DRAPER

Echo State Networks for Time Series Data

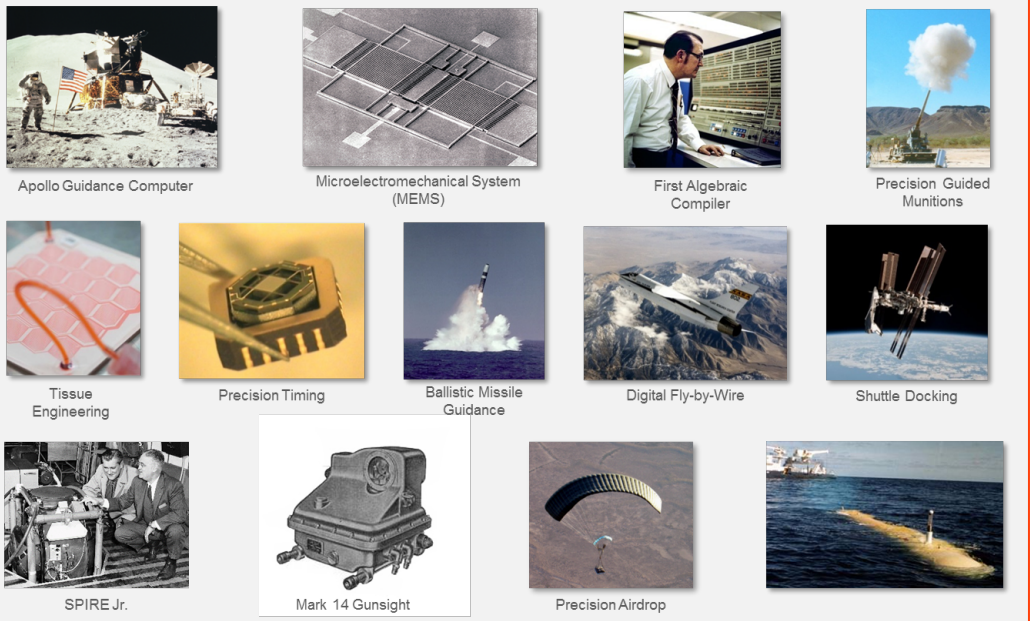
Open Data Science Conference, Boston

Teal Guidici, PhD

April 16th, 2020

The Charles Stark Draper Laboratory, Inc.
555 Technology Square, Cambridge Mass. 02139-3563

About Draper



Independent not-for-profit research and development company

- Headquartered in Cambridge, MA
- 1500+ employees, $520 million revenue
- Serving the nation since 1932
- Independent organization since 1973

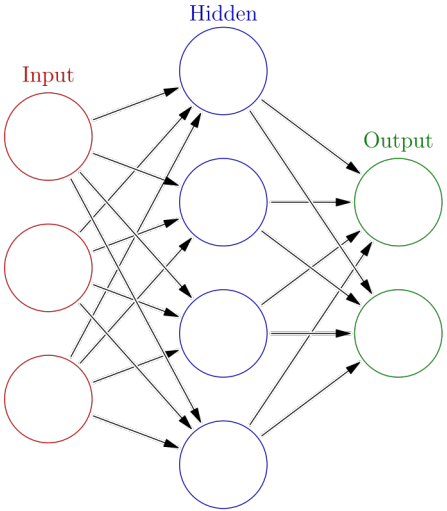
Outline

- Neural Networks in general
- Echo State Networks in detail
- Demo: https://github.com/tguidici/ODSC_2020_ESN
- Example: ESN for finance

Neural Networks (NN)

- Loosely (!) modeled on human/animal brains
  - Many simple processing nodes
  - Densely connected (usually)
  - “Learn” to perform a task by analyzing training examples

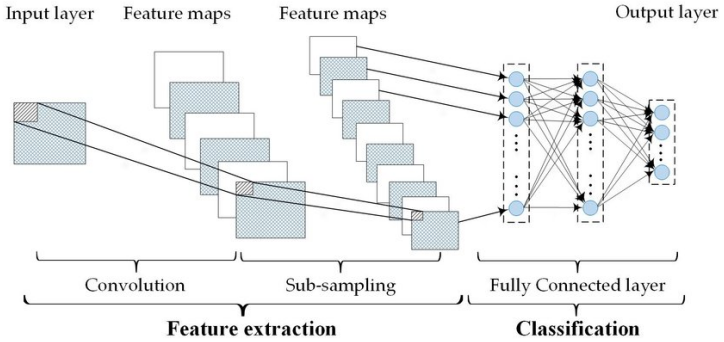
- First proposed in 1944; on a boom-bust cycle of interest
  - Dropped out of favor ~ 1969
  - Resurgence in 1980’s!
  - Disinterest in 2000’s!
  - ALL THE INTEREST IN 2010’s!!



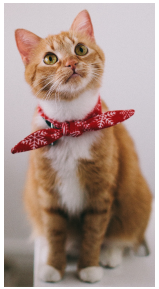
- Circular nodes represent artificial neurons
- Arrows illustrate connections between output of one neuron to input of another

Image credit: By Glosser.ca - Own work, Derivative of File:Artificial neural network.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24913461

Convolutional vs Recurrent Neural Nets



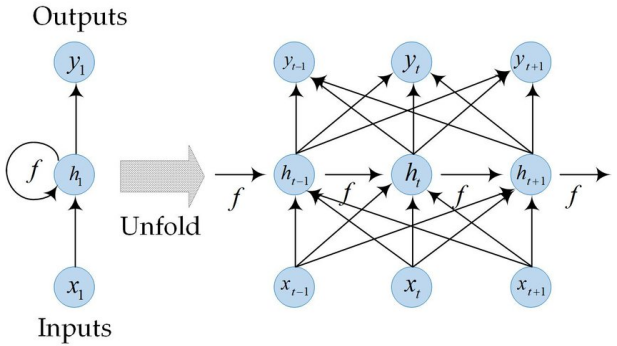
Convolutional Neural Net



Spatial structure matters

eg: image classification

Am I a CAT or a COWPOKE?



Recurrent Neural Net (RNN)

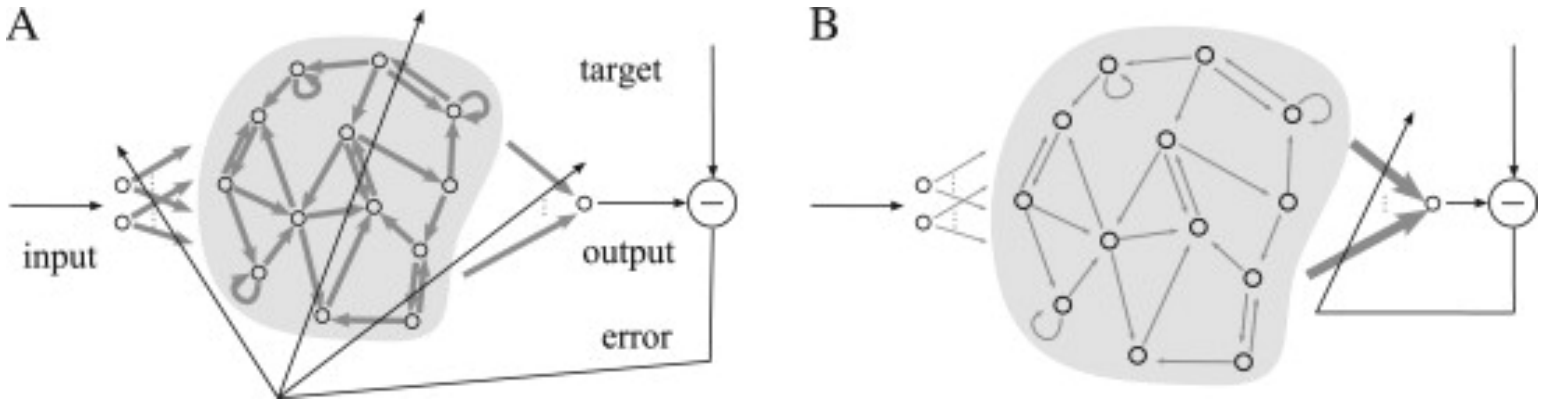
Sequential structure matters

eg: weather prediction

It rained the last two days, will it rain today?

Neural Net Images From: Zhang, Sushen & Hosseini Bamakan, Seyed Mojtaba & Qu, Qiang & Li, Sha. (2018). Learning for Personalized Medicine: A Comprehensive Review From a Deep Learning Perspective. IEEE Reviews in Biomedical Engineering. PP. 1-1. 10.1109/RBME.2018.2864254.

Reservoir Computing – a better RNN



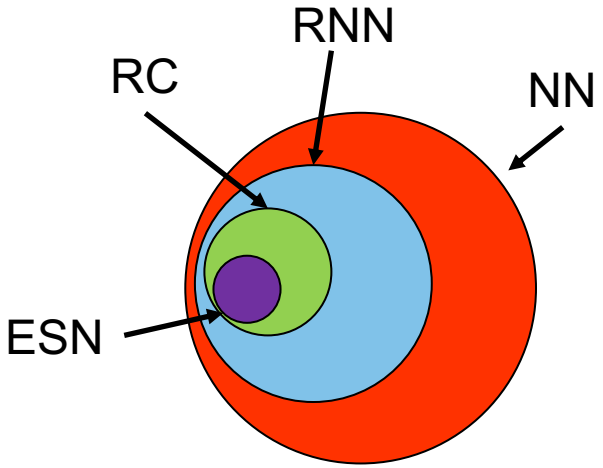
Bold arrows indicate trained weights

A. Traditional RNN: all connection weights adapted
   - Computationally intensive, hard to tune
B. Reservoir Computing (RC): only output weights are adapted
   - Fast! Accurate! Still challenging to tune

Image and caption from Lukoševičius, Mantas & Jaeger, Herbert. (2009). Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review 3, 127-149. Computer Science Review. 3. 127-149. 10.1016/j.cosrev.2009.03.005.

Echo State Networks (ESN) – introduction!

- ESNs are a recurrent neural network
  - Captures **dynamic behavior** of a time series or sequential data.
  - Internal state (**memory**) to process sequences of inputs and remember context
- ESNs are also type of reservoir computing
  - **Faster training** by only learning the output weights vs. back propagation approaches
  - Captures more complex interactions by manipulating topologies and neurons within network
  - Non-linear transformation of inputs into higher dimension of reservoir
  - Output weights are trained ~ “what linear combination of reservoir elements best approximates my target?”



ESNs are a type of Reservoir Computing, which is a subset of RNNs, which are a type of Neural Network

ESN: Overview

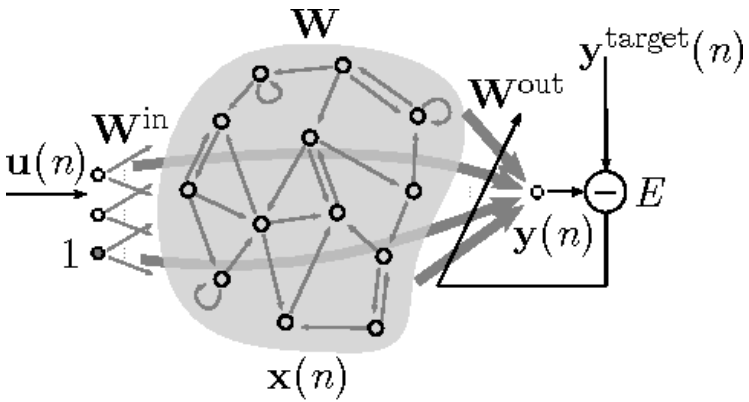


Fig. 1: An echo state network.

For each time step n

- u(n): input
- x(n): reservoir activations
- y(n): network output
- $y^{target}(n)$: desired output

Randomly Initialized & fixed

- W^{in}: input weights
  - Linearly map input into reservoir
- W: recurrent weights
  - How are the reservoir neurons connected to each other?

Learned parameters

- W^{out}: output weights
  - Linearly map (u(n), x(n)) to y(n)
  - Minimize Error($y^{target}(n)$, y(n))

Image From: M. Lukoševičius (2012): **A Practical Guide to Applying Echo State Networks.** In: G. Montavon, G. B. Orr, and K.-R. Müller (eds.) Neural Networks: Tricks of the Trade, 2nd ed. Springer LNCS 7700, pp 659-686

ESN: Overview

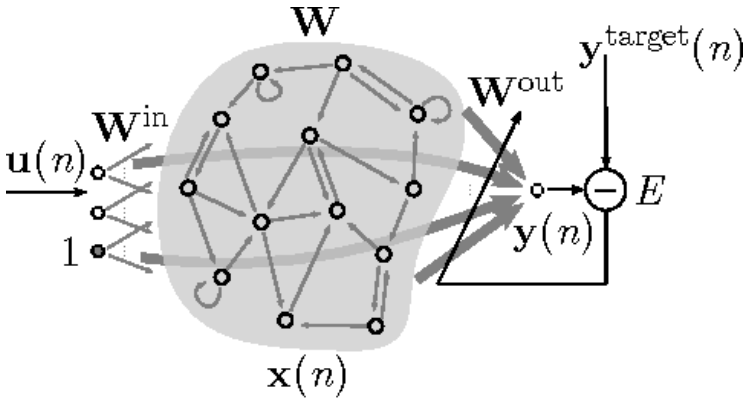


Fig. 1: An echo state network.

How does it work?

1. **Setup the network!** : Generate a large random reservoir RNN with
   - Input weights W^{in}
   - Recurrent weights W
   - Leak rate α
2. **Map to higher dimension!**: Run input u(n) through network and collect activation states x(n)
   - Non-linear high dimensional transform!
3. **Optimize!**: Compute W^{out} such that W^{out} minimizes Error(y(n), y^{target}(n))
   - The only training we need!
   - Error minimized across all time points
4. **Predict something new!**: Trained network can be used to compute predicted y(n) from new u(n)

Definitions and update equations

$W^{in} \in \mathbb{R}^{N_x \times (1+N_u)}$: input weights

$W \in \mathbb{R}^{N_x \times N_x}$: recurrent weights

$\alpha \in (0,1]$: leak rate

$x(n) \in \mathbb{R}^{N_x}$: neuron activations

$\tilde{x}(n) \in \mathbb{R}^{N_x}$: neuron updates

$$x(n) = (1-\alpha)x(n-1) + \alpha\tilde{x}(n)$$

$$\tilde{x}(n) = tanh(W^{in}[1; u(n)] + Wx(n-1))$$

Key Reservoir Parameters: Input Weights

$$W^{in} \in \mathbb{R}^{N_x \times (1+N_u)}$$
$$W^{in}_{ij} \sim Unif(-a, a)$$
Tune/scale input weights by varying a

- Generally drawn from a Uniform distribution, but not required!
- Input scaling determines how nonlinear reservoir responses are
  - a <1 (generally)
- Scaling input weights helps remove the influence of outliers.
- Different sets of inputs can be scaled differently if the corresponding portions of u(n) contribute differently to task

Key Reservoir Parameters: Spectral Radius

$$\boldsymbol{W} \in \mathbb{R}^{\boldsymbol{N_x} \times \boldsymbol{N_x}}$$

$$\boldsymbol{\rho(W)} = \textbf{ spectral radius of W}$$

- Spectral radius = eigenvalue of W with greatest magnitude
- W defines which reservoir neurons are connected and how
- $\rho(W) < 1$ ensures echo state property. But not always necessary.
- y(n) requires extensive historical knowledge? Large $\rho(W)$
- y(n) depends on more recent history? Small $\rho(W)$
- Practical approach: Select $\rho(W)$ that maximizes accuracy, using $\rho(W)=1$ as initial starting value

Key Reservoir Parameters: Leak Rate

$$\alpha \in (0, 1]\text{: leak rate}$$
$$x(n) = (1 - \alpha)x(n-1) + \alpha\tilde{x}(n)\text{: neuron activations}$$
$$\tilde{x}(n) = tanh(W^{in}[1; u(n)] + Wx(n-1))\text{: neuron updates}$$

- Captures temporal dynamics of system
- ”speed of reservoir update dynamics discretized in time”
- Goal: set α to match the speed of the dynamics of u(n) and/or $y^{target}(n)$
- Different components of u(n) have different speed dynamics? No problem!

$$\alpha \in \mathbb{R}^{N_x}$$

Other reservoir parameters

Can generally be set to default/“reasonable” values

- Sparsity of the reservoir (connectivity)
  - Reservoirs should be very sparsely connected, regardless of size.
    - Potentially each neuron only connected to 10 others!
  - This sparsity is what makes it fast! fast! fast!
- Size of the reservoir (number of neurons)
  - Pick as many as you think you’ll need computationally
  - But you probably need fewer than you think (10’s, not 100’s)
- Distribution used to generate weights for W
  - Uniform distribution is probably good enough.
  - Can also be used for generating weights for W^{in}

More choices – Which error?!

Output weights, W^{out}, computed to minimize error….but which error?

$$W^{out} = \underset{W^{out}}{\operatorname{argmin}} E(y, y^{target})$$

- Simplest: Mean Squared Error
- Potential for overfitting or feedback instability? Ridge Regression
- No overfitting and lots of compute power? Pseudoinverse
- Some time points more (less) important than others? Regression Weighting
  - Weight more (less) heavily error for important timepoints

DRAPER®

ESN DEMO

DEMO

DEMO can be found at: https://github.com/tguidici/ODSC_2020_ESN

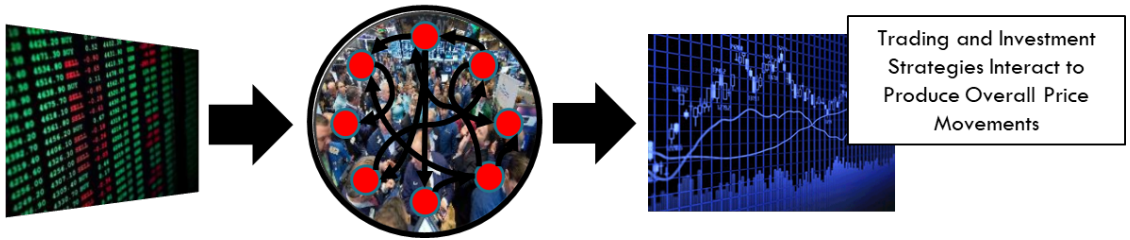
DRAPER®

ESN for Finance

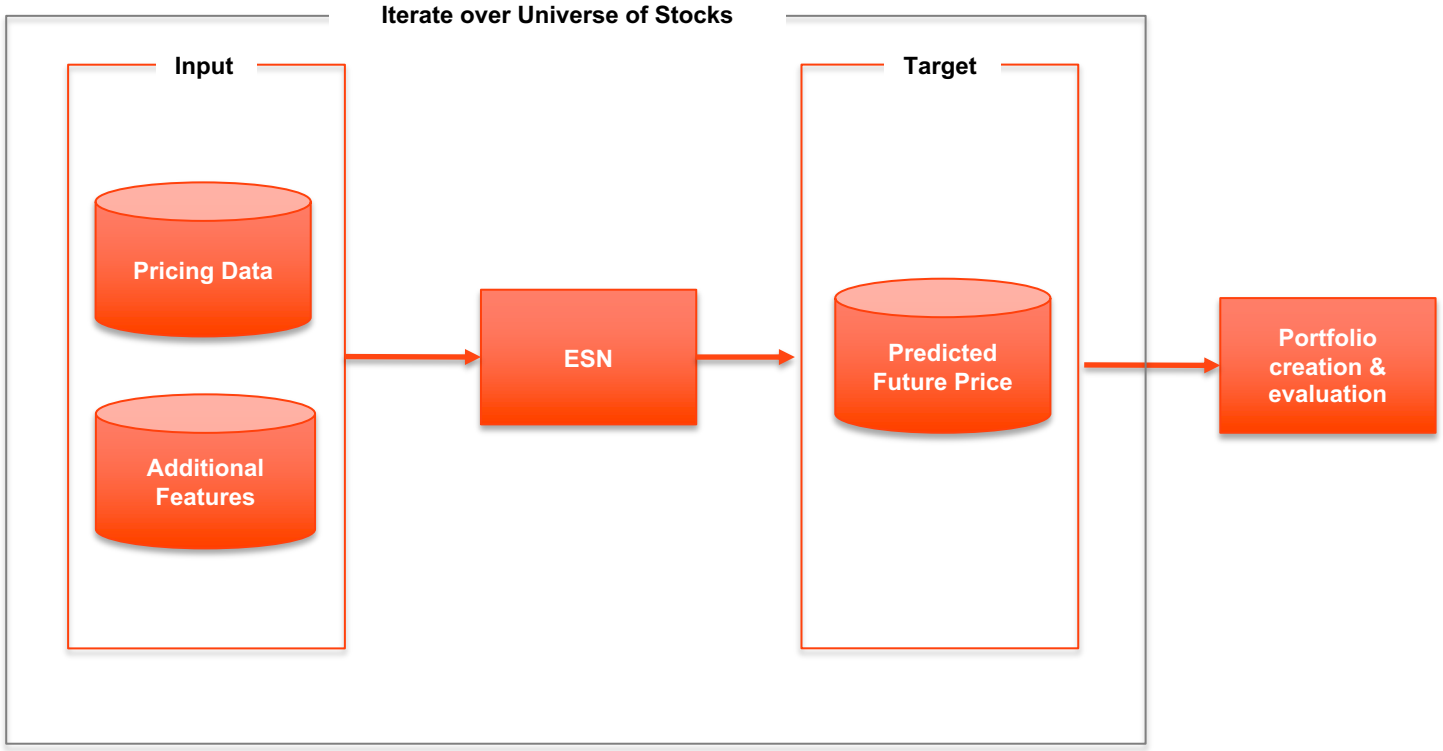
Echo State Networks for Market Modeling

1. The market is a complex system
    - Large system of non linearly interacting information and behaviors.
    - Temporal/sequential information crucial (“memory”)
2. Echo State Networks are uniquely capable of modeling non-linear, interacting dynamics with temporal dependencies.

Theory : The Stock Market is a Network of Interacting Strategies
Input – Market, company news, external news, world events, opinions
Computation – Complex system dynamics of strategies
Output – Predicted stock price movement and corresponding volume

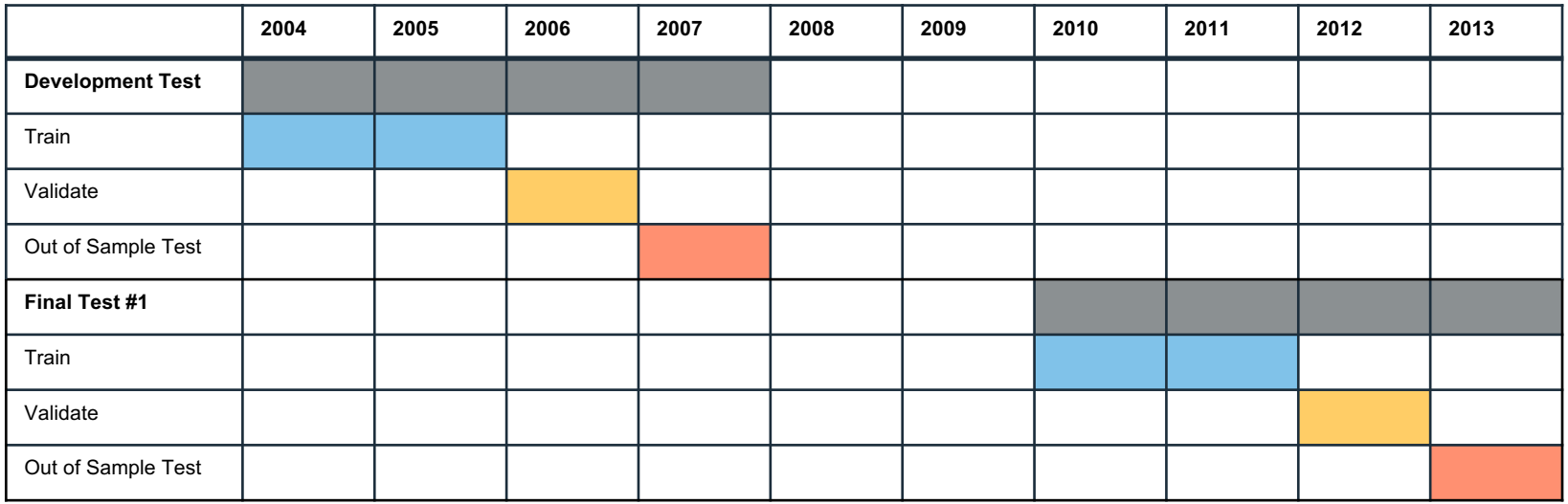


ESN for Market Modeling pipeline



Test Setup: parameter tuning & data leakage

Data Leakage: when your model knows more than it should



- All development tests in the pre-2010 timeframe
- True out-of-sample tests in the post-2010 timeframe
- Final tests were executed a single time using parameters determined during development

ESN for Market Modeling: data leakage

Data Leakage: when your model knows more than it should

What we did well:

- Held out “final” datasets for test of the entire pipeline
- Separate training and validation datasets for hyperparameter tuning
- Temporal buffer to separate validation and test datasets

What we didn’t do well:

- Optional PCA featurization used data before splitting into train/validate/test
  - features in training data “knew” something about variance of test data (Caught this error before final tests!)

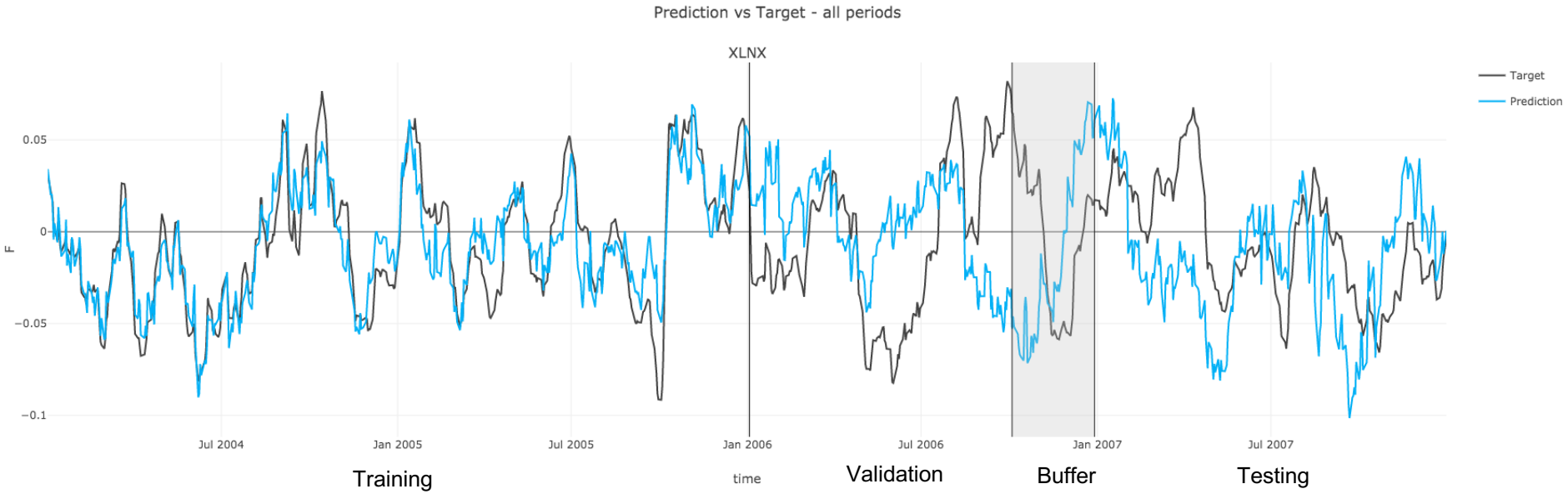
Highly recommend ODSC talk on data leakage from Yuriy Guts – Friday!
https://github.com/YuriyGuts/odsc-target-leakage-workshop

Automated hyperparameter tuning

ESNs have many hyperparameters – how to pick good values?
We automated the pipeline to iterate over a ”reasonable” range for select parameters, pick values with lowest error, on average.

- Training:
  - Spectral radius: tune over reservoir
  - Input scaling: tune over inputs
- Validation:
  - Leak rate: tune over temporal dynamics of input

ESN for Finance – Results



Portfolios constructed based on ESN predictions were robust and performed well

DRAPER

Where to next?

- Ensemble or on-line training
  - Data generating process not stationary? Model needs to adapt
- Output feedback
  - Trained readouts fed back into reservoir to change dynamics
- Semi-structured network topologies
  - Groups of neurons linked to specific inputs
- Different error metrics
  - Some types of error matter more than others (eg: when predicting counts, output should not be negative….)