UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

> Reinventa el mundo <

Deep Learning

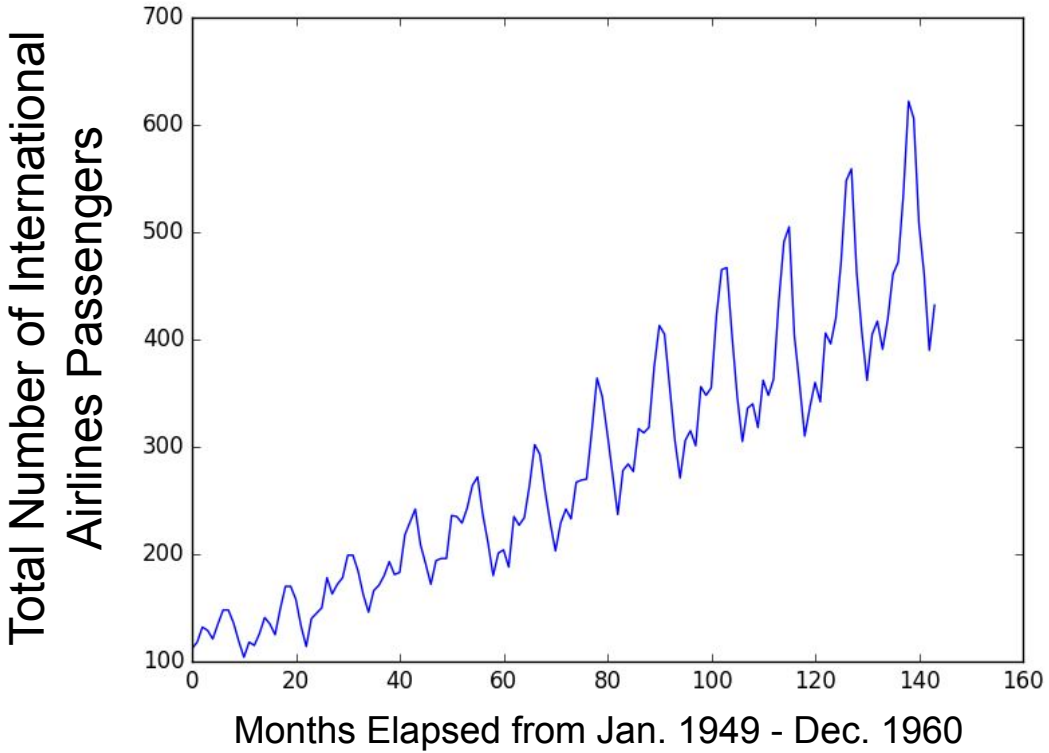
Week 11 : RNNs, LSTMs & Sequence-to-Sequence Models

TRANSFORMATEC


Marriott

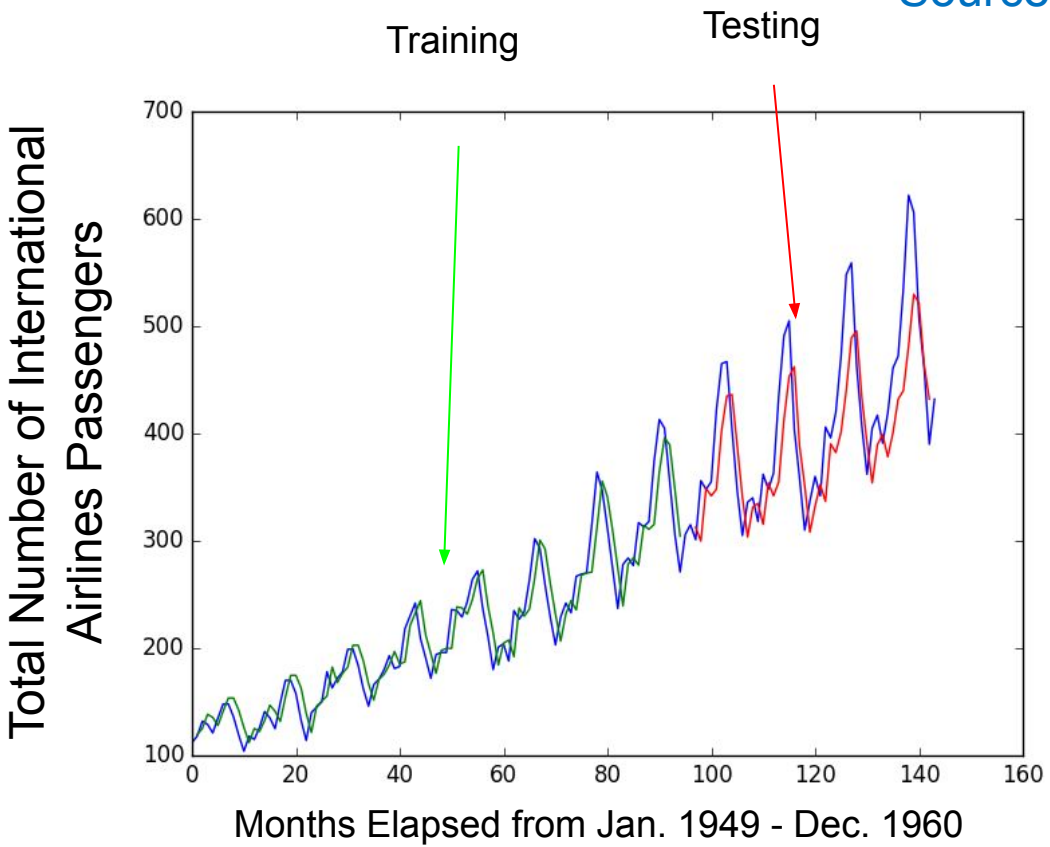
Time Series Forecasting

Source: www.datamarket.com



Time Series Forecasting

Source: www.datamarket.com

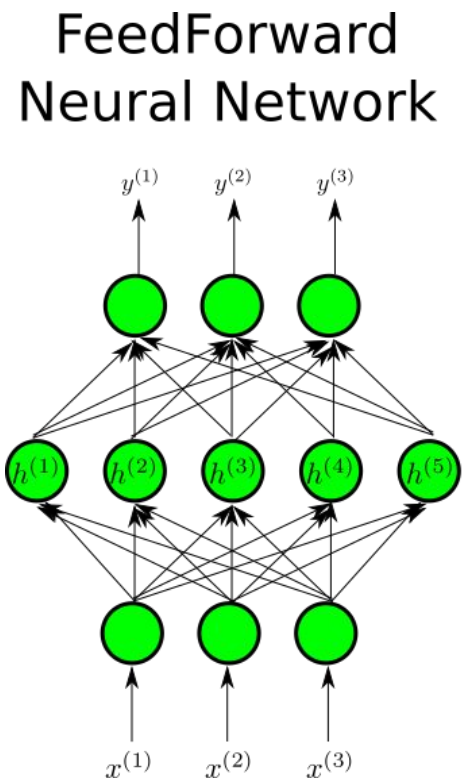


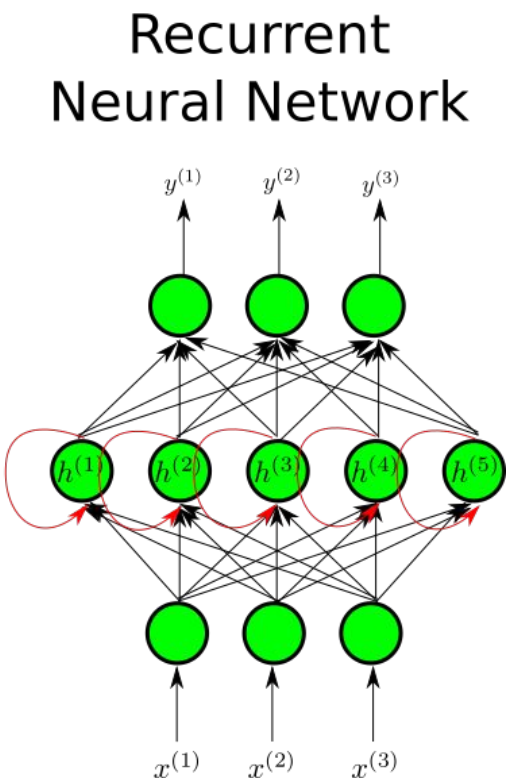
Sequence-to-Sequence Learning

Recurrent Neural Networks (RNNs)

Long Short Term Memory (LSTMs) units

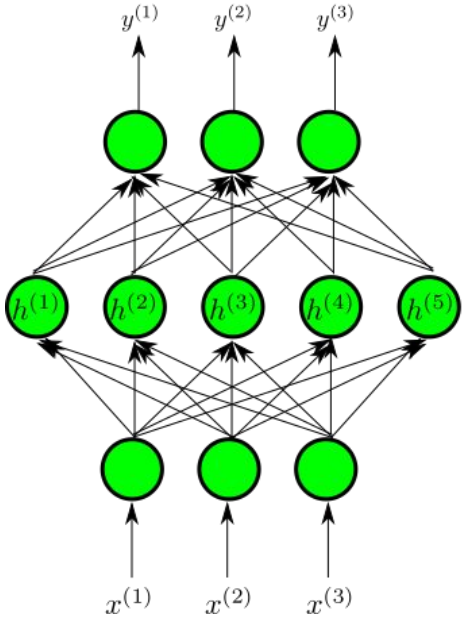
Recurrent Neural Network (RNN) Overview



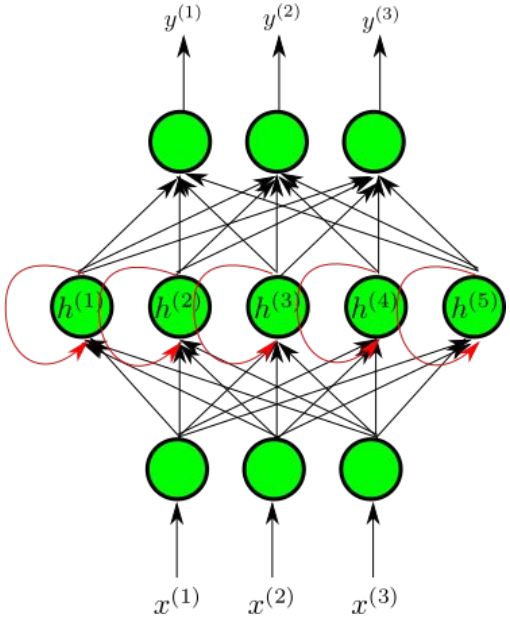


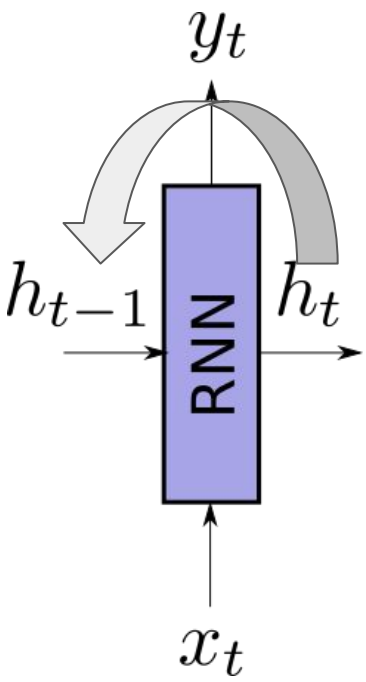
Recurrent Neural Network (RNN) Overview

FeedForward Neural Network

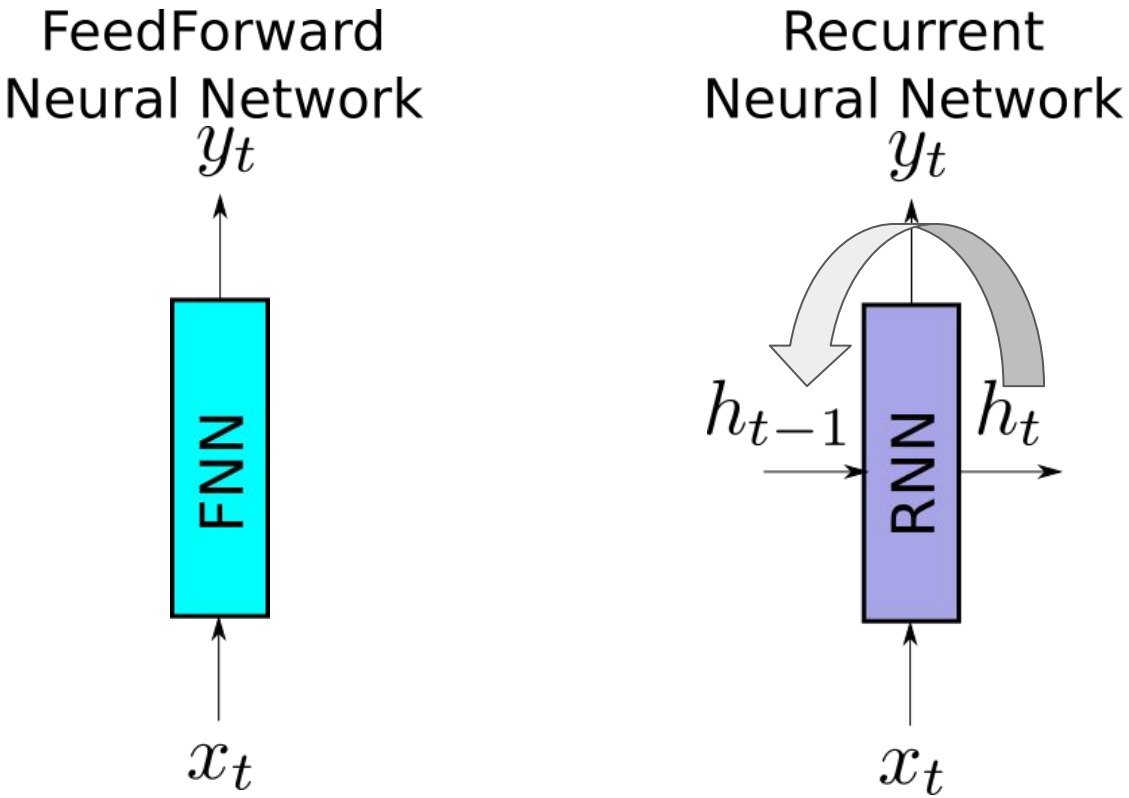


Recurrent Neural Network



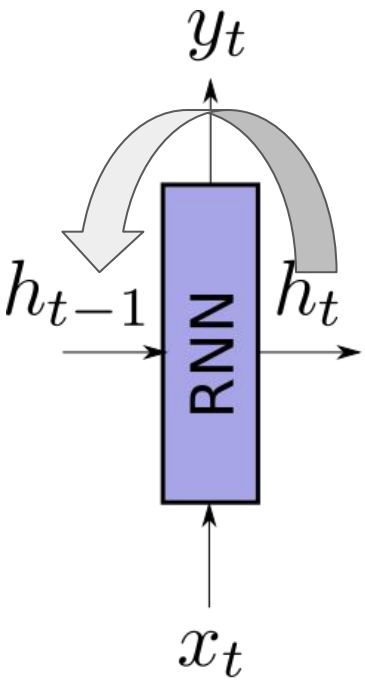


Recurrent Neural Network (RNN) Overview



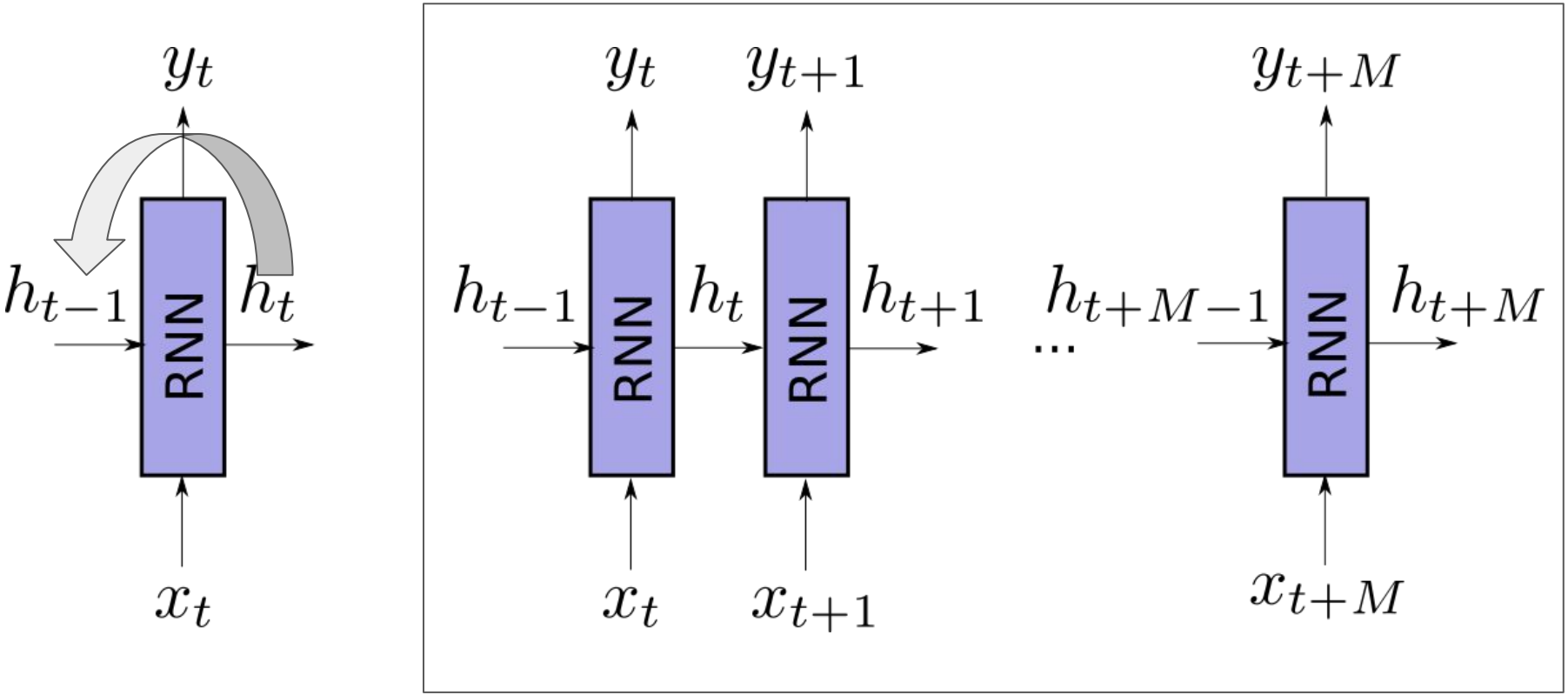
Recurrent Neural Network (RNN) Overview

The General Structure of an RNN



Recurrent Neural Network (RNN) Overview

The General Structure of an RNN

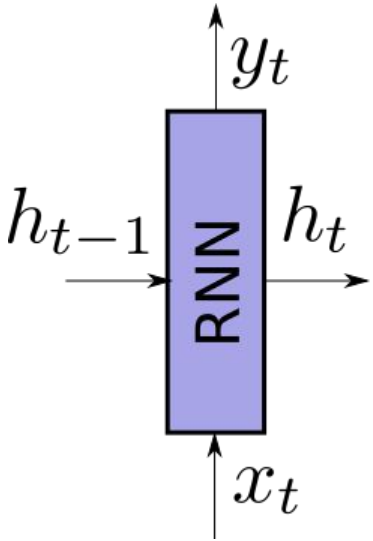


More Formally ...

RNN Equation:

$$h_t = \mathbf{W}^{hh}\phi(h_{t-1}) + \mathbf{W}^{hx}\phi(x_t)$$

$$y_t = \mathbf{W}^{yh}\phi(h_t)$$



More Formally ...

RNN Equation:

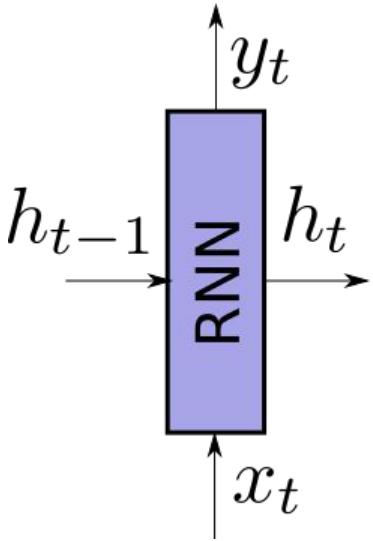
$$h_t = \mathbf{W}^{hh}\phi(h_{t-1}) + \mathbf{W}^{hx}\phi(x_t)$$

$$y_t = \mathbf{W}^{yh}\phi(h_t)$$

Dynamical System Equation:

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$

$$y = \mathbf{C}x + \mathbf{D}u$$



More Formally ...

RNN Equation:

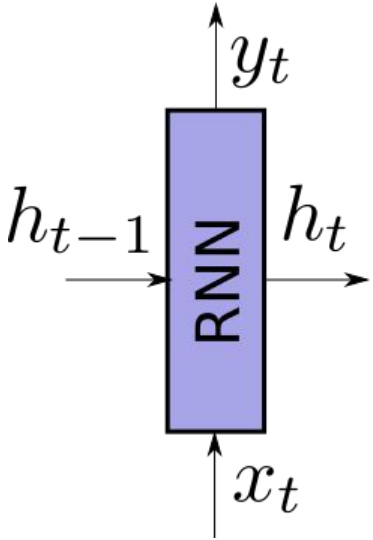
$$h_t = \mathbf{W}^{hh}\phi(h_{t-1}) + \mathbf{W}^{hx}\phi(x_t)$$

$$y_t = \mathbf{W}^{yh}\phi(h_t)$$

Dynamical System Equation:

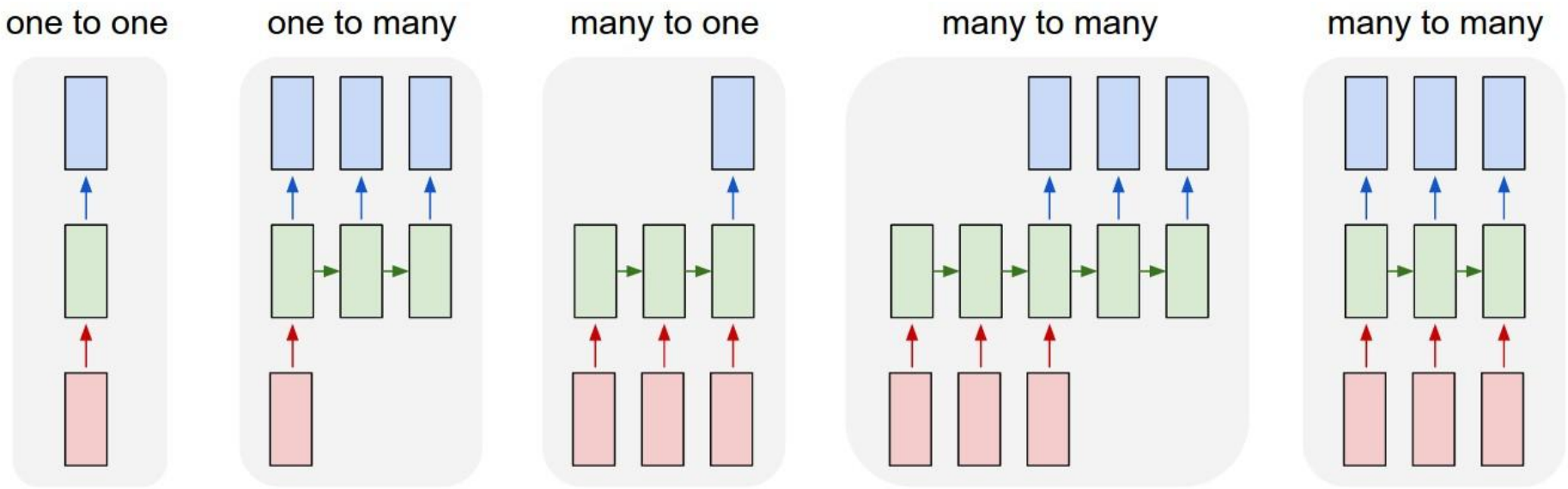
$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$

$$y = \mathbf{C}x + \mathbf{D}u$$



Recurrent Neural Network (RNN) Overview

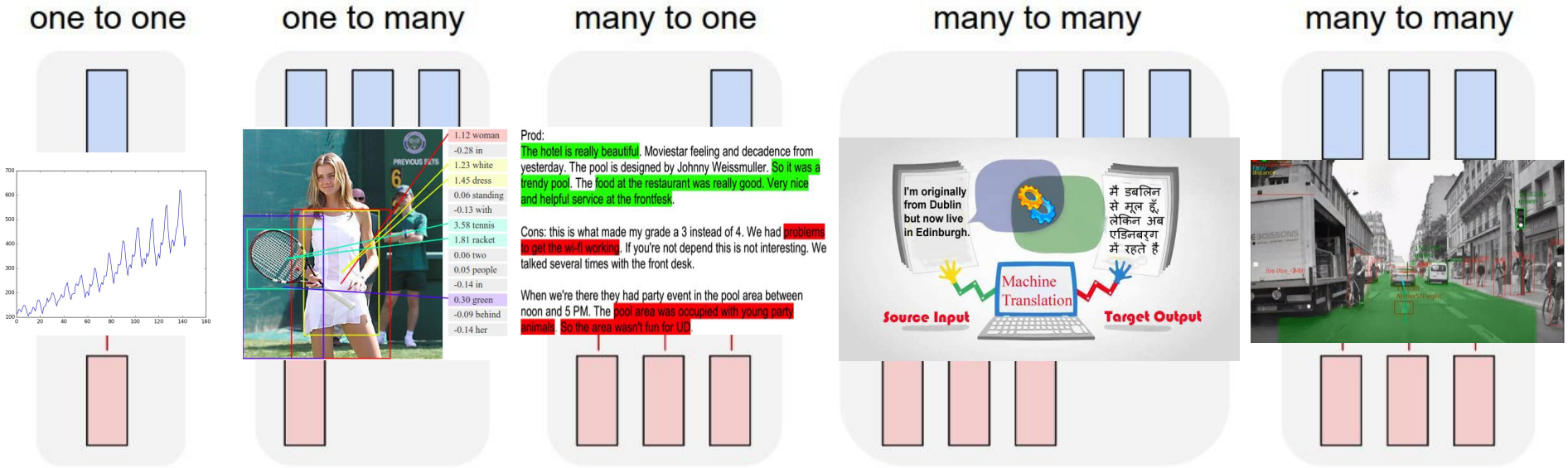
Different Flavors of RNNs



Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Network (RNN) Overview

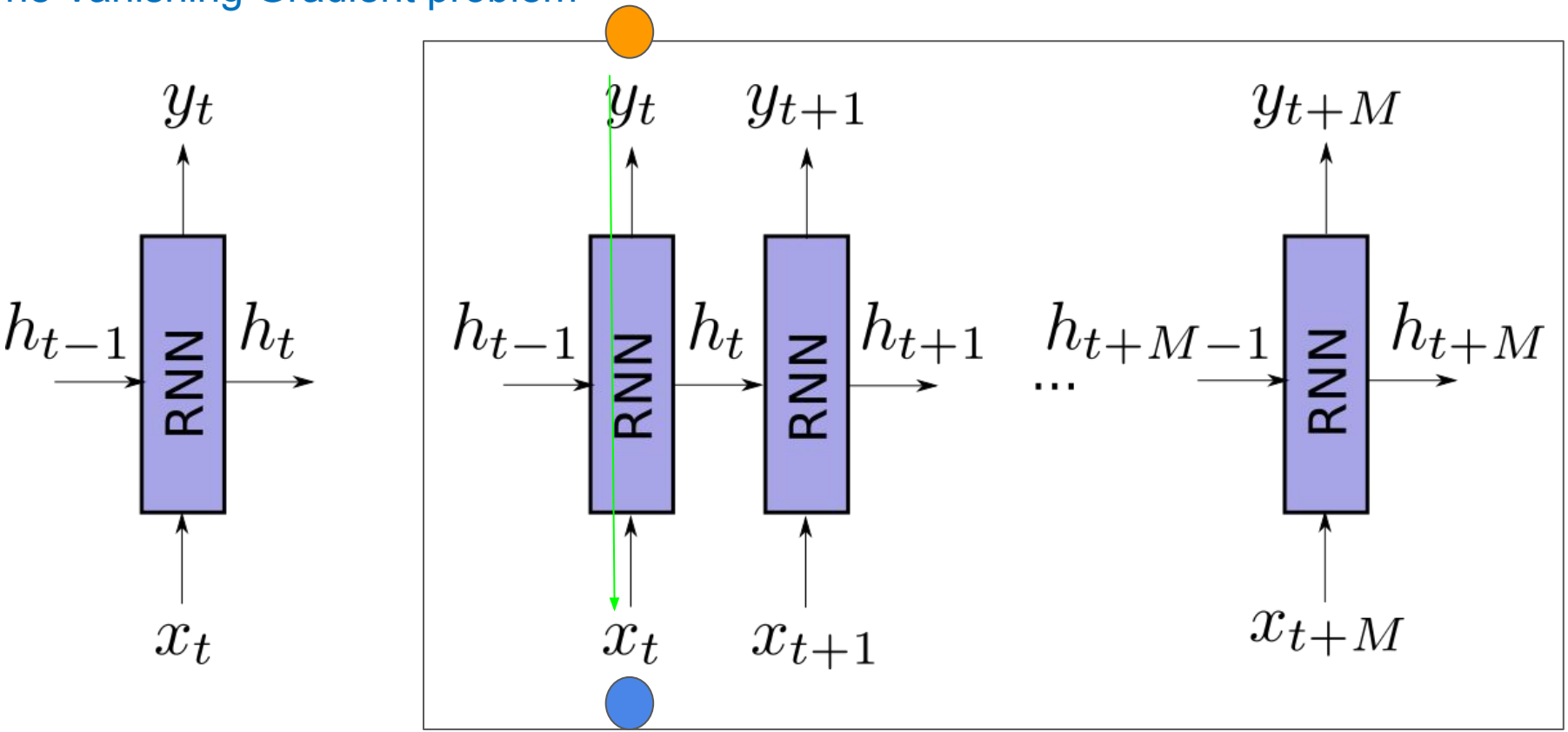
Different Flavors of RNNs



Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

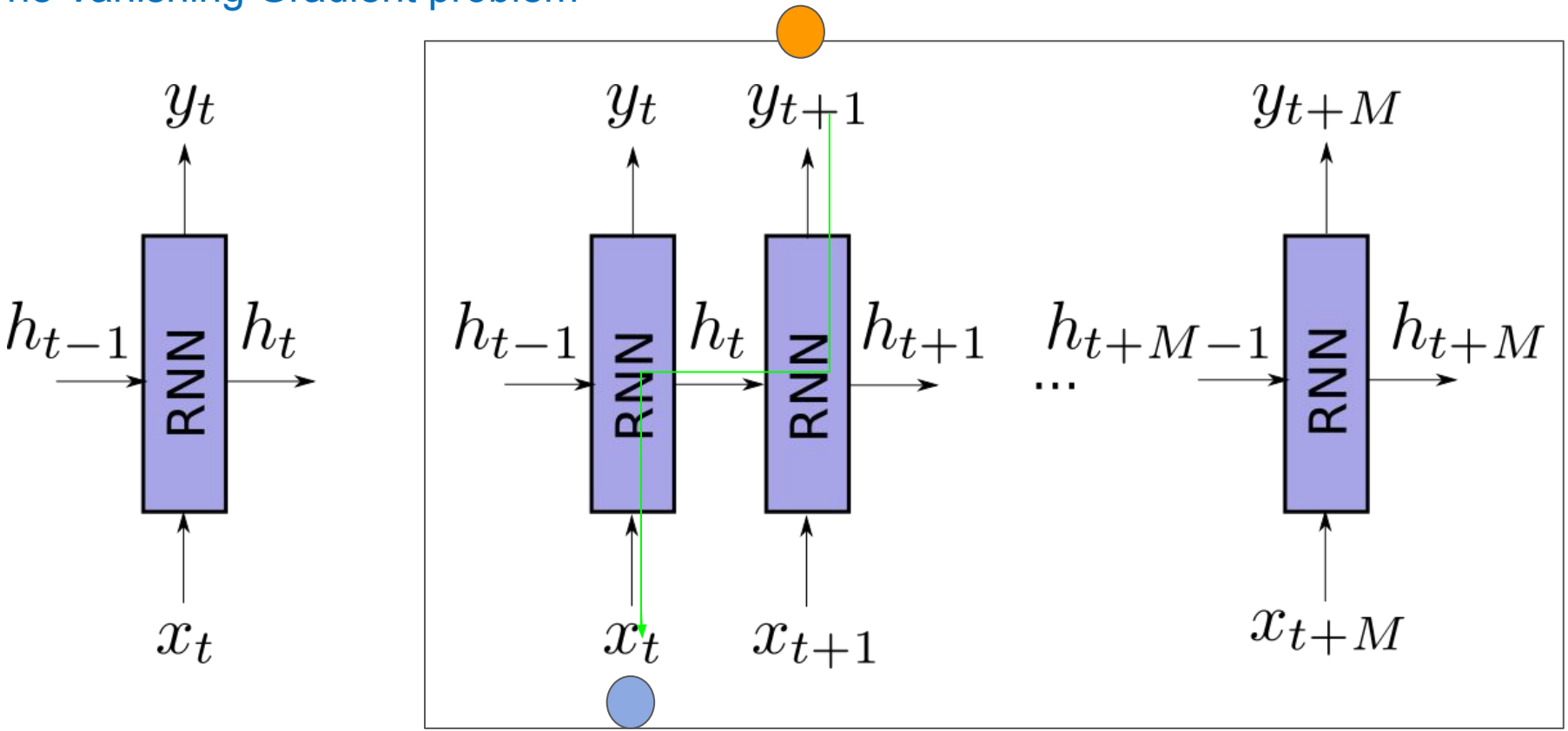
Recurrent Neural Network (RNN) Overview

The Vanishing Gradient problem



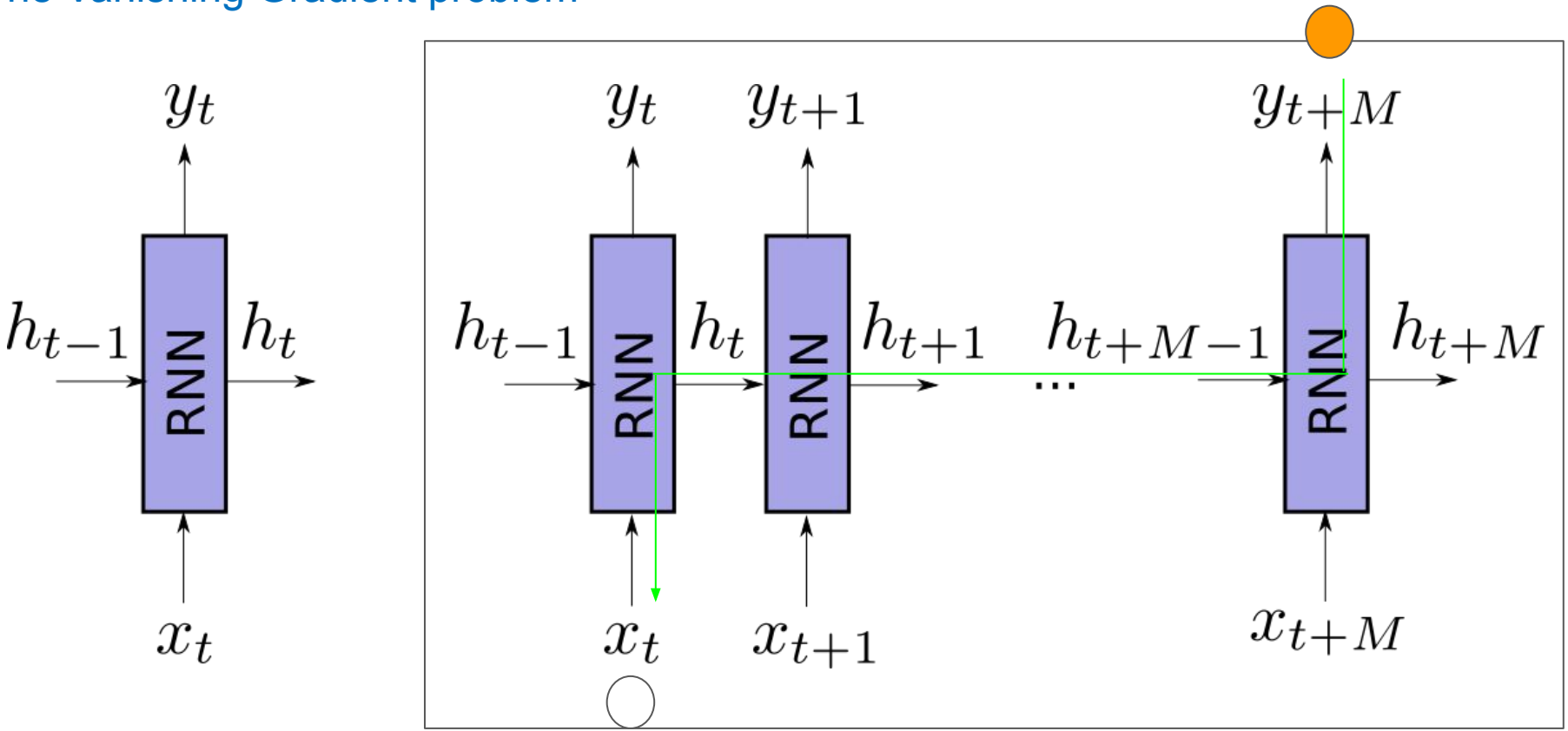
Recurrent Neural Network (RNN) Overview

The Vanishing Gradient problem



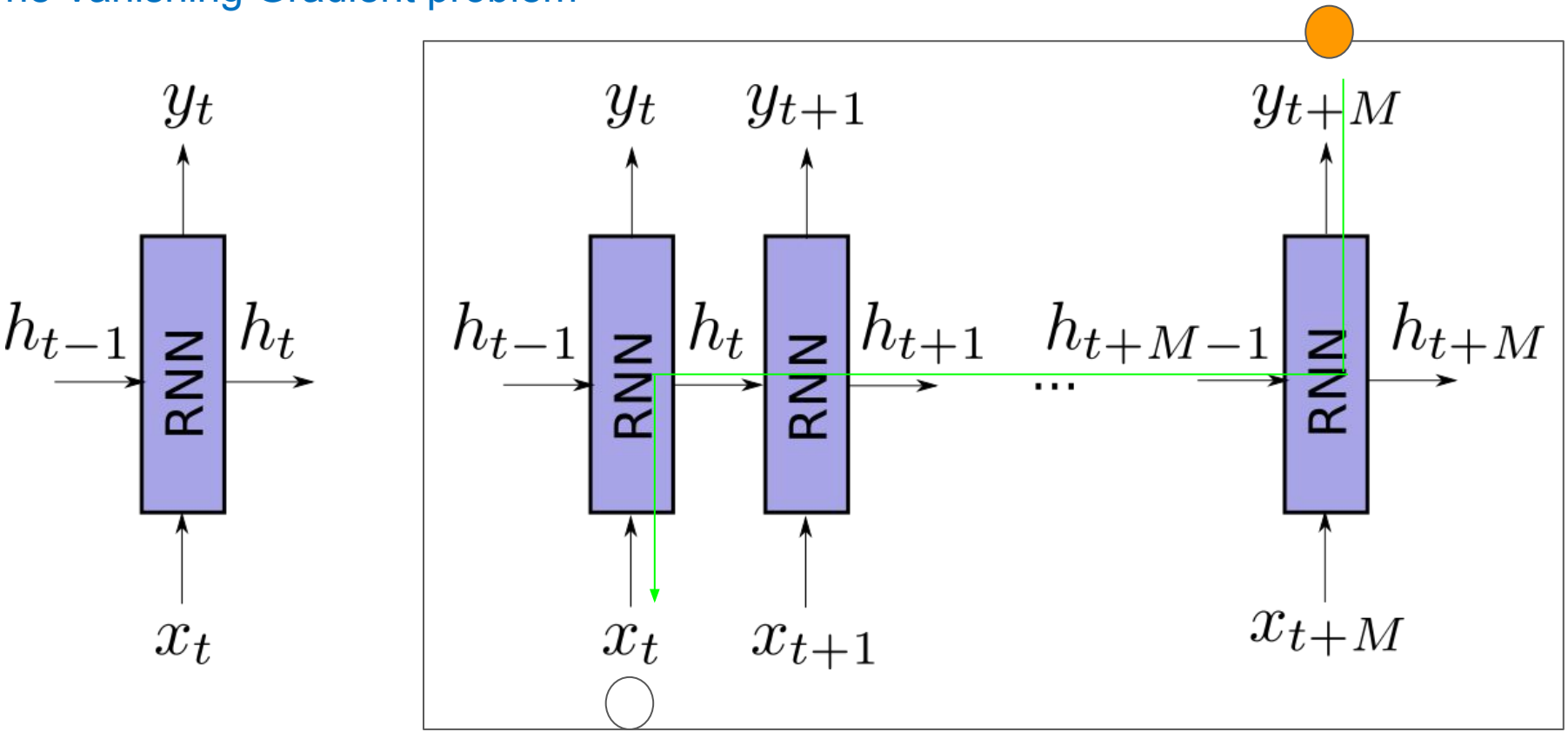
Recurrent Neural Network (RNN) Overview

The Vanishing Gradient problem



Recurrent Neural Network (RNN) Overview

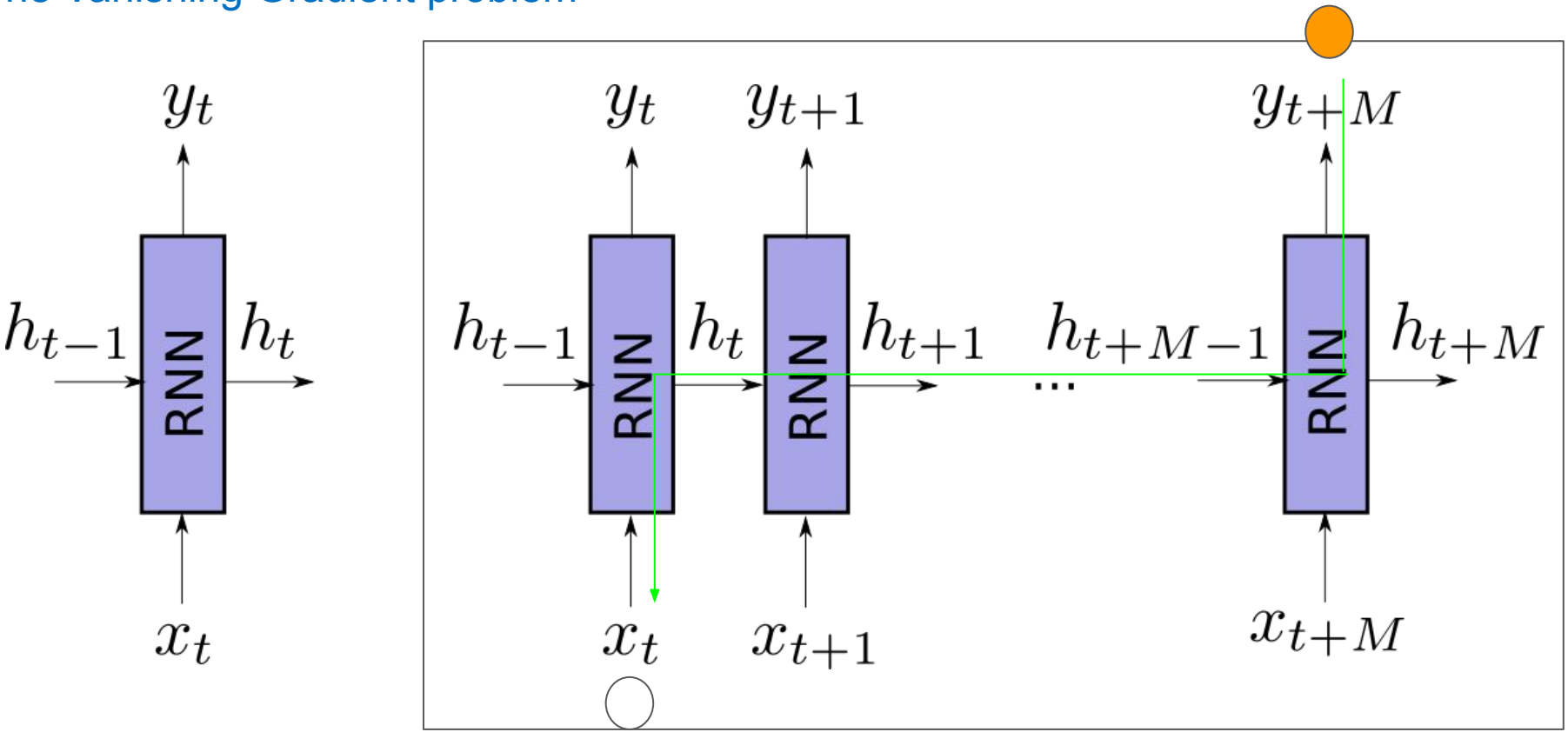
The Vanishing Gradient problem



The gradient we need to backpropagate the error can go to zero!

Recurrent Neural Network (RNN) Overview

The Vanishing Gradient problem



The gradient we need to backpropagate the error can go to zero!

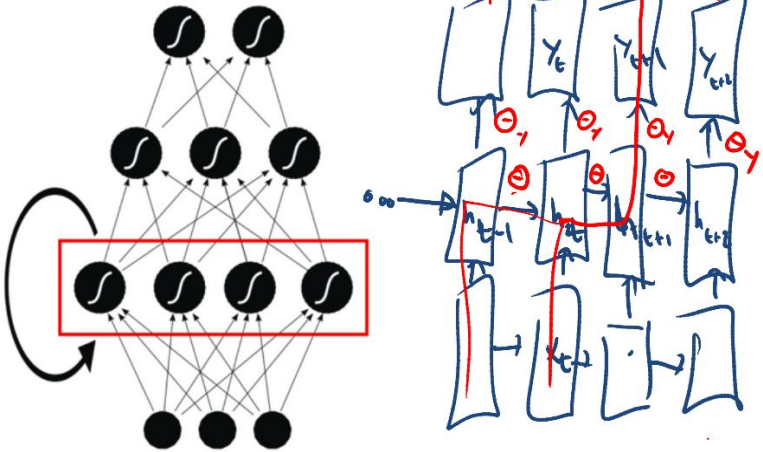
Problems with RNNs

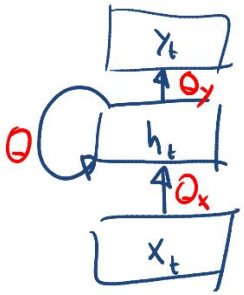
Vanishing Gradient problem

Source: Nando de Freitas

$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$

$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$



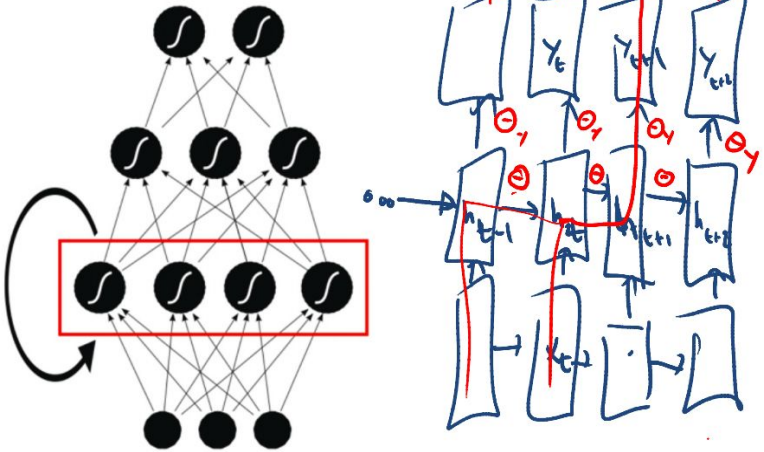


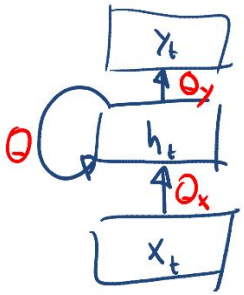
Problems with RNNs

Vanishing Gradient problem

Source: Nando de Freitas

$$\mathbf{h}_t = \boldsymbol{\theta}\phi(\mathbf{h}_{t-1}) + \boldsymbol{\theta}_x \mathbf{x}_t$$
$$\mathbf{y}_t = \boldsymbol{\theta}_y \phi(\mathbf{h}_t)$$





$$\mathbf{h}_t = \boldsymbol{\theta}\phi(\mathbf{h}_{t-1}) + \boldsymbol{\theta}_x \mathbf{x}_t$$
$$\mathbf{y}_t = \boldsymbol{\theta}_y \phi(\mathbf{h}_t)$$

$$\frac{\partial E}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{S} \frac{\partial E_t}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial E_t}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \boldsymbol{\theta}}$$

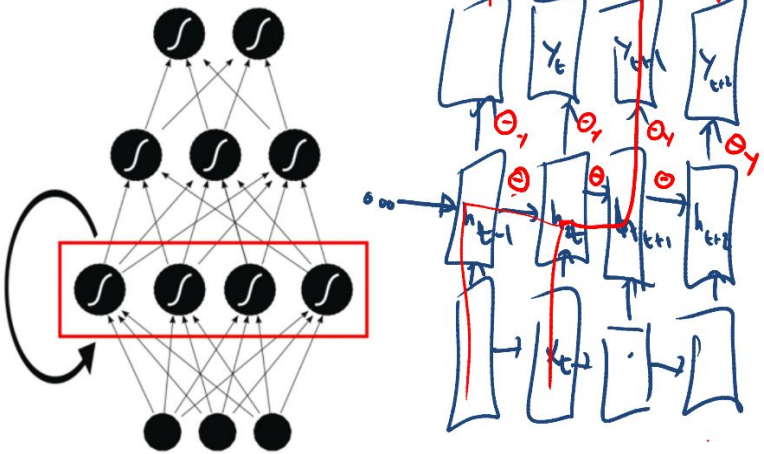
Problems with RNNs

Vanishing Gradient problem

Source: Nando de Freitas

$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$

$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$



$$\frac{\partial E_t}{\partial \theta} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \theta}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{i=k+1}^{t} \theta^T \mathrm{diag}[\phi'(\mathbf{h}_{i-1})]$$

$$\left\| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \|\theta^T\| \|\mathrm{diag}[\phi'(\mathbf{h}_{i-1})]\| \leq \gamma_\theta \gamma_\phi$$

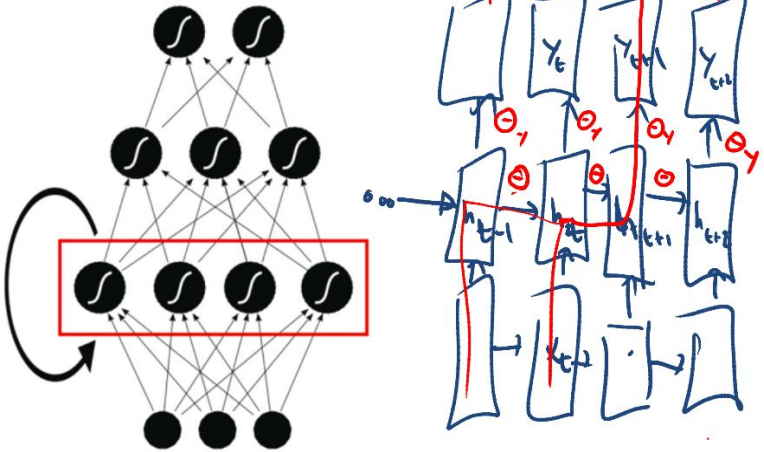
$$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \right\| \leq (\gamma_\theta \gamma_\phi)^{t-k}$$

Problems with RNNs

Vanishing Gradient problem

Source: Nando de Freitas

$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$
$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$



$$\frac{\partial E_t}{\partial \theta} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \theta}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{i=k+1}^{t} \theta^T \mathrm{diag}[\phi'(\mathbf{h}_{i-1})]$$

$$\left\| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \|\theta^T\| \|\mathrm{diag}[\phi'(\mathbf{h}_{i-1})]\| \leq \gamma_\theta \gamma_\phi$$

$$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \right\| \leq (\gamma_\theta \gamma_\phi)^{t-k}$$

Blows up, or goes down!

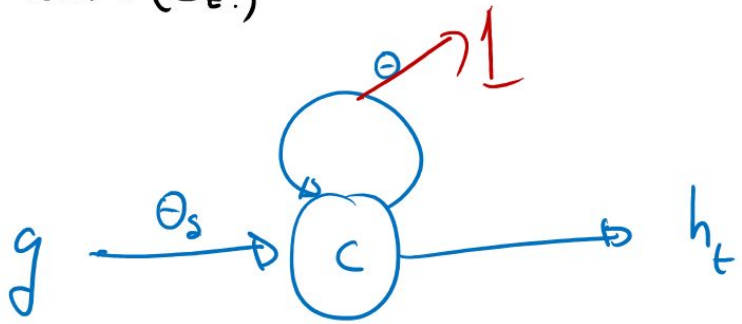
A “Hacky Insight” to the vanishing gradient problem

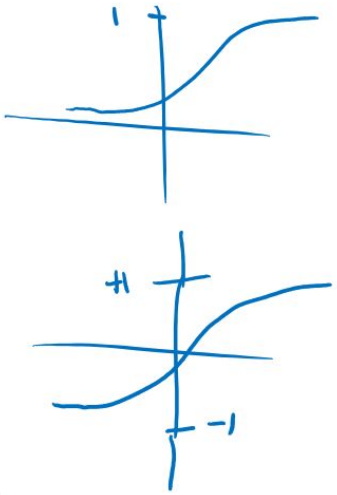
Source: Nando de Freitas

Simple solution

$$c_t = \theta c_{t-1} + \theta_2 g_t$$

$$h_t = \tanh(c_t)$$



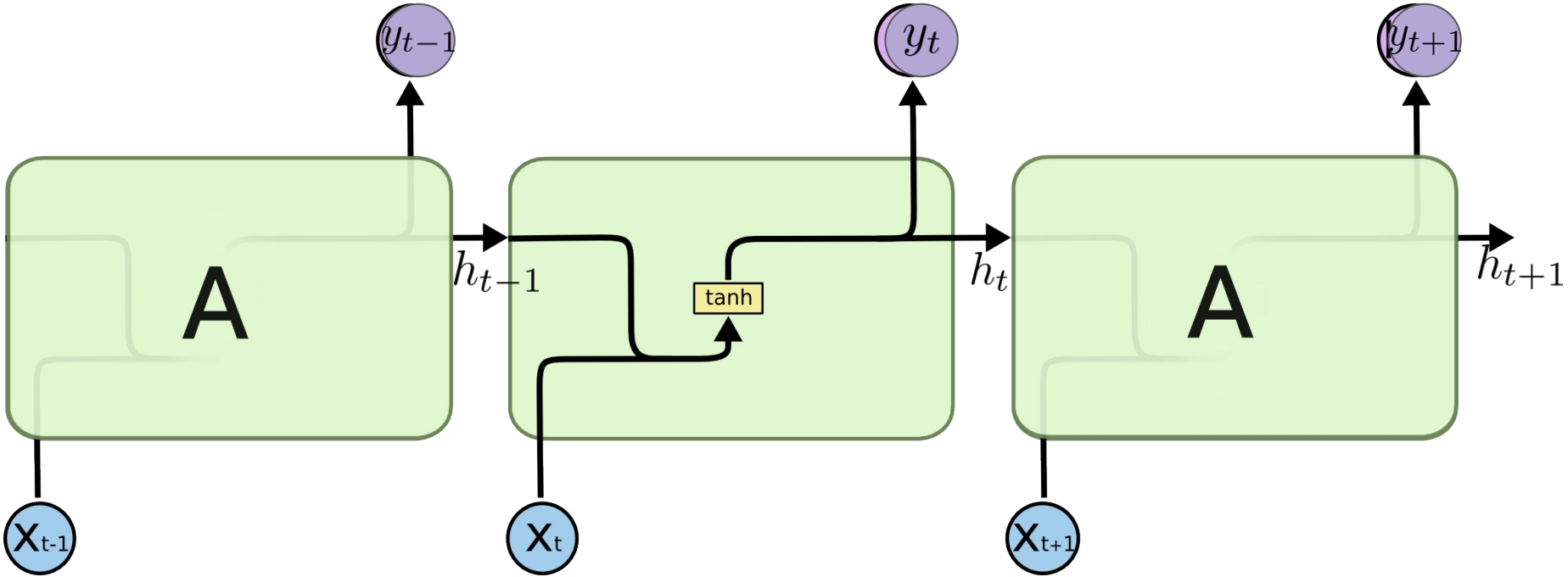


Recurrent Neural Network (RNN) Overview

A closer look!

Source: Christopher Olah's blog

- **A Recurrent Neural Network (RNN)**

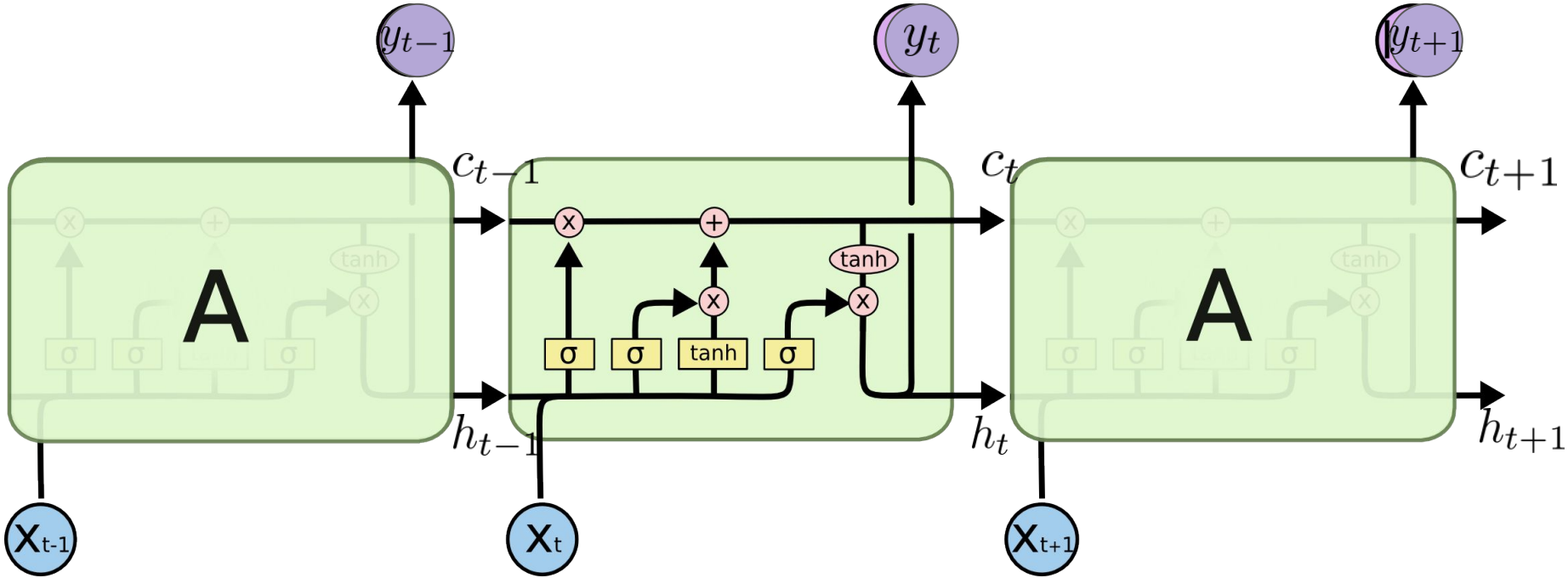


LSTM Overview

A solution to our closer look

Source: Christopher Olah's blog

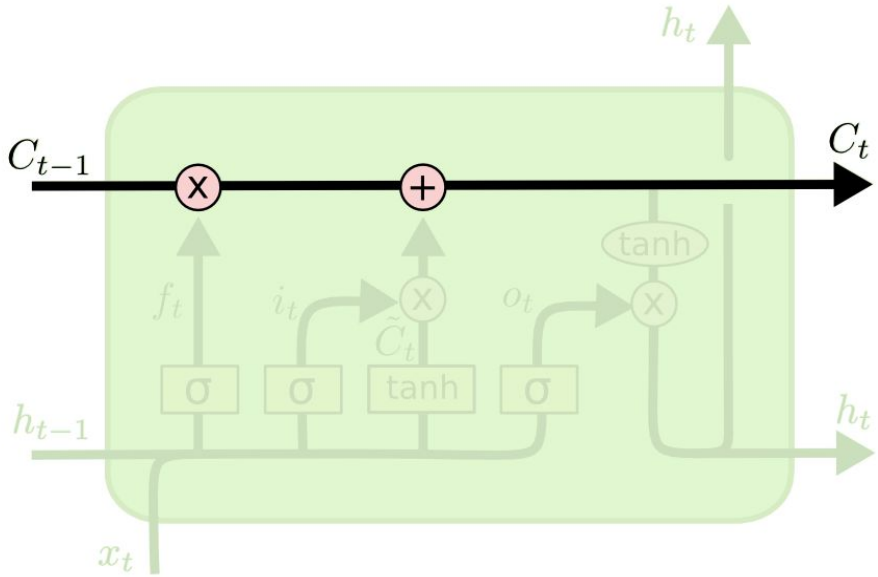
- **A Long Short-Term Memory (LSTM)**



LSTM Overview

Inside an LSTM Cell

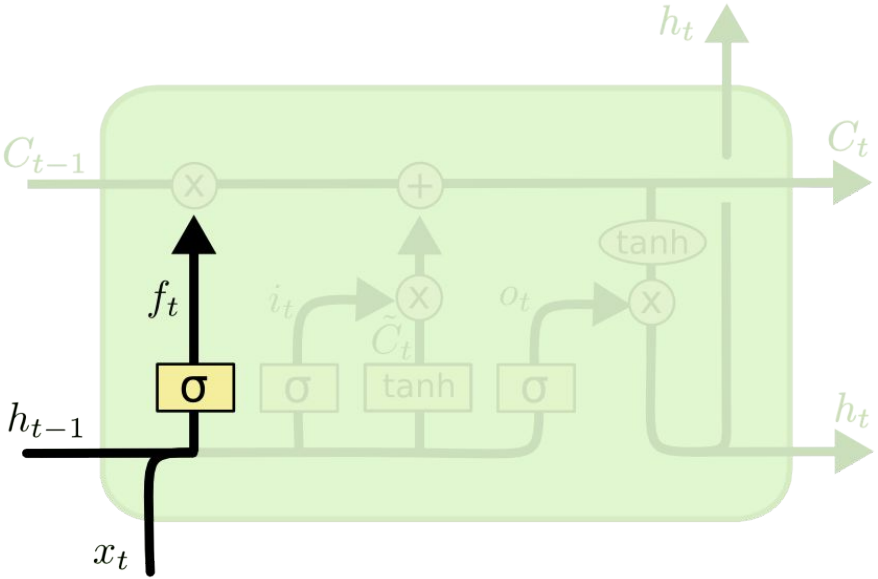
Source: Christopher Olah's blog



LSTM Overview

Inside an LSTM Cell

Source: Christopher Olah's blog

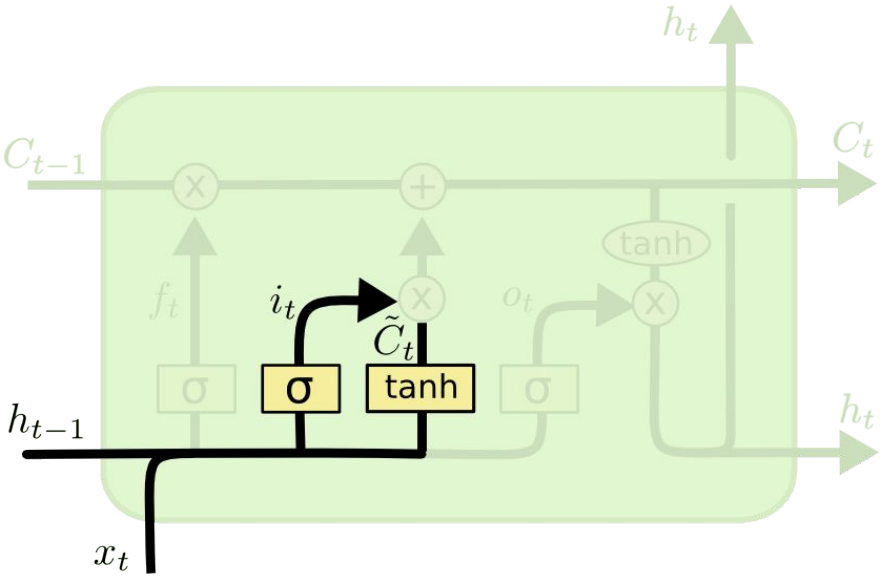


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM Overview

Inside an LSTM Cell

Source: Christopher Olah's blog



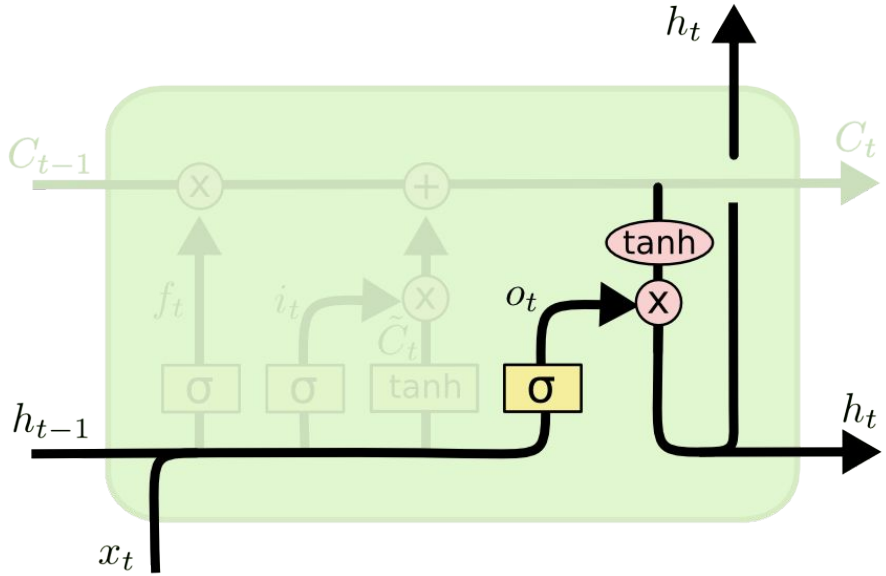
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM Overview

Inside an LSTM Cell

Source: Christopher Olah's blog

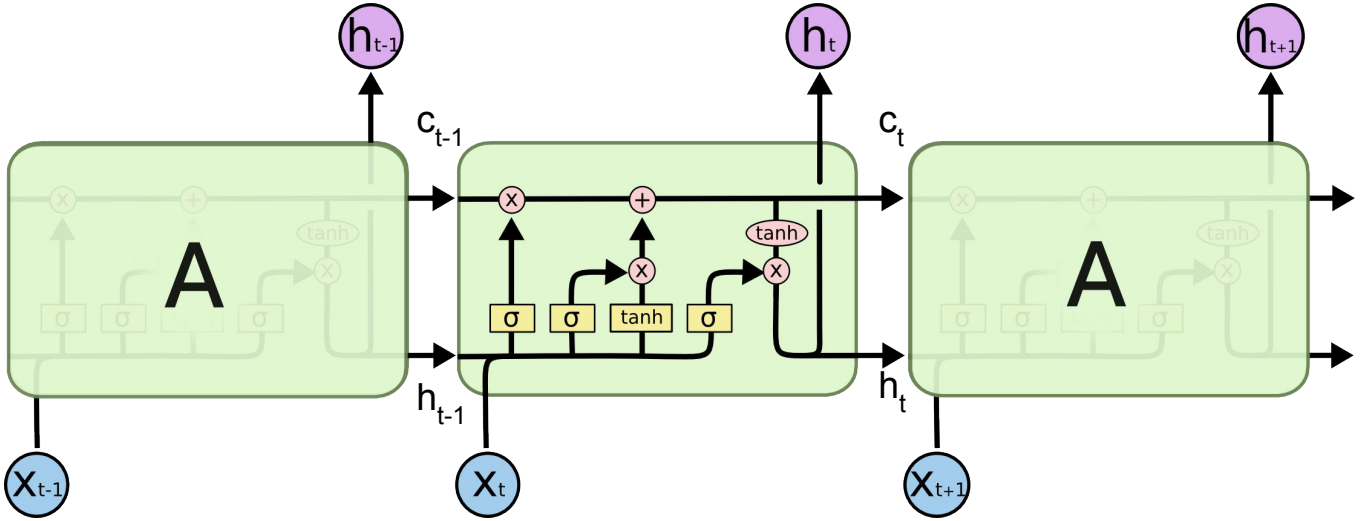


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM Summary

Source: Christopher Olah's blog

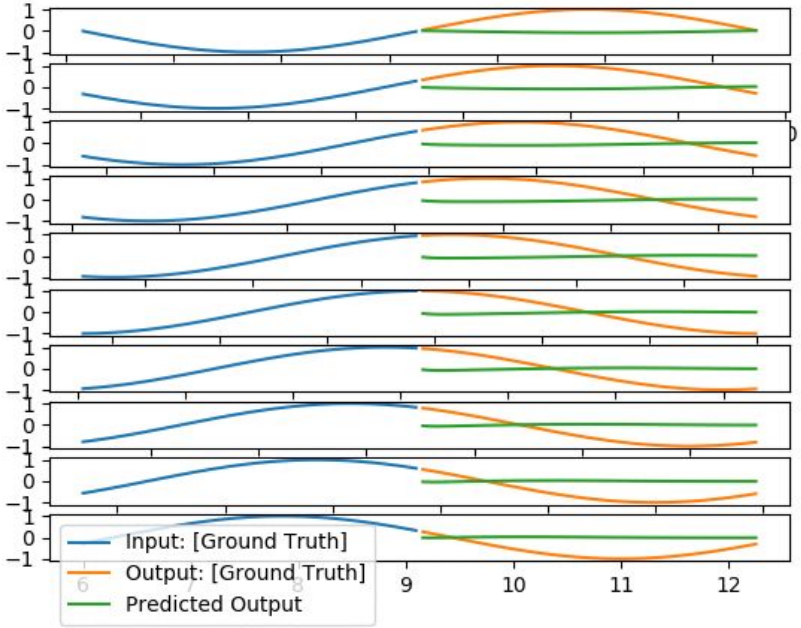


$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t)$$

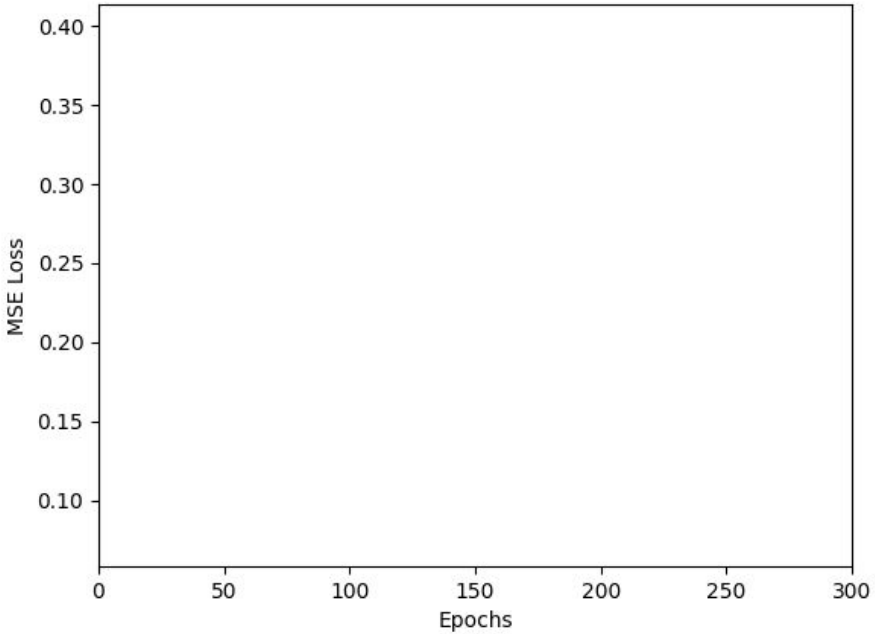
Example: Predicting the other half of a Sine Wave

LSTM hidden states: 10; Statefulness Off;

Signal + Reconstruction



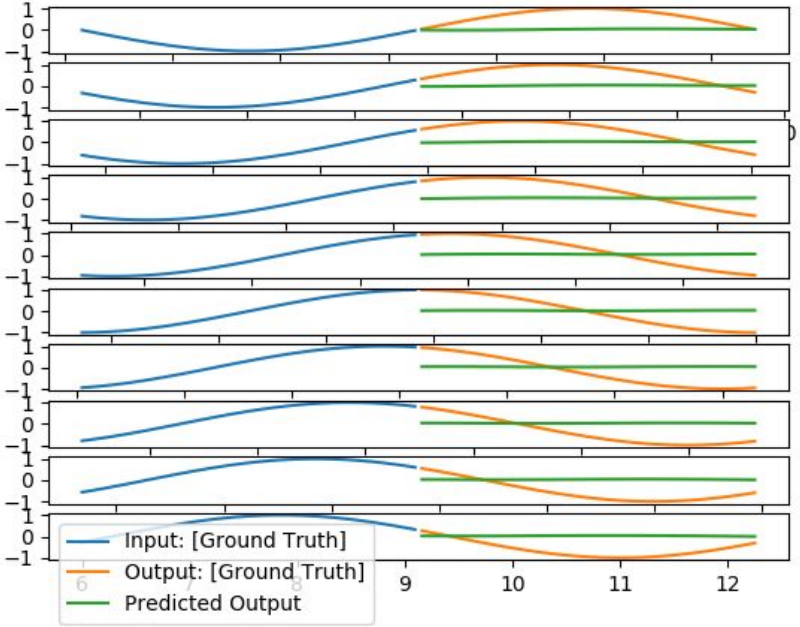
Training Error



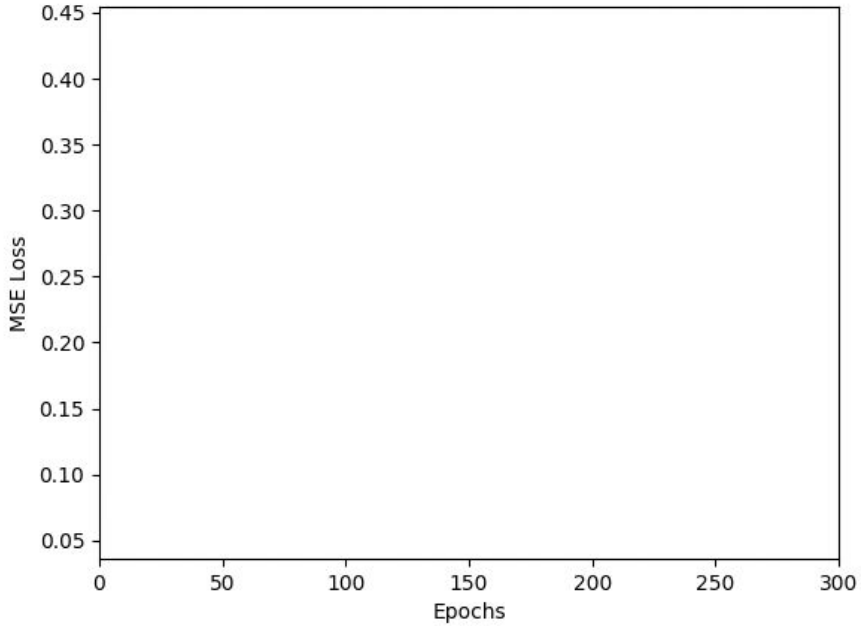
Example: Predicting the other half of a Sine Wave

LSTM hidden states: 10; Statefulness On;

Signal + Reconstruction



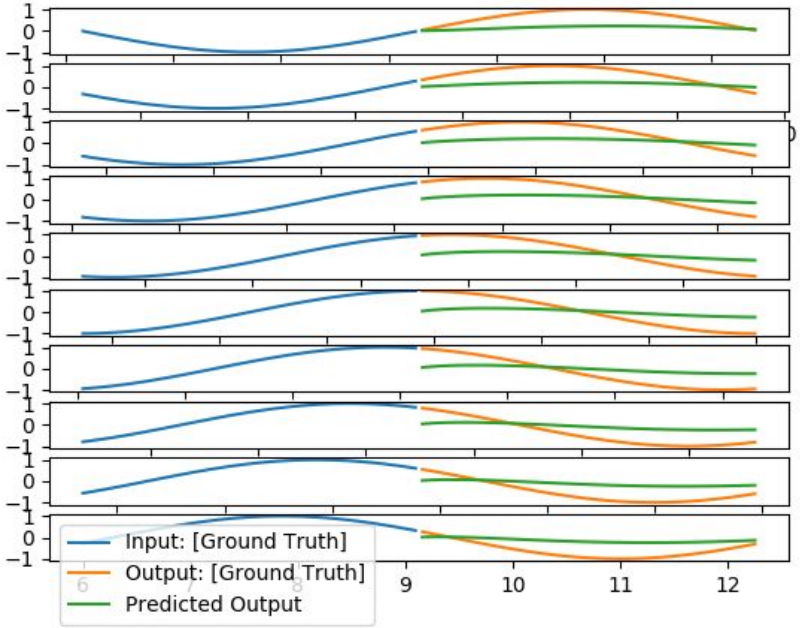
Training Error



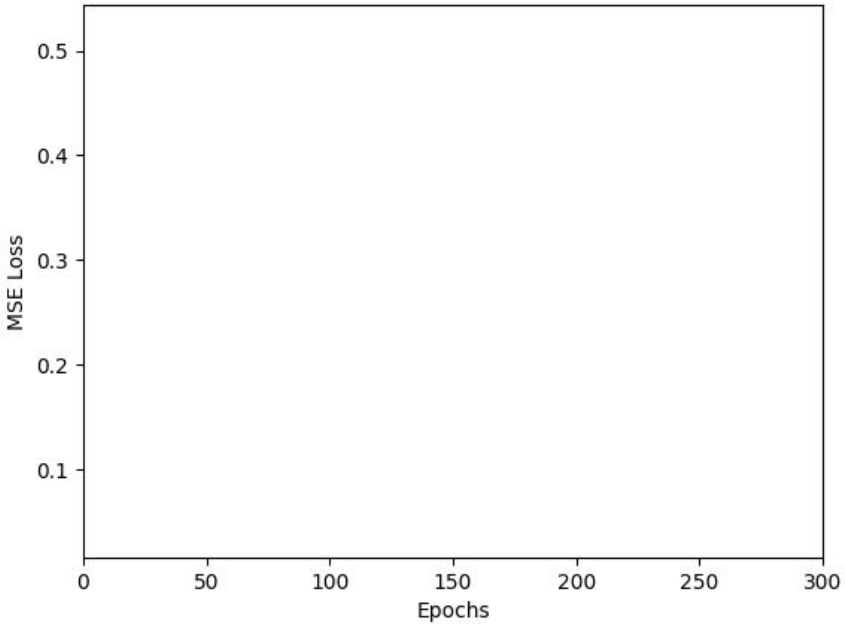
Example: Predicting the other half of a Sine Wave

LSTM hidden states: 25; Statefulness Off;

Signal + Reconstruction



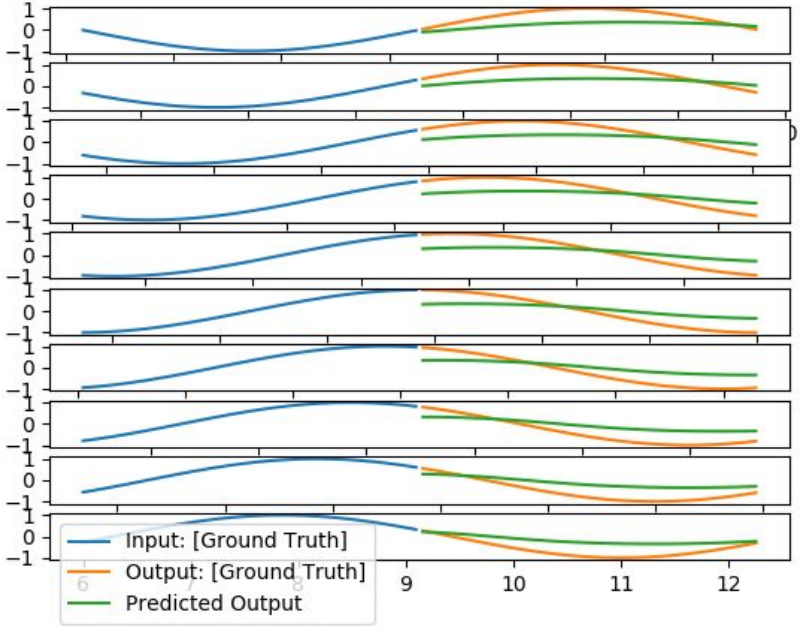
Training Error



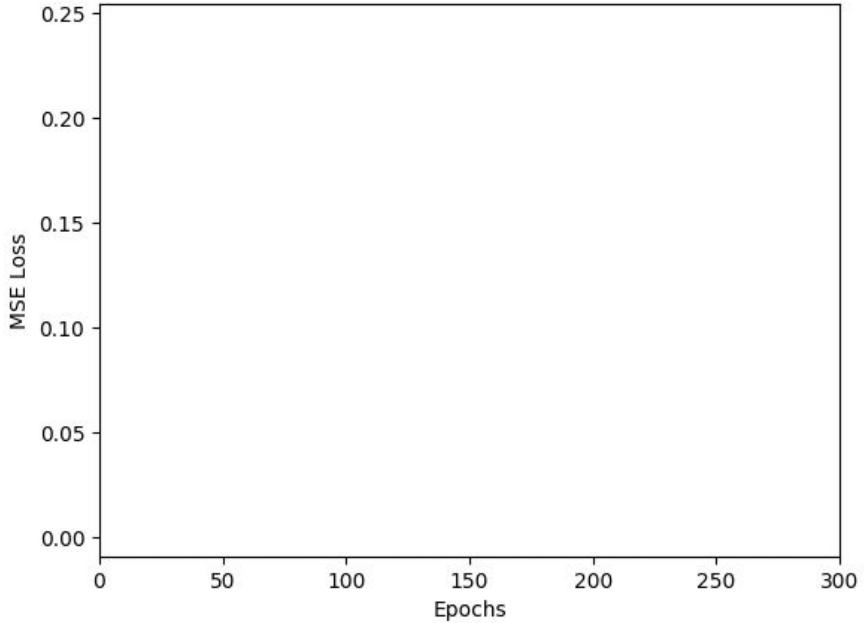
Example: Predicting the other half of a Sine Wave

LSTM hidden states: 25; Statefulness On;

Signal + Reconstruction



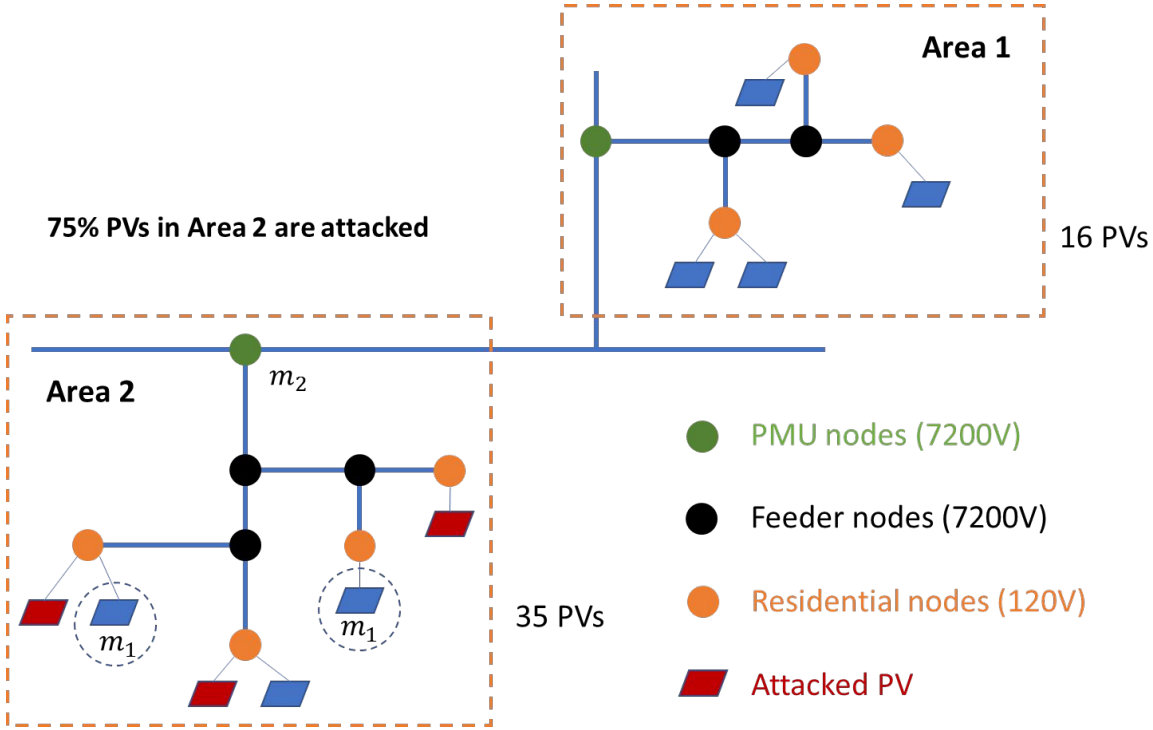
Training Error



Smart Grid Sensor Forecasting

Current Problem & Dataset

An overview of the Smart-Grid



Current LSTM Architecture

House_meter_Name

time

solar_flux (W/sf)

$z_t^{(0)}$ P house (W)

$z_t^{(1)}$ Q house (Var)

$z_t^{(2)}$ P solar (W)

$z_t^{(3)}$ Q solar (Var)

$z_t^{(4)}$ V1 solar (V)

$z_t^{(5)}$ ph1 solar (rad)

$z_t^{(6)}$ V2 solar (V)

$z_t^{(7)}$ ph2 solar (rad)

$z_t^{(8)}$ I1 solar (A)

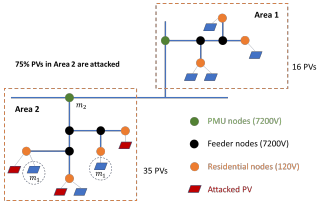
$z_t^{(9)}$ theta1 solar (rad)

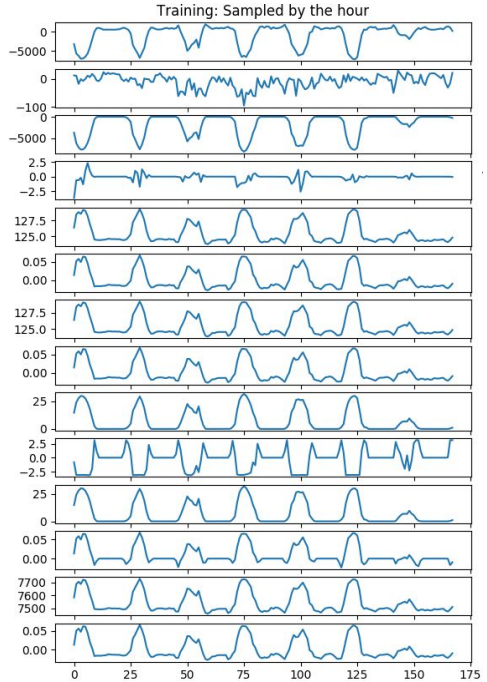
$z_t^{(10)}$ I2 solar (A)

$z_t^{(11)}$ theta2 solar (rad)

$z_t^{(12)}$ node V (V)

$z_t^{(13)}$ node ph (rad)





Current LSTM Architecture

House_meter_Name

time

solar_flux (W/sf)

$z_t^{(0)}$ P house (W)

$z_t^{(1)}$ Q house (Var)

$z_t^{(2)}$ P solar (W)

$z_t^{(3)}$ Q solar (Var)

$z_t^{(4)}$ V1 solar (V)

$z_t^{(5)}$ ph1 solar (rad)

$z_t^{(6)}$ V2 solar (V)

$z_t^{(7)}$ ph2 solar (rad)

$z_t^{(8)}$ I1 solar (A)

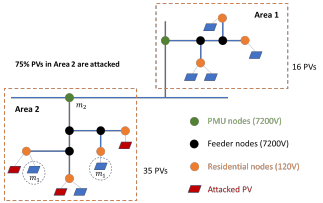
$z_t^{(9)}$ theta1 solar (rad)

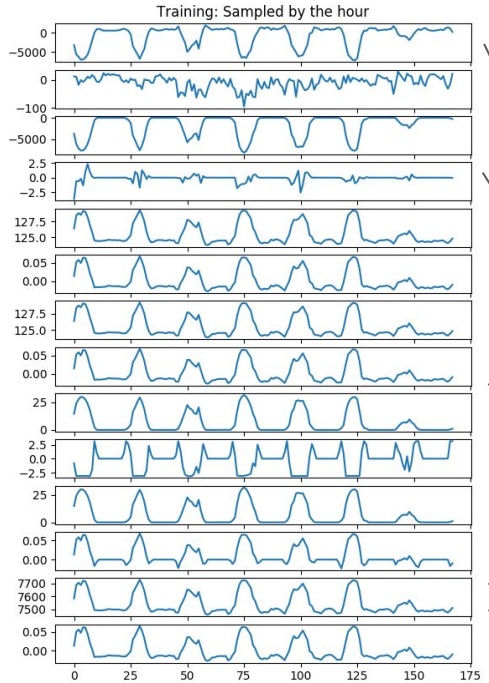
$z_t^{(10)}$ I2 solar (A)

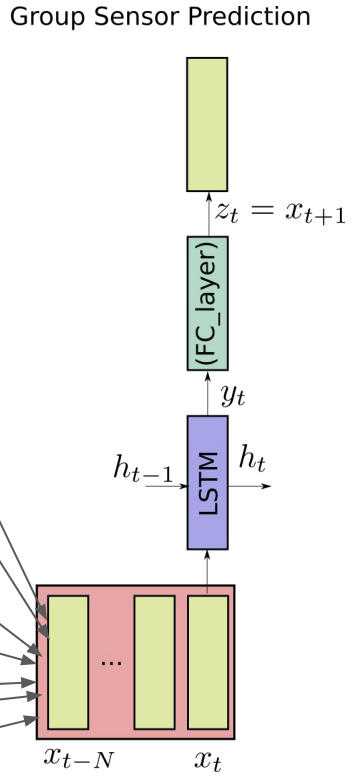
$z_t^{(11)}$ theta2 solar (rad)

$z_t^{(12)}$ node V (V)

$z_t^{(13)}$ node ph (rad)







Current LSTM Architecture

House_meter_Name

time

solar_flux (W/sf)

$z_t^{(0)}$ P house (W)

$z_t^{(1)}$ Q house (Var)

$z_t^{(2)}$ P solar (W)

$z_t^{(3)}$ Q solar (Var)

$z_t^{(4)}$ V1 solar (V)

$z_t^{(5)}$ ph1 solar (rad)

$z_t^{(6)}$ V2 solar (V)

$z_t^{(7)}$ ph2 solar (rad)

$z_t^{(8)}$ I1 solar (A)

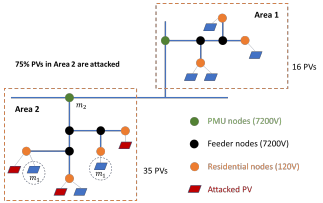
$z_t^{(9)}$ theta1 solar (rad)

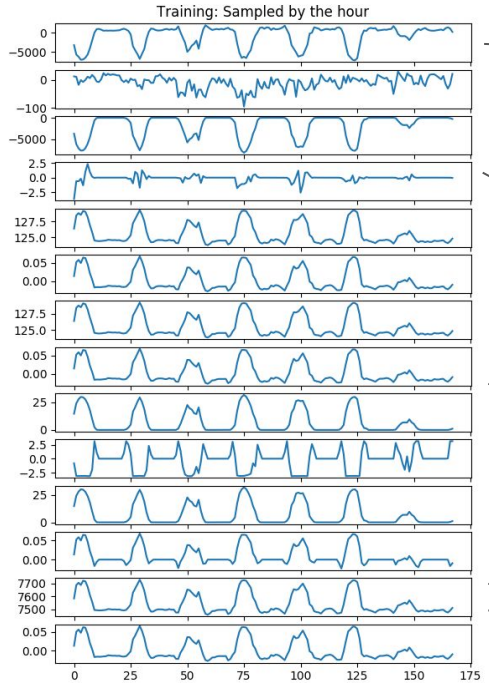
$z_t^{(10)}$ I2 solar (A)

$z_t^{(11)}$ theta2 solar (rad)

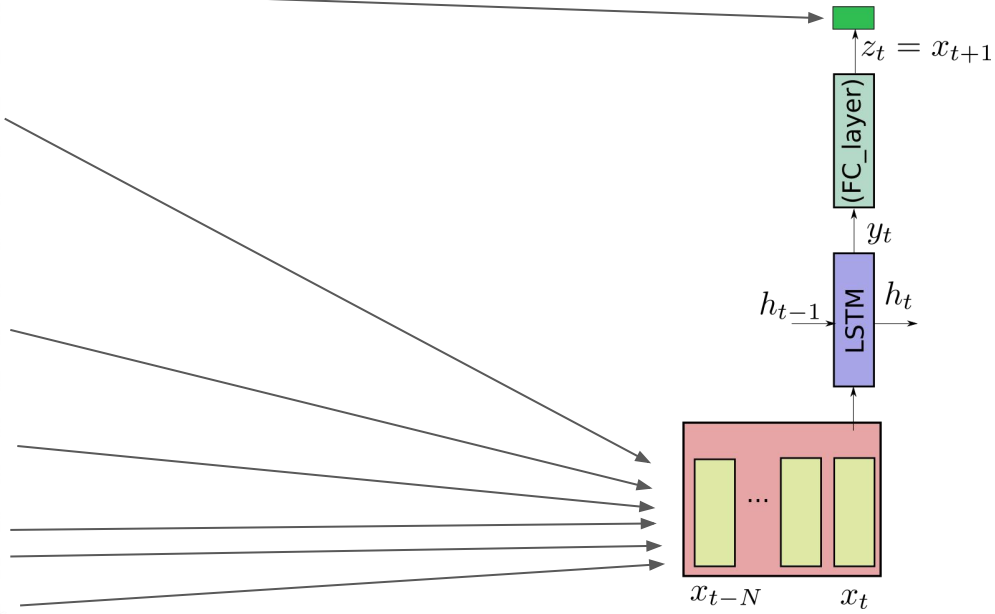
$z_t^{(12)}$ node V (V)

$z_t^{(13)}$ node ph (rad)





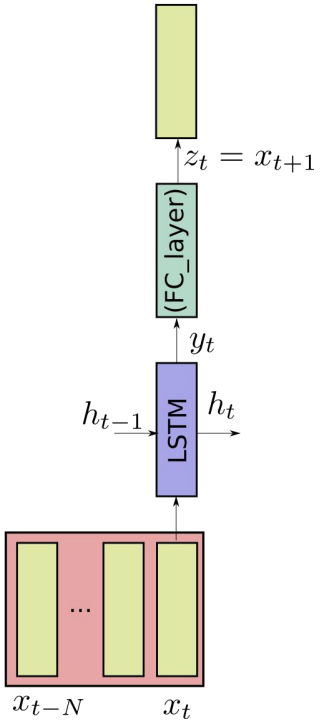
Single Sensor Prediction



Solar Panel Sensor Data

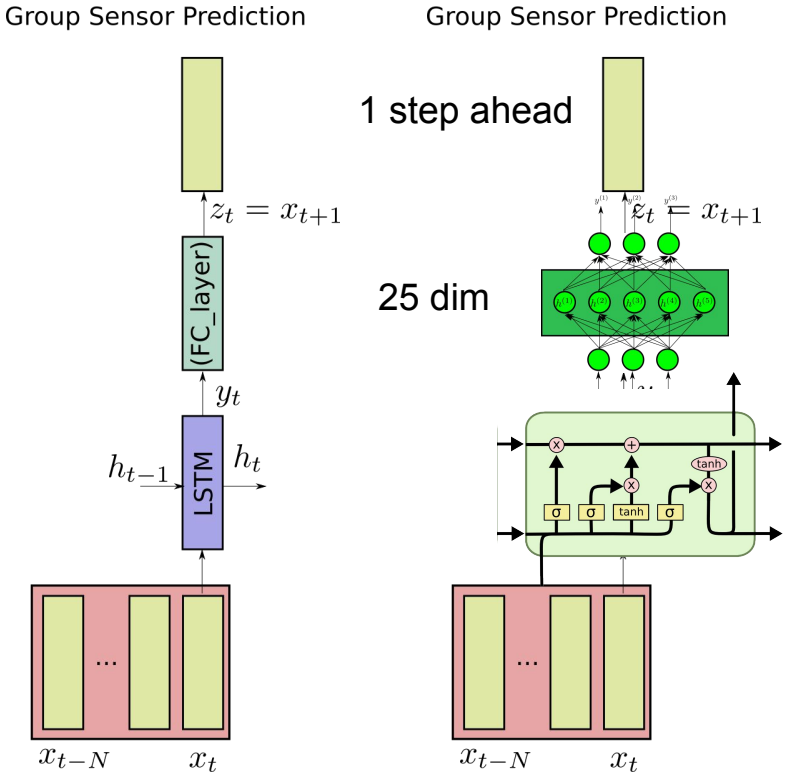
Qualitative Results of Training & Testing on the same house

Group Sensor Prediction



Solar Panel Sensor Data

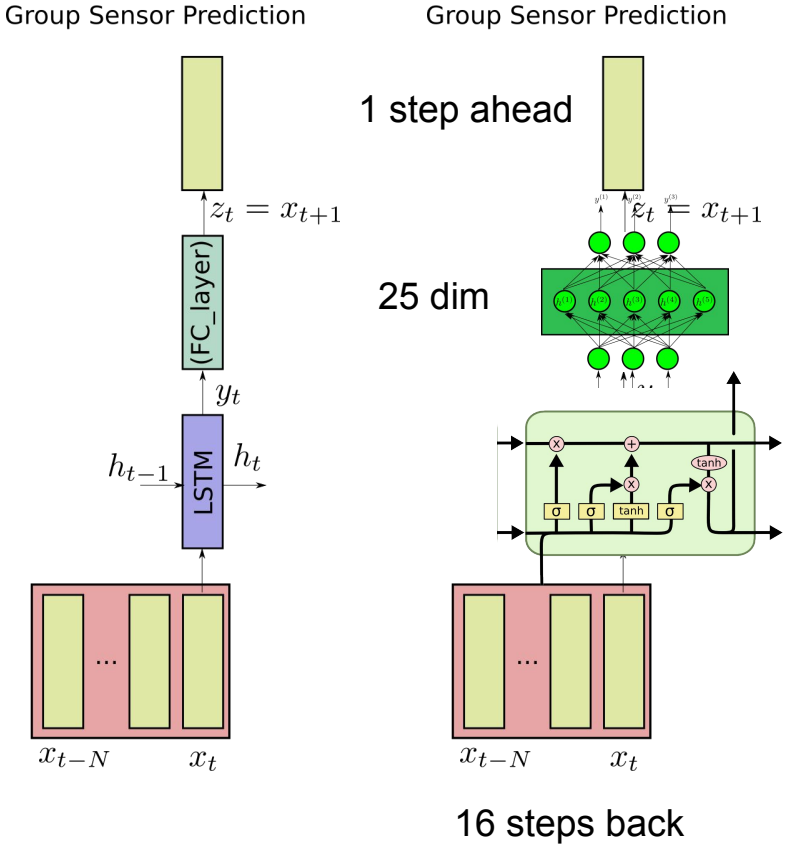
Qualitative Results of Training & Testing on the same house

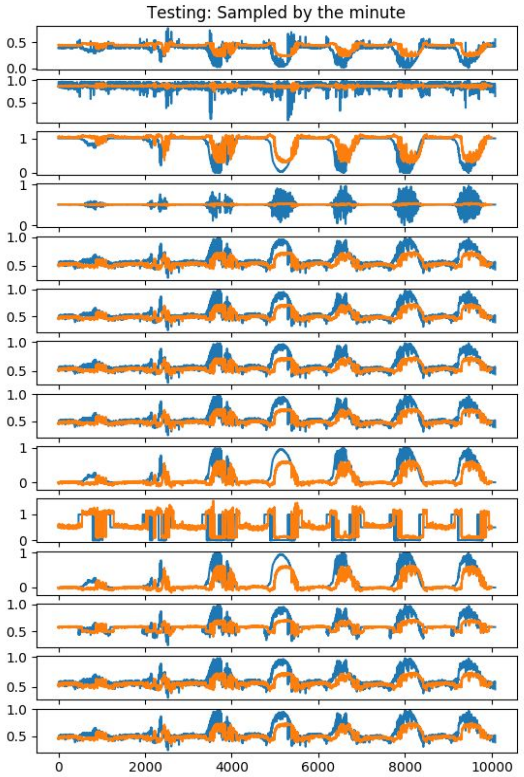


16 steps back

Solar Panel Sensor Data

Qualitative Results of Training & Testing on the same house





Ground Truth

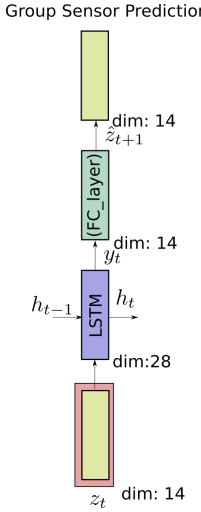
Prediction

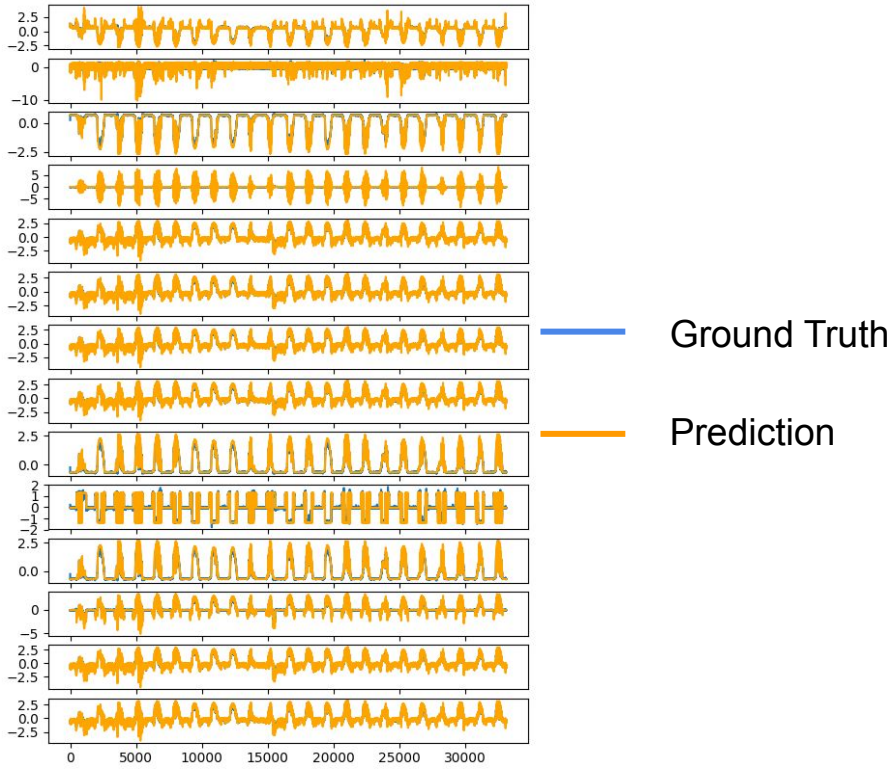
Solar Panel Sensor Data

But can we do better?

```
if training_on == 1:
    # Define Model:
    model = Sequential()
    hidden_state = feature_size * 2
    model.add(LSTM(hidden_state, batch_input_shape=(batch_size, 1, feature_size * look_back),
                   stateful=True, activation='tanh'))
    model.add(Dense(feature_size))
    model.compile(loss='mean_squared_error', optimizer='adam')

    # Train Model:
    for i in xrange(num_epochs):
        # EPOCHS HERE HAVE TO BE SET TO 1. WHAT DEFINES THE EPOCHS IS THE NUMBER OF LOOPS.
        model.fit(train_all_X, train_all_Y, batch_size=1, epochs=1, verbose=2, shuffle=False)
        # RESET STATE AFTER EACH LOOP IN THE LSTM (sweeps across the entire time series data)
        model.reset_states()
        print(i)
```

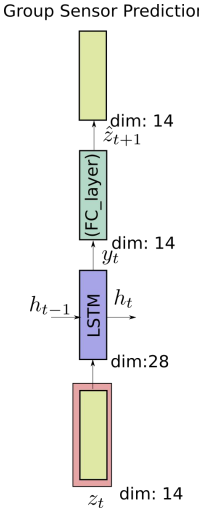




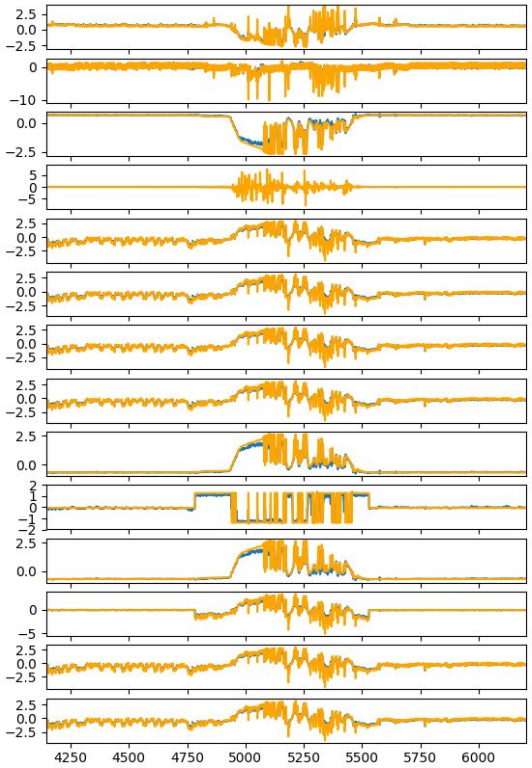
Solar Panel Sensor Data

```python
if training_on == 1:
    # Define Model:
    model = Sequential()
    hidden_state = feature_size * 2
    model.add(LSTM(hidden_state, batch_input_shape=(batch_size, 1, feature_size * look_back),
                   stateful=True, activation='tanh'))
    model.add(Dense(feature_size))
    model.compile(loss='mean_squared_error', optimizer='adam')

    # Train Model:
    for i in xrange(num_epochs):
        # EPOCHS HERE HAVE TO BE SET TO 1. WHAT DEFINES THE EPOCHS IS THE NUMBER OF LOOPS.
        model.fit(train_all_X, train_all_Y, batch_size=1, epochs=1, verbose=2, shuffle=False)
        # RESET STATE AFTER EACH LOOP IN THE LSTM (sweeps across the entire time series data)
        model.reset_states()
        print(i)
```



Zoom



Ground Truth

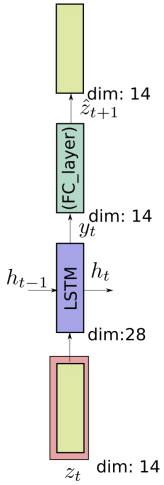
Prediction

Solar Panel Sensor Data

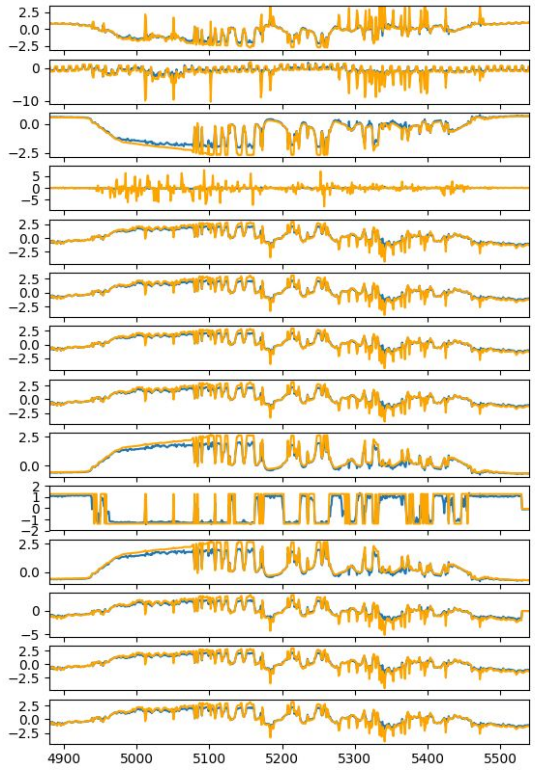
```python
if training_on == 1:
    # Define Model:
    model = Sequential()
    hidden_state = feature_size * 2
    model.add(LSTM(hidden_state, batch_input_shape=(batch_size, 1, feature_size * look_back),
                   stateful=True, activation='tanh'))
    model.add(Dense(feature_size))
    model.compile(loss='mean_squared_error', optimizer='adam')

    # Train Model:
    for i in xrange(num_epochs):
        # EPOCHS HERE HAVE TO BE SET TO 1. WHAT DEFINES THE EPOCHS IS THE NUMBER OF LOOPS.
        model.fit(train_all_X, train_all_Y, batch_size=1, epochs=1, verbose=2, shuffle=False)
        # RESET STATE AFTER EACH LOOP IN THE LSTM (sweeps across the entire time series data)
        model.reset_states()
        print(i)
```

Group Sensor Prediction



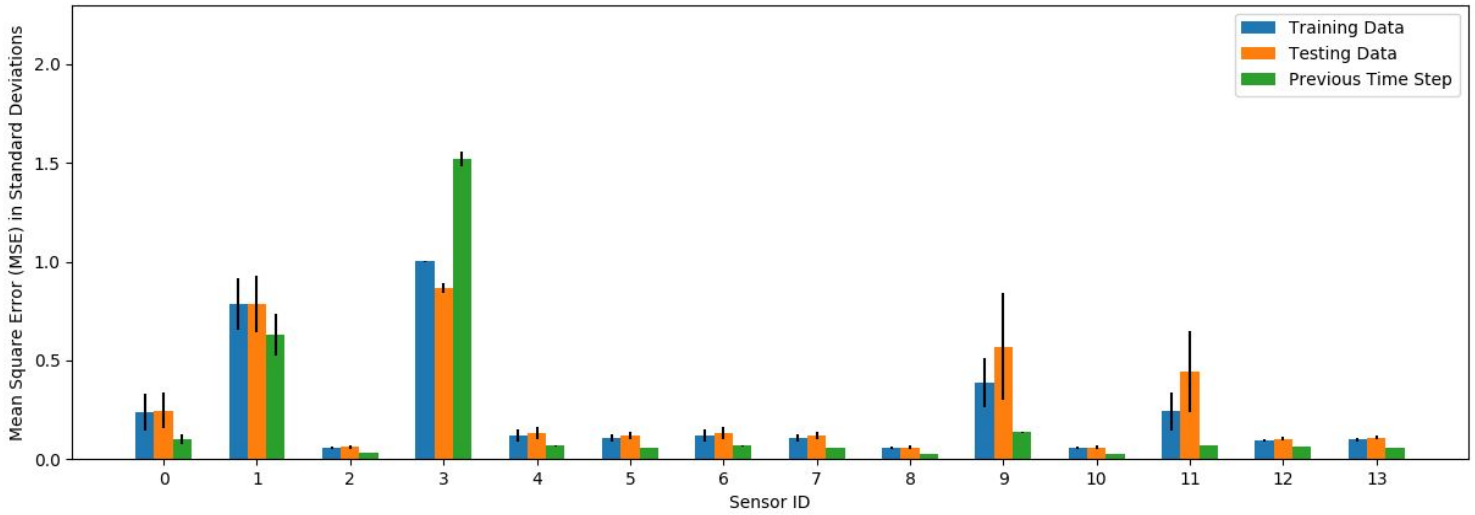
Zoom



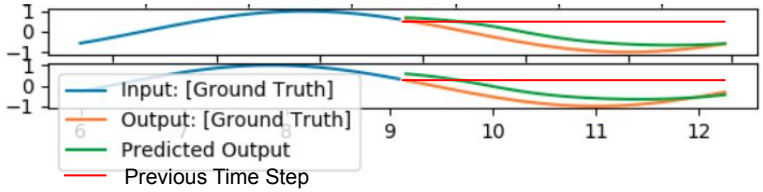
Ground Truth

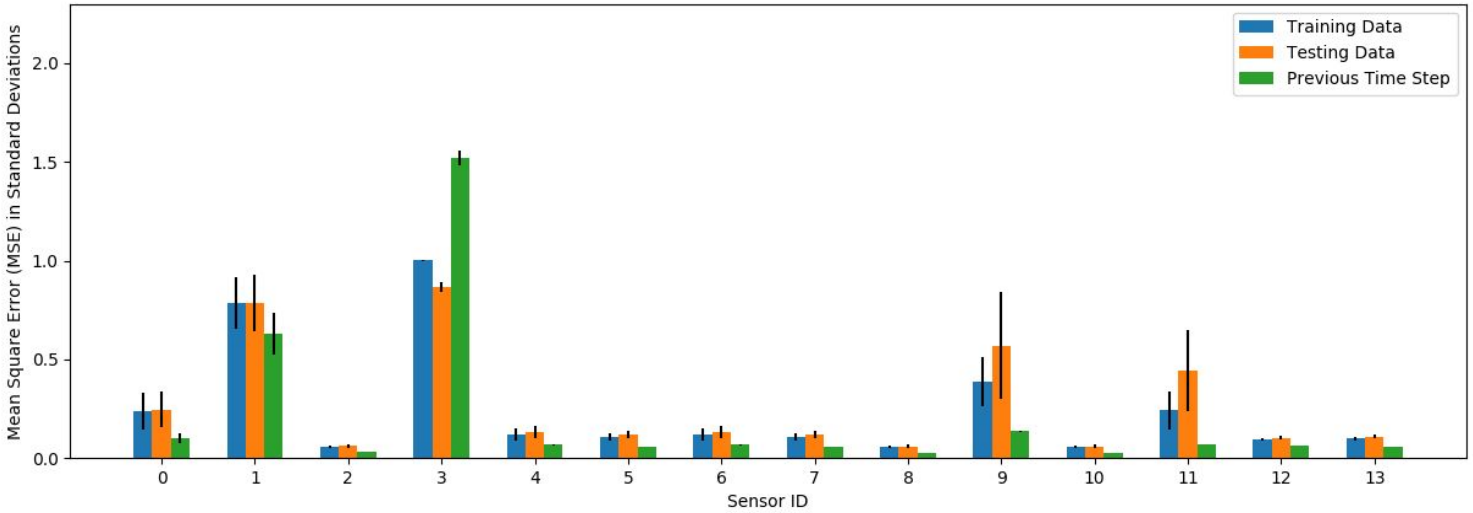
Prediction

Quantitative Results:



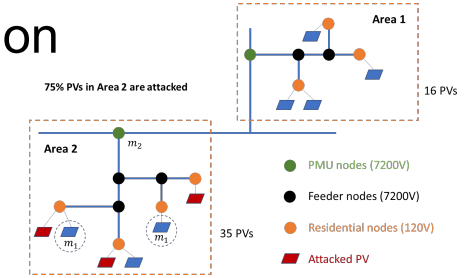
Quantitative Results:





Attacking/Compromising the Smart Grid and LSTM prediction

Disconnect Attack



House_meter_Name

time

solar_flux (W/sf)

$z_t^{(0)}$ P house (W)

$z_t^{(1)}$ Q house (Var)

$z_t^{(2)}$ P solar (W)

$z_t^{(3)}$ Q solar (Var)

$z_t^{(4)}$ V1 solar (V)

$z_t^{(5)}$ ph1 solar (rad)

$z_t^{(6)}$ V2 solar (V)

$z_t^{(7)}$ ph2 solar (rad)

$z_t^{(8)}$ I1 solar (A)

$z_t^{(9)}$ theta1 solar (rad)

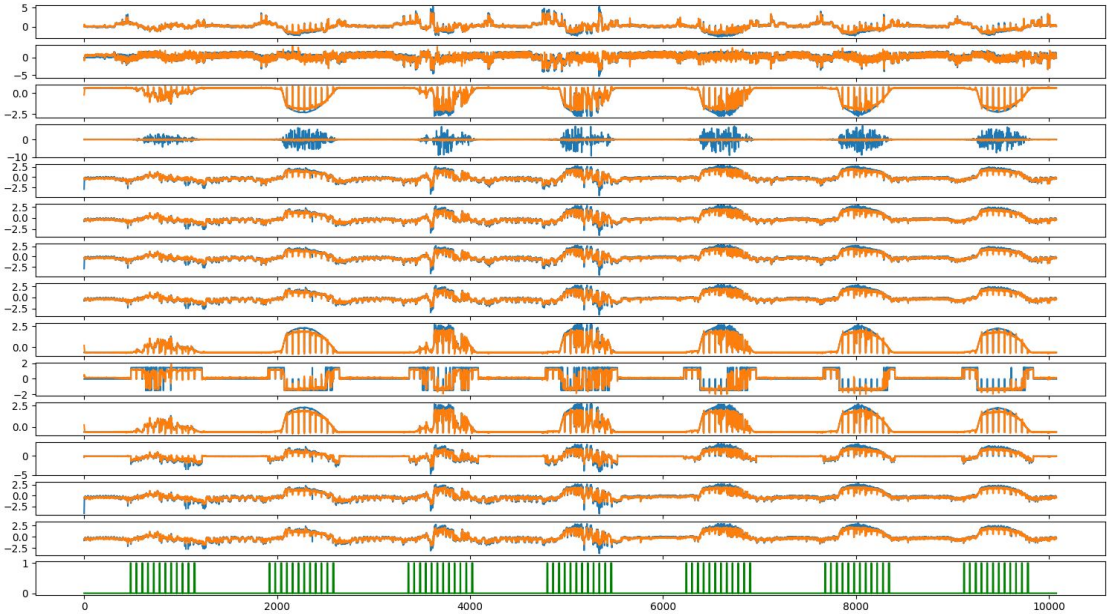
$z_t^{(10)}$ I2 solar (A)

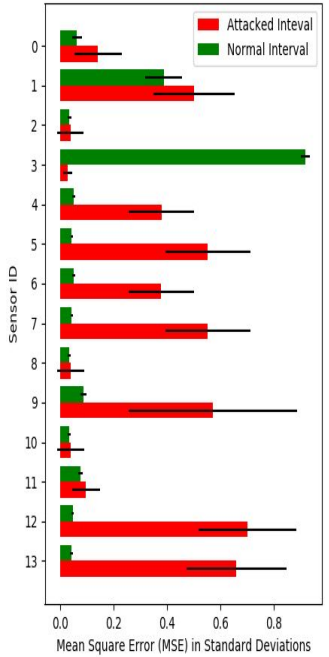
$z_t^{(11)}$ theta2 solar (rad)

$z_t^{(12)}$ node V (V)

$z_t^{(13)}$ node ph (rad)

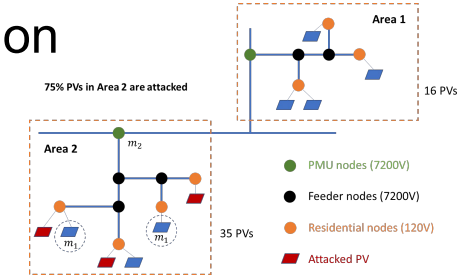
Attack Flag





Attacking/Compromising the Smart Grid and LSTM prediction

Load Attack



House_meter_Name

time

solar_flux (W/sf)

$z_t^{(0)}$ P house (W)

$z_t^{(1)}$ Q house (Var)

$z_t^{(2)}$ P solar (W)

$z_t^{(3)}$ Q solar (Var)

$z_t^{(4)}$ V1 solar (V)

$z_t^{(5)}$ ph1 solar (rad)

$z_t^{(6)}$ V2 solar (V)

$z_t^{(7)}$ ph2 solar (rad)

$z_t^{(8)}$ I1 solar (A)

$z_t^{(9)}$ theta1 solar (rad)

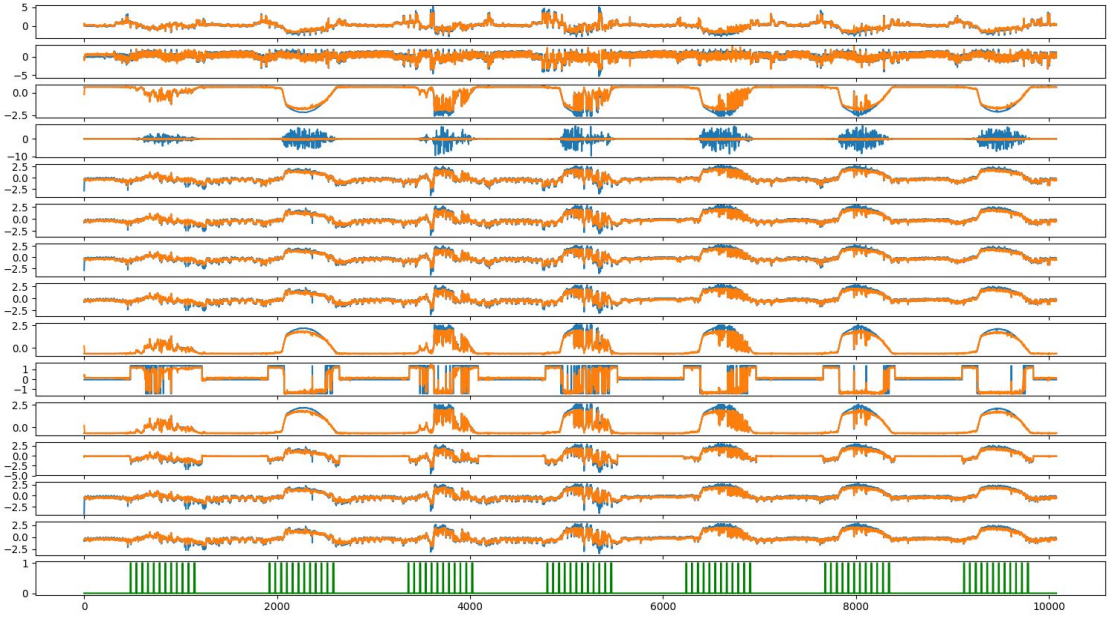
$z_t^{(10)}$ I2 solar (A)

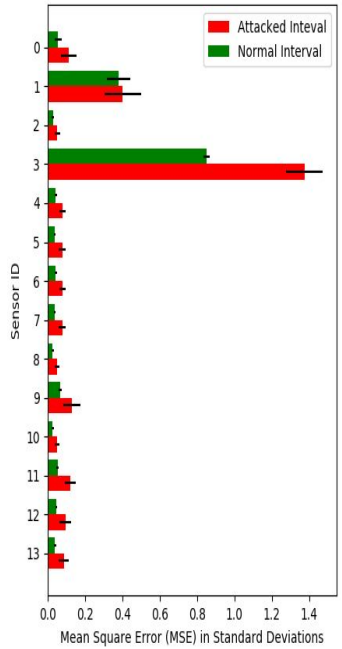
$z_t^{(11)}$ theta2 solar (rad)

$z_t^{(12)}$ node V (V)

$z_t^{(13)}$ node ph (rad)

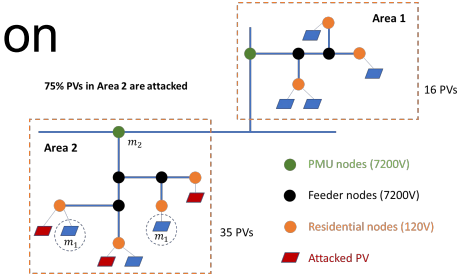
Attack Flag





Attacking/Compromising the Smart Grid and LSTM prediction

Power Attack



House_meter_Name

time

solar_flux (W/sf)

$z_t^{(0)}$ P house (W)

$z_t^{(1)}$ Q house (Var)

$z_t^{(2)}$ P solar (W)

$z_t^{(3)}$ Q solar (Var)

$z_t^{(4)}$ V1 solar (V)

$z_t^{(5)}$ ph1 solar (rad)

$z_t^{(6)}$ V2 solar (V)

$z_t^{(7)}$ ph2 solar (rad)

$z_t^{(8)}$ I1 solar (A)

$z_t^{(9)}$ theta1 solar (rad)

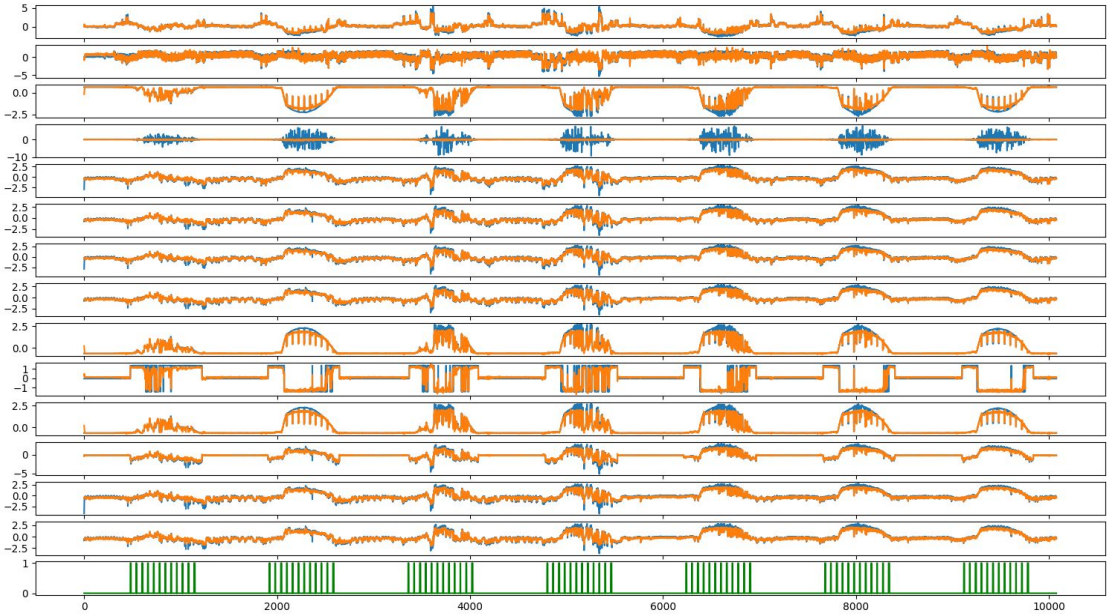
$z_t^{(10)}$ I2 solar (A)

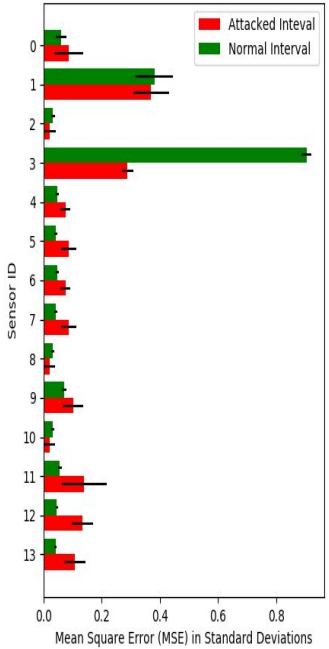
$z_t^{(11)}$ theta2 solar (rad)

$z_t^{(12)}$ node V (V)

$z_t^{(13)}$ node ph (rad)

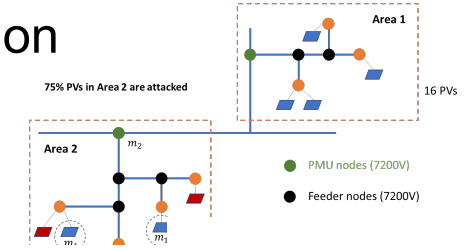
Attack Flag





Attacking/Compromising the Smart Grid and LSTM prediction

VAR Attack



House_meter_Name

time

solar_flux (W/sf)

$z_t^{(0)}$ P house (W)

$z_t^{(1)}$ Q house (Var)

$z_t^{(2)}$ P solar (W)

$z_t^{(3)}$ Q solar (Var)

$z_t^{(4)}$ V1 solar (V)

$z_t^{(5)}$ ph1 solar (rad)

$z_t^{(6)}$ V2 solar (V)

$z_t^{(7)}$ ph2 solar (rad)

$z_t^{(8)}$ I1 solar (A)

$z_t^{(9)}$ theta1 solar (rad)

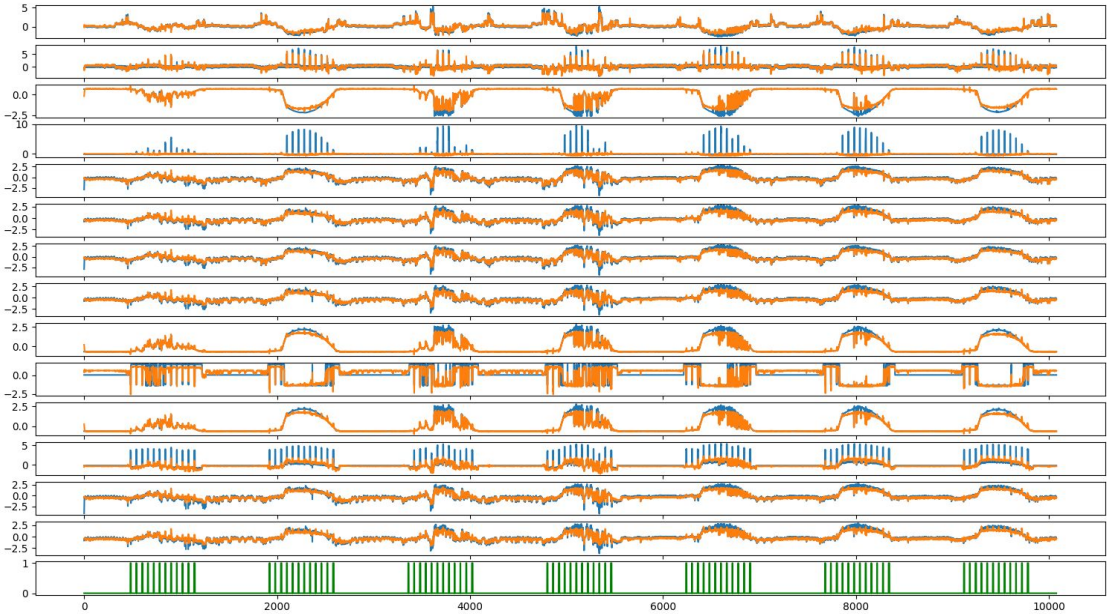
$z_t^{(10)}$ I2 solar (A)

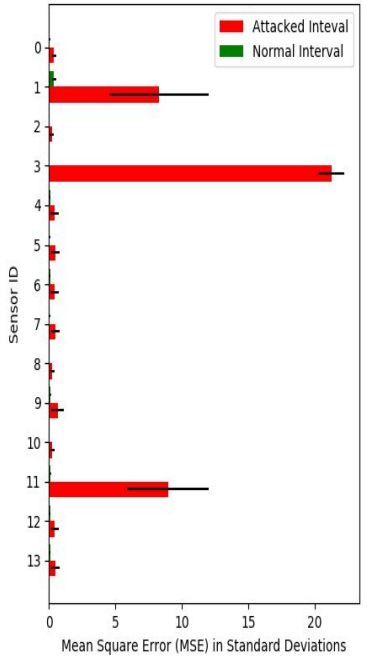
$z_t^{(11)}$ theta2 solar (rad)

$z_t^{(12)}$ node V (V)

$z_t^{(13)}$ node ph (rad)

Attack Flag

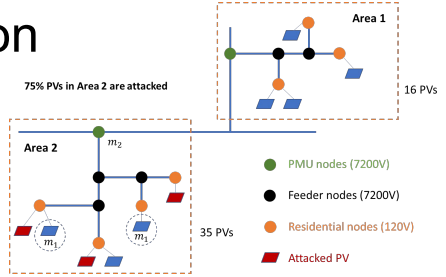


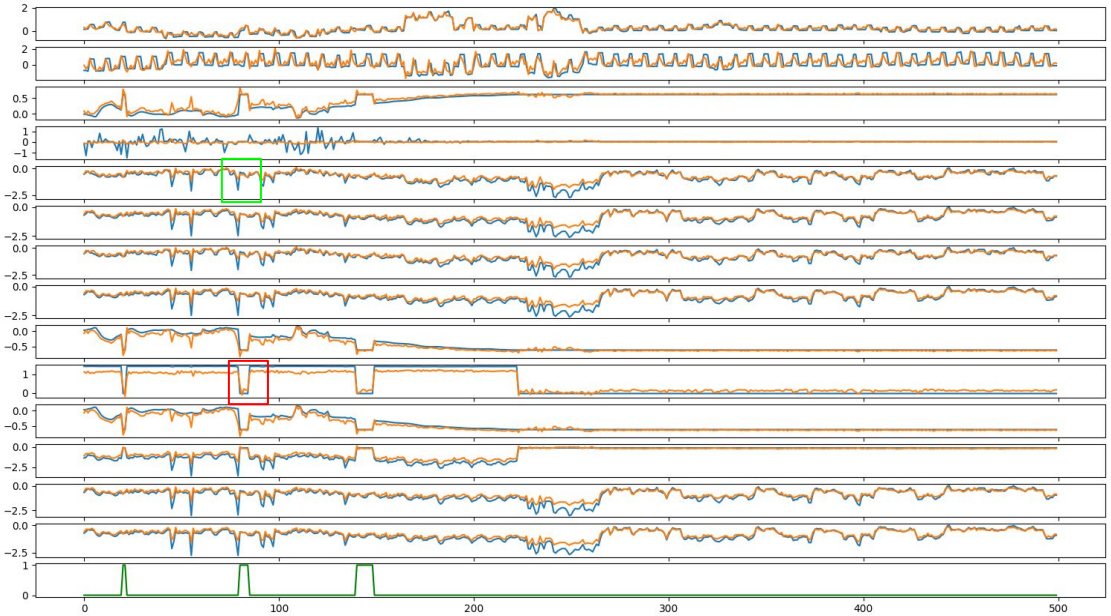


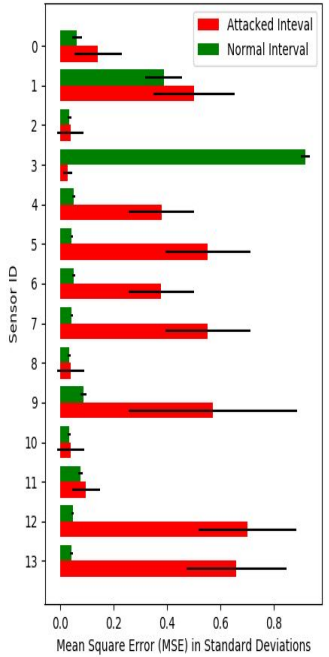
Zooms Across Attacks

Attacking/Compromising the Smart Grid and LSTM prediction

Disconnect Attack

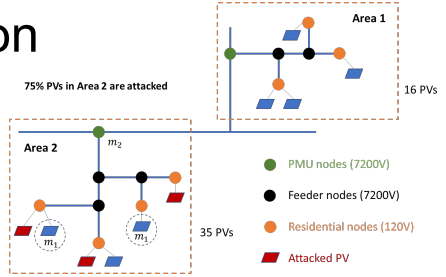


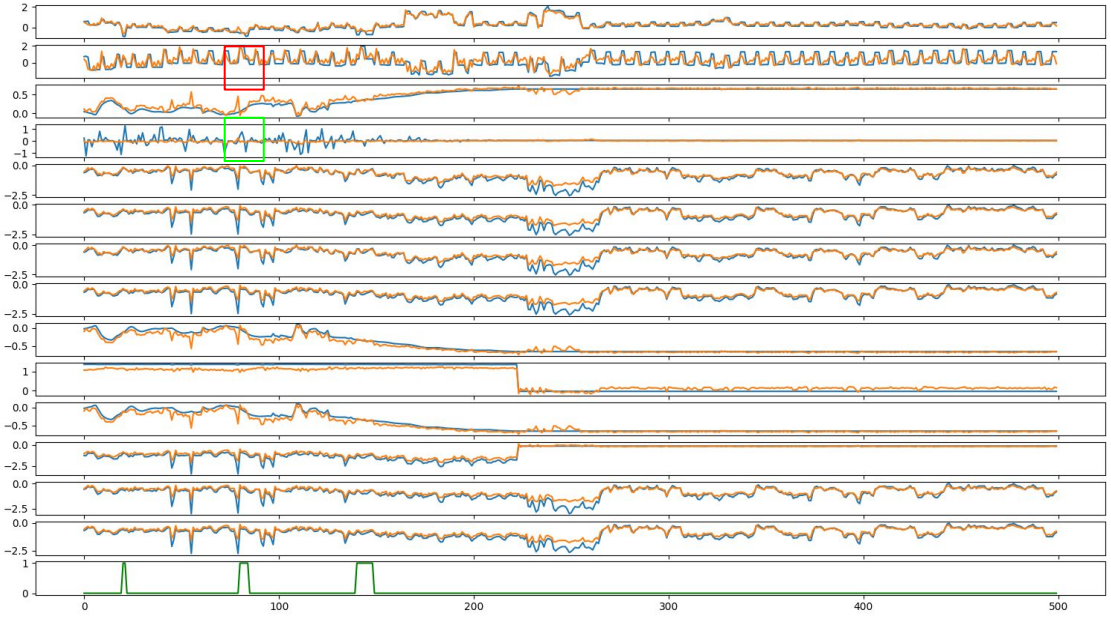


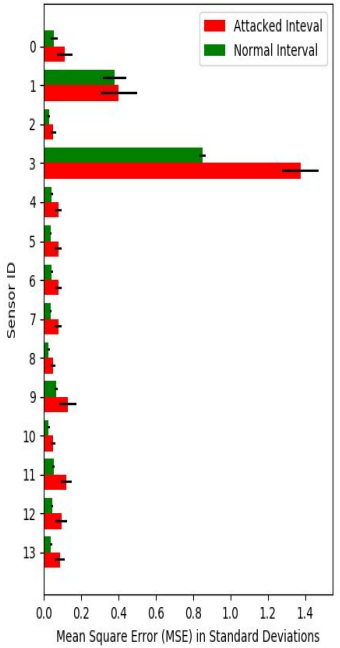


Attacking/Compromising the Smart Grid and LSTM prediction

Load Attack

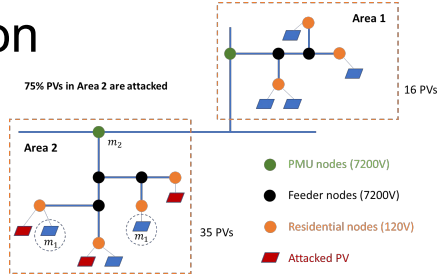


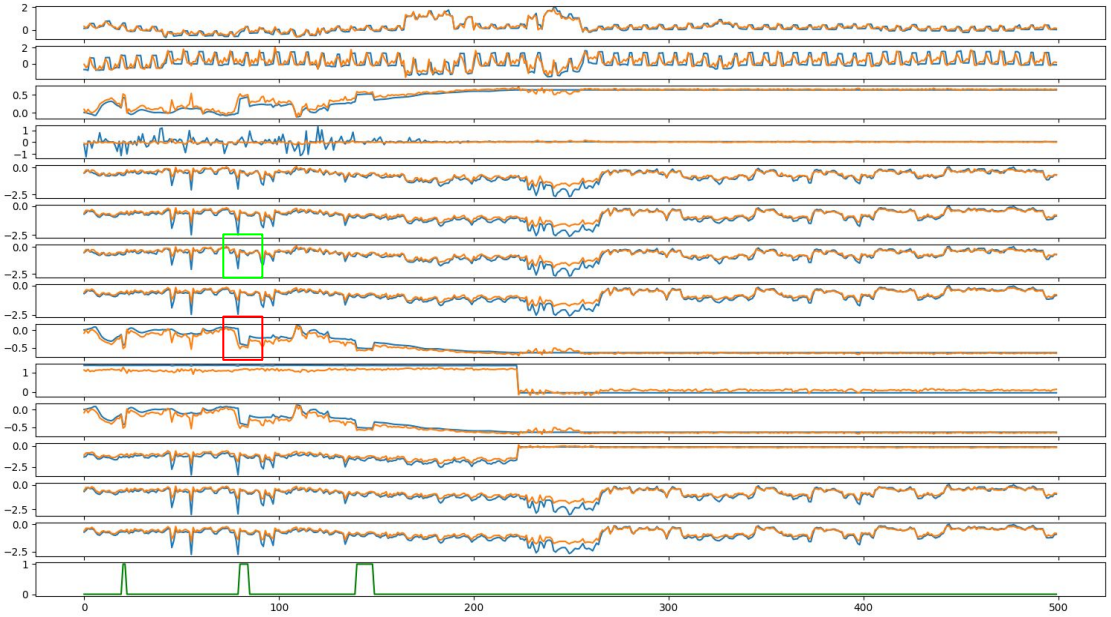


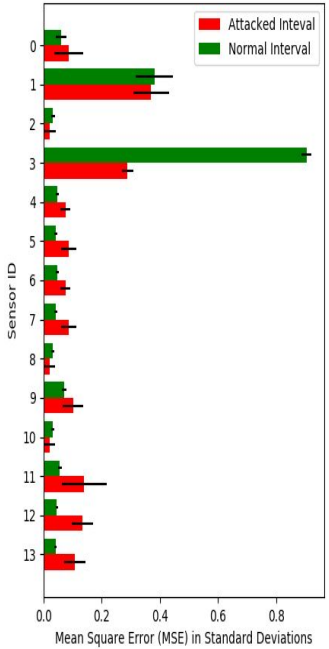


Attacking/Compromising the Smart Grid and LSTM prediction

Power Attack

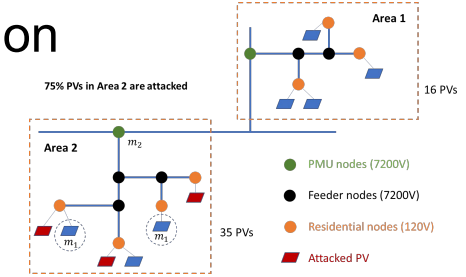


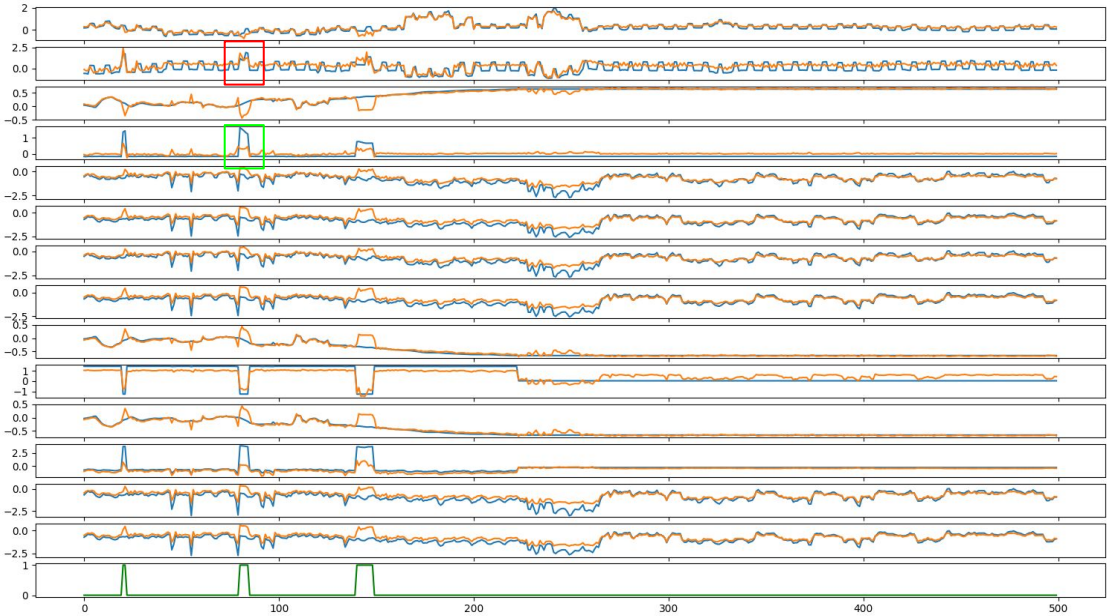


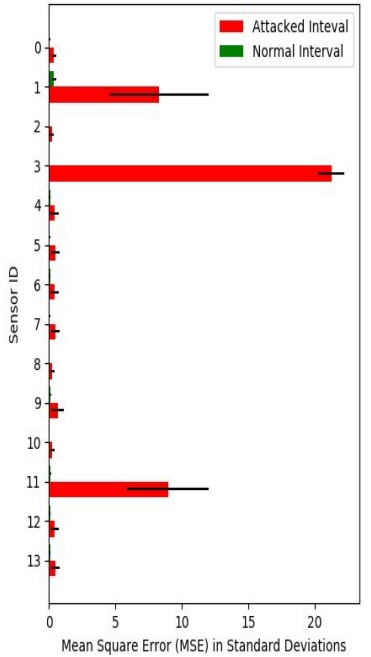


Attacking/Compromising the Smart Grid and LSTM prediction

VAR Attack



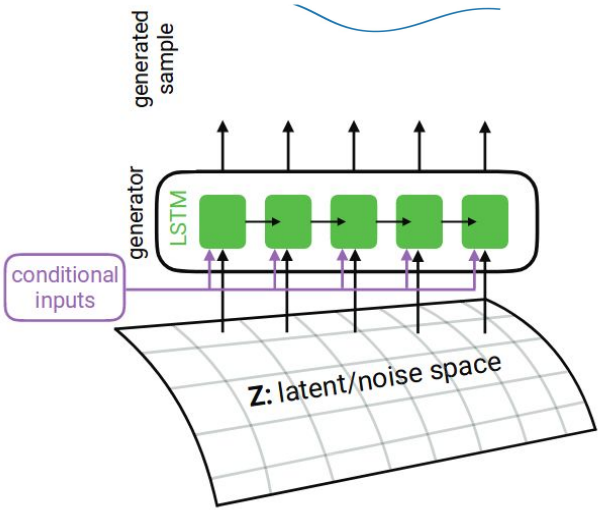




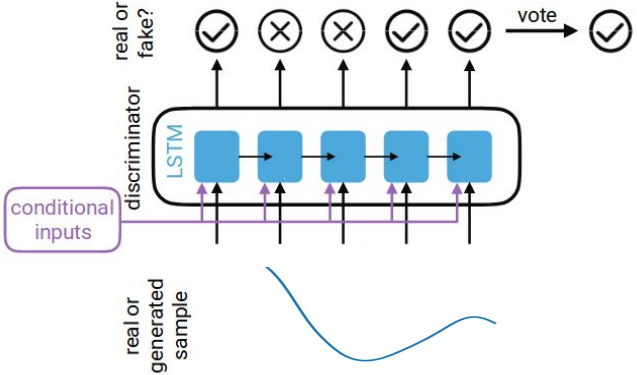
Other Interesting Mixes : GANS + RNNs

Real-Time valued GANs for medical time series data

Esteban et al. (2017)



(a) The generator RNN takes a different random seed at each temporal input, and produces a synthetic signal. In the case of the RCGAN, it also takes an additional input on each time step that conditions the output.



(b) The discriminator RNN takes real/synthetic sequences and produces a classification into real/synthetic for each time step. In the case of the RCGAN, it also takes an additional input on each time step that conditions the output.

Figure 1: Architecture of Recurrent GAN and Conditional Recurrent GAN models.

Disentangled Representations Networks (DrNet)

Denton & Birodkar (NeurIPS, 2017)

Unsupervised Learning of Disentangled Representations from Video

Remi Denton
Department of Computer Science
New York University
denton@cs.nyu.edu

Vighnesh Birodkar
Department of Computer Science
New York University
vighneshbirodkar@nyu.edu

Abstract

We present a new model DrNet that learns disentangled image representations from video. Our approach leverages the temporal coherence of video and a novel adversarial loss to learn a representation that factorizes each frame into a stationary part and a temporally varying component. The disentangled representation can be used for a range of tasks. For example, applying a standard LSTM to the time-vary components enables prediction of future frames. We evaluate our approach on a range of synthetic and real videos, demonstrating the ability to coherently generate hundreds of steps into the future.

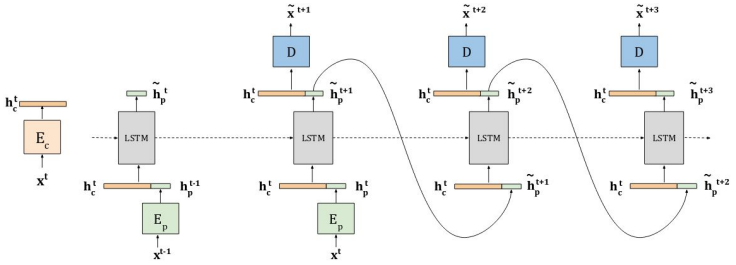
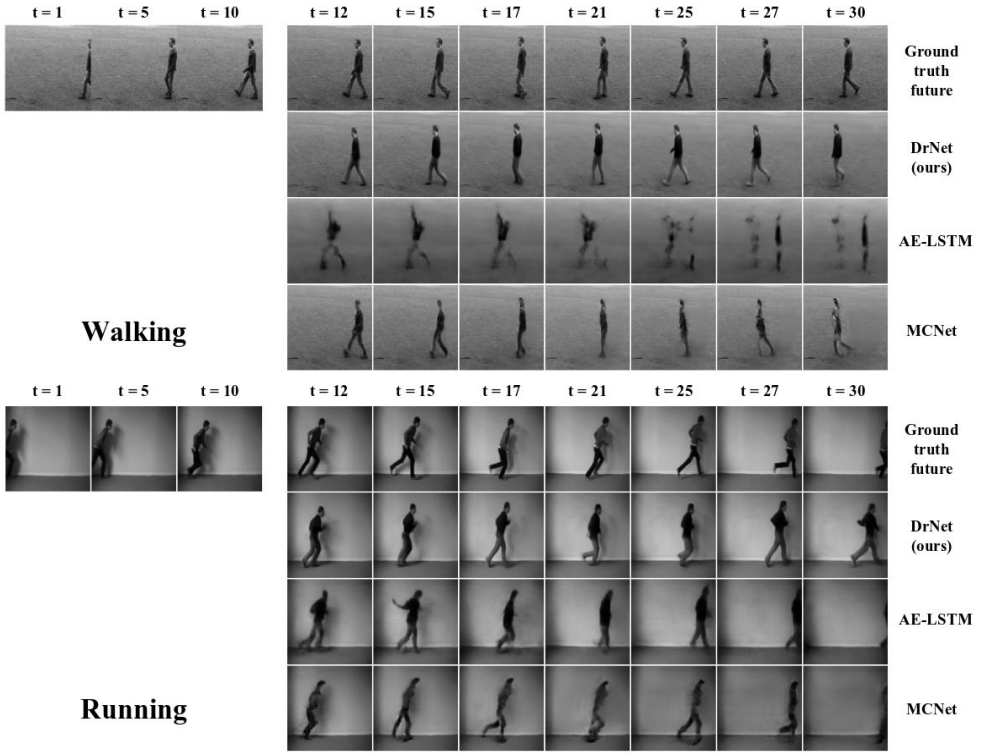


Figure 2: Generating future frames by recurrently predicting h_p, the latent pose vector.



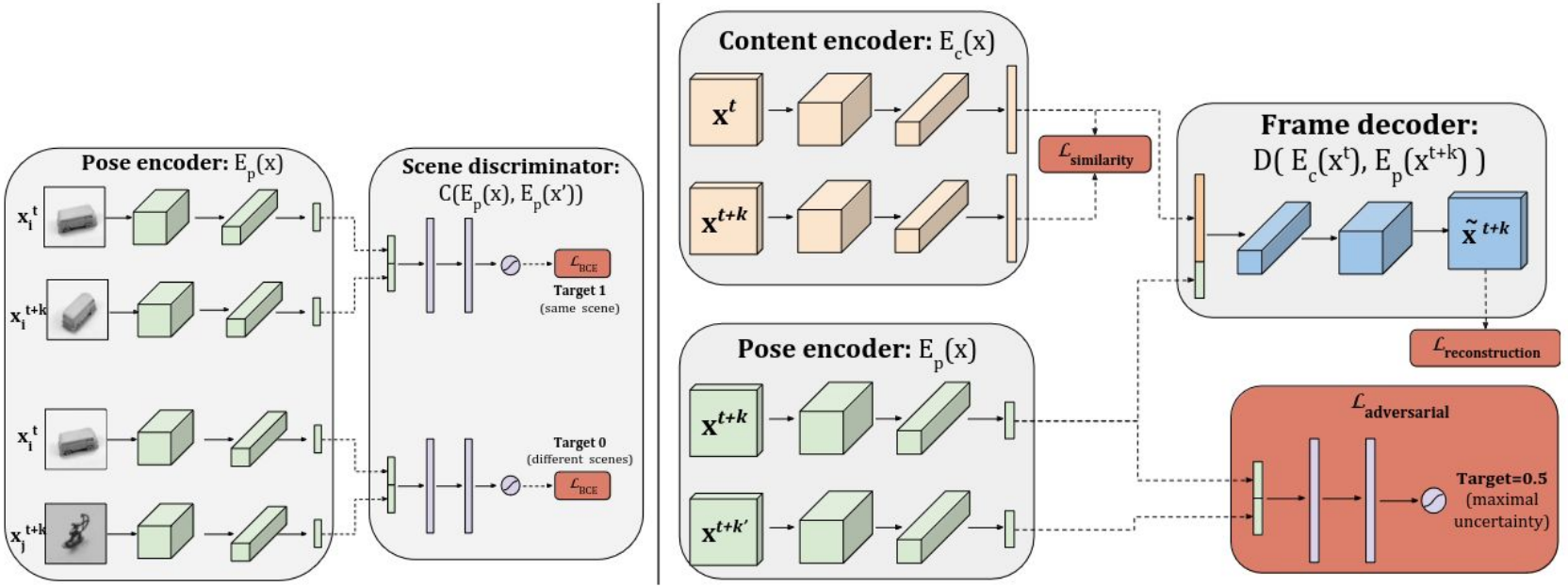


Figure 1: Left: The discriminator C is trained with binary cross entropy (BCE) loss to predict if a pair of pose vectors comes from the same (top portion) or different (lower portion) scenes. x_i and x_j denote frames from different sequences i and j. The frame offset k is sampled uniformly in the range $[0, K]$. Note that when C is trained, the pose encoder E_p is fixed. Right: The overall model, showing all terms in the loss function. Note that when the pose encoder E_p is updated, the scene discriminator is held fixed.

Training/Validation/Testing for Temporal Models

Splitting Time Series Data into Train/Test/Validation Sets

Asked 6 years ago Modified 1 year, 5 months ago Viewed 35k times

27

What's the best way to split time series data into train/test/validation sets, where the validation set would be used for hyperparameter tuning?

We have 3 years' worth of daily sales data, and our plan is to use 2015-2016 as the training data, then randomly sample 10 weeks from the 2017 data to be used as the validation set, and another 10 weeks from 2017 data for the test set. We'll then do a walk forward on each of the days in the test and validation set.

time-series cross-validation validation

Share Cite Improve this question

Follow

edited May 18, 2018 at 6:10

asked May 18, 2018 at 3:53

meraxes

739 2 7 19

Add a comment

Training/Validation/Testing for Temporal Models

PREDICT THE NEXT DAY USING PREVIOUS DAYS DATA