

# Dive into Deep Learning in 1 Day

1 Basics · 2 Convnets · 3 Computation · 4 Sequences

AMLC 2019

Mu Li and Alex Smola

<http://amlc-d2l.corp.amazon.com/>

# Outline

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

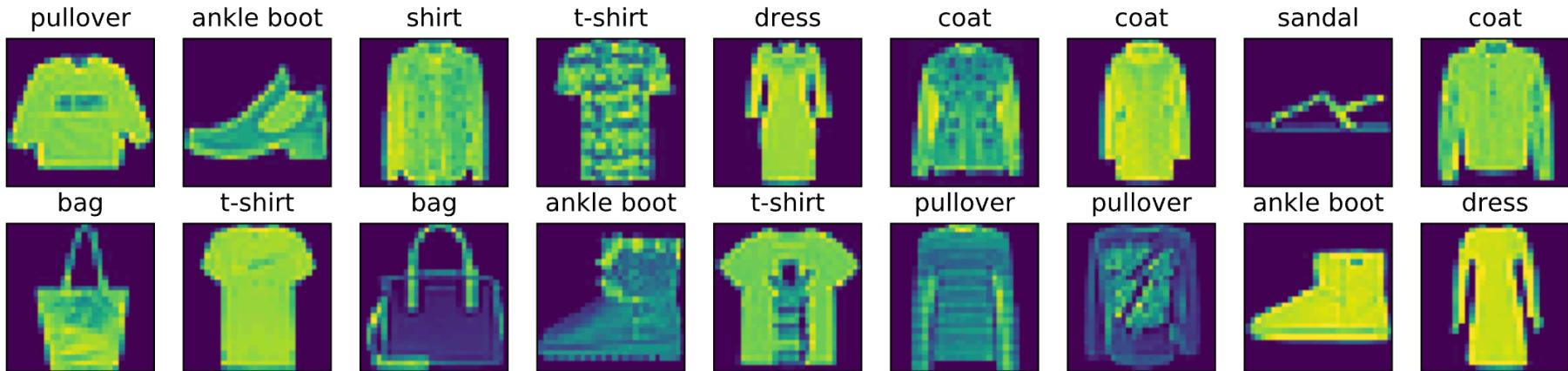
# Dependent Random Variables



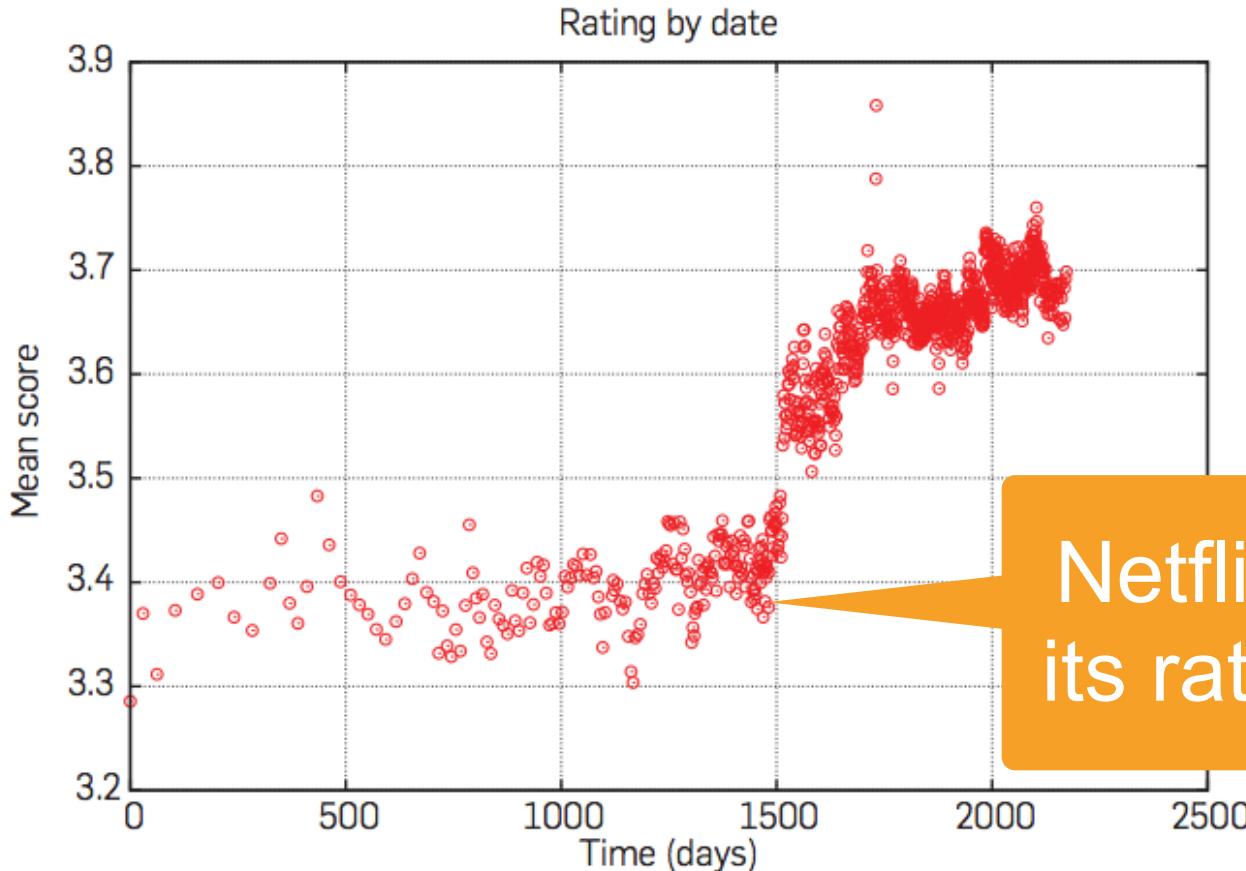
# Data

- So far, collect observation pairs  $(x_i, y_i)$  for training
- Assume examples are independent and identically distributed (IID)

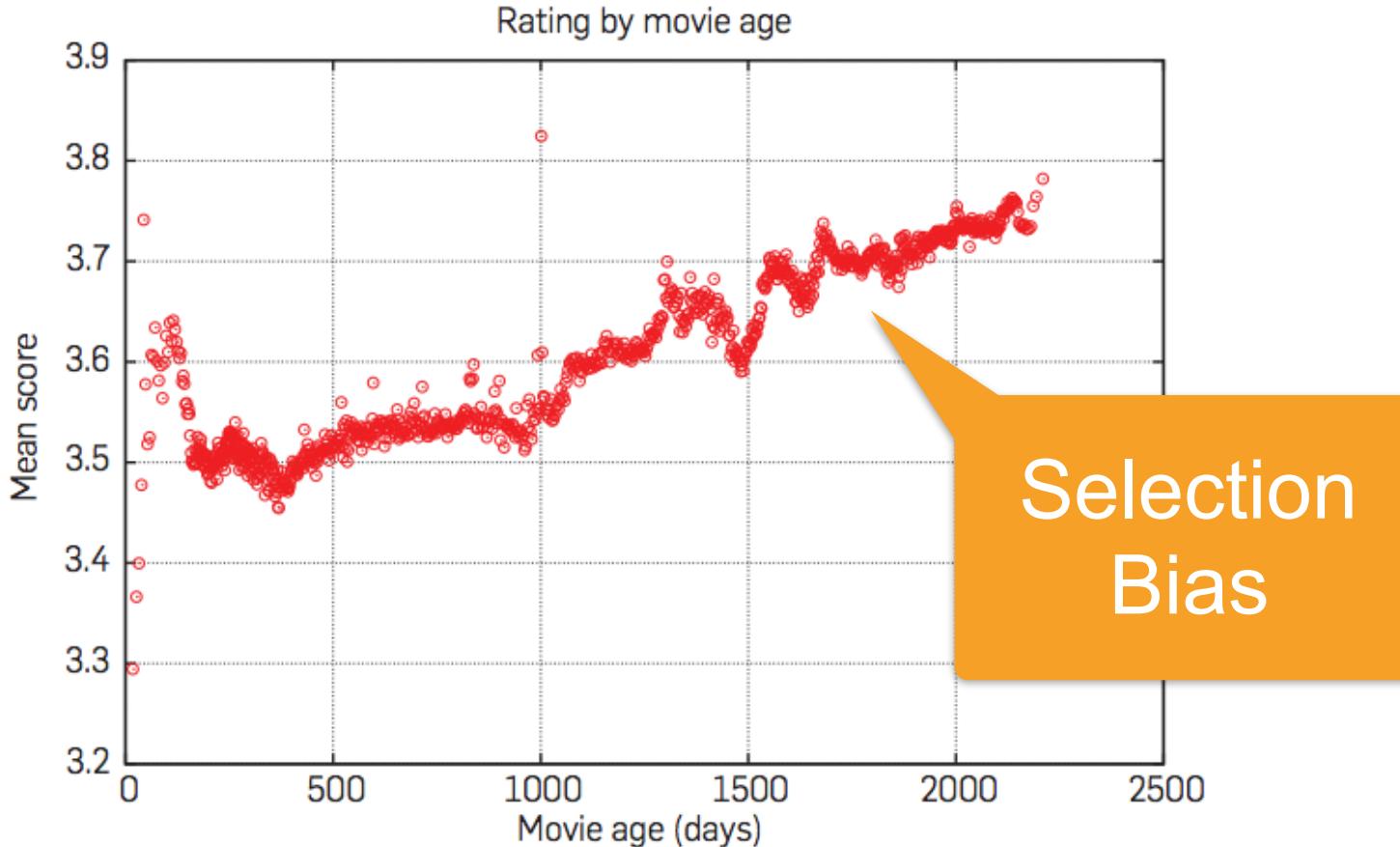
**The order of the data does not matter**



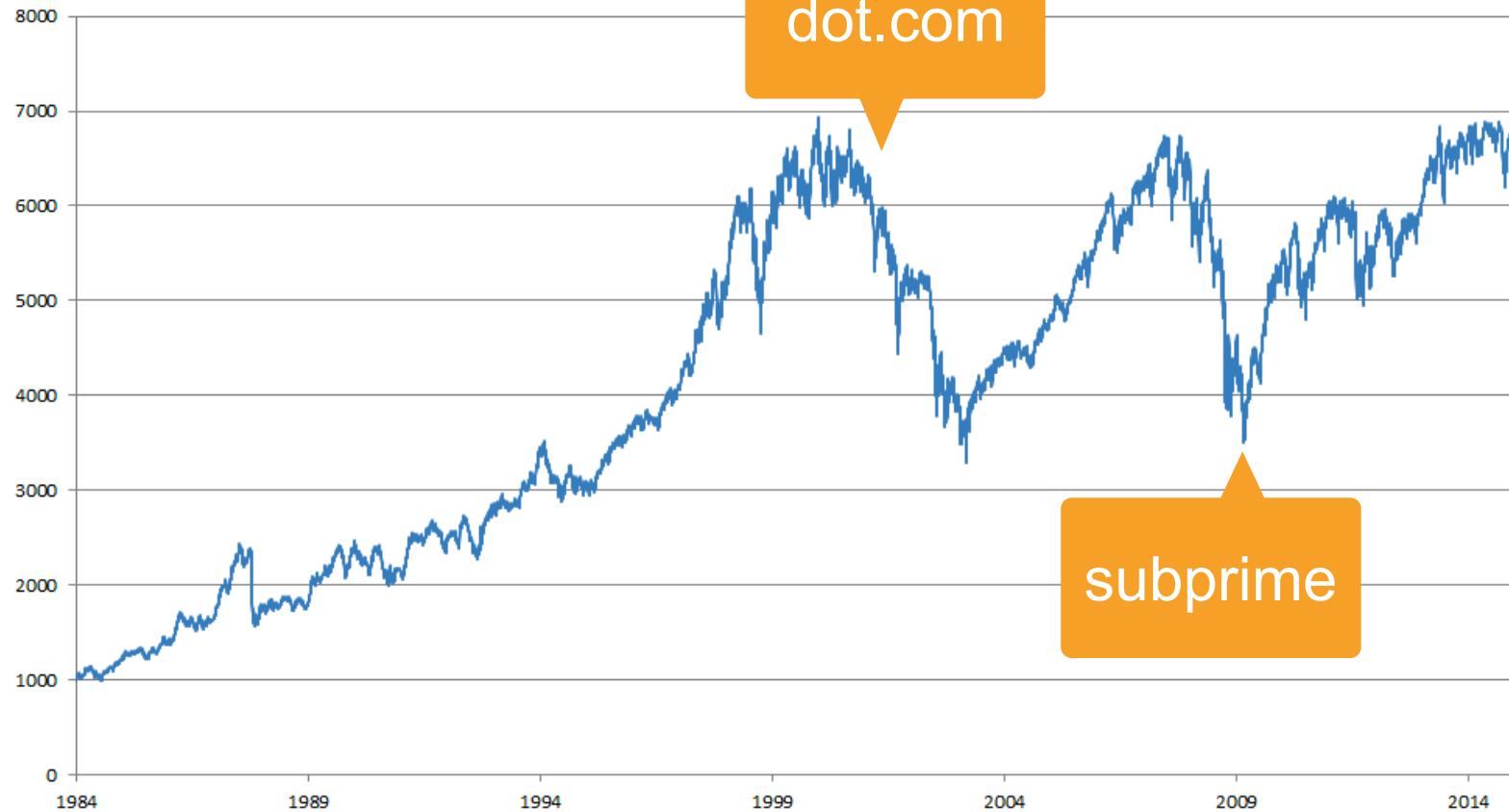
# Time matters (Koren, 2009)



# Time matters (Koren, 2009)



# FTSE 100



# TL;DR - Data usually isn't IID

inventory

Christmas

Black Friday

prime day

back to school

Q2 earnings

rate cuts

orange hair tweets

rating agencies

# Language Models

<http://amlc-d2l.corp.amazon.com/>

JWCPFWUGKA YKUJ JUOJEL  
NLOMIBIAMVHL XGUBX  
TWOQQOTDBOEFFDQAGKWBQAA  
GBGYXFUPEIXESNWRICEAQKBHHKMFU  
IVNLWIZRCZYGANONNAEQEUAKABVCENMBW  
RJXALUIMMGKJMJSFMSQIZFWHOJZSXEBCKZKHL  
JFEHPRGZWRXTPDNDWVNQRBHENVLGOWGFUNOHT  
UEZPHBZGOIBVDEVREZUAHPOLVRCZWDEWOEVNZKMH  
RIOYEGLWHIZCKYXHVDOHKDGXJDCQCGNWOCJYZMRCQMV  
MKTORSAUZAGYVPPUAONXTVBPMNDNLNSNPKMMOOPFCLAS  
WLFKNZWSKJMKUHMFIHFWVYDZPDMXVFNVNXXVAUSKSZUXDUIE  
BXXXDIIHFYKYYTVKWHJFHFWVYDZPDMXVFNVNXXVAUSKSZUXDUIE  
GNZZQOPTQOQSVSYSYDRMZIHLSSNGGYGDTSEVWBQOJXQTOXKHUMX  
MRVZPJKETDATZNPFIEBPDTQFAKEGVNRPAEHLNXMUYSDFKGOMW  
MWYDQBNRTOAVHDAJRHIZWZXZABANZLROHLOKYSMROYKKBHGOM  
ZSSCFNJKMPYZDGGWL GODFUORTKVNKHVNKFIVVXXXABLLITB  
ZOCBSKIMSCAZGHZVWHWBITZRDJOHODYBQYOVQAJLITB  
KVZSKKKBKFWXXQORGOCFHKNXFOOJLWARAPBFZELJDARZRLATLXZ  
SGHKMLRXOSGVJPISVAKAITEJUHCGXWTODJLQOCJRSZSHC  
WFTKMFESDUHCFJCPYCXFYXJPSFBAXZAKOPTPAHCUCUWXFO  
SLULIXBXFYICJPYCXFYXJPSFBAXZAKOPTPAHCUCUWXFO  
CDLMBMTOEJXZWCJXUJTPSFBDTQHOTPLD  
JIMHHQDBGAXHOSJGXBZPRTXPDPOTUEBNHPPQSNCLNP  
YIBOKGWJDDMMGKTFDLZRIPVRMPCYZZKENFYVCSTUIFT  
CIOMTJOYQZBYDQOFDSLZRIPVRMPCYZZKENFYVCSTUIFT  
PVPBFXQJCMJKPUXEOSDNTISCYZKJNIDLUKDCF  
REJGDDEOTKPTSOVXBYSVKDGGVLUKFYDYGNCRCRCSH  
CGPEEKQBTQVXBYSVKDGGVLUKFYDYGNCRCRCSH  
CGHPGFJDMOOSJQZMWPPLKJLVSBBAZAUAMUF  
JIAYPFUTNHETHJHHRQDZRUKGYTFSXSJTGEBE  
ONNLWPPWSVUGXZZOARINBTCTLUXBTQOT  
QBCTKQYKTUNVVVFVKPPIMYCMXFAPZVBR  
SSHYZLJUZKQGHJNIDL  
LBKDWCJASIDYVLFTXOLOTKJSSFLUCO  
VUJKWMPCLFYTLIWTOGAWONQOGLE  
IVDPVSRMRMOPWAYTSSAFK  
UEFQDJSZRUULNSNJHRNKFQOB  
ERGGGLVKGRMTZQCMRQN  
BURDKWLMXEXXYJCPCWGO  
WOINRIJVHGBOVUOES  
PRJPFHUYLYNHHAQBJ  
TUTIOZTGQARP  
YFJE  
ISOZE  
ANKP  
WKN



# Language Modeling

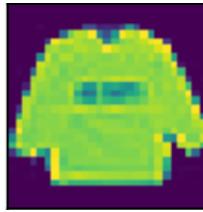
- Goal: predict the probability of a sentence, e.g.

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

- A fundamental task in Natural Language Processing
  - 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

# 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 called here) was expounding a recondite matter to us. He twinkled, and his usually pale face was flushed; the fire burned brightly, and the soft radiance of the lights in the lilies of silver caught and passed in our glasses. Our chairs, being caressed us rather than submitted to be, were in a luxurious after-dinner atmosphere when free of the trammels of precision. And



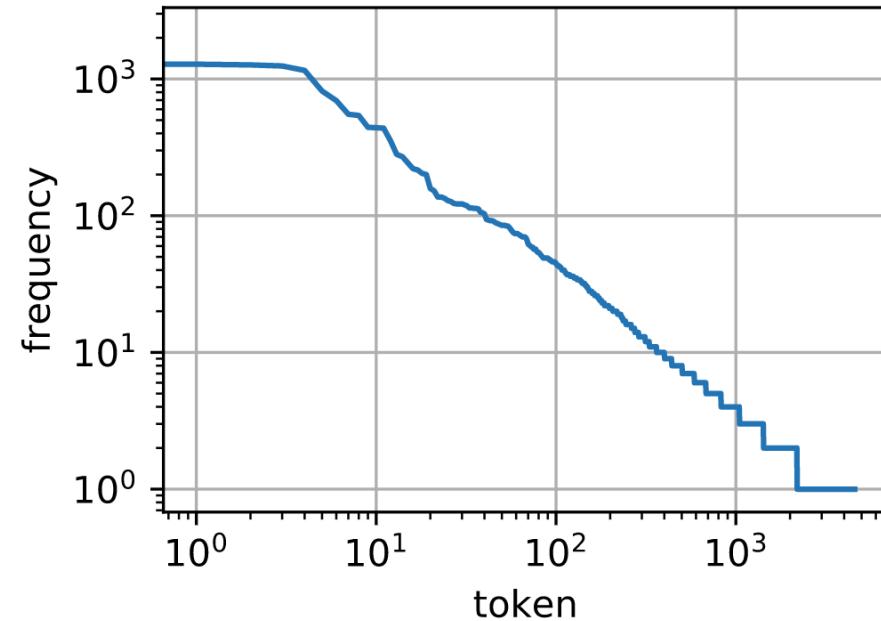
(batch size, sentence length)

# 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

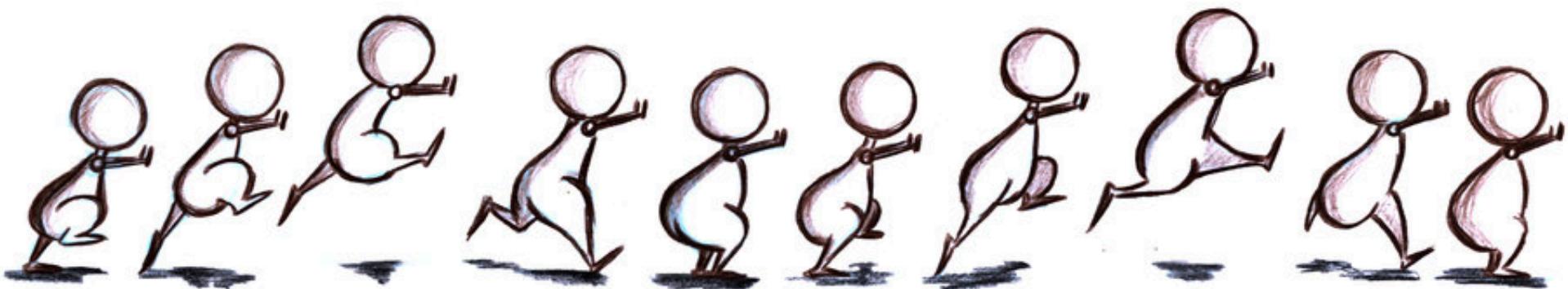
The Time Machine by H. G. Wells

# Text Preprocessing Notebook

<http://amlc-d2l.corp.amazon.com/>



# Sequence Models



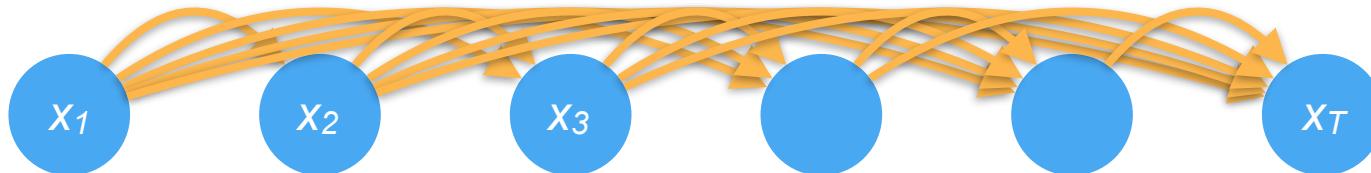
# Sequence Model

- Dependent random variables

$$(x_1, \dots, x_T) \sim p(\mathbf{x})$$

- Conditional probability expansion

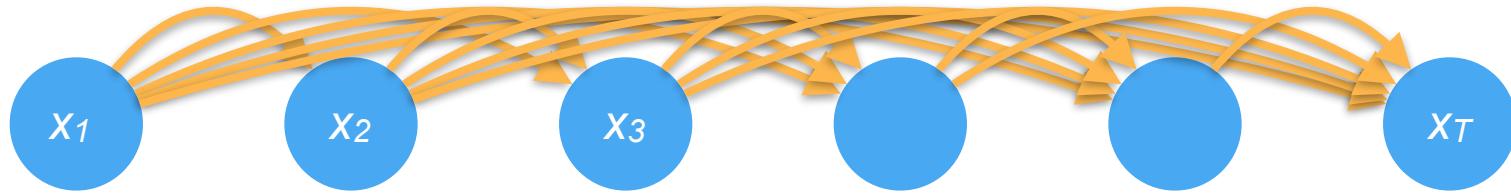
$$p(\mathbf{x}) = p(x_1) \cdot p(x_2 | x_1) \cdot p(x_3 | x_1, x_2) \cdot \dots \cdot p(x_T | x_1, \dots, x_{T-1})$$



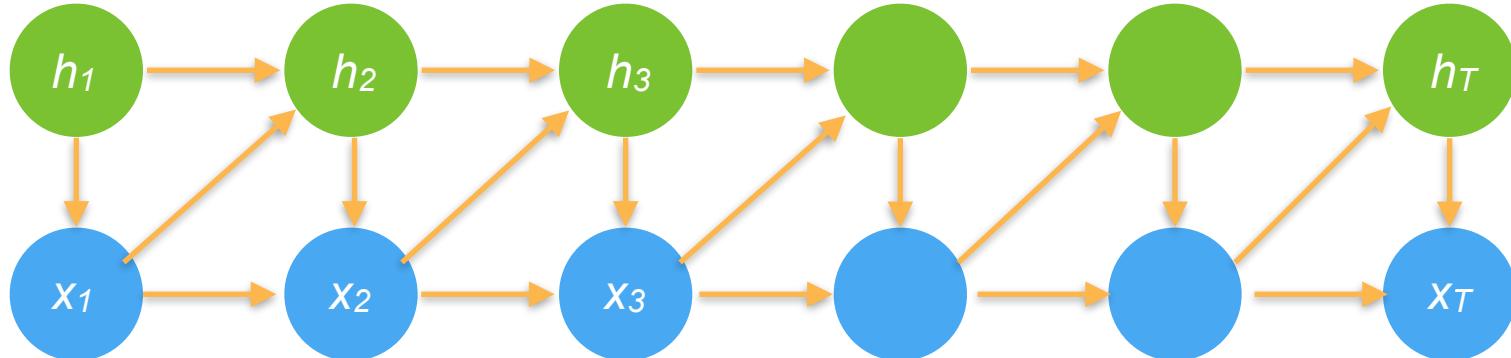
Exponential complexity!

# Latent Variable Model

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



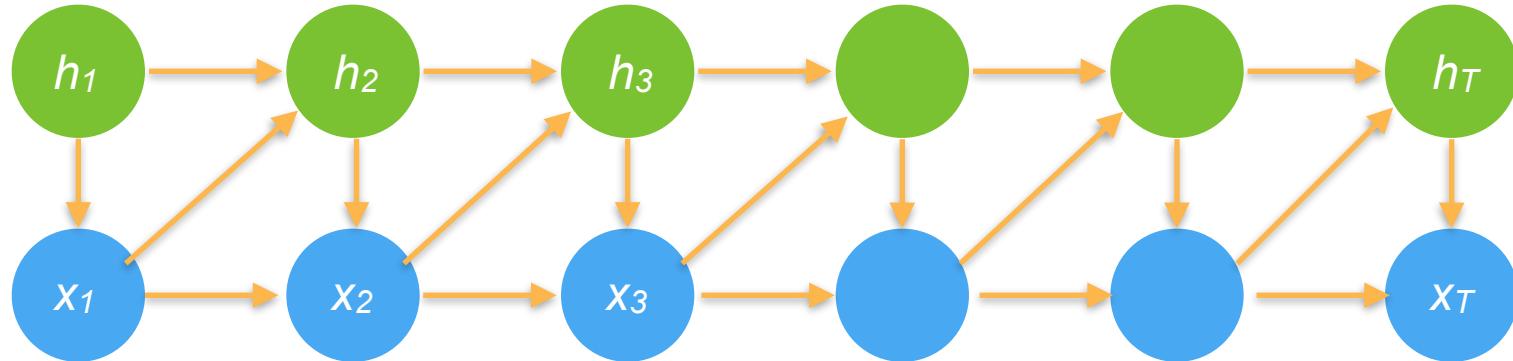
$$p(h_t | h_{t-1}, x_{t-1}) \text{ and } p(x_t | h_t, x_{t-1})$$

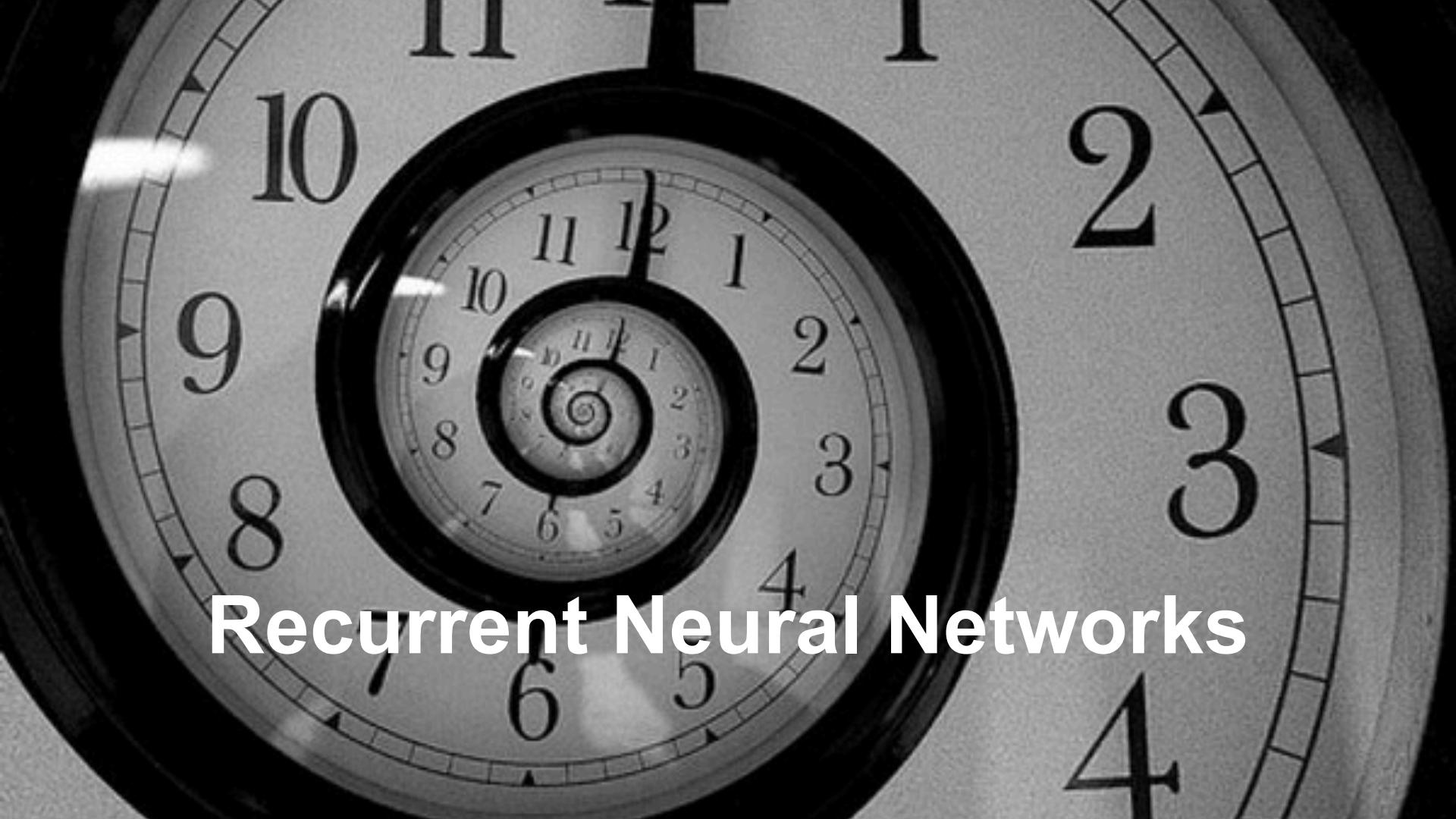


# Latent Variable Model

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

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



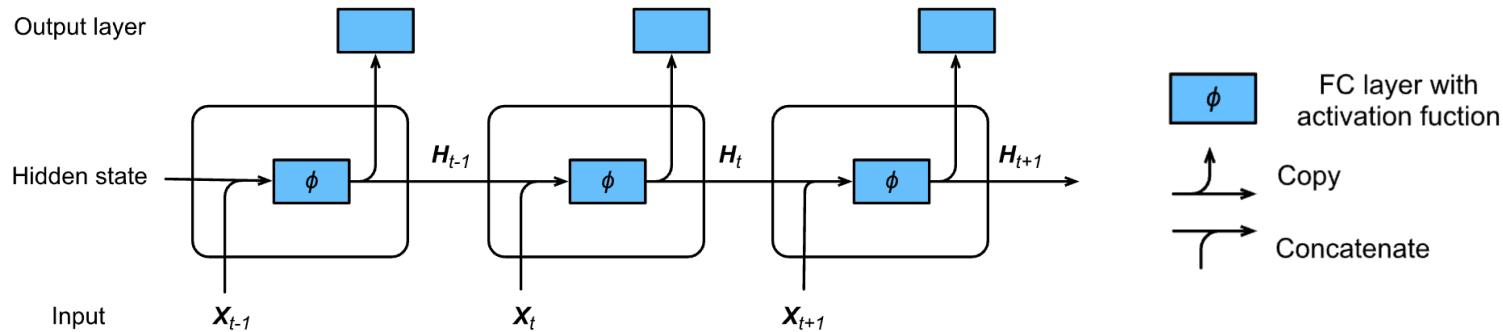


# Recurrent Neural Networks

# Steps in a Language Model

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

# Recurrent Networks 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

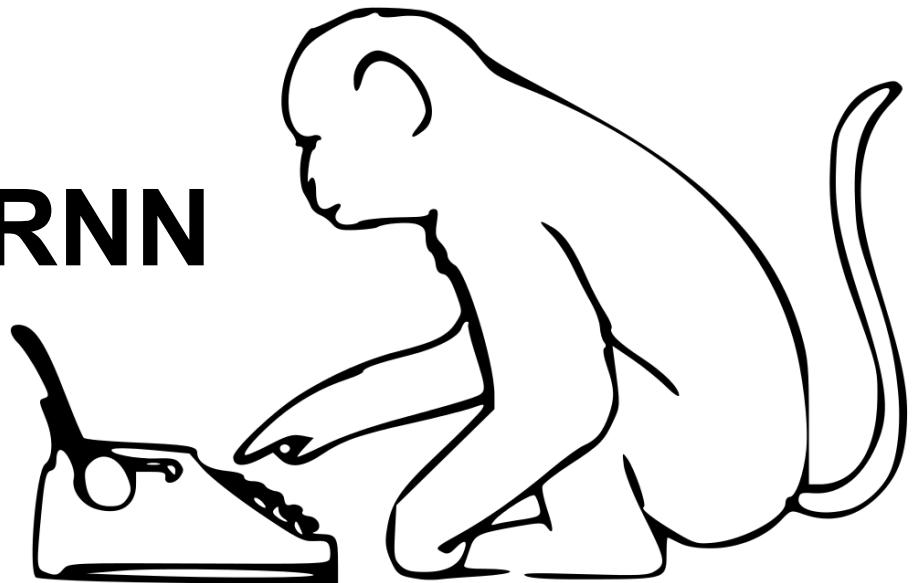
$$\mathbf{o}_t = \mathbf{W}_{ho}\mathbf{H}_t + \mathbf{b}_o$$

- Compare to MLP

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

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

# Implementing an RNN Language Model



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

# Perplexity

- Typically measure accuracy with log-likelihood
  - This makes outputs of different length incomparable (e.g. bad model on short output has higher likelihood than excellent model on very long output)
  - Normalize log-likelihood to sequence length
    - $\sum_{t=1}^T \log p(y_t | \text{model})$  vs.  $\pi := -\frac{1}{T} \sum_{t=1}^T \log p(y_t | \text{model})$

- Perplexity is exponentiated version  $\exp(\pi)$   
(effectively number of possible choices on average)

# Gradients

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

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

rescales to gradient of size at most  $\theta$

# RNN Notebook

<http://amlc-d2l.corp.amazon.com/>

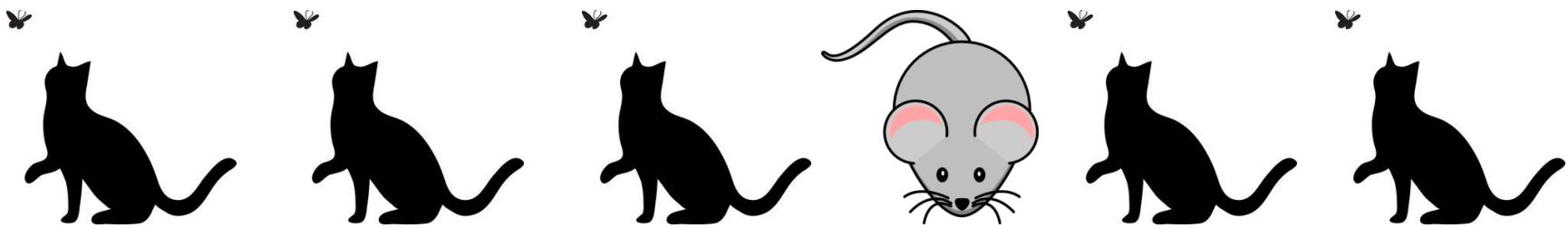


# Gated Recurrent Unit (GRU)



# Paying attention to a sequence

- Not all observations are equally relevant

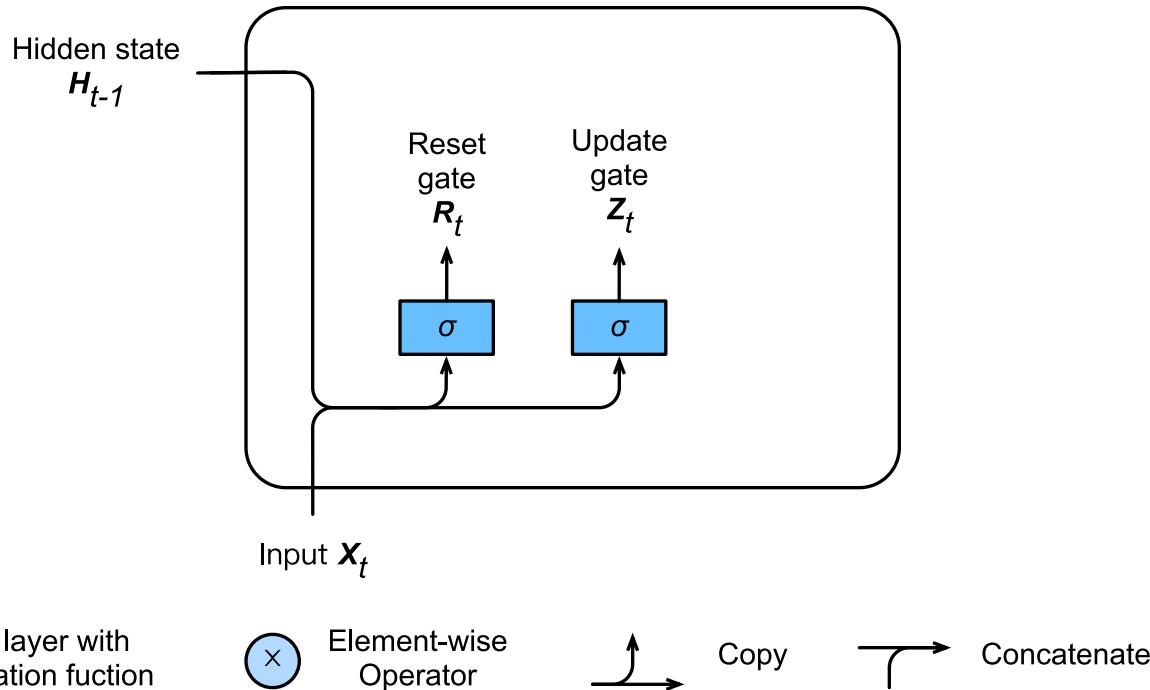


- Only remember the relevant ones
  - Need mechanism to **pay attention (update gate)**
  - Need mechanism to **forget (reset gate)**

# Gating

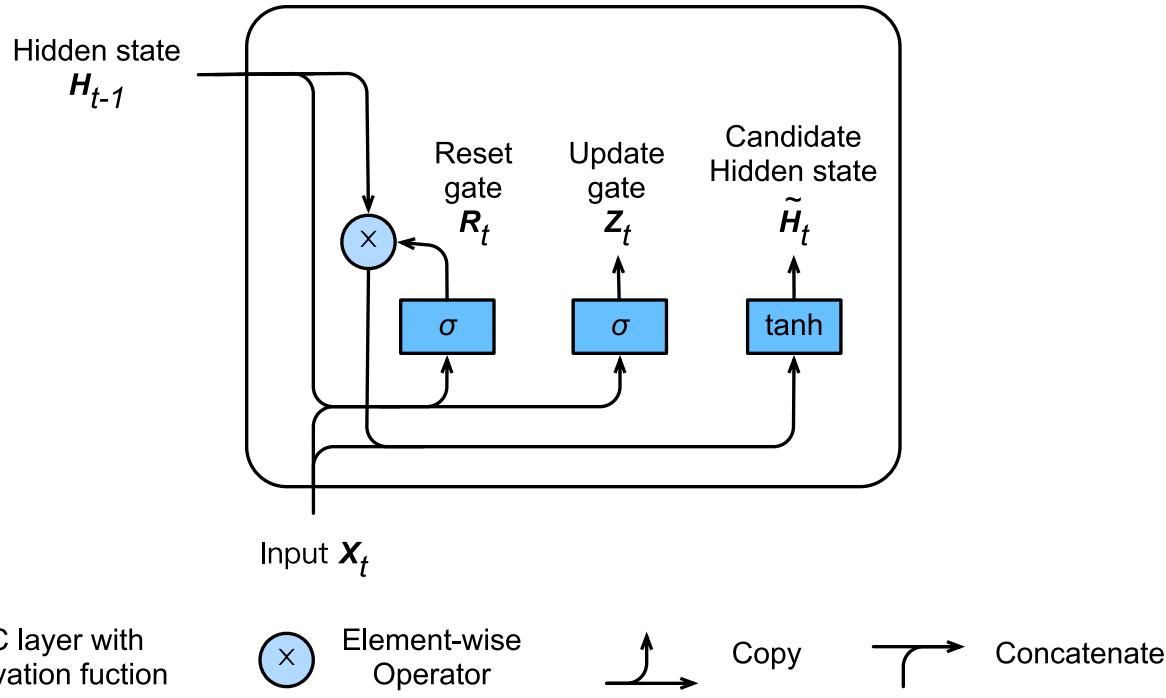
$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r),$$

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



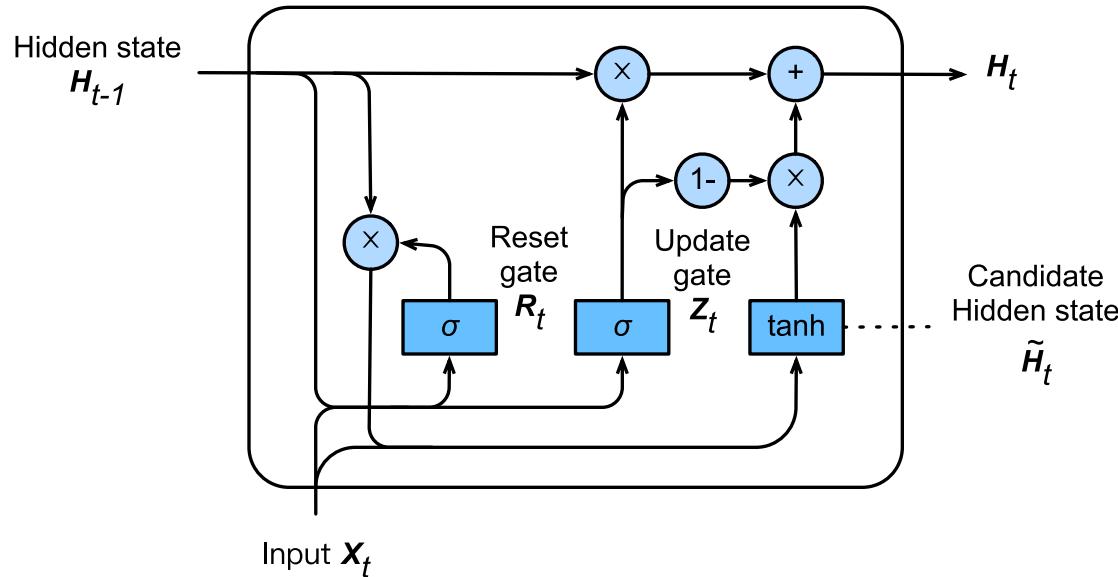
# Candidate Hidden State

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



# Hidden State

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



FC layer with  
activation function



Element-wise  
Operator



Copy



Concatenate

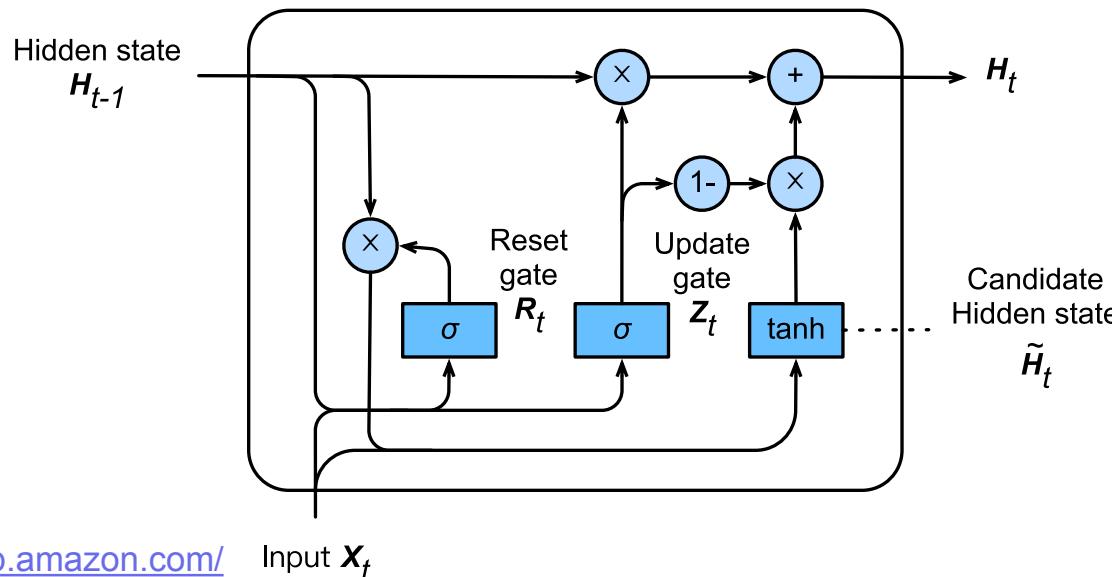
# Summary

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

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

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

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

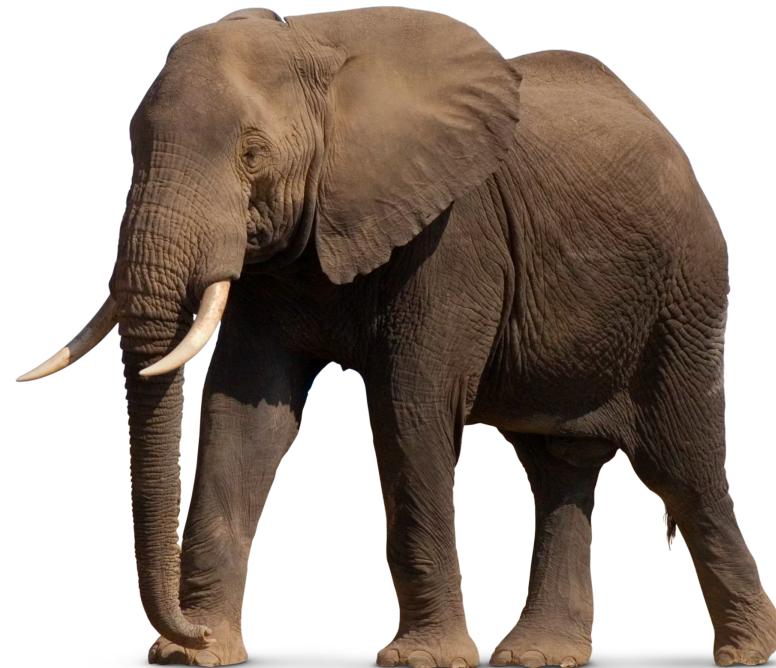


# GRU Notebook

<http://amlc-d2l.corp.amazon.com/>



# Long Short Term Memory



# Long Short Term Memory

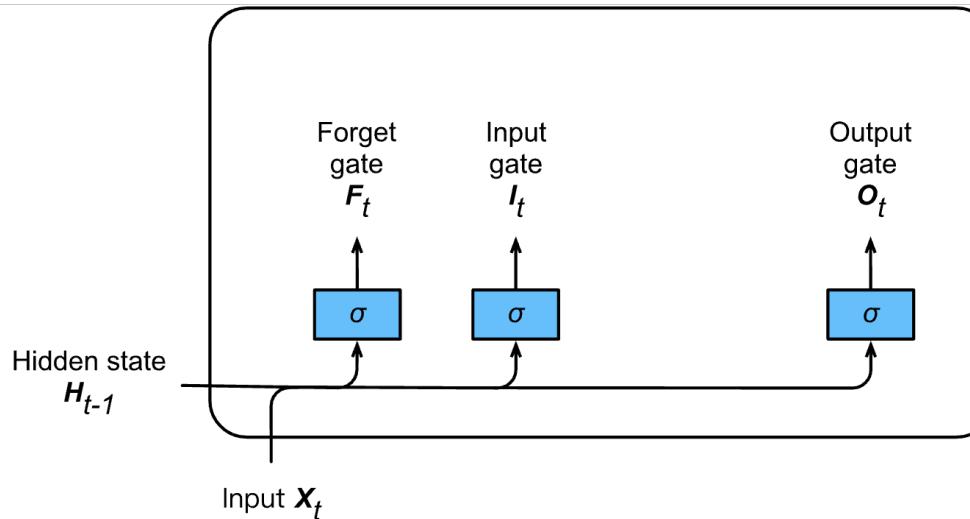
- **Forget gate**  
Shrink values towards zero
- **Input gate**  
Decide whether we should ignore the input data
- **Output gate**  
Decide whether the hidden state is used for the output generated by the LSTM
- **Hidden state and Memory cell**

# Gates

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

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

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



FC layer with  
activation function



Element-wise  
Operator



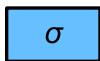
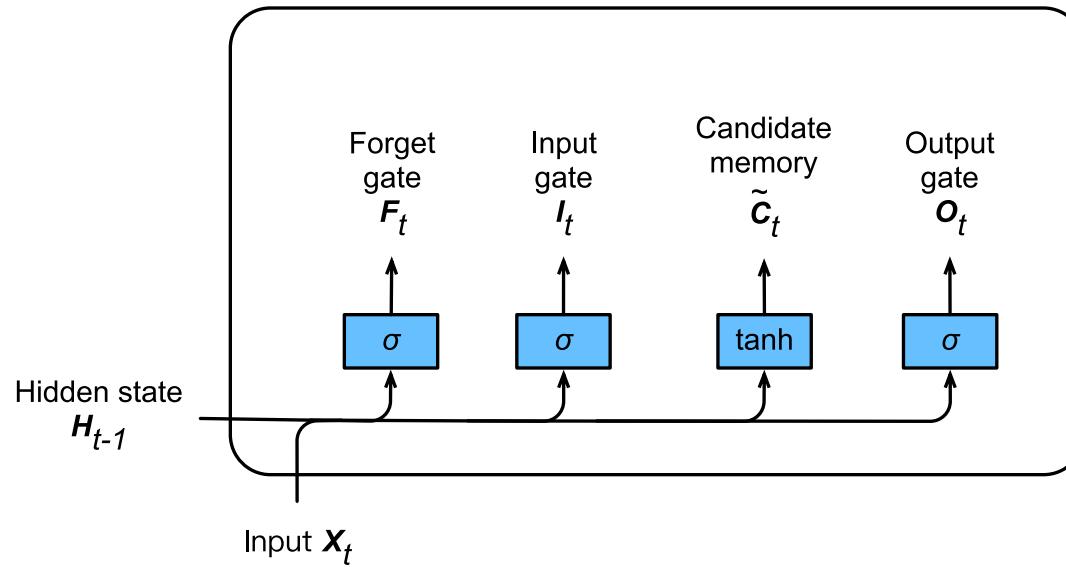
Copy



Concatenate

# Candidate Memory Cell

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



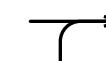
FC layer with  
activation function



Element-wise  
Operator



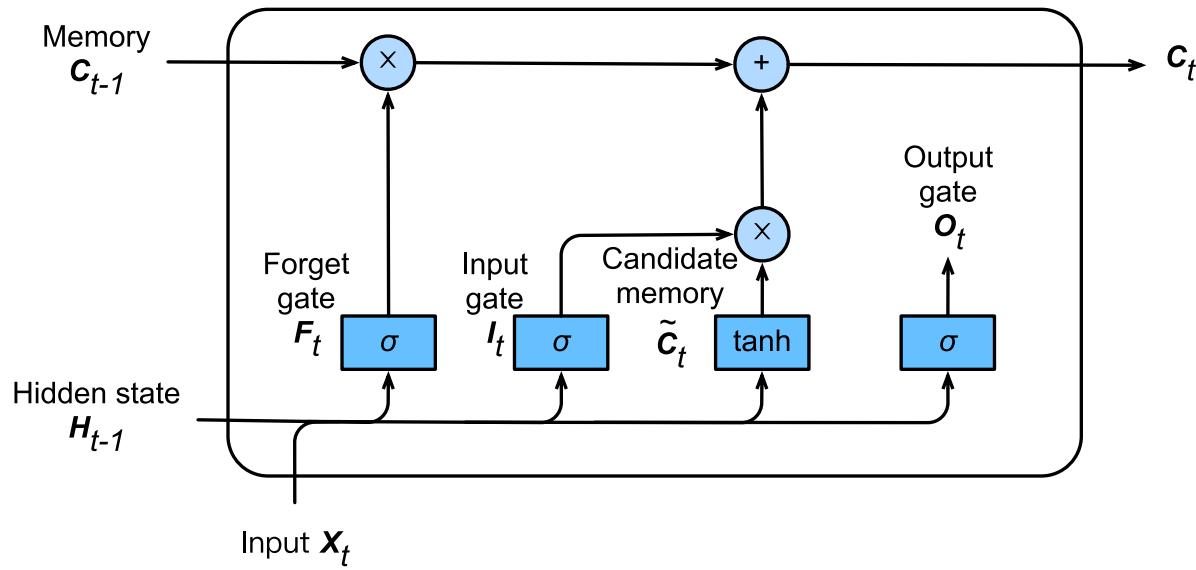
Copy



Concatenate

# Memory Cell

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



FC layer with  
activation function



Element-wise  
Operator



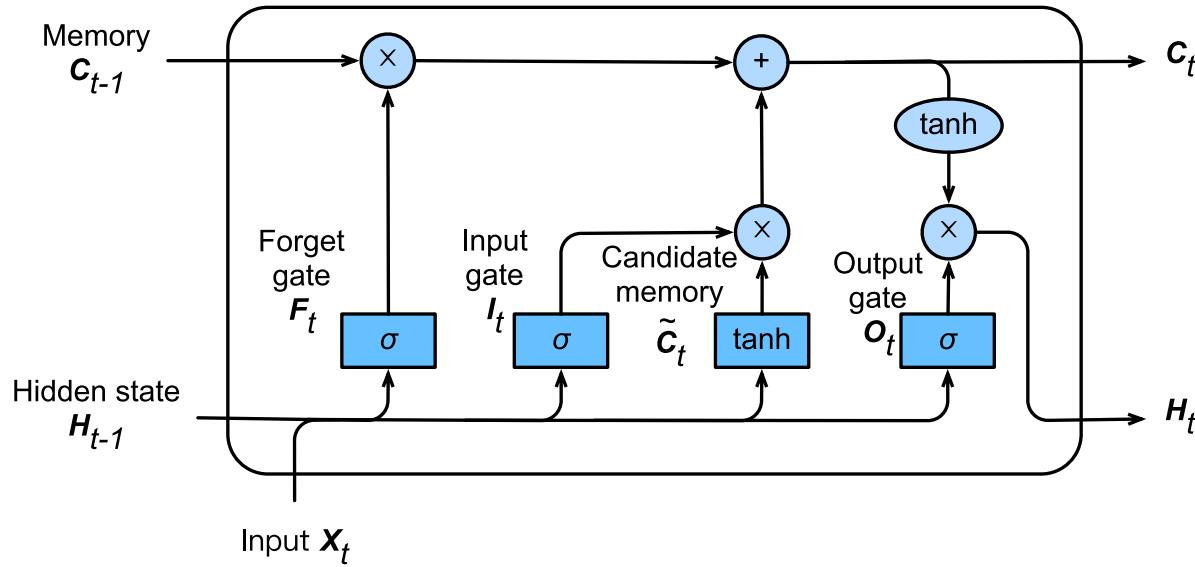
Copy



Concatenate

# Hidden State / Output

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



FC layer with  
activation function



Element-wise  
Operator

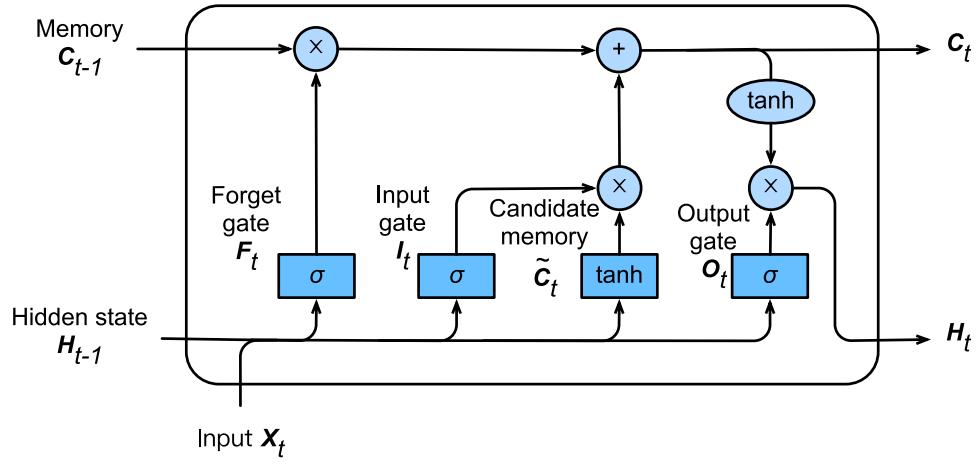


Copy



Concatenate

# Hidden State / Output



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

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

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

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

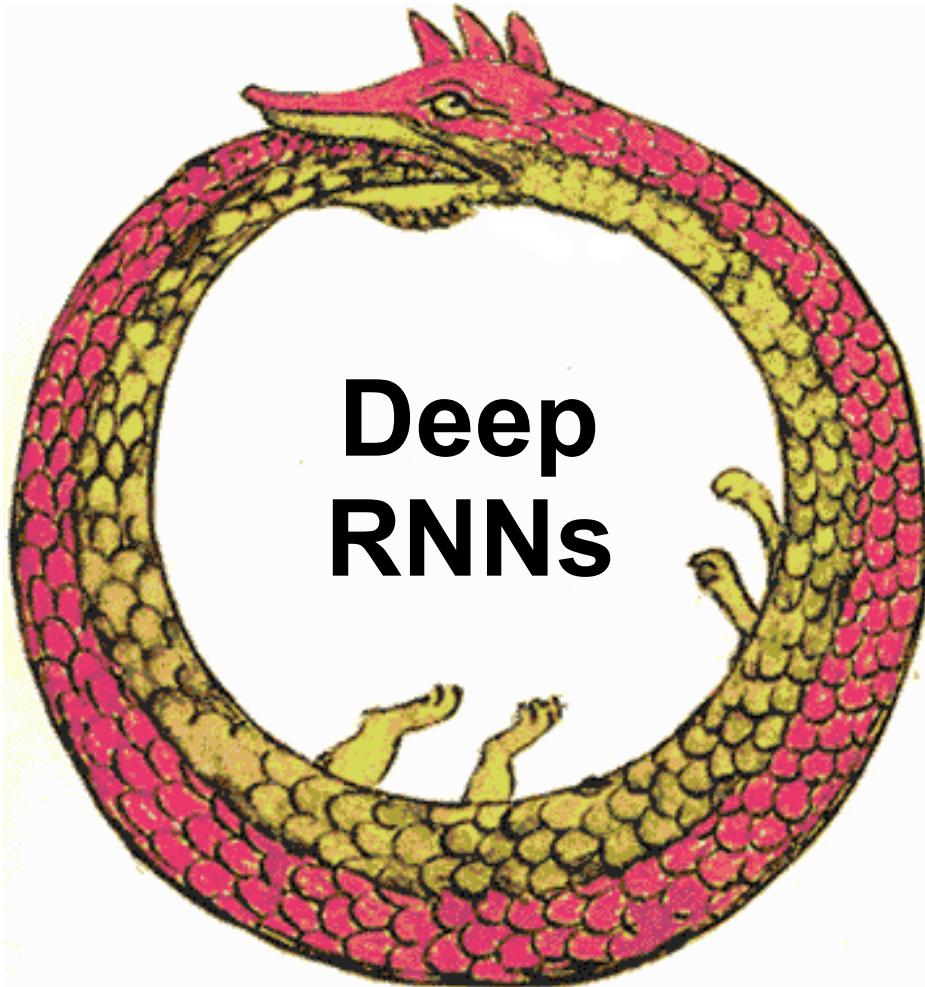
$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t$$

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

# LSTM Notebook

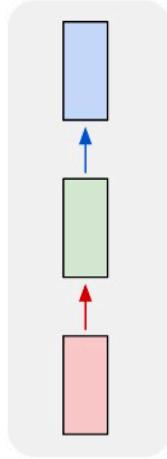
<http://amlc-d2l.corp.amazon.com/>



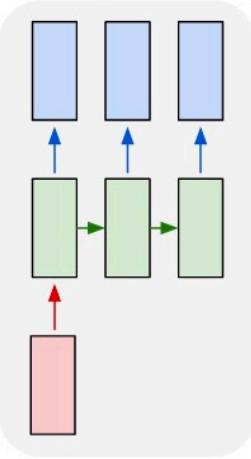


# Using RNNs

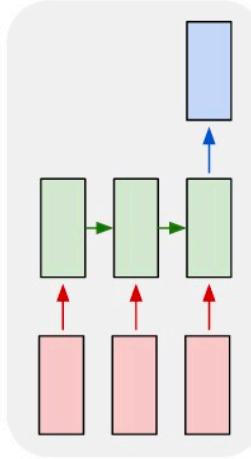
one to one



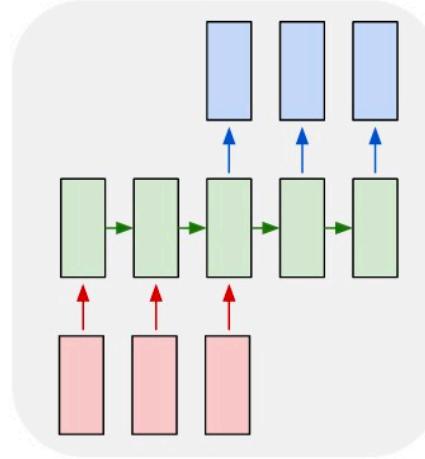
one to many



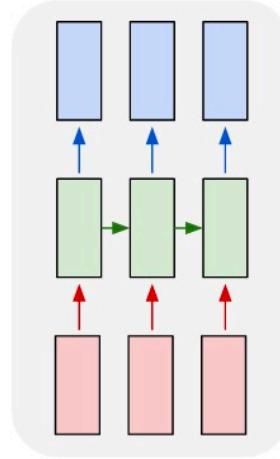
many to one



many to many



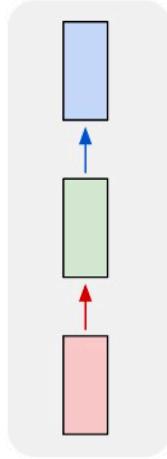
many to many



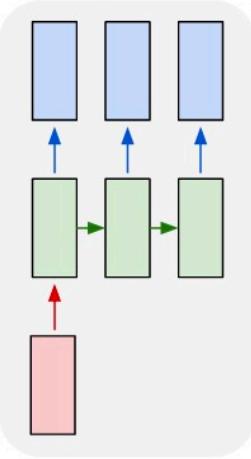
- Encode sequence
- Decode sequence
- Do both

# Using RNNs

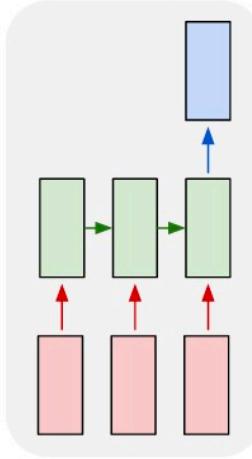
one to one



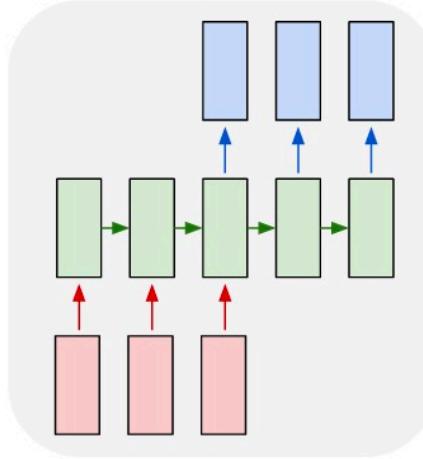
one to many



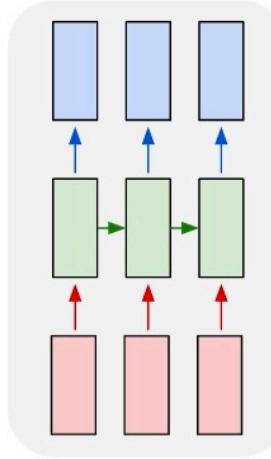
many to one



many to many



many to many



Poetry  
Generation

Sentiment  
Analysis

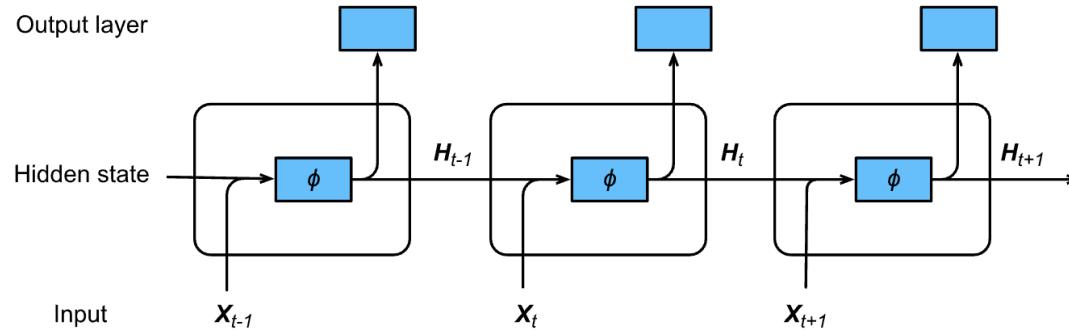
Question  
Answering

Named  
Entity  
Tagging

Document  
Classification

Machine  
Translation

# Recall - Recurrent Neural Networks



- Hidden State update

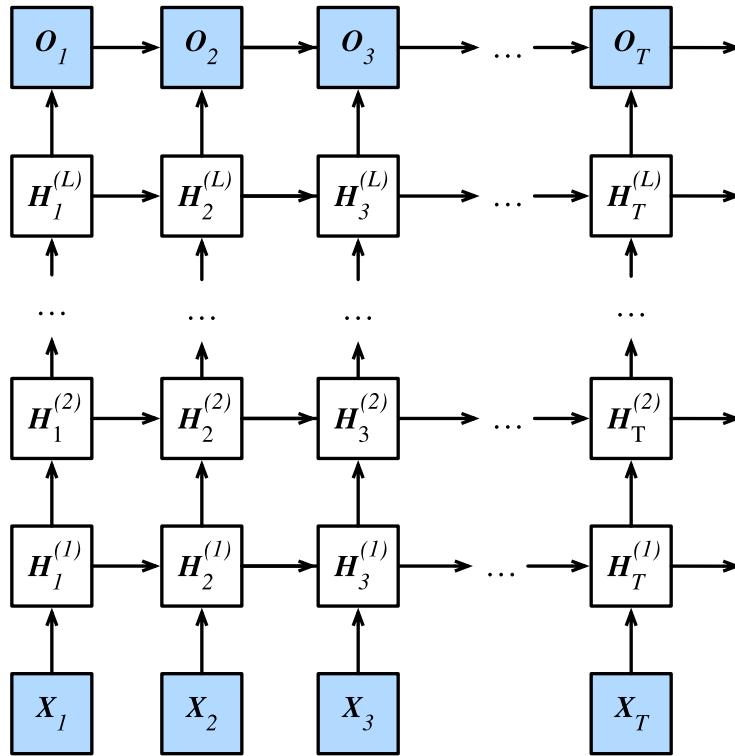
$$\mathbf{H}_t = \phi(\mathbf{W}_{hh}\mathbf{H}_{t-1} + \mathbf{W}_{hx}\mathbf{X}_{t-1} + \mathbf{b}_h)$$

- Observation update

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

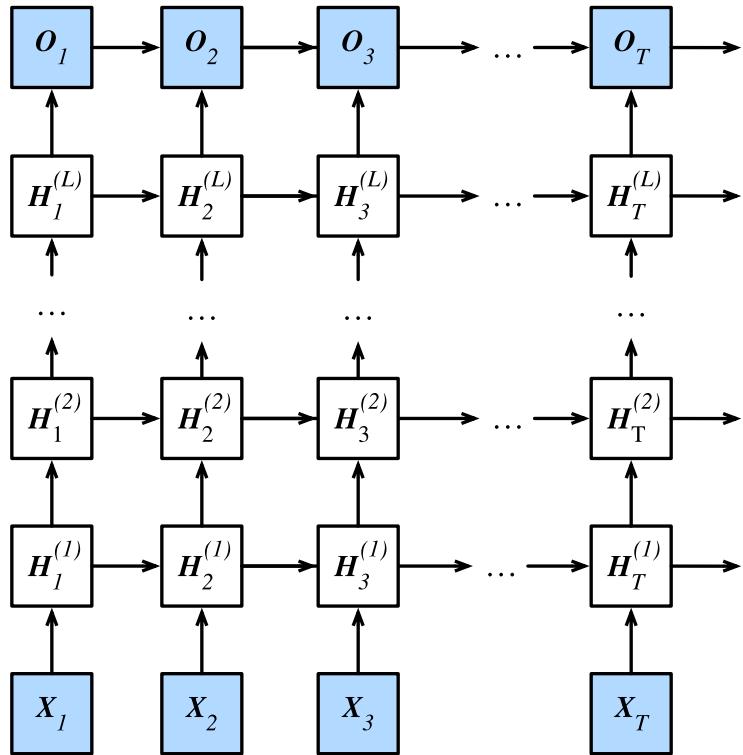
How to make  
more nonlinear?

# We go deeper



- Shallow RNN
  - Input
  - Hidden layer
  - Output
- Deep RNN
  - Input
  - **Hidden layer**
  - **Hidden layer**
  - ...
  - Output

# Plan B - We go deeper



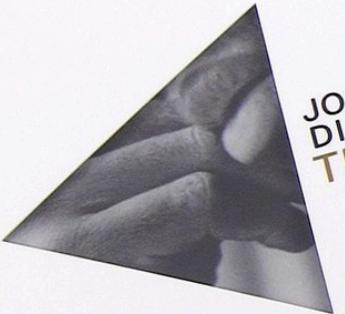
$$\mathbf{H}_t = f(\mathbf{H}_{t-1}, \mathbf{X}_t)$$

$$\mathbf{O}_t = g(\mathbf{H}_t)$$

$$\mathbf{H}_t^1 = f_1(\mathbf{H}_{t-1}^1, \mathbf{X}_t)$$

$$\mathbf{H}_t^j = f_j(\mathbf{H}_{t-1}^j, \mathbf{H}_t^{j-1})$$

$$\mathbf{O}_t = g(\mathbf{H}_t^L)$$



JOHN COLTRANE BOTH  
DIRECTIONS AT ONCE  
THE LOST ALBUM

# Bidirectional RNNS



# The Future Matters

I am \_\_\_\_\_

I am \_\_\_\_\_ very hungry,

I am \_\_\_\_\_ very hungry, I could eat half a pig.

# The Future Matters

I am **happy**.

I am **not** very hungry,

I am **very** very hungry, I could eat half a pig.

# The Future Matters

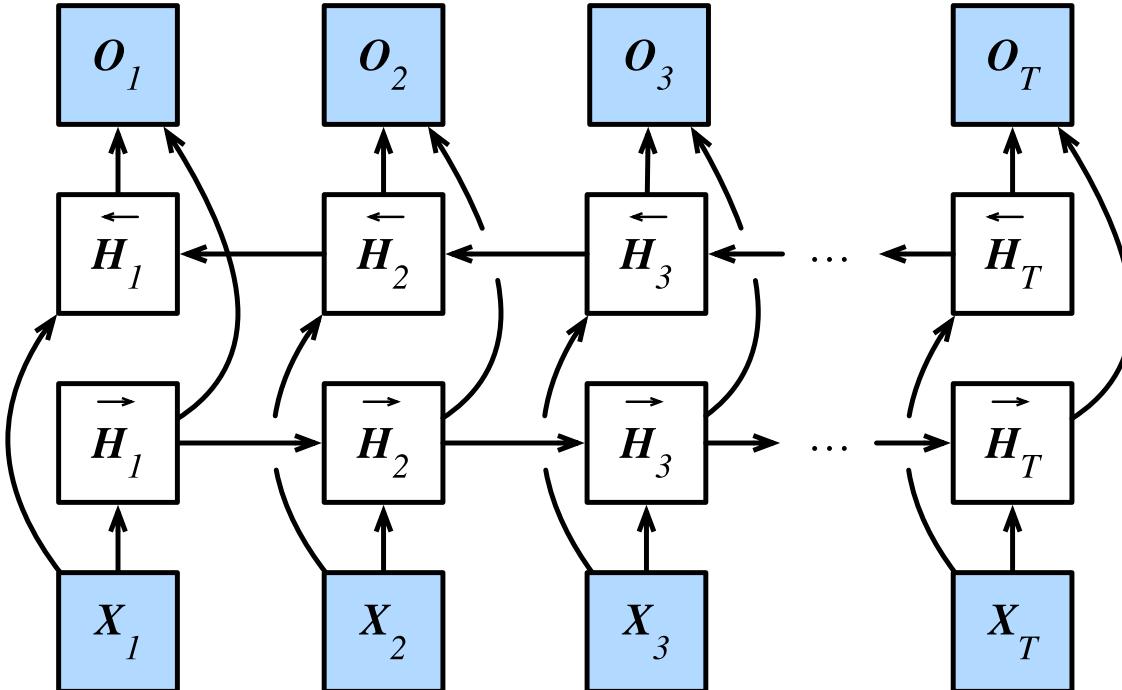
I am **happy**.

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

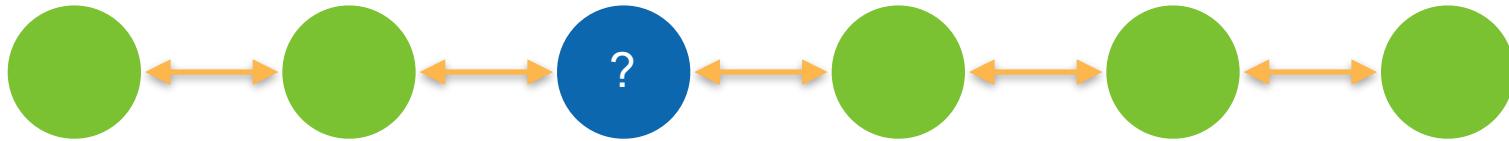
Perplexity 1.2, 86219 tokens/sec on gpu(0)

time travellererererererererererererererer  
travellererererererererererererererererer

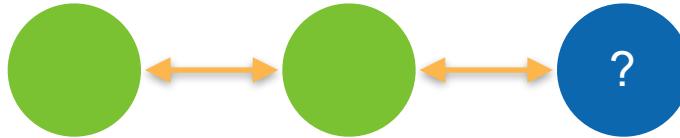
**This does not work for  
sequence generation**

# Reasons

- Training time



- Test time



Can still use it to **encode** the sequence

# Summary

- RNN stores sequence information in hidden state
- GRU and LSTM use more sophisticated gates to handle long sequence information
- RNNs can go deeper and bi-directional

## Math

- Linear algebra
- Prob & statistics
- Gradient

## Machine learning

- Loss function
- Regularization
- Model selection
- Environment

## Optimization

- Convex Optimization
- Momentum, RMSProp, Adam

## Attention

- Seq2seq w/ attention
- Transformer
- BERT

## Basic programming

- NDarray
- Autograd
- Gluon

## Basic models

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

## RNNs and NLP

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

## Performance

- Numerical stability
- multi-GPU Training

## CNNs

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

## CV

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

## GAN

- Generative Adversarial Networks
- DCGAN

## Contents in D2L.ai

- Covered today
- Partially covered today
- Not covered

# Next steps

- Textbook: [numpy.d2l.ai](https://numpy.d2l.ai)
- Toolkit for computer vision: [gluon-cv.mxnet.io](https://gluon-cv.mxnet.io)
- Toolkit for natural language processing: [gluon-nlp.mxnet.io](https://gluon-nlp.mxnet.io)
  - 1pm-5:30pm, 7/12, Meeting Center 02.100
- Toolkit for time series: [gluon-ts.mxnet.io](https://gluon-ts.mxnet.io)
  - 2:50pm-4:20pm, 7/10, Houdini North 02.200/201
- Toolkit for graph neural networks: [dgl.ai](https://dgl.ai)
  - 9am-5pm, 7/10, Van Vorst 02.204