Promoting In-depth Development of Neural Net Applications for Climate Prediction and Services Improvement

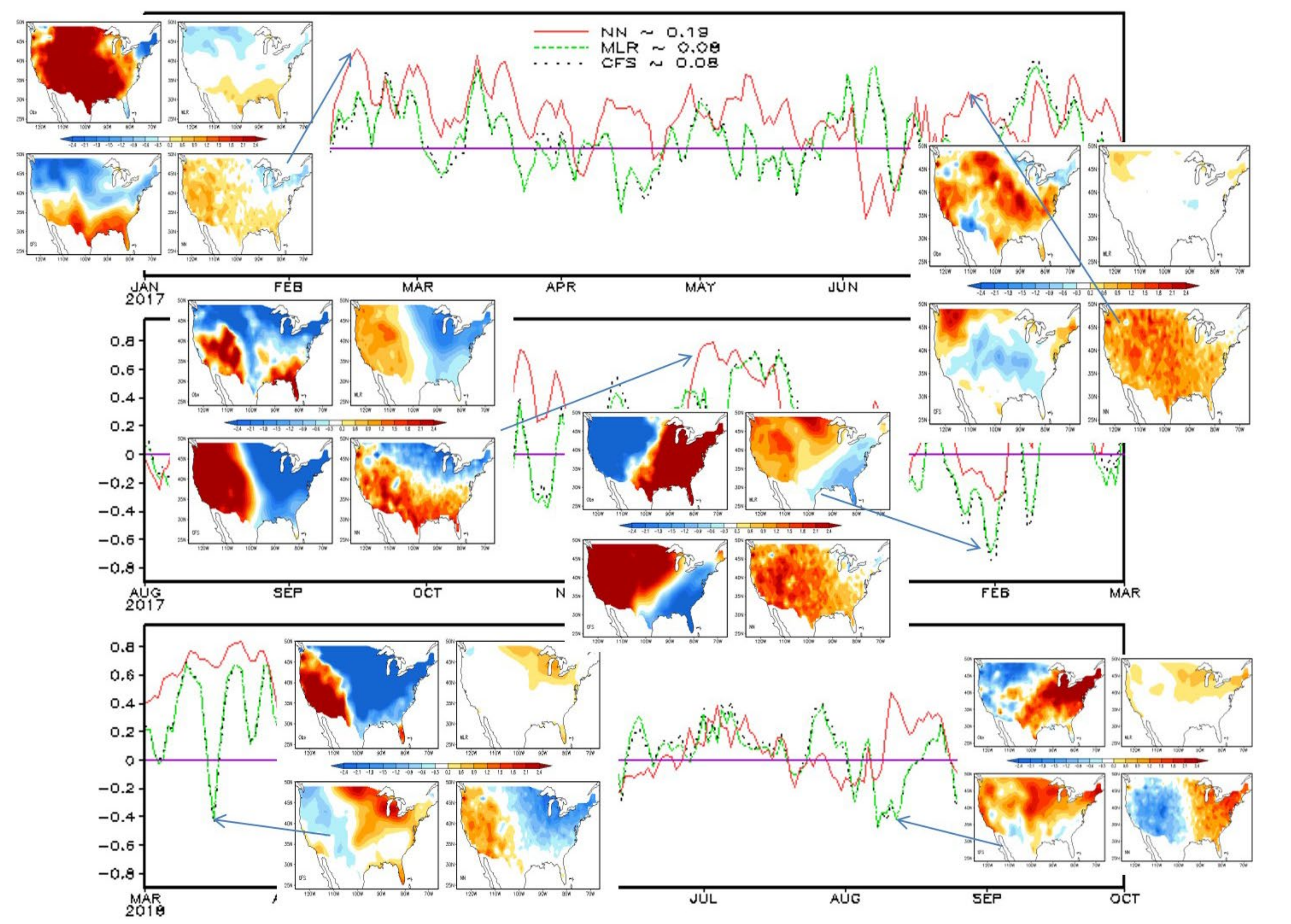
Jiayu Zhou, Climate Mission, Office of Science and Technology Integration
David DeWitt, Climate Prediction Center, National Centers for Environmental Prediction
National Weather Service, National Oceanic and Atmospheric Administration, U.S. Department of Commerce

Compared to weather forecasting, climate prediction is more difficult in terms of improvement of skill and reliability along with the development of data, models and statistical methodologies. It advances slowly not only because of the limitation of climate predictability imposed by natural variabilities but also due to the complexity of the climate system, which is not properly handled by prediction models.

Neural Nets (NN) stemming from the character of human brain activities are designed to make nonlinear mapping without any presuppositions. It attracts more interests accompanied by data explosion and rapid advancement in parallel computing capabilities. This presentation highlights NWS recent in-depth development of NN applications (both feedforward and recurrent networks) to assist conventional physics-based model outcomes for operational subseasonal-to-seasonal climate prediction with promising improvement in products and services.

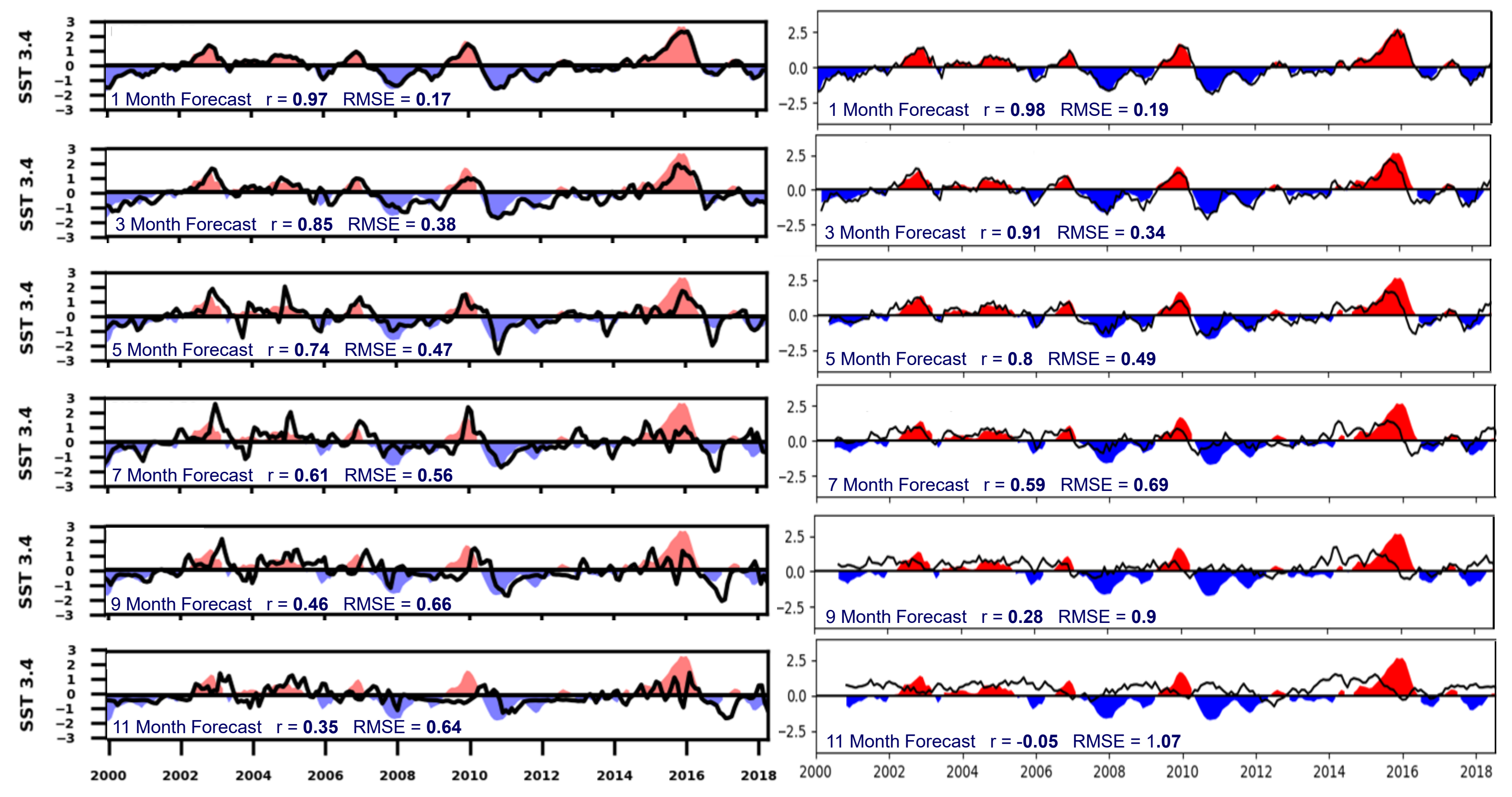
Promising Improvement

Week 3 – 4 Forecast



Time series (Jan 2017 - Oct 2018) of spatial anomaly correlation between observation and week 3-4 T_{2m} forecast by ***Feedforward*** NN, Multiple Linear Regression (MLR), and NCEP Climate Forecast System (CFS) indicated by red solid, green dotted and black dotted line, respectively. Each four-panel inset shows the anomaly of forecast by CFS (lower left), MLR (upper right) and NN (lower right) in comparison with that of observation (upper left) at a specific time. It shows some CFS forecasts and corresponding observations are out of phase, which are reversed by NN, but not by MLR, prompting the value of NN. (by courtesy of Yun Fan *et al.*)

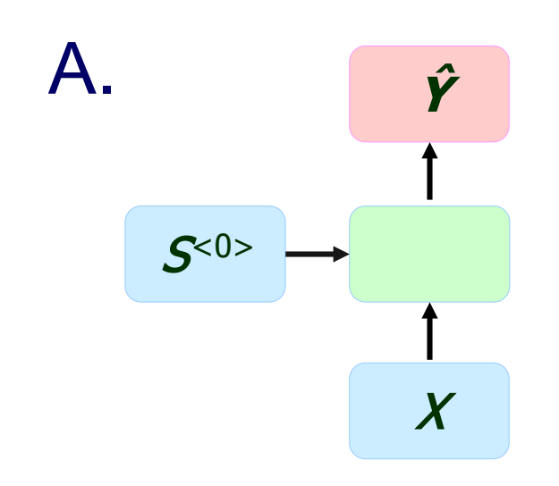
ENSO Prediction



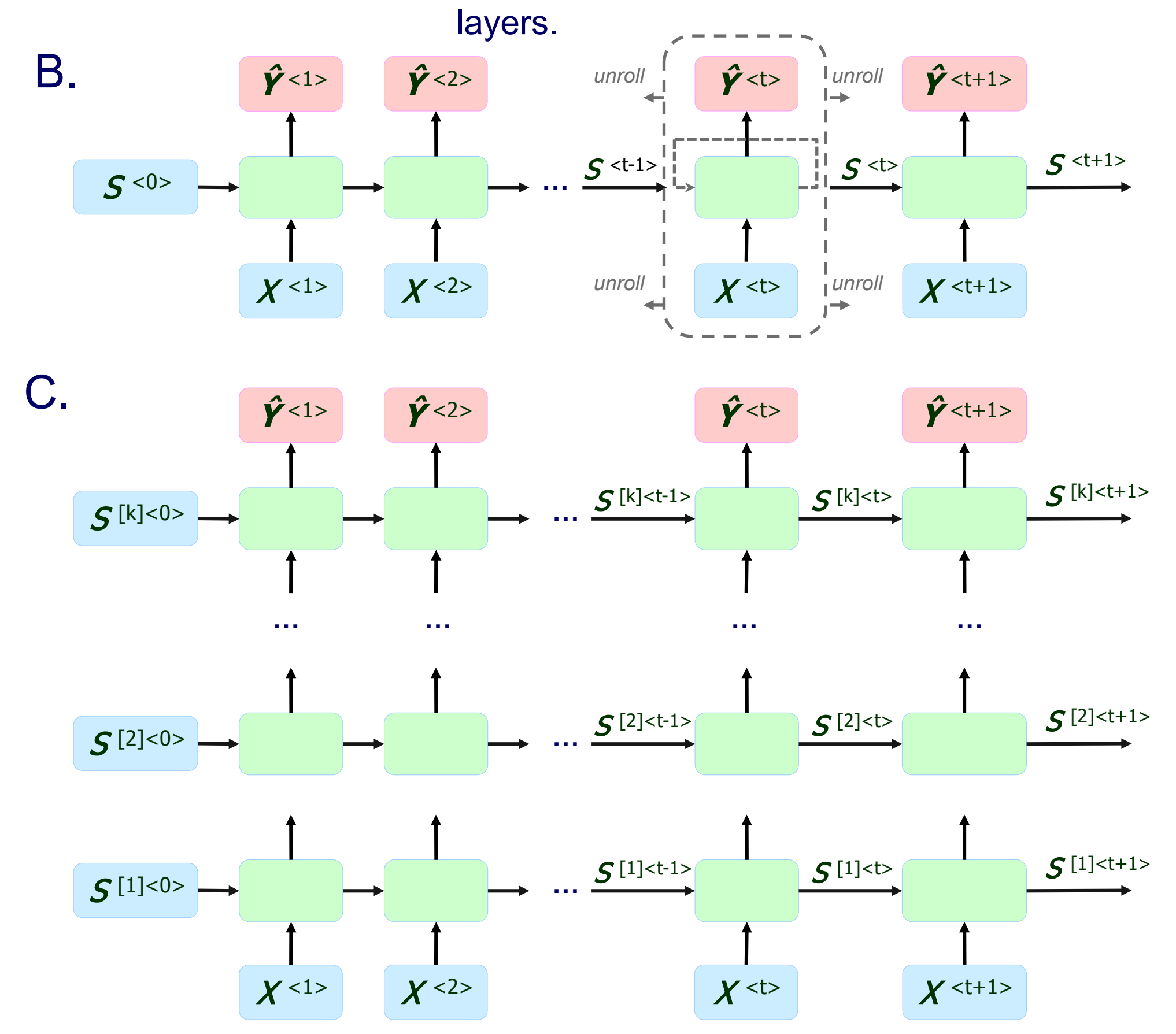
Nino 3.4 SST predictions at 1-month to 11-month lead (from top to bottom) by ***Recurrent LSTM network*** (left) with 20 nodes, 3 layers and 1000 epochs and by Linear Regression (right). The LSTM inputs are Niño 3.4 SST full values, Niño 3.4 SST anomalies, Markov model PC 2, and surface wind at 3 months back. The linear regression predictors are SST anomaly, CTP, WWV, WPAC, EPAC, and Markov model PC 2 with lags of 6 months. Predictions are indicated by contours and observations by shadings. The figure shows the prediction skill (measured by r and RMSE) of Recurrent LSTM network surpasses Linear Regression beyond 6 months. (by courtesy of Kyle MacRitchie)

Architectures and Prospectus

Generalized Framework



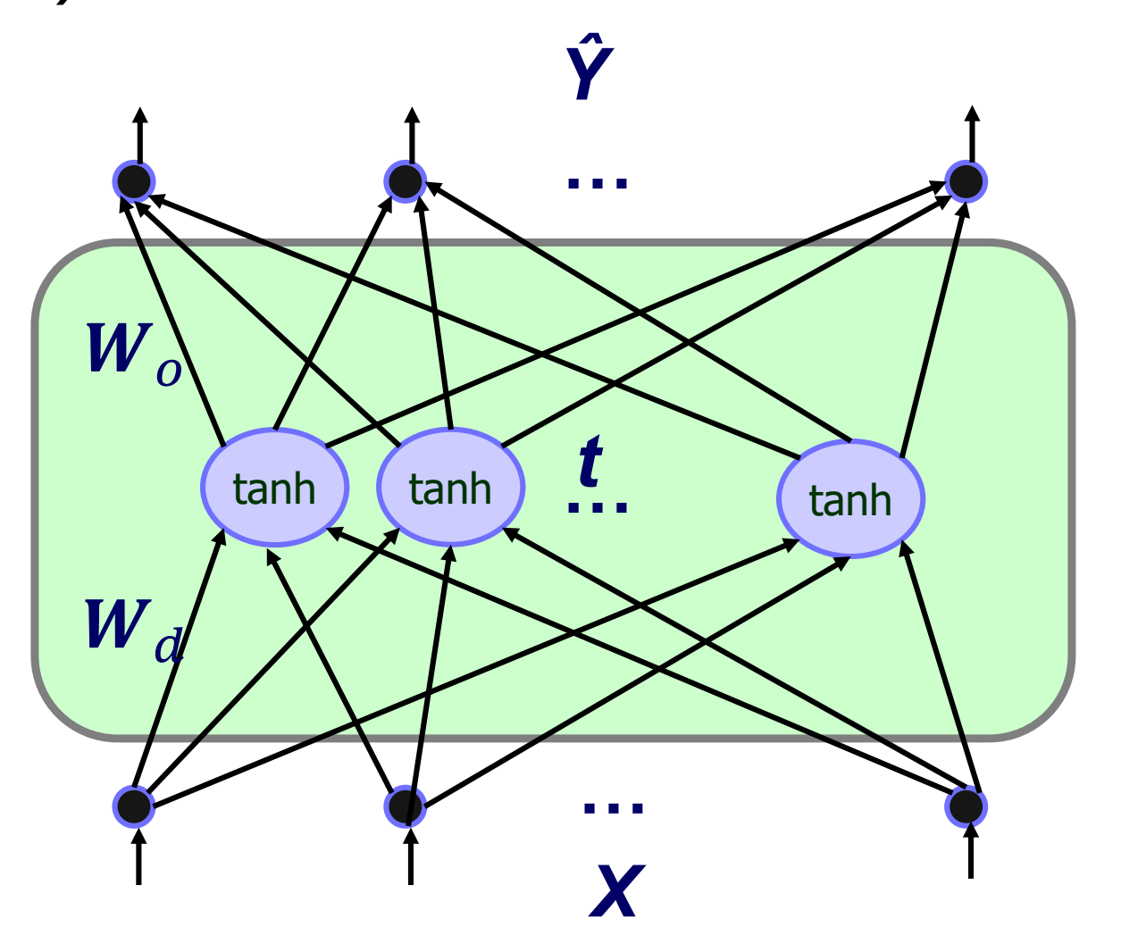
- ***A. Feedforward***: The simplest type of NN. Connections between the nodes do not form a cycle.
- ***B. Recurrent***: A chain of repeating modules (unrolled from the basic module is shown), activations not only depend on current input but also earlier inputs.
- ***C. Deep Recurrent:*** Recurrent network, activations not only at earlier times but also in previous hidden layers.



Cost/Error function: $C(\hat{\mathbf{Y}}, Y) = \frac{1}{T}\sum_{t=1}^{T} E\,(\hat{\mathbf{Y}}^{<t>}, \mathbf{Y}^{<t>})$

NN Modules

i) Feedforward

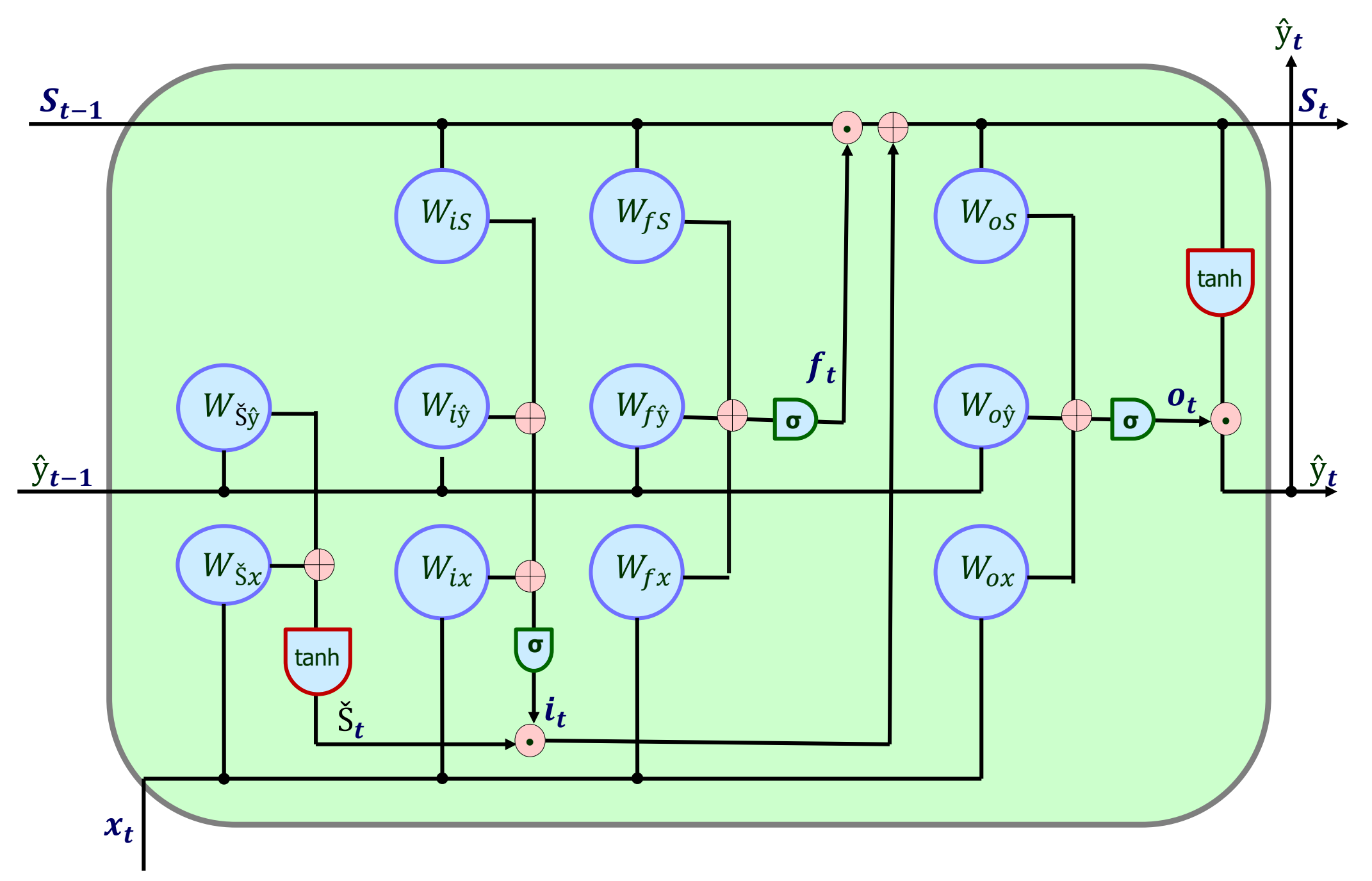


$$\hat{\mathbf{Y}} = NN(\mathbf{X}, \mathbf{W}_d, \mathbf{W}_o, \mathbf{b}_d, \mathbf{b}_o) = \mathbf{W}_o \cdot \mathbf{t} + \mathbf{b}_o$$

$$\mathbf{t} = \tanh(\mathbf{W}_d \cdot \mathbf{X} + \mathbf{b}_d)$$

NOTE: The bias parameter **b** has been omitted in both network figures for brevity.

ii) Recurrent - Long Short Term Memory (LSTM)



$$\mathbf{f}_t = \sigma\,(\mathbf{W}_f \cdot [\mathbf{S}_{t-1}, \hat{y}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$
$$\mathbf{i}_t = \sigma\,(\mathbf{W}_i \cdot [\mathbf{S}_{t-1}, \hat{y}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$
$$\check{S}_t = \tanh\,(\mathbf{W}_{\check{S}} \cdot [\hat{y}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{\check{S}})$$
$$\mathbf{S}_t = \mathbf{f}_t \odot \mathbf{S}_{t-1} + \mathbf{i}_t \odot \check{S}_t$$
$$\mathbf{o}_t = \sigma\,(\mathbf{W}_o \cdot [\mathbf{S}_t, \hat{y}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$
$$\hat{y}_t = \mathbf{o}_t \odot \tanh(\mathbf{S}_t)$$

$\odot$ denotes the element-wise multiplication between two vectors.

Gates are used to remedy the vanishing gradient problem to capture long term dependencies.

Hyperparameters

Numbers of hidden neurons and layers: Empirically determined and trimmed.

Weights initialization: Prevents layer activation outputs from exploding or vanishing during the course of a forward pass through a deep neural network.

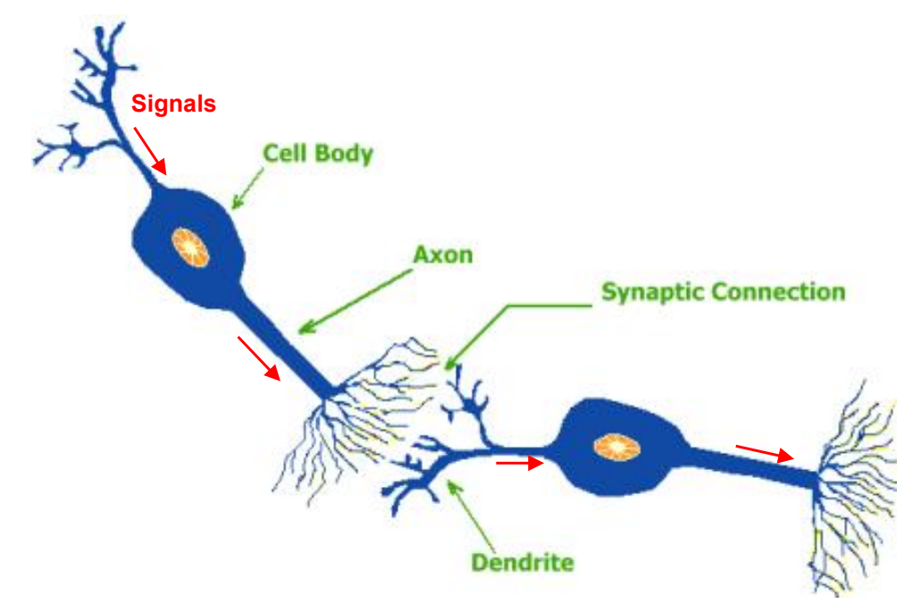
Batch size: Number of parts that the dataset is divided to feed to the computer.

Training epochs: The number of times all of the training vectors are used once to update the weights, which goes from underfitting to optimal to overfitting as the number of epochs increases.

Learning rate: Controls the amount that the weights are updated with respect to the loss gradient during training.

How Do Neural Nets Learn?

Biological Analogy



A typical nerve cell/neuron contains three parts, 1) dendrites carry signals in, 2) cell body contains nucleus, and 3) axon carries signals away. A synapse is a microscopic gap between two nerve cells in a neural pathway, where the impulse traveling in the first neuron initiates an impulse in the second neuron. The signals come into synapses are weighted and resulting quantities are summed. If the sum ≥ threshold, the neuron fires. An artificial neuron is intended to emulate the neuron through following basic components.

- *Inputs and outputs:* Act as sensory receptors for input and motor neurons for output as that of the brain.
- *Weighting factors:* Like the varying synaptic strengths of the biological neurons, weights are adaptive coefficients as a measure of the connection strength, determining the intensity of the input signals.
- *Activation/Transfer functions:* Determine the "firing rate" of a neuron in response to an input or stimulus. Introduce non-linearities in complex functional mappings between the inputs and response variable. Sigmoid (σ) and tanh are classic activation functions used in NN.

NN Learning

NN learning is through aggregation of a variable length of causal chains of neural computations seeking to approximate a certain simulation task through linear/nonlinear modulation of the activation of the neurons across the architecture. By changing the distribution of weights, each stimulation redistributes the neural weights a little bit until the error is below a defined lower bound. Here are the keys of the learning algorithm.

- *Supervised learning:* Analyzes the training data, which consists of input-output pairs, and produces an inferred function, which can be used for mapping new instances in forecast.
- *Optimization:* The error function can be determined, after initial weights are defined and a forward pass performed to generate the initial prediction. To find the weights that generate the smallest error to the known true value, backpropagation with gradient descent is commonly used for its computational efficiency, especially when dealing with large neural networks.
- *Overfitting:* Happens when the neural network is good at learning its training set, but is not able to generalize its predictions to additional, unseen examples. It can be avoid by methods such as regularization, early stopping, tuning performance ratio, retraining *etc.*