

UC Berkeley · CSW182 | [Deep Learning]

Designing, Visualizing and Understanding Deep Neural Networks (2021)

CSW182 (2021) · 课程资料包 @ShowMeAI



视频
中英双语字幕



课件
一键打包下载



笔记
官方笔记翻译



代码
作业项目解析



视频 · B 站 [扫码或点击链接]
<https://www.bilibili.com/video/BV1Ff4y1n7ar>



课件 & 代码 · 博客 [扫码或点击链接]
<http://blog.showmeai.tech/berkeley-csw182>

Berkeley
Q-Learning
计算机视觉

循环神经网络
风格迁移
机器学习基础

可视化
模仿学习
生成模型

梯度策略
元学习
卷积网络

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2：扫码点击**底部菜单栏**
称为 **AI 内容创作者**？回复 [添砖加瓦]

Recurrent Networks

Designing, Visualizing and Understanding Deep Neural Networks

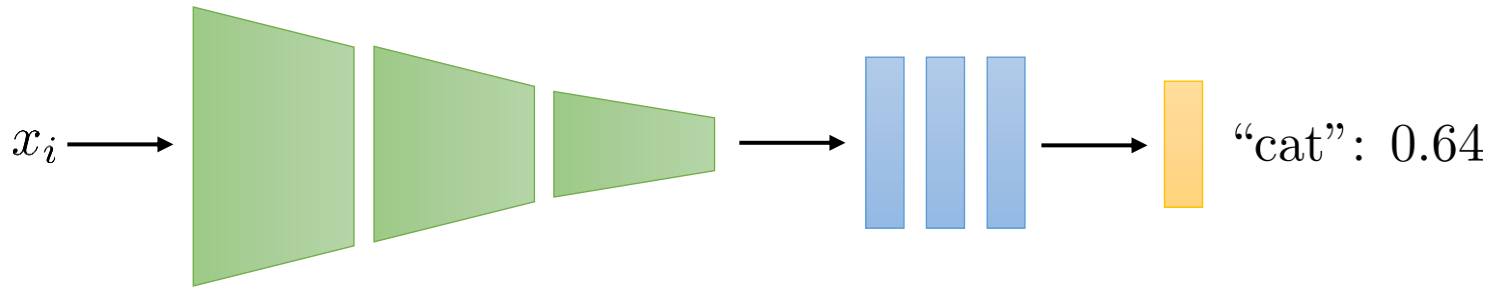
CS W182/282A

Instructor: Sergey Levine
UC Berkeley



What if we have variable-size inputs?

Before:



Now:

$$x_1 = (x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4})$$

$$x_2 = (x_{2,1}, x_{2,2}, x_{2,3})$$

$$x_3 = (x_{3,1}, x_{3,2}, x_{3,3}, x_{3,4}, x_{3,5})$$

Examples:

classifying sentiment for a phrase (sequence of words)

recognizing phoneme from sound (sequence of sounds)

classifying the activity in a video (sequence of images)

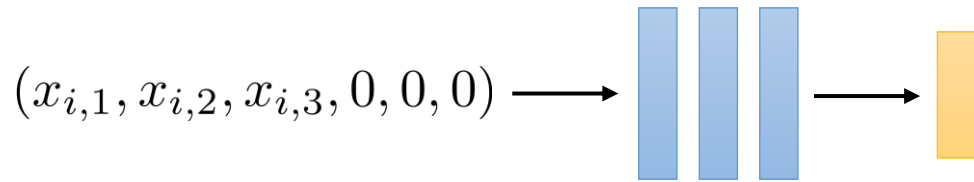
What if we have variable-size inputs?

$$x_1 = (x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4})$$

$$x_2 = (x_{2,1}, x_{2,2}, x_{2,3})$$

$$x_3 = (x_{3,1}, x_{3,2}, x_{3,3}, x_{3,4}, x_{3,5})$$

Simple idea: zero-pad up to length of longest sequence



+ very simple, and can work in a pinch

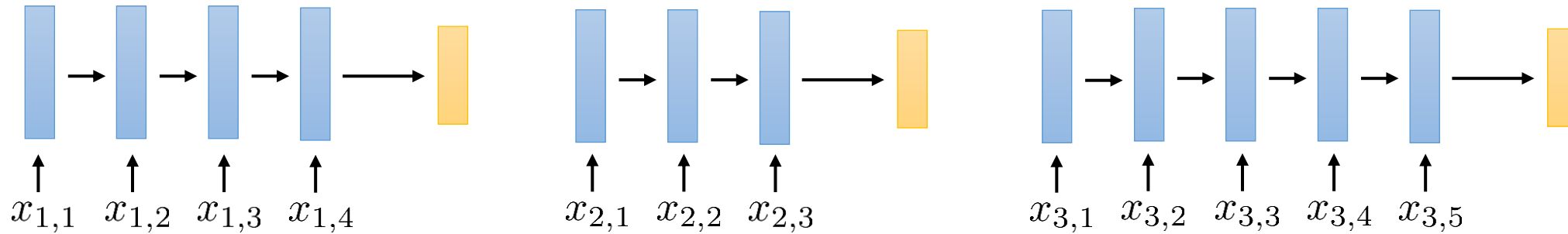
- doesn't scale very well for very long sequences

One input per layer?

$$x_1 = (x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4})$$

$$x_2 = (x_{2,1}, x_{2,2}, x_{2,3})$$

$$x_3 = (x_{3,1}, x_{3,2}, x_{3,3}, x_{3,4}, x_{3,5})$$



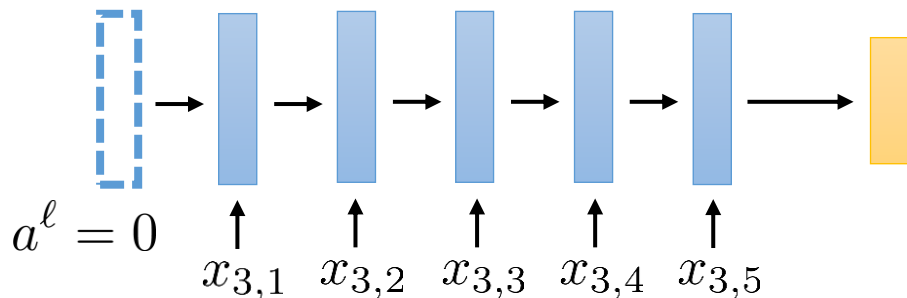
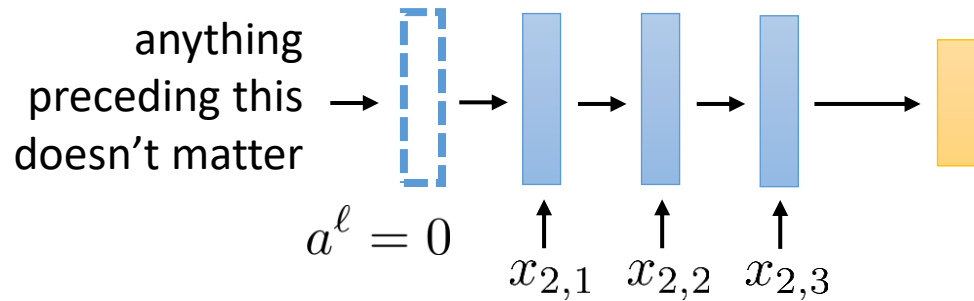
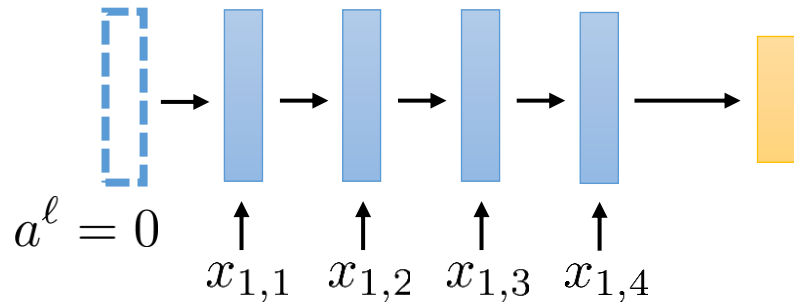
each layer:

$$\bar{a}^{\ell-1} = \begin{bmatrix} a^{\ell-1} \\ x_{i,t} \end{bmatrix} \quad z^\ell = W^\ell \bar{a}^{\ell-1} + b^\ell \quad a^\ell = \sigma(z^\ell)$$

Note: this doesn't actually work very well in practice, we'll discuss this more later

Obvious question: what happens to the missing layers?

Variable layer count?



This is more efficient than always 0-padding the sequence up to max length

Each layer is much smaller than the giant first layer we would need if we feed in the whole sequence at the first layer

The shorter the sequence, the fewer layers we have to evaluate

But the total number of weight matrices increases with max sequence length!

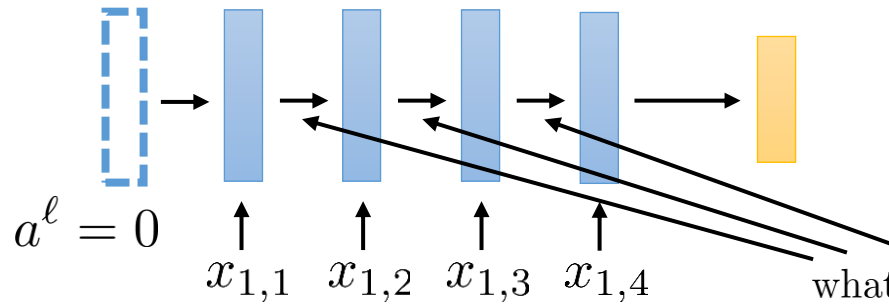
each layer:

$$\bar{a}^{\ell-1} = \begin{bmatrix} a^{\ell-1} \\ x_{i,t} \end{bmatrix}$$

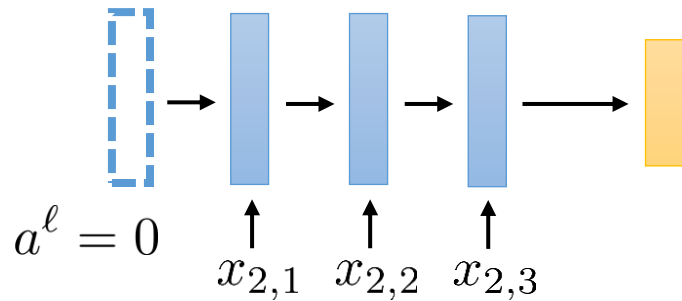
$$z^\ell = W^\ell \bar{a}^{\ell-1} + b^\ell$$

$$a^\ell = \sigma(z^\ell)$$

Can we share weight matrices?



what if W^ℓ is the same for all these layers?



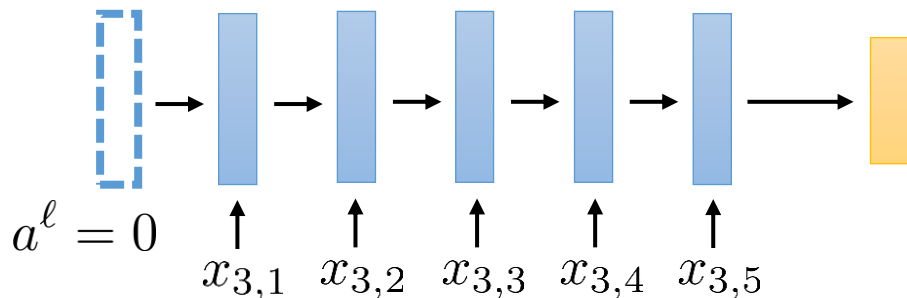
i.e., $W^{\ell_i} = W^{\ell_j}$ for all i, j

$b^{\ell_i} = b^{\ell_j}$ for all i, j

we can have as many “layers” as we want!

this is called a **recurrent** neural network (RNN)

could also call this a “variable-depth” network perhaps?



each layer:

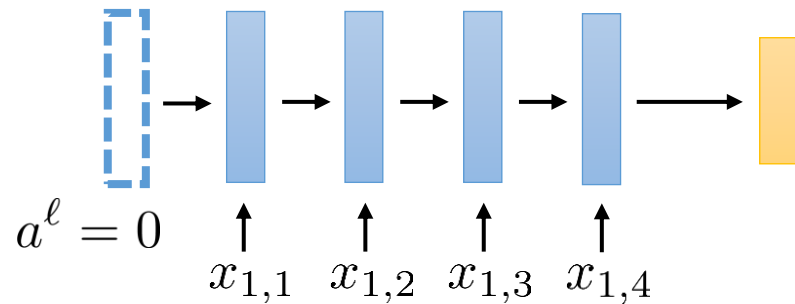
$$\bar{a}^{\ell-1} = \begin{bmatrix} a^{\ell-1} \\ x_{i,t} \end{bmatrix}$$

$$z^\ell = W^\ell \bar{a}^{\ell-1} + b^\ell$$

$$a^\ell = \sigma(z^\ell)$$

Aside: RNNs and time

What we just learned:

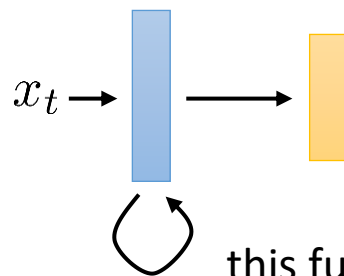


“a recurrent neural network extends a standard neural network along the time dimension”

(or some other assertion of this sort)

This is technically true, but somewhat unhelpful for actually understanding how RNNs work, and makes them seem more mystical than they are

What you often see in textbooks/classes:



this funny thing represents the fact that this layer also gets its own “previous” value as input

RNNs are just neural networks that share weights across multiple layers, take an input at each layer, and have a variable number of layers

How do we train this?

Backpropagation:

forward pass: calculate each $a^{(i)}$ and $z^{(i)}$

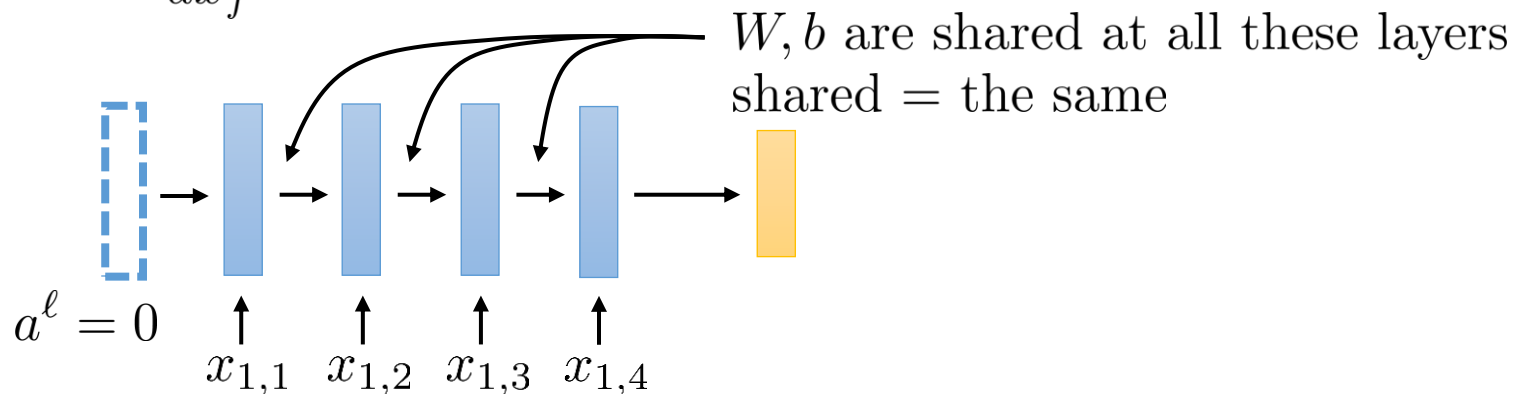
backward pass:

initialize $\delta = \frac{d\mathcal{L}}{dz^{(n)}}$

for each f with input x_f & params θ_f from end to start:

~~$\frac{d\mathcal{L}}{d\theta_f} \leftarrow \frac{df}{d\theta_f} \delta$~~ ← taken literally, gradient at $\ell - 1$ will “overwrite” gradient at ℓ
most libraries don’t have this problem, because they do it differently

$\delta \leftarrow \frac{df}{dx_f} \delta$ $\frac{d\mathcal{L}}{d\theta_f} += \frac{df}{d\theta_f} \delta$ “accumulate” the gradient during the backward pass



To convince yourself that this is true:

$f(x) = g(x, h(x))$ how does this resemble role of W in the RNN?

$$\frac{d}{dx} f(x) = \frac{dg}{dx} + \frac{dh}{dx} \frac{dg}{dh}$$

↑ derivative through first argument

← derivative through second argument (via chain rule)

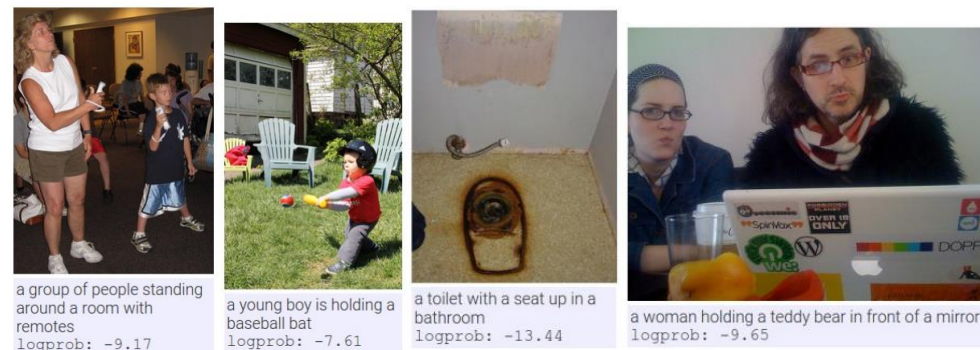
What if we have variable-size outputs?

Examples:

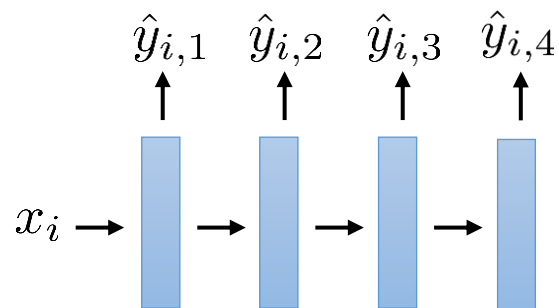
generating a text caption for an image

predicting a sequence of future video frames

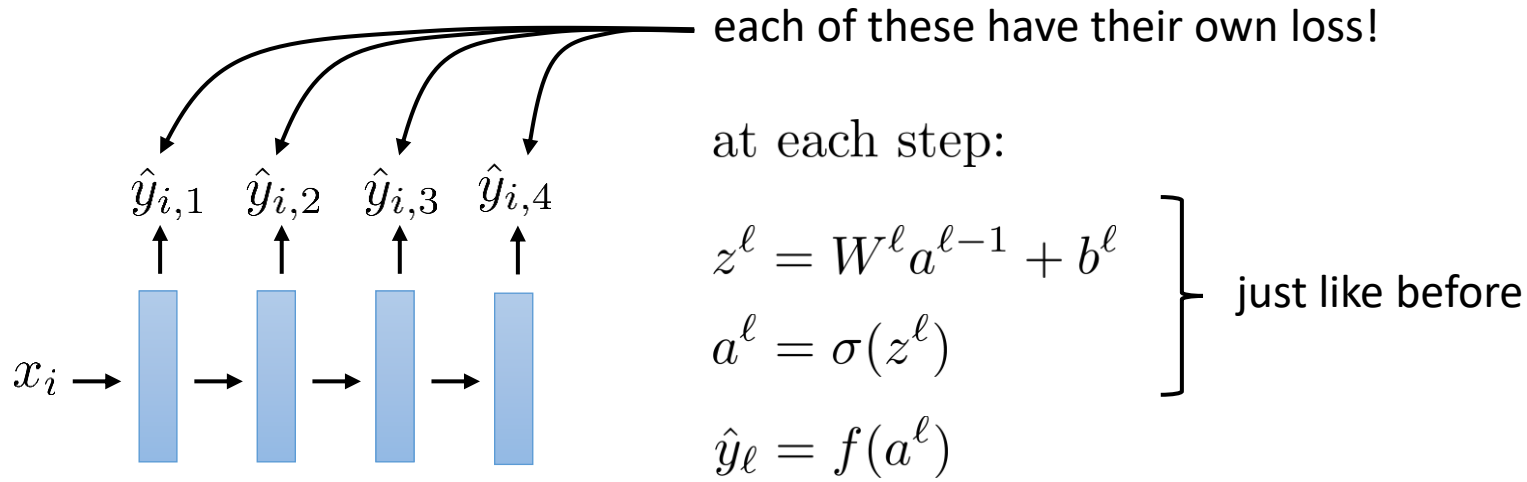
generating an audio sequence



frames with yellow labels are predictions



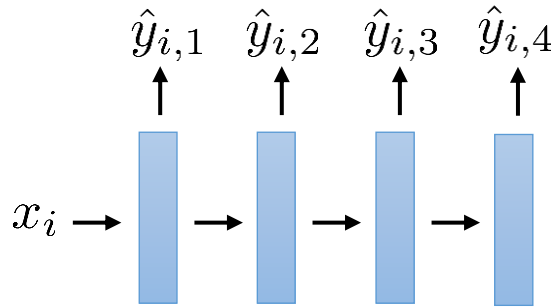
An output at every layer



we have a loss on *each* \hat{y}_ℓ
(e.g., cross-entropy)

$$\mathcal{L}(\hat{y}_{1:T}) = \sum_{\ell} \mathcal{L}_\ell(\hat{y}_\ell)$$

Let's draw the computation graph!

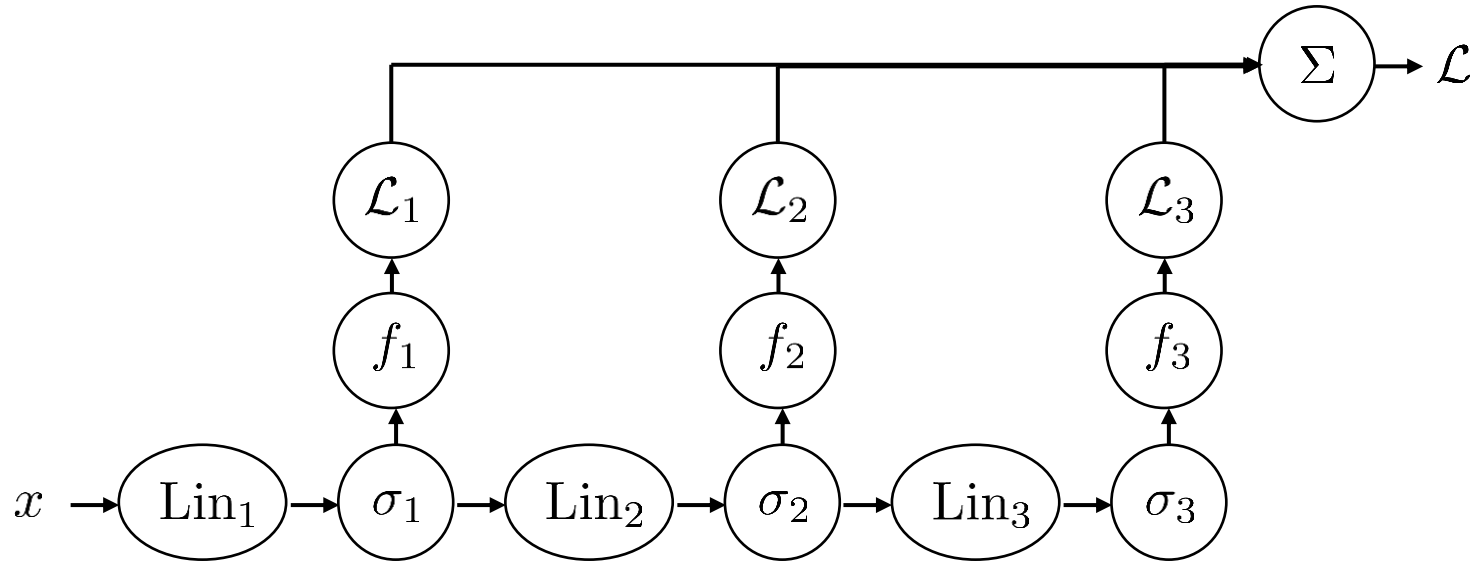


$$z^\ell = W^\ell a^{\ell-1} + b^\ell$$

$$a^\ell = \sigma(z^\ell)$$

$$\hat{y}_\ell = f(a^\ell)$$

$$\mathcal{L}(\hat{y}_{1:T}) = \sum_{\ell} \mathcal{L}_\ell(\hat{y}_\ell)$$

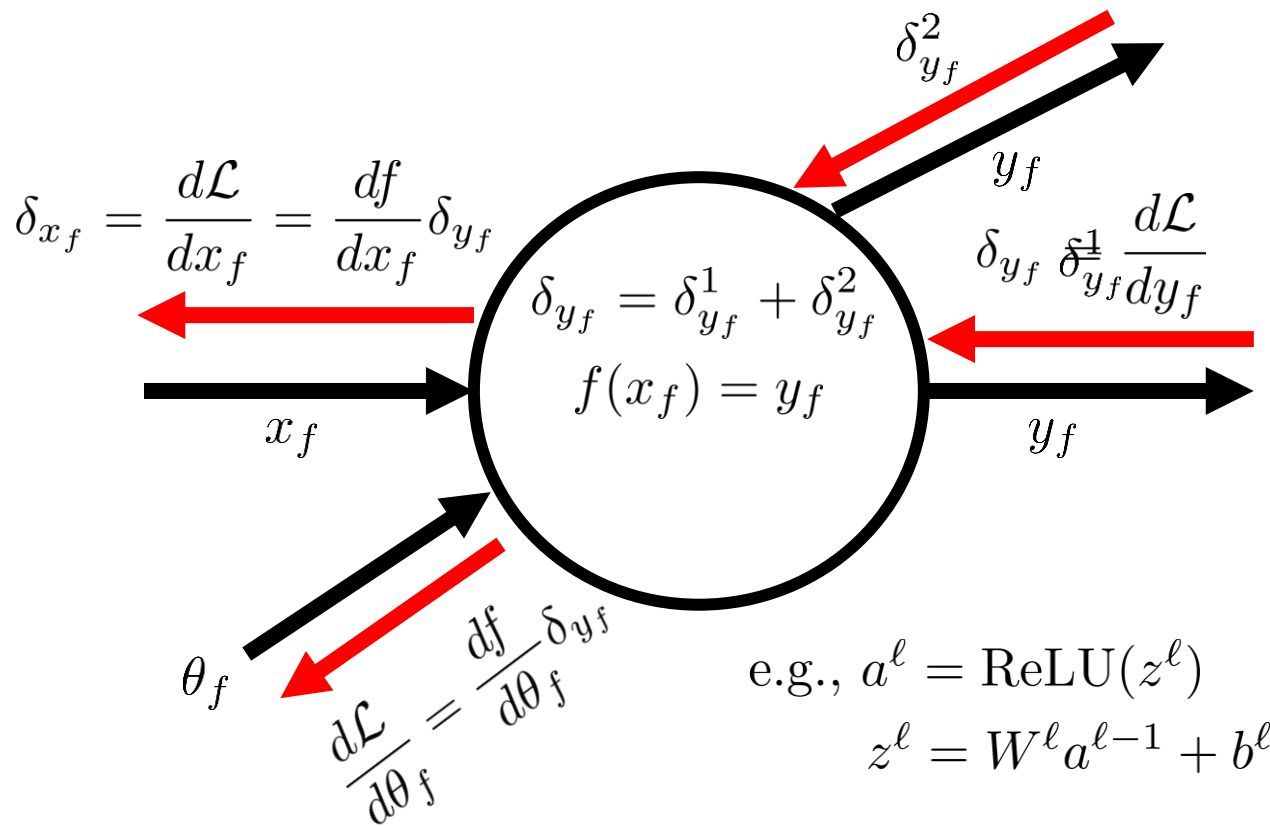


not completely obvious how to do backprop on this!

Graph-structured backpropagation

Also called reverse-mode automatic differentiation

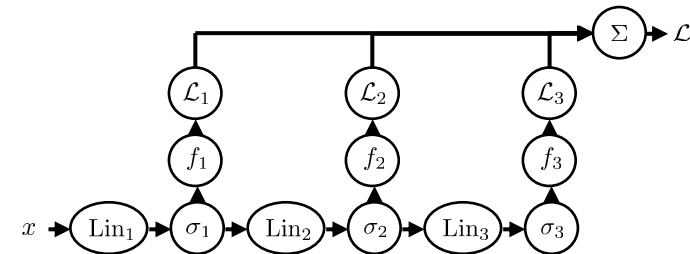
do the following at each layer $f(x_f) \rightarrow y_f$
starting with the last function, where $\delta = 1$



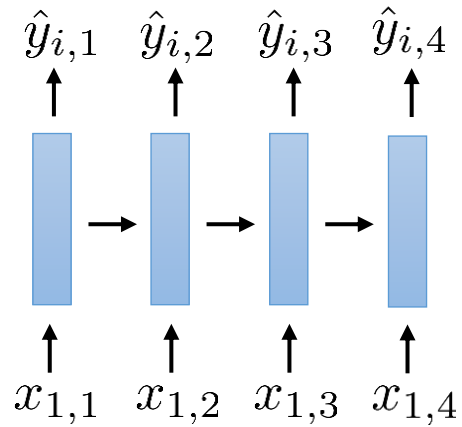
Very simple rule:

For each node with multiple descendants
in the computational graph:

Simply add up the delta vectors coming
from all of the descendants



Inputs and outputs at each step?



at each step:

$$\left. \begin{aligned} \bar{a}^{\ell-1} &= \begin{bmatrix} a^{\ell-1} \\ x_{i,t} \end{bmatrix} \\ z^{\ell} &= W^{\ell} \bar{a}^{\ell-1} + b^{\ell} \\ a^{\ell} &= \sigma(z^{\ell}) \end{aligned} \right\} \text{just like before}$$

$$\hat{y}_{\ell} = f(a^{\ell})$$

Examples:

generating a text caption for an image

← a bit subtle why there are
inputs at each time step!
we'll discuss this later

translating some text into a different language

← though there are much
better ways to do it!

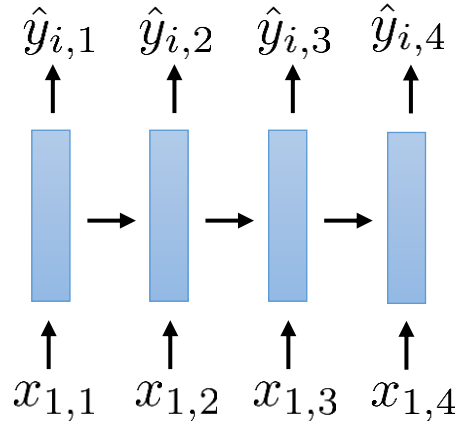
What makes RNNs difficult to train?

RNNs are extremely deep networks

Intuitively:

vanishing gradients = gradient signal from later steps never reaches the earlier steps

very bad – this prevents the RNN from “remembering” things from the beginning!



imagine our sequence length was 1000+

that's like backpropagating through 1000+ layers!

$$\frac{d\mathcal{L}}{dW^{(1)}} = \frac{dz^{(1)}}{dW^{(1)}} \frac{da^{(1)}}{dz^{(1)}} \frac{dz^{(2)}}{da^{(1)}} \frac{d\mathcal{L}}{dz^{(2)}}$$

$$\frac{d\mathcal{L}}{dW^{(1)}} = J_1 J_2 J_3 \dots J_n \frac{d\mathcal{L}}{dz^{(n)}}$$

If we multiply many many numbers together, what will we get?

If most of the numbers are < 1 , we get 0

If most of the numbers are > 1 , we get infinity

We only get a reasonable answer if the numbers are all close to 1!

“vanishing gradients”

big problem!

“exploding gradients”

could fix with gradient clipping

Promoting better gradient flow

Basic idea: (similar to what we saw before) we would really like the gradients to be close to 1

which gradients?

each layer:

$$\bar{a}_{t-1} = \begin{bmatrix} a_{t-1} \\ x_t \end{bmatrix} \quad z_t = W\bar{a}_{t-1} + b \quad a_t = \sigma(z_t)$$

$$a_t = q(a_{t-1}, x_t) \quad \text{“RNN dynamics”}$$

dynamics Jacobian $\frac{dq}{da_{t-1}} \approx \mathbf{I}$ \longleftarrow best gradient flow

not always good – only good when we want to **remember**

sometimes we may want to **forget**

Promoting better gradient flow

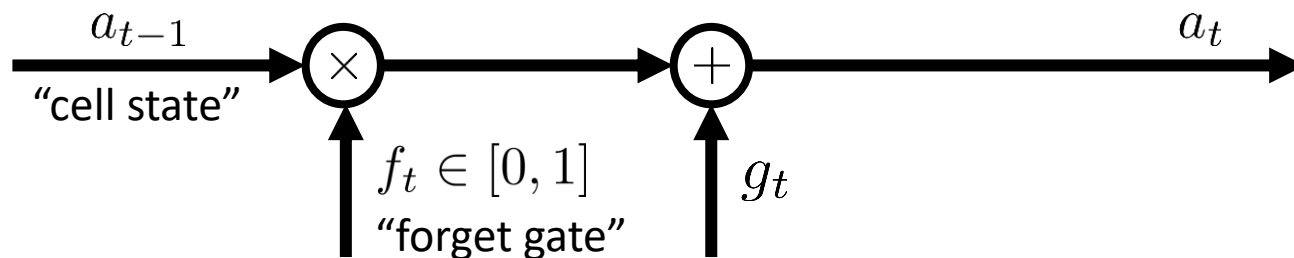
Basic idea: (similar to what we saw before) we would really like the gradients to be close to 1

Intuition: want $\frac{dq_i}{da_{t-1,i}} \approx 1$ if we choose to *remember* $a_{t-1,i}$

for each unit, we have a little “neural circuit” that decides whether to remember or overwrite

if “remembering,” just copy the previous activation as it is

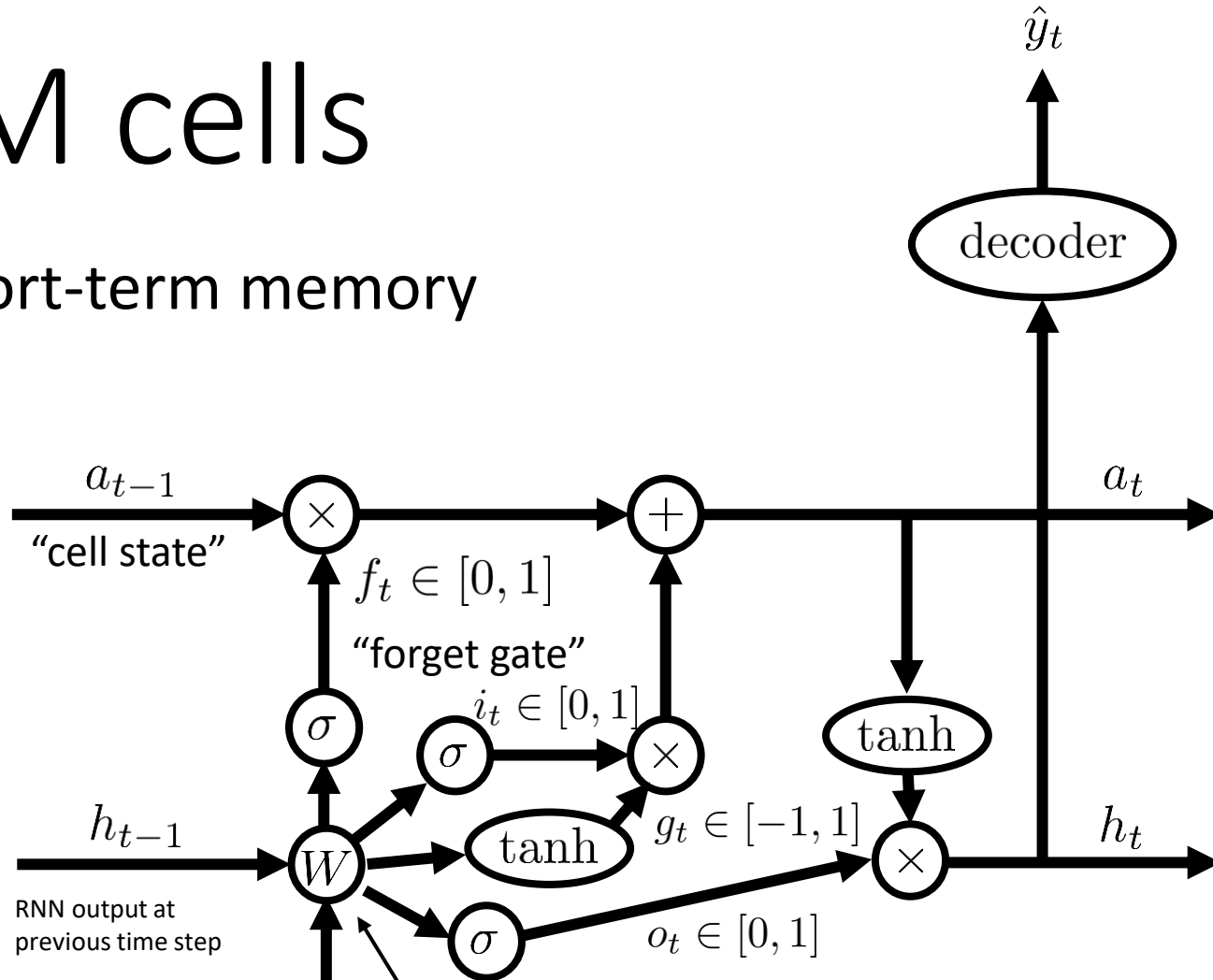
if “forgetting,” just overwrite it with something based on the current input



$$\begin{array}{c} f_t \in [0, 1] \\ \downarrow \\ a_t = a_{t-1}f_t + g_t \\ \frac{dq_i}{da_{t-1,i}} = f_t \in [0, 1] \end{array}$$

LSTM cells

Long short-term memory



Isn't this all a little arbitrary?

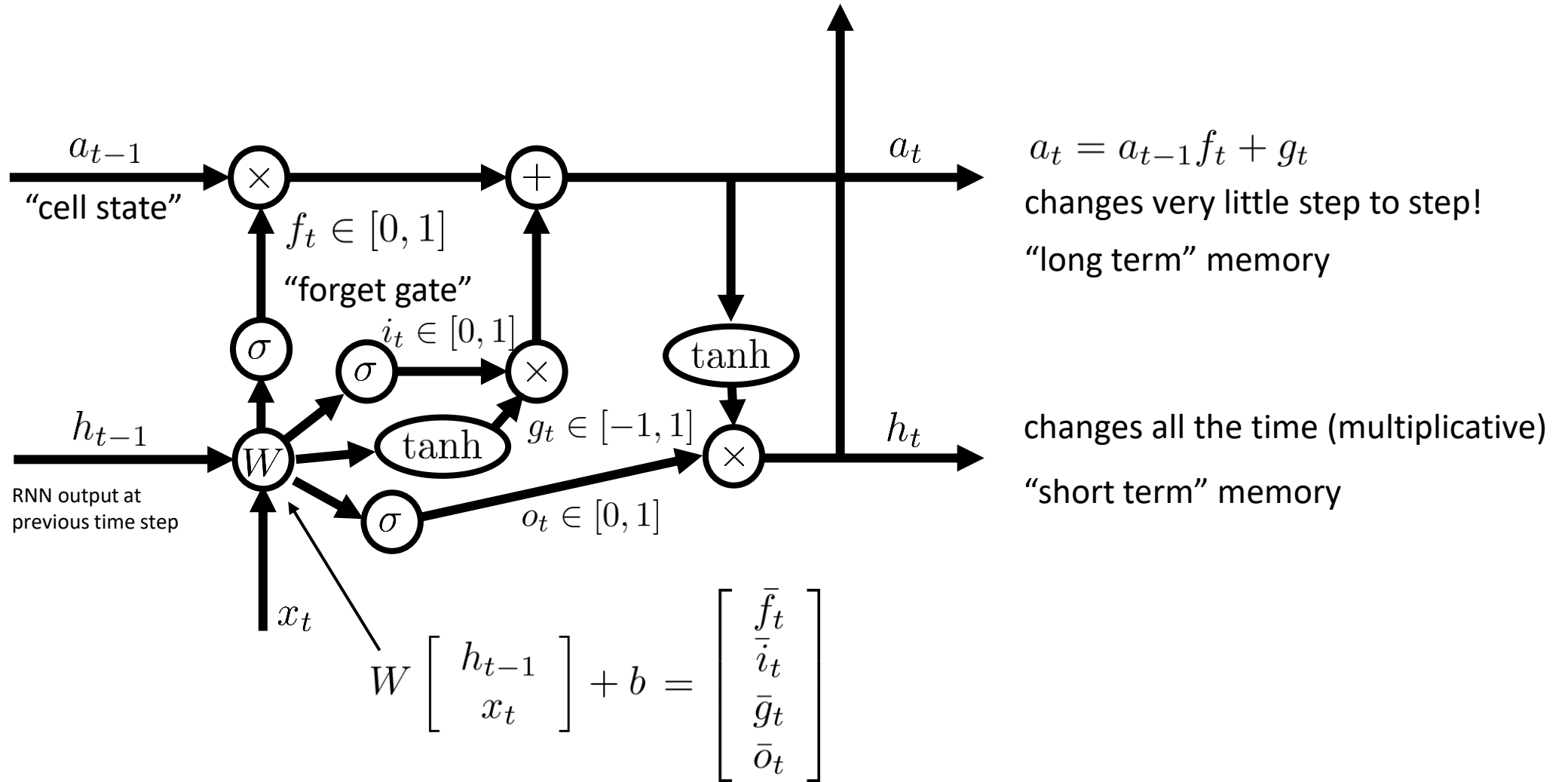
Well, yes, but it ends up working quite well in practice, and much better than a naïve RNN!

output is **4x** larger in dimensionality than RNN cell!

$$W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b = \begin{bmatrix} \bar{f}_t \\ \bar{i}_t \\ \bar{g}_t \\ \bar{o}_t \end{bmatrix}$$

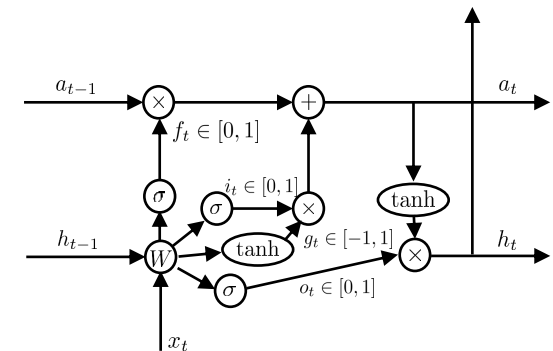
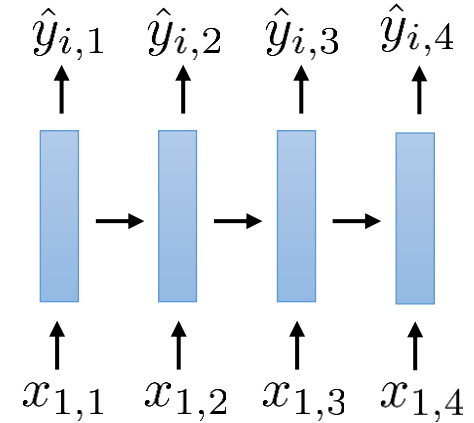
each of these is a vector with same dims as h_{t-1}

Why do LSTMs train better?



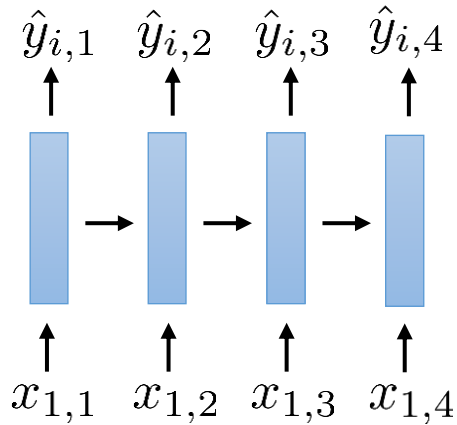
Some practical notes

- In practice, RNNs almost always have both an input and an output at each step (we'll see why in the next section)
- In practice, naïve RNNs like in part 1 almost never work
- LSTM units are OK – they work fine in many cases, and dramatically improve over naïve RNNs
 - Still require way more hyperparameter tuning than standard fully connected or convolutional networks
- Some alternatives (that we'll learn about later) can work better for sequences
 - Temporal convolutions
 - Transformers (temporal attention)
- LSTM cells are annoyingly complicated, but once implemented, they can be used the same as any other type of layer (hurray for abstraction!)
- There some variants of the LSTM that are a bit simpler and work just as well
 - Gated recurrent unit (GRU)



Using RNNs

Autoregressive models and structured prediction



most RNNs used in practice look like this

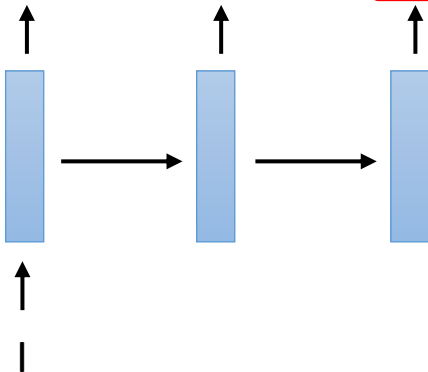
why?

most problems that require multiple outputs have strong **dependencies** between these outputs

this is sometimes referred to as **structured** prediction

Example: text generation

think: 0.3 therefore: 0.3 I: 0.3
like: 0.3 machine: 0.3 learning: 0.3
am: 0.4 not: 0.4 just: 0.4



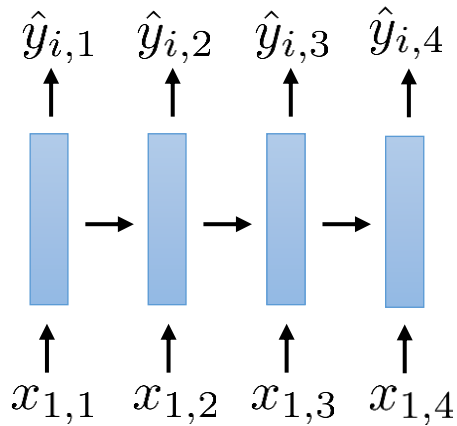
I think therefore I am

I like machine learning

I am not just a neural network

we get a nonsense output
even though the network
had exactly the right
probabilities!

Autoregressive models and structured prediction



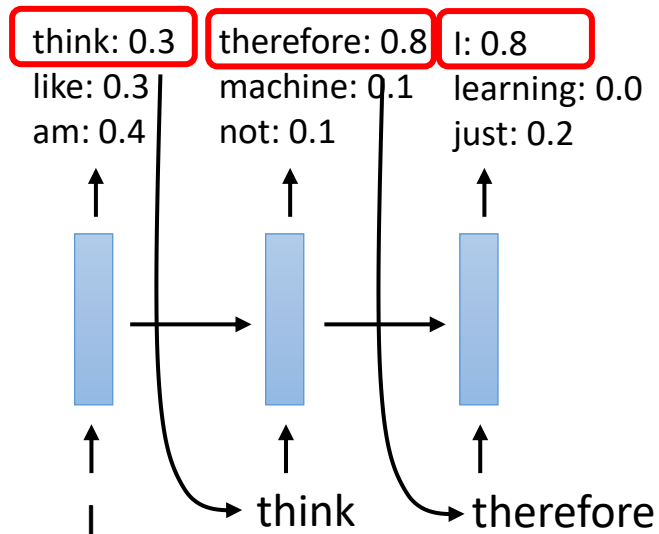
most RNNs used in practice look like this

why?

most problems that require multiple outputs have strong **dependencies** between these outputs

this is sometimes referred to as **structured** prediction

Example: text generation



I think therefore I am

I like machine learning

I am not just a neural network

Key idea: past outputs should influence future outputs!

we get a nonsense output even though the network had exactly the right probabilities!

Autoregressive models and structured prediction

How do we train it?

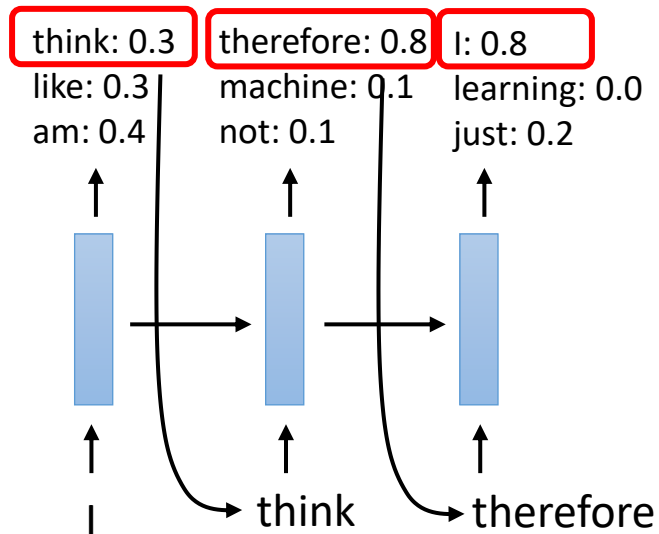
Basic version: just set inputs to be entire training sequences, and ground truth outputs to be those same sequences (offset by one step)

$x_{1:5} = (\text{"I"}, \text{"think"}, \text{"therefore"}, \text{"I"}, \text{"am"})$

$y_{1:5} = (\text{"think"}, \text{"therefore"}, \text{"I"}, \text{"am"}, \text{stop_token})$

This teaches the network to output “am” if it sees “I think therefore I”

Example: text generation

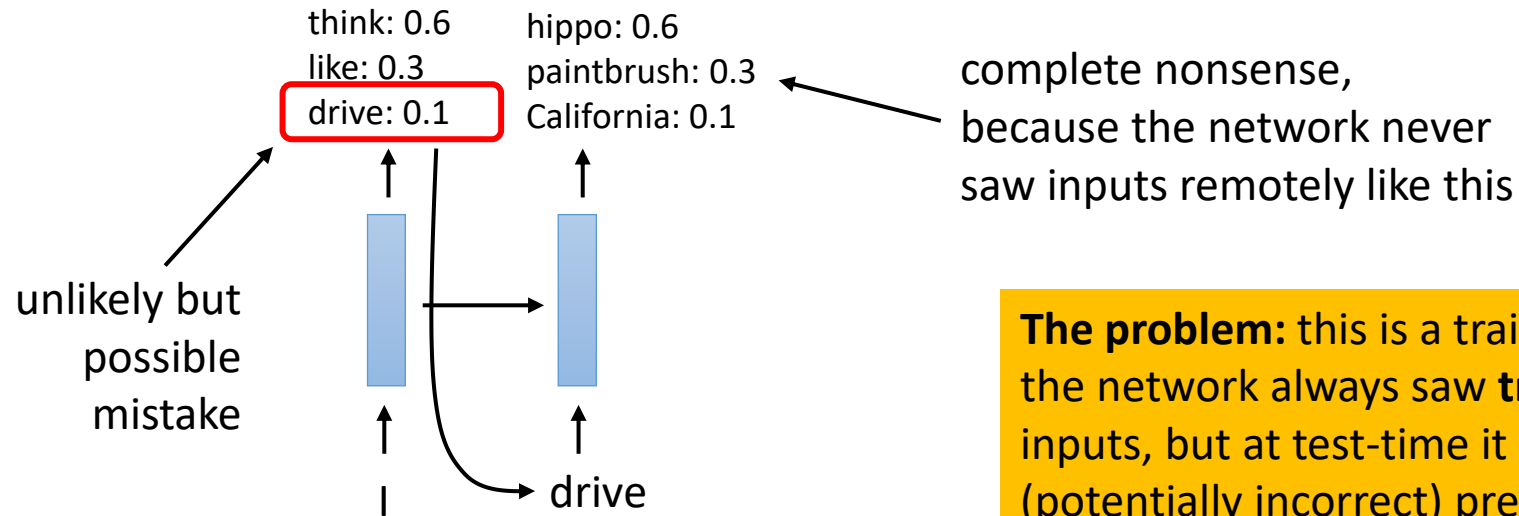


I think therefore I am

I like machine learning

I am not just a neural network

Aside: distributional shift



we got unlucky, but now the
model is completely confused
it never saw "I drive" before

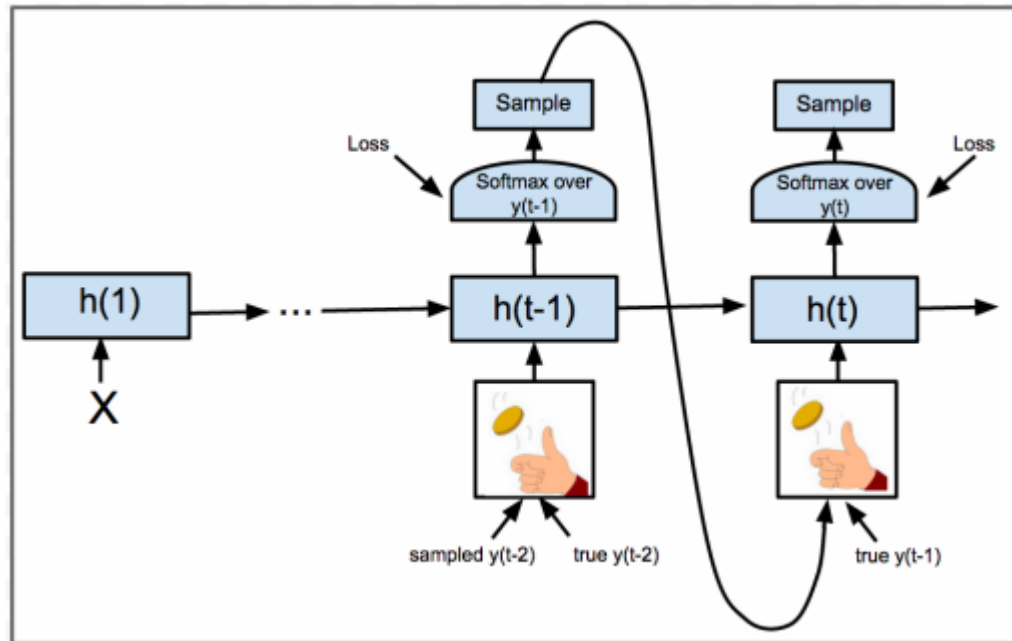
The problem: this is a training/test discrepancy:
the network always saw **true** sequences as
inputs, but at test-time it gets as input its own
(potentially incorrect) predictions

This is called **distributional shift**, because the
input distribution **shifts** from true strings (at
training) to synthetic strings (at test time)

Even **one** random mistake can completely
scramble the output!

Aside: scheduled sampling

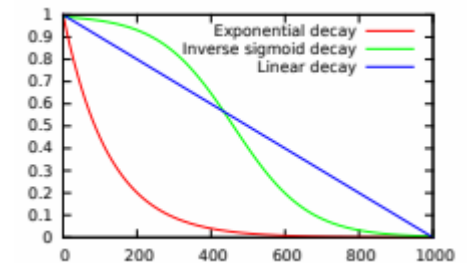
An old trick from reinforcement learning adapted to training RNNs



At the beginning of training, mostly feed in ground truth tokens as input, since model predictions are mostly nonsense

At the end of training, mostly feed in the model's own predictions, to mitigate distribution shift

schedules for probability of using ground truth input token

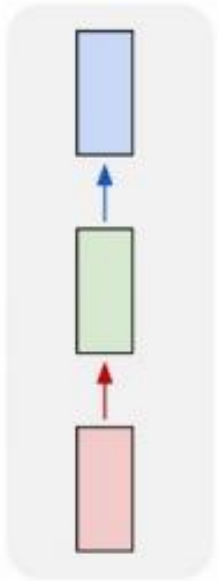


Randomly decide whether to give the network a ground truth token as input during training, or its own previous prediction

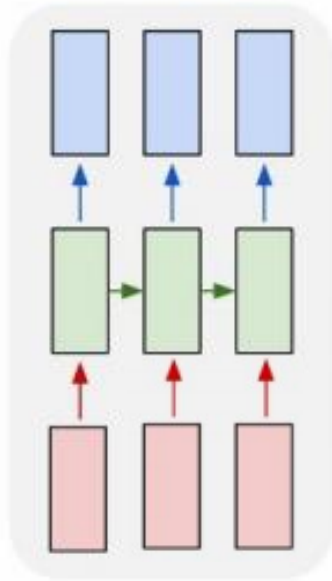
Different ways to use RNNs

in reality, we almost always use
autoregressive generation like this

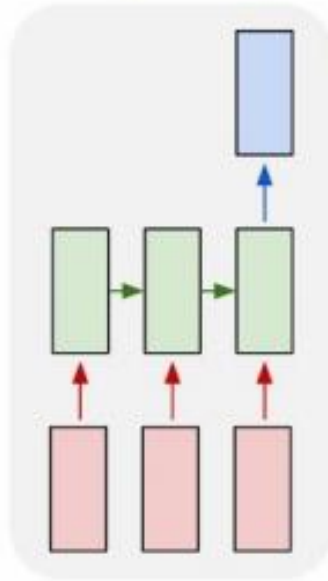
one to one



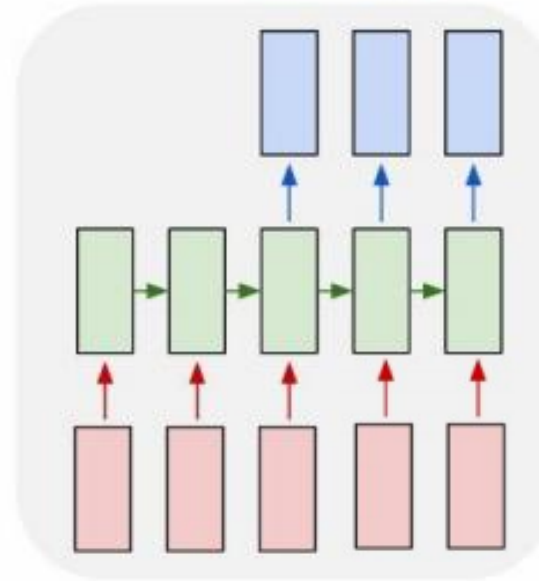
one to many



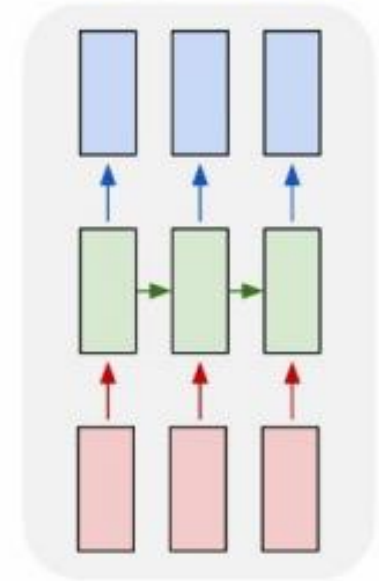
many to one



many to many



many to many



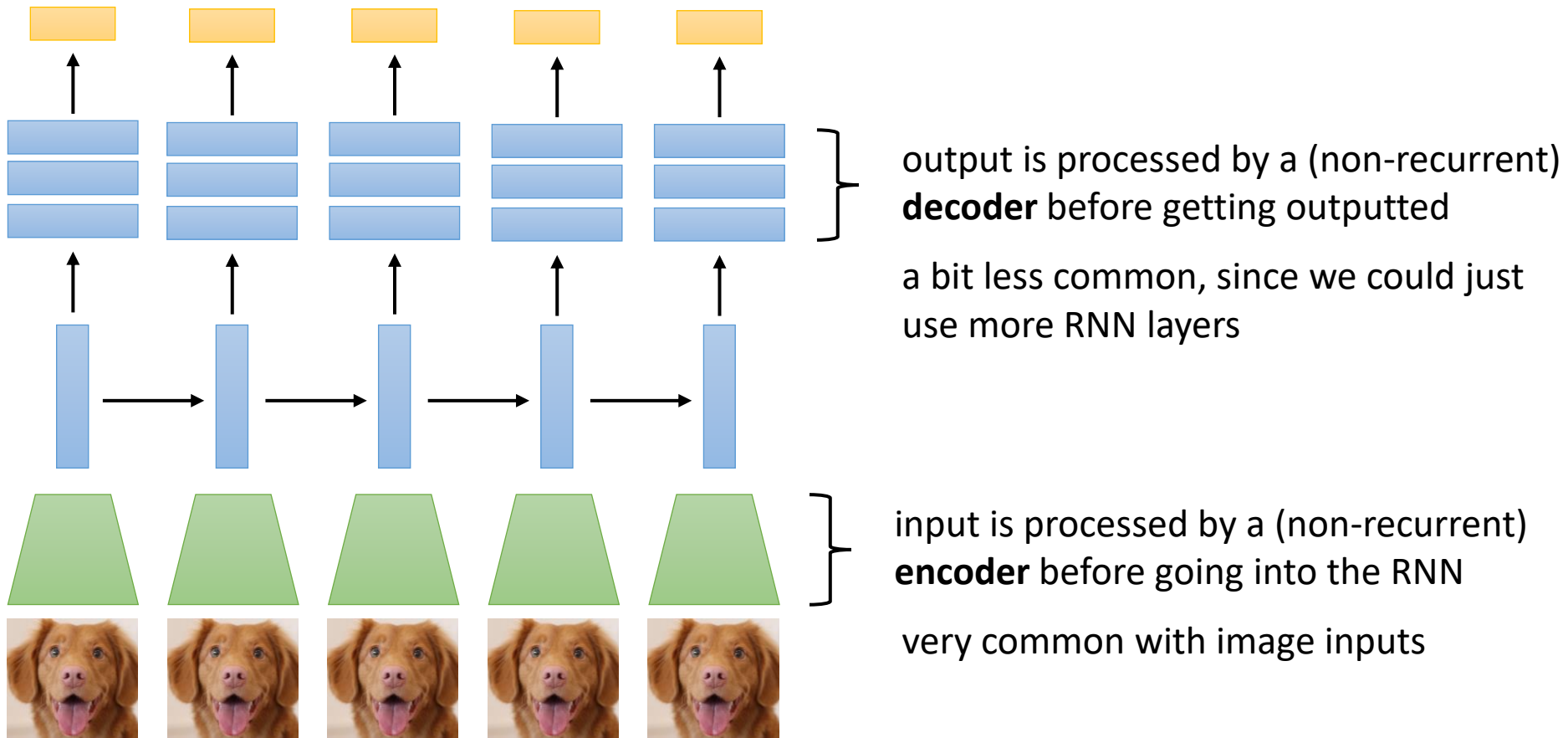
e.g., activity recognition

e.g., frame-level video annotation

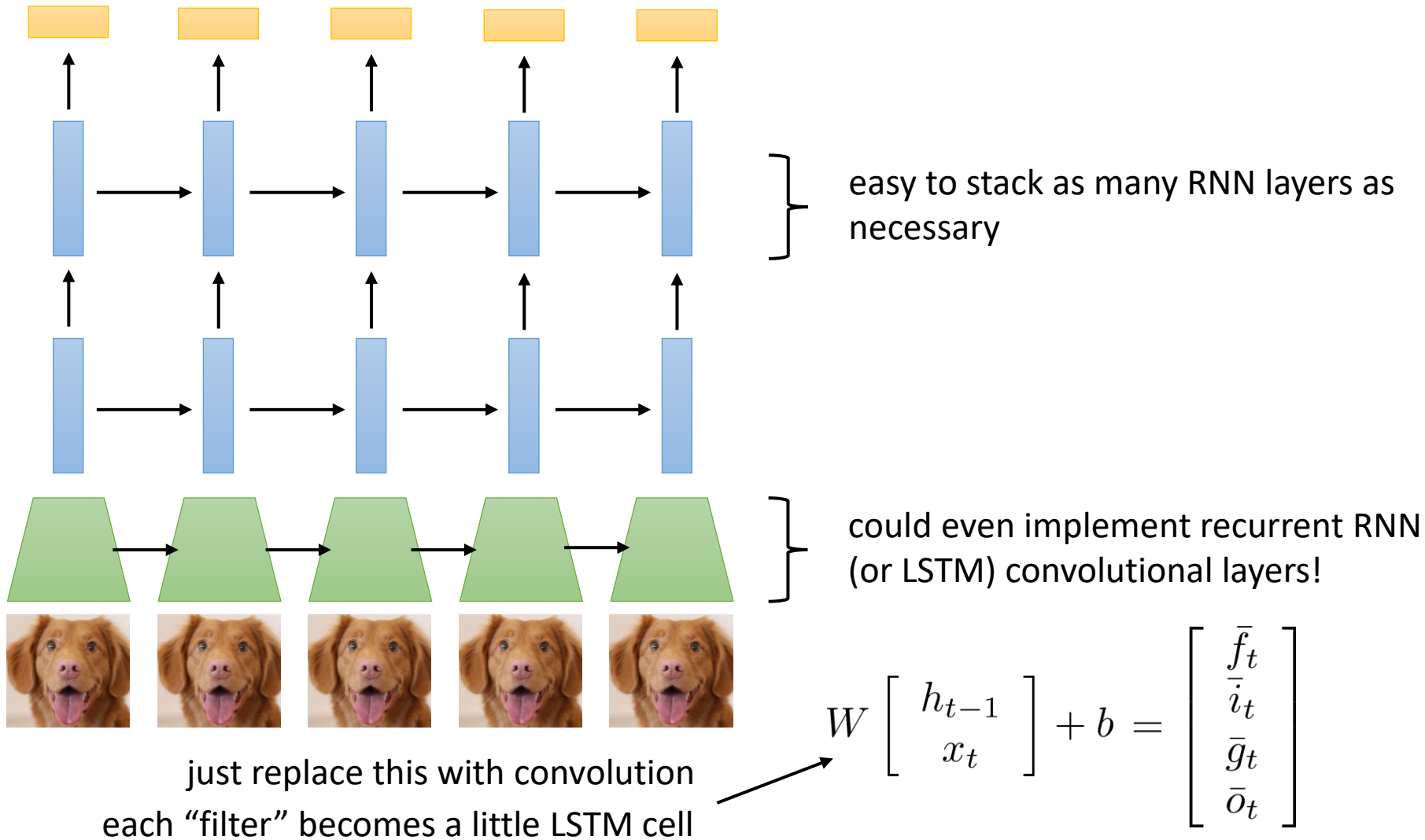
e.g., image captioning

e.g., machine translation

RNN encoders and decoders

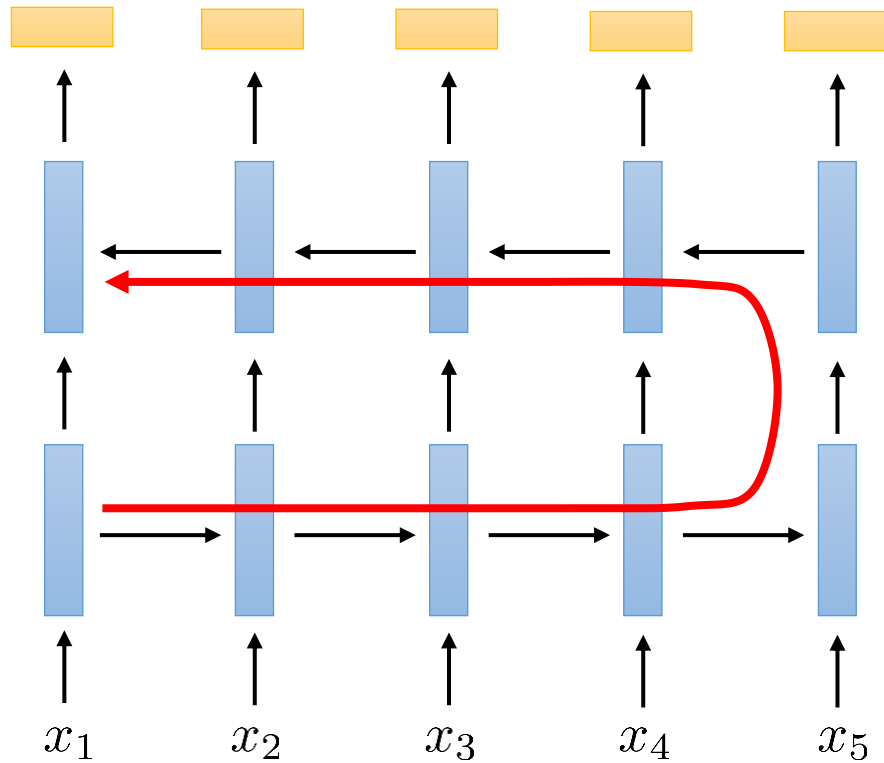


RNNs with many layers



Bidirectional models

Example: speech recognition



Problem: the word at a particular time step might be hard to guess without looking at the rest of the utterance!

(for example, can't tell if a word is finished until hearing the ending)

This is an even bigger problem in machine translation, but there we use slightly different types of models

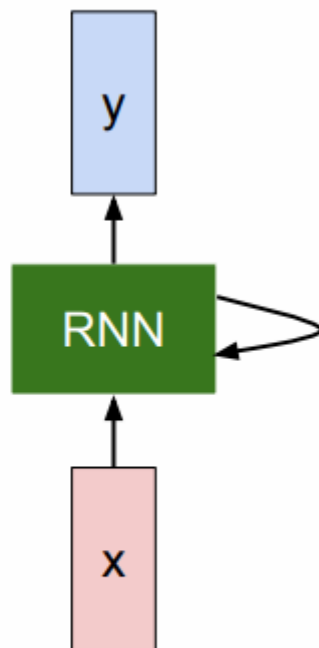
Some (vivid) examples

THE SONNETS

by William Shakespeare

From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the ripper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,
Thyself thy foe, to thy sweet self too cruel:
Thou that art now the world's fresh ornament,
And only herald to the gaudy spring,
Within thine own buduriest thy content,
And tender churl mak'st waste in niggarding:
Pity the world, or else this glutton be,
To eat the world's due, by the grave and thee.

When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a tatter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.



Some (vivid) examples

at first:

tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrqd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

↓
train more

"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuw y fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

↓
train more

Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.

↓
train more

"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftended him.
Pierre aking his soul came to the packs and drove up his father-in-law women.

Some (vivid) examples

PANDARUS:

Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:

Come, sir, I will make did behold your worship.

VIOLA:

I'll drink it.

VIOLA:

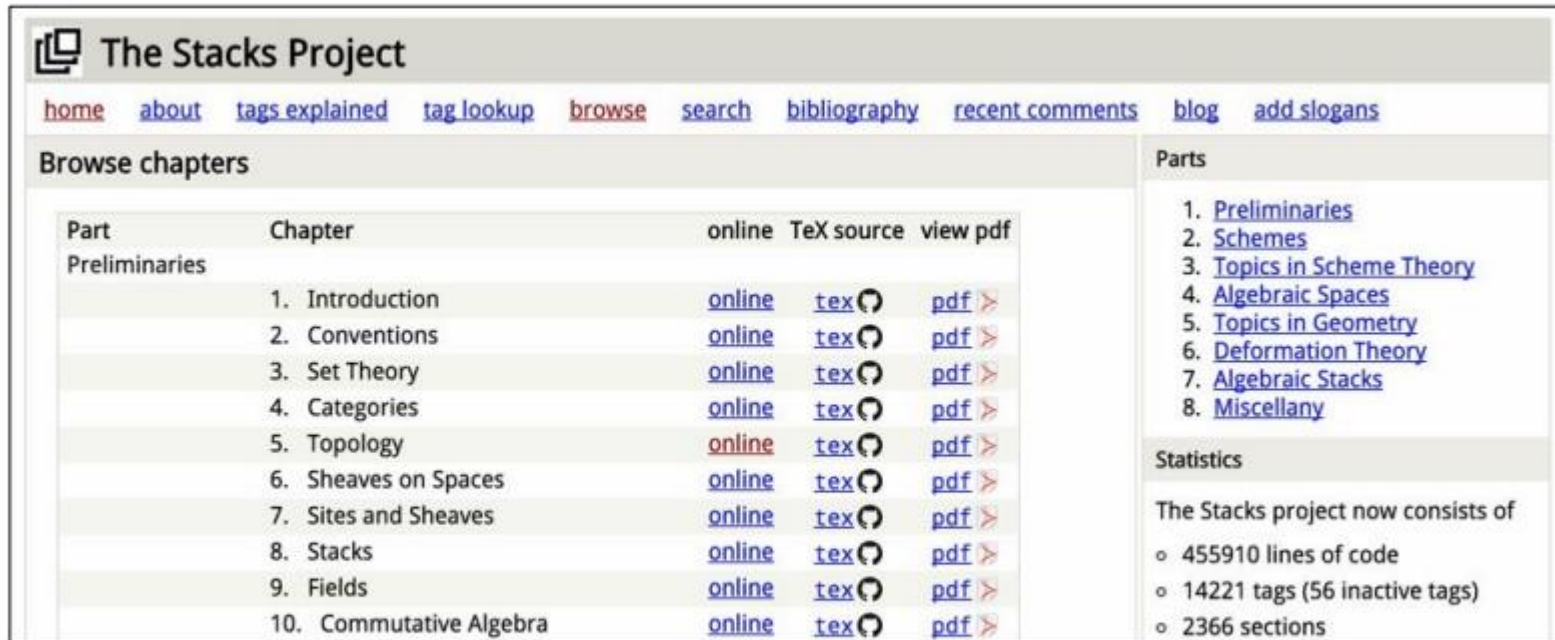
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:

O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

Some (vivid) examples

The Stacks Project: open source algebraic geometry textbook



The screenshot shows the homepage of The Stacks Project. At the top is a navigation bar with links: [home](#), [about](#), [tags explained](#), [tag lookup](#), [browse](#), [search](#), [bibliography](#), [recent comments](#), [blog](#), and [add slogans](#). Below this is a section titled "Browse chapters" which contains a table with columns: Part, Chapter, online, TeX source, and view pdf. The table lists 10 chapters under the "Preliminaries" part. To the right of the table is a section titled "Parts" which lists 8 parts: 1. Preliminaries, 2. Schemes, 3. Topics in Scheme Theory, 4. Algebraic Spaces, 5. Topics in Geometry, 6. Deformation Theory, 7. Algebraic Stacks, and 8. Miscellany. Below the "Parts" section is a "Statistics" section which states: "The Stacks project now consists of" followed by a list: 455910 lines of code, 14221 tags (56 inactive tags), and 2366 sections.

Part	Chapter	online	TeX source	view pdf
Preliminaries	1. Introduction	online	tex	pdf
	2. Conventions	online	tex	pdf
	3. Set Theory	online	tex	pdf
	4. Categories	online	tex	pdf
	5. Topology	online	tex	pdf
	6. Sheaves on Spaces	online	tex	pdf
	7. Sites and Sheaves	online	tex	pdf
	8. Stacks	online	tex	pdf
	9. Fields	online	tex	pdf
	10. Commutative Algebra	online	tex	pdf

Parts

1. [Preliminaries](#)
2. [Schemes](#)
3. [Topics in Scheme Theory](#)
4. [Algebraic Spaces](#)
5. [Topics in Geometry](#)
6. [Deformation Theory](#)
7. [Algebraic Stacks](#)
8. [Miscellany](#)

Statistics

The Stacks project now consists of

- 455910 lines of code
- 14221 tags (56 inactive tags)
- 2366 sections

Latex source

<http://stacks.math.columbia.edu/>

The stacks project is licensed under the [GNU Free Documentation License](#)

Some (vivid) examples

For $\bigoplus_{n=1, \dots, m} \mathcal{L}_{m,n} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ???. Hence we obtain a scheme S and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of \mathcal{X}' , and \mathcal{T}_i is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}^{\text{opp}}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \mapsto (U, \text{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

Proof. See discussion of sheaves of sets. \square

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{\text{spaces}, \text{étale}}$ which gives an open subspace of X and T equal to S_{Zar} , see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S .

Lemma 0.1. Assume (3) and (3) by the construction in the description.

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\text{Proj}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X, \mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S . Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) f is locally of finite type. Since $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X . But given a scheme U and a surjective étale morphism $U \rightarrow X$. Let $U \cap U = \coprod_{i=1, \dots, n} U_i$ be the scheme X over S at the schemes $X_i \rightarrow X$ and $U = \lim_i X_i$. \square

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X, \dots, 0}$.

Lemma 0.2. Let X be a locally Noetherian scheme over S , $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \bar{A}_2$ works.

Lemma 0.3. In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.

Proof. We will use the property we see that \mathfrak{p} is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_X) = \mathcal{O}_X(D)$$

where K is an F -algebra where δ_{n+1} is a scheme over S . \square

Some (vivid) examples

OpenAI GPT-2 generated text

[source](#)

Basically the same principle, but uses a different type of model that we'll learn about later

Input: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Output: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Summary

- Recurrent neural networks (RNNs): neural networks that can process variable-length inputs and outputs
 - Could think of them as networks with an input & output at each layer
 - Variable depth
 - Depth = time
- Training RNNs is very hard
 - Vanishing and exploding gradients
 - Can use special cells (LSTM, GRU)
 - Generally need to spend more time tuning hyperparameters
- In practice, we almost always have both inputs **and** outputs at each step
 - This is because we usually want structured prediction
 - Can use scheduled sampling to handle distributional shift
- Many variants for various purposes
 - Sequence to sequence models (more on this later)
 - Bidirectional models
 - Can even “RNN-ify” convolutional layers!

UC Berkeley · CSW182 | [Deep Learning]

Designing, Visualizing and Understanding Deep Neural Networks (2021)

CSW182 (2021) · 课程资料包 @ShowMeAI



视频
中英双语字幕



课件
一键打包下载



笔记
官方笔记翻译



代码
作业项目解析



视频 · B 站 [扫码或点击链接]
<https://www.bilibili.com/video/BV1Ff4y1n7ar>



课件 & 代码 · 博客 [扫码或点击链接]
<http://blog.showmeai.tech/berkeley-csw182>

Berkeley
Q-Learning
计算机视觉

循环神经网络
风格迁移
机器学习基础

可视化
模仿学习
生成模型

梯度策略
元学习
卷积网络

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2：扫码点击**底部菜单栏**
称为 **AI 内容创作者**？回复 [添砖加瓦]