Recurrent Neural Network

MLRS 2019

Rachel Hu

Amazon AI

Outline

- Sequence models and language models
- Recurrent neural networks (RNN)
- GRU and LSTM
- Deep RNNs, Bi-RNNs

Last but not the least ...

AWS Setting-up

1. The region

Asia Pacific (Mumbai)

Asia Pacific (Osaka-Local)

Asia Pacific (Seoul)

Asia Pacific (Singapore)

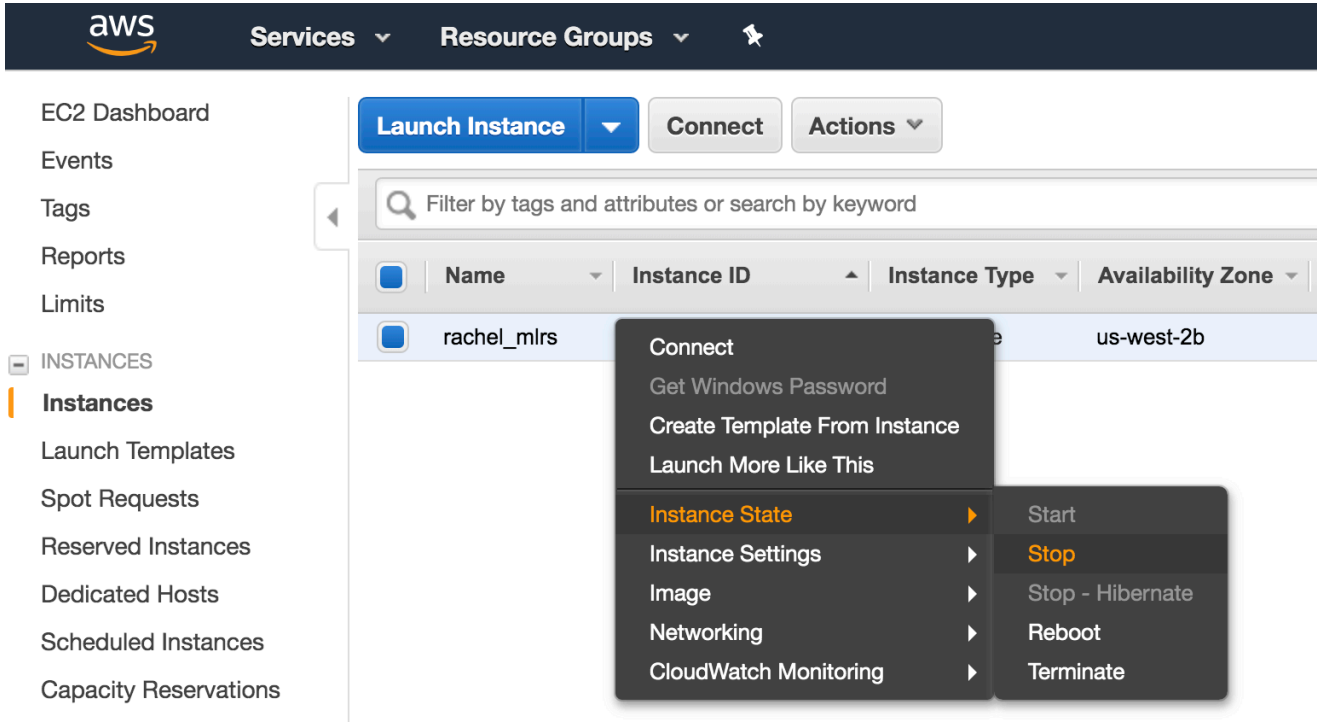
Asia Pacific (Sydney)

Asia Pacific (Tokyo)

aws

AWS Setting-up - Billing

2. Billing: Don’t forget to stop it. (right click your instance to stop)



AWS Setting-up - Billing

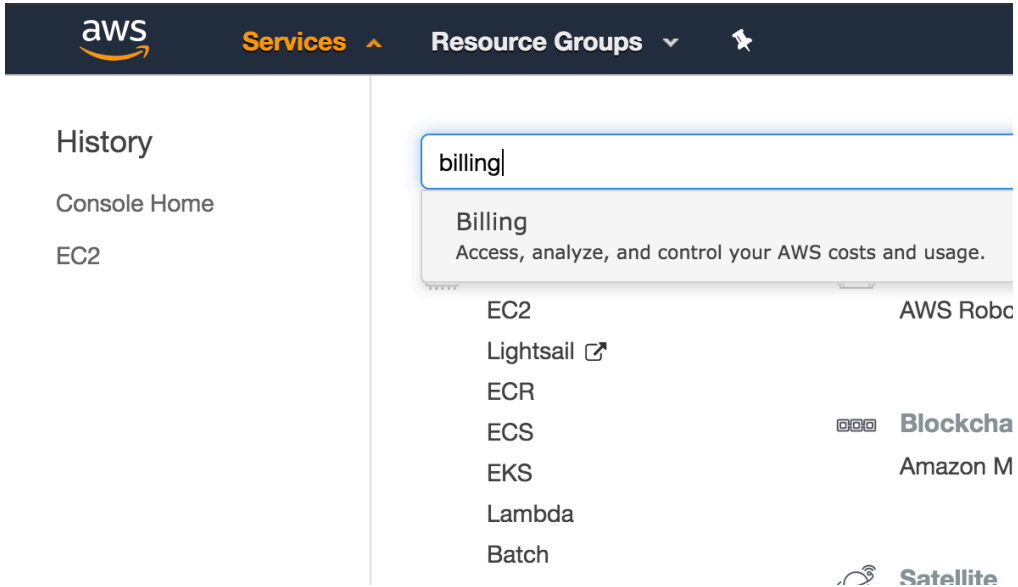
2. Billing

“Stopping” an instance means that you will be able to start it again. (However, stopped instances will still be billed a small amount for the hard disk space retained.)

“Terminate” deletes all data associated with it.

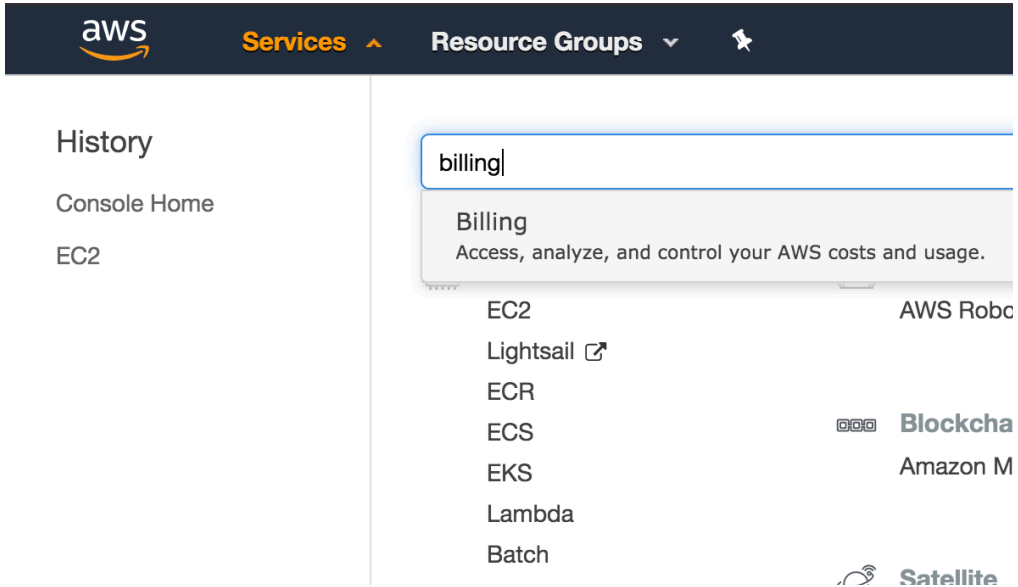
AWS Setting-up - Billing

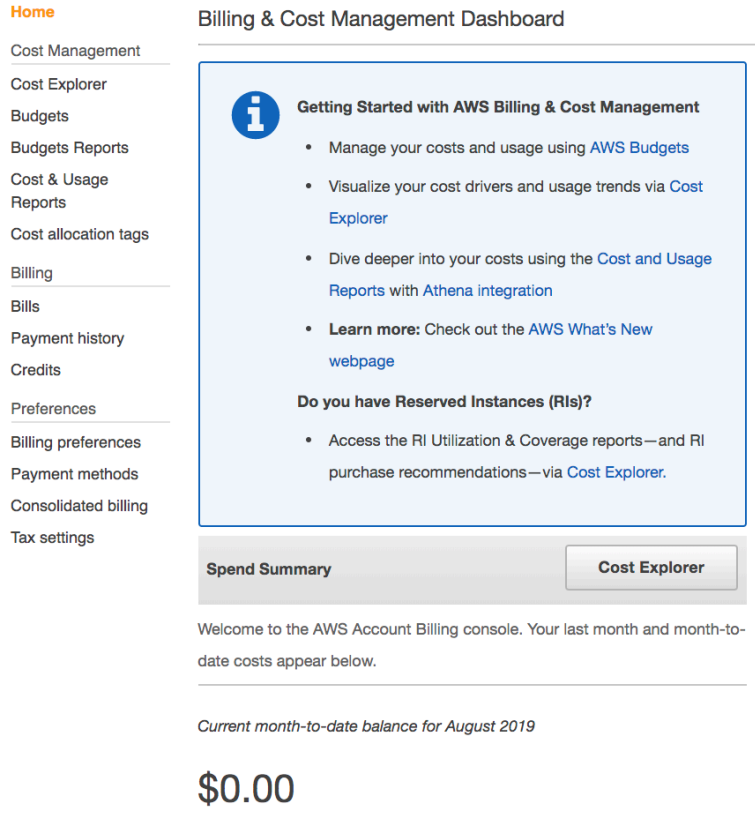
Enter “billing” in the search box



AWS Setting-up - Billing

Enter “billing” in the search box





AWS Deepracer

Who is this???

https://aws.amazon.com/deepracer/

aws

AWS Deepracer League

https://aws.amazon.com/deepracer/

You can access the AWS DeepRacer 3D racing simulator in the AWS console to train your models, evaluate them, and take part in the the AWS DeepRacer League.

AWS Deepracer League

https://aws.amazon.com/deepracer/

How do I get started with AWS DeepRacer simulator?

The AWS DeepRacer simulator provides a tutorial to get you started.

The AWS DeepRacer Developer Documentation provides additional details on building your first model and also how to improve your models.

Recommended Courses/Textbooks/Resources (Online)

https://github.com/goldmermaid/mlrs

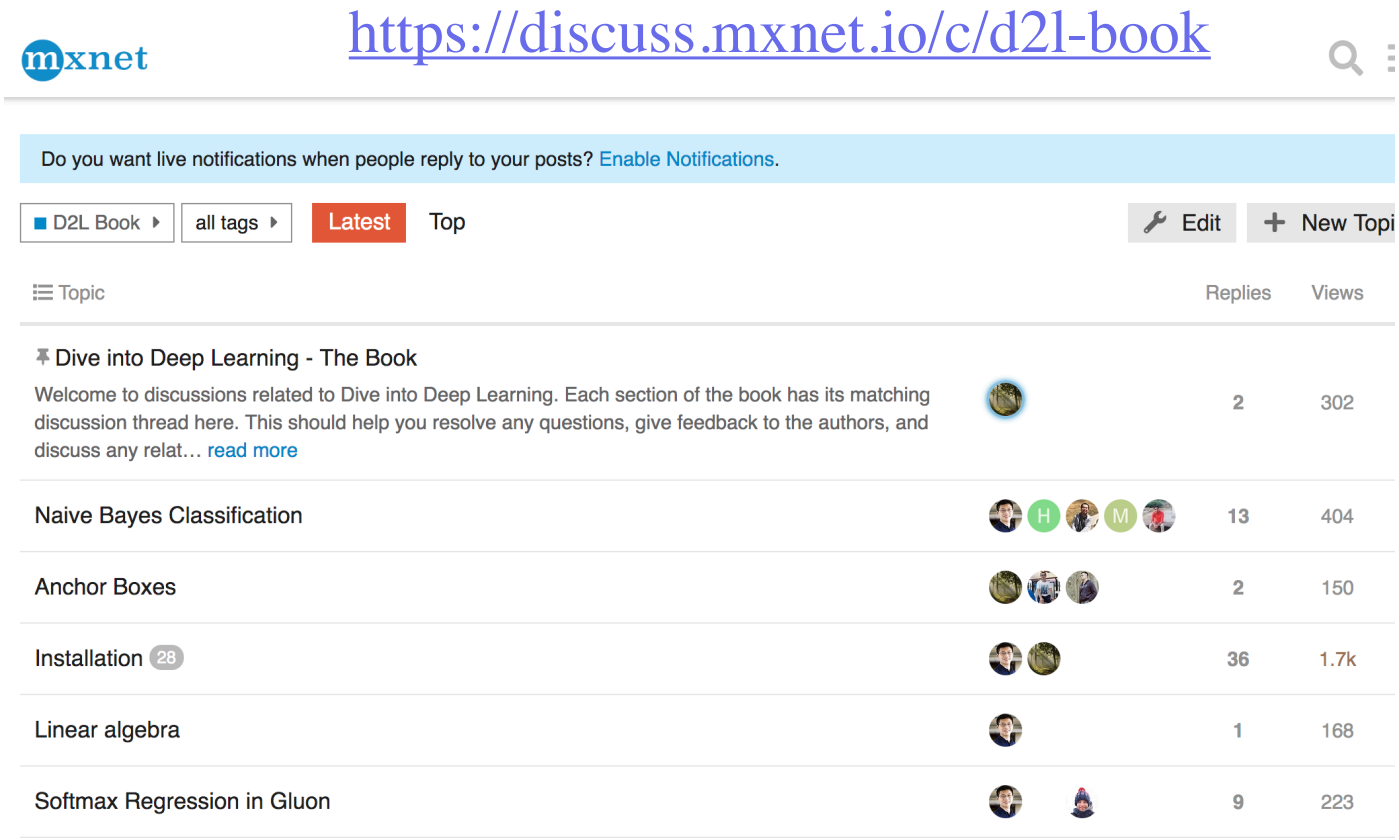
Title	Website	Notes
Introduction to Deep Learning	Course	Introductory to medium level deep learning course (that we taught at UC Berkeley). Lecture videos, slides, homework, etc. are all available online.
Introduction to Machine Learning	Course	Introductory level machine learning course. Lecture videos, slides, homework, etc. are all available online.
The Foundations of Data Science	Course	Introductory level machine learning course. Lecture videos, slides, homework, etc. are all available online.
Deep Reinforcement Learning	Course	Advanced level reinforcement learning course. Lecture videos, slides, homework, etc. are all available online.

Recommended Courses/Textbooks/Resources (Online)

https://github.com/goldmermaid/mlrs

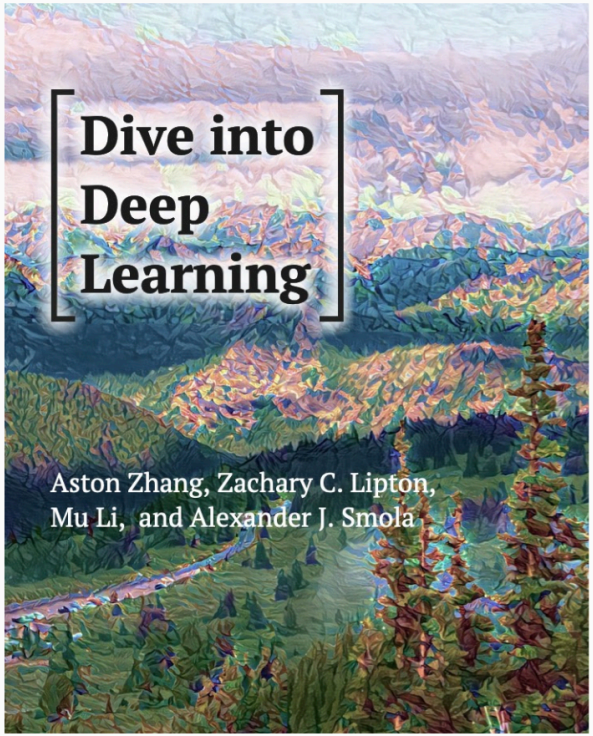
Dive into Deep Learning	Textbook	Textbook with interactive jupyter notebook to get your hand dirty on deep learning.
GluonCV	GitHub	A python toolkit which provides implementations of the state-of-the-art (SOTA) deep learning models in computer vision.
GluonNLP	GitHub	A python toolkit that enables easy text preprocessing, datasets loading and neural models building to help you speed up your Natural Language Processing (NLP) research.
GluonTS	GitHub	A python toolkit for probabilistic time series modeling, which provides utilities for loading and iterating over time series datasets, state of the art models ready to be trained, and building blocks to define your own models and quickly experiment with different solutions.
MXNet Online Discussion	Website	An online protal to discuss deep learning, machine learning, how to do things efficiently in MxNet/Gluon, ask for help, or suggest new things.

Just ask online!



aws

D2L Textbooks http://numpy.d2l.ai/



Dive into Deep Learning

An interactive deep learning book with code, math, and discussions

Based on the NumPy interface

The contents are under revision

aws

D2L Textbooks http://numpy.d2l.ai/

Aston Zhang

Amazon Applied Scientist
UIUC Ph.D.

Zack C. Lipton

Amazon Applied Scientist
CMU Assistant Professor
UCSD Ph.D.

Mu Li

Amazon Principal Scientist
CMU Ph.D.

Alex J. Smola

Amazon VP/Distinguished Scientist
TU Berlin Ph.D.

We thank all the community contributors
for making this open source book better for everyone.

Contribute to the book.

aws

Last but not the least

Please reach me at rlhu@amazon.com for further cooperations, if you would like to:

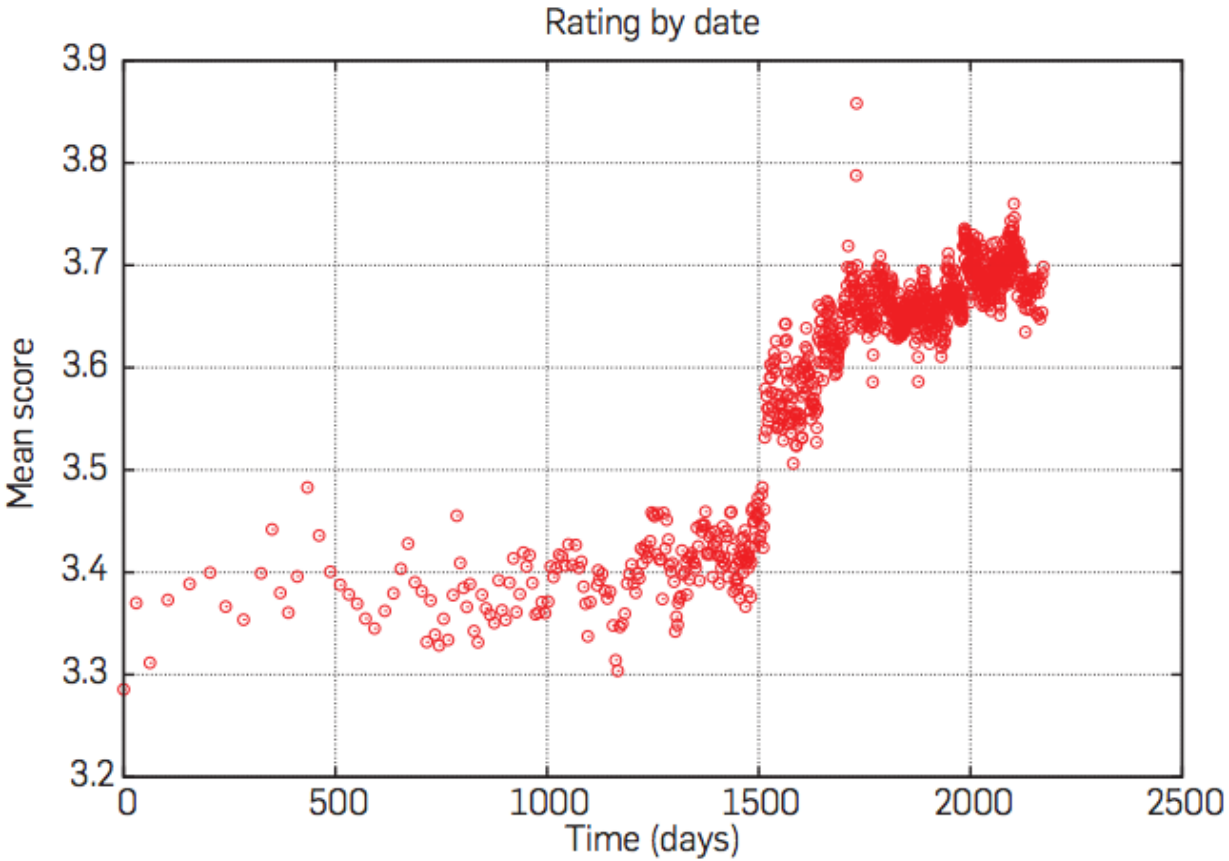
1. teach Deep Learning with our textbook;
2. translate to different languages;
3. complain about the errors, typos, unclearness, etc. about the book;

Dependent Random Variables

Data

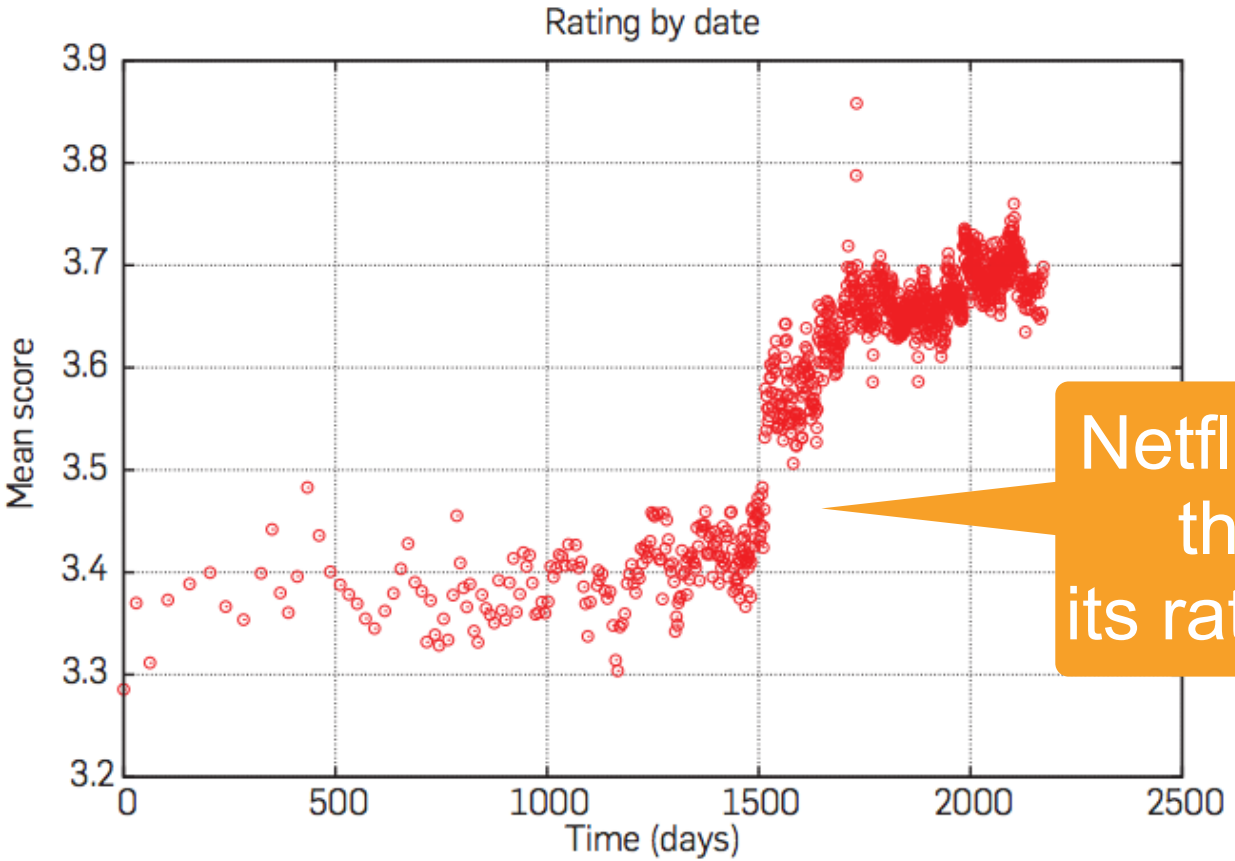
- So far …
  - Collect observation pairs $(x_i, y_i) \sim p(x, y)$ for training
  - Estimate $y \mid x \sim p(y \mid x)$ for unseen $x' \sim p(x)$
- Examples
  - Images classification & objects recognition
  - Disease prediction
  - Housing price prediction
- **The order of the data does not matter**

Time matters (Koren, 2009)



Yehuda Koren, 2009

Time matters (Koren, 2009)

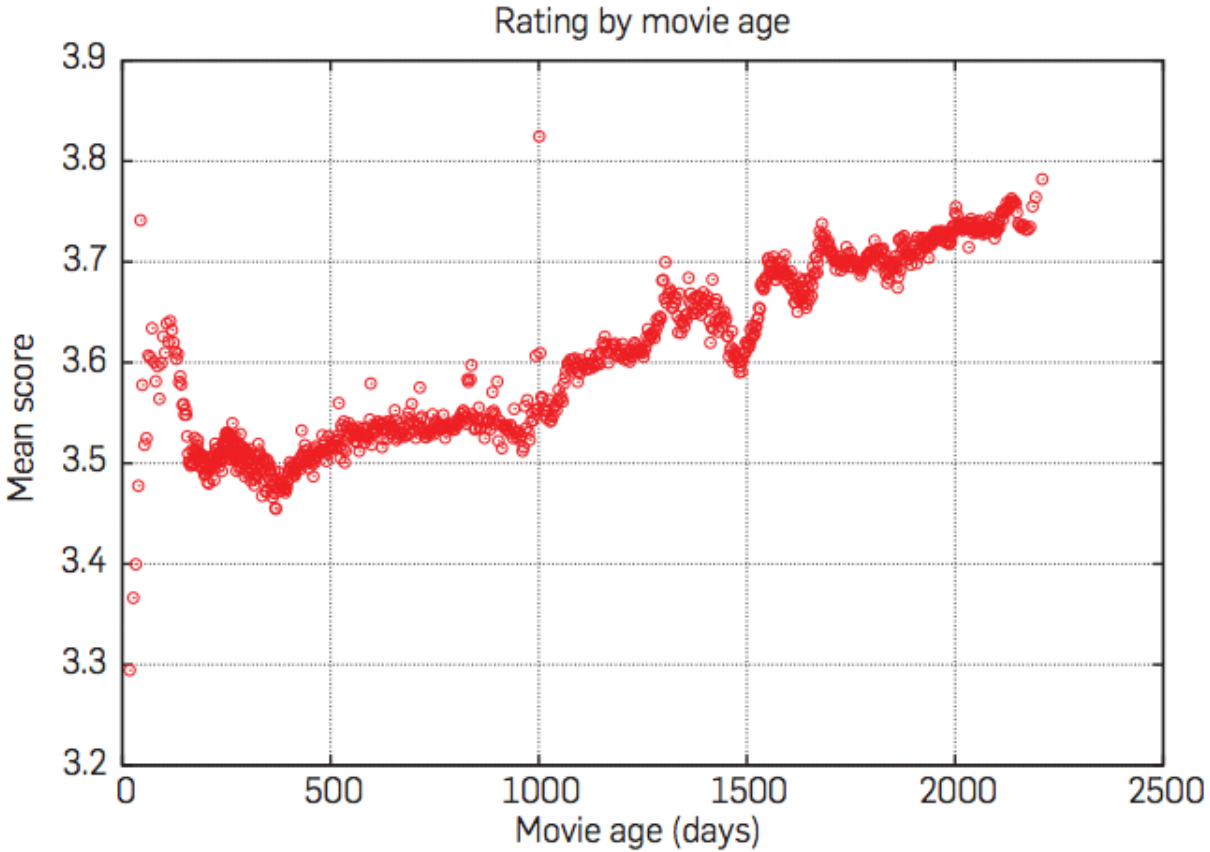


Netflix changed the labels its rating system

Yehuda Koren, 2009

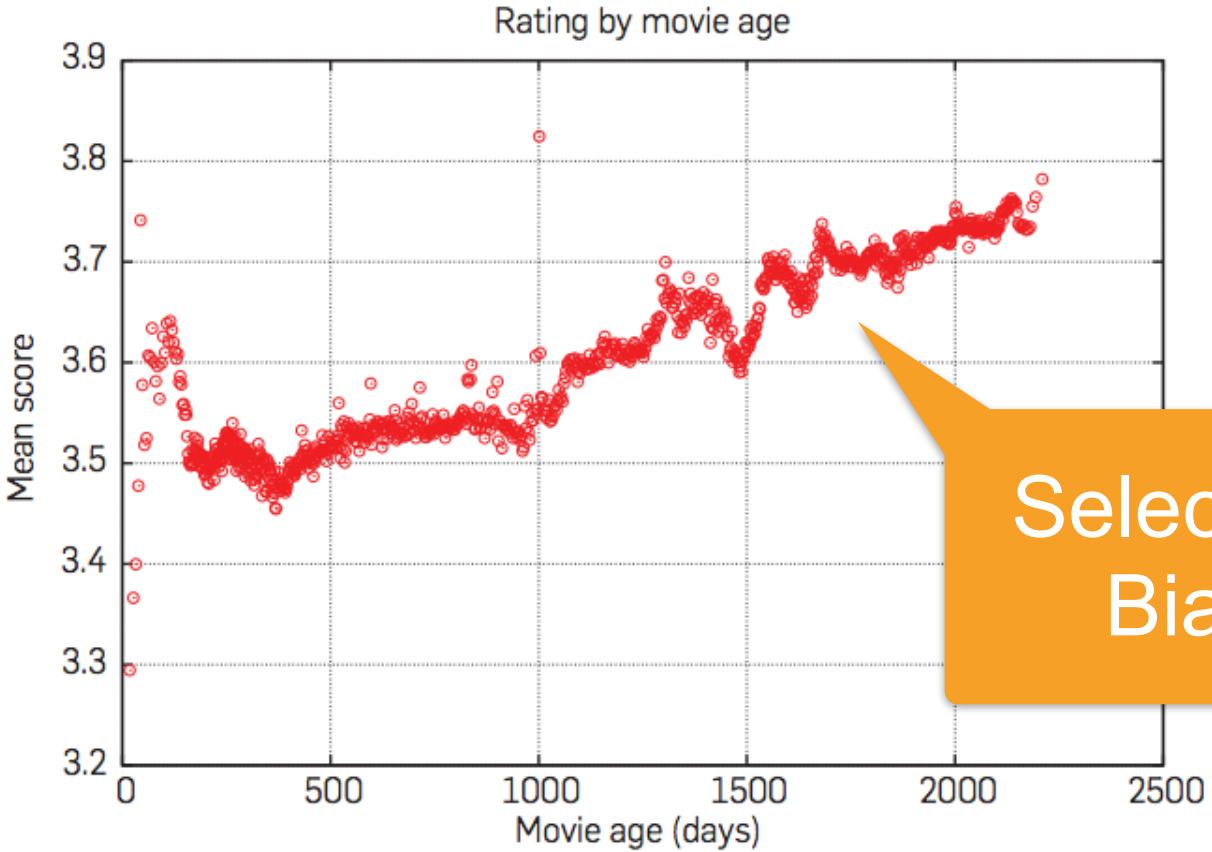
aws

Time matters (Koren, 2009)



Yehuda Koren, 2009

Time matters (Koren, 2009)

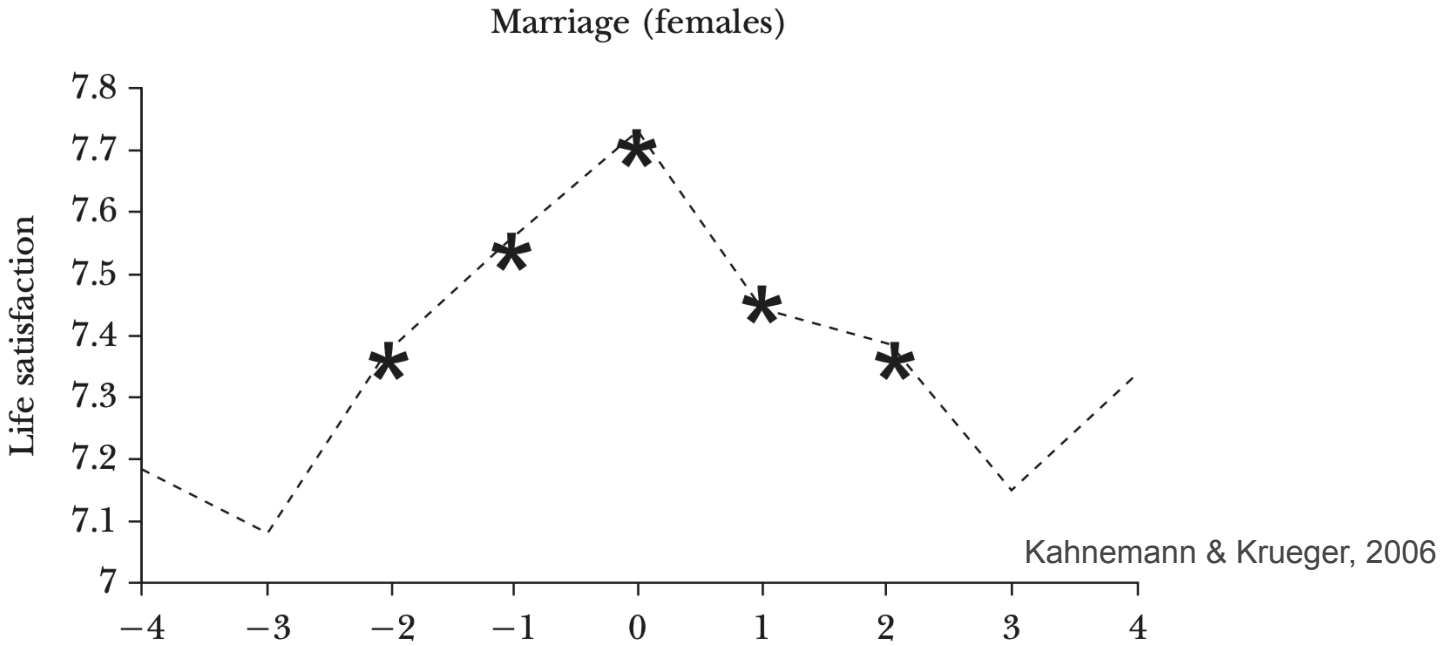


Selection Bias

Yehuda Koren, 2009

Average Life Satisfaction for a Sample of German Women

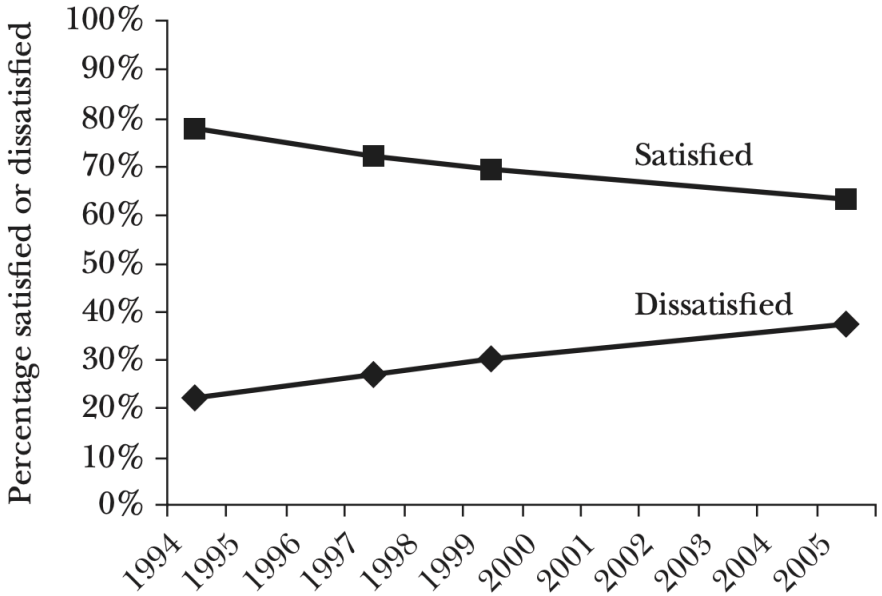
(by year of marriage $t = 0$)



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.

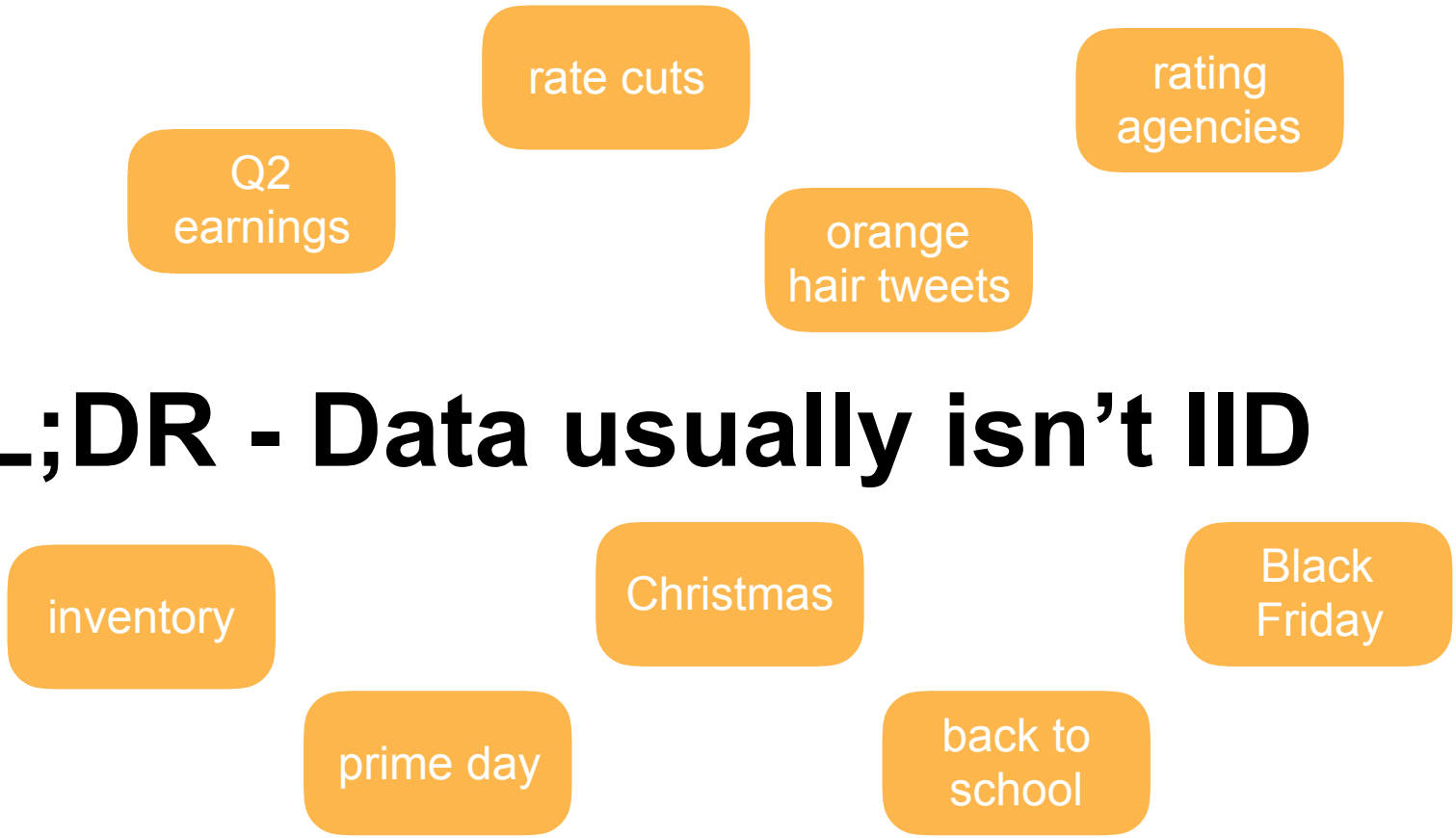
Life Satisfaction in China as Average Real Income Rises by 250 Percent

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?



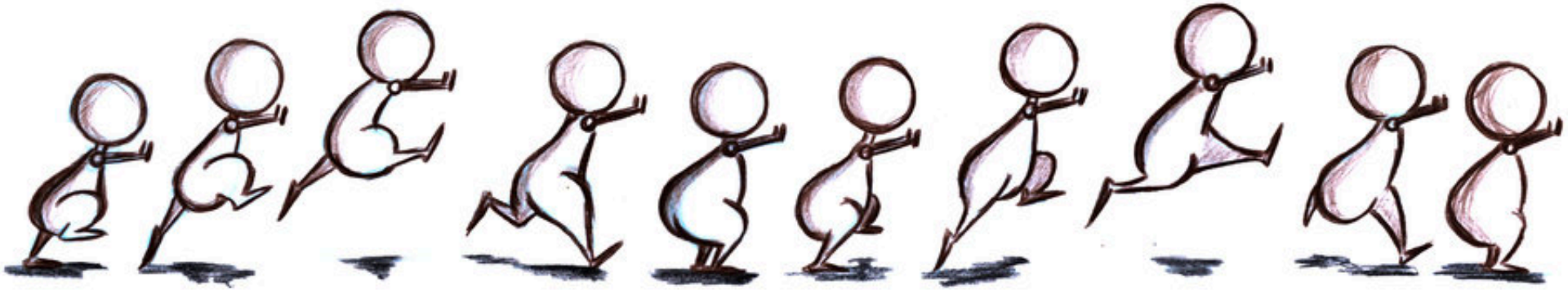
Kahnemann & Krueger, 2006

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

aws

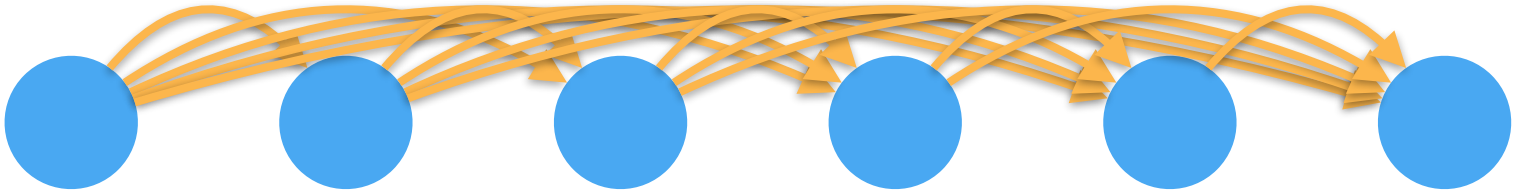
Sequence Models



Sequence Model

$$(x_1, \ldots x_T) \sim p(x)$$

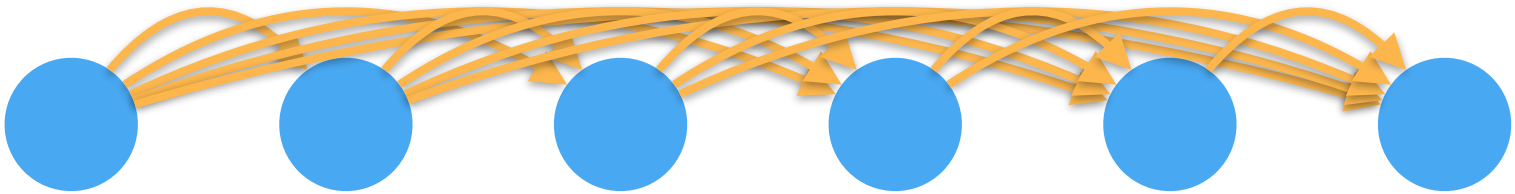
$$p(x) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_1, x_2) \cdot \ldots p(x_T \mid x_1, \ldots x_{T-1})$$



Sequence Model

$$(x_1, \ldots x_T) \sim p(x)$$

$$p(x) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_1, x_2) \cdot \ldots p(x_T \mid x_1, \ldots x_{T-1})$$



Exponentially expensive

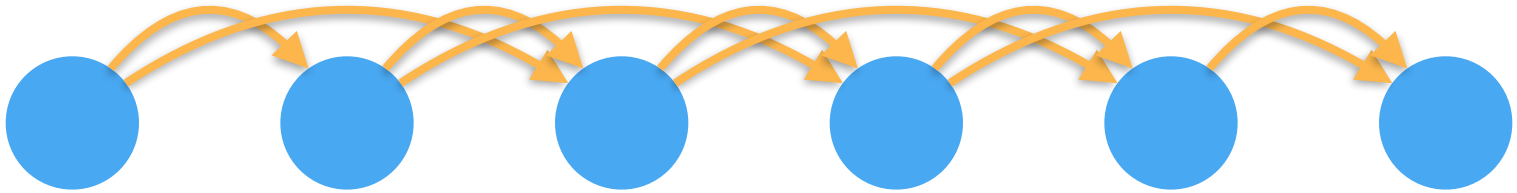
- Autoregressive model

$$p(x_t \mid x_1, \ldots x_{t-1}) = p(x_t \mid f(x_1, \ldots x_{t-1}))$$

Some function of previously seen data

Plan A - Markov Assumption

$$p(x) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_1, x_2) \cdot \ldots p(x_T \mid x_{T-\tau}, \ldots x_{T-1})$$



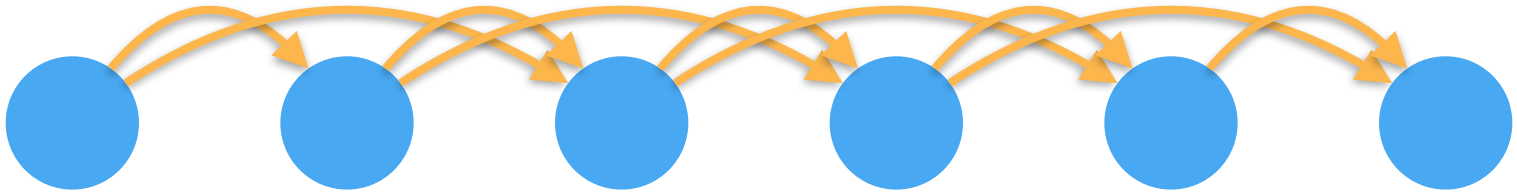
- Assume that only a few steps in the past matter
- Autoregressive model

$$p(x_t \mid x_1, \ldots x_{t-1}) = p(x_t \mid f(x_{t-\tau}, \ldots x_{t-1}))$$

Some function of previously seen data

Plan A - Markov Assumption

$$p(x) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_1, x_2) \cdot \ldots p(x_T \mid x_{T-\tau}, \ldots x_{T-1})$$



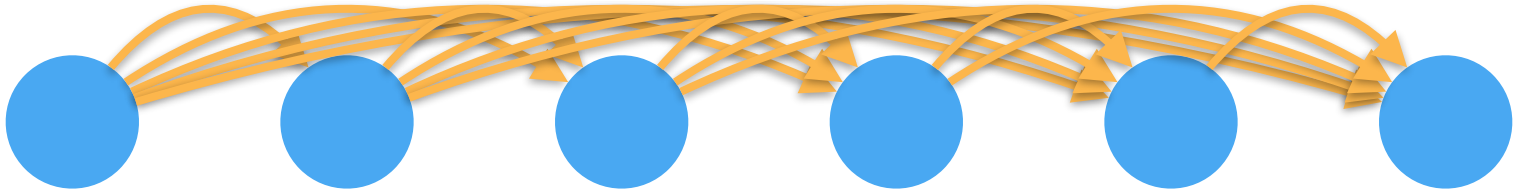
- In practice solve regression problem

$$\hat{x}_t = f(x_{t-\tau}, \ldots x_{t-1})$$

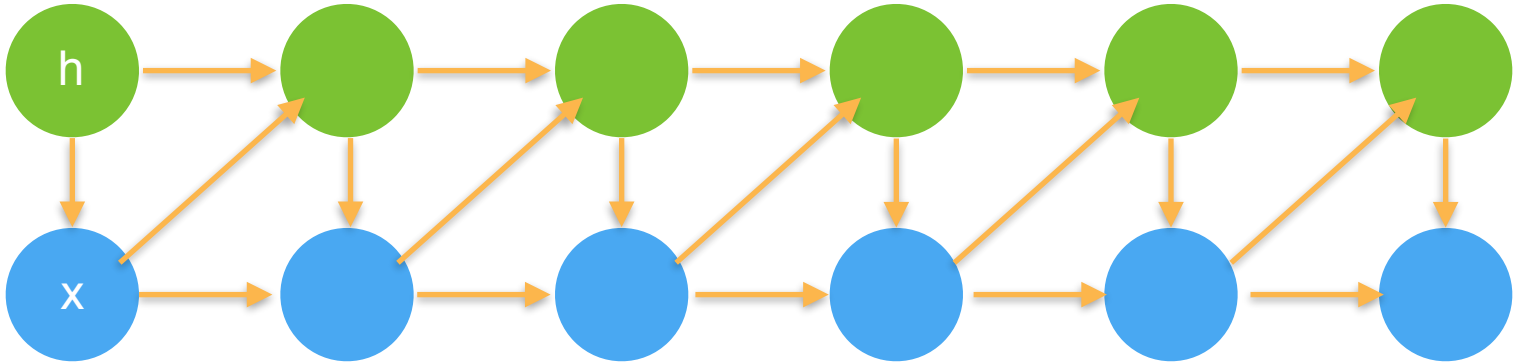
e.g. train an MLP on previously seen data

Plan B - Latent Variable Model

$$p(x) = p(x_1) \cdot p(x_2 \mid x_1) \cdot p(x_3 \mid x_1, x_2) \cdot \ldots p(x_T \mid x_1, \ldots x_{T-1})$$



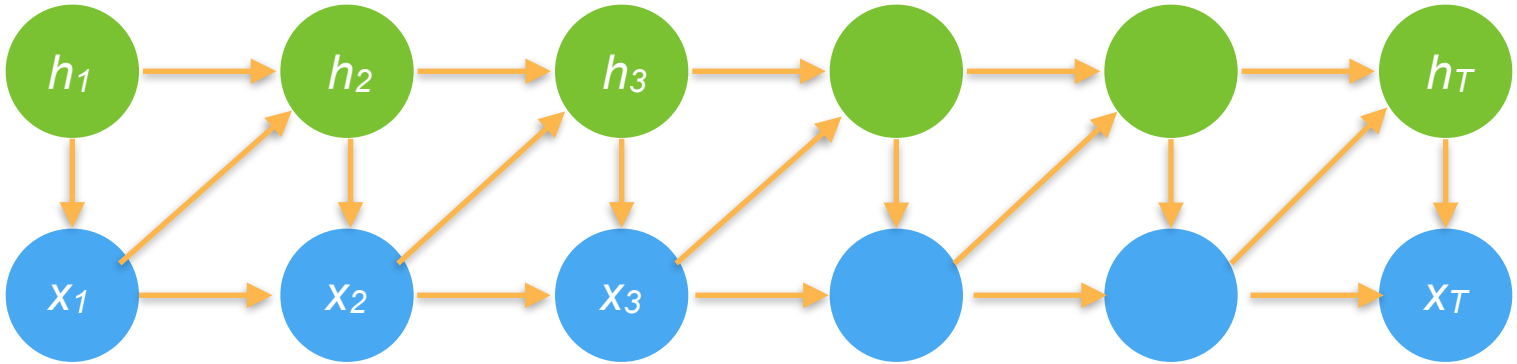
$p(h_t \mid h_{t-1}, x_{t-1})$ and $p(x_t \mid h_t, x_{t-1})$



Plan B - Latent Variable Model

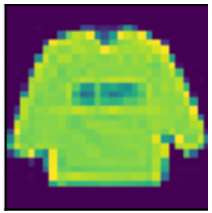
- Latent state, h, summarizes all the relevant information about the past. So we get $h_t = f(x_1, \ldots x_{t-1}) = f(h_{t-1}, x_{t-1})$

$p(h_t \mid h_{t-1}, x_{t-1})$ and $p(x_t \mid h_t, x_{t-1})$



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
```

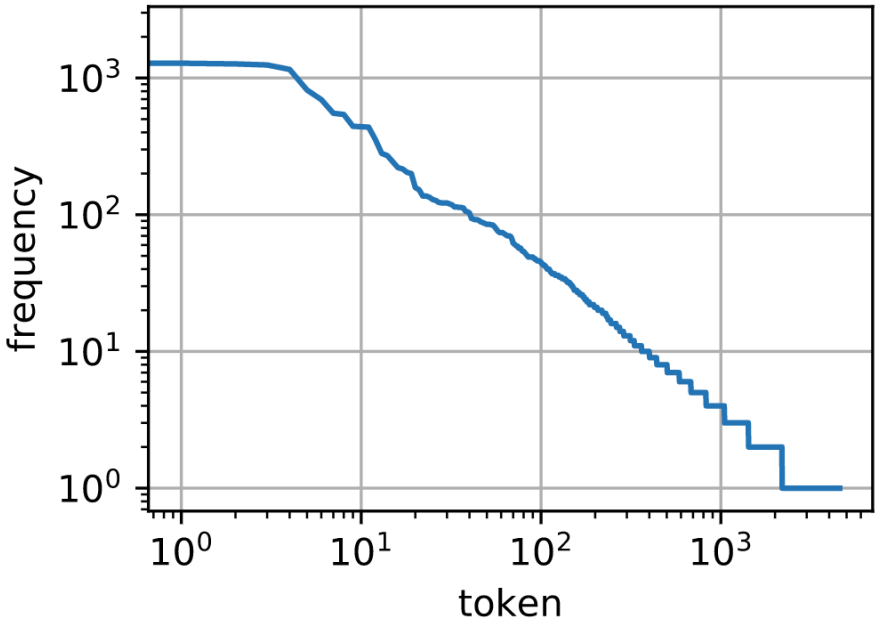
(batch size, sentence length)

Tokenization

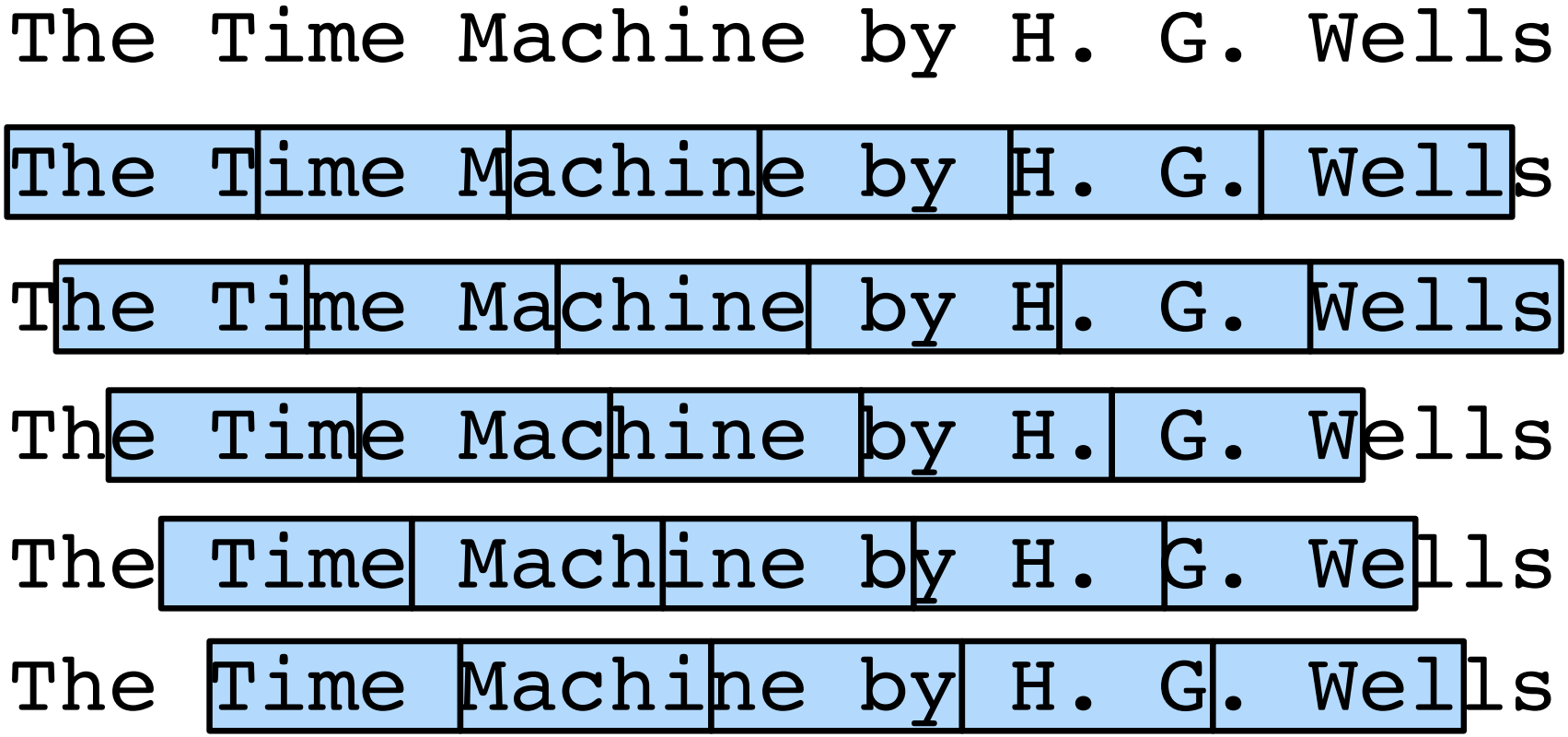
- Basic Idea - map text into sequence of tokens
  - “Deep learning is fun” -> [“Deep”, “learning”, “is”, “fun”, “.”]
- **Character Encoding** (each character as a token)
  - Small vocabulary
  - Doesn’t work so well (needs to learn spelling)
- **Word Encoding** (each word as a token)
  - Accurate spelling
  - 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



Text Preprocessing Notebook

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.,*

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

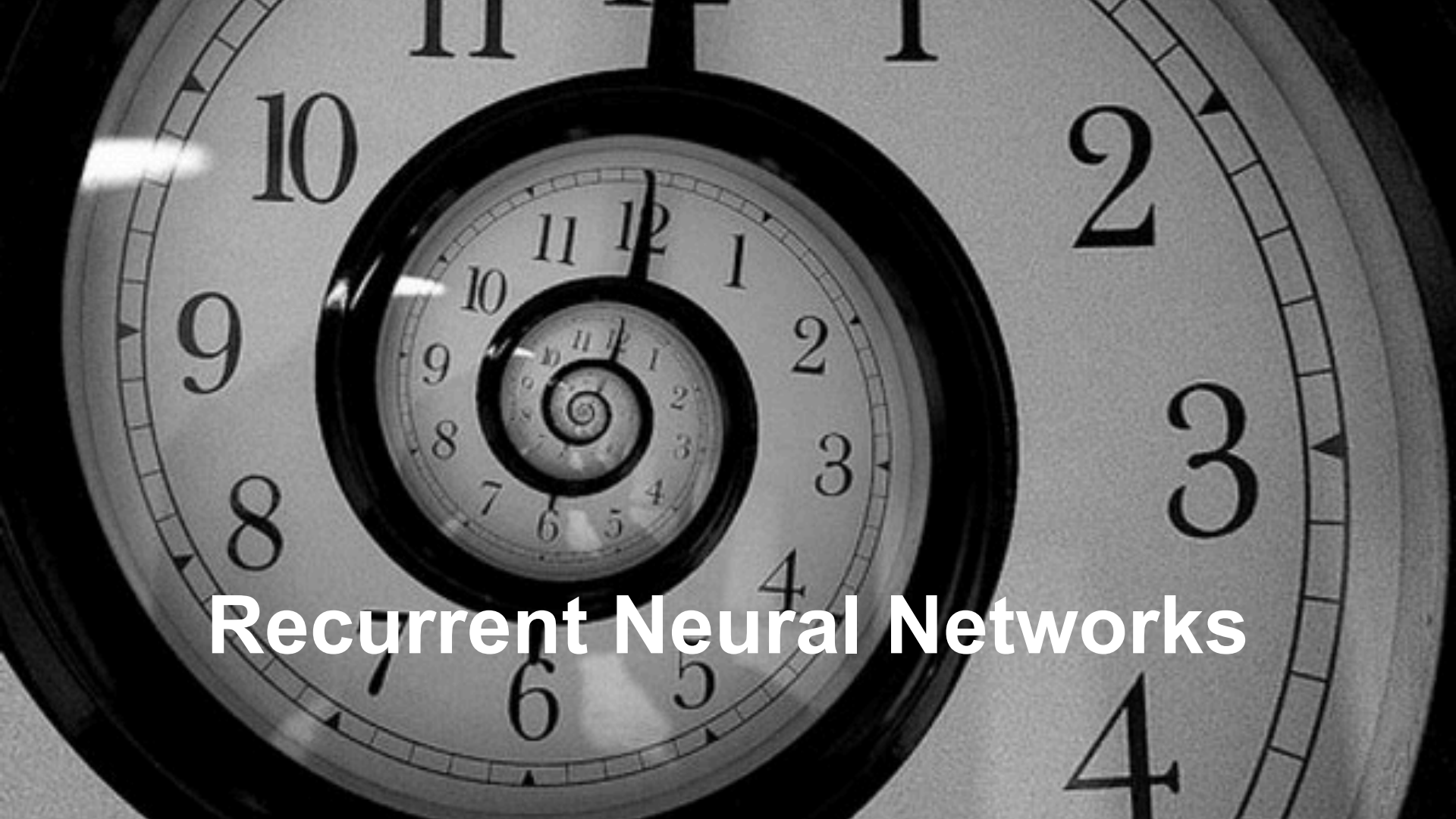
- Estimating it

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

Need Smoothing

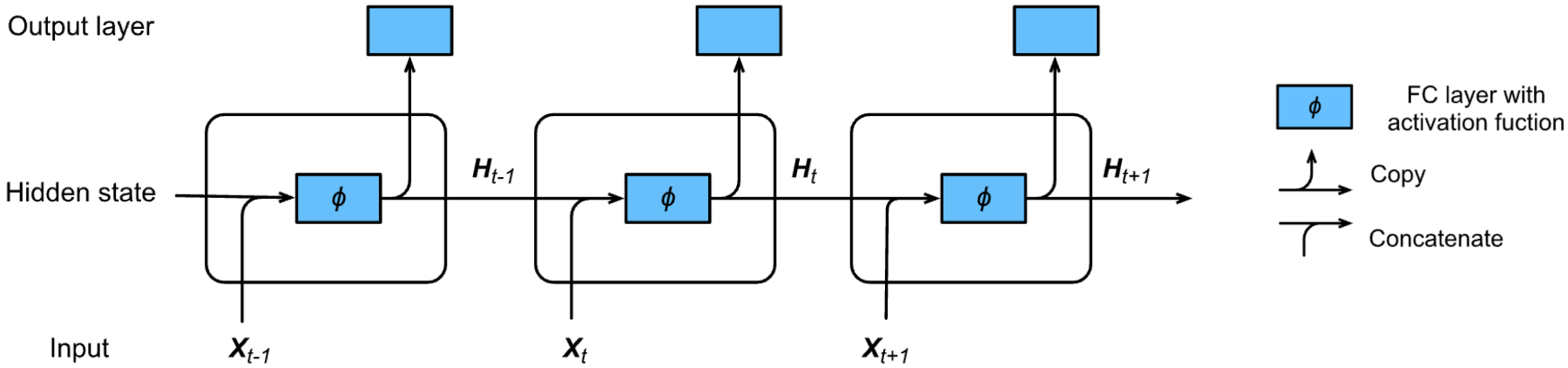
Language Modeling

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

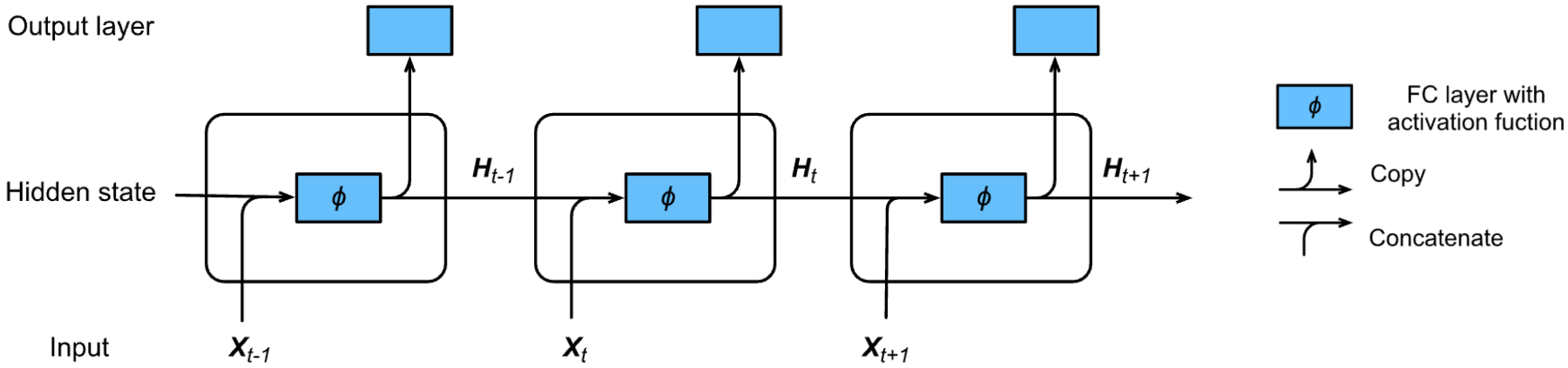


Recurrent Neural Networks

RNN with Hidden States



RNN with Hidden States



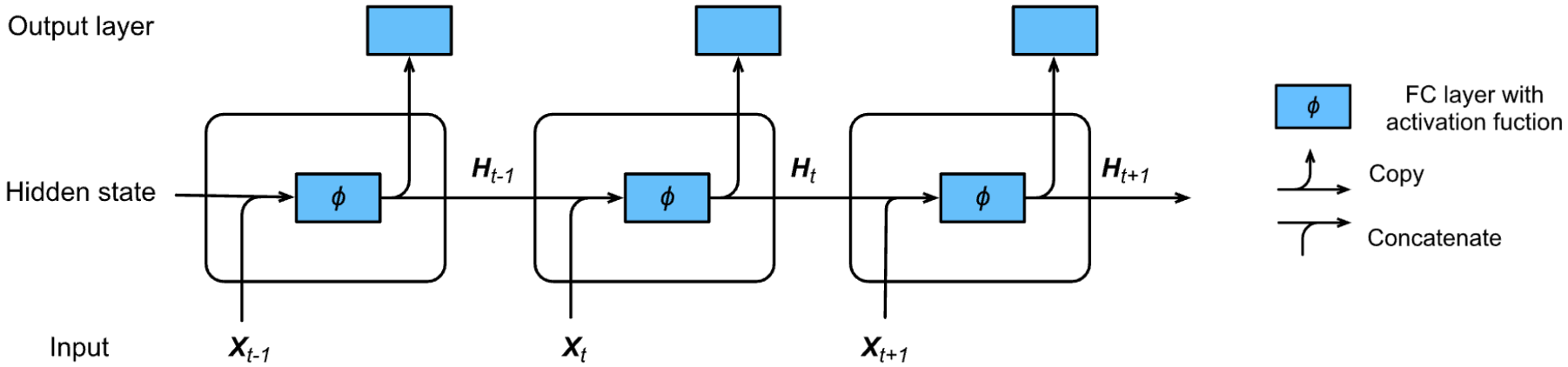
- Hidden State update

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

- Observation update

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

RNN with Hidden States



- 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

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

- 2-layer MLP

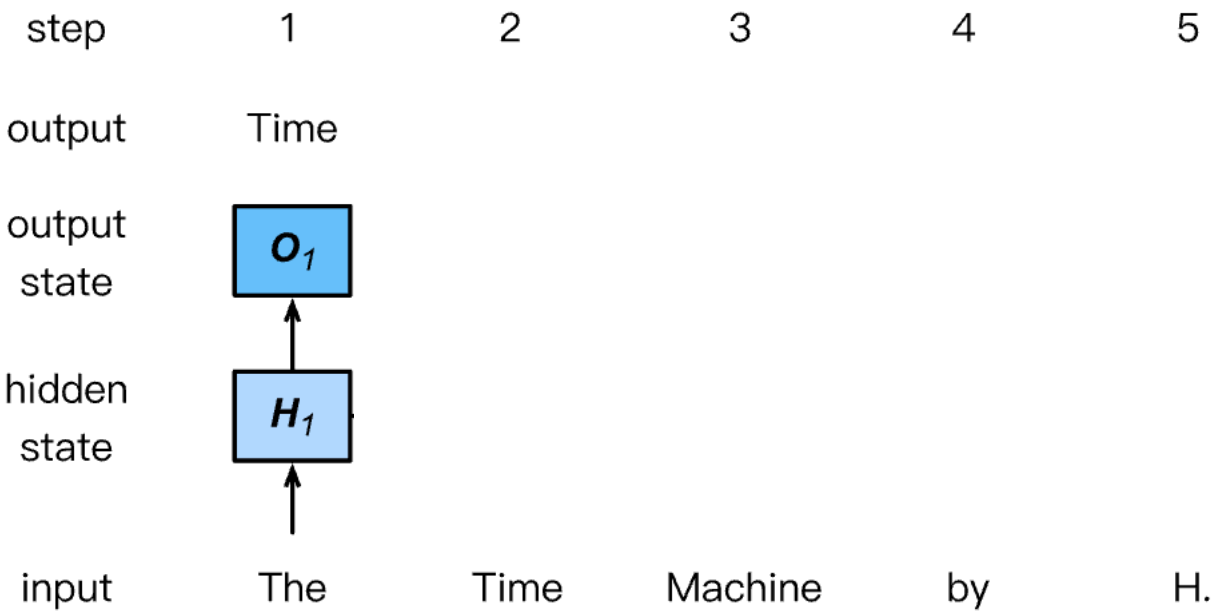
$$\mathbf{H}_t = \phi(\mathbf{W}_{hx}\mathbf{X}_{t-1} + \mathbf{b}_h)$$

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

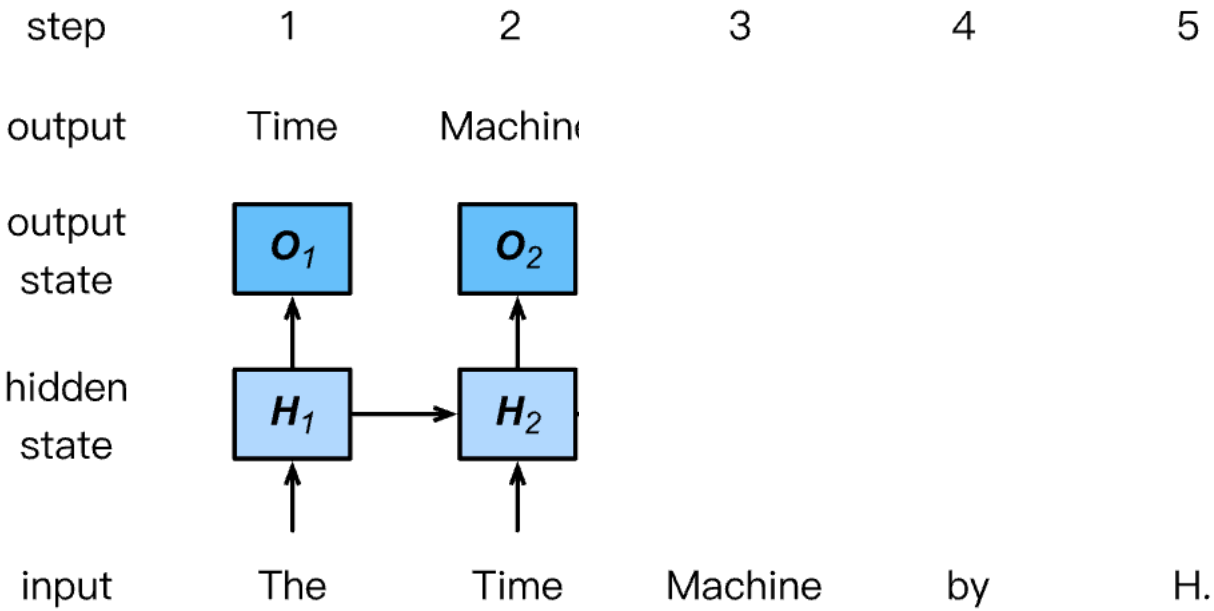
Next word prediction

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

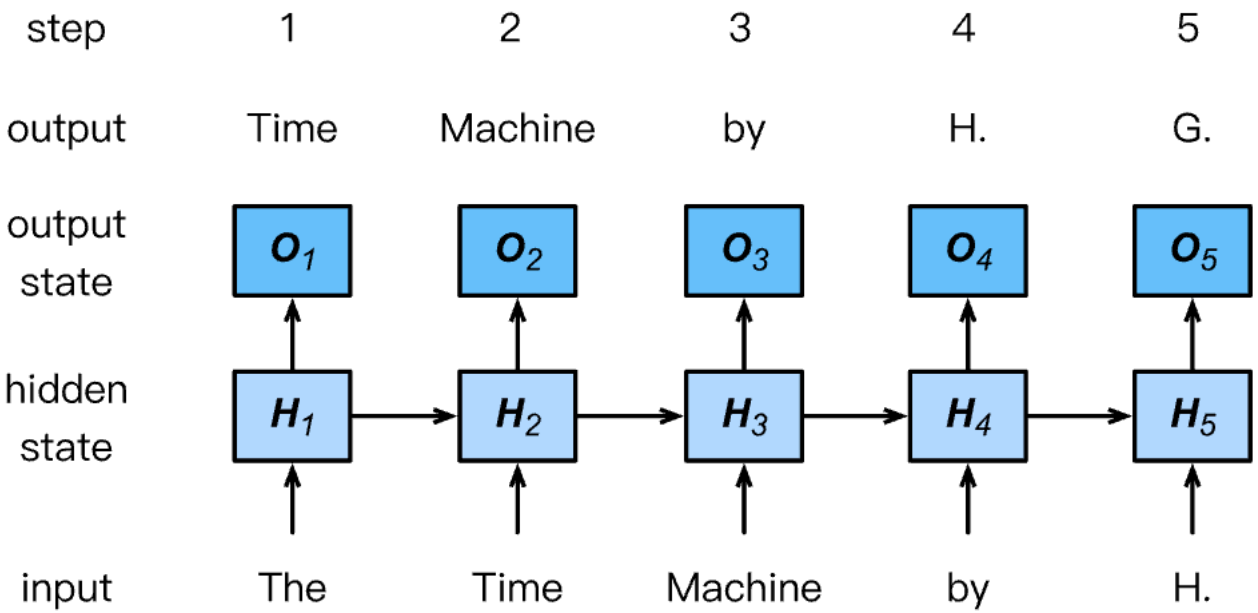
Next word prediction



Next word prediction



Next word prediction



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

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

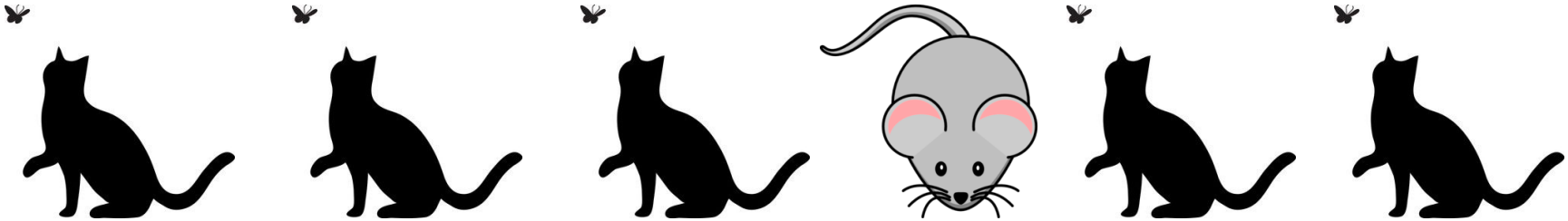
rescales to gradient of size at most θ

RNN Notebook

Gated Recurrent Unit (GRU)

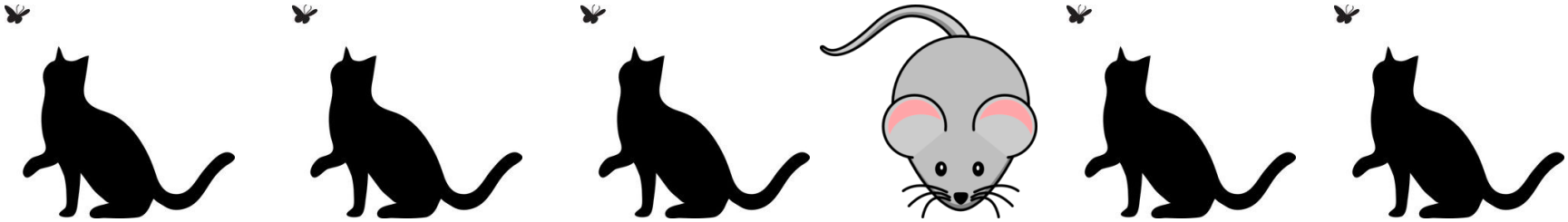
Paying attention to a sequence

- Not all observations are equally relevant



Paying attention to a sequence

- Not all observations are equally relevant

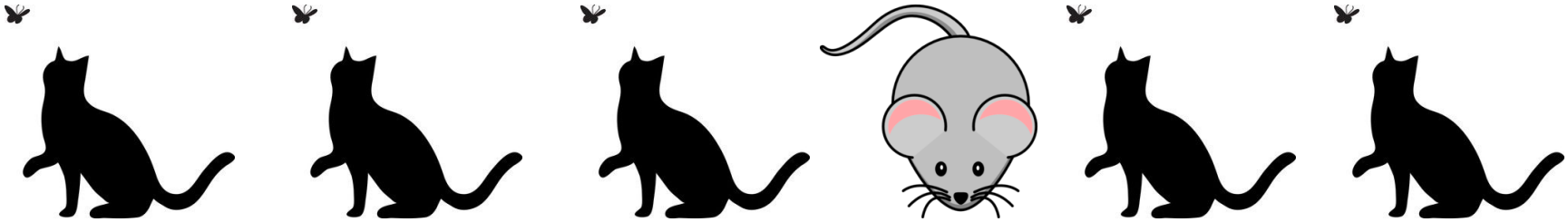


- Need mechanism to **pay attention (update gate)**

  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

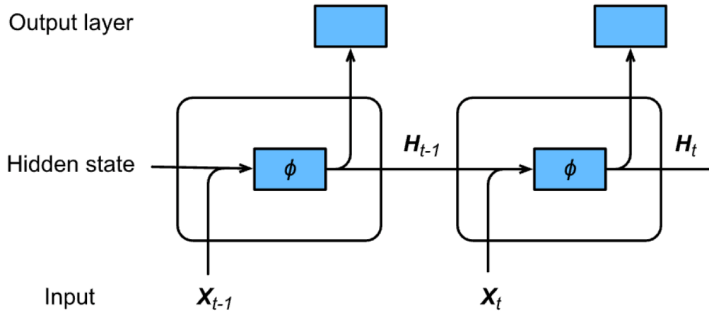


- Need mechanism to **forget (reset gate)**

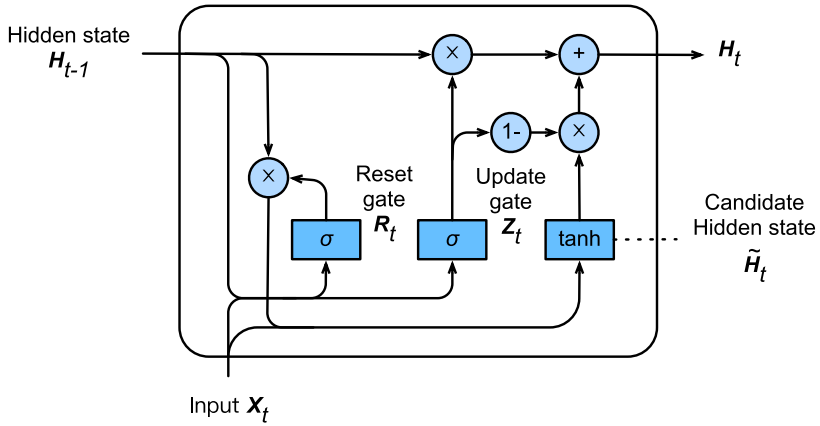
  e.g., There is a logical break between parts of a sequence. For instance there might be a transition between chapters in a book, a transition between a bear and a bull market for securities, etc.

From RNN to GRU

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

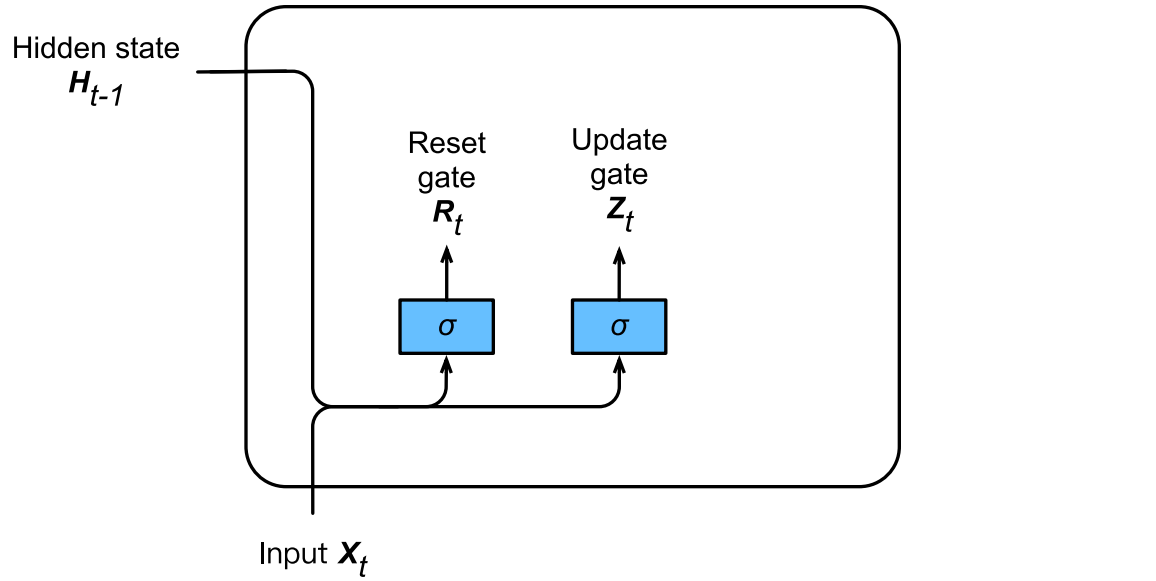


$$\boldsymbol{R}_t = \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hr} + \boldsymbol{b}_r),$$
$$\boldsymbol{Z}_t = \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hz} + \boldsymbol{b}_z)$$
$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t\boldsymbol{W}_{xh} + (\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1})\boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$
$$\boldsymbol{H}_t = \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$
$$\boldsymbol{o}_t = \boldsymbol{W}_{ho}\boldsymbol{H}_t + \boldsymbol{b}_o$$



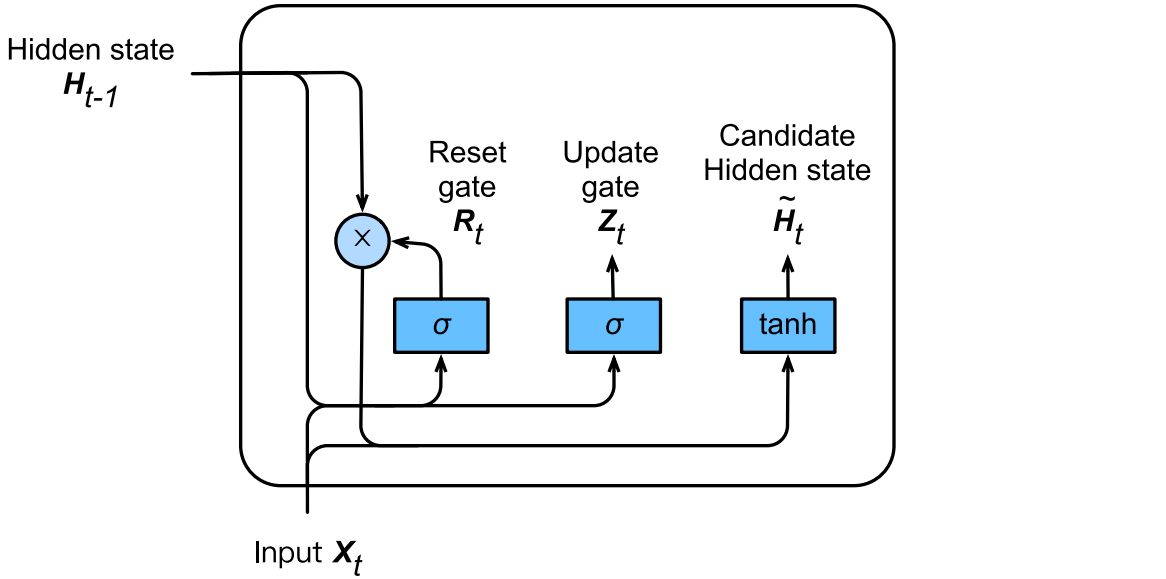
GRU - Gates

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



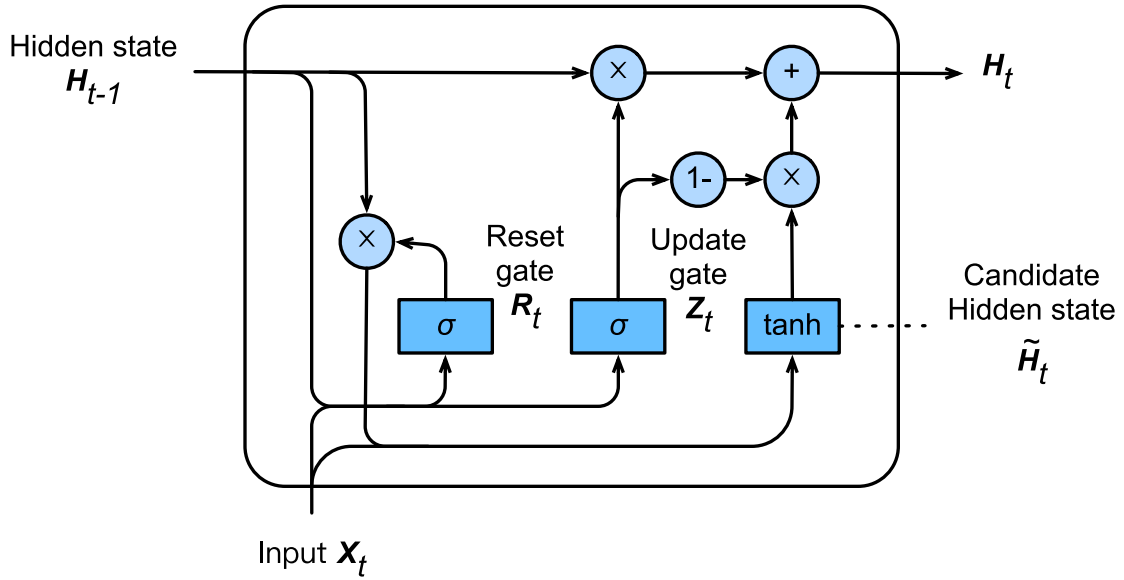
GRU - Candidate Hidden State

$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + \left(\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}\right) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$



Hidden State

$$\boldsymbol{H}_t = \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$



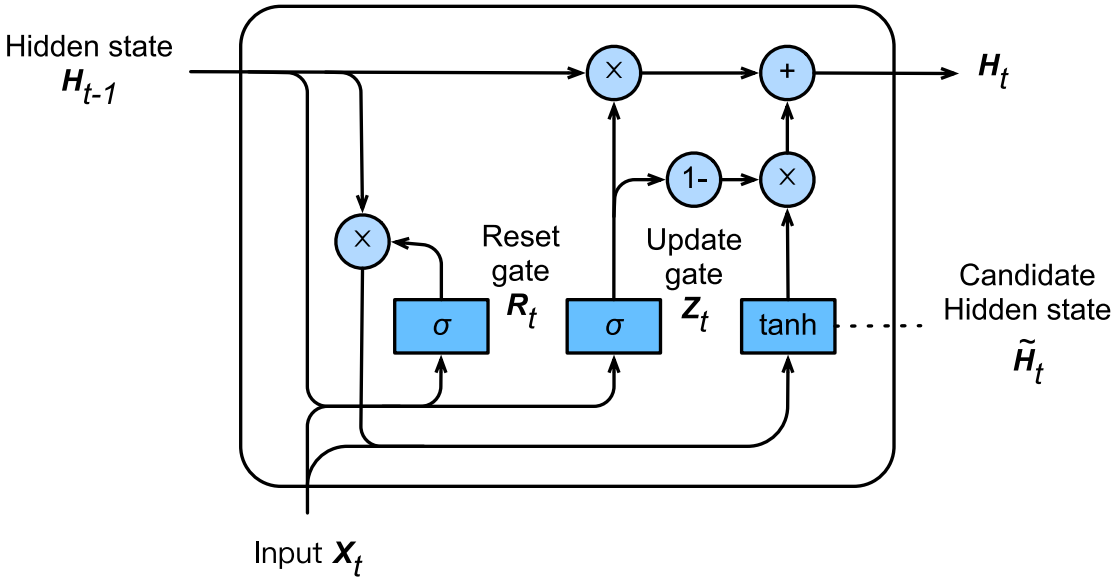
Summary

$$\boldsymbol{R}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hr} + \boldsymbol{b}_r),$$

$$\boldsymbol{Z}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hz} + \boldsymbol{b}_z)$$

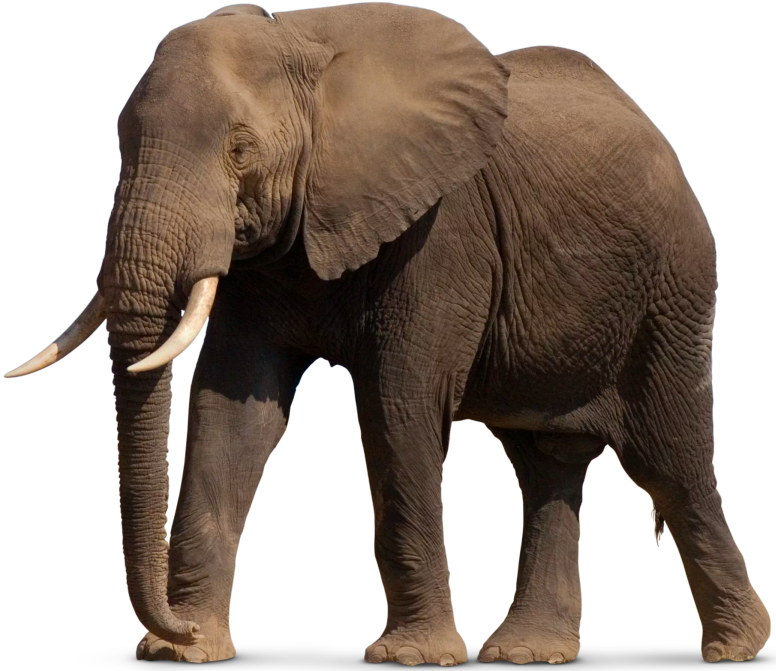
$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + (\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$

$$\boldsymbol{H}_t = \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$



GRU Notebook

Long Short Term Memory

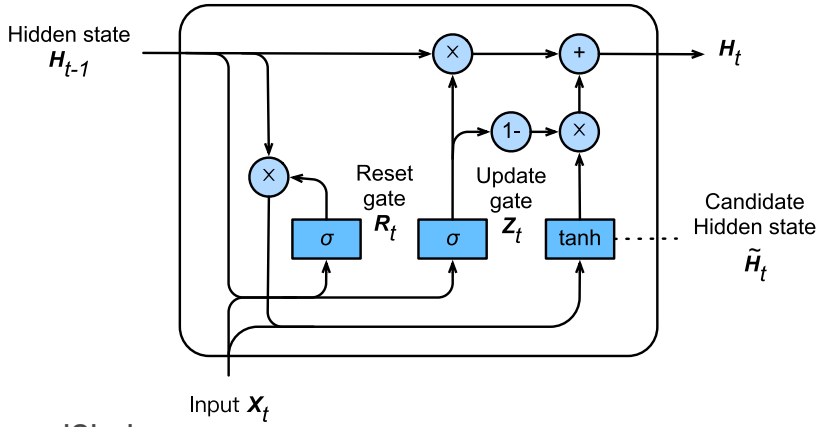


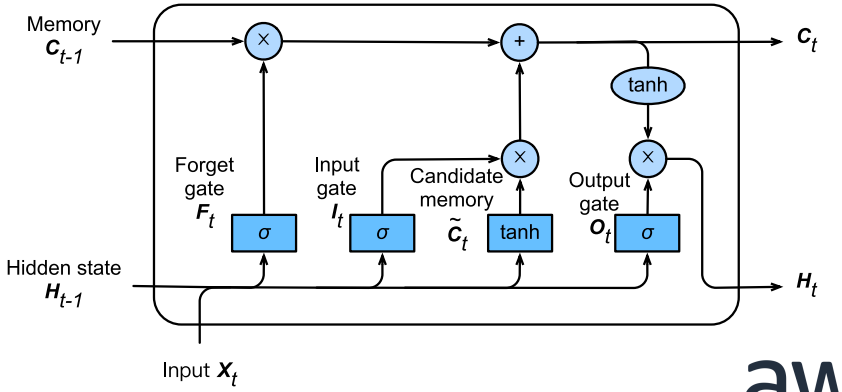
aws

GRU and LSTM

$$
\begin{aligned}
\boldsymbol{R}_t &= \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hr} + \boldsymbol{b}_r),\\
\boldsymbol{Z}_t &= \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hz} + \boldsymbol{b}_z)\\
\tilde{\boldsymbol{H}}_t &= \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + (\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)\\
\boldsymbol{H}_t &= \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{I}_t &= \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hi} + \boldsymbol{b}_i)\\
\boldsymbol{F}_t &= \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hf} + \boldsymbol{b}_f)\\
\boldsymbol{O}_t &= \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{ho} + \boldsymbol{b}_o)\\
\tilde{\boldsymbol{C}}_t &= \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hc} + \boldsymbol{b}_c)\\
\boldsymbol{C}_t &= \boldsymbol{F}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \odot \tilde{\boldsymbol{C}}_t\\
\boldsymbol{H}_t &= \boldsymbol{O}_t \odot \tanh(\boldsymbol{C}_t)
\end{aligned}
$$

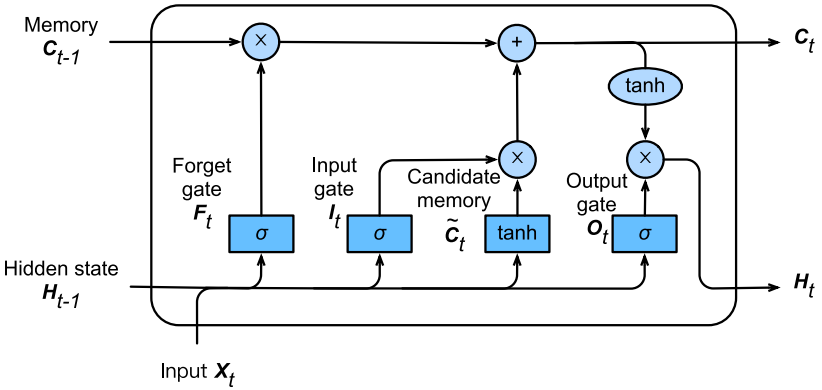




numpy.d2l.ai

aws

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

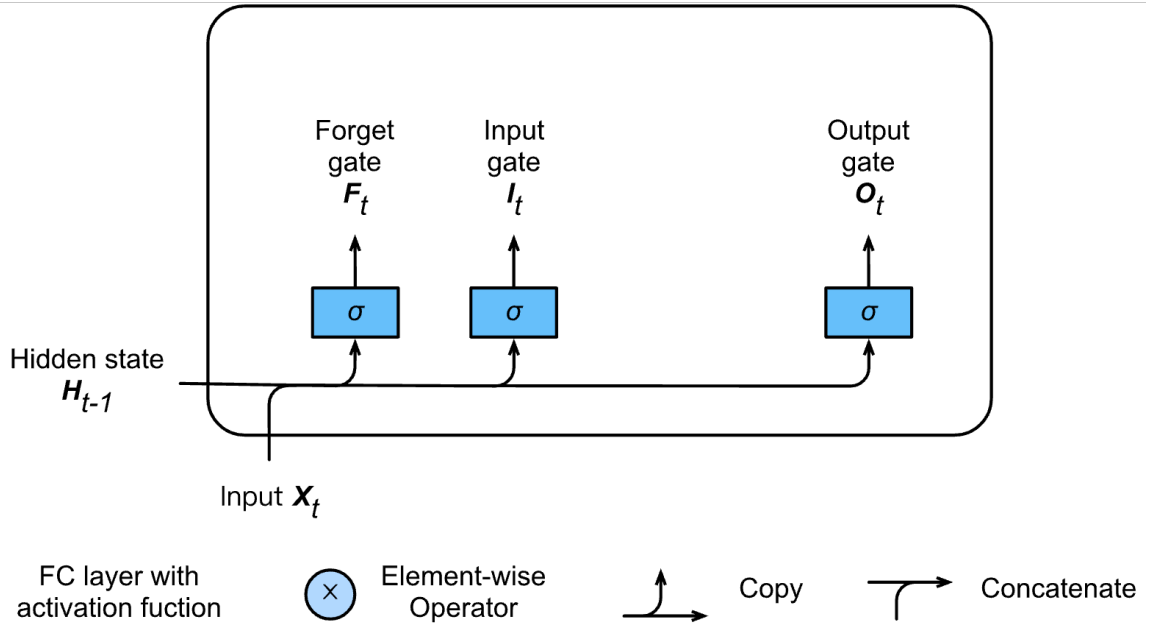
- **Hidden state** and **Memory cell**

Gates

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

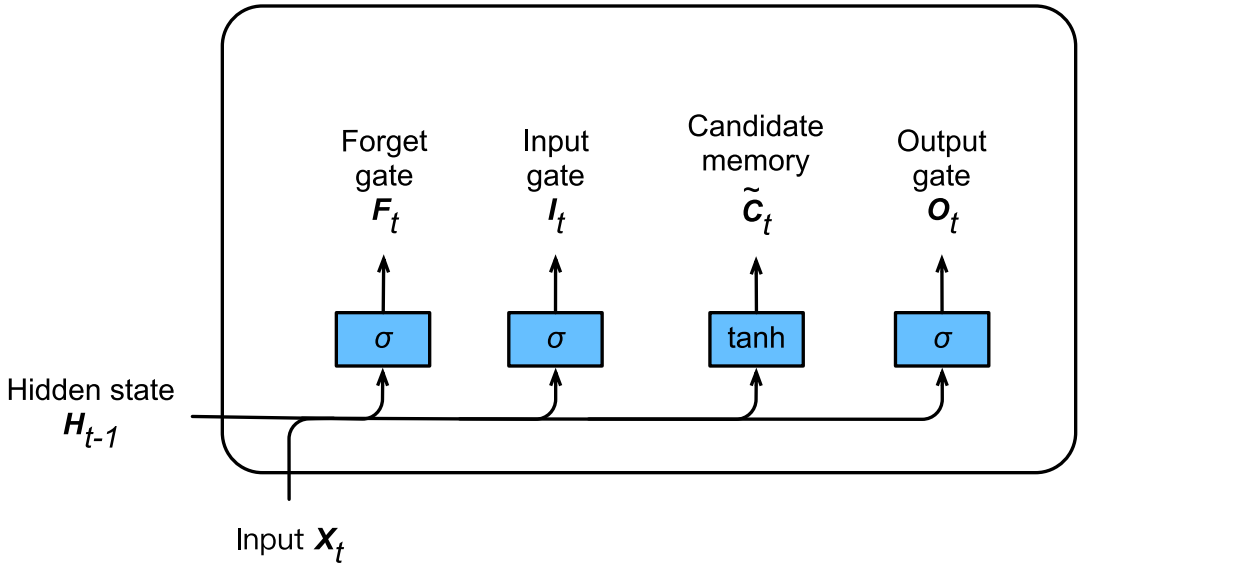
$$\boldsymbol{F}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hf} + \boldsymbol{b}_f)$$

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



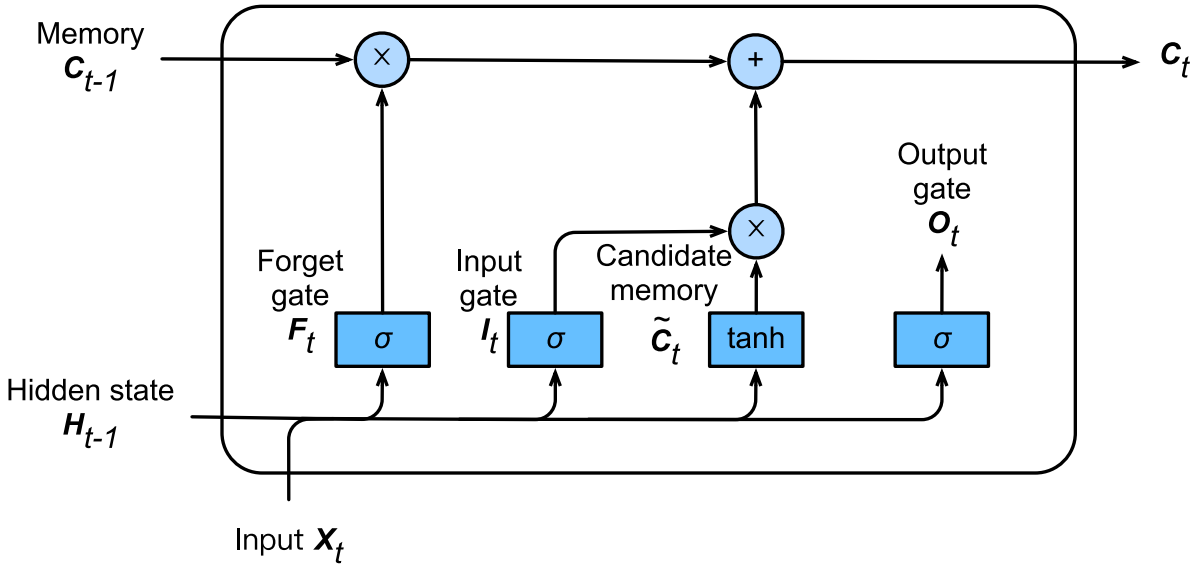
Candidate Memory Cell

$$\tilde{C}_t = \tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c)$$



Memory Cell

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



σ FC layer with activation fuction

Element-wise Operator

Copy

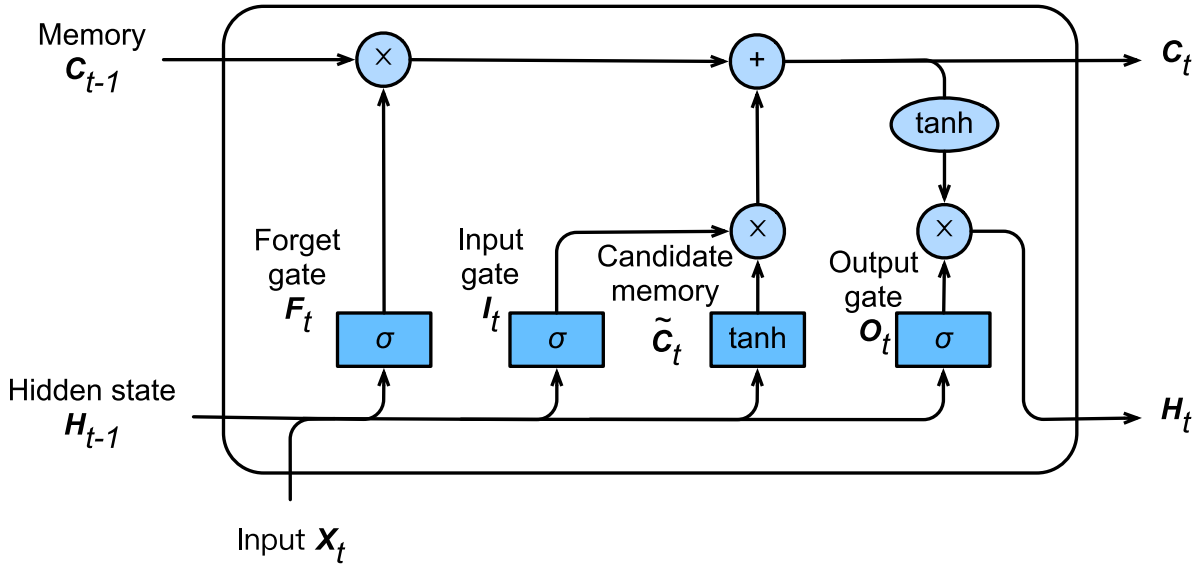
Concatenate

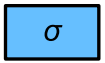
numpy.d2l.ai

aws

Hidden State / Output

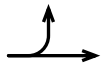
$$\boldsymbol{H}_t = \boldsymbol{O}_t \odot \tanh(\boldsymbol{C}_t)$$



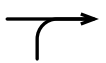


FC layer with activation fuction

Element-wise Operator

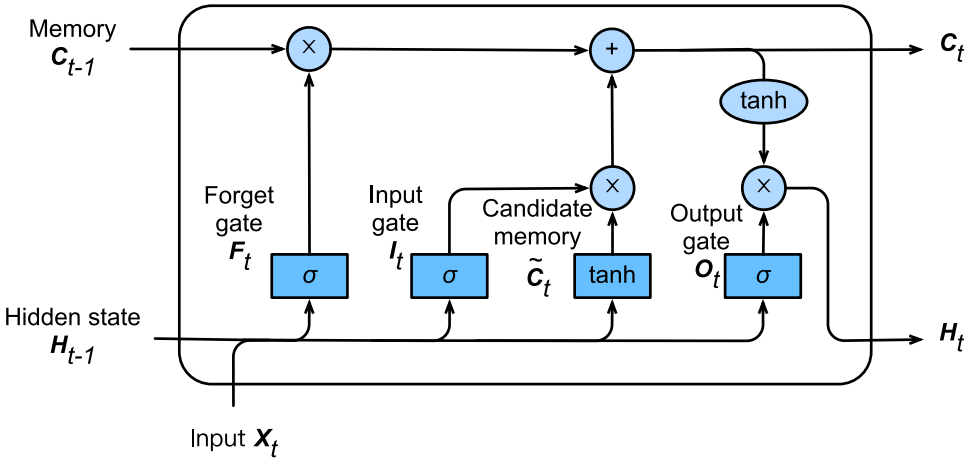


Copy



Concatenate

Hidden State / Output



$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i)$$

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f)$$

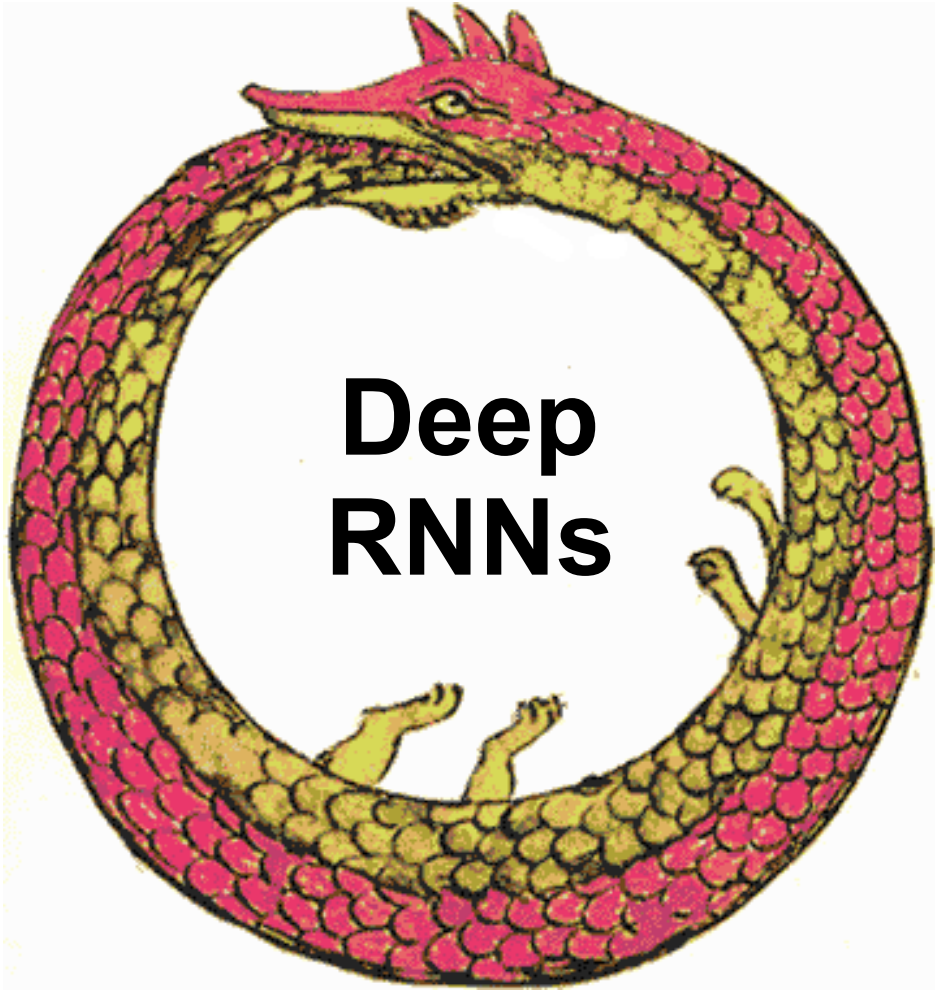
$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o)$$

$$\tilde{C}_t = \tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c)$$

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

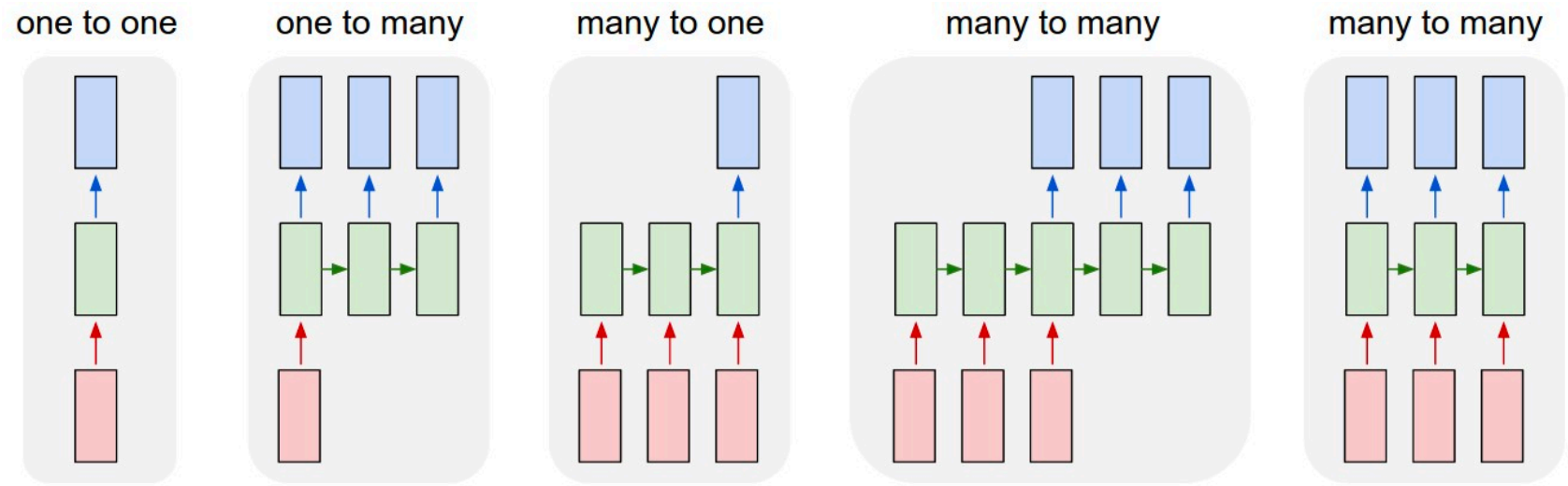
$$H_t = O_t \odot \tanh(C_t)$$

LSTM Notebook



aws

Using RNNs



Poetry Generation

Sentiment Analysis

Document Classification

Question Answering

Machine Translation

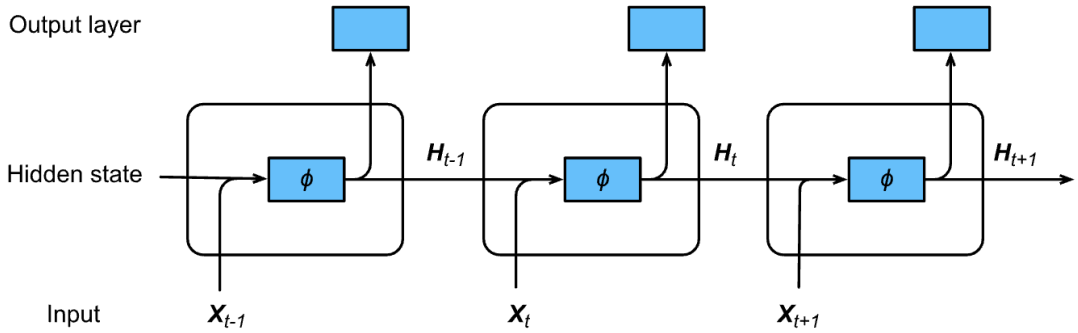
Named Entity Tagging

numpy.d2l.ai

(image courtesy of karpathy.github.io)

aws

Recall - RNNs Architecture



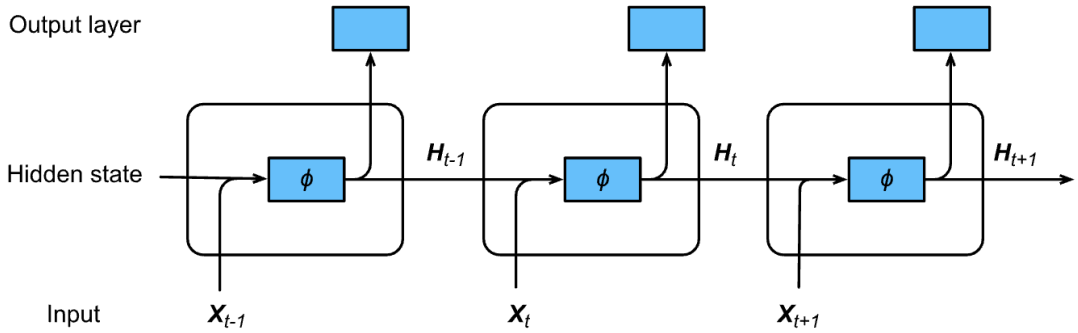
- Hidden State update

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

- Observation update

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

Recall - RNNs Architecture



- Hidden State update

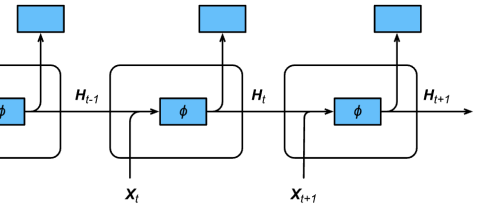
$$\mathbf{H}_t = \phi(\mathbf{W}_{hh}\mathbf{H}_{t-1} + \mathbf{W}_{hx}\mathbf{X}_{t-1} + \mathbf{b}_h)$$

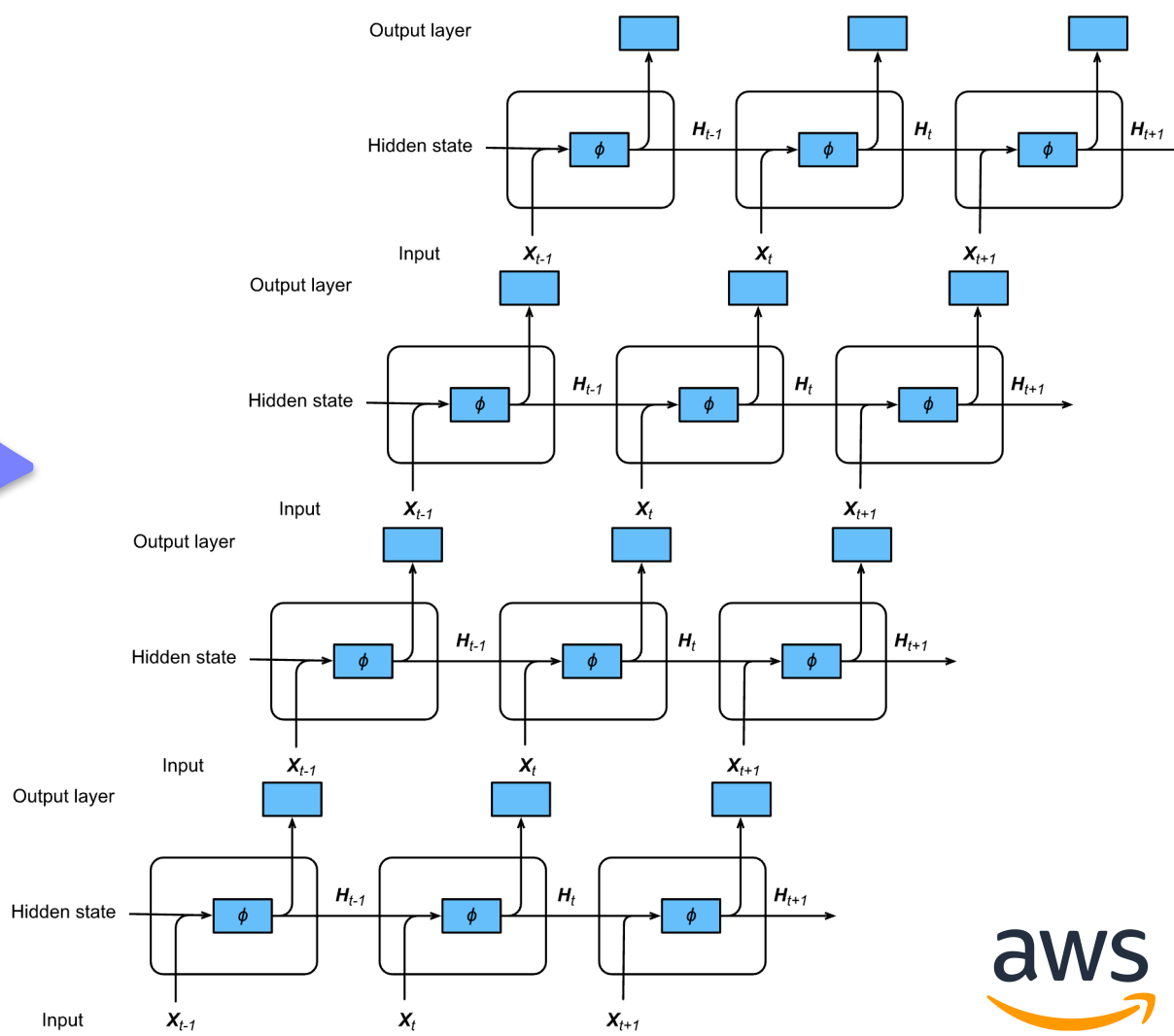
- Observation update

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

How to make more nonlinear?

We go deeper

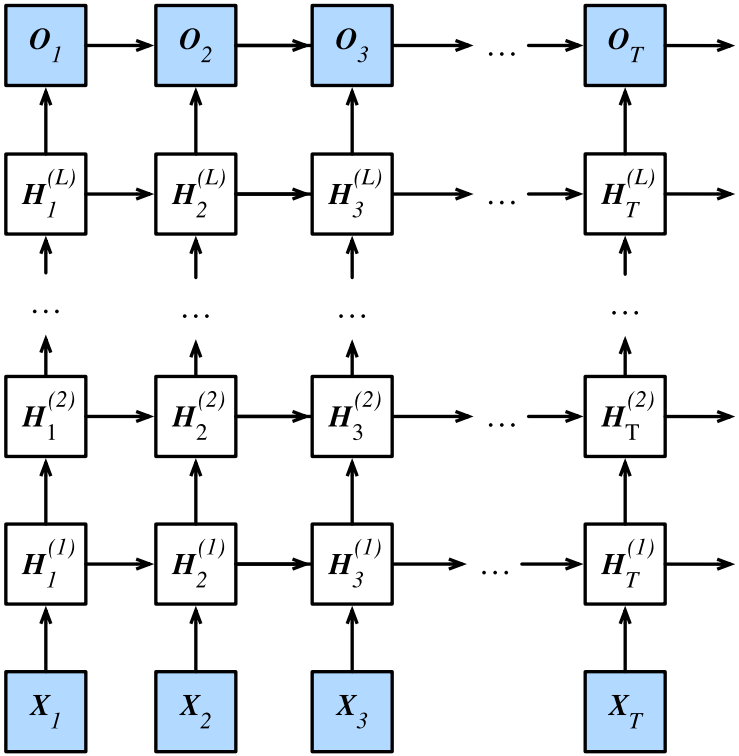




numpy.d2l.ai

aws

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

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

Bidirectional RNNs

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

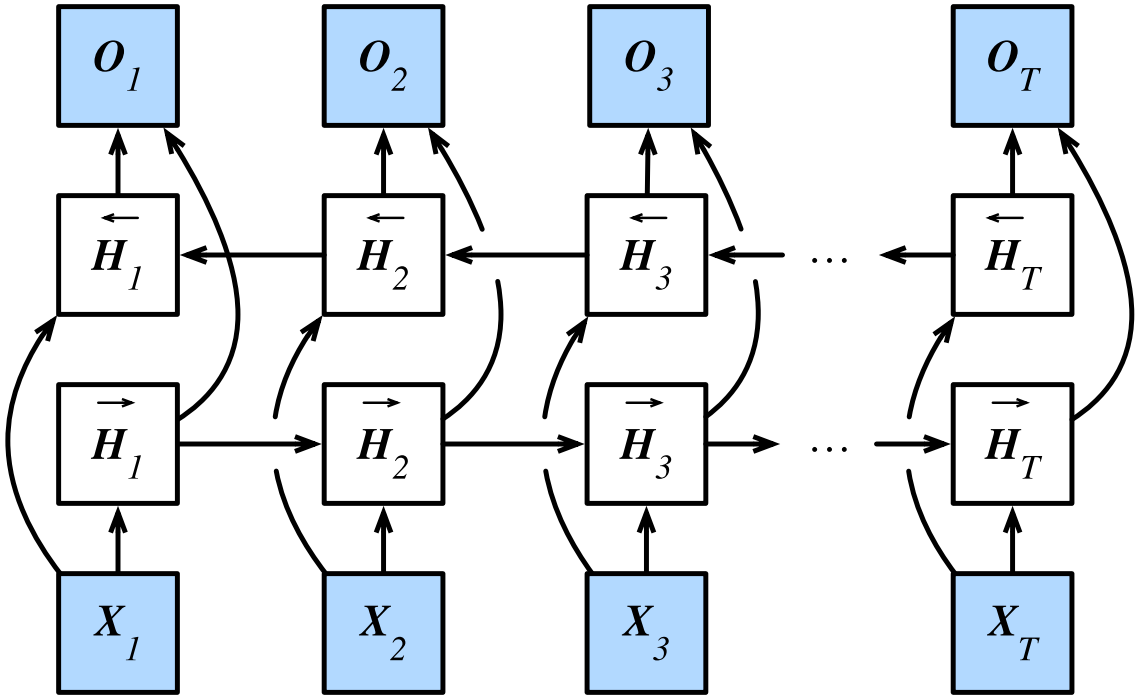
```
I am hungry.
I am not    very hungry,
I am very   very hungry, I could eat half a pig.
```

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

Summary

- Sequence models and language models
- Recurrent neural networks (RNN)
- GRU and LSTM
- Deep RNNs, Bi-RNNs

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

numpy.d2l.ai