Recurrent Neural Network

Rachel Hu and Zhi Zhang

Amazon AI

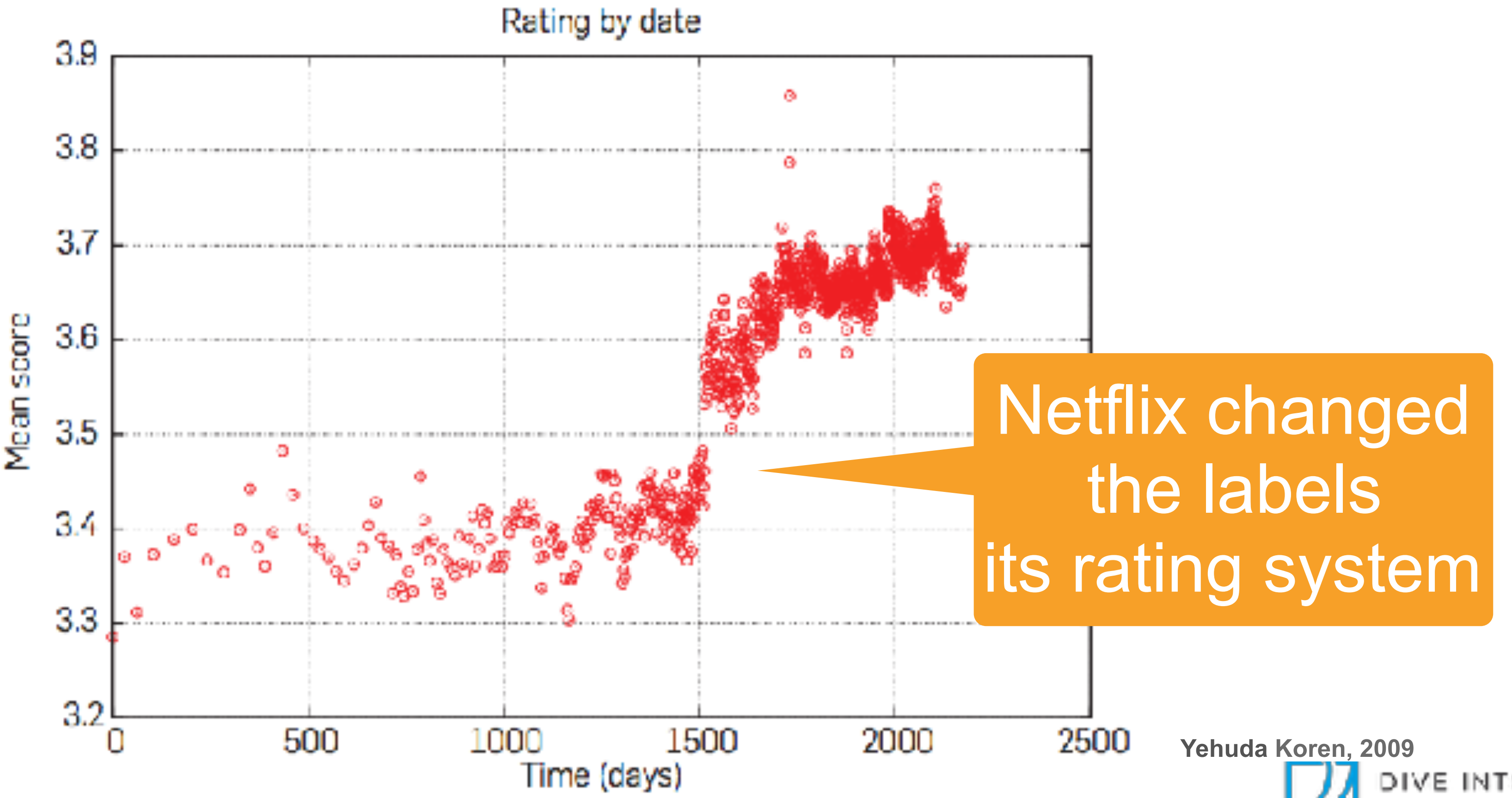
DIVE INTO DEEP LEARNING

Outline

- Dependent Random Variables
- Text Preprocessing
- Language Modeling
- Recurrent Neural Networks (RNN)
- LSTM
- Bidirectional RNN
- Deep RNN

Dependent Random Variables

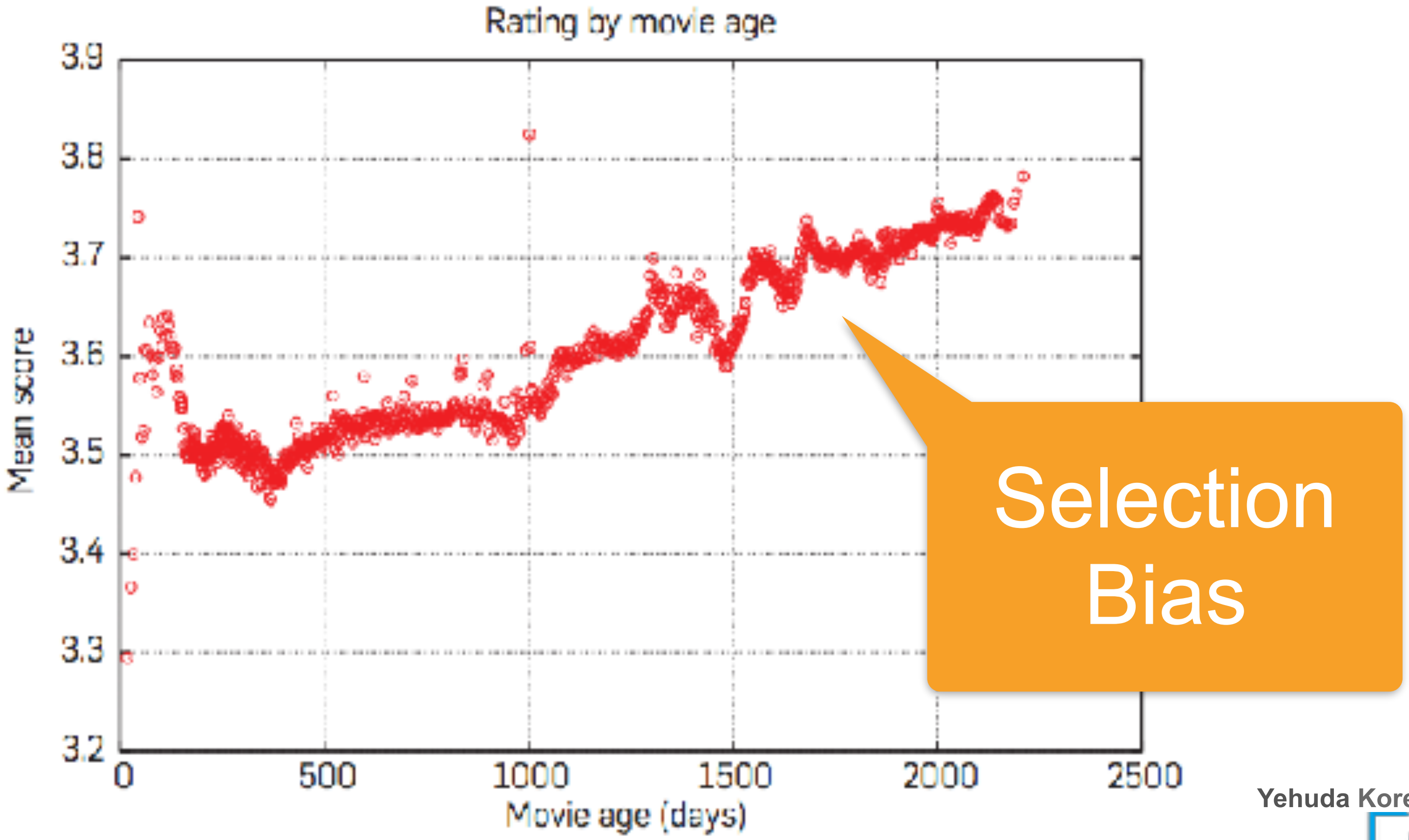
Time matters (Koren, 2009)



Yehuda Koren, 2009

DIVE INTO DEEP LEARNING

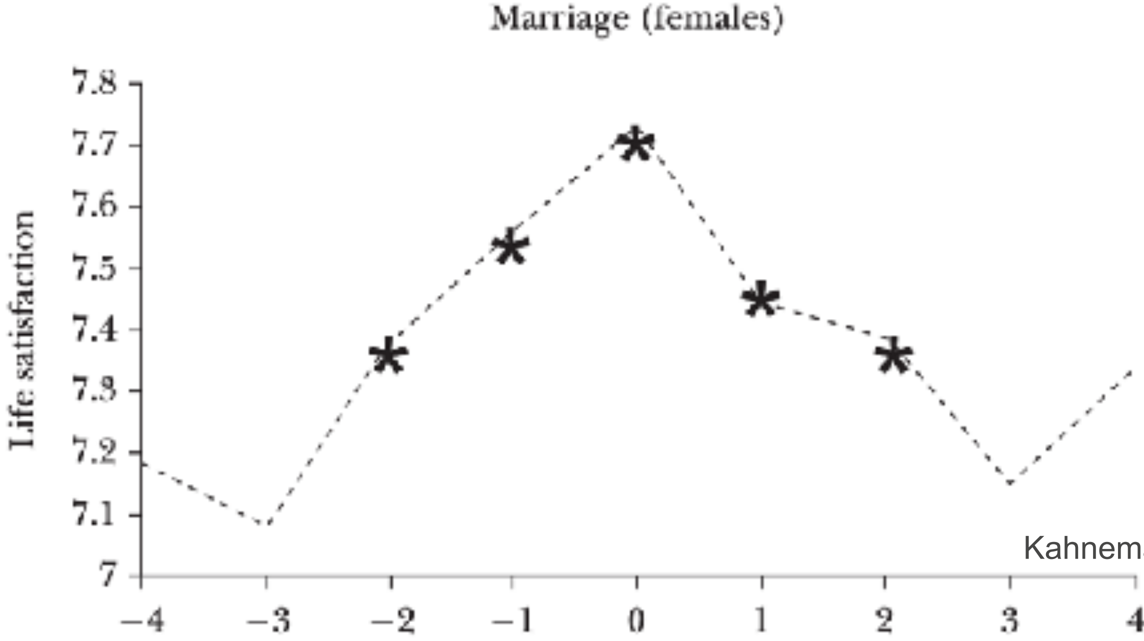
Time matters (Koren, 2009)



Yehuda Koren, 2009

Average Life Satisfaction for a Sample of German Women

(by year of marriage t = 0)

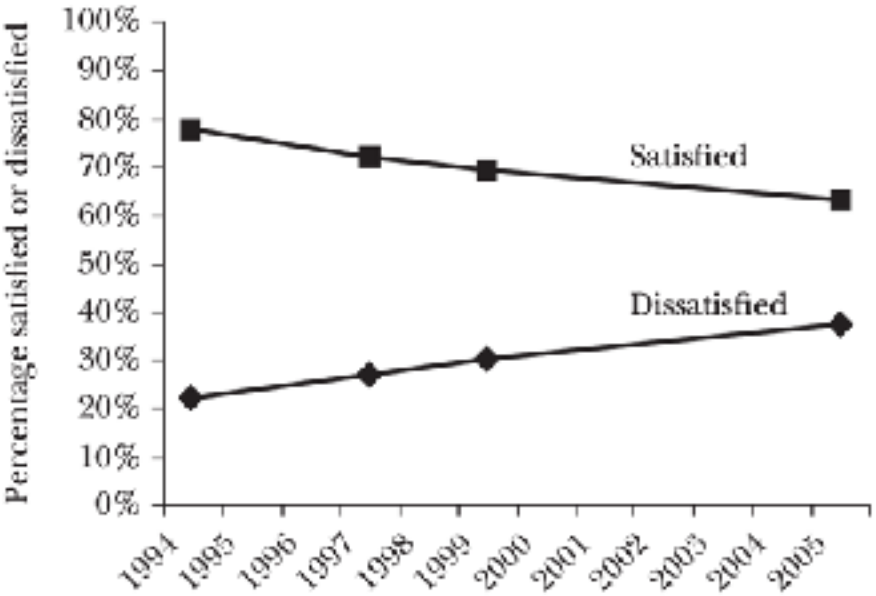


Kahnemann & Krueger, 2006

Source: Clark, Diener, Georgellis and Lucas (2003), using data from the German Socioeconomic Panel.
Note: An asterisk indicates that life satisfaction is significantly different from the baseline level.

Life Satisfaction in China as Average Real Income Rises by 250 Percent

Overall, how satisfied or dissatisfied are you with the way things are going in your life today? Would you say you are very satisfied, somewhat satisfied, somewhat dissatisfied, or very dissatisfied?



Kahnemann & Krueger, 2006

Source: Derived from Richard Burkholder, "Chinese Far Wealthier Than a Decade Ago—but Are They Happier?" The Gallup Organization, (http://sww.gallup.com/poll/content/login.aspx?ci=14548).

rate cuts

rating agencies

Q2 earnings

orange hair tweets

TL;DR - Data usually isn't IID

inventory

Christmas

Black Friday

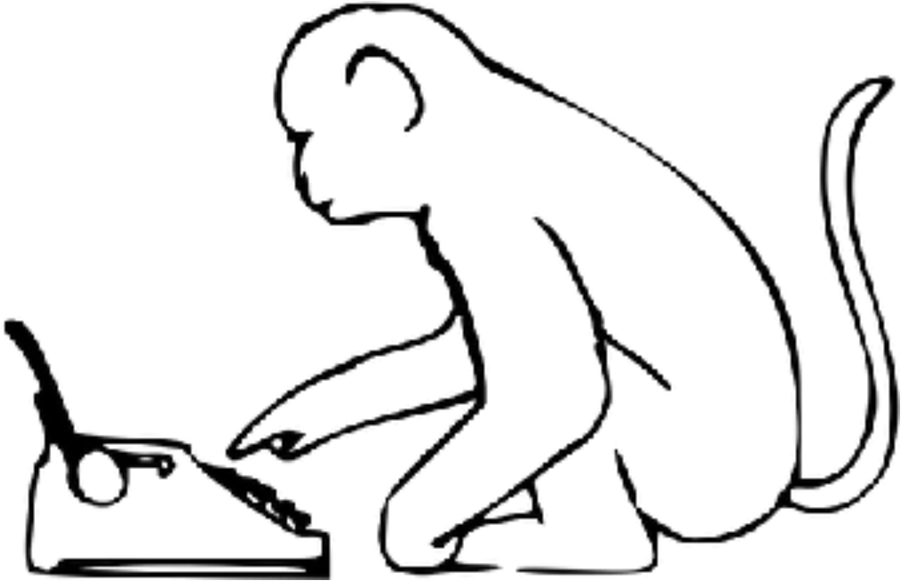
prime day

back to school

Data

- So far …
  - Collect observation pairs $(x_i, y_i) \sim p(x, y)$ for training
  - Estimate $y \mid x \sim p(y \mid x)$ for unseen $x' \sim p(x)$
- Examples
  - Images classification & objects recognition
  - Disease prediction
  - Housing price prediction
- **The order of the data does not matter**

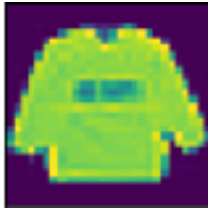
Text Processing



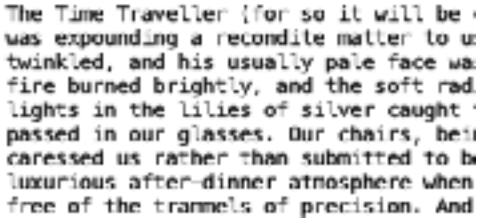
DIVE INTO DEEP LEARNING

Text Preprocessing

- Sequence data has long dependency (very costly)
- Truncate into shorter fragments
- Transform examples into mini-batches with ndarrays



(batch size, width, height, channel)



The Time Traveller (for so it will be
was expounding a recondite matter to u
twinkled, and his usually pale face wa
fire burned brightly, and the soft rad
lights in the lilies of silver caught
passed in our glasses. Our chairs, bei
caressed us rather than submitted to b
luxurious after-dinner atmosphere when
free of the trammels of precision. And

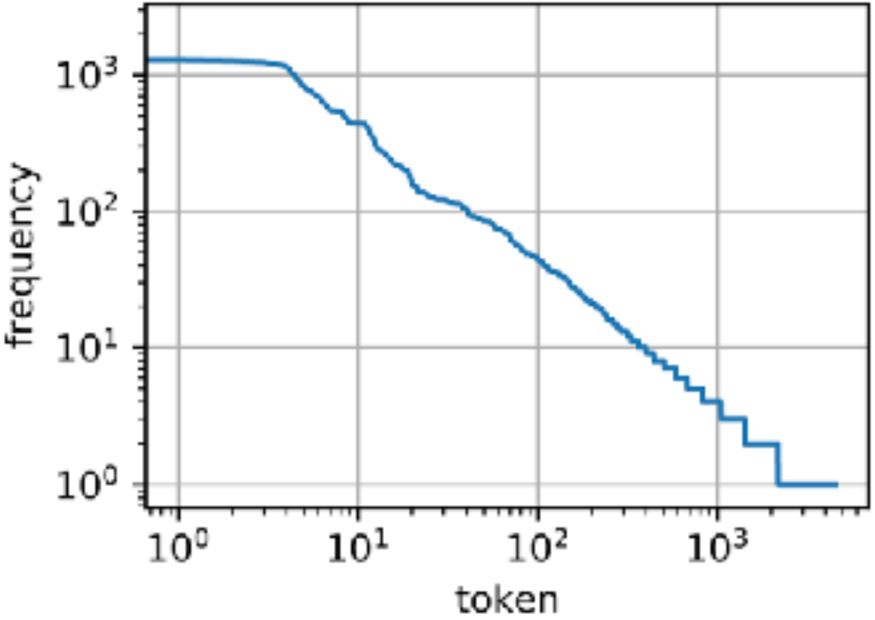
(batch size, sentence length)

Tokenization

- Basic Idea - map text into sequence of tokens
  - “Deep learning is fun” -> [“Deep”, “learning”, “is”, “fun”, “.”]
- **Character Encoding** (each character as a token)
  - Small vocabulary
  - Doesn’t work so well (needs to learn spelling)
- **Word Encoding** (each word as a token)
  - Accurate spelling
  - Doesn’t work so well (huge vocabulary = costly multinomial)
- **Byte Pair Encoding** (Goldilocks zone)
  - Frequent subsequences (like syllables)

Vocabulary

- Find unique tokens, map each one into a numerical index
  - “Deep” : 1, “learning” : 2, “is” : 3, “fun” : 4, “.” : 5
- The frequency of words often obeys a power law distribution
  - Map the tailing tokens, e.g. appears < 5 times, into a special “unknown” token



Minibatch Generation

The Time Machine by H. G. Wells

The Time Machine by H. G. Wells

The Time Machine by H. G. Wells

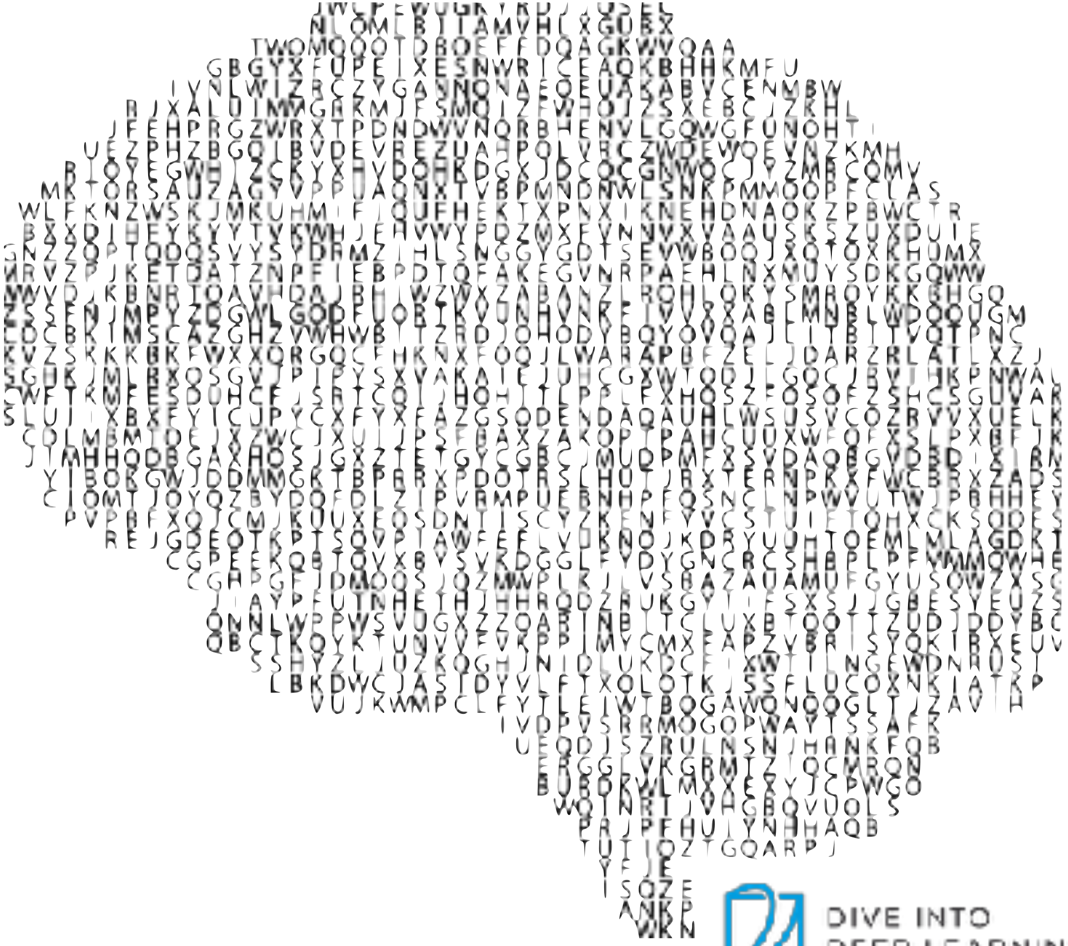
The Time Machine by H. G. Wells

The Time Machine by H. G. Wells

The Time Machine by H. G. Wells

Text Preprocessing Notebook

Language Models



DIVE INTO DEEP LEARNING

Language Models

- Tokens not real values (domain is countably finite)

$$p(w_1, w_2, \ldots, w_T) = p(w_1) \prod_{t=2}^{T} p(w_t \mid w_1, \ldots, w_{t-1})$$

- *e.g.,*

$$\begin{aligned} & p(\text{deep, learning, is, fun, .}) \\ = & \, p(\text{deep}) p(\text{learning} \mid \text{deep}) p(\text{is} \mid \text{deep, learning}) \\ & \, p(\text{fun} \mid \text{deep, learning, is}) p(\text{.} \mid \text{deep, learning, is, fun}) \end{aligned}$$

- Estimating it

$$\hat{p}(\text{learning} \mid \text{deep}) = \frac{n(\text{deep, learning})}{n(\text{deep})}$$

Need Smoothing

Language Modeling

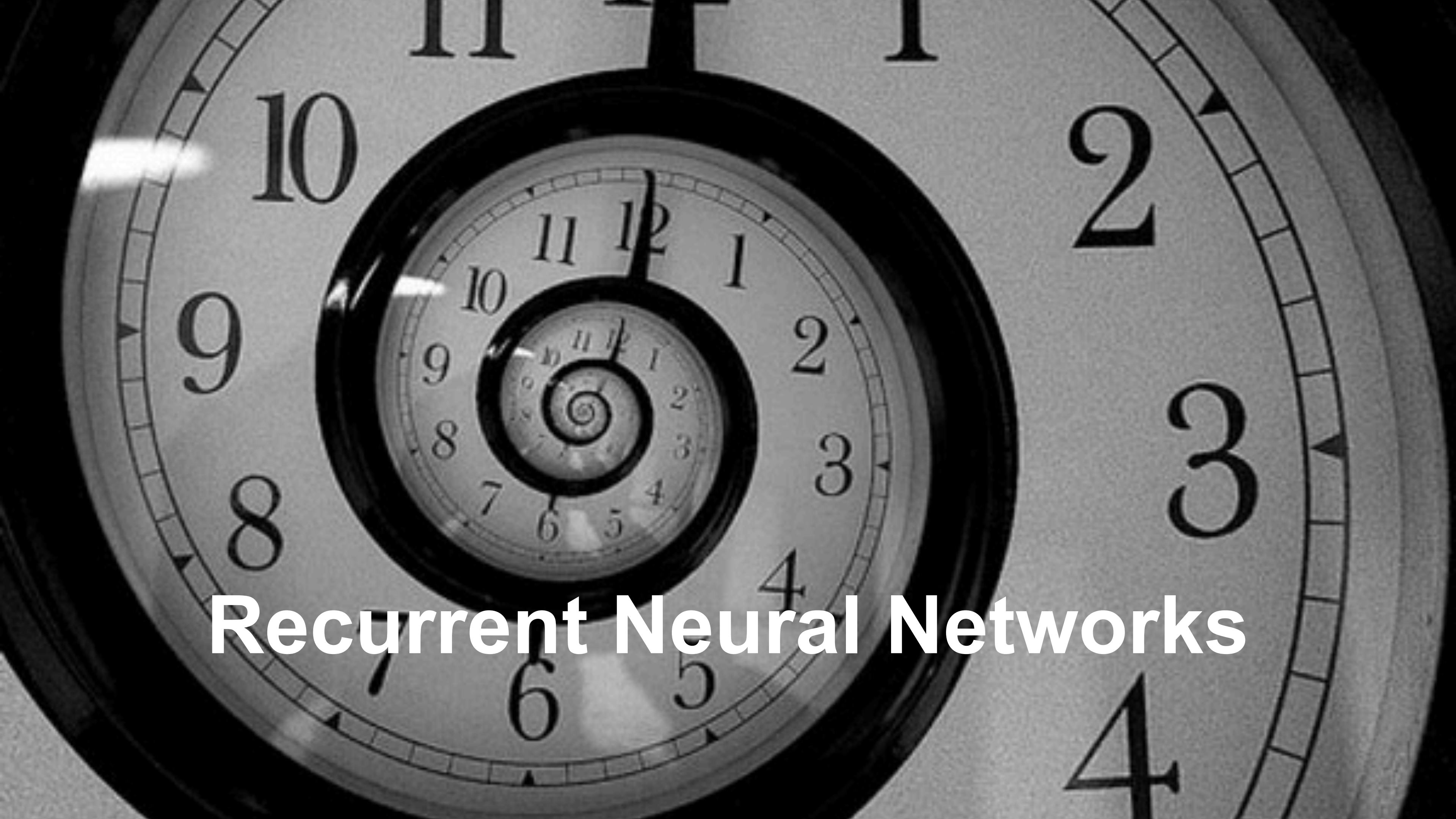
- Goal: predict the probability of a sentence, e.g.

$$p(\text{Deep}, \text{learning}, \text{is}, \text{fun}, .\,)$$

- NLP fundamental tasks
  - Typing - predict the next word
  - Machine translation - `dog bites man` vs `man bites dog`
  - Speech recognition
    `to recognize speech` vs `to wreck a nice beach`

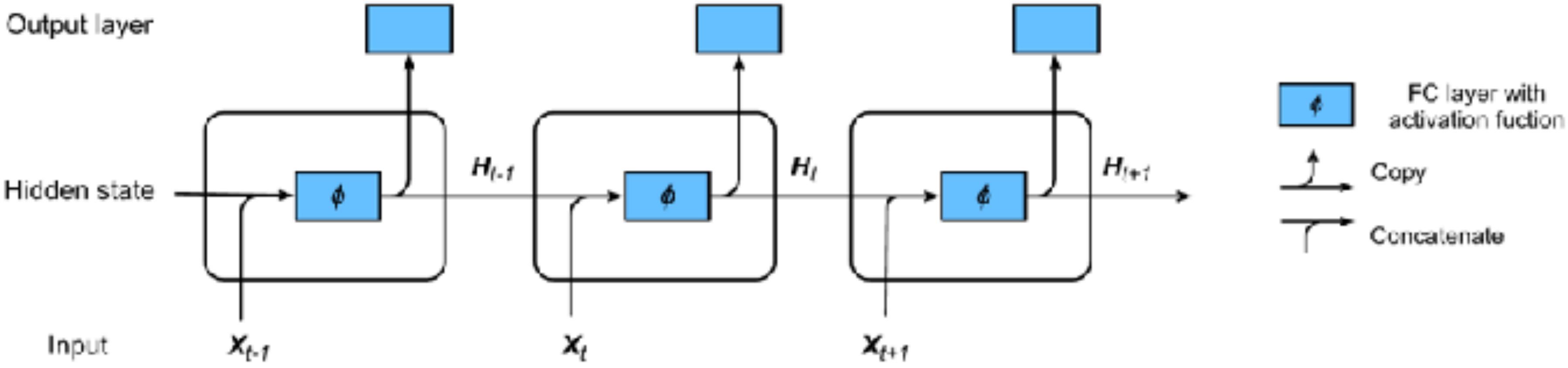
Language Modeling

- NLP fundamental tasks
  - Named-entity recognition
  - Part-of-speech tagging
  - Machine translation
  - Question answering
  - Automatic Summarization
  - …



Recurrent Neural Networks

RNN with Hidden States



- Hidden State update

$$\mathbf{H}_t = \phi(\boxed{\mathbf{W}_{hh}\mathbf{H}_{t-1}} + \mathbf{W}_{hx}\mathbf{X}_{t-1} + \mathbf{b}_h)$$

- Observation update

$$\mathbf{o}_t = \mathbf{W}_{ho}\mathbf{H}_t + \mathbf{b}_o$$

- 2-layer MLP

$$\mathbf{H}_t = \phi(\mathbf{W}_{hx}\mathbf{X}_{t-1} + \mathbf{b}_h)$$

$$\mathbf{o}_t = \mathbf{W}_{ho}\mathbf{H}_t + \mathbf{b}_o$$

Next word prediction

step	1	2	3	4	5
output					
output state					
hidden state					
input	The	Time	Machine	by	H.

Input Encoding

- Need to map input numerical indices to vectors
  - Pick granularity (words, characters, subwords)
  - Map to indicator vectors

```
npx.one_hot(np.array([0, 2]), len(vocab))
```

```
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
```

RNN with hidden state mechanics

- Input: vector sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^d$
- Hidden states: $\mathbf{h}_1, \ldots, \mathbf{h}_T \in \mathbb{R}^h$ where $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$
- Output: vector sequence $\mathbf{o}_1, \ldots, \mathbf{o}_T \in \mathbb{R}^p$ where $\mathbf{o}_t = g(\mathbf{h}_t)$
    - *p* is the vocabulary size
    - $\mathbf{o}_{t,j}$ is confident score that the *t*-th timestamp in the sequence equals to *j*-th token in the vocabulary
- Loss: measure the classification error on *T* tokens

Gradient Clipping

- Long chain of dependencies for backprop
  - Need to keep a lot of intermediate values in memory
  - Butterfly effect style dependencies
  - Gradients can vanish or explod
- Clipping to prevent divergence

$$\mathbf{g} \leftarrow \min\left(1, \frac{\theta}{\|\mathbf{g}\|}\right)\mathbf{g}$$

rescales to gradient of size at most θ

RNN Notebook

Paying attention to a sequence

- Not all observations are equally relevant

Paying attention to a sequence

- Not all observations are equally relevant

- Need mechanism to **pay attention (update gate)**

  e.g., an early observation is highly significant for predicting all future observations. We would like to have some mechanism for **storing/updaing** vital early information in a memory cell.

Paying attention to a sequence

- Not all observations are equally relevant

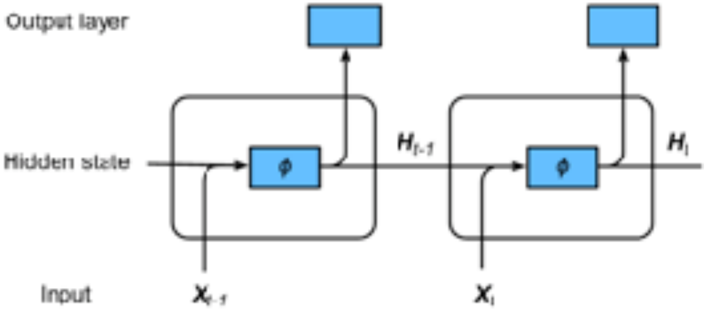
- Need mechanism to **forget (reset gate)**

  e.g., There is a logical break between parts of a sequence. For instance there might be a transition between chapters in a book, a transition between a bear and a bull market for securities, etc.

From RNN to GRU

$$\mathbf{H}_t = \phi(\mathbf{W}_{hh}\mathbf{H}_{t-1} + \mathbf{W}_{hx}\mathbf{X}_{t-1} + \mathbf{b}_h)$$

$$o_t = W_{ho}H_t + b_o$$



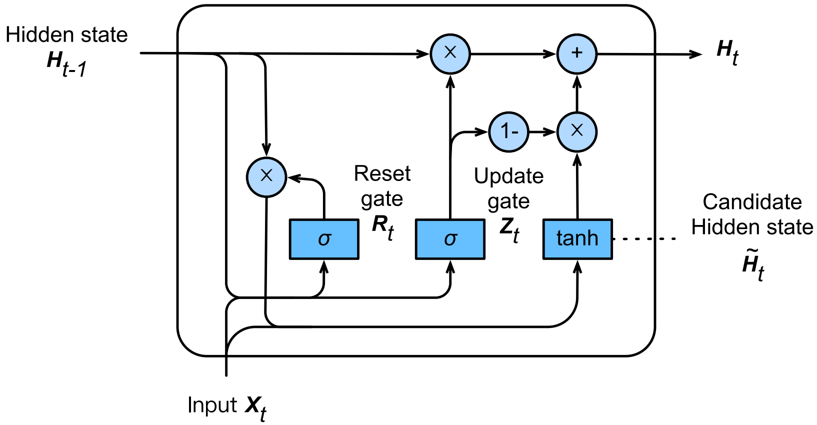
$$R_t = \sigma(X_t W_{xr} + H_{t-1}W_{hr} + b_r),$$

$$Z_t = \sigma(X_t W_{xz} + H_{t-1}W_{hz} + b_z)$$

$$\tilde{H}_t = \tanh(X_t W_{xh} + (\boxed{R_t} \odot H_{t-1})\, W_{hh} + b_h)$$

$$H_t = \boxed{Z_t} \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t$$

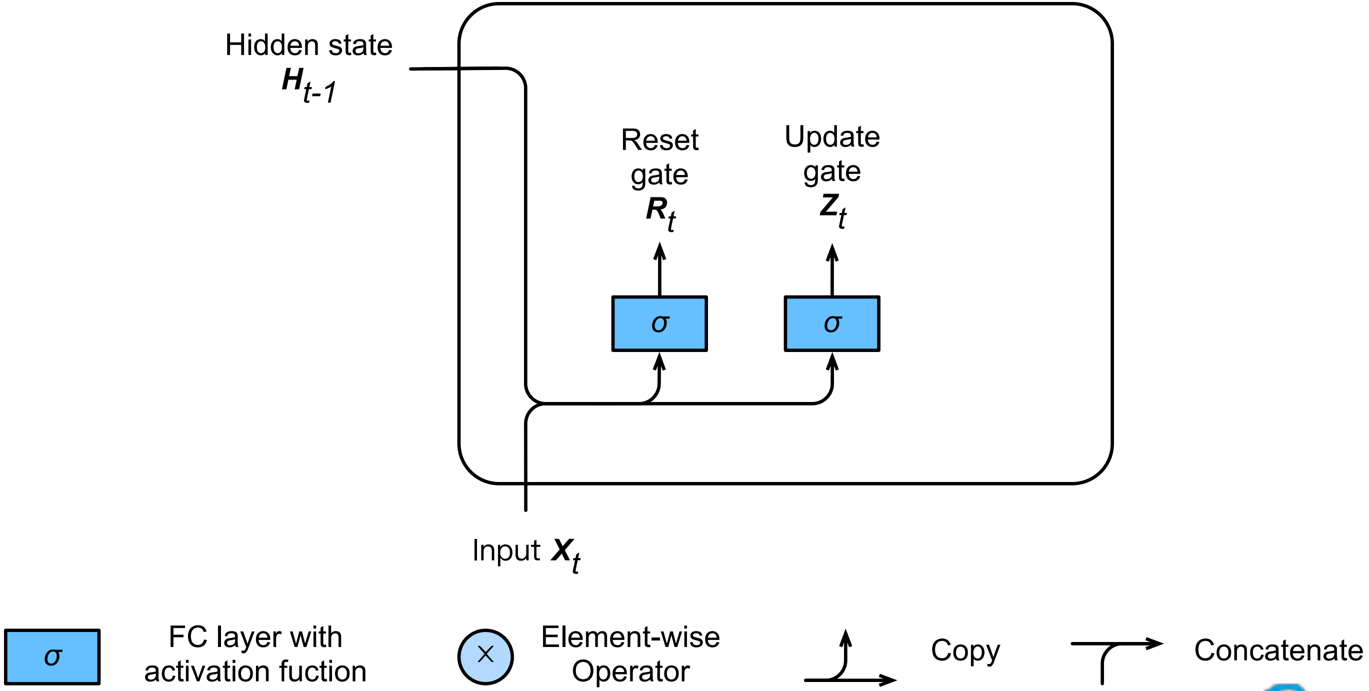
$$o_t = W_{ho}H_t + b_o$$



DIVE INTO DEEP LEARNING

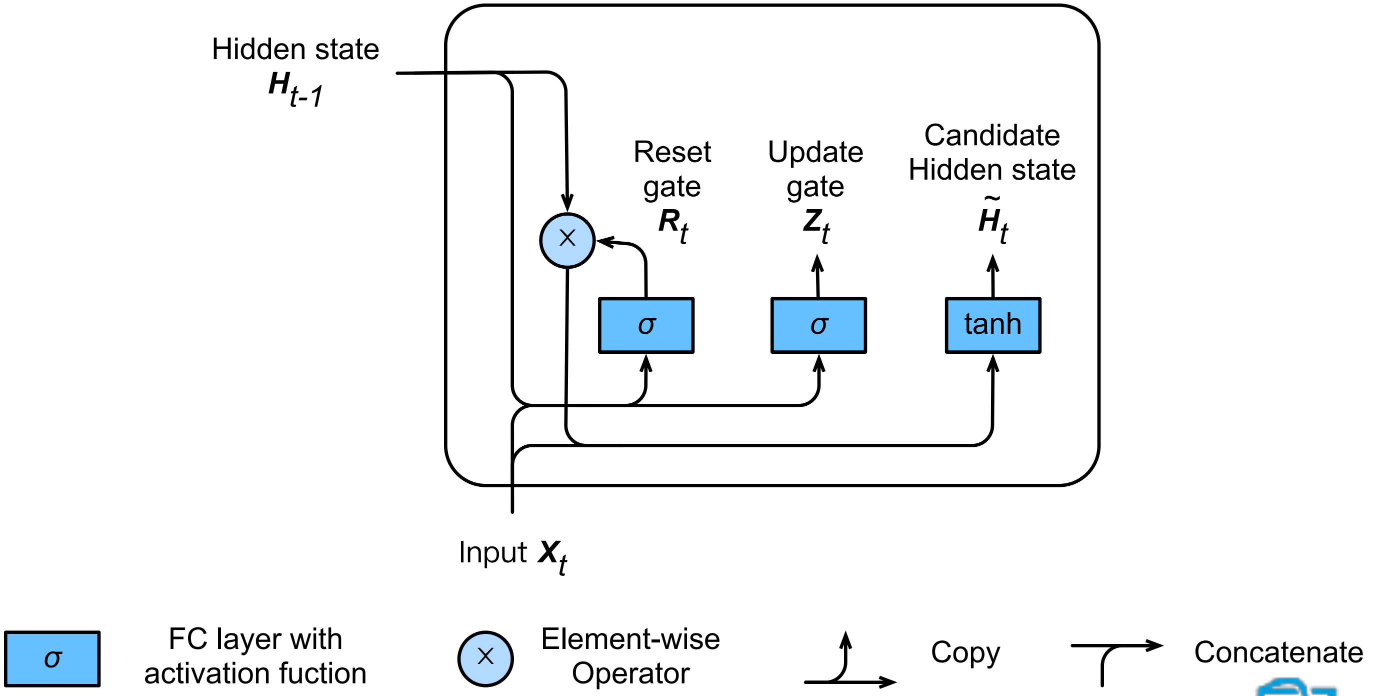
GRU - Gates

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r),$$
$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z)$$



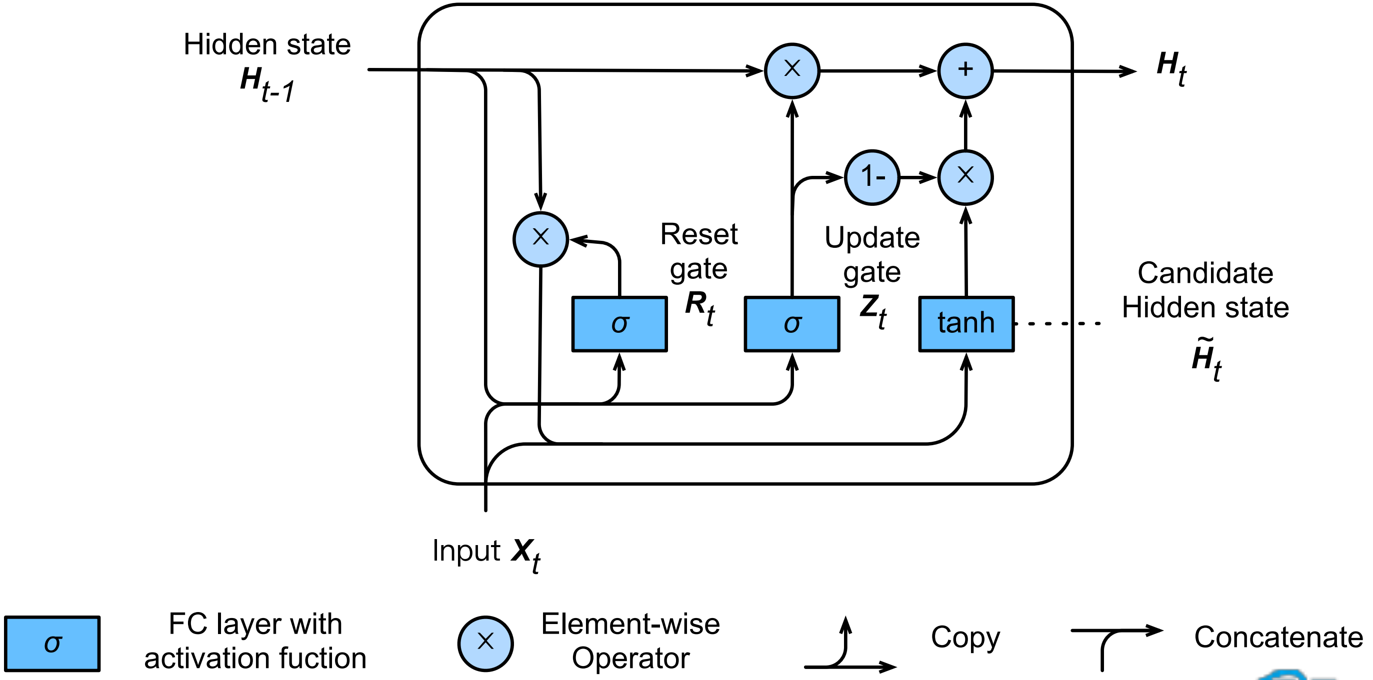
GRU - Candidate Hidden State

$$\tilde{H}_t = \tanh(X_t W_{xh} + \left(R_t \odot H_{t-1}\right) W_{hh} + b_h)$$



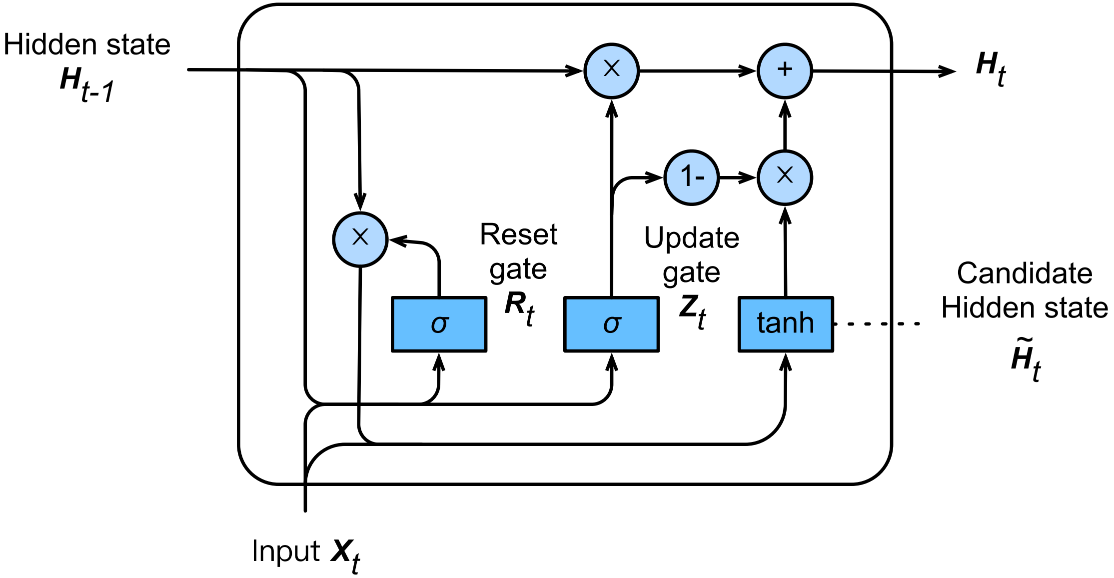
Hidden State

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t$$



Summary

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r),$$
$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z)$$
$$\tilde{H}_t = \tanh(X_t W_{xh} + \left(\boxed{R_t} \odot H_{t-1}\right) W_{hh} + b_h)$$
$$H_t = \boxed{Z_t} \odot H_{t-1} + (1 - \boxed{Z_t}) \odot \tilde{H}_t$$

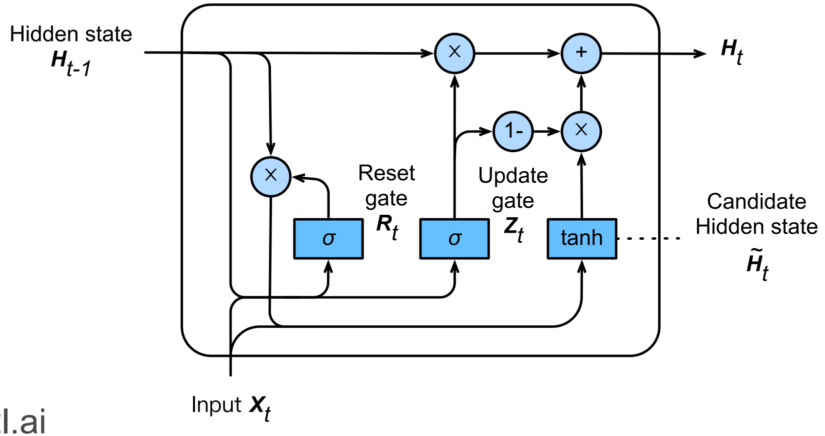


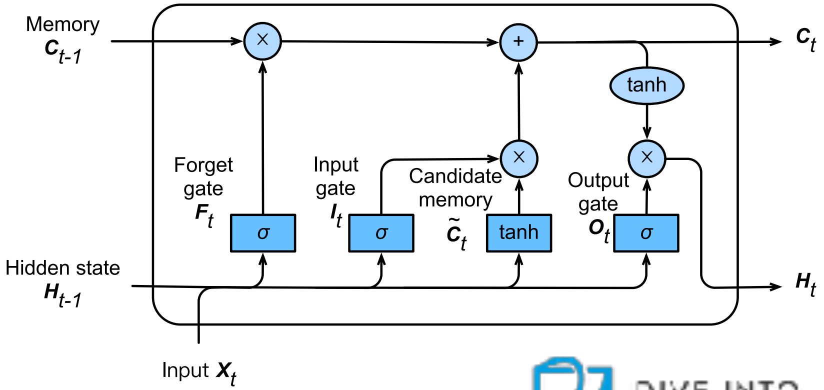
Long Short Term Memory

GRU and LSTM

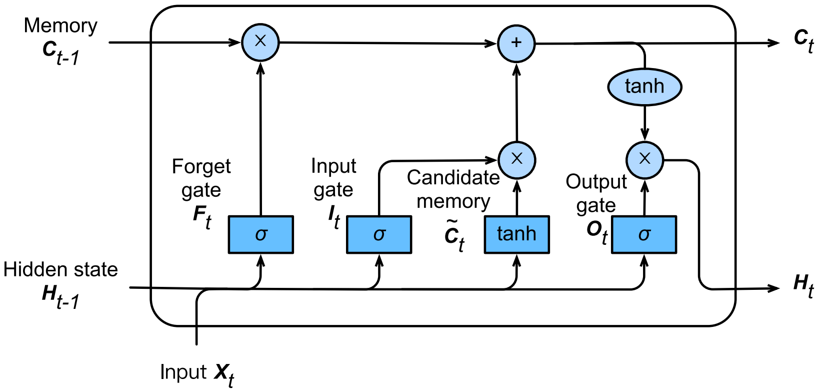
$$
\begin{aligned}
\boxed{R_t} &= \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r),\\
\boxed{Z_t} &= \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z)\\
\tilde{H}_t &= \tanh(X_t W_{xh} + \left(\boxed{R_t} \odot H_{t-1}\right) W_{hh} + b_h)\\
H_t &= \boxed{Z_t} \odot H_{t-1} + \left(\boxed{1 - Z_t}\right) \odot \tilde{H}_t
\end{aligned}
$$

$$
\begin{aligned}
\boxed{I_t} &= \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i)\\
\boxed{F_t} &= \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f)\\
\boxed{O_t} &= \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o)\\
\tilde{C}_t &= \tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c)\\
C_t &= \boxed{F_t} \odot C_{t-1} + \boxed{I_t} \odot \tilde{C}_t\\
H_t &= \boxed{O_t} \odot \tanh(C_t)
\end{aligned}
$$





Long Short Term Memory

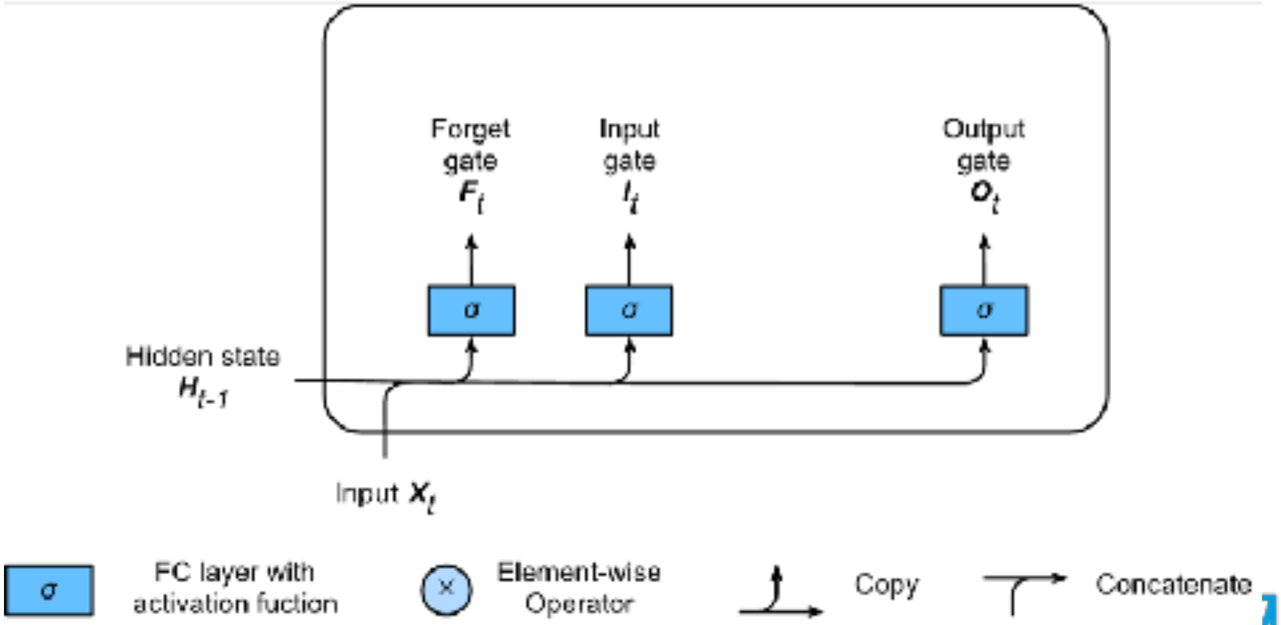


- **Forget gate**
  Reset the memory cell values
- **Input gate**
  Decide whether we should ignore the input data
- **Output gate**
  Decide whether the hidden state is used for the output generated by the LSTM

- **Hidden state** and **Memory cell**

Gates

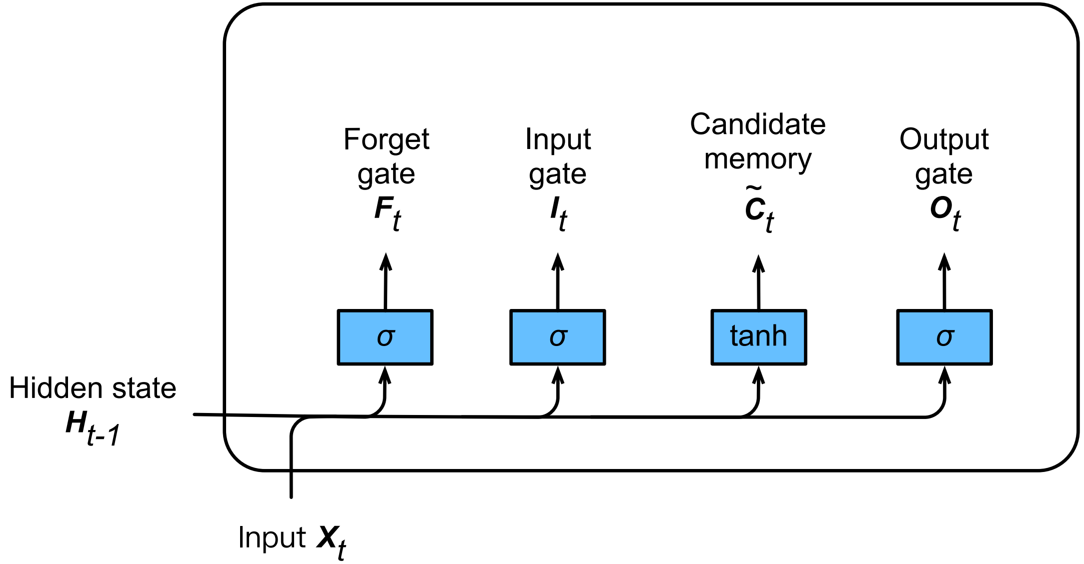
$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i)$$
$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f)$$
$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o)$$



DIVE INTO DEEP LEARNING

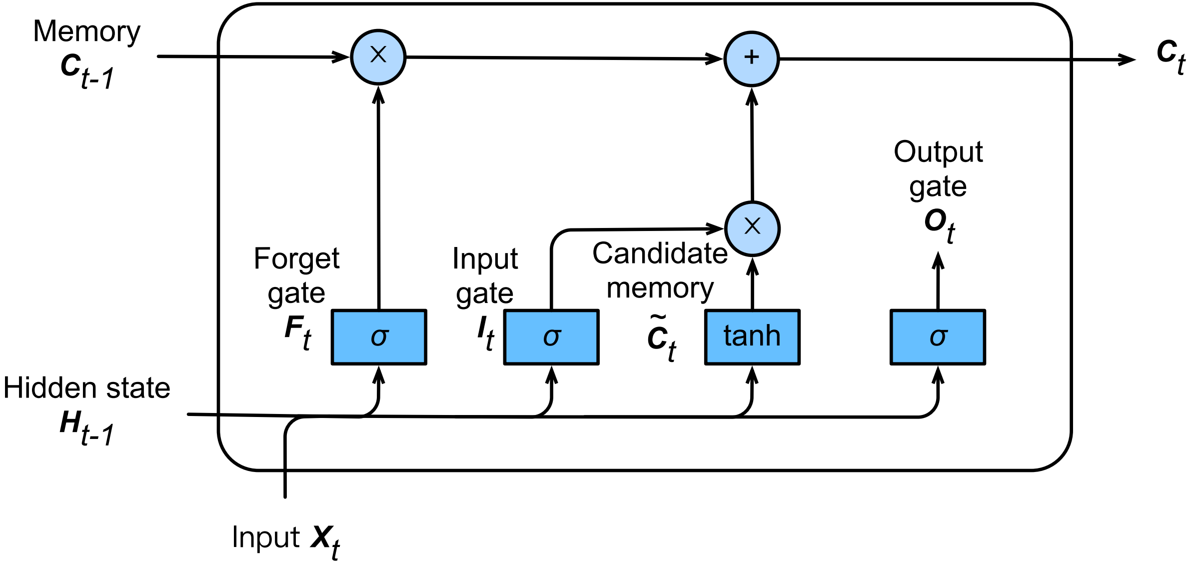
Candidate Memory Cell

$$\tilde{C}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hc} + \boldsymbol{b}_c)$$



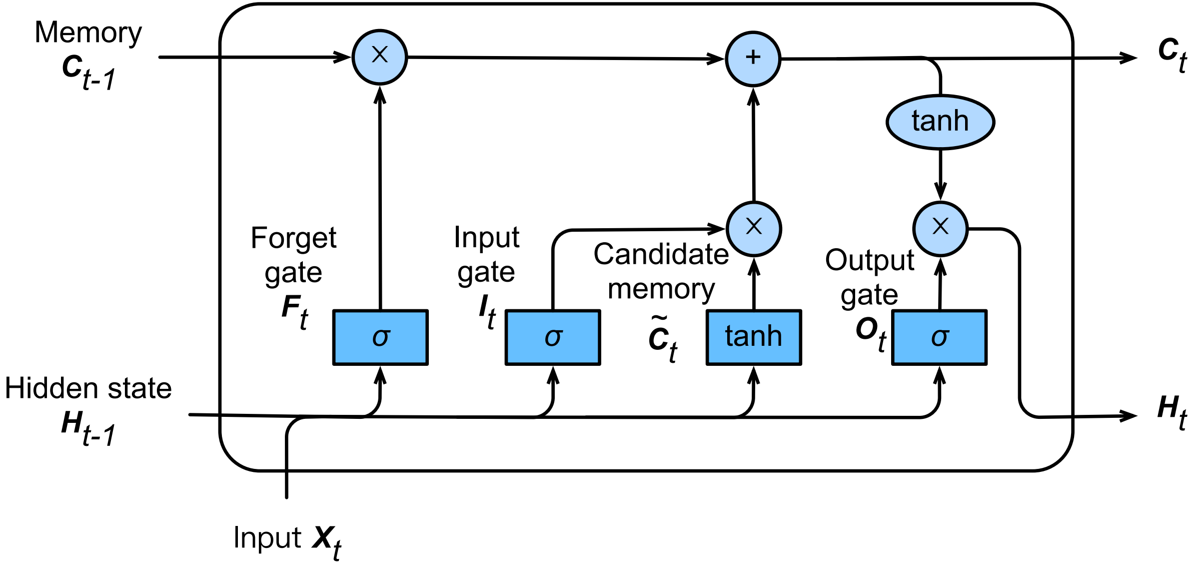
Memory Cell

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t$$



Hidden State / Output

$$H_t = O_t \odot \tanh(C_t)$$



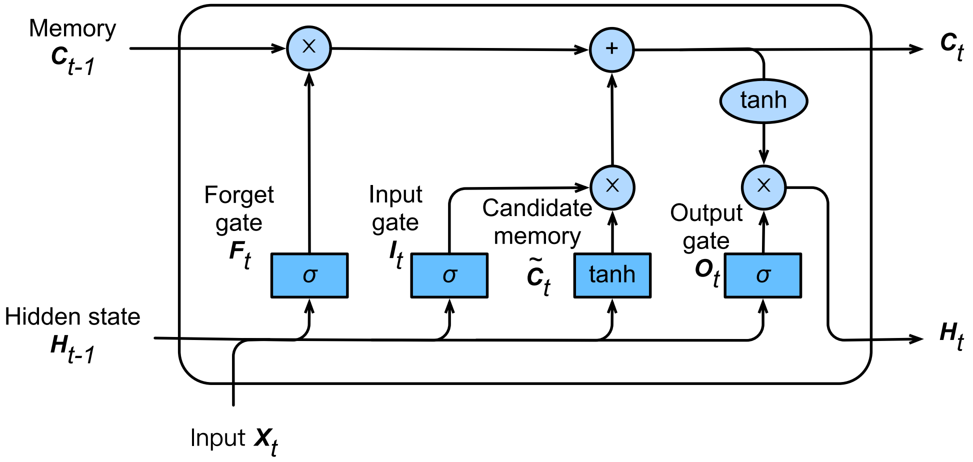
σ FC layer with activation fuction

× Element-wise Operator

Copy

Concatenate

Hidden State / Output



$$\boldsymbol{I}_t = \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hi} + \boldsymbol{b}_i)$$

$$\boldsymbol{F}_t = \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hf} + \boldsymbol{b}_f)$$

$$\boldsymbol{O}_t = \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{ho} + \boldsymbol{b}_o)$$

$$\tilde{\boldsymbol{C}}_t = \text{tanh}(\boldsymbol{X}_t\boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hc} + \boldsymbol{b}_c)$$

$$\boldsymbol{C}_t = \boldsymbol{F}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \odot \tilde{\boldsymbol{C}}_t$$

$$\boldsymbol{H}_t = \boldsymbol{O}_t \odot \text{tanh}(\boldsymbol{C}_t)$$

LSTM Notebook

Bidirectional RNNs

```
I am _____
I am _____ very hungry,
I am _____ very hungry, I could eat half a pig.
```

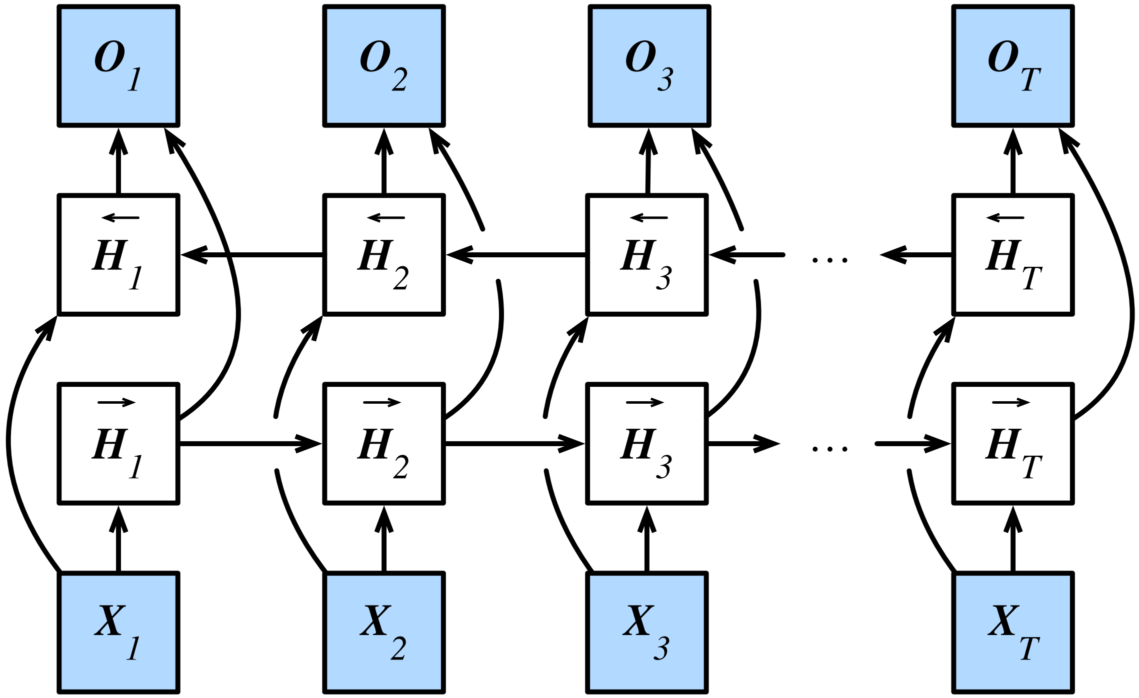
```
I am hungry.
I am not     very hungry,
I am very    very hungry, I could eat half a pig.
```

The Future Matters

```
I am hungry.
I am not      very hungry,
I am very     very hungry, I could eat half a pig.
```

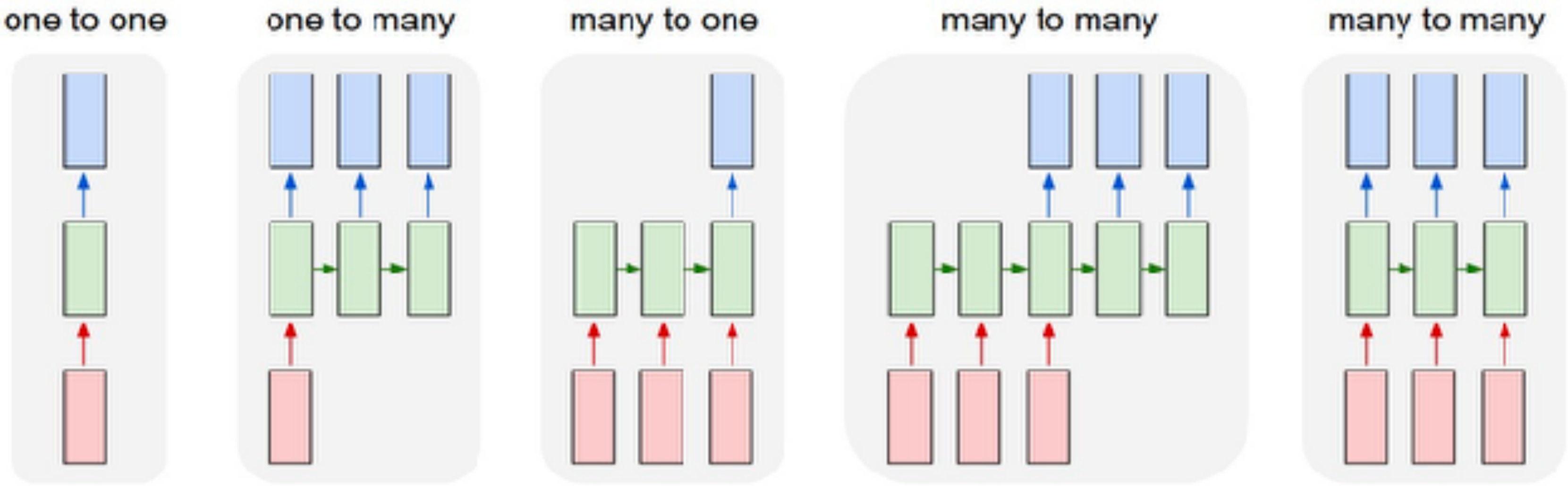
- Very different words to fill in, depending on past and **future** context of a word.
- RNNs so far only look at the past
- In interpolation (fill in) we can use the future, too.

Bidirectional RNN



- One RNN forward
- Another one backward
- Combine both hidden states for output generation

Using RNNs



Poetry Generation

Sentiment Analysis

Document Classification

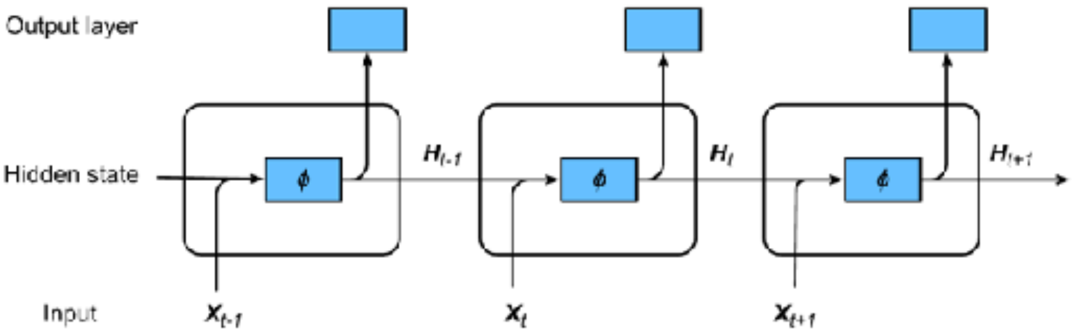
Question Answering

Machine Translation

Named Entity Tagging

(image courtesy of karpathy.github.io)

Recall - RNNs Architecture



- Hidden State update

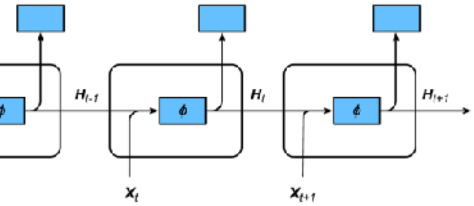
$$\mathbf{H}_t = \phi(\mathbf{W}_{hh}\mathbf{H}_{t-1} + \mathbf{W}_{hx}\mathbf{X}_{t-1} + \mathbf{b}_h)$$

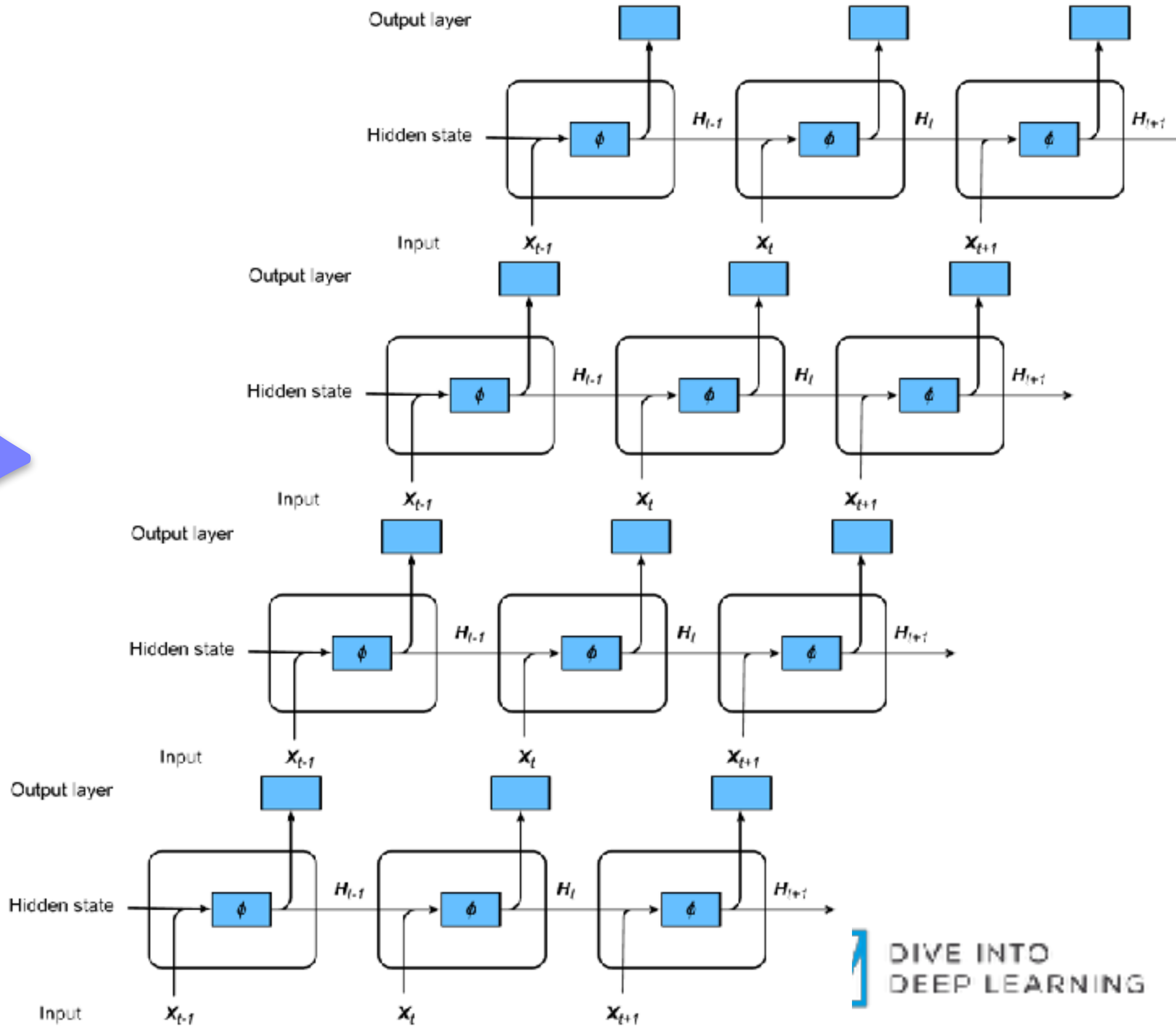
- Observation update

$$\mathbf{o}_t = \mathbf{W}_{ho}\mathbf{H}_t + \mathbf{b}_o$$

How to make more nonlinear?

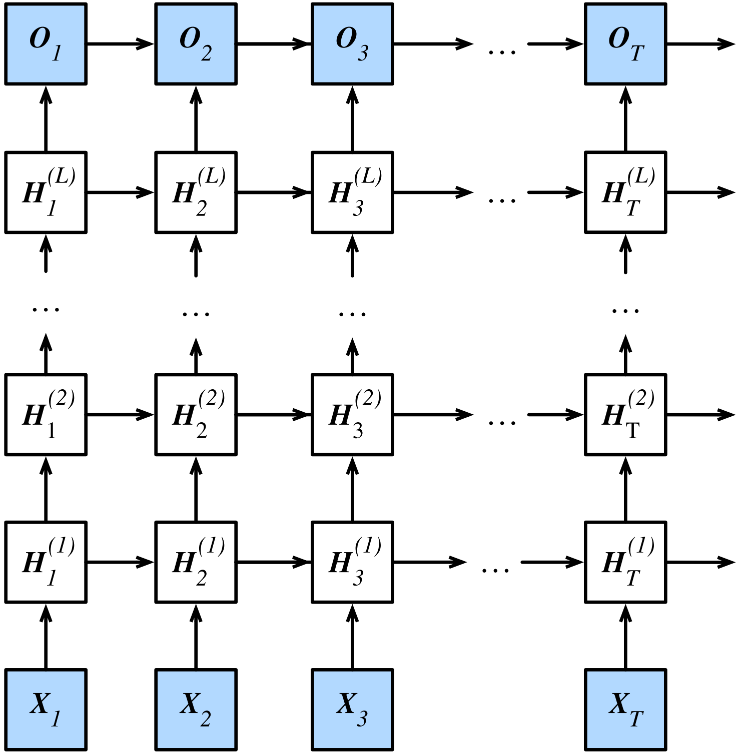
We go deeper





DIVE INTO
DEEP LEARNING

We go deeper



- Shallow RNN
  - Input
  - Hidden layer
  - Output

$$\mathbf{H}_t = f(\mathbf{H}_{t-1}, \mathbf{X}_t)$$
$$\mathbf{O}_t = g(\mathbf{H}_t)$$

- Deep RNN
  - Input
  - **Hidden layer**
  - **Hidden layer**

  ...
  - Output

$$\mathbf{H}_t^1 = f_1(\mathbf{H}_{t-1}^1, \mathbf{X}_t)$$
$$\mathbf{H}_t^j = f_j(\mathbf{H}_{t-1}^j, \mathbf{H}_t^{j-1})$$
$$\mathbf{O}_t = g(\mathbf{H}_t^L)$$

DIVE INTO DEEP LEARNING

Summary

- Dependent Random Variables
- Text Preprocessing
- Language Modeling
- Recurrent Neural Networks (RNN)
- LSTM
- Bidirectional RNN
- Deep RNN

Math

- Linear algebra, Prob, Calculus & Statistics Gradient

Machine learning

- Loss function
- Regularization
- Model selection
- Environment

Optimizatio

- Convex Optimization Momentum, RMSProp, Adam

Attention

- Seq2seq w/ attention
- Transformer
- BERT

Basic

- NDarray
- Autograd
- Gluon

Basic models

- Linear regression
- Image classification
- Softmax regression
- Multilayer perceptron

RNNs and

- Recurrent networks (RNN, GRU, LSTM) for language modeling
- Word embedding
- Seq2seq for machine translation

Performanc

- Numerical stability
- Multi-GPU Training

CNN

- Convolution, LeNet
- Alex, VGG, Inception, ResNet

CV

- Data Augmentation
- Fine-tuning
- Object detection
- Segmentation

GAN

- Generative Adversarial Networks
- DCGAN

More Contents in D2L.ai

- What we covered
- Not

DIVE INTO DEEP LEARNING

Resources

- Textbook: numpy.d2l.ai
- Toolkit for computer vision: gluon-cv.mxnet.io
- Toolkit for natural language processing: gluon-nlp.mxnet.io
- Toolkit for time series: gluon-ts.mxnet.io
- Toolkit for graph neural networks: dgl.ai
- Discussion forum: https://discuss.mxnet.io/