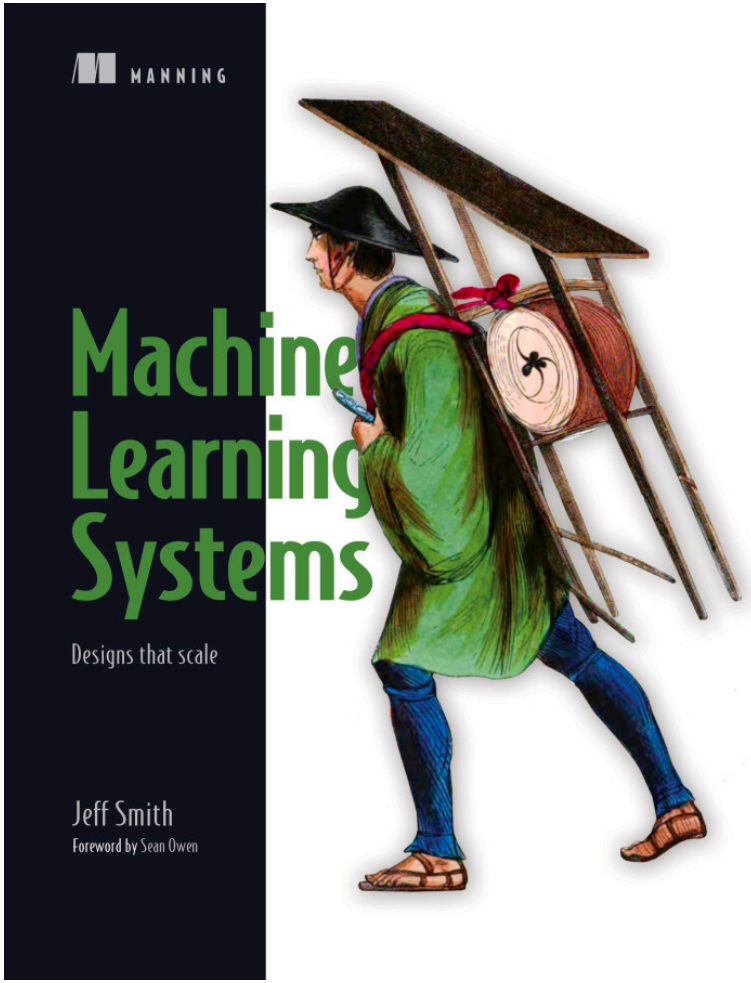
Recurrent Neural Networks for Event Forecasting

Pooyan Jamshidi
USC

Learning goals

- Understand how to build a system that can put the power of machine learning to use.
- Understand how to incorporate ML-based components into a larger system.
- Understand the principles that govern these systems, both as software and as predictive systems.

Main Sources



colah's blog

Understanding LSTM Networks

Posted on August 27, 2015

UBER Engineering

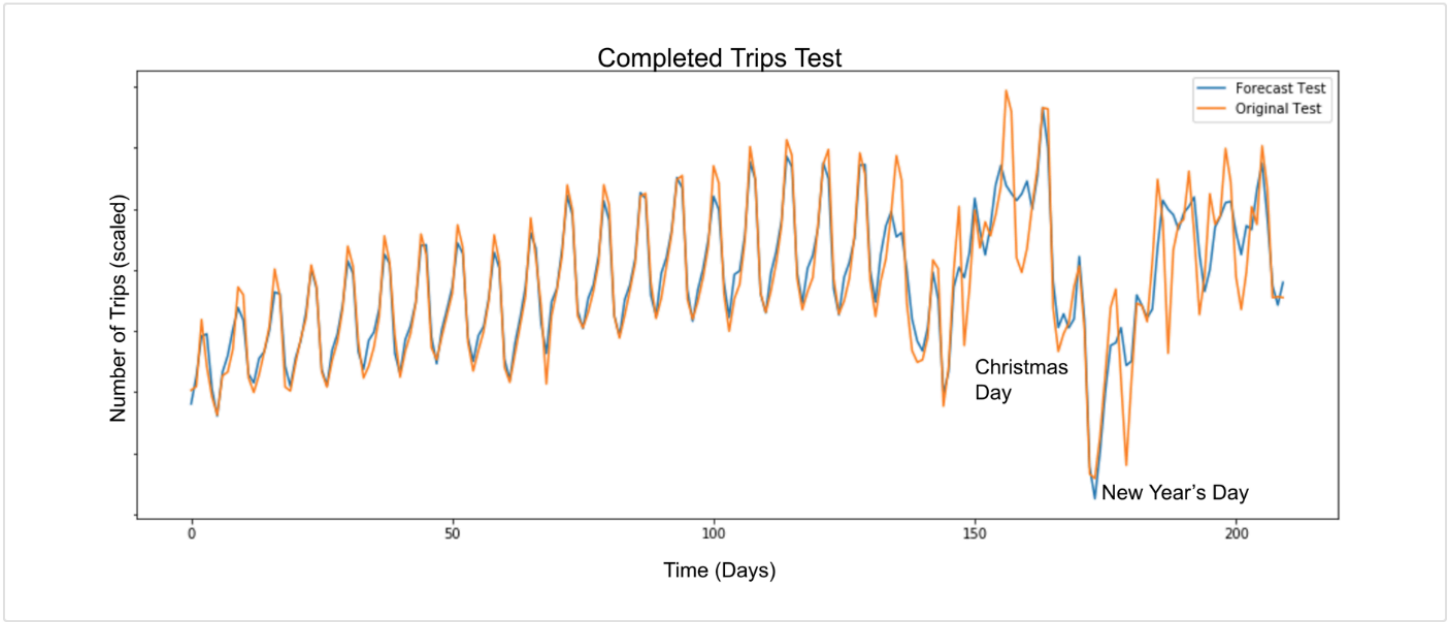
THE NETFLIX TECH BLOG

Spotify Labs

Engineering Extreme Event Forecasting at Uber with Recurrent Neural Networks

By Nikolay Laptev, Slawek Smyl, & Santhosh Shanmugam

June 9, 2017



Why does Uber need event forecasting?

Why does Uber need an event forecasting?

- To accurately predict where, when, and how many ride requests Uber will receive at any given time.

Extreme events

Extreme events

- Extreme events—peak travel times such as holidays, concerts, inclement weather, and sporting events.
- Calculating demand time series forecasting during extreme events is a critical component of anomaly detection, optimal resource allocation, and budgeting.

Data sparsity make accurate prediction challenging

- Consider New Year's Eve (NYE), one of the busiest dates for Uber.
- We only have a handful of NYEs to work with,
- Each instance might have a different cohort of users.
- It also depends on numerous external factors, including weather, population growth, and marketing changes such as driver incentives.

What is the natural choice?

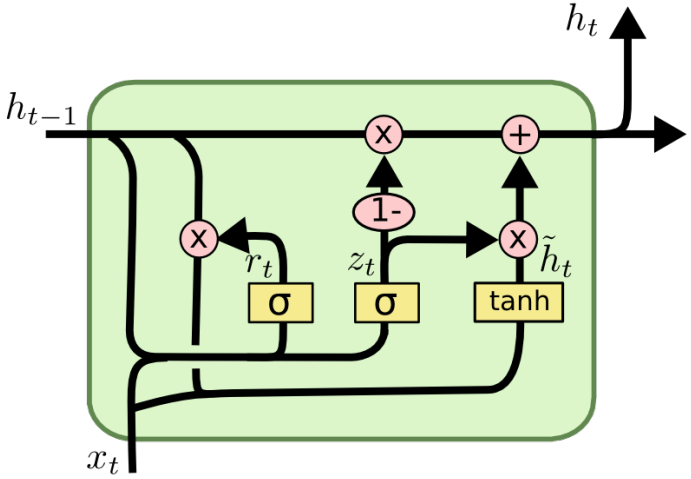
What is the natural choice?

- A combination of classical time series models, such as those found in the standard R forecast package, and machine learning methods are often used to forecast special events.

- These approaches, however, are neither flexible nor scalable enough for Uber.

What we review here

- Uber forecasting model that combines historical data and external factors to more precisely predict extreme events,
- highlighting its new architecture
- how it compares to our previous model.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

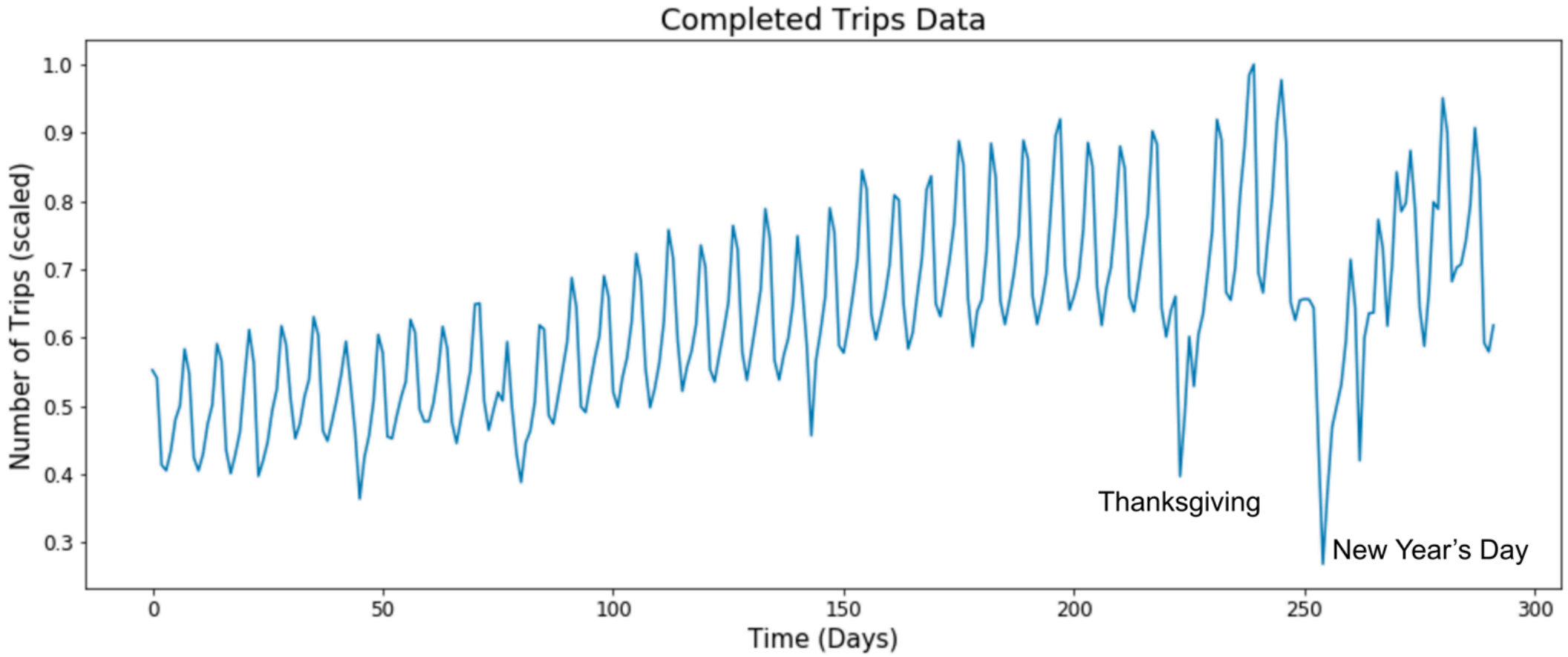
Long Short Term Memory

Why LSTM architecture?

- End-to-end modeling,
- Ease of incorporating external variables,
- Automatic feature extraction abilities.

By providing a large amount of data across numerous dimensions, an LSTM approach can model complex nonlinear feature interactions.

Data for training the LSTM models



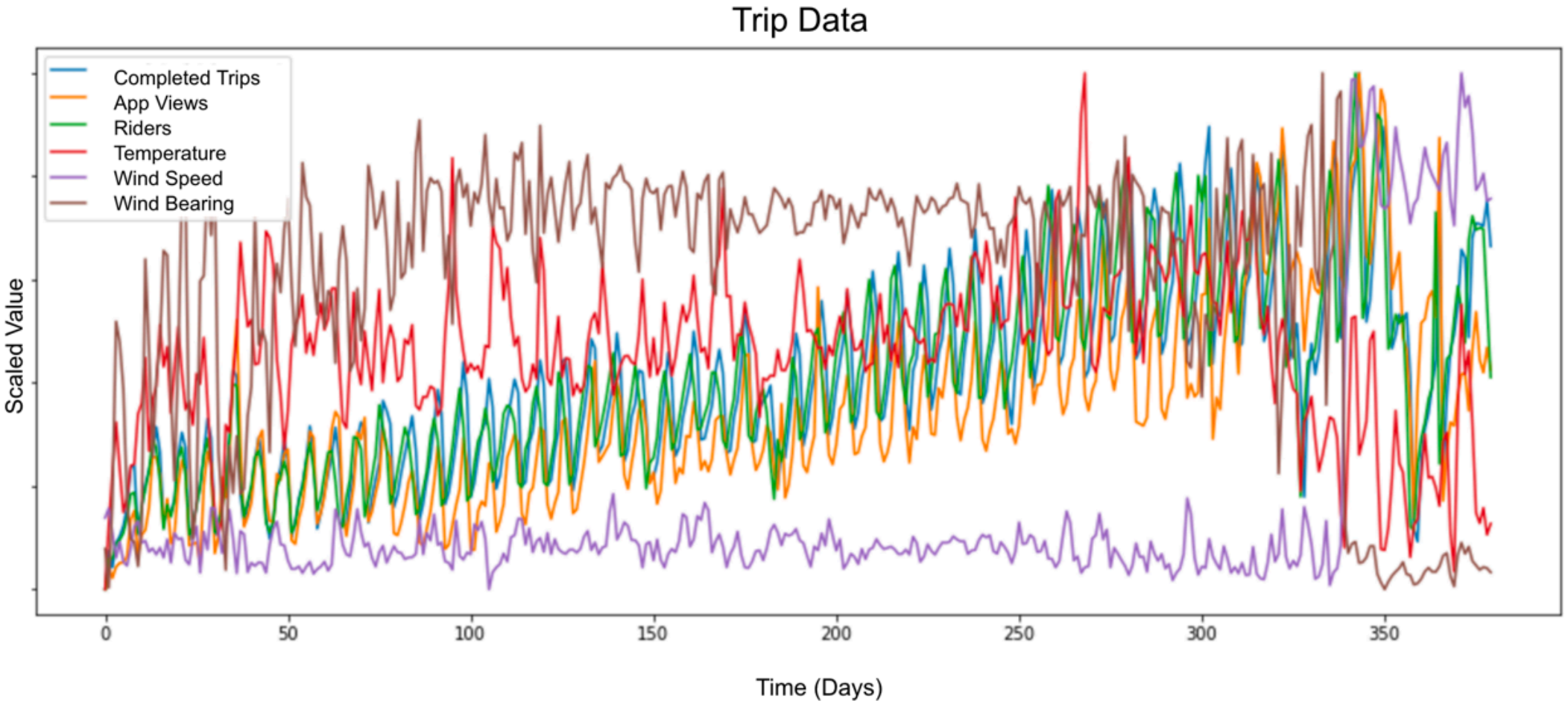
Goal

- Design a generic, end-to-end time series forecasting model that is
  - scalable,
  - accurate, and
  - applicable to heterogeneous time series.

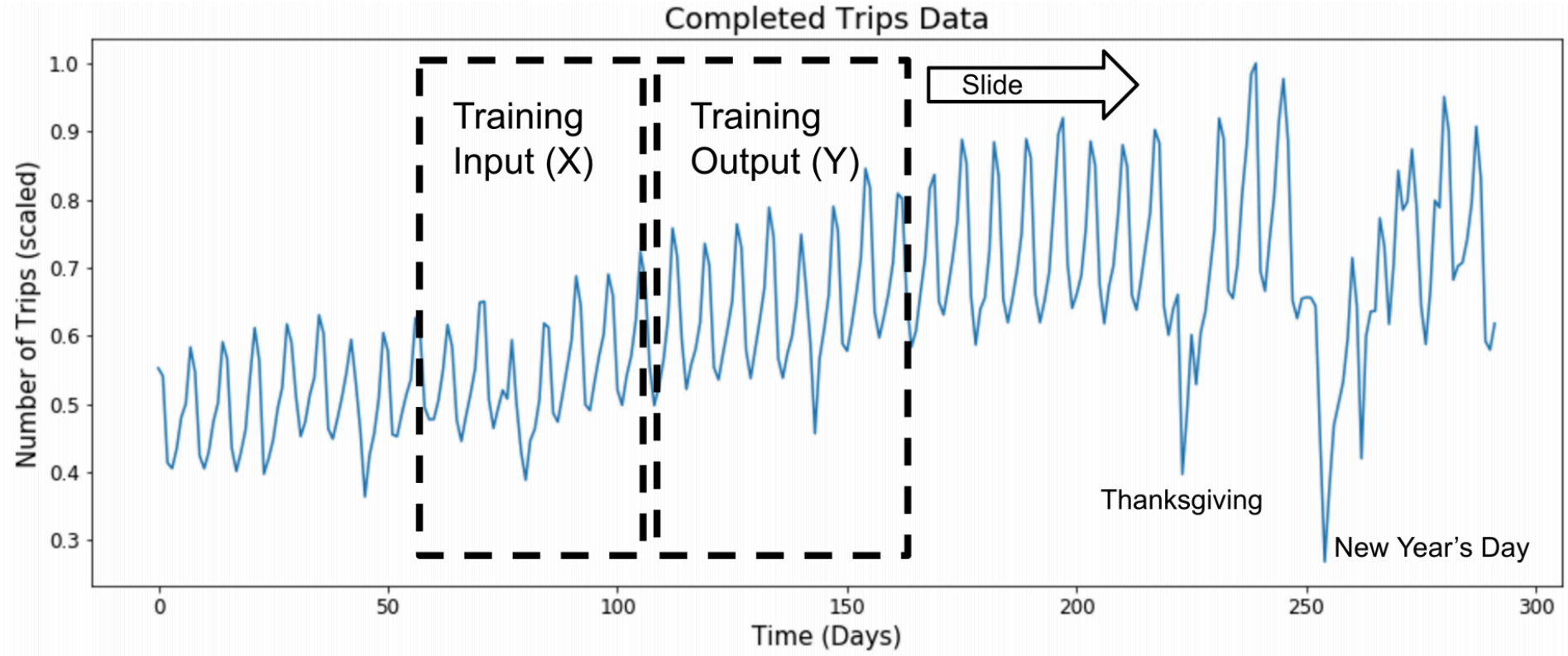
Where the data come from?

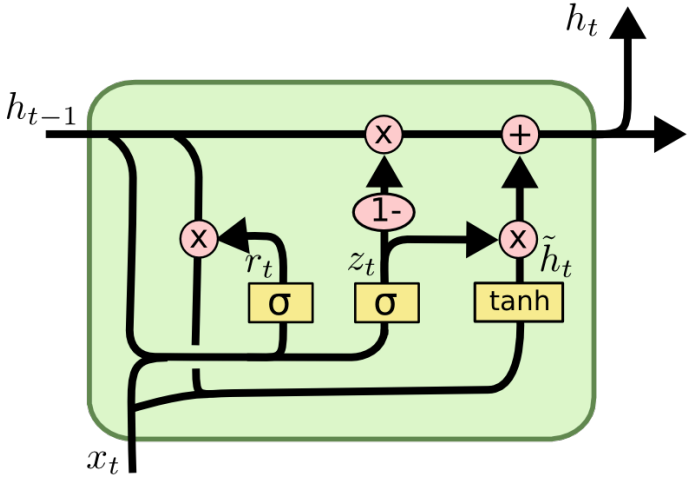
- Used thousands of time series to train a multi-module neural network.

The time series



Training with sliding window





$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

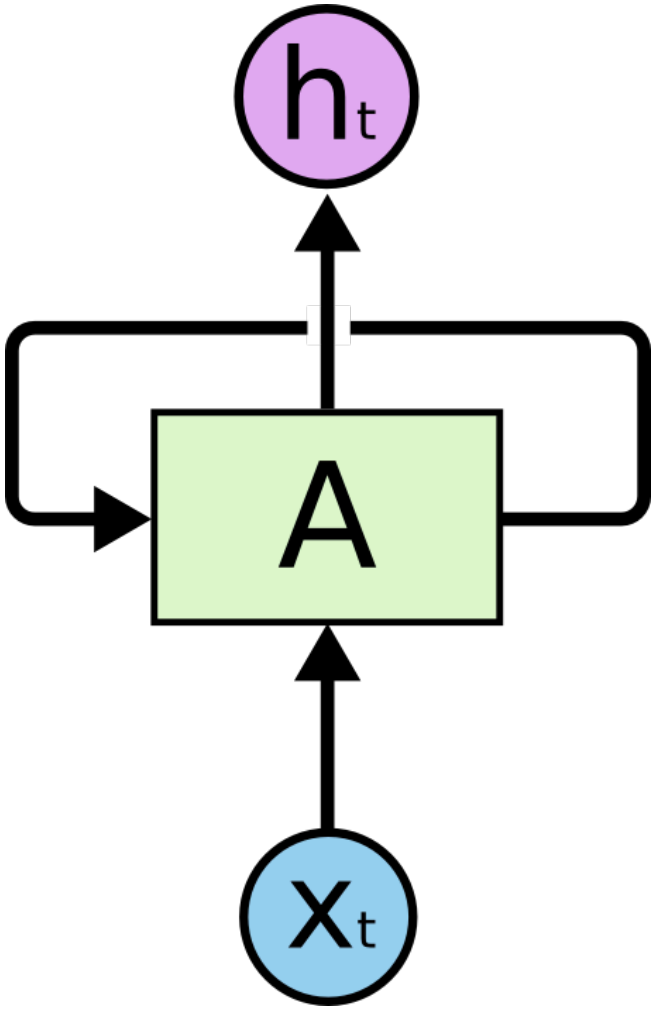
Long Short Term Memory

Let me walk you through this

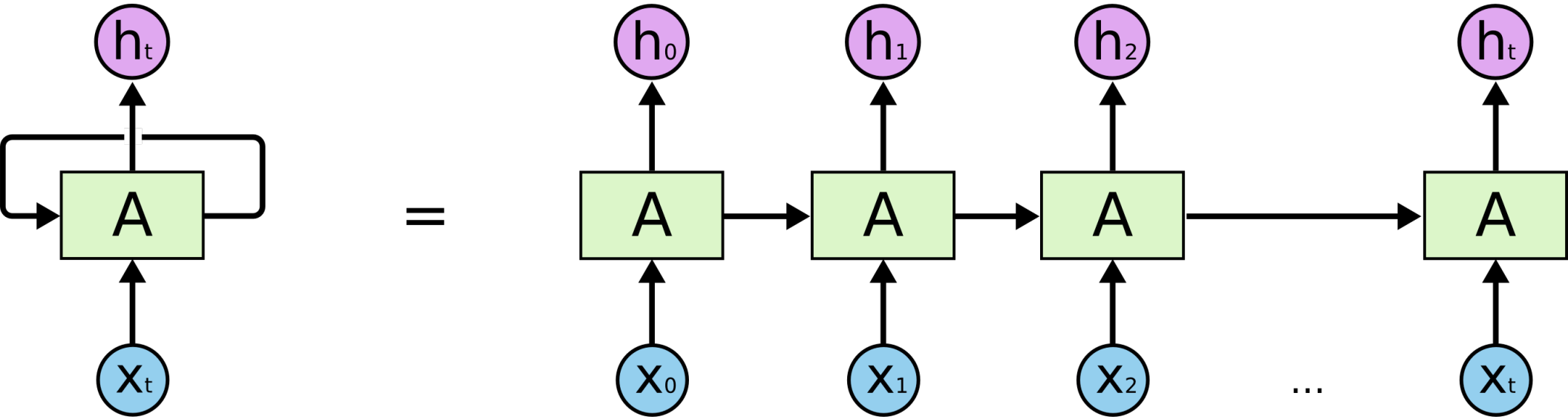
Acknowledgement

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

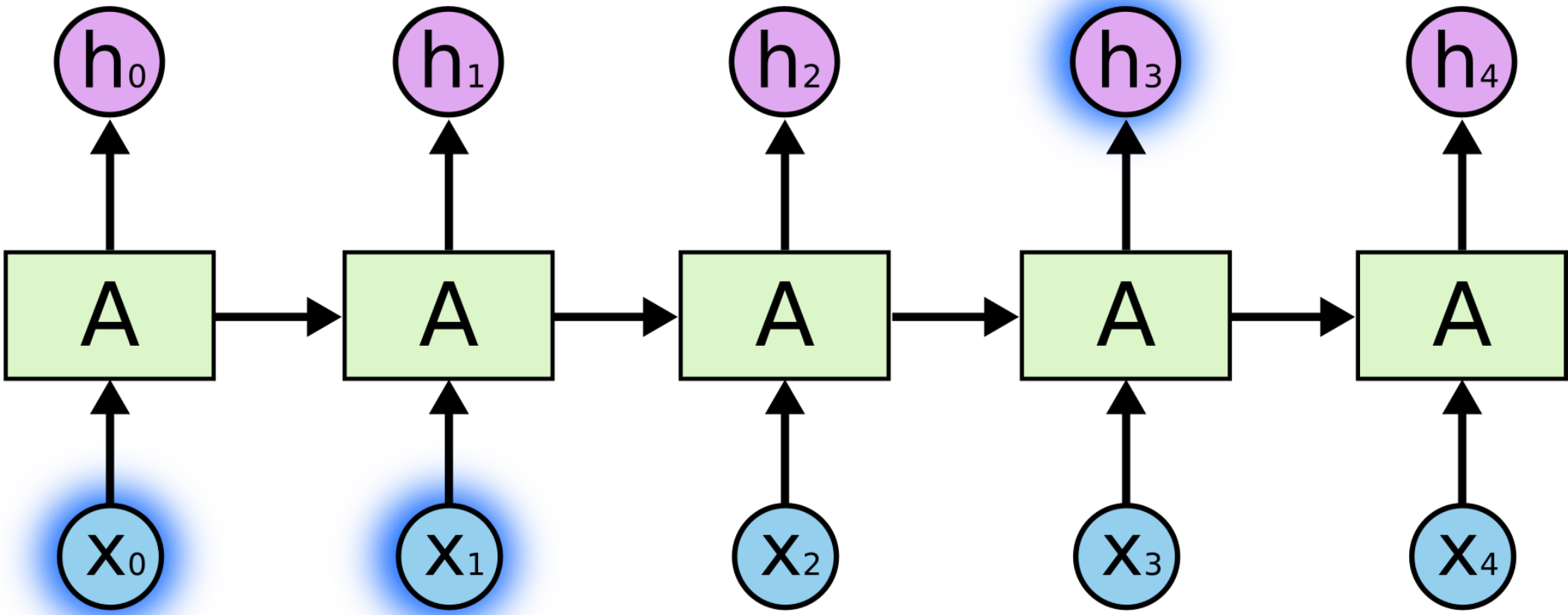
RNNs



Lets unroll the loop

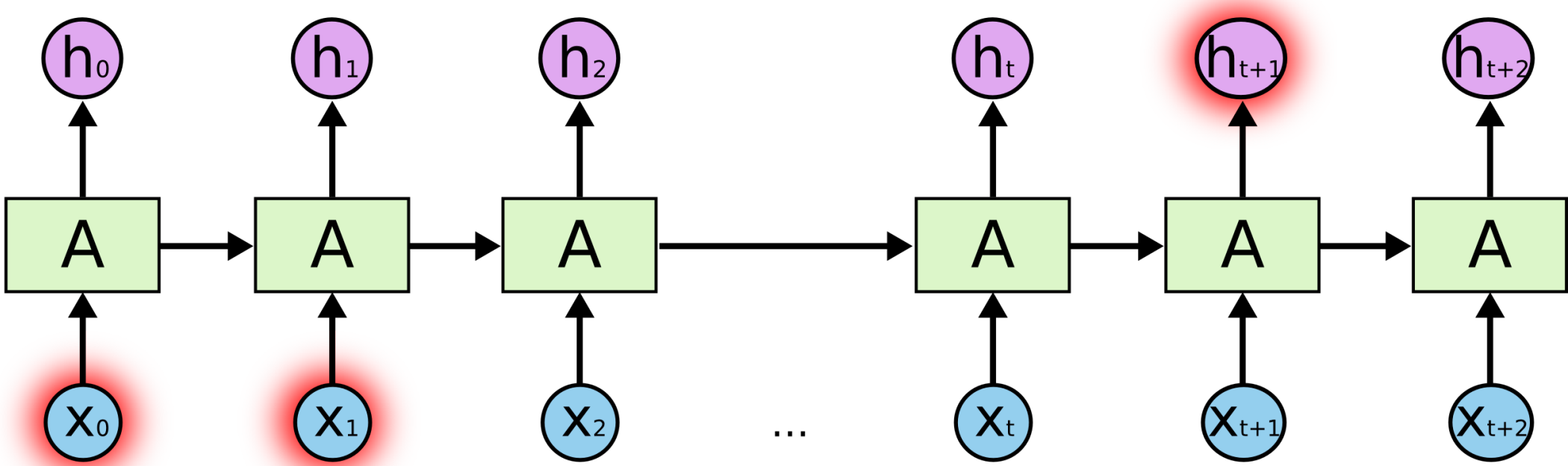


The Problem of Long-Term Dependencies



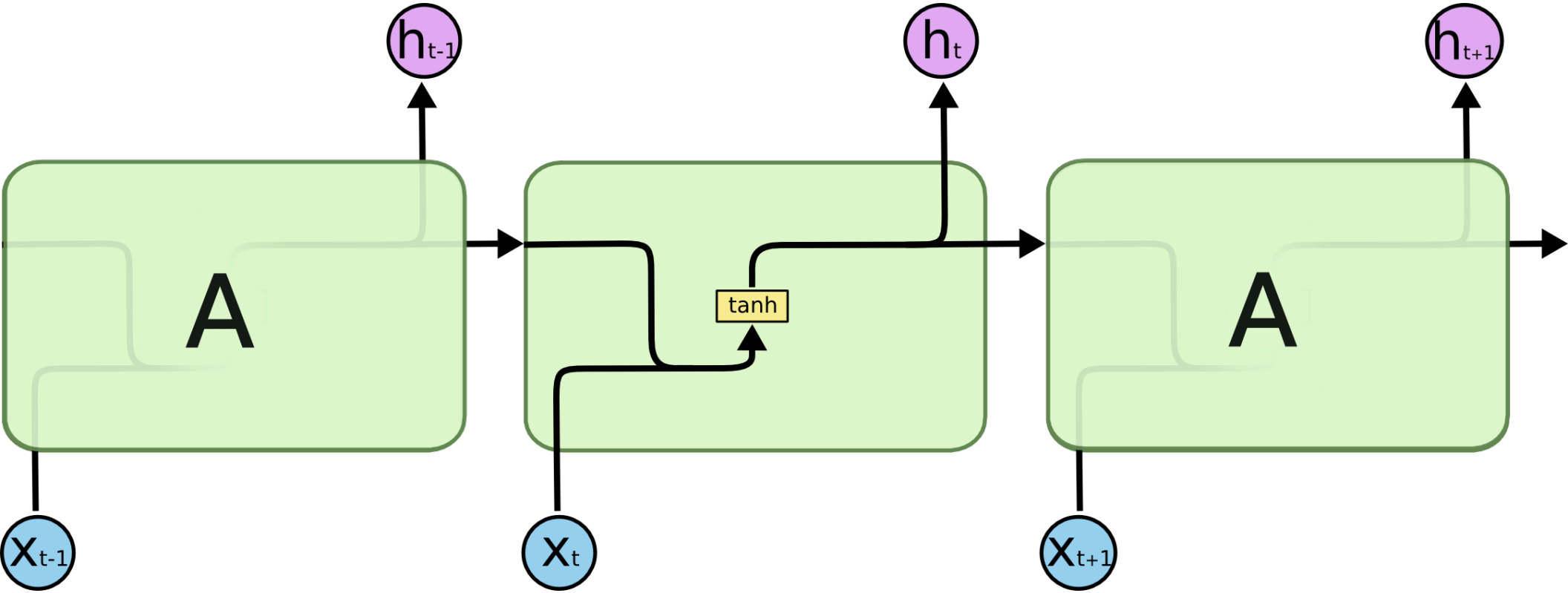
“the clouds are in the ??”

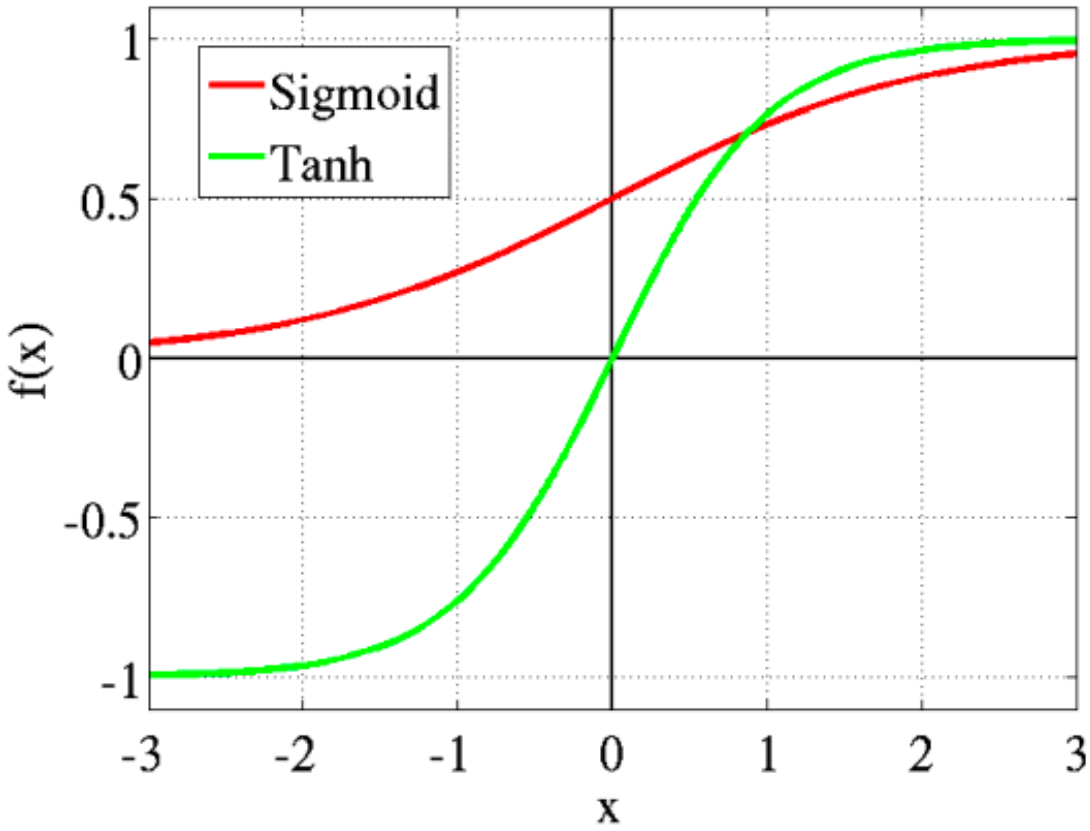
The Problem of Long-Term Dependencies



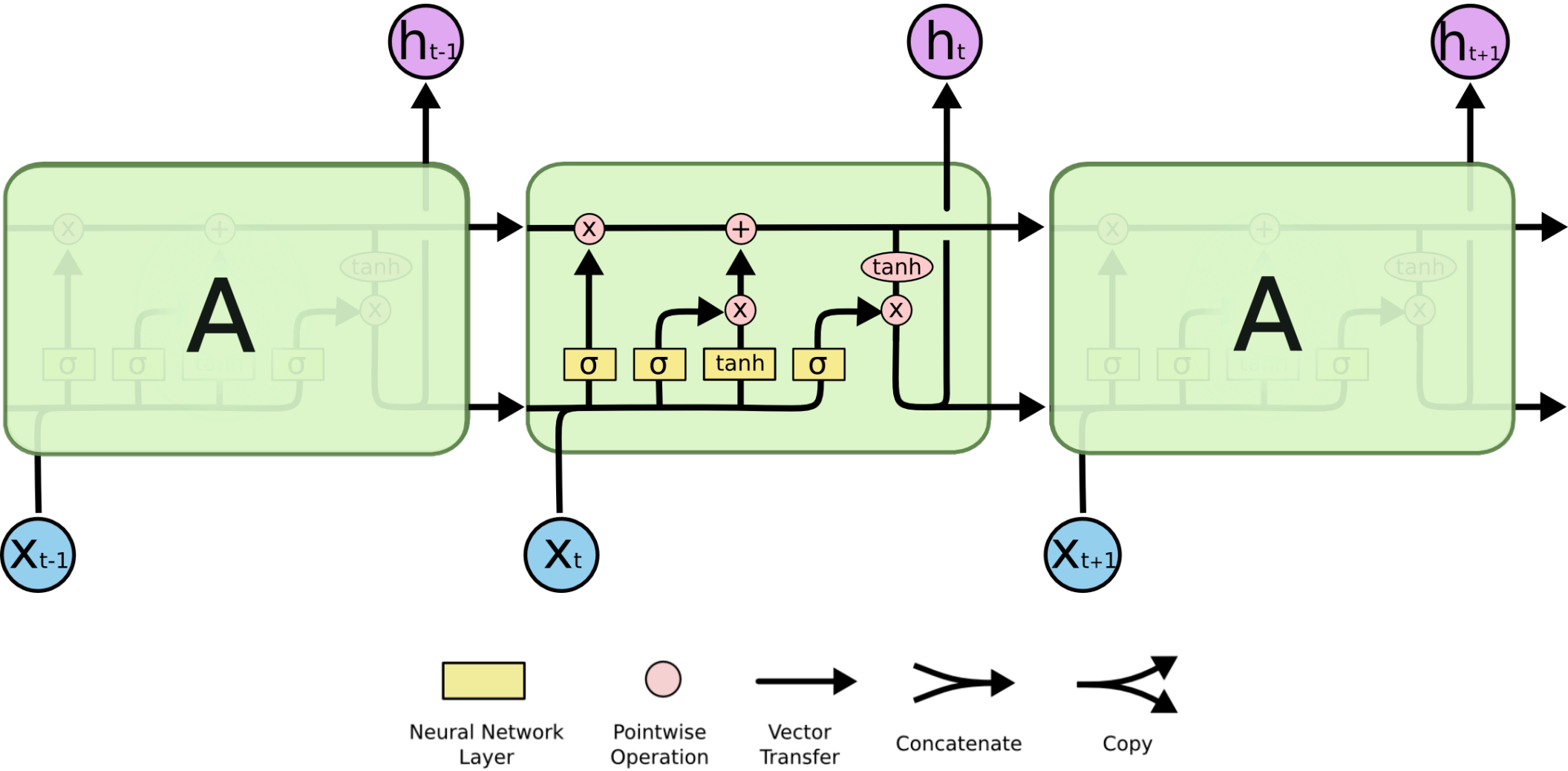
“I grew up in France… I speak fluent ??.”

The repeating module in a standard RNN contains a single layer

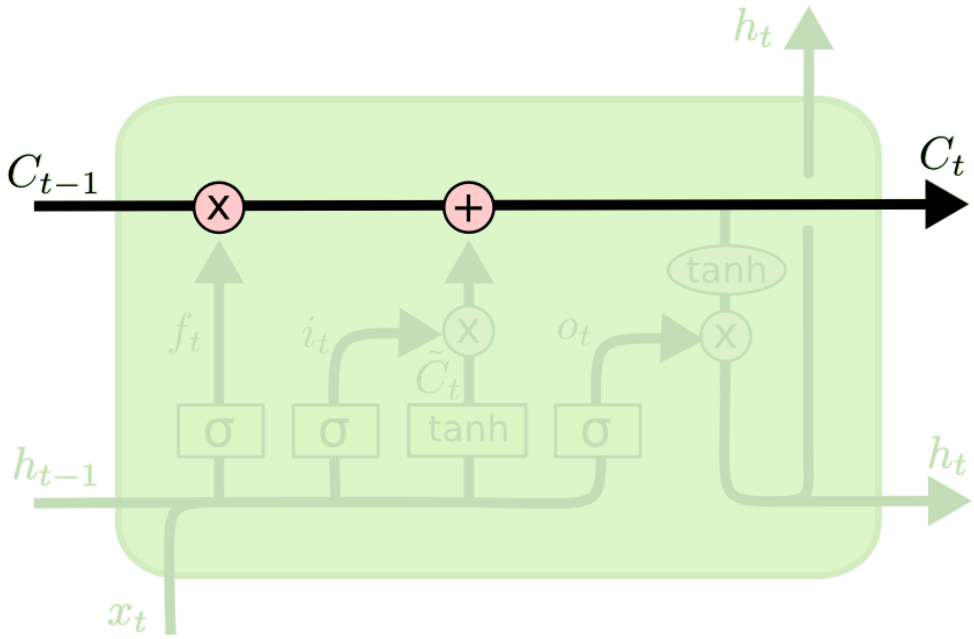



Sigmoid
Tanh
f(x)
x
1
0.5
0
-0.5
-1
-3
-2
-1
0
1
2
3

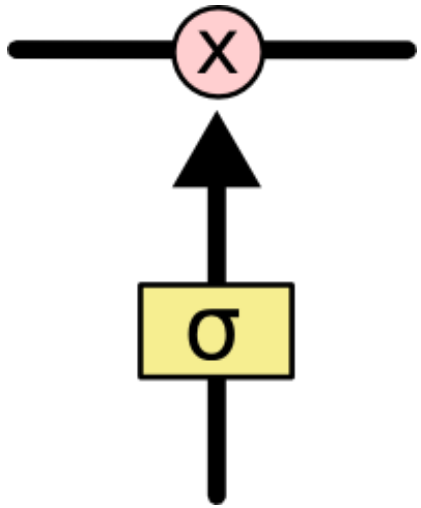
The repeating module in an LSTM contains four interacting layers



The Core Idea Behind LSTMs

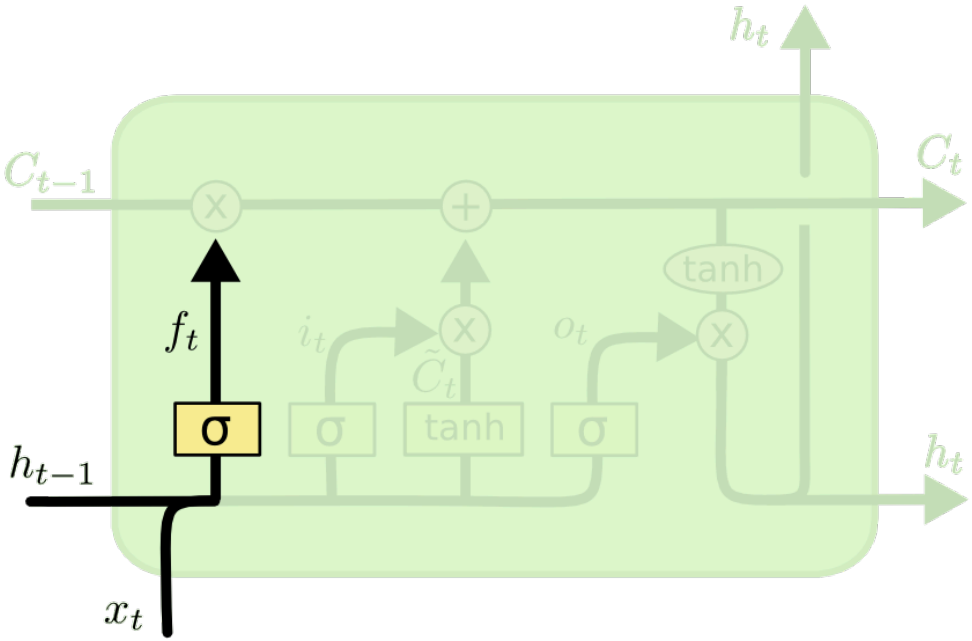


Gates are a way to optionally let information through



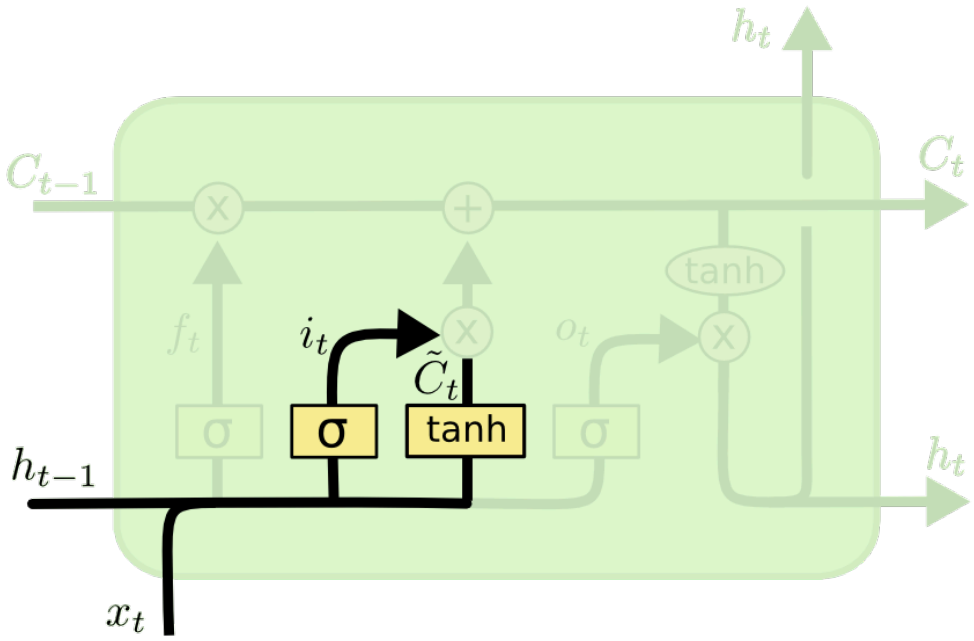
Step-by-Step LSTM Walk Through

Forgetting gate layer



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

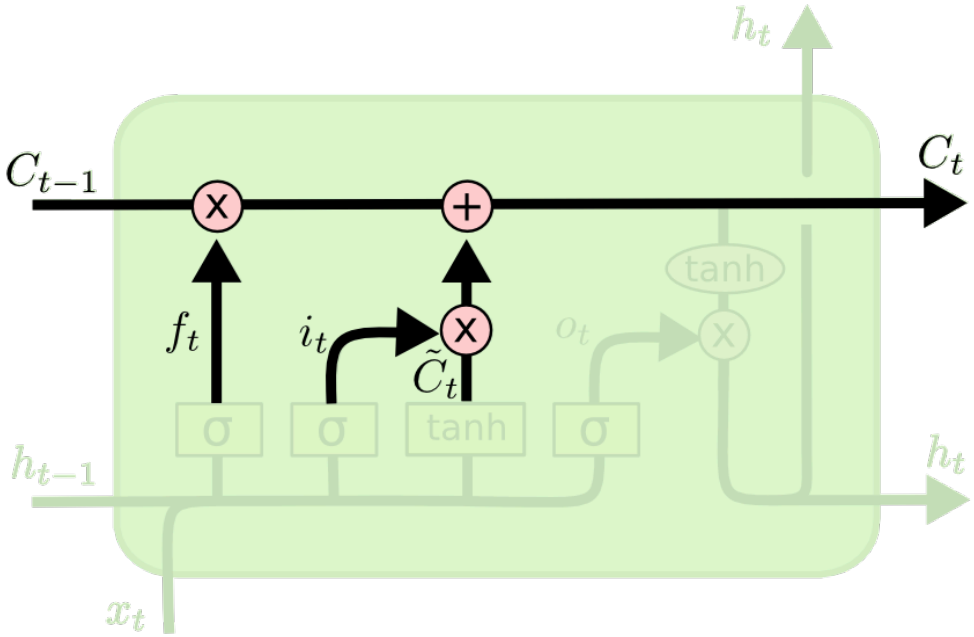
It forgot, now what to replace it with?



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

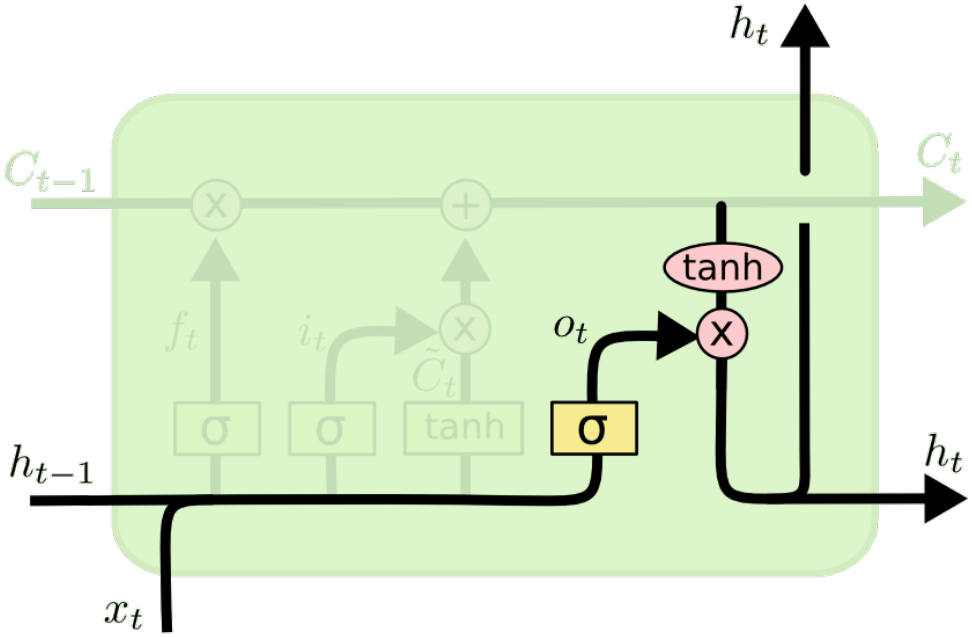
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Lets update the old info



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Now is the time to generate the output

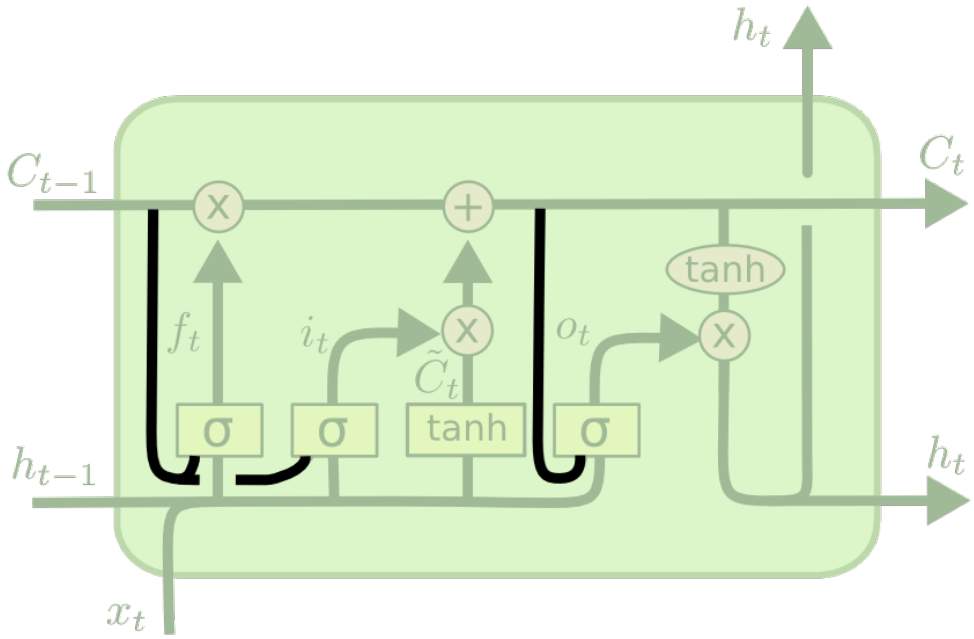


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Variants on Long Short Term Memory

Peephole

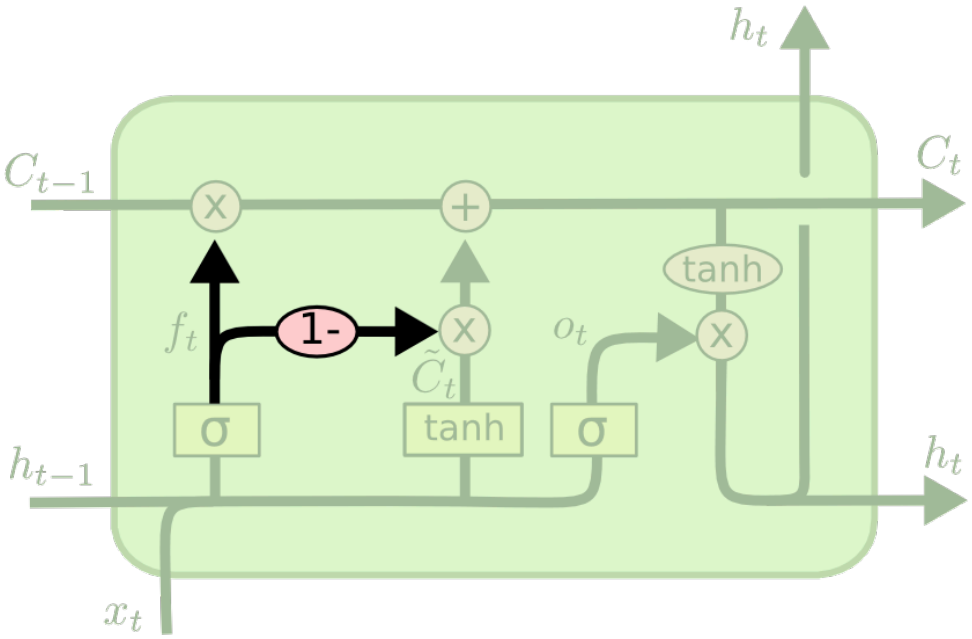


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$

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

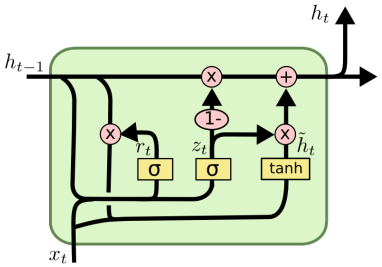
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Coupled forget and input gates



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

And so many others!



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Do these differences really make things better or worst?

Do these differences really matter?

- Greff, et al. (2015): They're all about the same
- Jozefowicz, et al. (2015): Some worked better than LSTMs on certain tasks

Lets continue with the case study

The vanilla LSTM didn’t work well

- Did not exhibit superior performance compared to the baseline model, which included a combination of univariate forecasting and machine learning elements.

- The vanilla model **could not adapt to time series with domains it was not trained on**, which led to poor performance when using a single neural network.

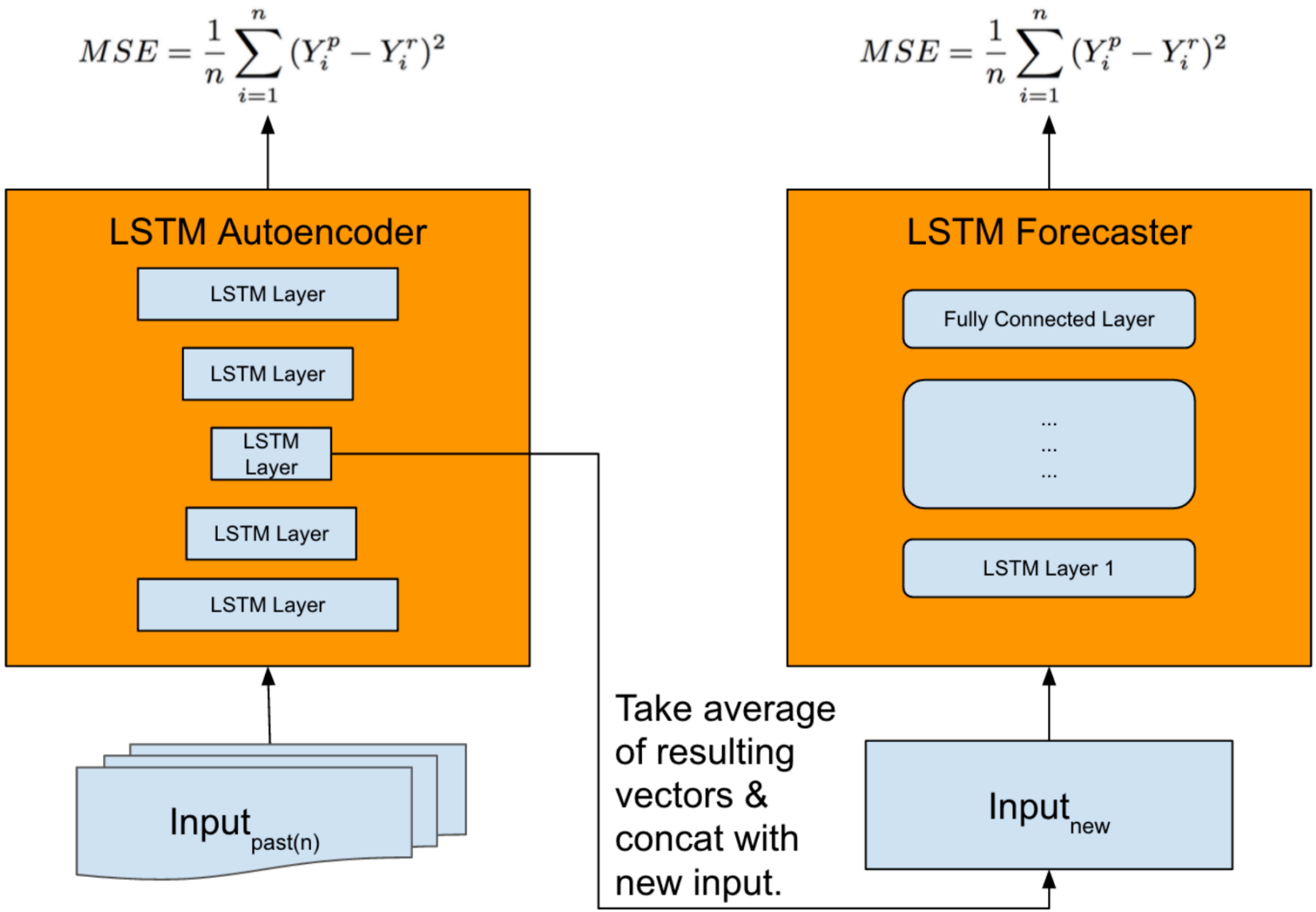
How about training one model per metric? Or one model for all metrics?

- It is impractical to train one model per time series for millions of metrics; there are simply not enough resources available, let alone hours in the day.

- Furthermore, training a single vanilla LSTM does not produce competitive results because the model cannot distinguish between different time series.

- While time series features and inputs can be manually loaded into the vanilla LSTM model, this approach is tedious and error-prone.

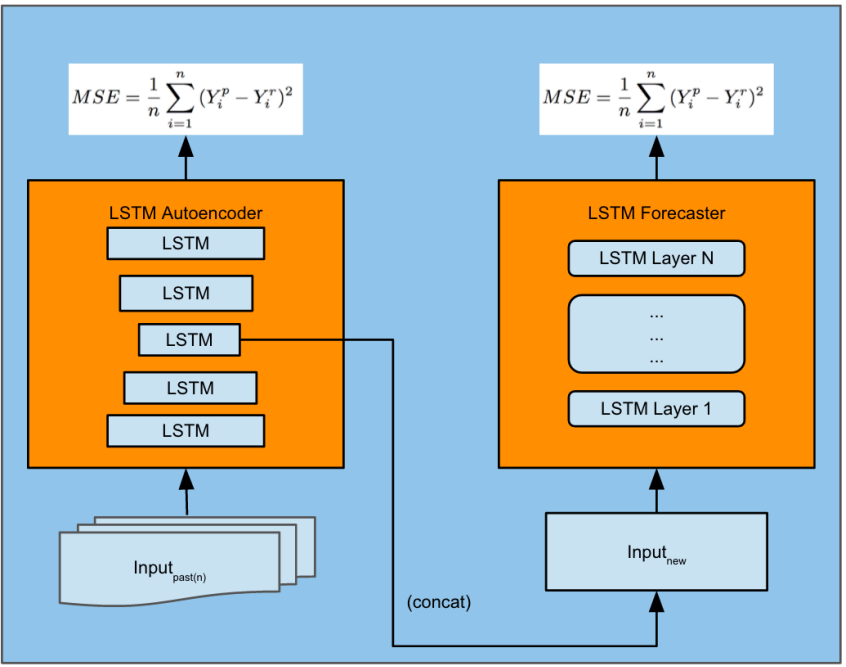
Tailoring LSTM model



Results

- Achieved a 14.09 percent symmetric mean absolute percentage error (SMAPE) improvement over the base LSTM architecture.
- 25 percent improvement over the classical time series models.

How the model works in production?



Export weights and operations

```
func (tt timeTravel) nnForecast(x []float64) ([]float64, error) {
  layerWeights := tt.layerWeights
  layerBias := tt.layerBias
  var h1 []float64
  if layerWeights == nil || layerBias == nil {
    return h1, errNNForecast
  }
  prev := x

  for i := range layerWeights {
    h1, _ = stats.ElemSum(stats.DotProduct(prev,
                          layerWeights[i]), layerBias[i])
    prev = h1
  }
  return h1, nil
}
```

How they tested the model?

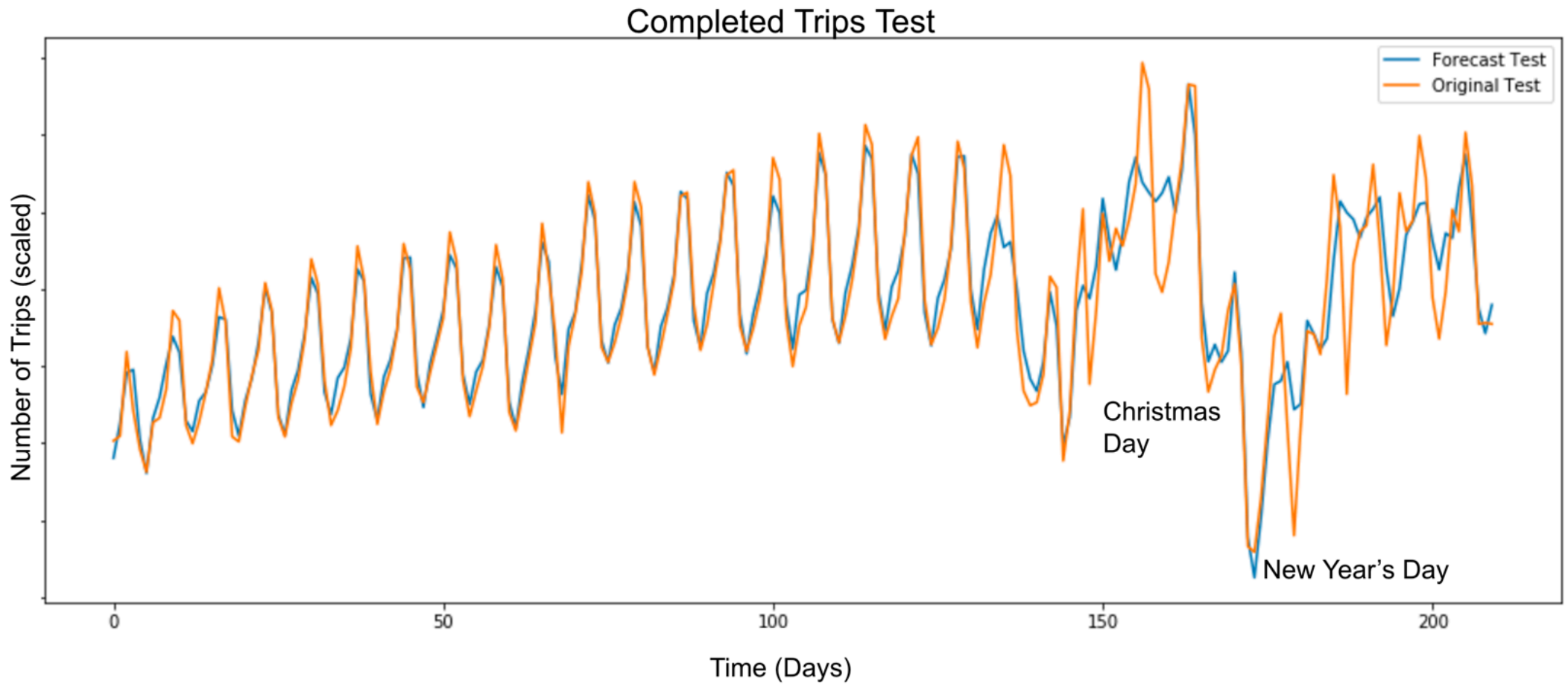
- Uber built a model using the **five-year daily history** of **completed Uber trips** across the **US** over the course of **seven days before, during, and after** major holidays like Christmas Day and New Year’s Day.

New forecasting model dramatically outperformed previous one

Holiday	Described Neural Network Architecture Error	Previous Model Error
Christmas Day	**11.1**	29.2
Martin Luther King, Jr. Day	**8.7**	20.2
Independence Day	**2.8**	17.6
Labor Day	**2.9**	6.9
New Year's Day	**6.8**	7.8
Veteran's Day	**4.7**	8.9

For instance, the new model found that one of the most difficult holidays to predict is Christmas Day, which corresponds to the greatest error and uncertainty in rider demand.

Comparison


Completed Trips Test
Forecast Test
Original Test
Number of Trips (scaled)
Christmas Day
New Year's Day
0
50
100
150
200
Time (Days)

Neural nets are not silver bullet

- Number of time series,
- Length of time series,
- Correlation among time series.

Summary

- We reviewed Uber forecasting model to more precisely predict rare events.
- We reviewed the challenges of predicting rare events at scale.
- We walked through basics of RNNs and LSTMs.
- We highlighted Uber LSTM architecture and how it compares to their previous model.