omlin: An architecture for encoding two-dimensional cyclone track prediction problem in coevolutionary recurrent neural networks. IJCNN 2016: 4865-4872

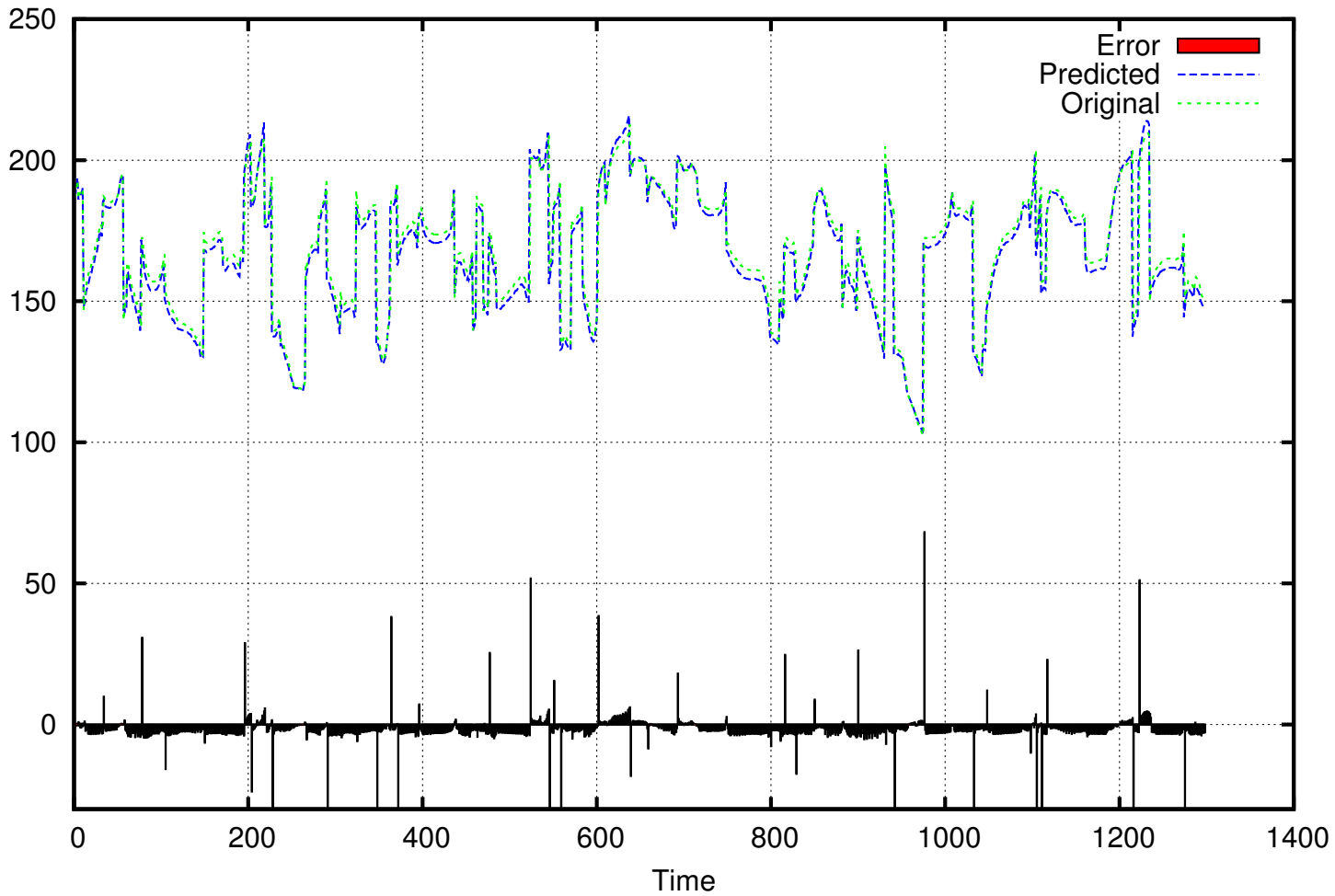
Results - cont

Lattitude Prediction



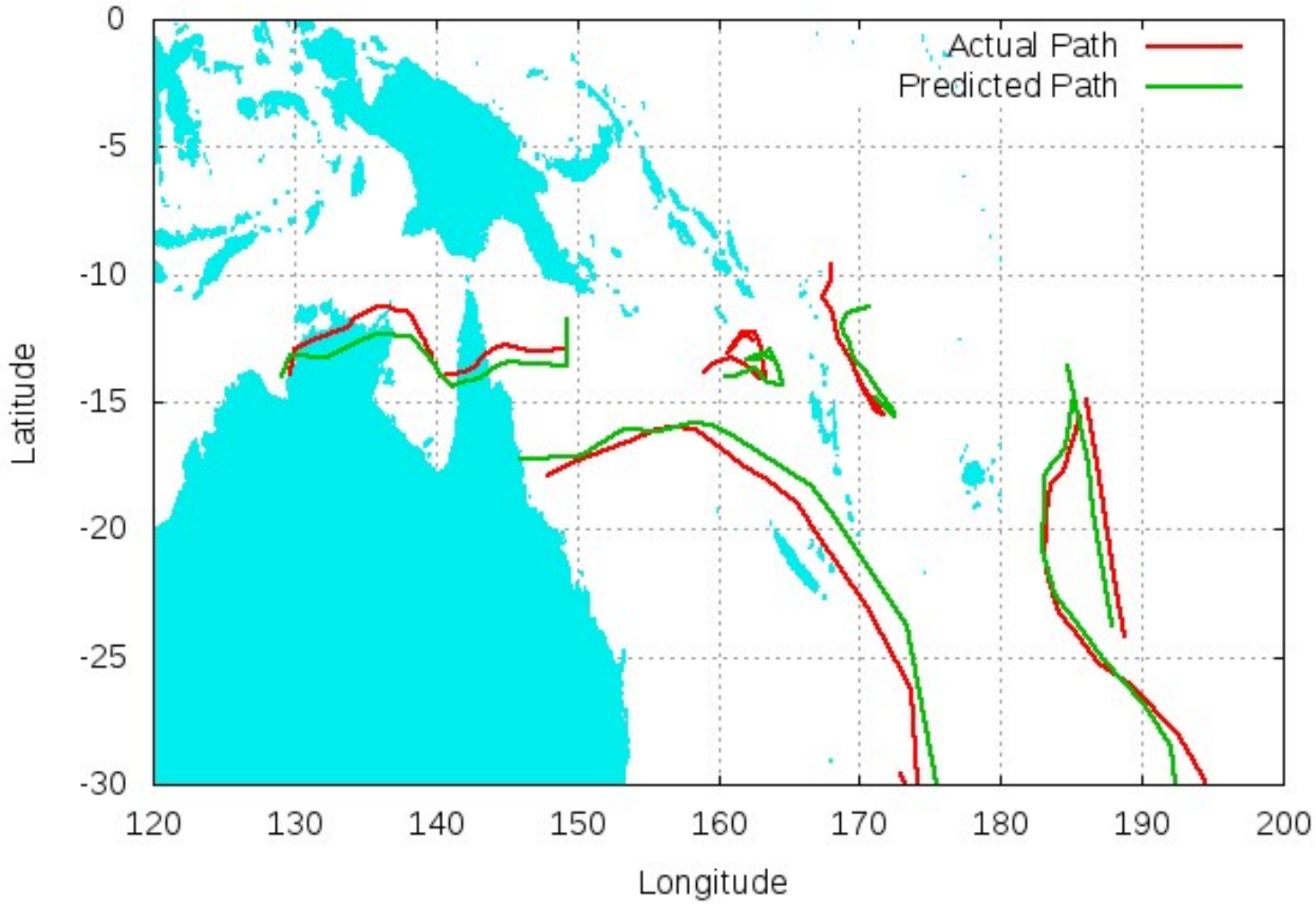
Results - cont

Longitude Prediction



Results - cont

Track Prediction for 5 cyclones



Discussion

- In the first method, the latitude and longitude is presented to a recurrent neural network with two input neurons whereas the second method combines both variables in a single data stream and employs a network with single input neuron which interleaves successive longitude and latitude values.
- The latitude and longitude encoded into a recurrent neural network as a single stream of data the prediction performance improves significantly regardless of the training algorithm.
- The results show that a single input and output layer neuron network trained with either of the algorithms outperforms networks trained with separate inputs for longitude and latitude.
- There is motivation for using additional atmospheric conditions that are major attributes in the formation of cyclones such as the sea surface temperature, pressure and humidity and the change of their intensity with time.

Minimal timespan problem

- The span of past data points is important for some applications since prediction will not be possible unless the minimal timespan of the data points is available.
- This is a problem for cyclone wind-intensity prediction, where prediction needs to be made as a cyclone is identified.
- An empirical study is given on minimal timespan required for robust prediction using Elman recurrent neural networks. Two different training methods are evaluated for training Elman recurrent network that includes cooperative coevolution and backpropagation though time (BPTT) [6].

[6] R. Deo, R. Chandra: Identification of minimal timespan problem for recurrent neural networks with application to cyclone wind-intensity prediction. IJCNN 2016: 489-496

Minimal timespan

- Performance of cooperative neuro-evolution (CNE) and BPTT in wind intensity prediction in the testing data set (2006 -2013) for tropical cyclones in the South Pacific. The RNN is training on Timespan of 5 and tested on the respective situations.

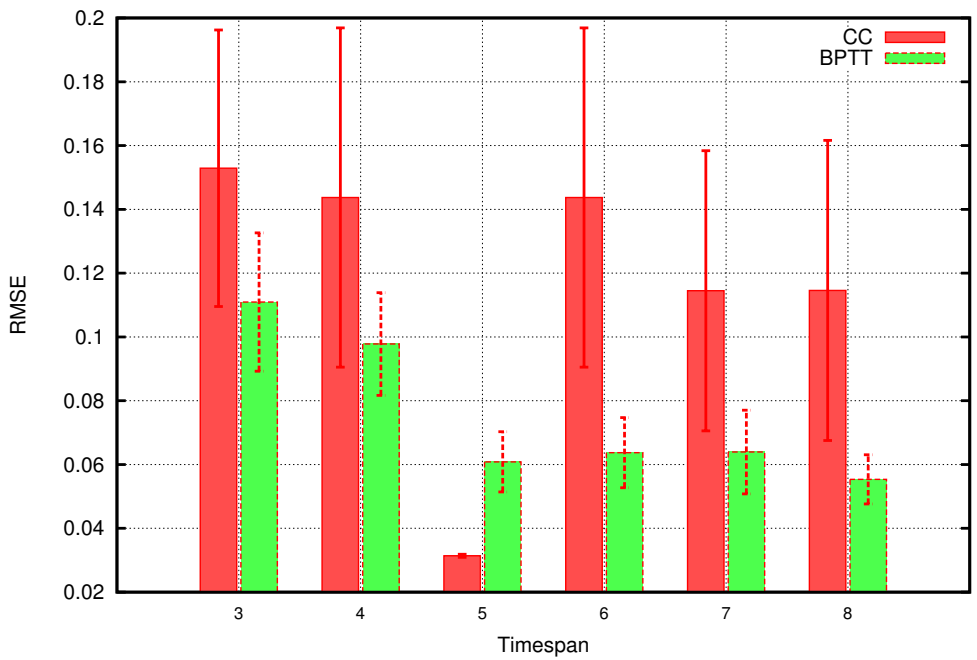


Figure: RNN trained on Timespan = 5

Rapid intensification in tropical cyclones

- The problem where a tropical cyclone intensifies dramatically within a short period of time is known as rapid intensification. This has been one of the major challenges for tropical weather forecasting.
- Recurrent neural networks are used to predict rapid intensification cases of tropical cyclones from the South Pacific and South Indian Ocean regions.
- In Strategy I, the distinction between positive and negative cases are made when rapid intensification is by 30 knots. This results in a highly unbalanced datasets which makes detection of true positives very difficult. Hence, Strategy II is used where the distinction between positive and negative cases is when rapid intensification is by 10 knots. In Strategy II, we achieve a poorer generalisation performance when compared to Strategy I, however, there is better detection of rapid intensification cases as shown by rate of true positives.

Rapid intensification cases over time

- Number of RI cases and duration of each cyclone over the cyclone identification number (ID). Each point of cyclone duration in y-axis represents 6 hours. It is observed that in several cases,
  the number of cases of RI does not directly relate to the duration of the cyclone.

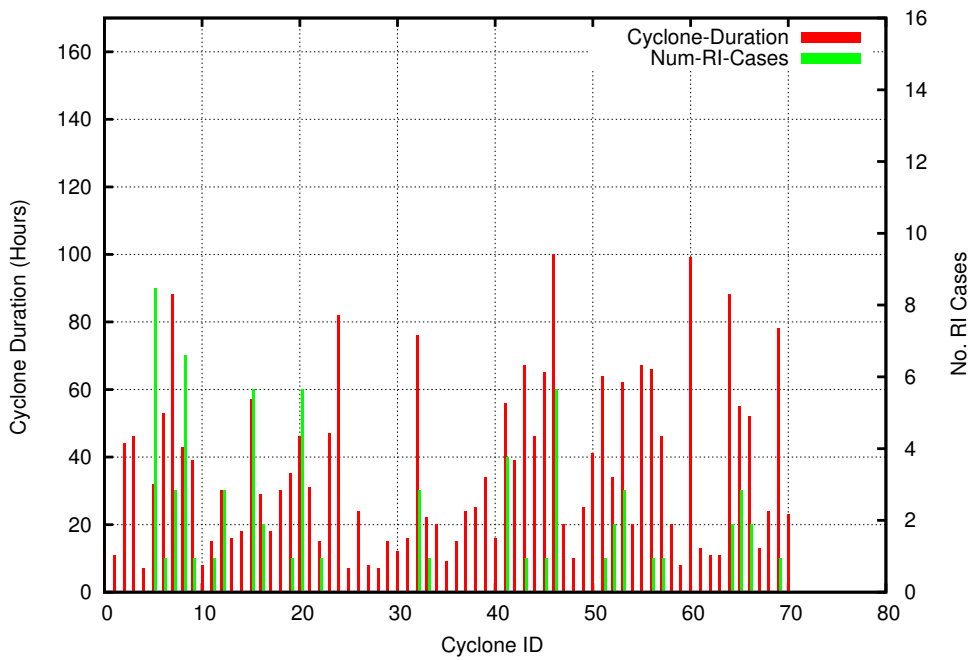


Figure: RI cases

Rapid intensification

Table: Detection of RI

Problem	Strategy	Percentage (Test)
South Pacific	I	97.214 ± 0.013
South Pacific	II	79.779 ± 0.169

Table: Strategy I Confusion Matrix for South Pacific

		Predicted		
		Positive	Negative	Total
Actual	Positive	0	7	7
	Negative	2	1999	2001
	Total	2	2006	2008

Table: Strategy II Confusion Matrix for South Pacific

		Predicted		
		Positive	Negative	Total
Actual	Positive	50	308	358
	Negative	66	1452	1518
	Total	116	1760	1876

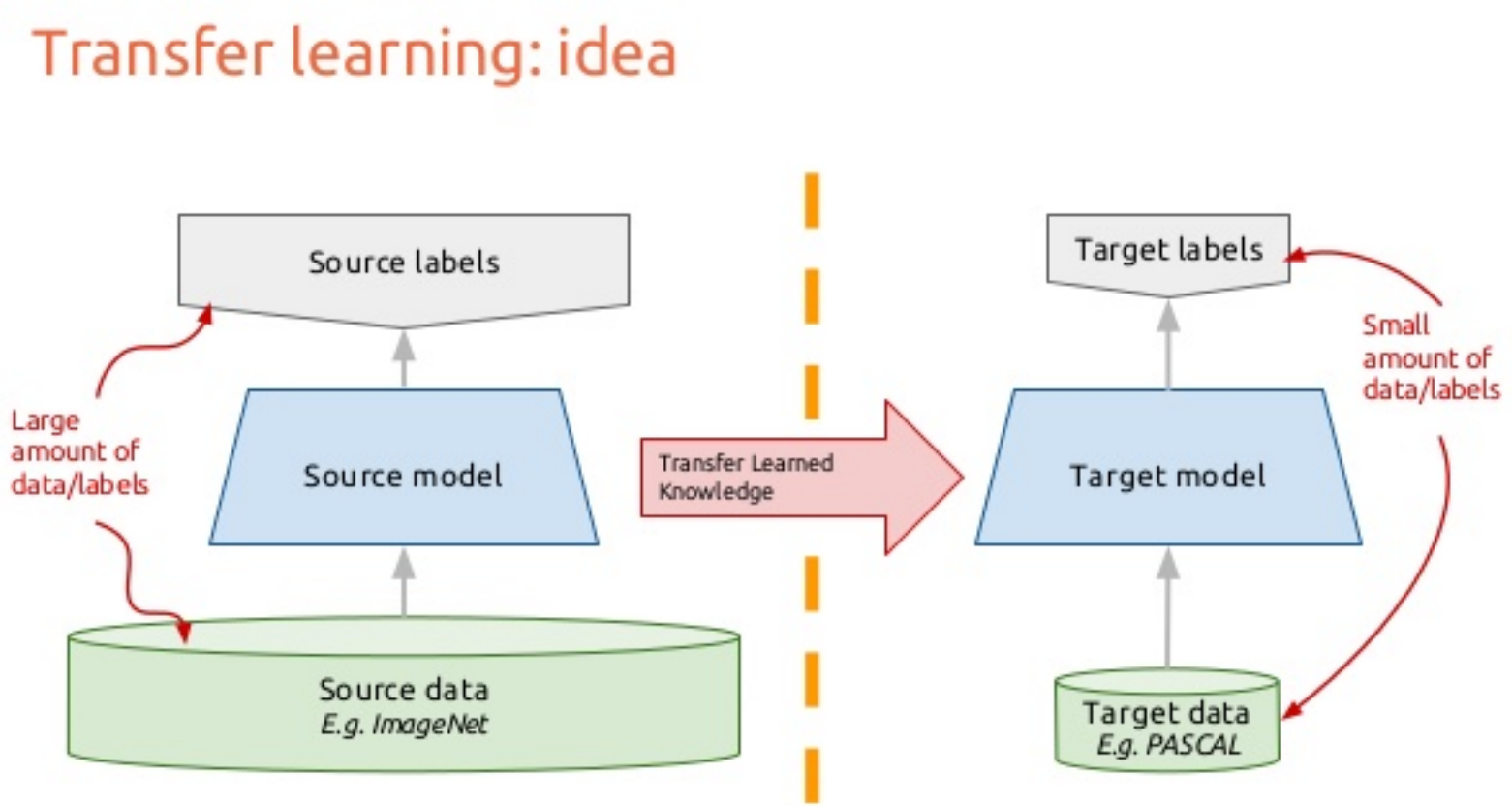
Preliminary results have been given in recent technical report [7]

[7] R. Chandra: Towards prediction of rapid intensification in tropical cyclones with recurrent neural networks. CoRR abs/1701.04518 (2017)

Transfer Learning (Single and Multi-Source)

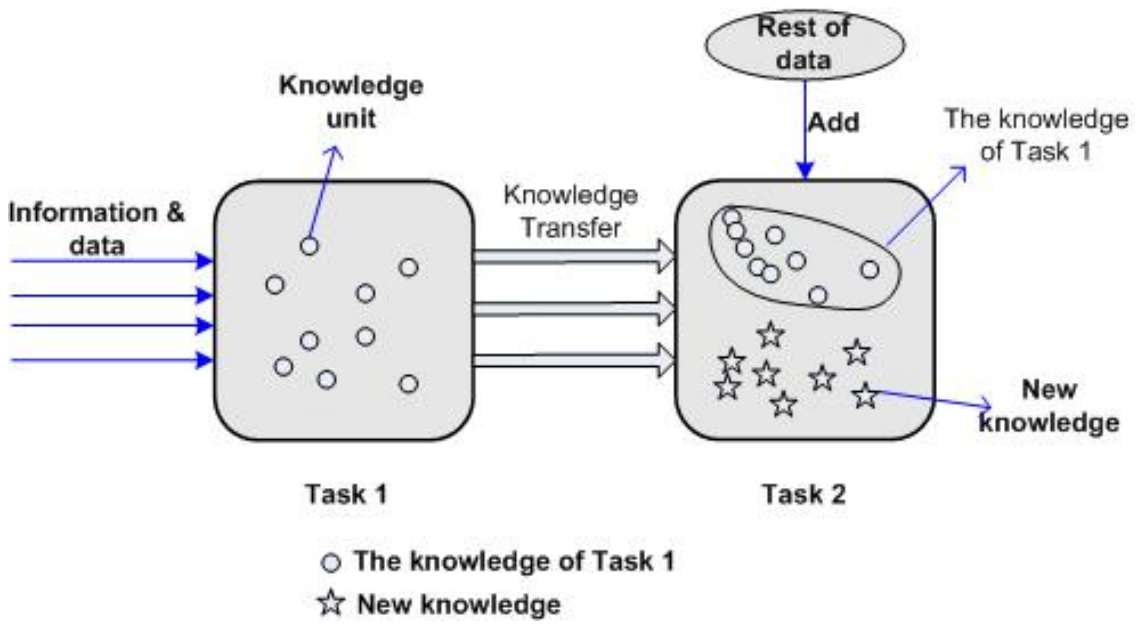
- Transfer learning focuses on utilizing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to drive tractors could be useful when learning to drive cars.

Transfer learning

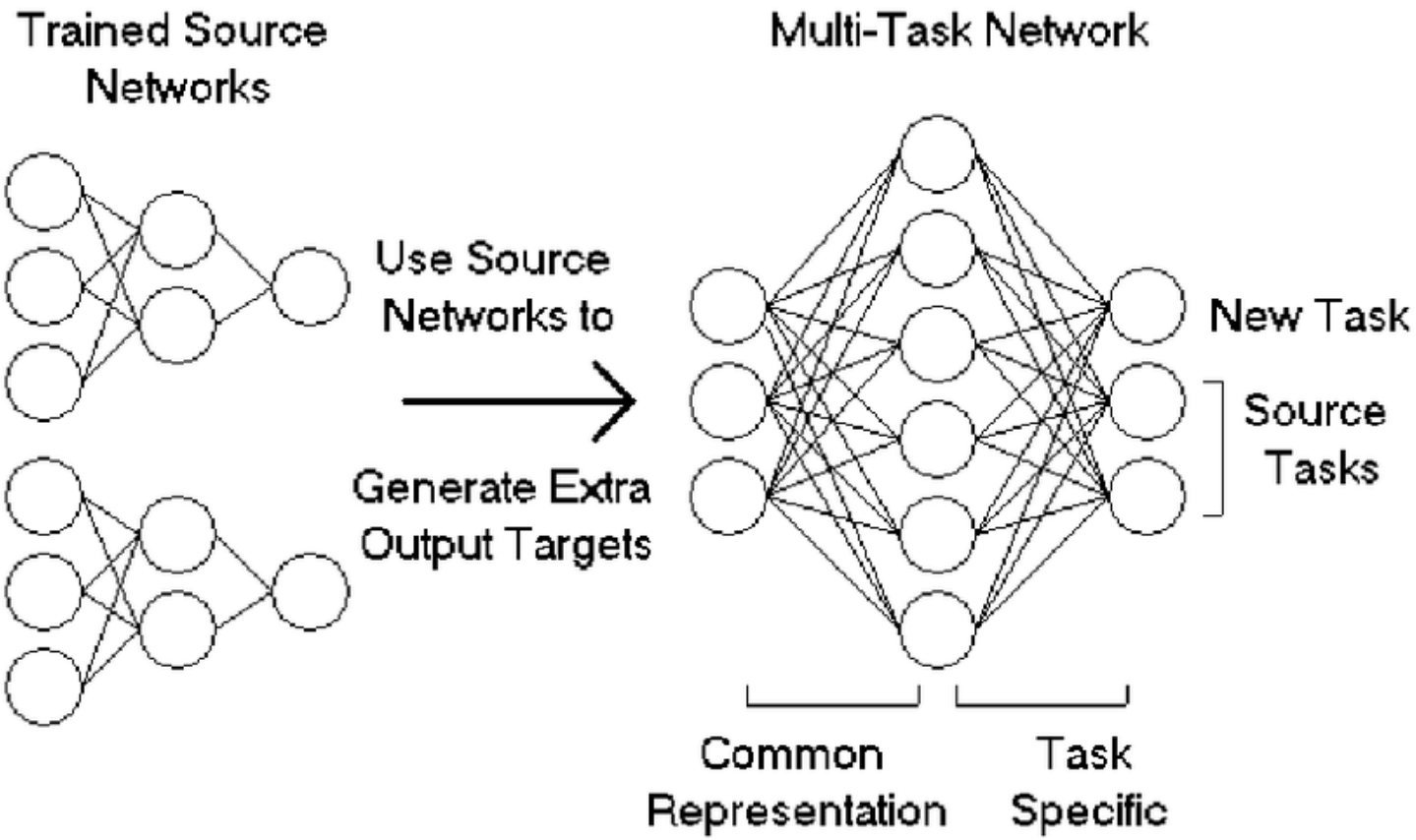


Multi-Task learning

- Multi-task learning implements shared knowledge representation where multiple learning tasks are solved at the same time, while exploiting commonalities and differences across tasks.
- The different tasks can be viewed as different data-sources, which can be hetrogenous (different number of features).



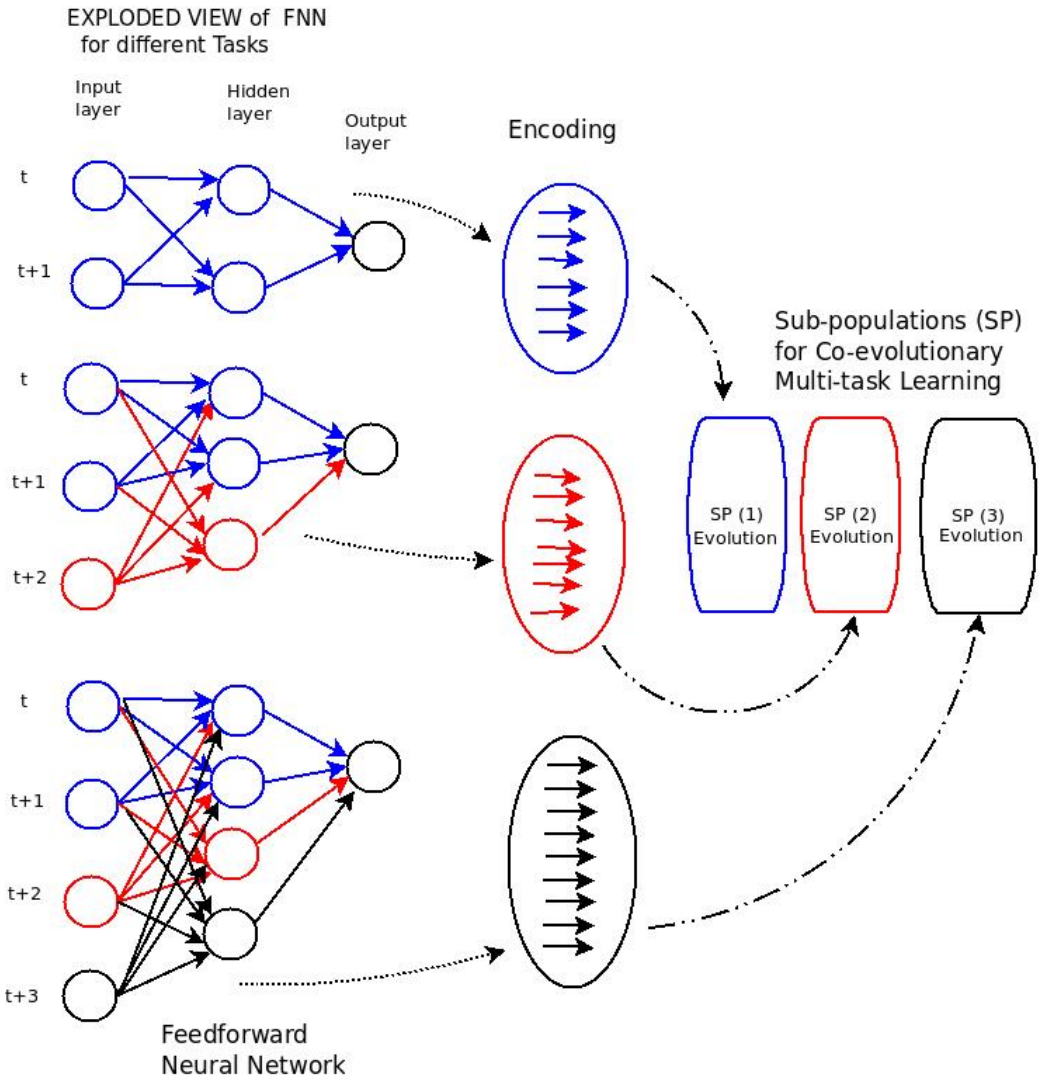
Multi-task learning with Neural networks



Co-evolutionary multi-task learning

- A co-evolutionary multi-task learning method is presented which incorporates notions from modular, multi-task and ensemble learning. The method is used for prediction of tropical cyclone wind intensity.
- This addresses the problem of the need for a robust and dynamic prediction model during the occurrence of a cyclone. The results show that the method addresses dynamic time series over conventional methods.
- The goal of the experiments was to evaluate if the proposed coevolutionary multi-task learning (CMTL) method can deliver similar results when compared to single task learning approaches (CNE and EA) for the introduced dynamic time series problems. Therefore, the comparison was to ensure that the approach does not lose quality in terms of generalisation performance when compared to single-task learning approaches .

Dynamic time series prediction



Results

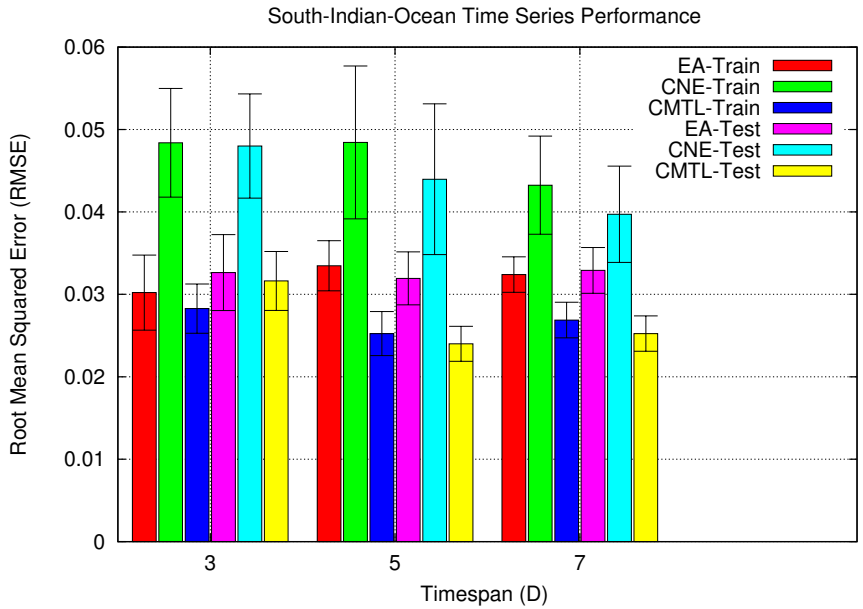


Figure: Performance given by EA, CNE, CMTL for South Indian Ocean time series.

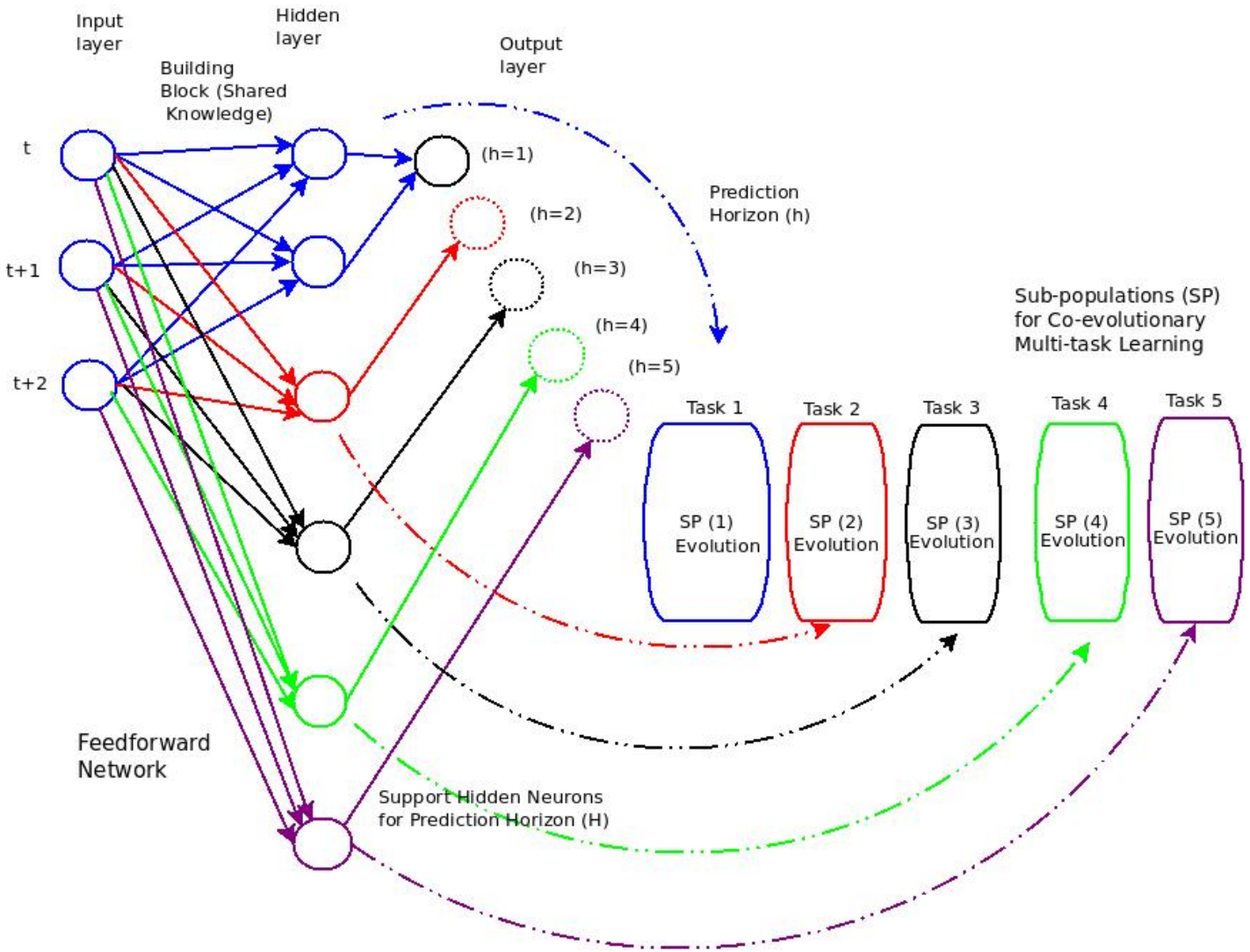
Further results given in this technical report. [8]

[8] R. Chandra, YS Ong, CK Goh: Co-evolutionary multi-task learning for dynamic time series prediction. CoRR abs/1703.01887 (2017)

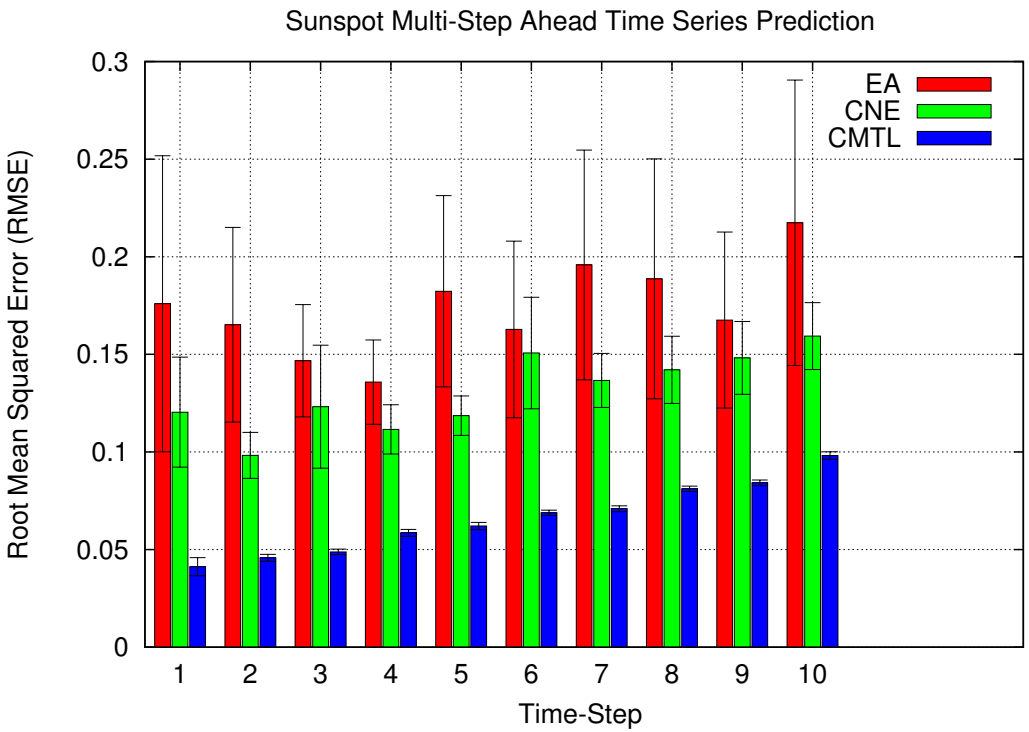
Multi-step-ahead prediction

- Multi-task learning employs a shared representation of knowledge for learning several instances of the same problem. Multi-step time series problem is one of the most challenging problems for machine learning methods.
- The performance of a prediction model face challenges for higher prediction horizons due to the accumulation of errors.
- Recently, co-evolutionary multi-task learning has been proposed for dynamic time series prediction. In this paper, we adapt co-evolutionary multi-task learning for multi-step prediction where predictive recurrence is developed to feature knowledge from previous states for future prediction horizon. The major goal of the paper is to present a network architecture with predictive recurrence which is capable of multi-step prediction through features of multi-task learning.

Multi-step-ahead prediction



Results for Sunspot time series problem



Further results are given in this publication. [9]

[9] R. Chandra, YS Ong, CK Goh: Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction. Neurocomputing 243: 21-34 (2017)

Uncertainty quantification with Bayesian methods

Bayesian inference

- Bayesian inference applies Bayes' theorem to update the probability for a hypothesis as more evidence or information becomes available.
- Bayesian inference computes the posterior probability according to

$$P(H \mid E) = \frac{P(E \mid H) \cdot P(H)}{P(E)} \tag{5}$$

- H is any hypothesis whose probability may be affected by data. E corresponds to new data that were not used in computing the prior probability.
- $P(H)$ is the prior probability and $P(H \mid E)$ is the posterior probability.
- $P(E \mid H)$ is the probability of observing E given H. This is the likelihood as it indicates the compatibility of the evidence with the given hypothesis.
- $P(E)$ is the marginal likelihood. Further information is given online.[10]

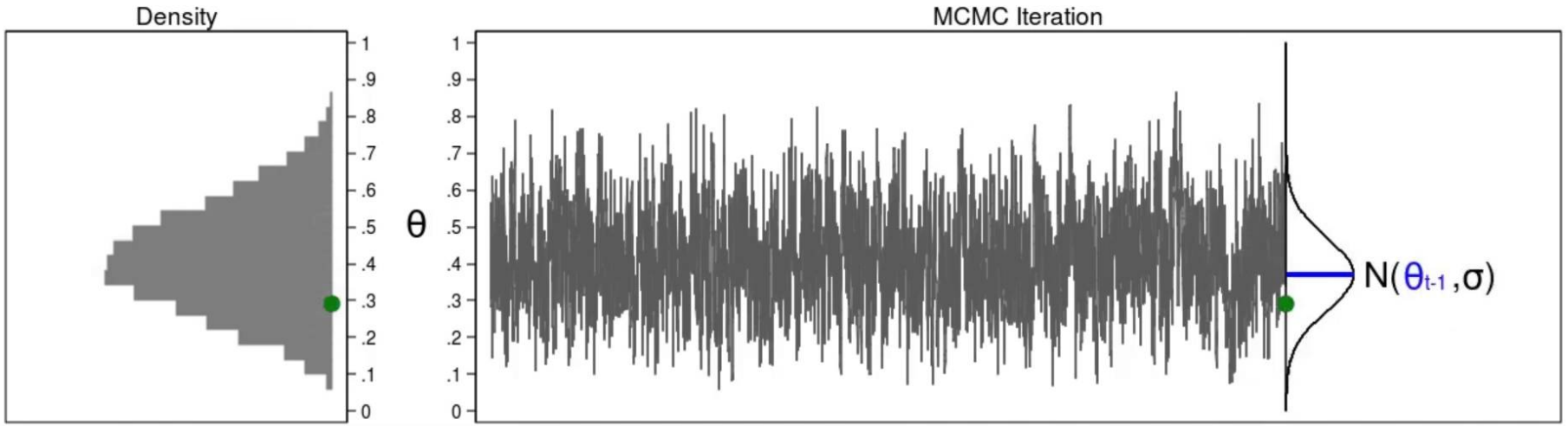
[10] Wikipedia page on Bayesian inference

Markov Chain Monte Carlo algorithms

- Markov chain Monte Carlo (MCMC) methods are a class of algorithms for sampling from a probability distribution
- Examples of random walk Monte Carlo methods include the following: Metropolis-Hastings algorithm, Gibbs sampling, Slice sampling, Multiple-try Metropolis, and Reversible-jump. Further information is given online. [11]

[11] Wikipedia page on Markov Chain Monte Carlo

MCMC



Step 1: $r(\theta_{new}, \theta_{t-1}) = \frac{\text{Posterior}(\theta_{new})}{\text{Posterior}(\theta_{t-1})} = \frac{\text{Beta}(1,1,0.290) \times \text{Binomial}(10,4,0.290)}{\text{Beta}(1,1,0.371) \times \text{Binomial}(10,4,0.371)} = 0.773$

Step 2: Acceptance probability $\alpha(\theta_{new}, \theta_{t-1}) = \min\{r(\theta_{new}, \theta_{t-1}), 1\} = \min\{0.773, 1\} = 0.773$

Step 3: Draw $u \sim \text{Uniform}(0,1) = 0.420$

Step 4: If $u < \alpha(\theta_{new}, \theta_{t-1})$ → If $0.420 < 0.773$ Then $\theta_t = \theta_{new} = 0.290$

Otherwise $\theta_t = \theta_{t-1} = 0.371$

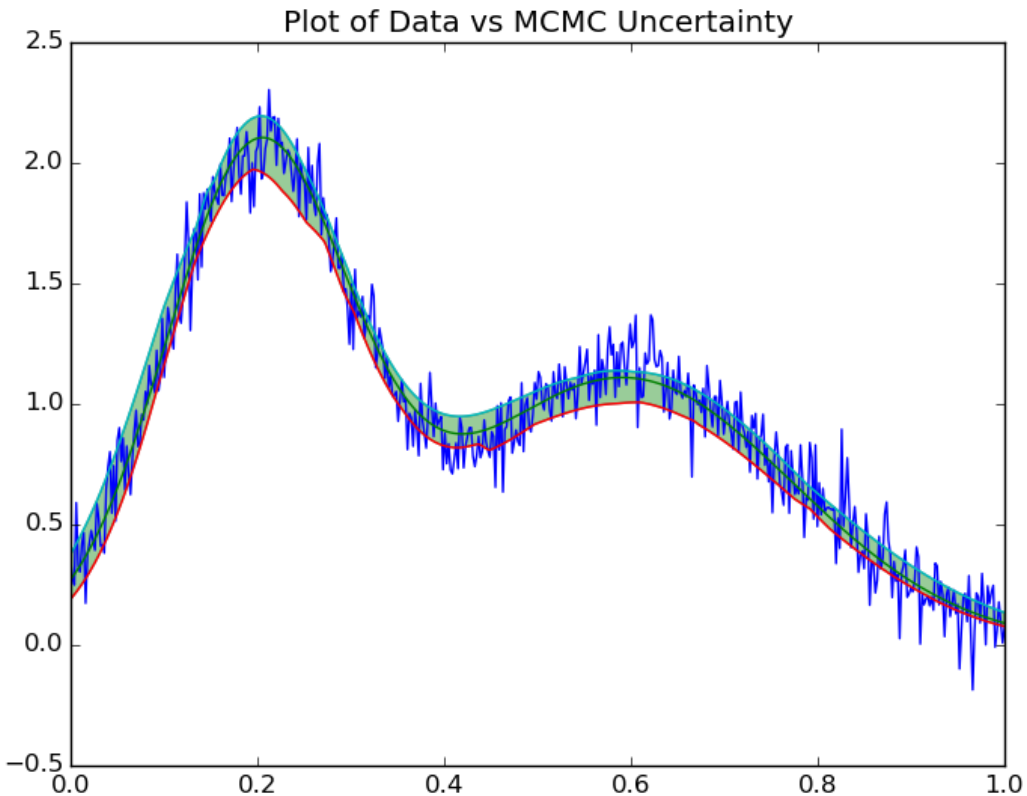
Figure:

MCMC Random-Walk

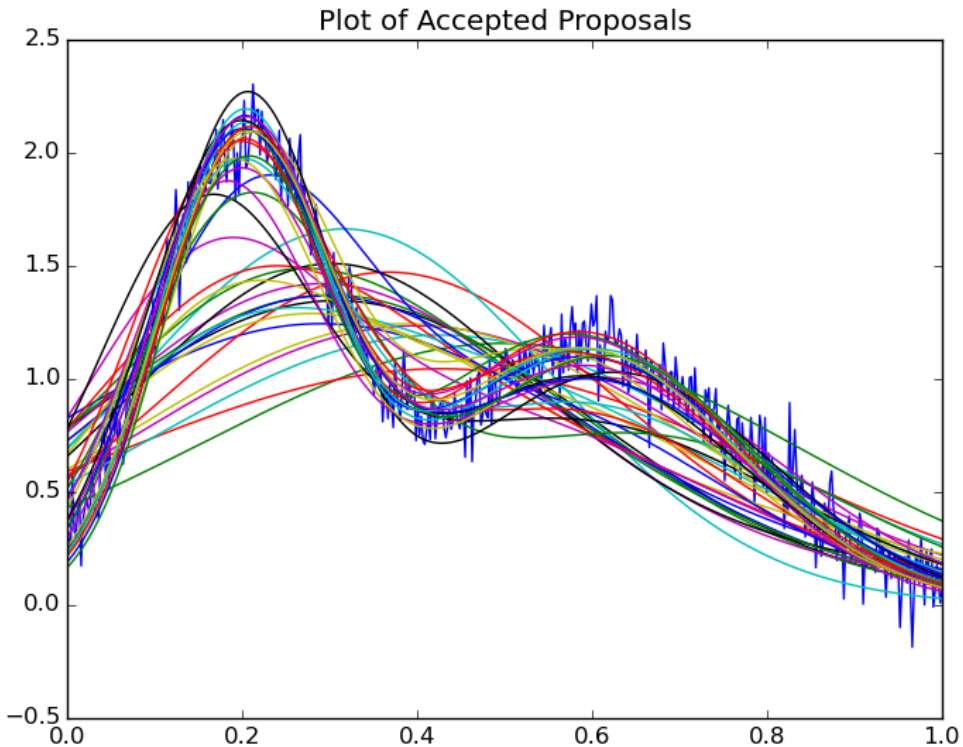
Model: Weighted Mixture of Normal Distribution

$$y = \sum_{i=1}^{n} w_i \, N_i(\mu, \sigma) \tag{6}$$

where, μ is the mean, and σ is the variance and N is a normal distribution.



MCMC Random-Walk



MCMC Random-Walk in Python [12]

[12] https://github.com/rohitash-chandra/mcmc-randomwalk

MCMC examples

- Comparison of Rejection Sampling (REJ), Sequential Monte Carlo methods (SMC) and MCMC algorithm.

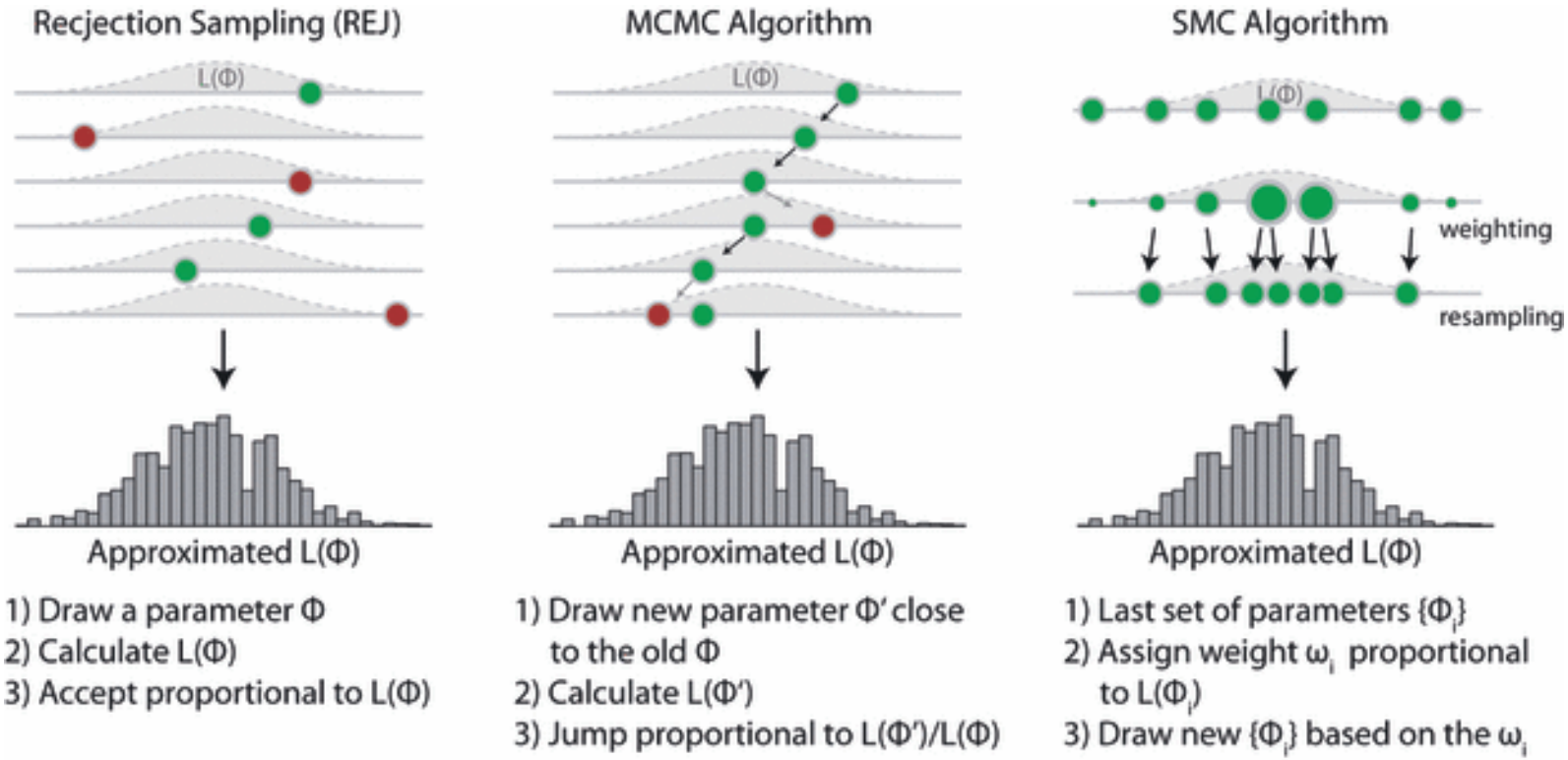


Figure: Comparison of selected MCMC algorithms

Bayesian Neural Networks

- Through canonical backpropagation neural network, we can only obtain point estimates of the weights in the network which do not account for uncertainty in the parameters (weights).
- A Bayesian approach to neural networks can potentially avoid some of the pitfalls of stochastic-optimization.
- Bayesian neural networks (BNNs) use Markov Chain Monti-Carlo (MCMC) methods such as those based on e.g the Laplace approximation, Hamiltonian Monte Carlo, expectation propagation and variational inference. However, these approaches have not seen widespread adoption due to their lack of scalability in both network architecture and data size. A number of attempts have been proposed combining MCMC techniques with the gradient optimization algorithms

Bayesian Neural Networks

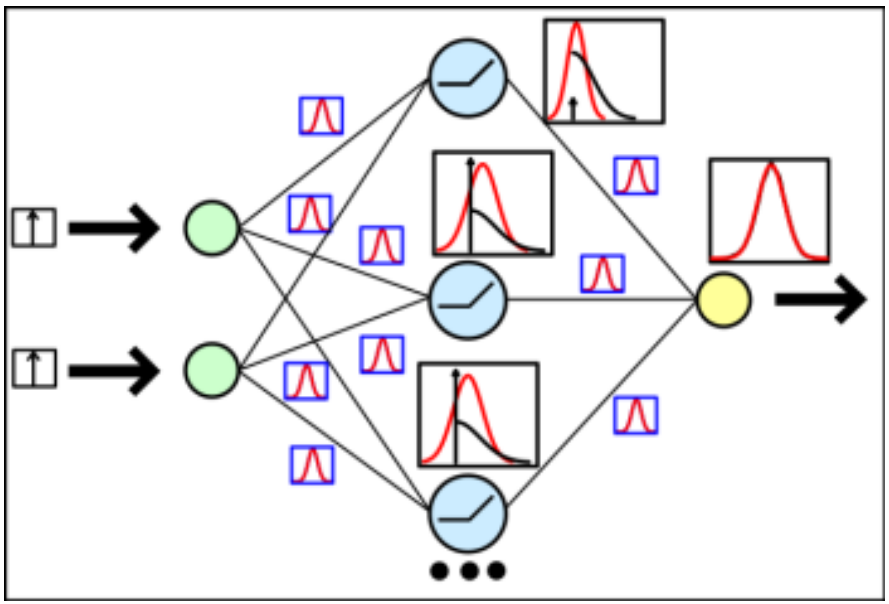


Figure: Bayesian Neural Network

Bayesian Neural Networks

- To conduct a Bayesian analysis we need to specify prior distributions for the elements of θ, which we choose to be

$$\begin{aligned} v_h &\sim \mathcal{N}(0, \sigma^2) \text{ for } h = 1, \ldots, H, \\ \delta_0 &\sim N(0, \sigma^2) \\ \delta_h &\sim N(0, \sigma^2) \\ w_{dj} &\sim \mathcal{N}(0, \sigma^2) \text{ for } h = 1, \ldots, H \text{ and } d = 1, \ldots, D, \\ \tau^2 &\sim \mathcal{IG}(\nu_1, \nu_2) \end{aligned} \tag{7}$$

  where H is the number of hidden neurons.
- The implementation in Python is also given online [13].

[13] https://github.com/rohitash-chandra/LDMCMC_timeseries

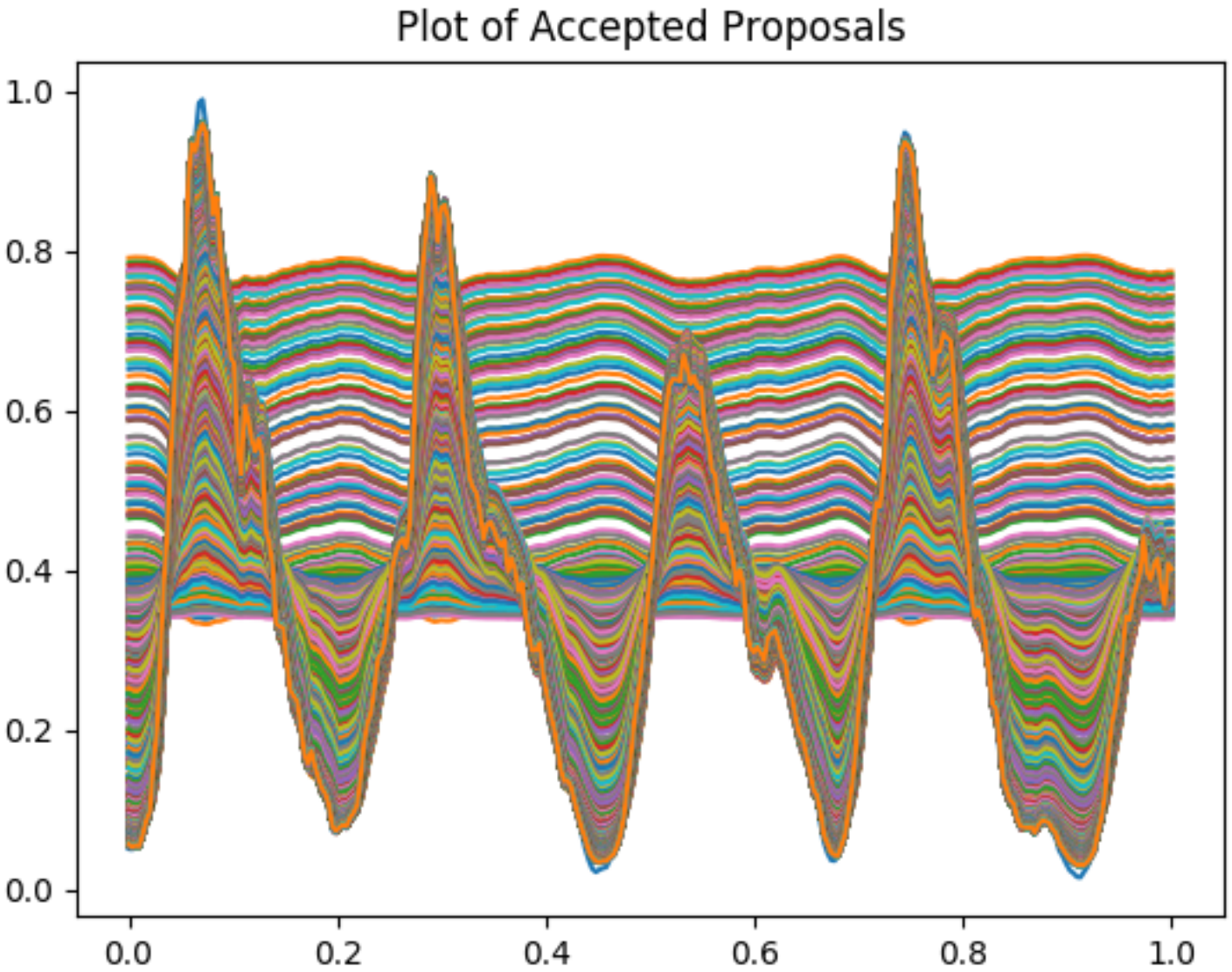
Langevin dynamics MCMC algorithm

Alg. 1 Langevin dynamics MCMC for neural networks

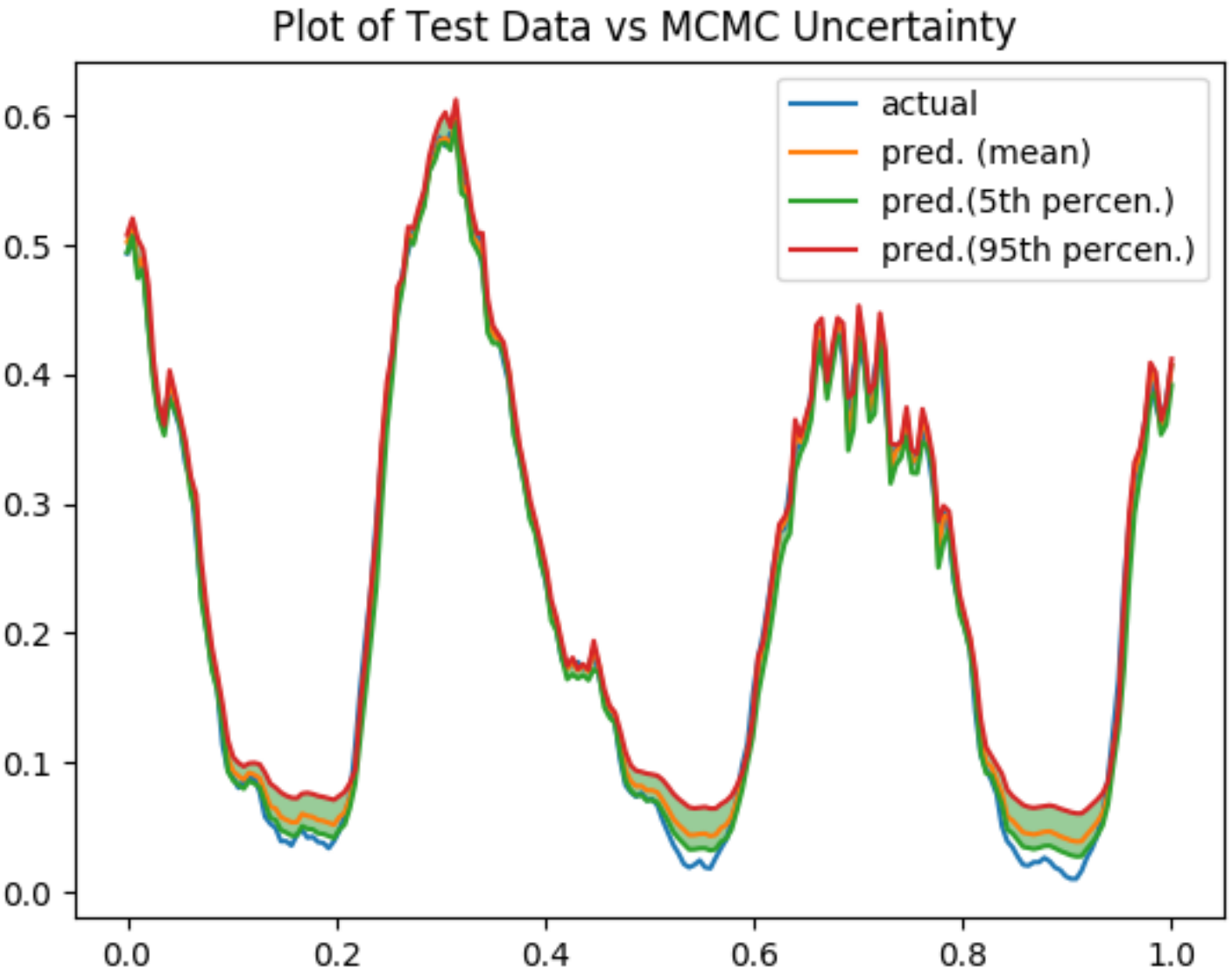
Step 1: Define feedforward neural network: $net = network(input, hidden, output)$. Note D =input, H=hidden, and output=1.
Step 2: Data reconstruction into state-space vector using Taken's theorem. Definition of training and test datasets with embedding dimension (D) and time lag (T), $\mathbf{y}_{\mathcal{A}_{D,T}}$ defined by (2)
Step 3: Define weights vector $\tilde{\mathbf{w}}$ from the feedforward neural network:
Step 4: $k = 0$
Step 5: Initialize $\theta = \theta^{[k]}$.
while $k <$ max samples **do**
  1.Compute $\Delta\boldsymbol{\theta}^{[k]}; \mathbf{y}_{\mathcal{A}_{D,T}}$ given by (7).
  for Depth d **do**
    i. Forwardpropagate
    ii. Backpropagate
  end for
  2. Draw $\boldsymbol{\eta}^{[1]}$ from $\mathcal{N}(0, \Sigma_\eta)$.
  3. Propose $\boldsymbol{\theta}^* = \theta^{[k]} + \Delta\boldsymbol{\theta}^{[k]} + \boldsymbol{\eta}$
  4. Draw $u \sim \mathcal{U}[0, 1]$.
  5. If $u < \alpha$, where α is given by (8) set $\boldsymbol{\theta}^{[k+1]} = \boldsymbol{\theta}^*$ else set $\boldsymbol{\theta}^{[k+1]} = \boldsymbol{\theta}^{[k]}$
  6. $k = k + 1$
end while

Figure: Algorithm

Bayesian Neural Networks: Time Series Prediction



Bayesian Neural Networks: Time Series Prediction



Bayesian Neural Networks: Time Series Prediction

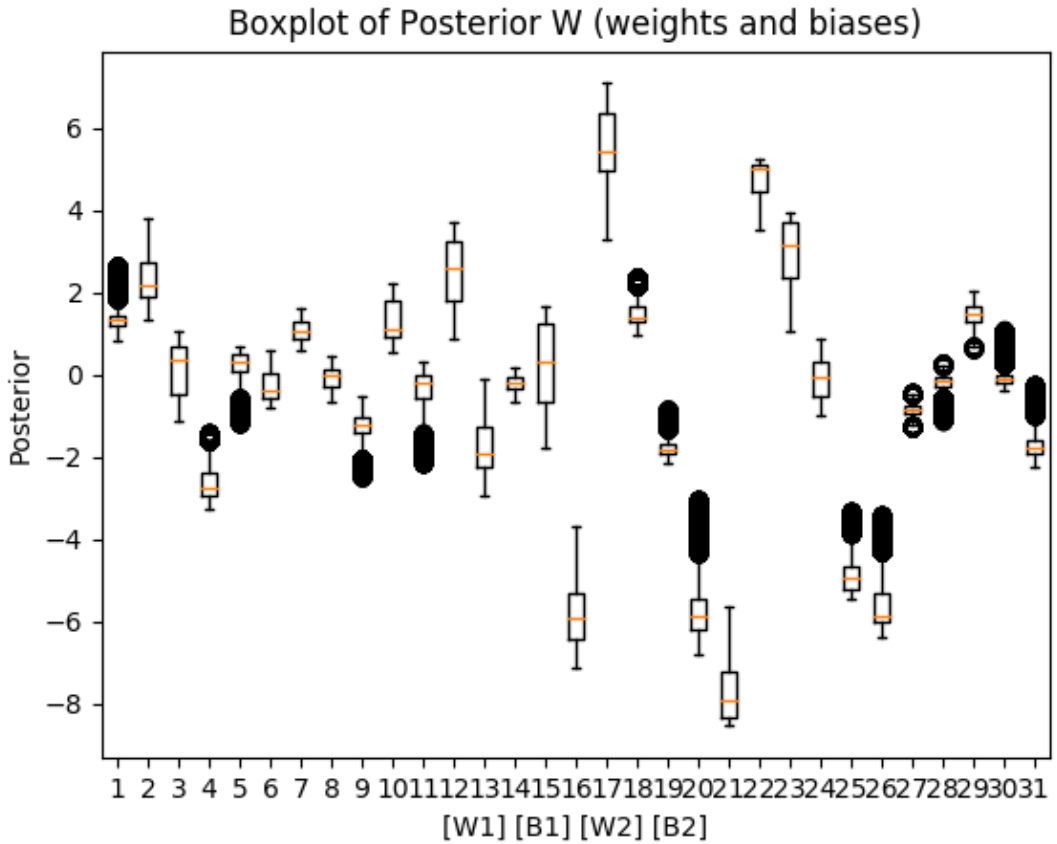


Figure: Weights (posterior) [14]

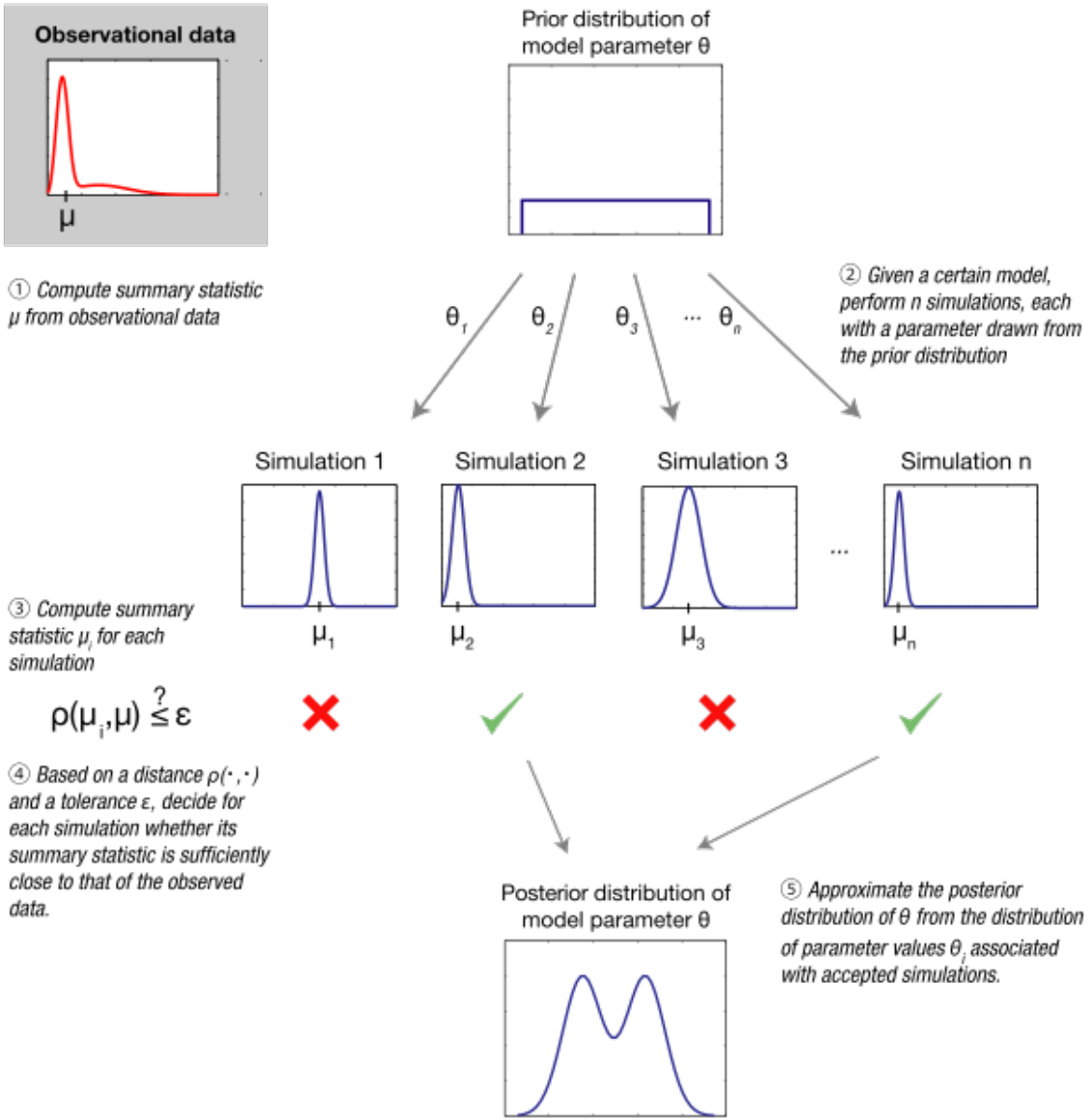
[14] R. Chandra, L. Azizi, S. Cripps, Bayesian neural learning via Langevin dynamics for chaotic time series prediction, ICONIP 2017 (Under Review).

Approximate Bayesian Computation

- In real world applications, an analytical formula for likelihood function could be elusive or the likelihood function might be computationally very costly to evaluate. Approximate Bayesian Computation (ABC) attempts to address this problem.
- "ABC methods bypass the evaluation of the likelihood function. In this way, ABC methods widen the realm of models for which statistical inference can be considered. ABC methods are mathematically well-founded, but they inevitably make assumptions and approximations whose impact needs to be carefully assessed. " [15]
- Implementation in Python for a simple problem [16]

[15] Wikipedia page on Approximate Bayesian Computation.
[16] Python code for Approximate Bayesian Computation.

ABC


Observational data
μ
Prior distribution of model parameter θ
① Compute summary statistic μ from observational data
θ1
θ2
θ3
... θn
② Given a certain model, perform n simulations, each with a parameter drawn from the prior distribution
Simulation 1
Simulation 2
Simulation 3
Simulation n
μ1
μ2
μ3
μn
③ Compute summary statistic μi for each simulation
ρ(μi,μ) ≤? ε
④ Based on a distance ρ(·,·) and a tolerance ε, decide for each simulation whether its summary statistic is sufficiently close to that of the observed data.
Posterior distribution of model parameter θ
⑤ Approximate the posterior distribution of θ from the distribution of parameter values θi associated with accepted simulations.

Landscape dynamics and geocoastal modeling

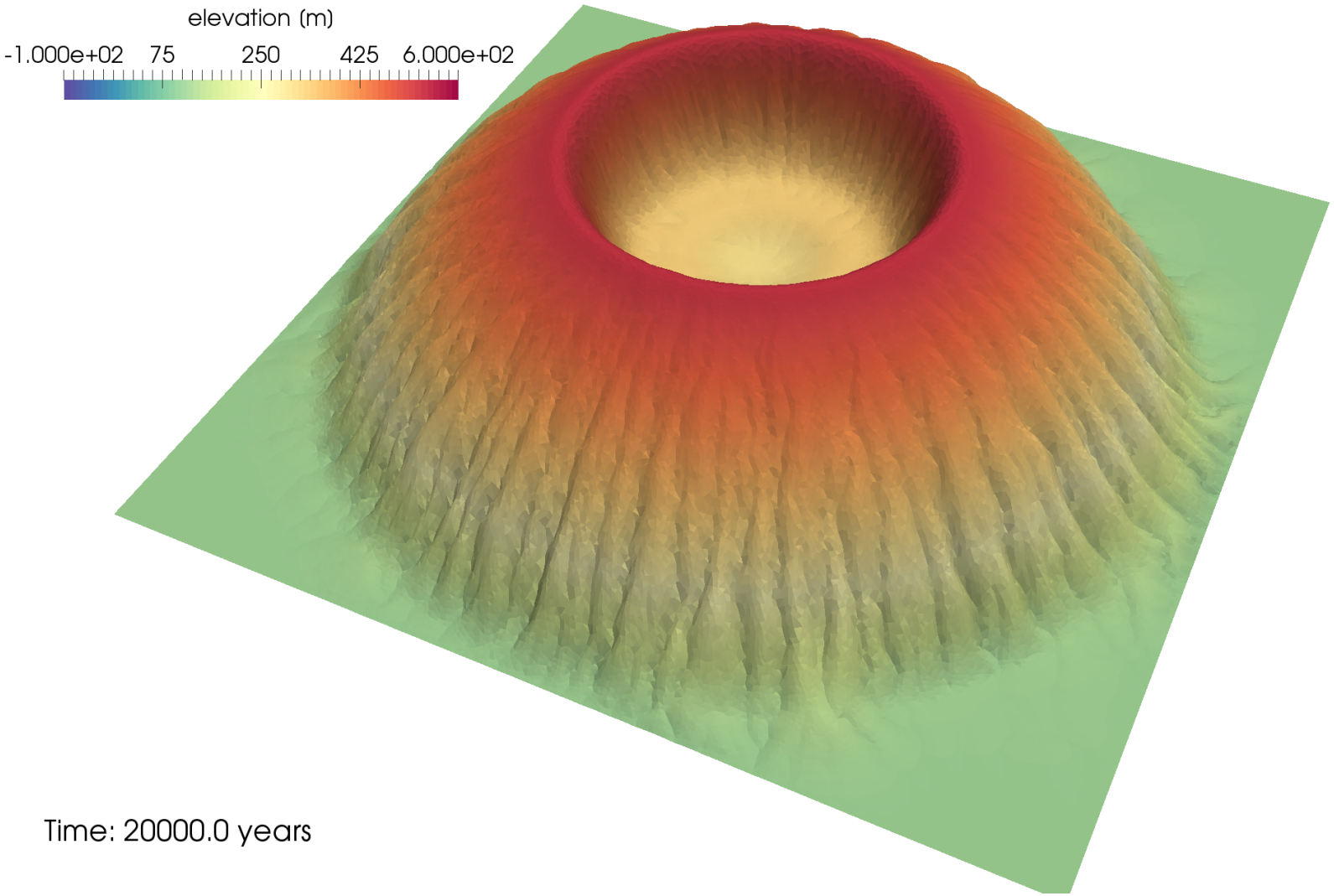
1. Introduction to machine learning and neural networks
2. Neural networks for climate extremes: cyclones
   - Encoding cyclone track prediction problem in coevolutionary RNNs
   - Minimal timespan problem for cyclone wind-intensity prediction
   - Rapid intensification identification in tropical cyclones
   - Transfer and Multi-task learning
   - Multi-task learning for wind intensity prediction
   - Multi-task learning for multi-step-ahead prediction
3. Uncertainty quantification with Bayesian methods
4. Landscape dynamics and geocoastal modeling
5. Prospects for machine learning in climate modeling

Badlands

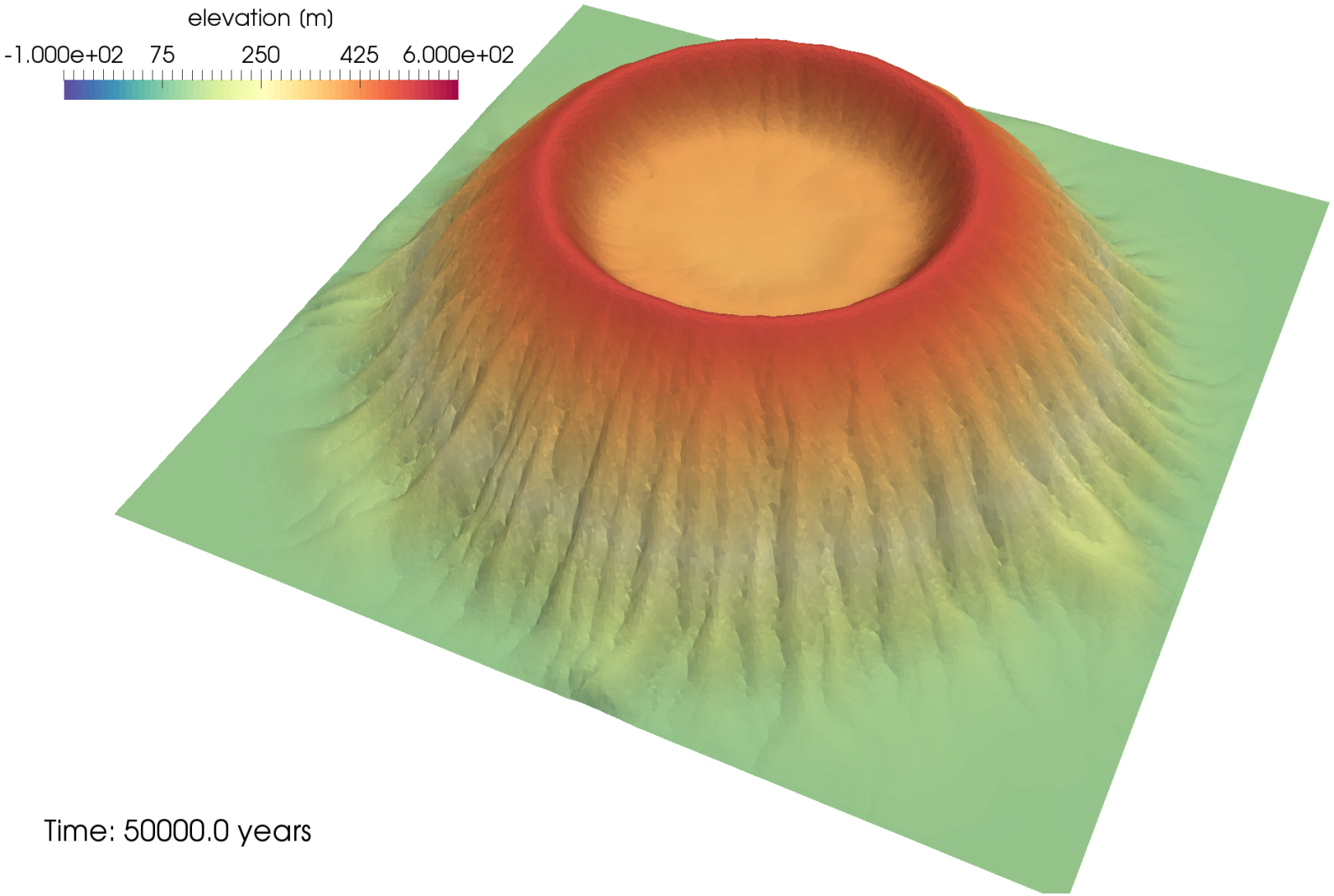
- Basin and Landscape Dynamics (Badlands) is used to simulate topography development at various space and time scales.
- "The model is capable of simulating hillslope processes (linear & non-linear diffusion), fluvial incision ('modified' Stream Power Law, Transport Capacity Law both for sediment erosion/transport/deposition), spatially and temporally varying geodynamic (horizontal + vertical displacements) and climatic forces which can be used to simulate changes in base level, as well as effects of climate changes or sea-level fluctuations. " [17]

[17] https://github.com/badlands-model/pyBadlands

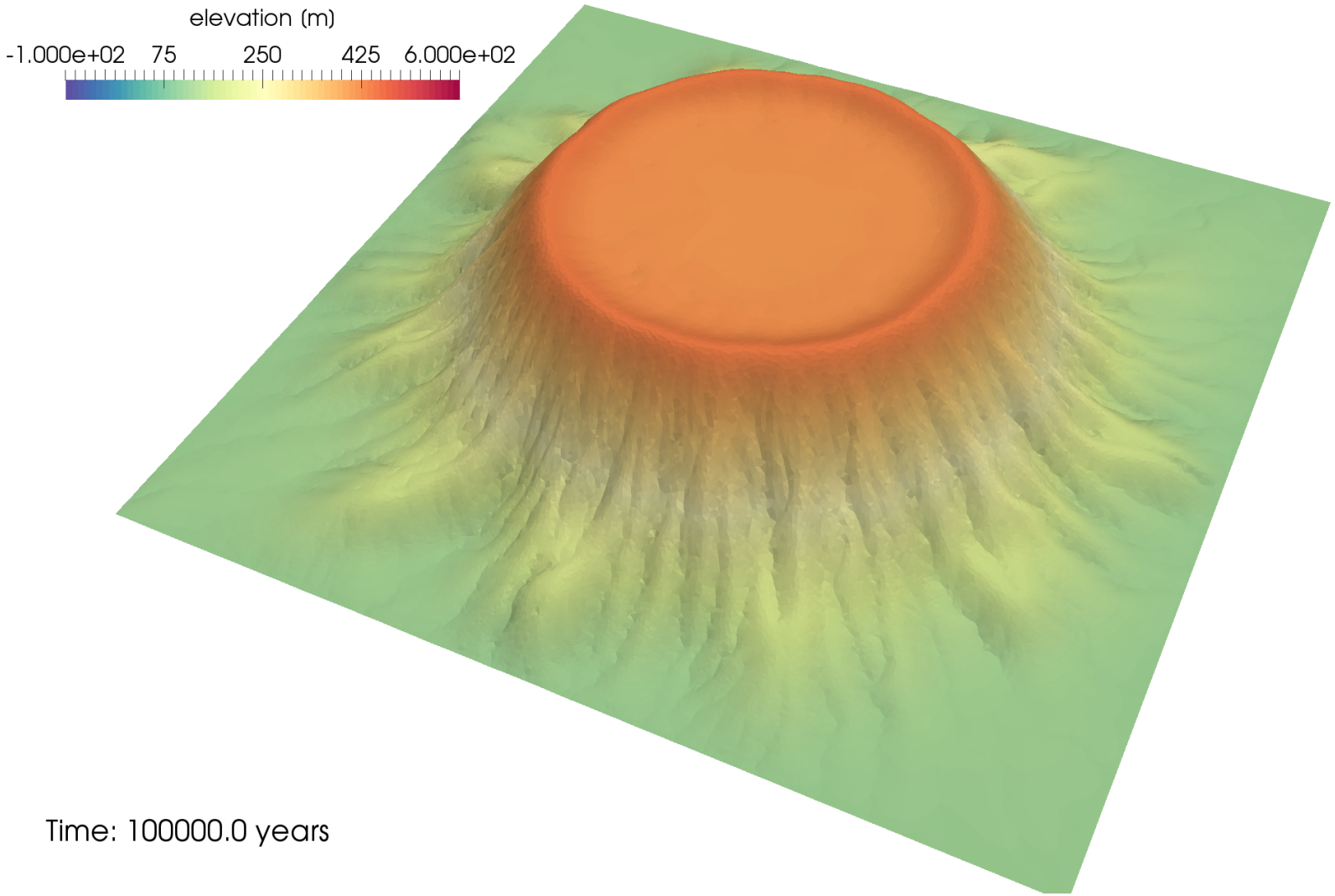
Badlands example



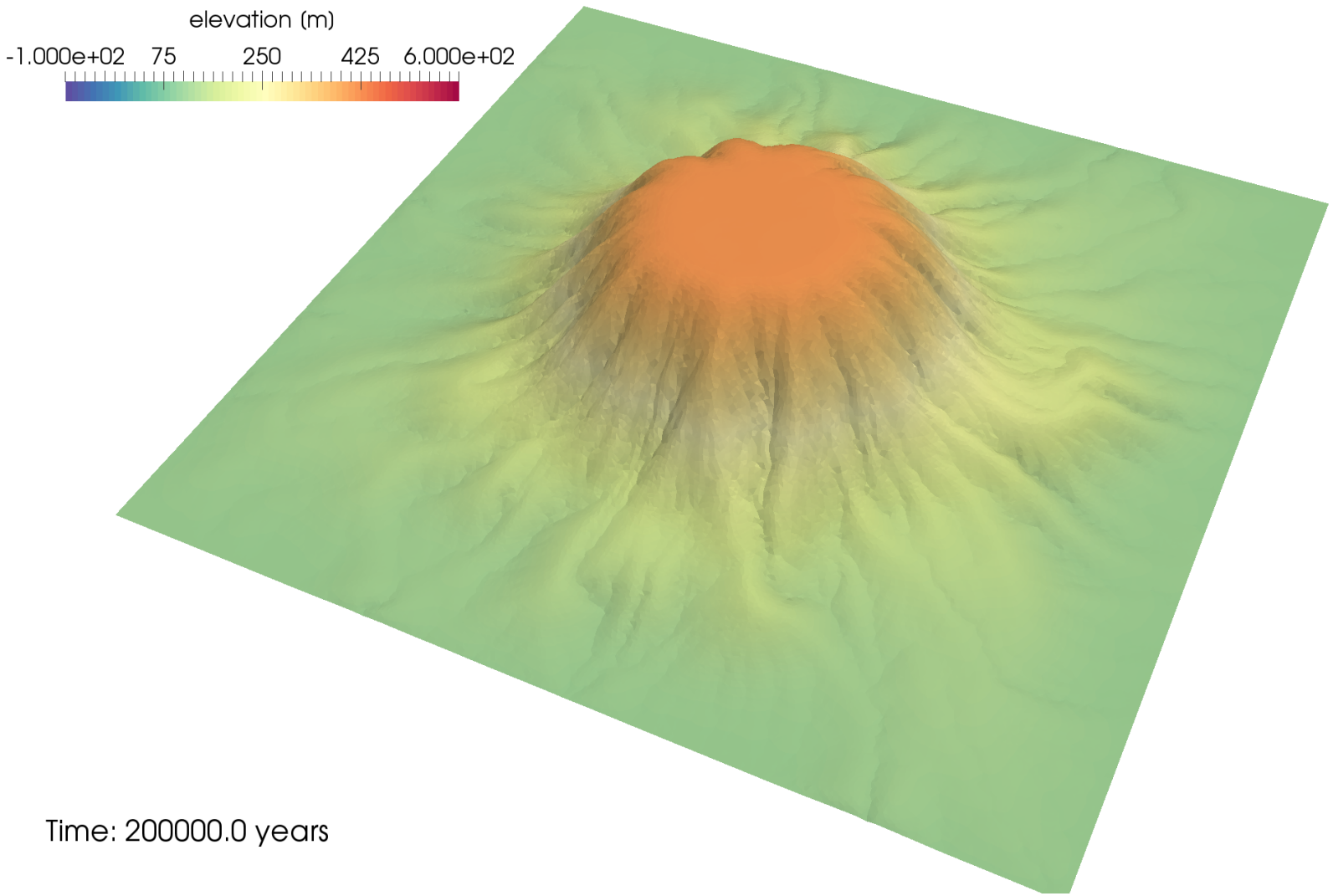
Badlands example



Badlands example

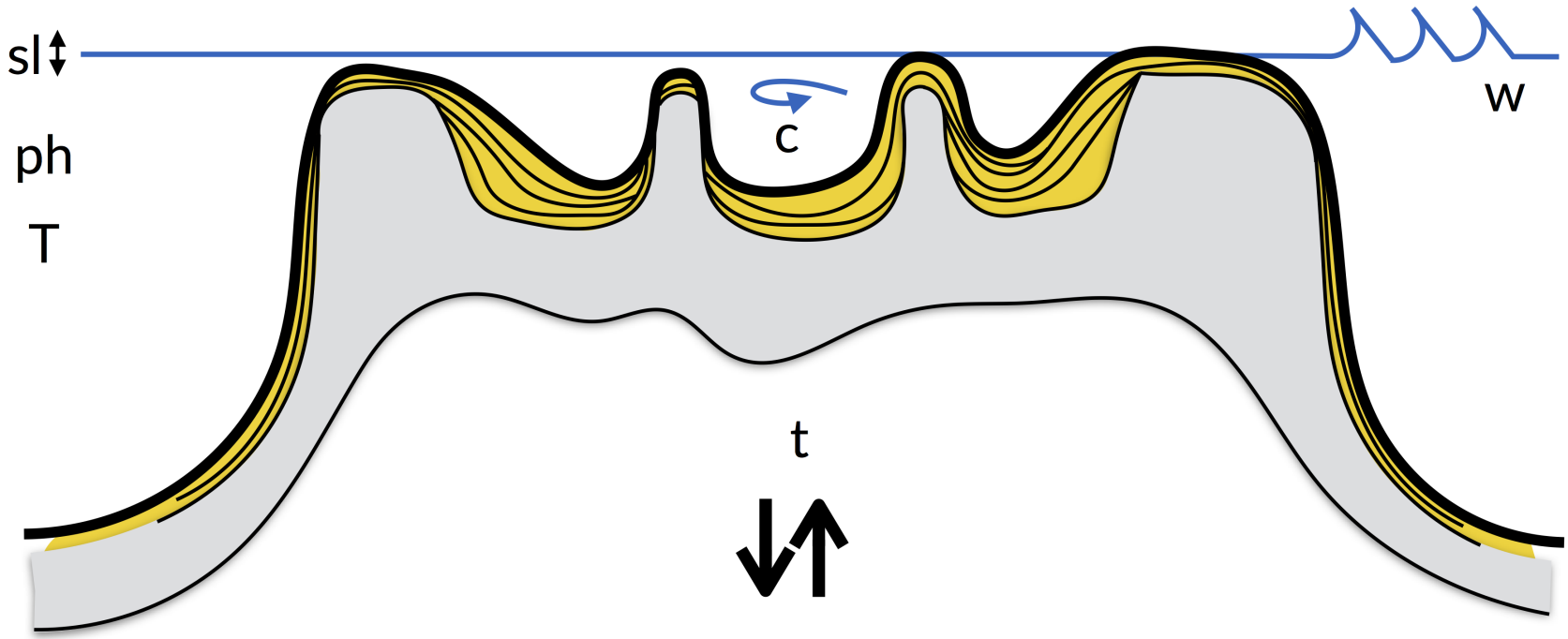


Badlands example



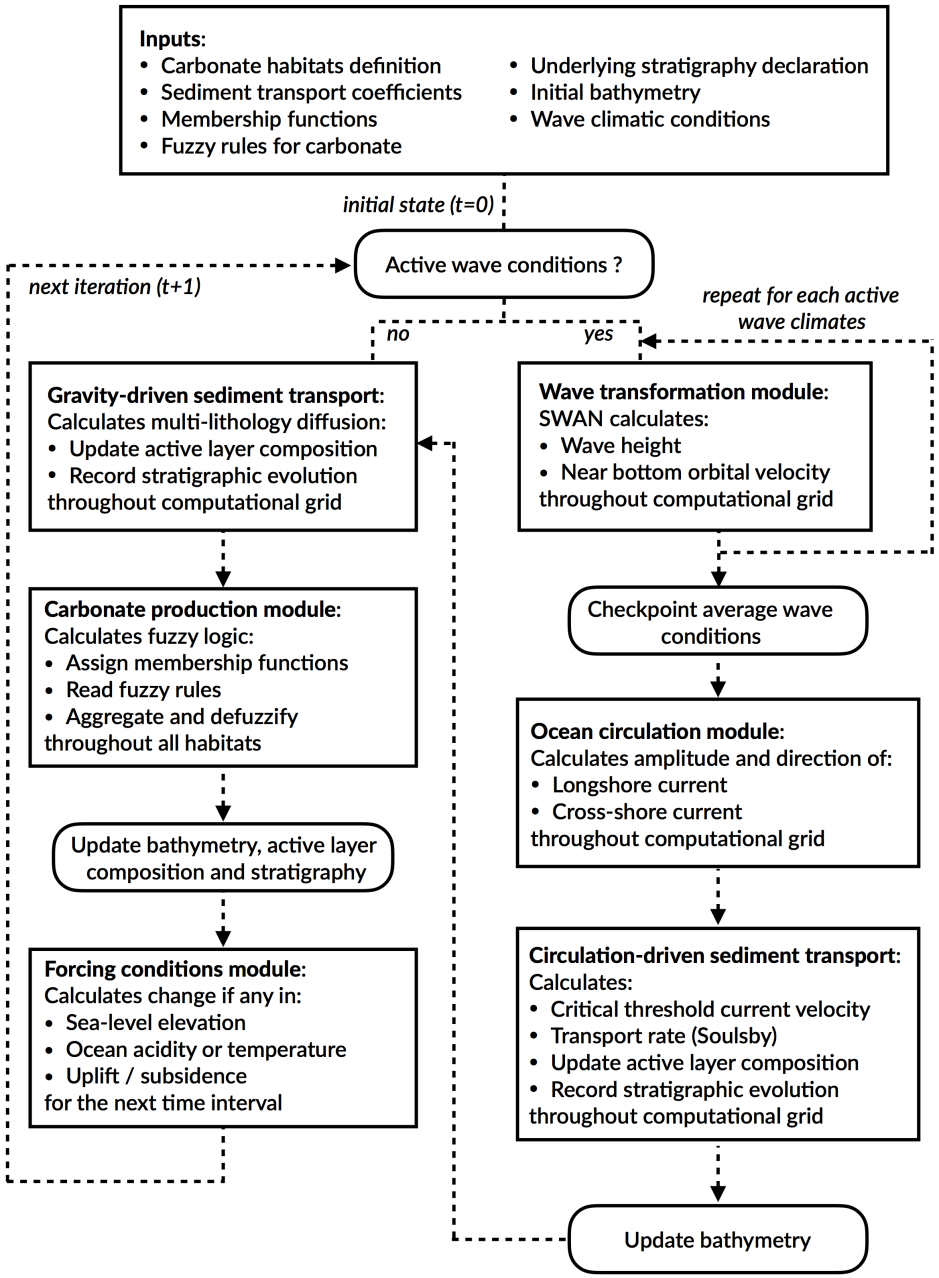
Modeling Coral Reefs using PyReef

"pyReef is a stratigraphic forward model that predicts reef evolution over geological time scale. This parallel, open-source model uses swan (wave generation model) to simulate wave propagation under different climatic conditions and associated sediment transport" [18]



[18] https://github.com/pyReef-model/pyReef

PyReef example



Prospects for machine learning in climate modeling

Prospects for machine learning in climate modeling

- Handle uncertainty in models and data through Bayesian machine learning and optimization.
- Address problems where data is difficult to obtain or not much data available through methods such as Approximate Bayesian Computation
- Use multiple sources of data and multiple models through transfer learning and multi-task learning approaches.
- Use multi-task and transfer learning in areas of climate extremes such as storms and cyclone prediction.
- Address problems of unbalanced datasets such as rapid intensification of cyclones.
- Use Bayesian neural networks and related methods to provide uncertainty quantification in prediction for extreme climate events.
- Use machine learning methods for surrogate assist models in order to extend physical models for large scale implementation.

Resources

- Open course (MOOCs) for Neural Networks: openlearning.com/courses/neural-networks-fundamentals-and-applications
- Code for related methods: https://github.com/rohitash-chandra
- My research papers: https://www.researchgate.net/profile/Rohitash_Chandra
- Keras deep learning library https://keras.io/
- Skitlearn machine learning library http://scikit-learn.org/stable/
- MCMC in Python: https://pymc-devs.github.io/pymc/

Download the slides [19]

[19] https://github.com/rohitash-chandra/research

Questions and comments :)