Intro to Neural Networks with Keras: An Application in Hydrologic Modeling

Kathy Breen

February 15, 2019

Artificial Intelligence

Artificial Intelligence: Any method that teaches computers to mimic human intelligence or behavior.

EX: IBM Deep Blue, the chess-playing computer (1996)

Machine Learning: Subset of AI algorithms largely using statistical methods to learn patterns in data.

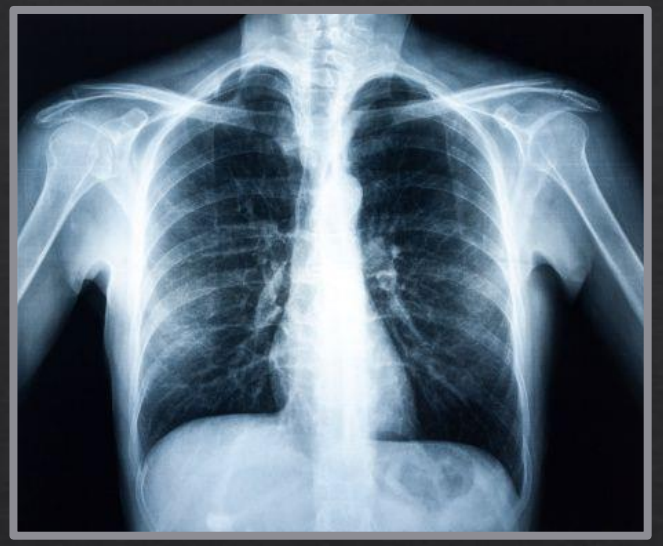
EX: Supervised learning (RF, K-means)

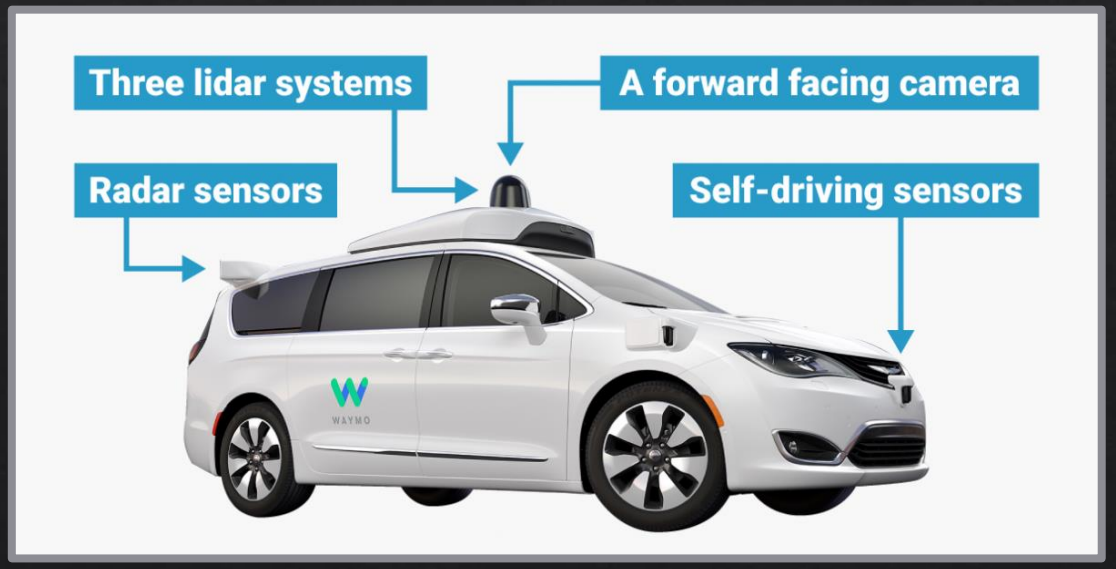
Deep Learning: Subset of machine learning algorithms using neural networks to replicate cause and effect relationships represented by data.

EX: Multilayer Perceptron

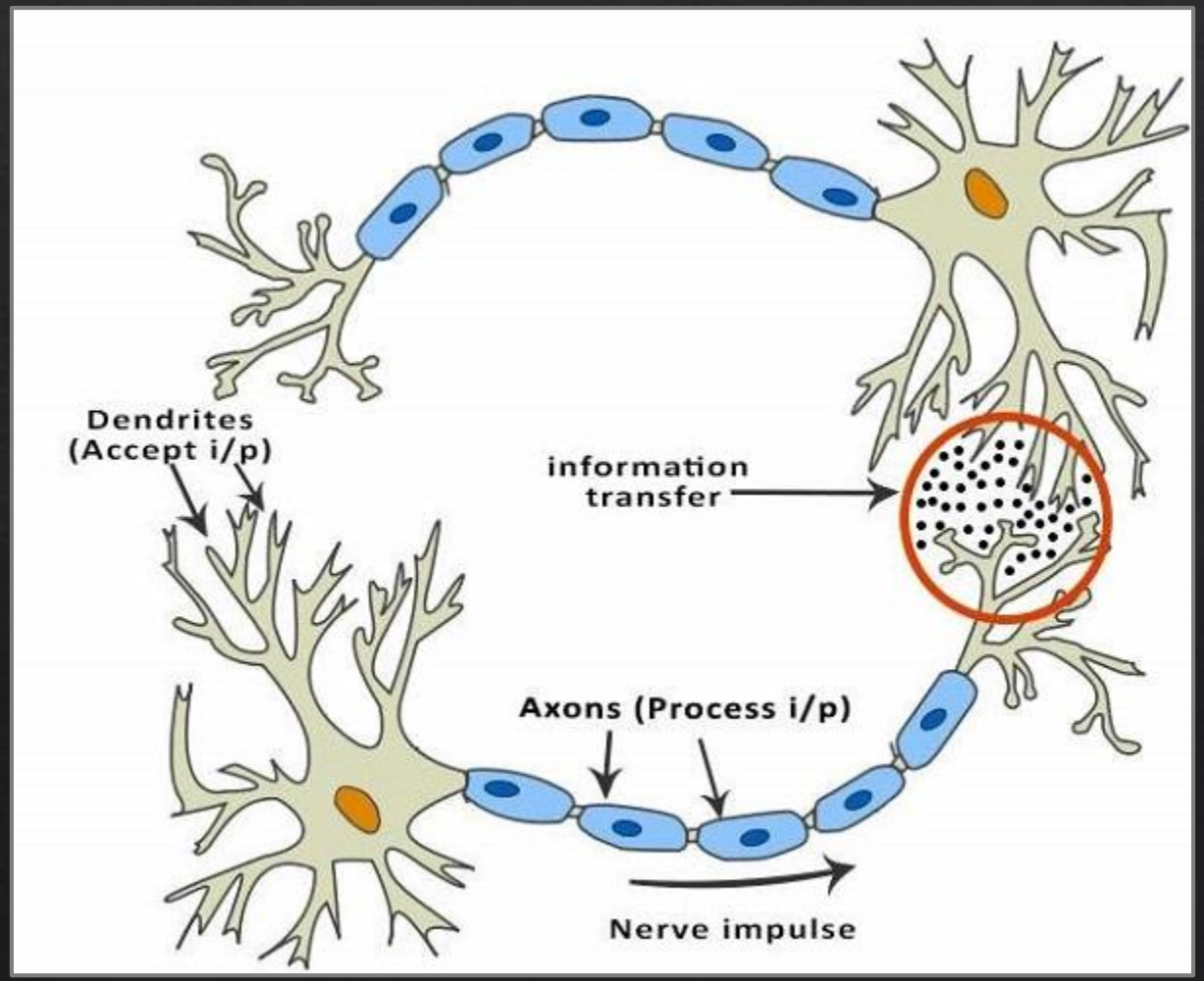
What is Deep Learning?

- Data-driven models
- Close-to-human performance
- Advanced pattern recognition
- Decrease computational time
- Decrease user/model bias

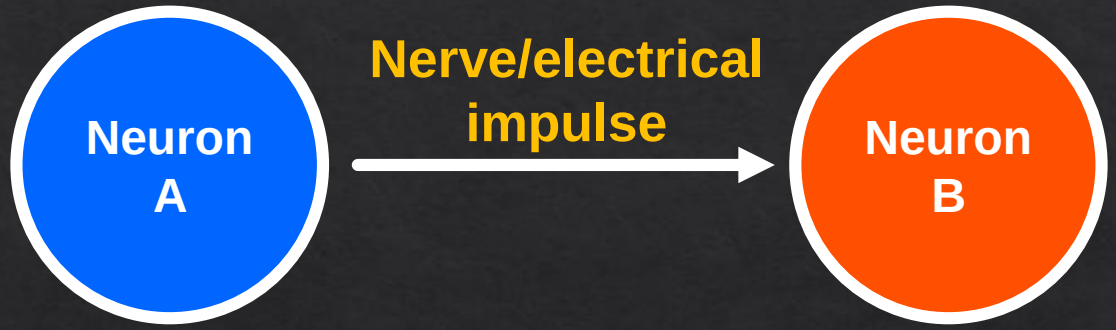




Artificial Neural Networks (ANN)



Anatomy of the Brain:



Artificial Neural Network:

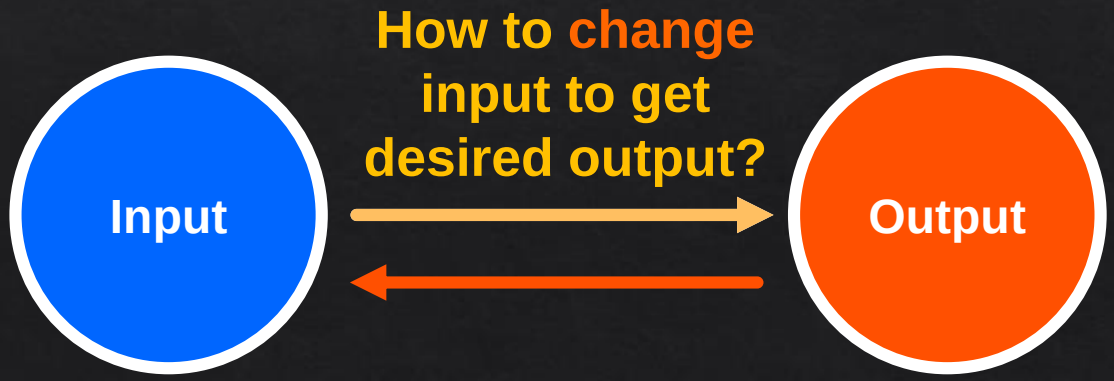


Table 1: Data generated using the "known" function.

X	Y
1	3
2	5
3	7
4	9
5	11

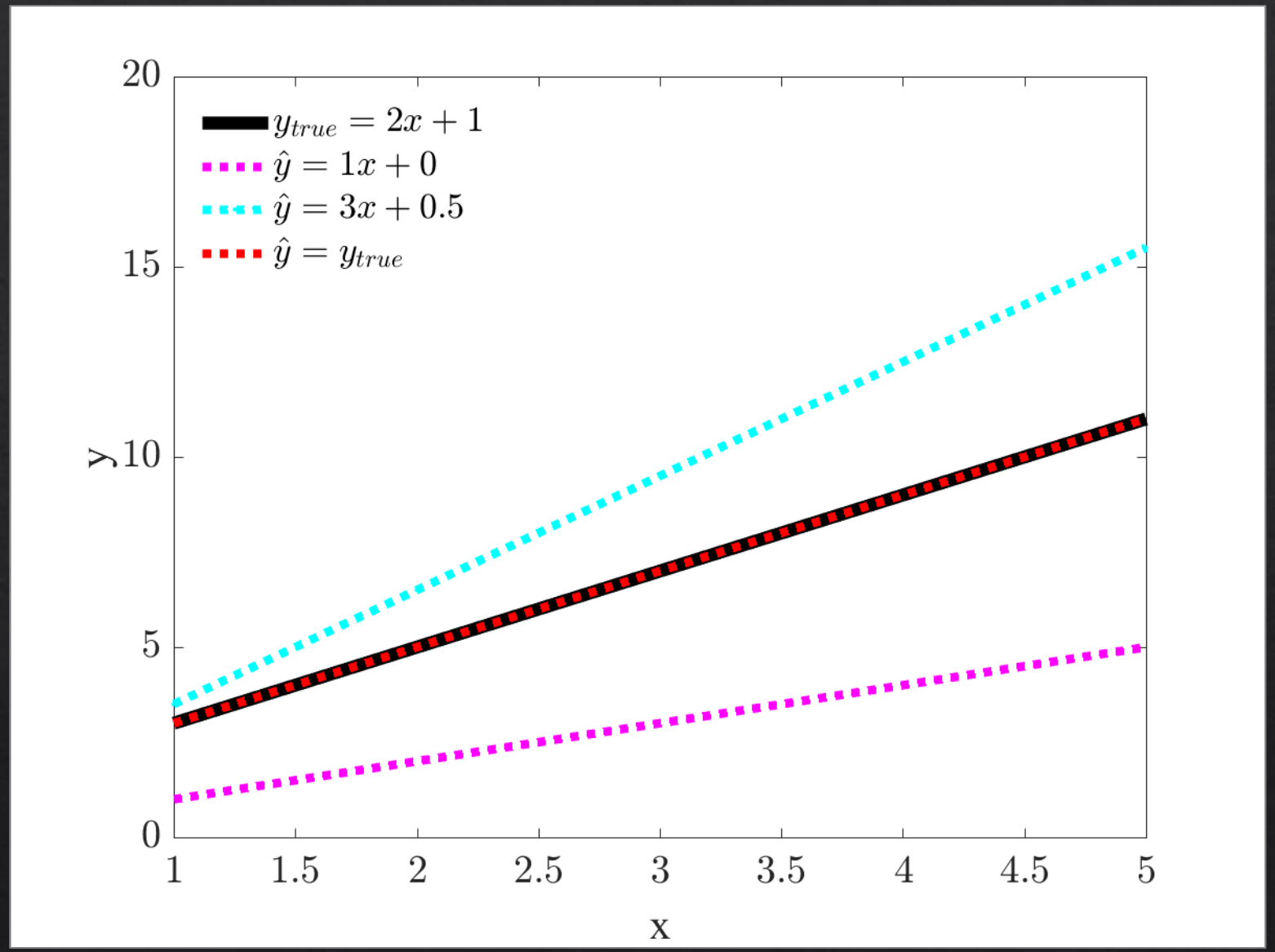
KNOWN:

$$y = 2x + 1$$

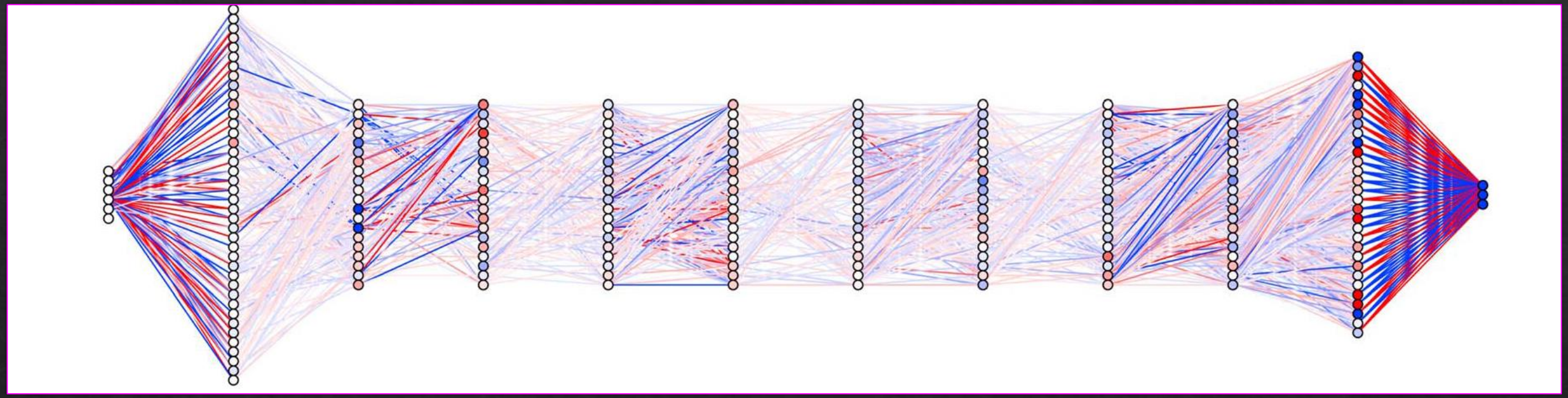
TASK: Given the data in Table 1, approximate the "known" function by altering the weight and bias applied to x to get the estimate $\hat{y}$ such that the difference between y and $\hat{y}$ is minimized.

$$\hat{y} = Wx + b$$

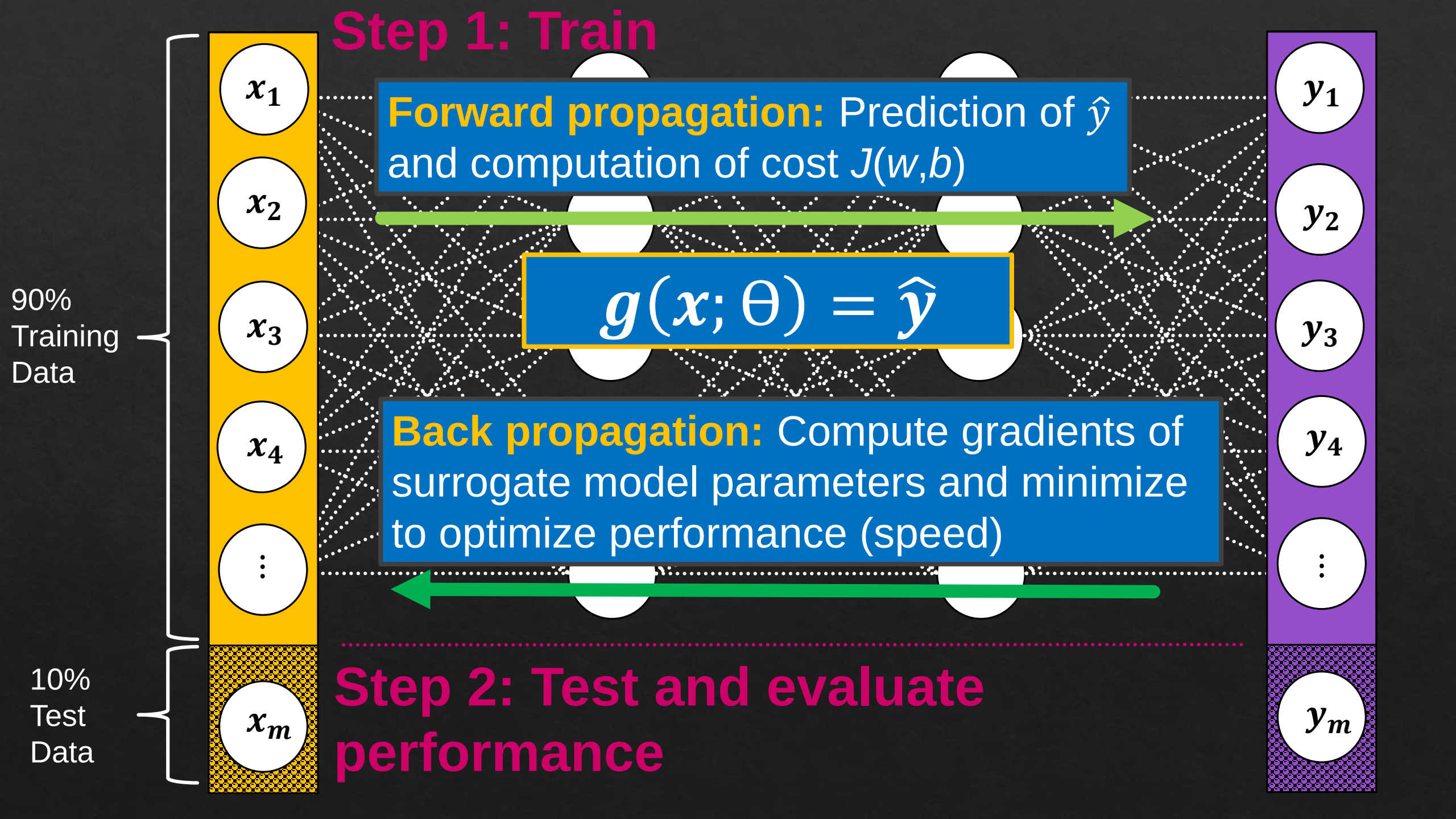
Epoch	W	b	RMSE
1	1	0	4.2426
2	3	0.5	2.8723
3	2	1	0



ANN Example



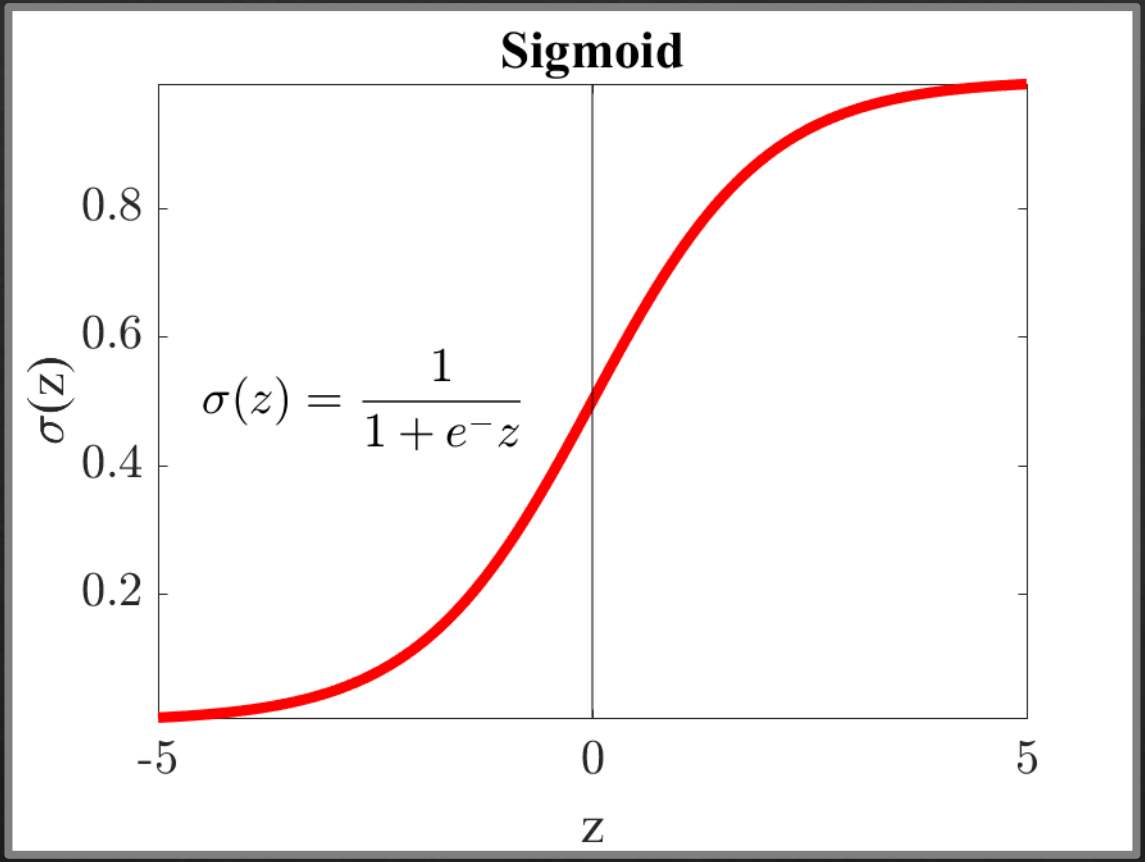
- Lines indicate connection weights between neurons (red < 0, **blue > 0** and thickness illustrates magnitude).
- Circles represent individual neurons, with color and intensity indicating the sign and magnitude of the bias associated with each neuron.

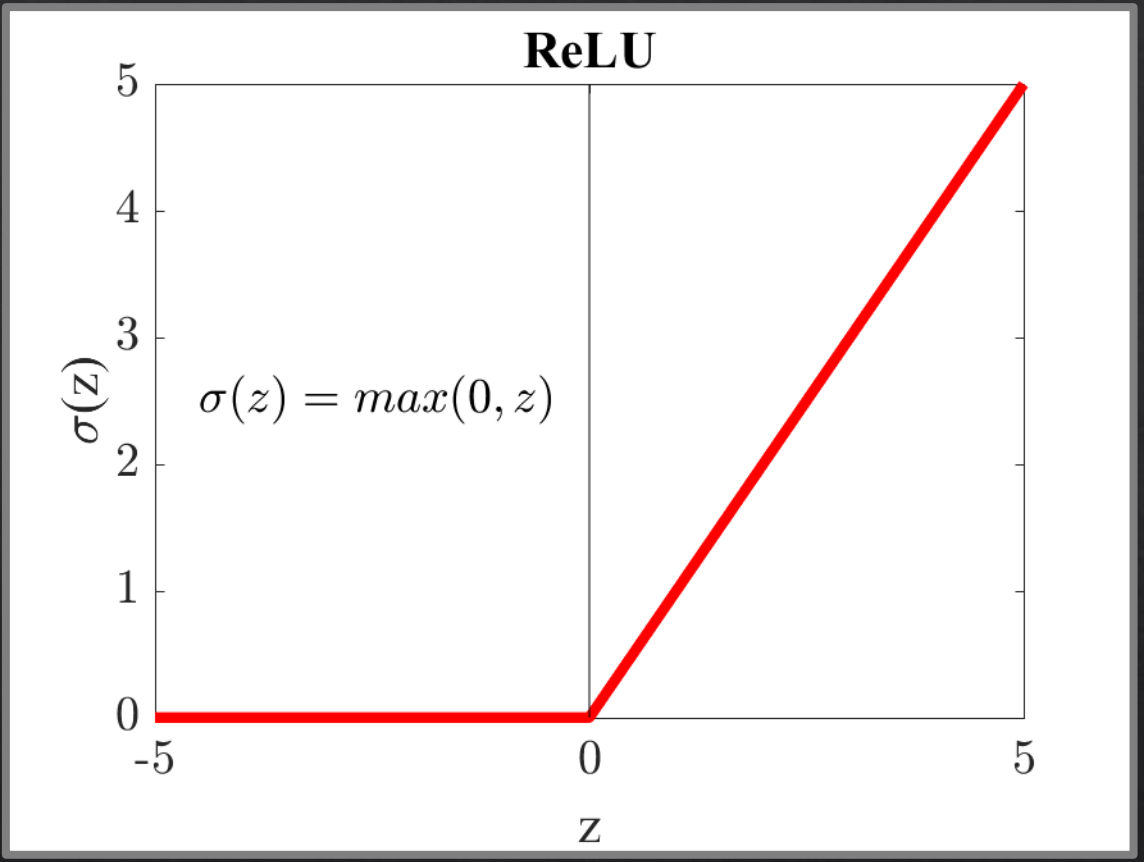

Step 1: Train
Forward propagation: Prediction of $\hat{y}$ and computation of cost $J(w,b)$
$g(x;\theta) = \hat{y}$
Back propagation: Compute gradients of surrogate model parameters and minimize to optimize performance (speed)
90% Training Data
10% Test Data
x_1
x_2
x_3
x_4
x_m
y_1
y_2
y_3
y_4
y_m
Step 2: Test and evaluate performance

$$a_n^{(\ell)} = \sigma\left(\sum_{k=1}^{N_{\ell-1}} w_{k,n}^{(\ell)} a_k^{(\ell-1)} + b_n^{(\ell)}\right)$$

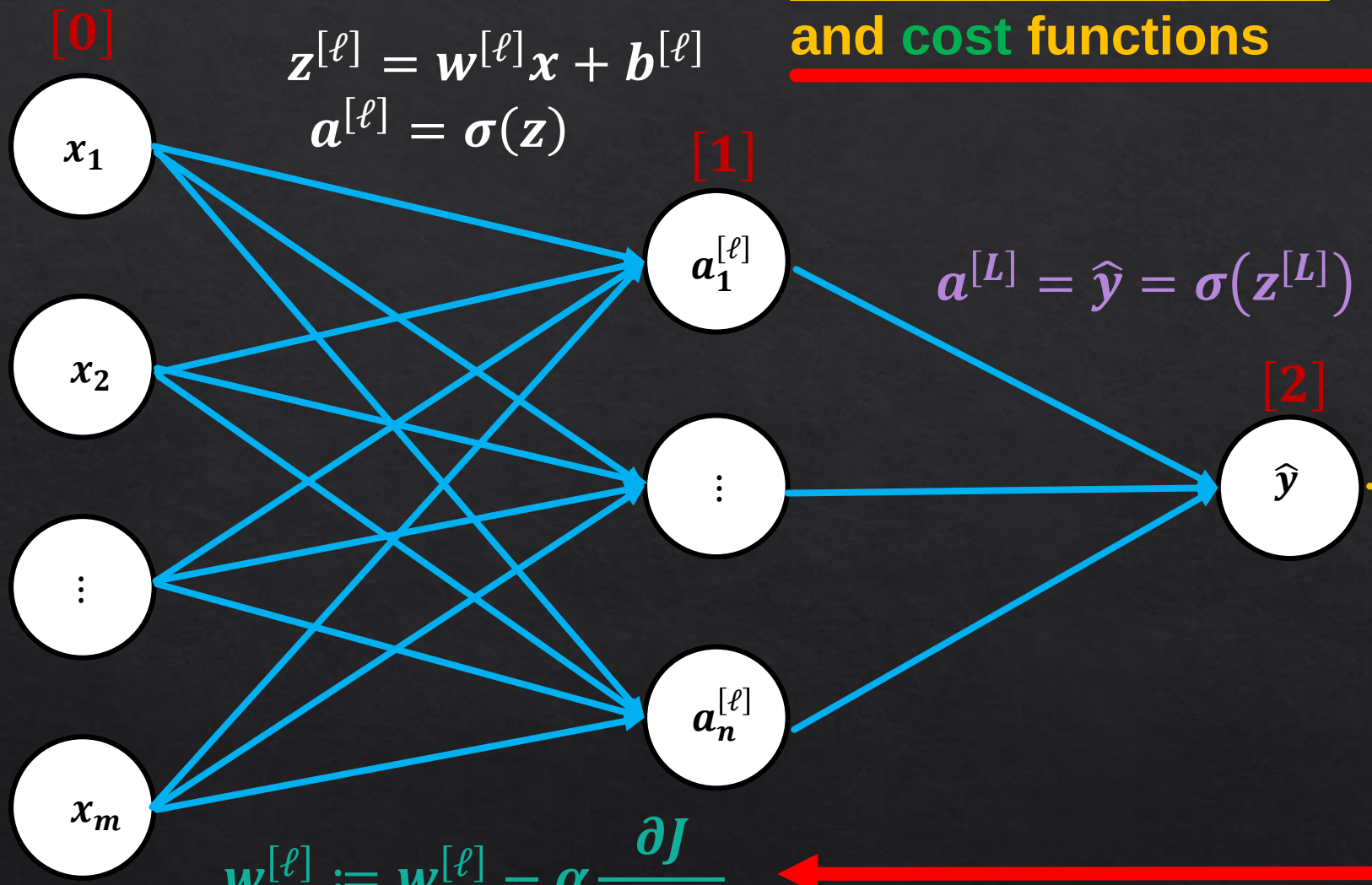
- σ is the activation function
- $a_k^{(\ell-1)}$ is the activation of node *k* in layer *ℓ-1*
- $w_{k,n}^{(\ell)}$ is the weight from node *k* in layer $\ell - 1$ acting on node *n* in layer *ℓ*
- $b_n^{(\ell)}$ is the bias on the activation of node *n* in layer *ℓ*

Activation Functions





- The Rectified Linear Unit (ReLU) – a major breakthrough!
- Not intuitive because it is nonlinear.
- Practically, it means there is a threshold for activation followed by a linear response.



Forward propagation: estimate $\hat{y}$, compute loss and cost functions

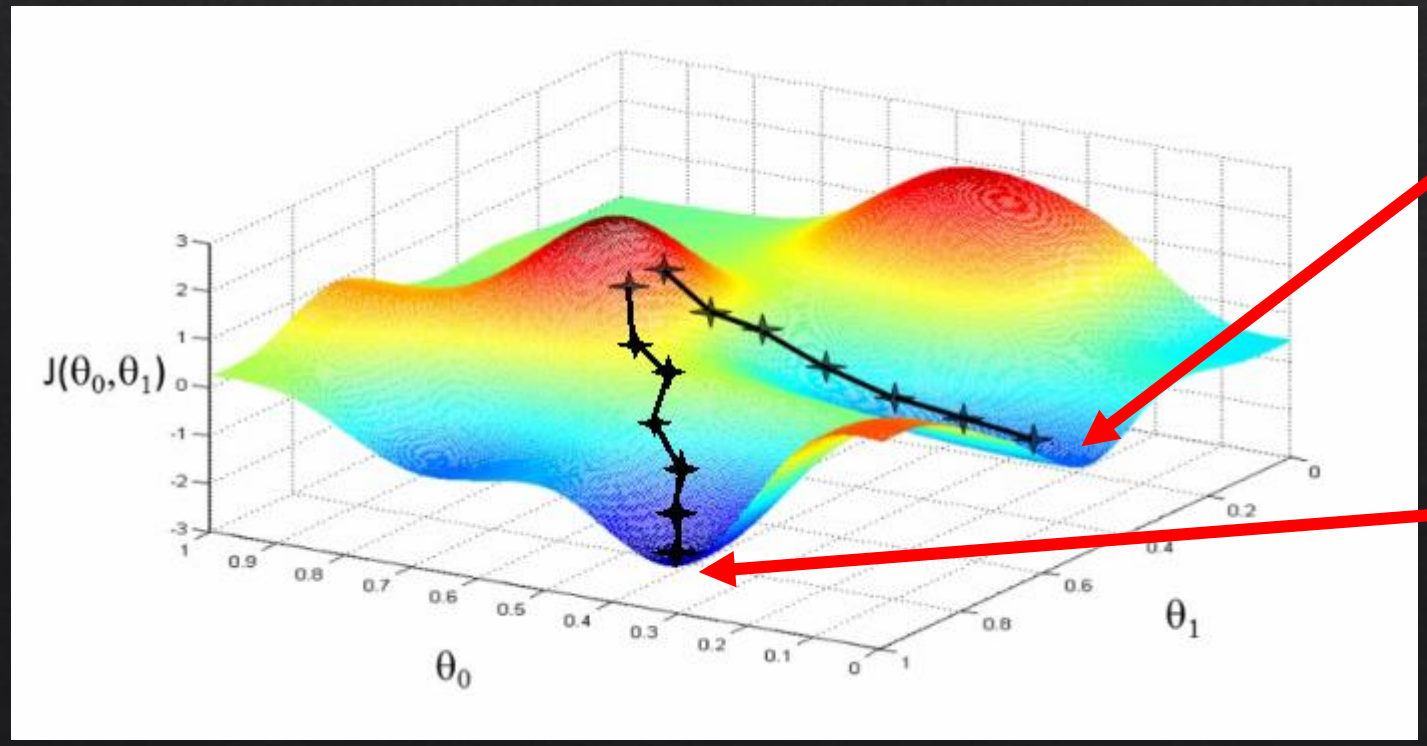
$$w^{[\ell]} := w^{[\ell]} - \alpha \frac{\partial J}{\partial w^{[\ell]}}$$

$$b^{[\ell]} := b^{[\ell]} - \alpha \frac{\partial J}{\partial b^{[\ell]}}$$

Back propagation: compute gradients to find parameter values which minimize $J(w, b)$, update weights and biases

Minimizing the Cost Function

Objective: Iteratively minimize the cost (difference between NN predictions and “true” values) by making small adjustments to weights, biases, learning rate, etc. such that the global minimum is reached.



Local minima reached – a better solution exists!

Global minimum - “best” predictions made with the current model configuration!

Image credit: https://medium.com/@prashant81995/introducing-gradient-descent-minimizing-the-cost-function-65fff3de8ac6

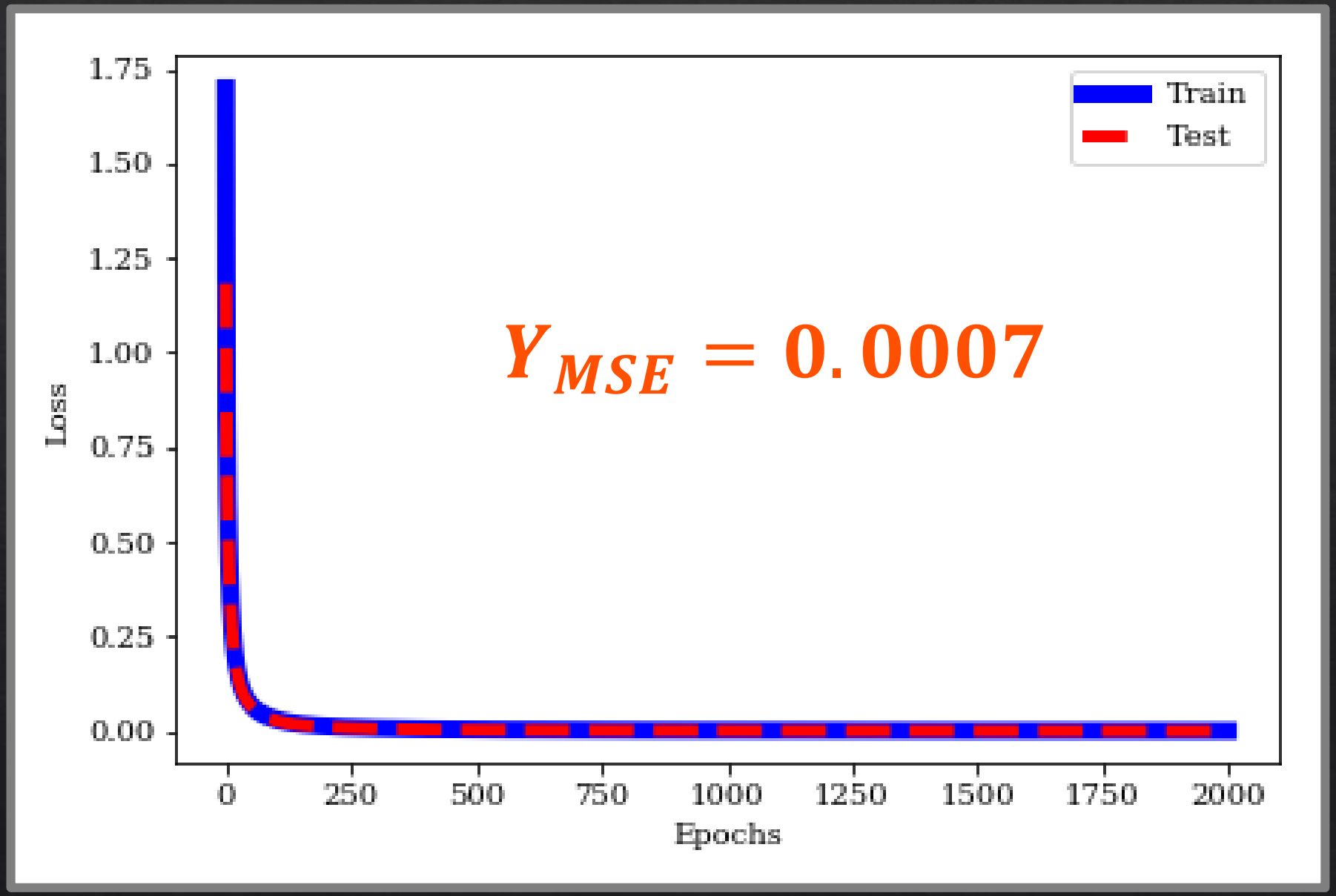
Keras Example $y = x^2$

```
# Import Packages
from keras.layers import Input, Dense, Dropout
from keras.models import Model
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, CSVLogger
from sklearn.metrics import mean_squared_error
import numpy as np

# Create some simple data for a known function (y = x^2)

# generate 100,000 samples (x dimension) with 1 feature each (y dimension)
X = np.random.randn(100000,1)
Y = np.square(X)
X_Nfeatures = X.shape[1]
Y_Nfeatures = Y.shape[1]

# partition X and Y into training (90%) and test (10%) sets
X_train = X[:90000,:]
X_test = X[90000:,:]
Y_train = Y[:90000,:]
Y_test = Y[90000:,:]
```

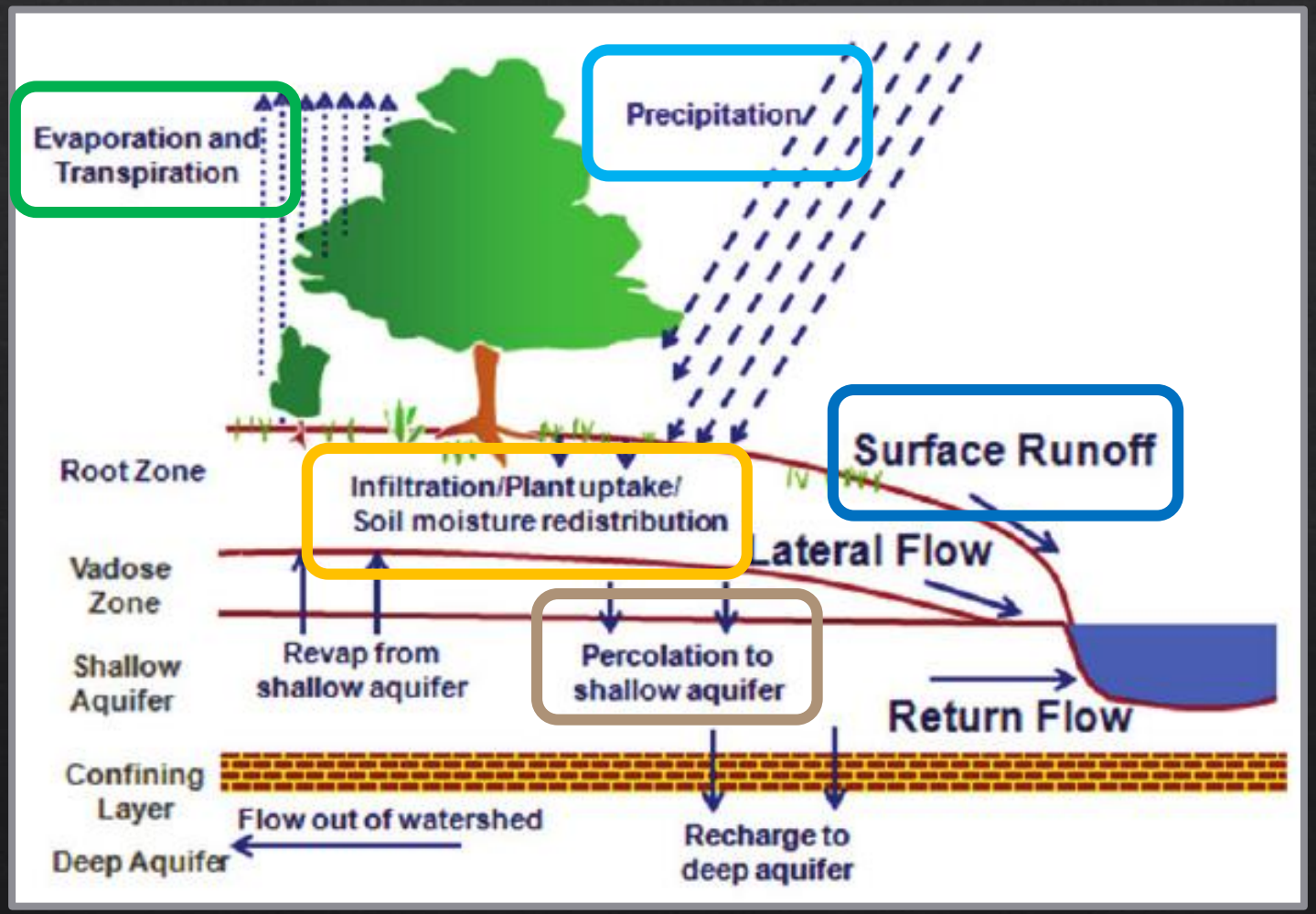
```
# HYPERPARAMETERS
epochs = 2000
batch_size = 5000
do = 0.2
N_nodes = 64

# create input layer.....
main_input = Input(shape=(X_Nfeatures),
                   dtype='float',
                   batch_shape=(batch_size,X_Nfeatures),
                   name='main_input'
                   )
# create hidden layer.....
hidden_layer1 = Dense(N_nodes, activation='relu',
name='hidden_layer1')(main_input)
Dropout(do)(hidden_layer1)

# create output layer
main_output = Dense(Y_Nfeatures,
name='main_output')(hidden_layer1)  # default
activation is linear
```

```
# feed datasets into model for training
model = Model(inputs=[main_input],
              outputs=[main_output]
              )

# compile the model with desired configuration
model.compile(optimizer='adam')

# train the model, and store training information
in the history object
history = model.fit([X_train],[Y_train],
                    epochs=epochs,
                    batch_size = batch_size,
                    validation_data=(X_test, Y_test) )

# evaluate the trained model on the test data set
X_pred = np.random.randn(100000,1)
Y_pred = np.square(X_pred)
predict =
model.predict([X_pred],batch_size=batch_size)
Y_mse = mean_squared_error(predict,Y_pred)
```

Keras Example: Results



Application: Predict Soil Water Flux



$$\Delta Soil\ Moisture = Precipitation - Runoff - Evapotranspiration - Groundwater\ Percolation$$

Physics vs. Data-driven Models

Physics-driven

- $Soil\ Moisture = Precipitation - Runoff - Evapotranspiration - Groundwater\ Percolation$
- Equations developed to simulate physical processes
- Models must be calibrated/validated to be of any use, therefore investigators must have access to C/V data
- Sources of error:
  - Assumptions made to simplify equations
  - Unable to fully capture the complexity of a physical system

Data-driven

- Model trained to recognize relationships between observed inputs (**precipitation**, **runoff**, **ET**, **gw perco**) and desired output (**soil moisture**)
- Physical relationships that cannot be measured/approx. less likely to be ignored.
- Sources of error:
  - The trained model is only as accurate as the data used for training.
  - Cannot adapt to predict new scenarios outside the scope of training data.

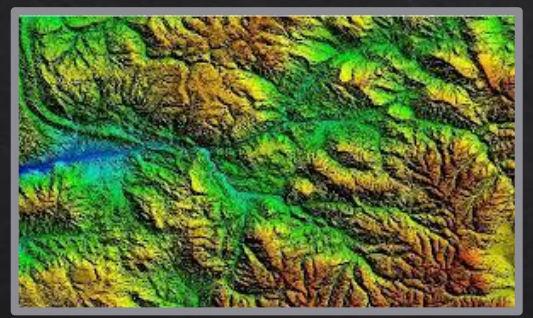
Purpose – Big Picture

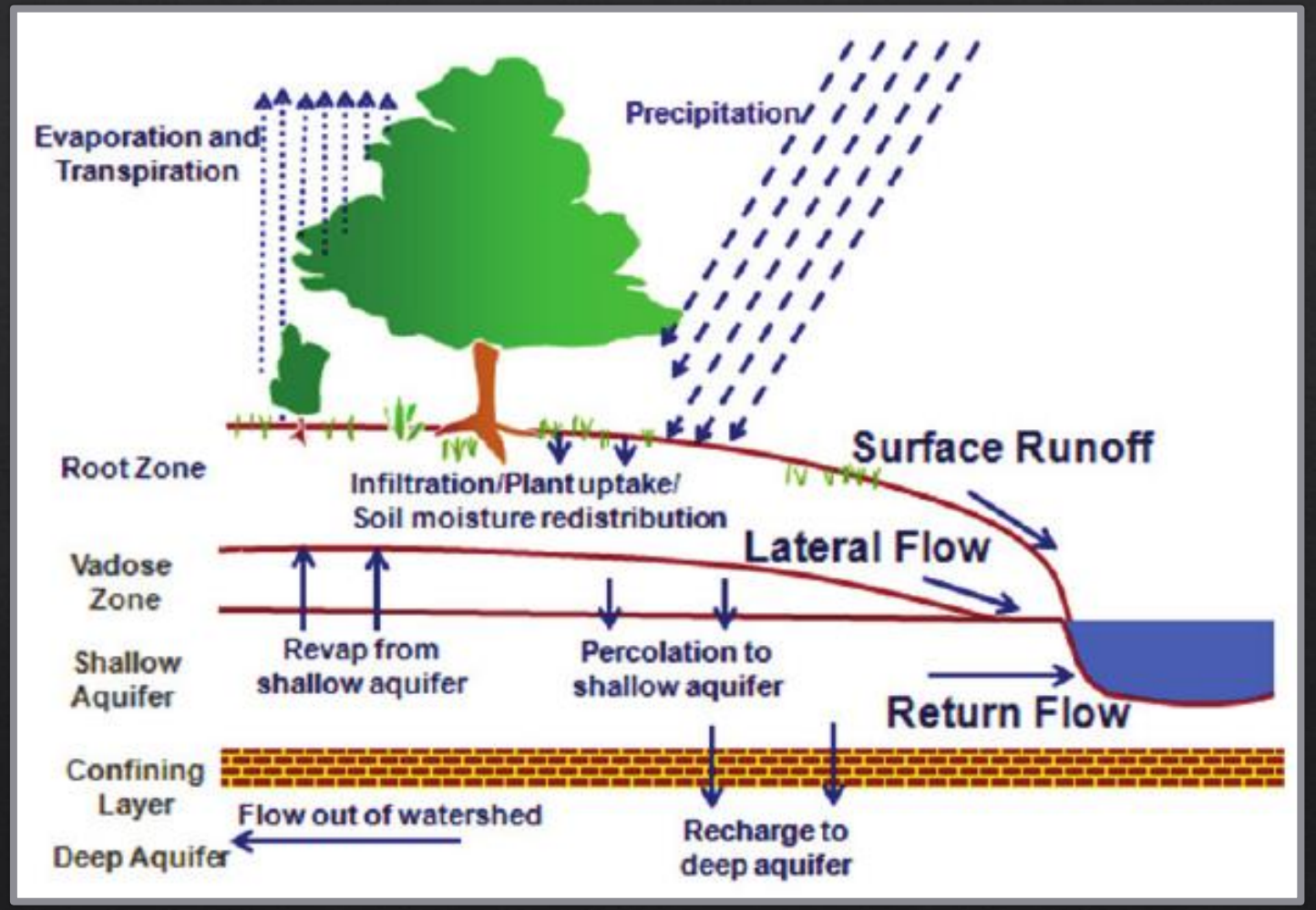
- Need:
  - Accurate, near-real-time, high-resolution estimates for key metrics (i.e. soil moisture) to assess hydrologic activity.
  - Computationally efficient, lightweight tools to analyze data and make predictions in real time.
- Soil moisture can be used as a metric to assess:
  - Soil health – does a given soil contain sufficient moisture to sustain plant and animal life?
  - Risk potential – in the event of extreme weather, which areas are most sensitive to extreme water flux?

Research Objectives

- Estimate current soil-moisture conditions using a neural network surrogate model for SWAT+-lte (SPL) daily model inputs and outputs augmented with satellite datasets for leaf area index (LAI) and soil moisture (SM), respectively, and *in situ* soil moisture and weather data.
- Create a data product that is scalable and produces real-time estimates that can be applied in flood risk and growing season crop viability assessments.

SWAT




Evaporation and Transpiration
Precipitation
Root Zone
Infiltration/Plant uptake/
Soil moisture redistribution
Surface Runoff
Lateral Flow
Vadose Zone
Shallow Aquifer
Revap from shallow aquifer
Percolation to shallow aquifer
Return Flow
Confining Layer
Flow out of watershed
Deep Aquifer
Recharge to deep aquifer

SWAT: Nonlinear Mapping

Weather

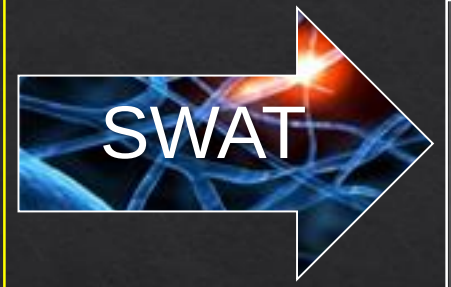
$$\begin{bmatrix} P^{(1)} & T^{(1)} & RH^{(1)} & S^{(1)} & WS^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ P^{(m)} & T^{(m)} & RH^{(m)} & S^{(m)} & WS^{(m)} \end{bmatrix}$$

Water flux

$$\begin{bmatrix} Q_{\text{surf}}^{(1)} & ET^{(1)} & Q_{gw}^{(1)} & w_{seep}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ Q_{surf}^{(m)} & ET^{(m)} & Q_{gw}^{(m)} & w_{seep}^{(m)} \end{bmatrix}$$

Biomass

$$\begin{bmatrix} LAI^{(1)} & N^{(1)} \\ \vdots & \vdots \\ LAI^{(m)} & N^{(m)} \end{bmatrix}$$



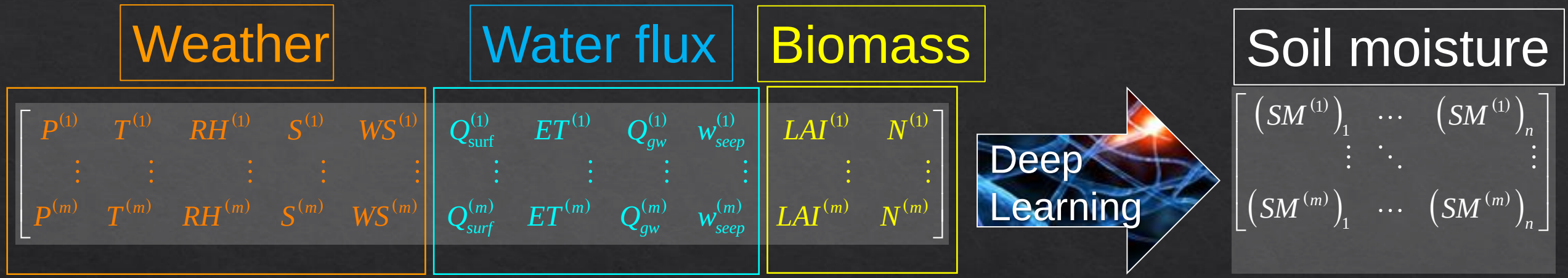
Soil moisture

$$\begin{bmatrix} \left(SM^{(1)}\right)_1 & \cdots & \left(SM^{(1)}\right)_n \\ \vdots & \ddots & \vdots \\ \left(SM^{(m)}\right)_1 & \cdots & \left(SM^{(m)}\right)_n \end{bmatrix}$$

Prediction

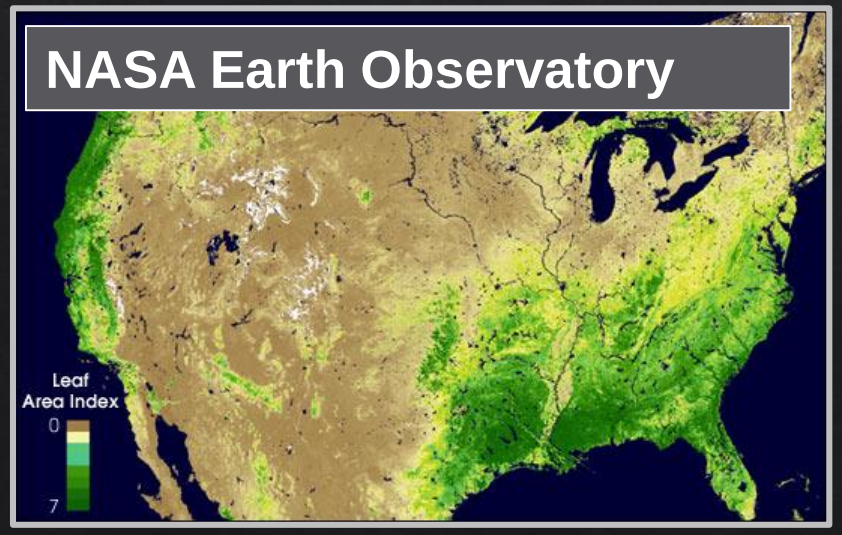
- Physically-based model
- Runs daily on a global grid
- Simplified to reduce computational expense
- Does not model nutrient cycling

Deep Learning



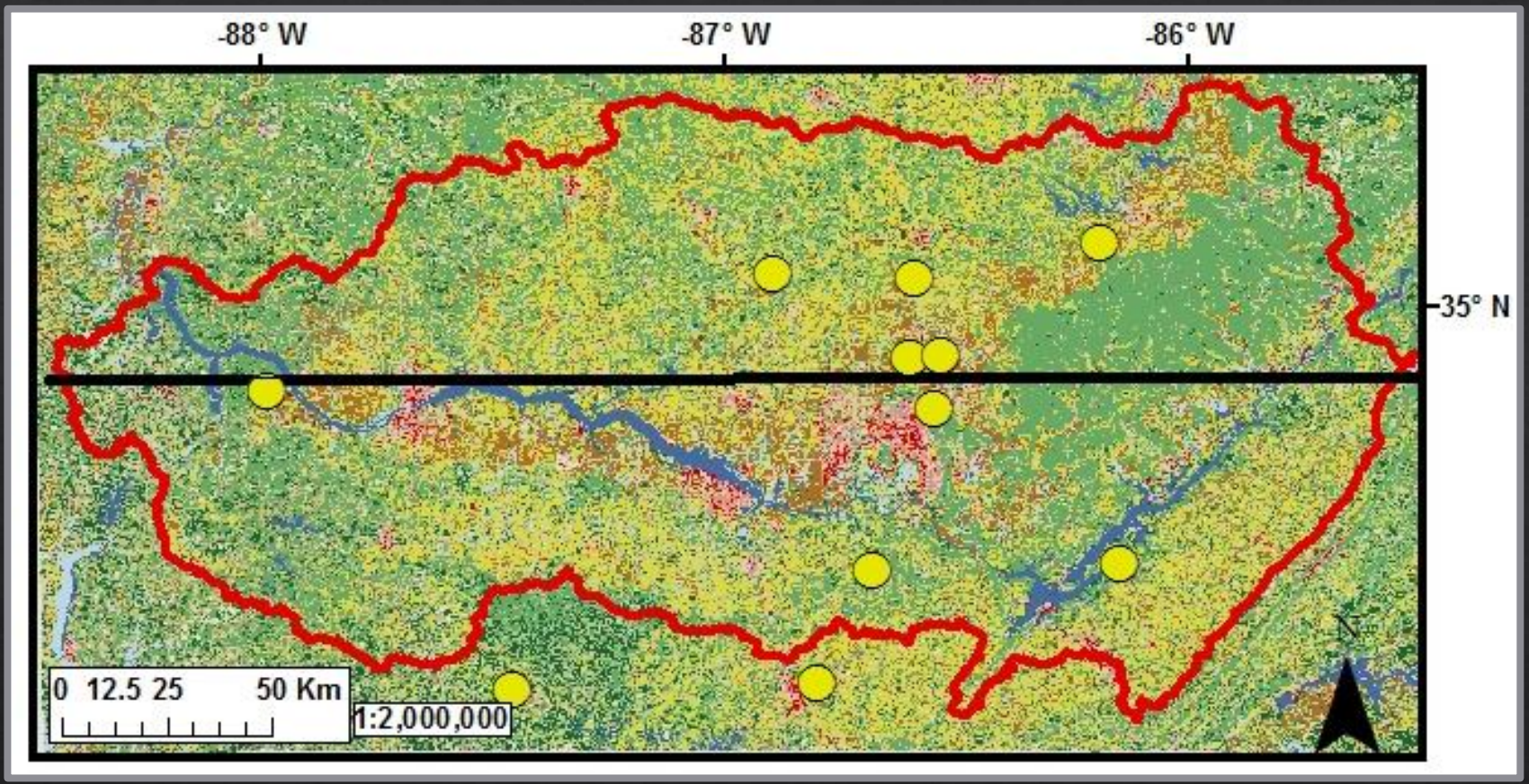
- **Accelerate soil-moisture forecasting by replacing SWAT with a deep-learning surrogate.** **Remotely Sensed**
- **However:**
  - **To train the deep learning model, many, many historical data sets are required,**
  - **Run SWAT many thousands of times to generate training data.**

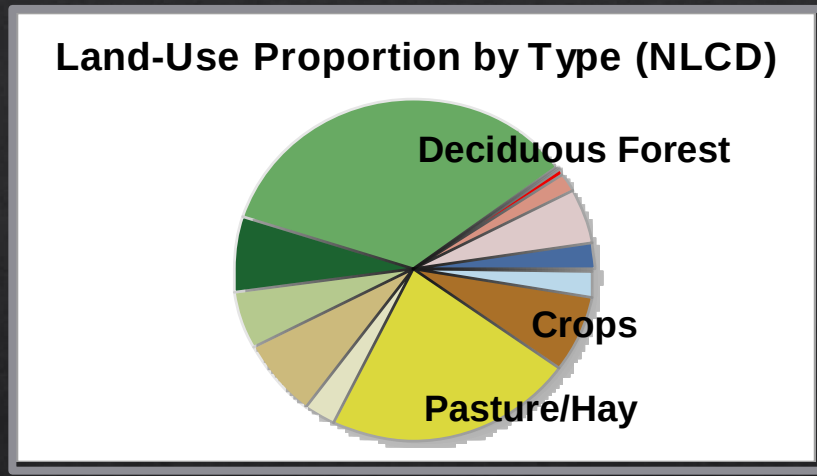
Remote Sensing

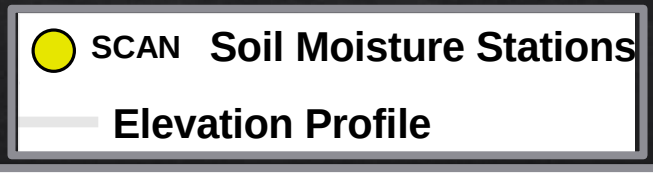


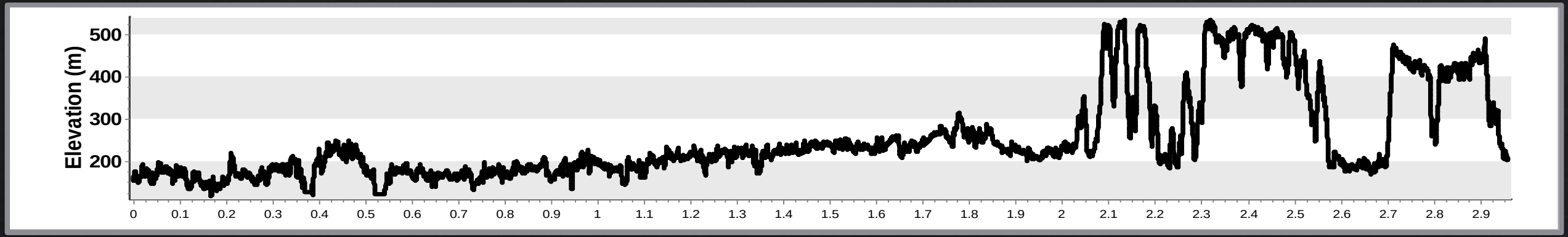
Satellite	Data Product	Spatial Resolution	Temporal Resolution	Data Latency
SMAP	Soil moisture	40 km	2 – 3 days	24 hours
MODIS	LAI	500 m	8 days	1 week

Target Area Selection









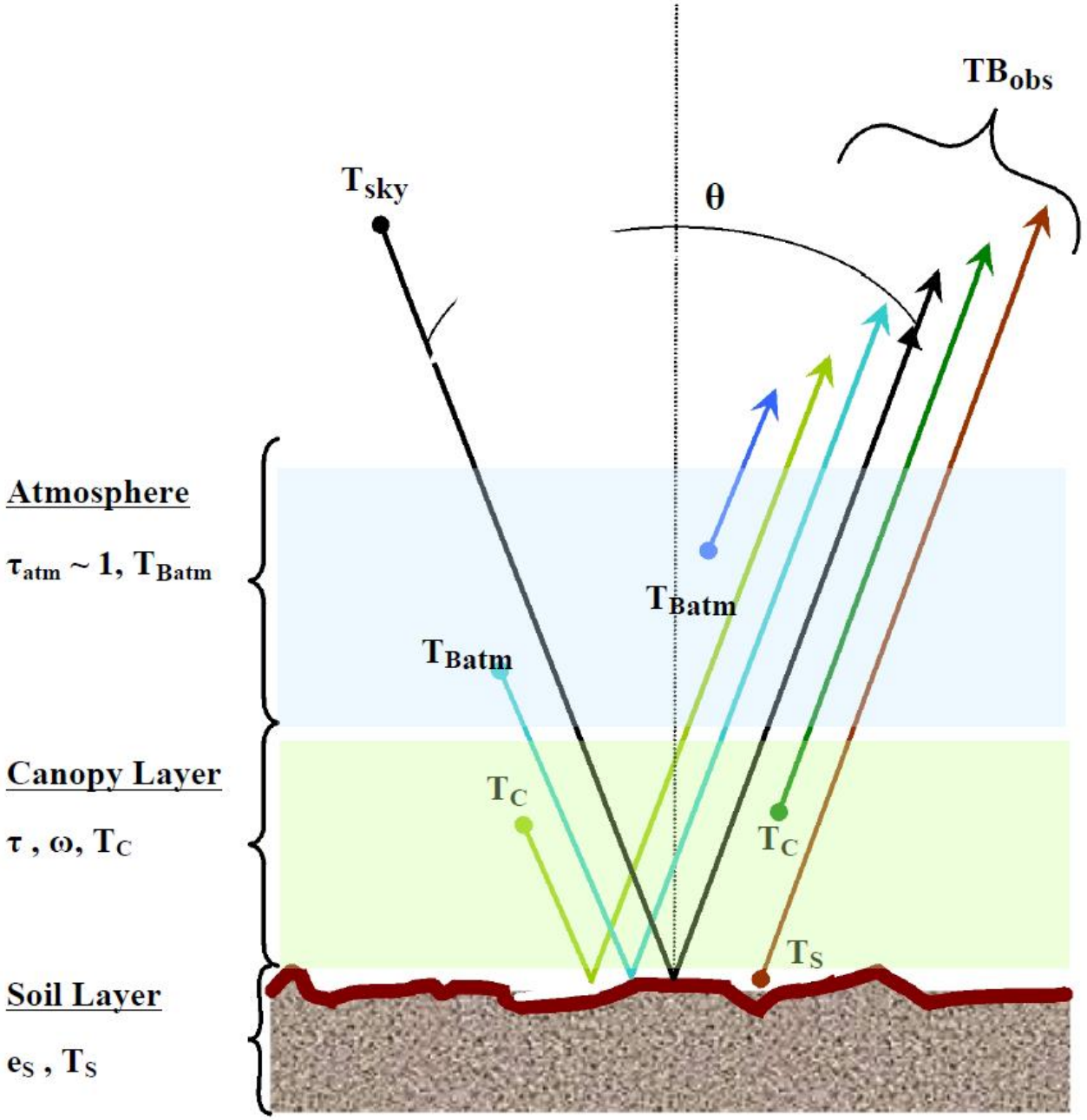


Figure 2. Contributions to the observed brightness temperature T_B from orbit [from SMOS ATBD, ref. 12] .

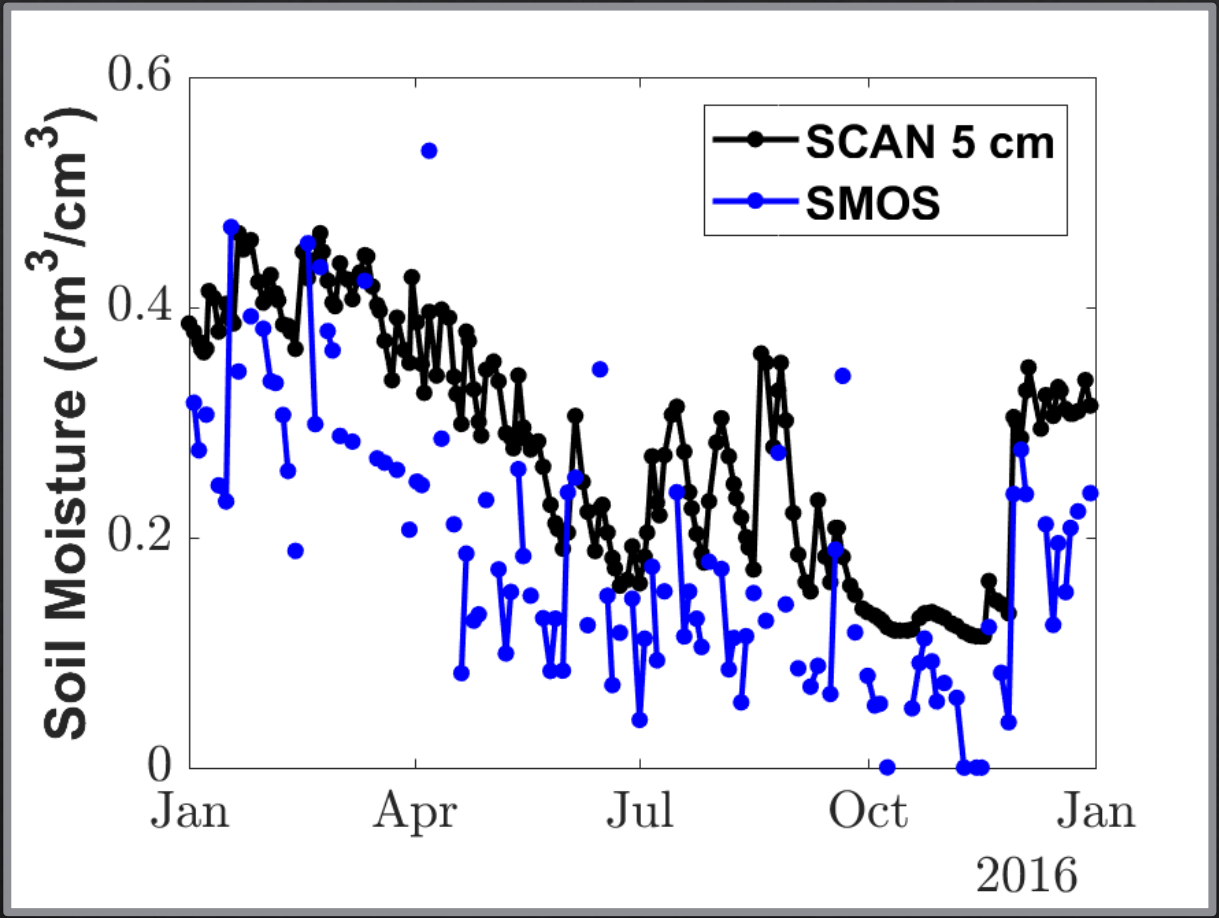
$$T_B = f(emissivity)$$

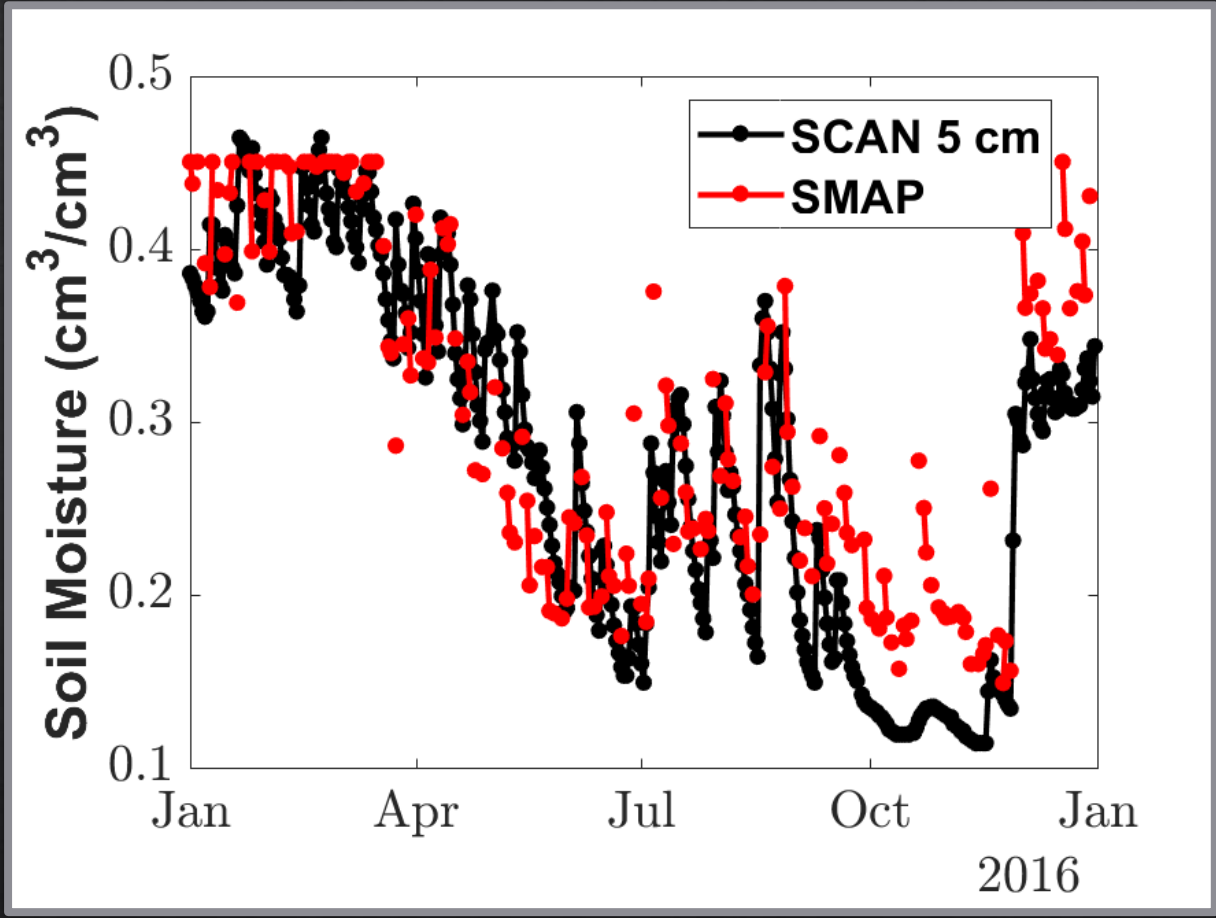
$$\varepsilon = g(T_{B_{soil}})$$

$$SM = h(\varepsilon)$$

O'Neill et al. (2015)

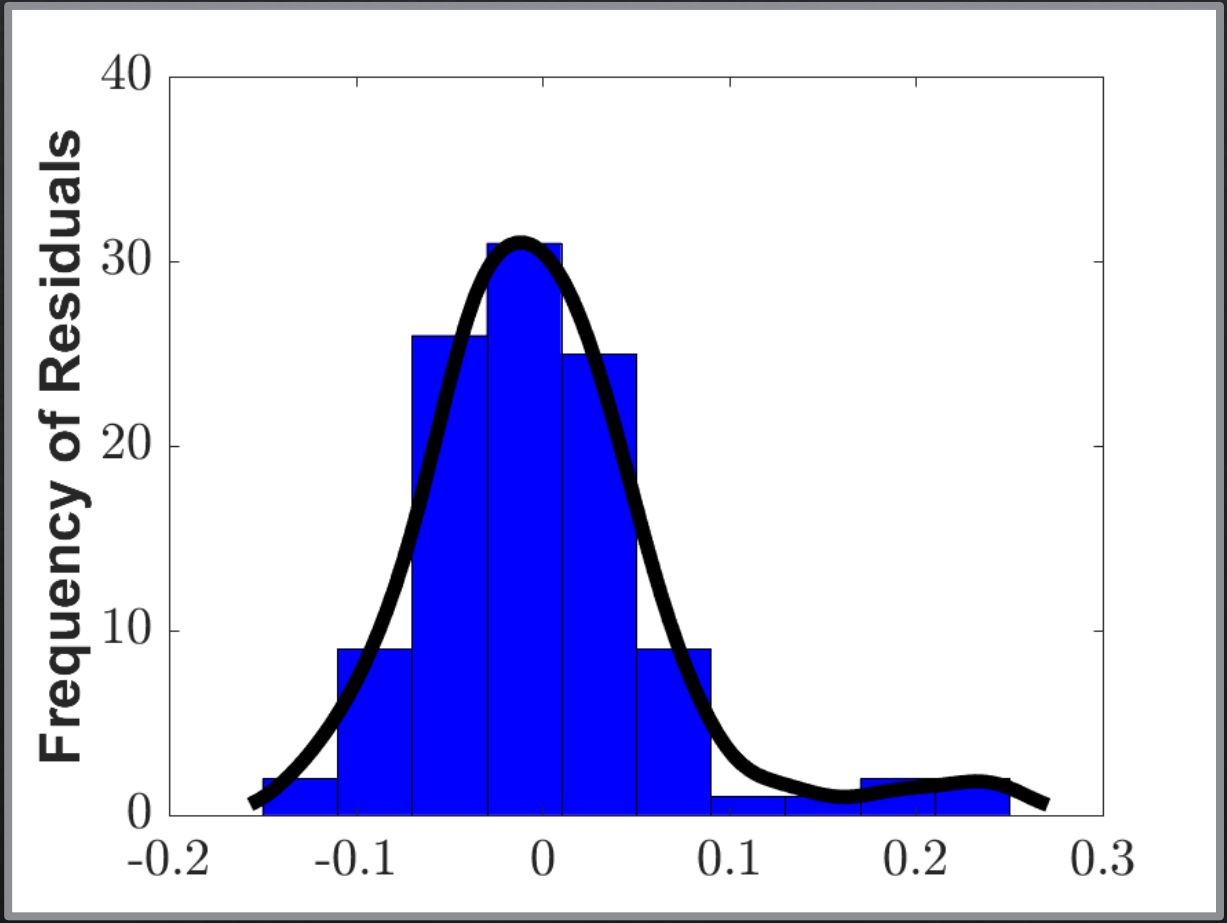
In situ Soil Moisture Comparison with Satellite Data



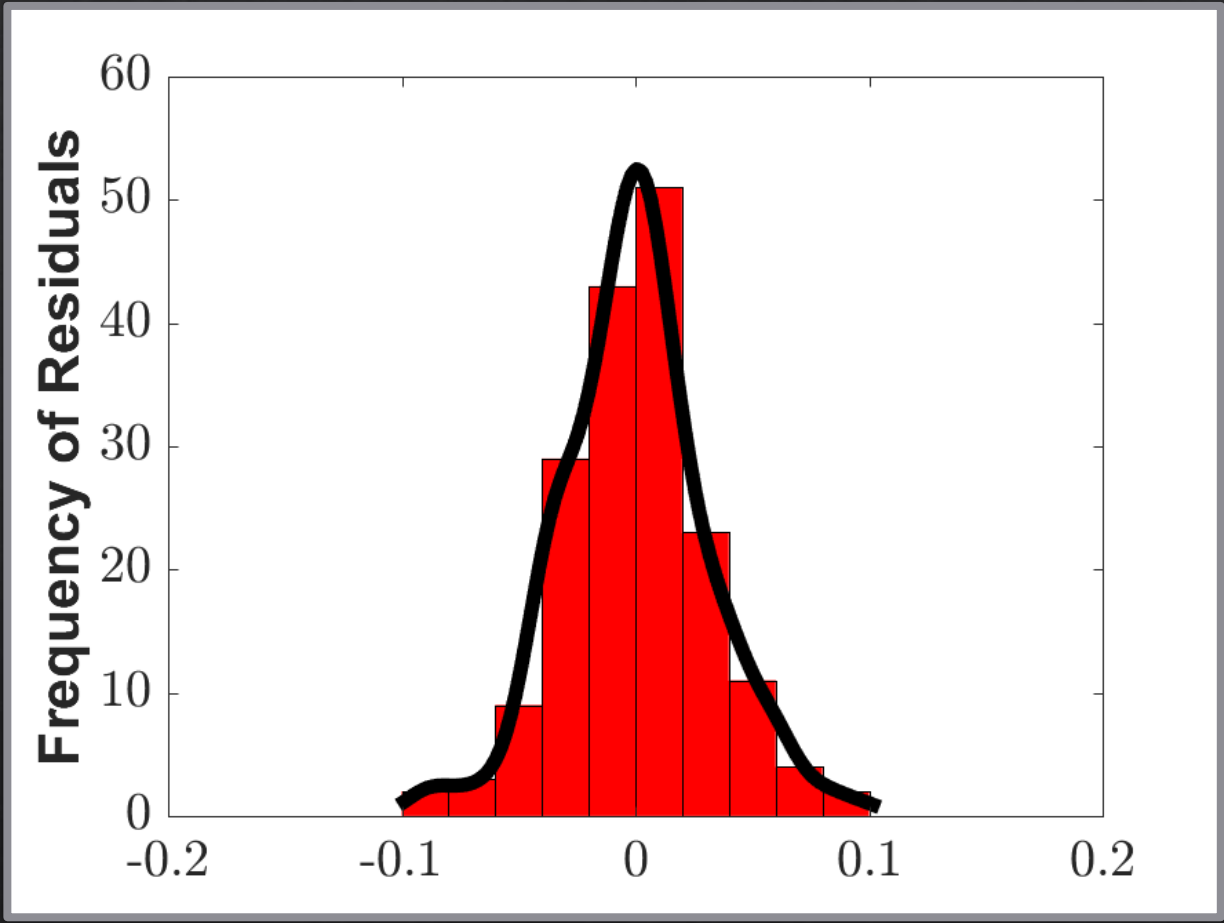


While both satellites follow similar trends, SMOS has a "dry bias" relative to SMAP and SCAN data. Neither satellite picks up seasonal changes, specifically at the onset, peak, and end of the growing season. Time series represent spatial averages for target area.

Histograms of Residuals (Observed – Least Squares Estimate)

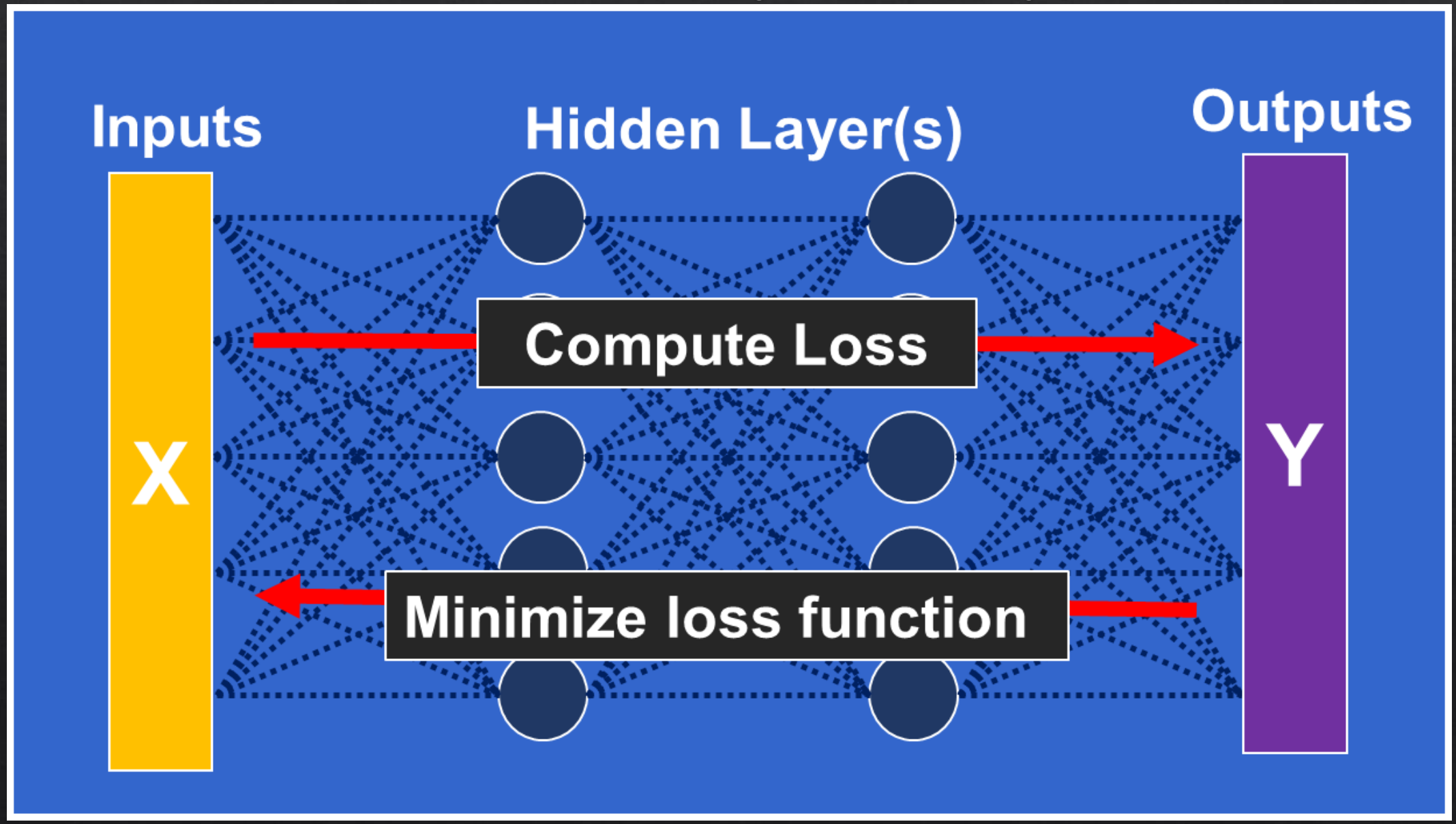


SMOS vs. SCAN

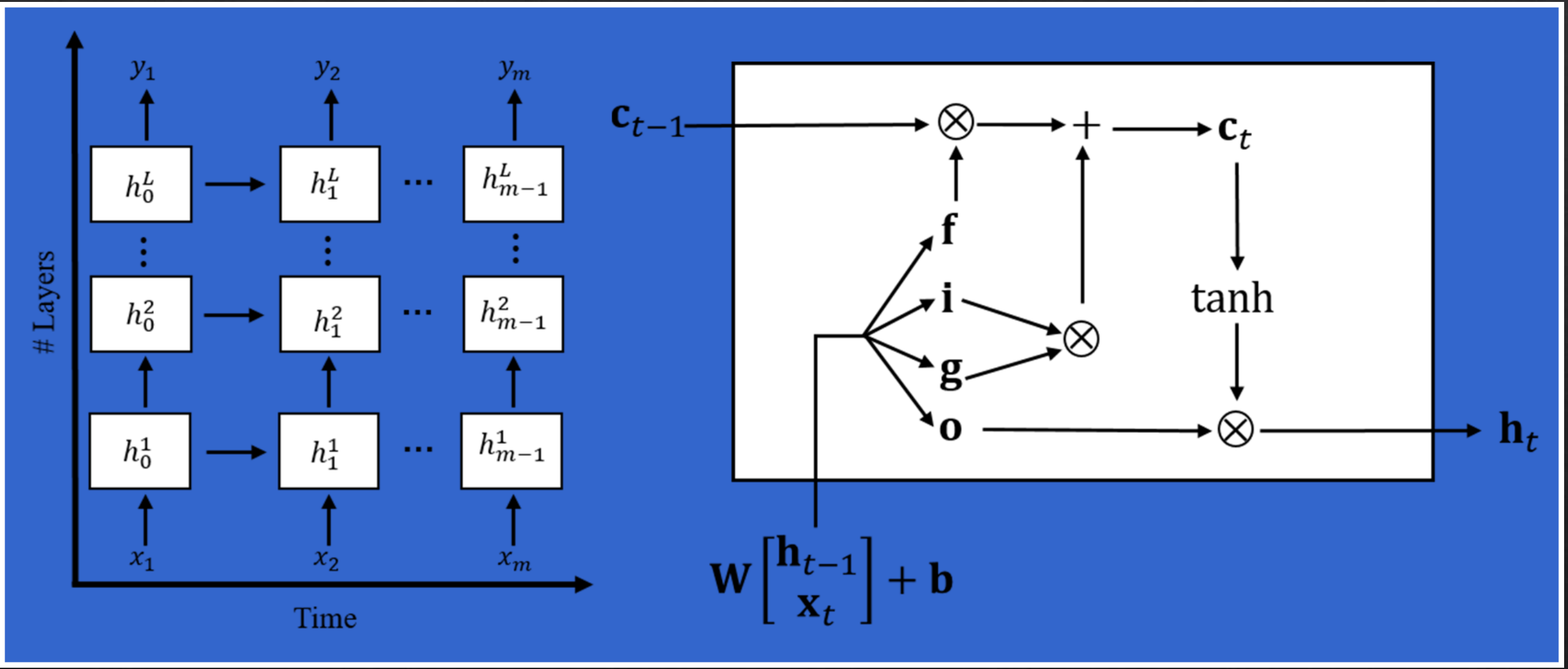


SMAP vs. SCAN

DLM Architectures by Data Type: Static



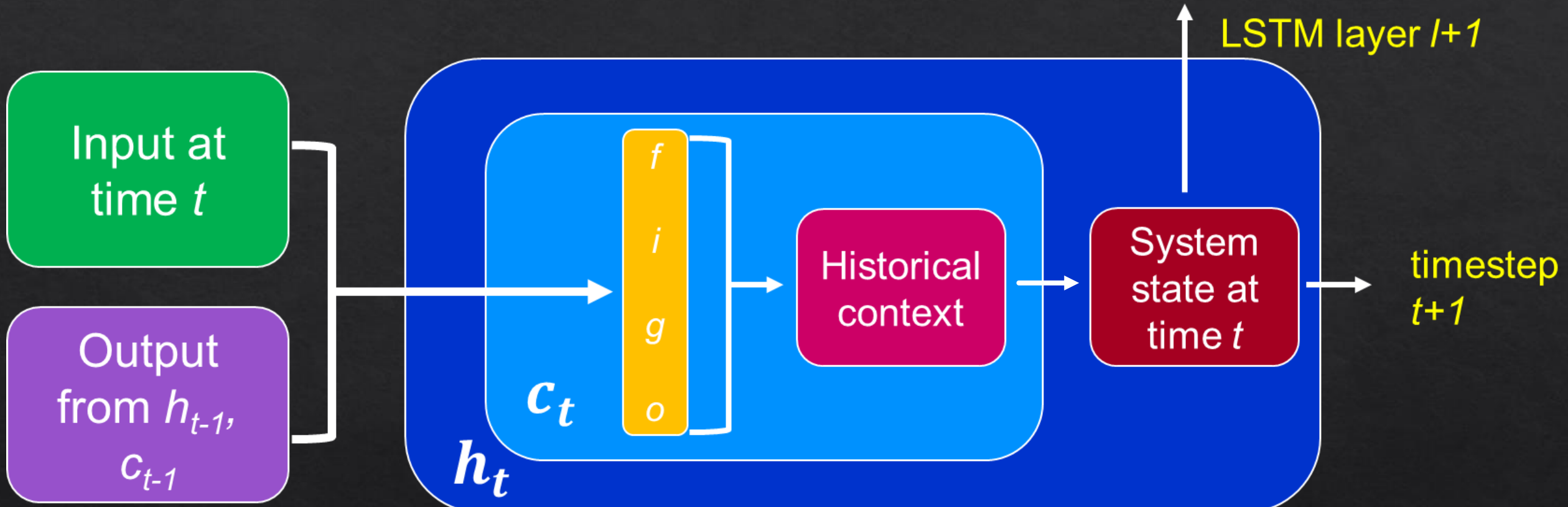
DLM Architectures by Data Type: Dynamic



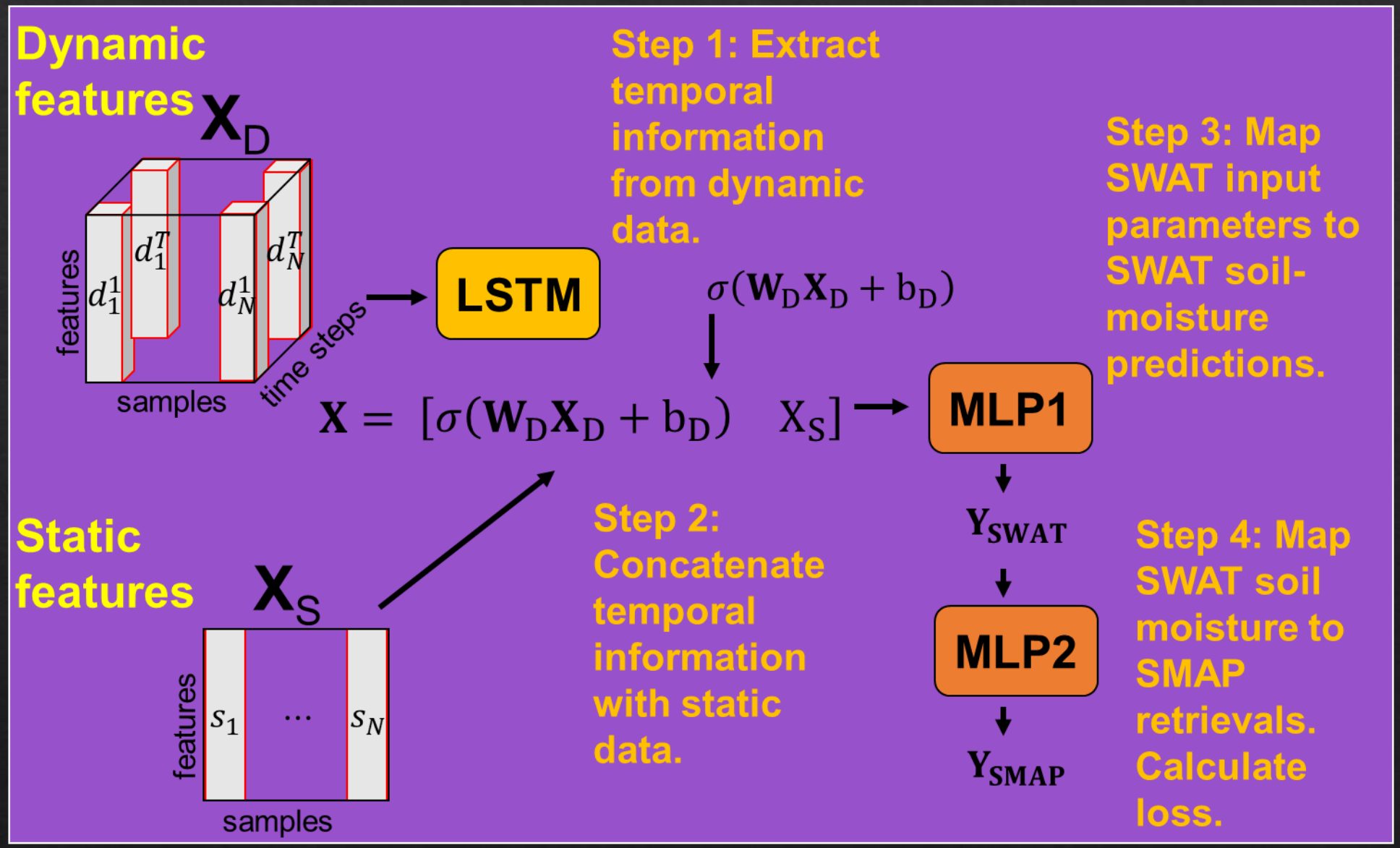
Recurrent Neural Network (RNN)

Long Short-Term Memory (LSTM) cell

LSTM Cell


LSTM layer *l+1*
Input at time *t*
Output from h_{t-1}, c_{t-1}
c_t
f
i
g
o
Historical context
System state at time *t*
timestep *t+1*
h_t

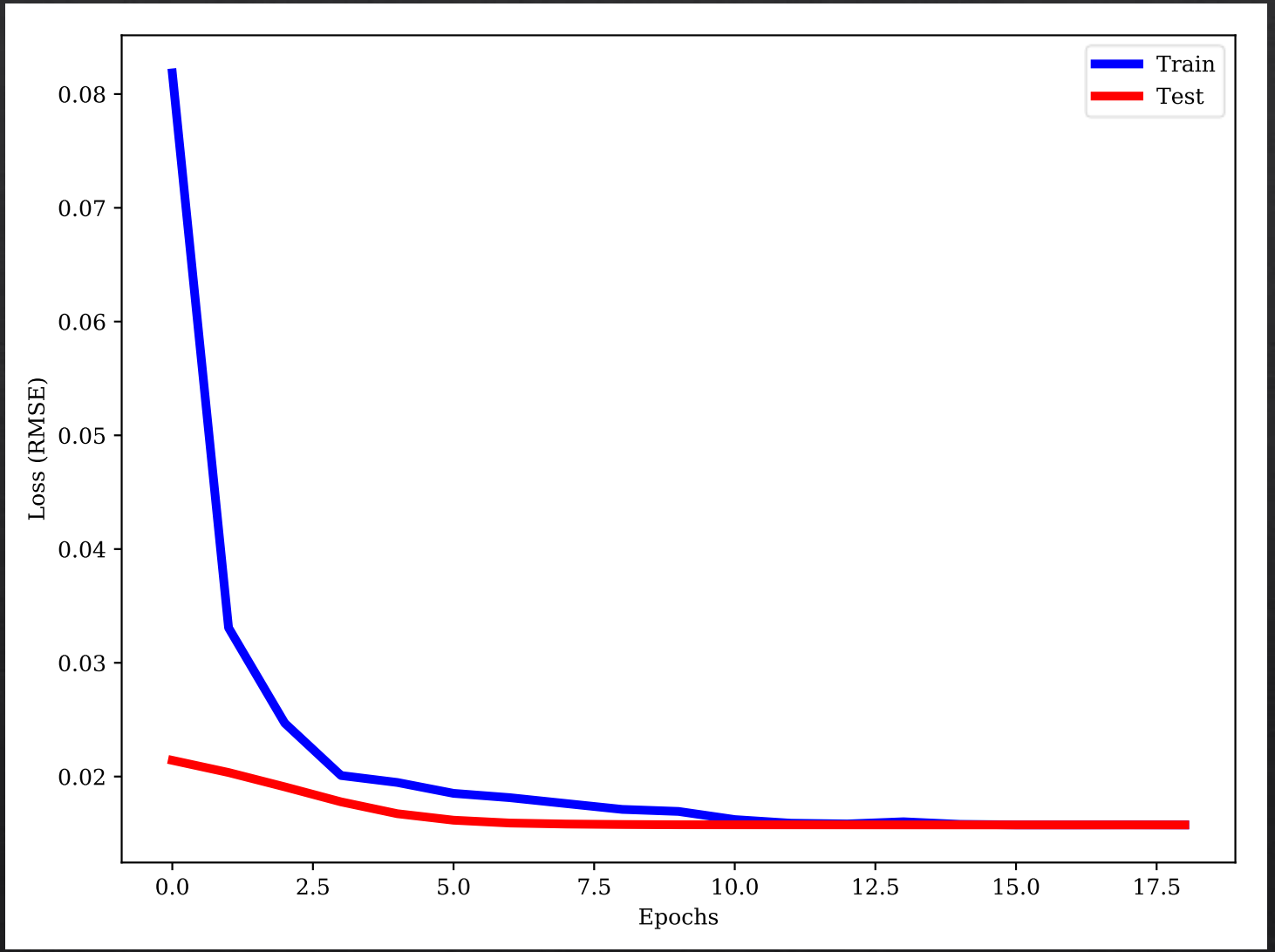
Hybrid Network Architecture



Results

- Attempts to process static and dynamic data with a single MLP network did not yield prediction errors within SMAP mission objectives (MO).

- SMAP soil-moisture MO are accurate within $\pm\, 0.04\ \mathrm{cm^3 cm^{-3}}$ (1-σ). The hybrid network estimated SMAP soil moisture with a training accuracy of about $0.03\ \mathrm{cm^3 cm^{-3}}$ and a test accuracy of $0.035\ \mathrm{cm^3 cm^{-3}}$.



References

- DeVries, P. M., Thompson, T. B., and Meade, B. J. (2017). Enabling large-scale viscoelastic calculations via neural network acceleration. Geophysical Research Le.ers, 44(6):2662–2669.
- Ditty, J. K., Allen, P., David, O., Arnold, J., White, M., and Arabi, M. (2014). Deployment of SWAT-DEG as a web infrastructure utilizing cloud computing for stream restoration.
- James, S. C., Jones, C. A., Grace, M. D., and Roberts, J. D. (2010). Advances in sediment transport modelling. Journal of Hydraulic Research, 48(6):754–763.
- Narasimhan, B., Allen, P., Colman, S., Arnold, J., and Srinivasan, R. (2017). Development and testing of a physically based model of streambank erosion for coupling with a basin-scale hydrologic model SWAT. JAWRA Journal of the American Water Resources Association, 53(2):344–364.
- Neitsch, S., Arnold, J., Kiniry, J., and Williams, J. (2011). Soil water assessment tool theoretical documentation.Version 2011. Texas Water Resource Institute, College Station, Texas. TWRI Report. Technical report, TR-406.
- Noori, N. and Kalin, L. (2016). Coupling SWAT and ANN models for enhanced daily stream.ow prediction. Journal of Hydrology, 533:141–151.