max planck institut
informatik

SIC Saarland Informatics Campus

High Level Computer Vision

Recurrent Neural Networks @ June 5, 2019

Bernt Schiele & Mario Fritz

www.mpi-inf.mpg.de/hlcv/

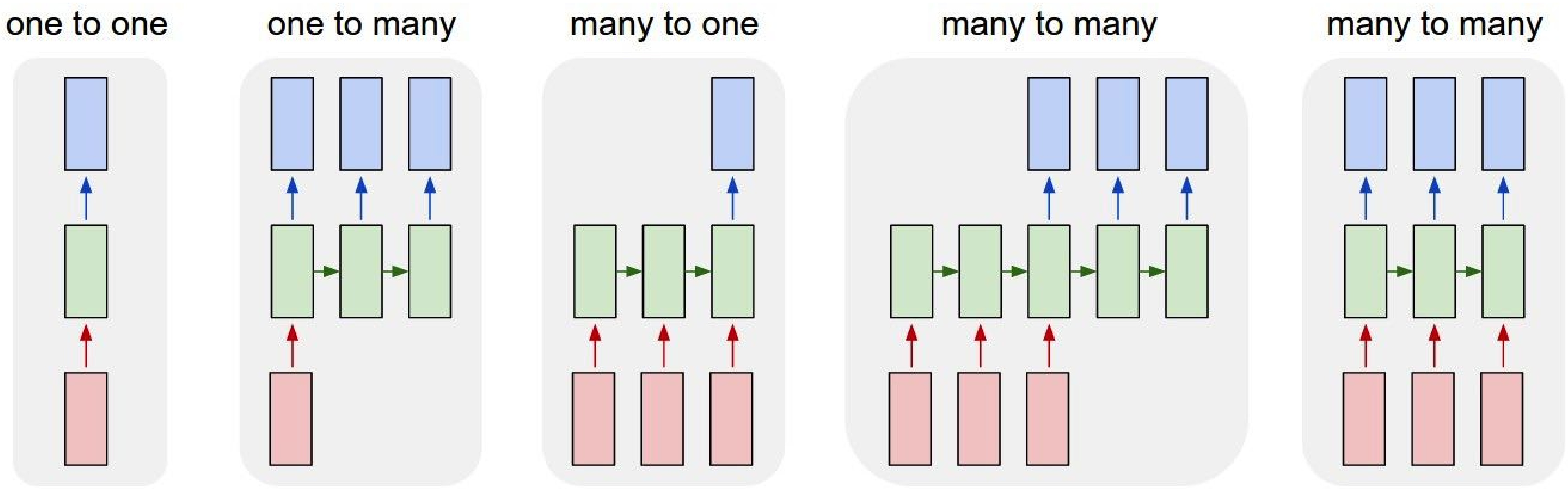
Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken

Overview Today's Lecture

- Recurrent Neural Networks (RNNs)
  - Motivation & flexibility of RNNs (some recap from last week)
  - Language modeling
    - including "unreasonable effectiveness of RNNs"
  - RNNs for image description / captioning
  - Standard RNN and a particularly successful RNN: Long Short Term Memory (LSTM)
    - including "visualizations of RNN cells"

slide credit: Andrej Karpathy

Recurrent Networks offer a lot of flexibility:



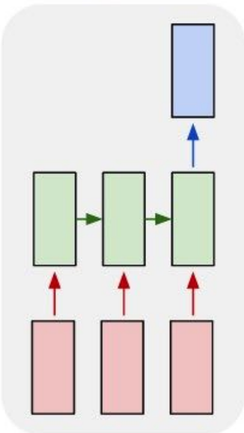
slide credit: Jeff Donahue

Sequences in Vision

Sequences in the input…
(many-to-one)

- [ ] Running
- [x] Jumping
- [ ] Dancing
- [ ] Fighting
- [ ] Eating

many to one

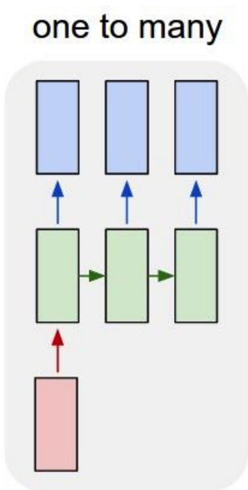


slide credit: Jeff Donahue

Sequences in Vision

Sequences in the output…
(one-to-many)

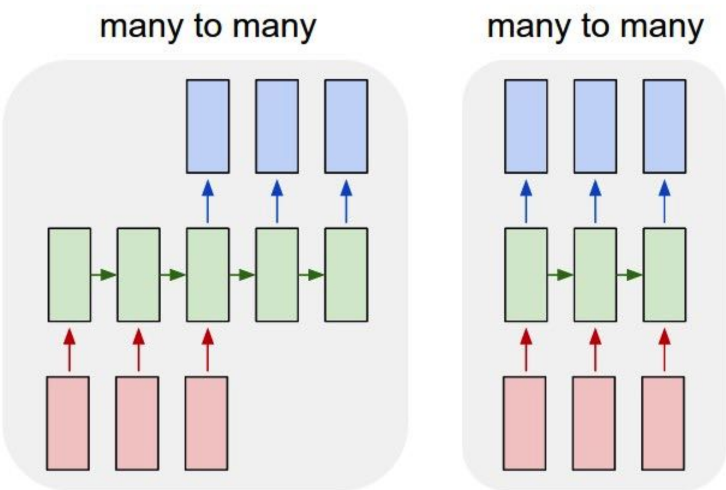
A happy brown dog.



slide credit: Jeff Donahue

Sequences in Vision

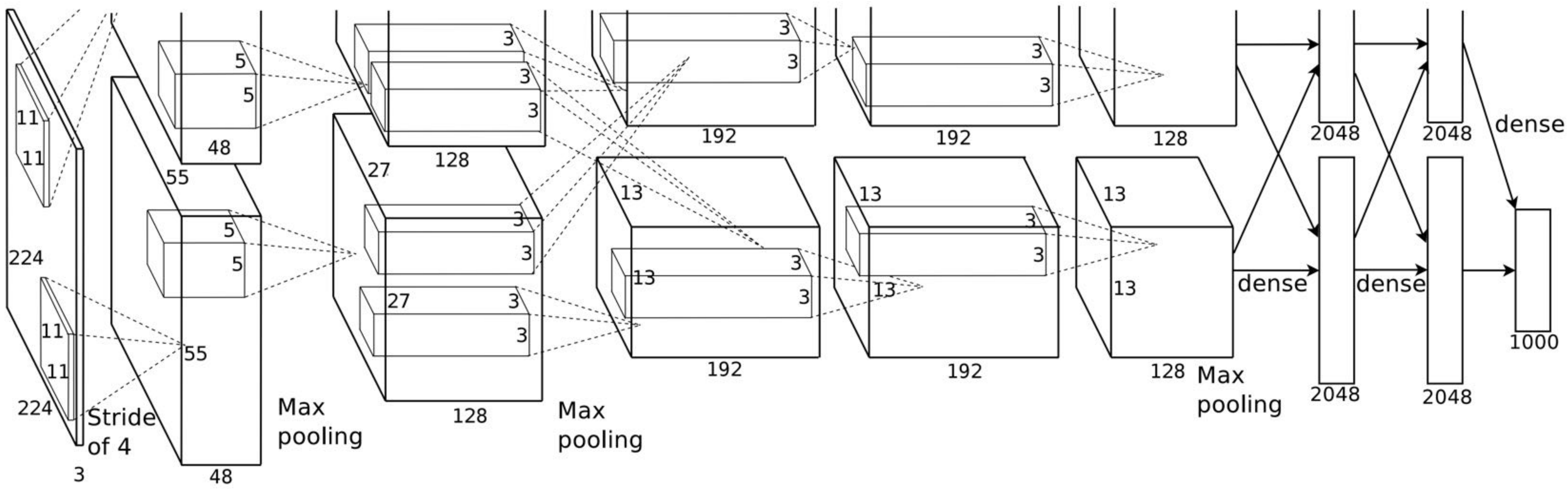
Sequences everywhere!
(many-to-many)



A dog jumps over a hurdle.

slide credit: Jeff Donahue

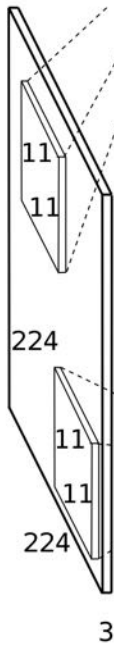
ConvNets



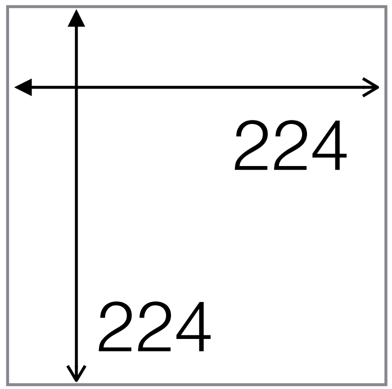
Krizhevsky et al., NIPS 2012

slide credit: Jeff Donahue

Problem #1

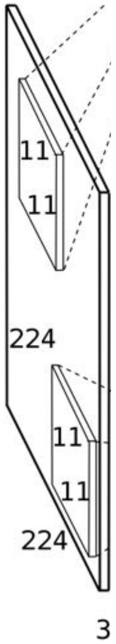


fixed-size, static input

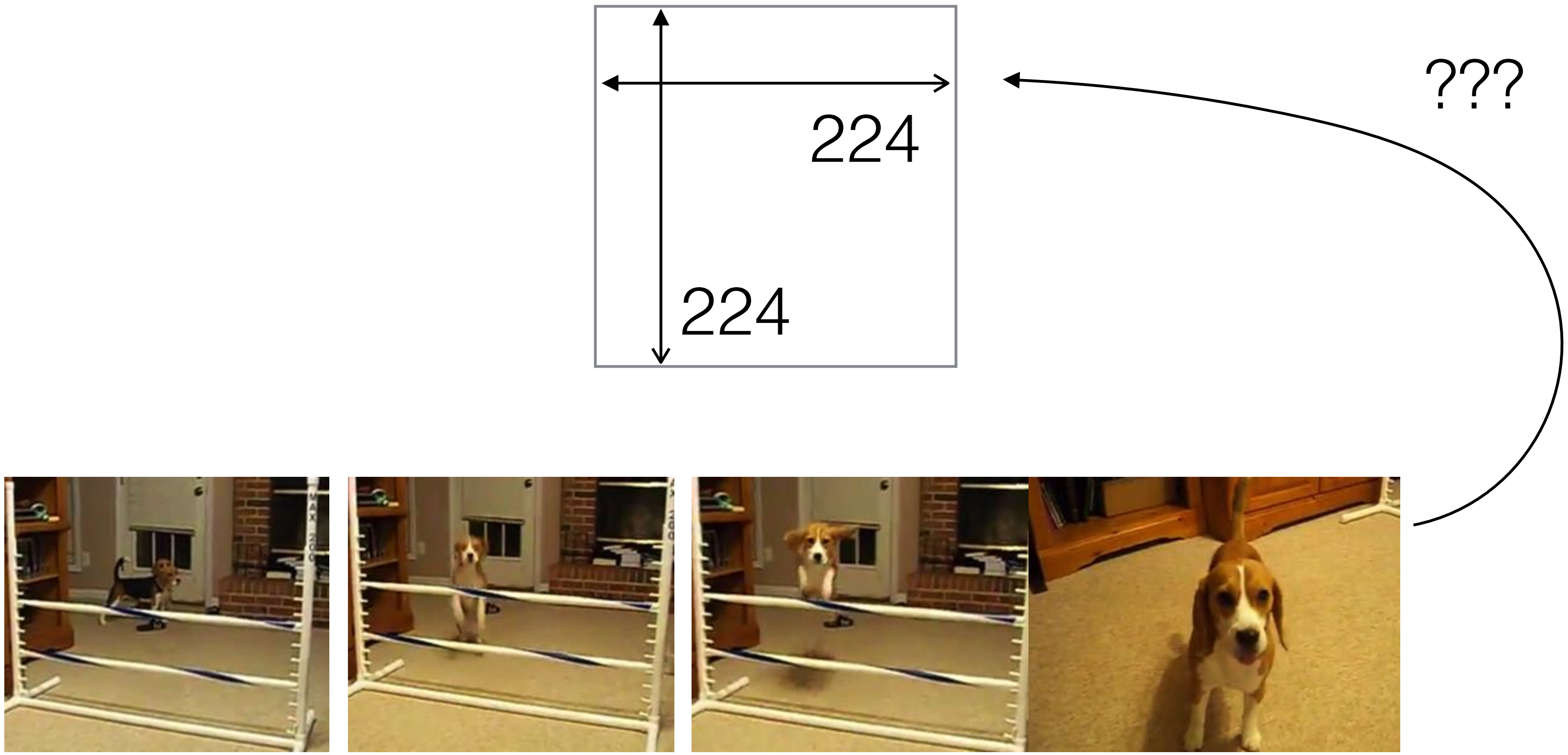


slide credit: Jeff Donahue

Problem #1

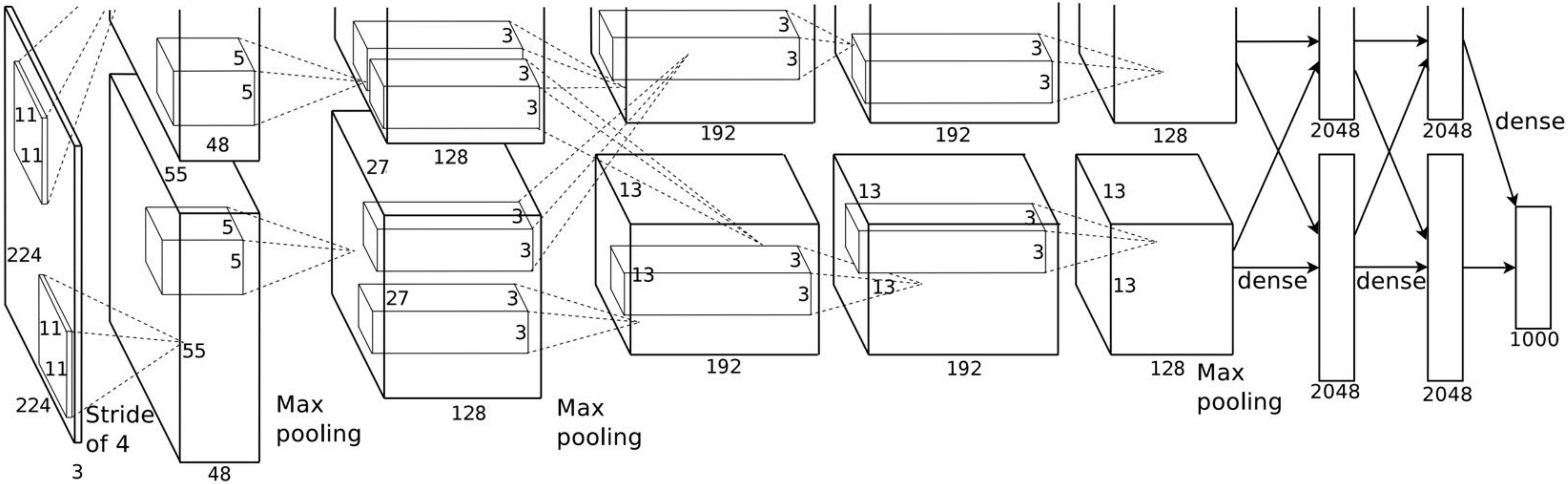


fixed-size, static input



slide credit: Jeff Donahue

Problem #2

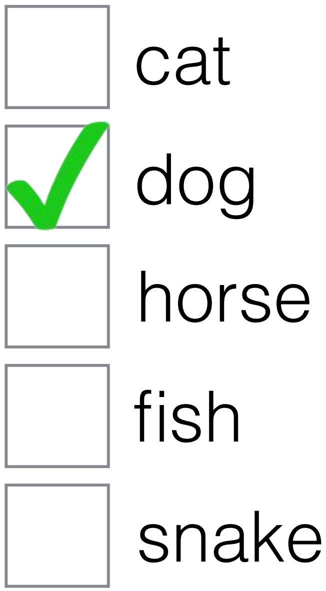


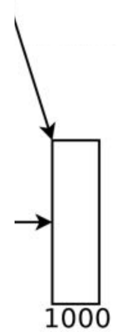
Krizhevsky et al., NIPS 2012

slide credit: Jeff Donahue

Problem #2

output is a single choice from a fixed list of options



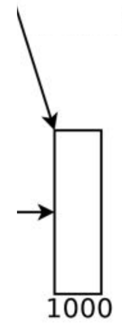


slide credit: Jeff Donahue

Problem #2

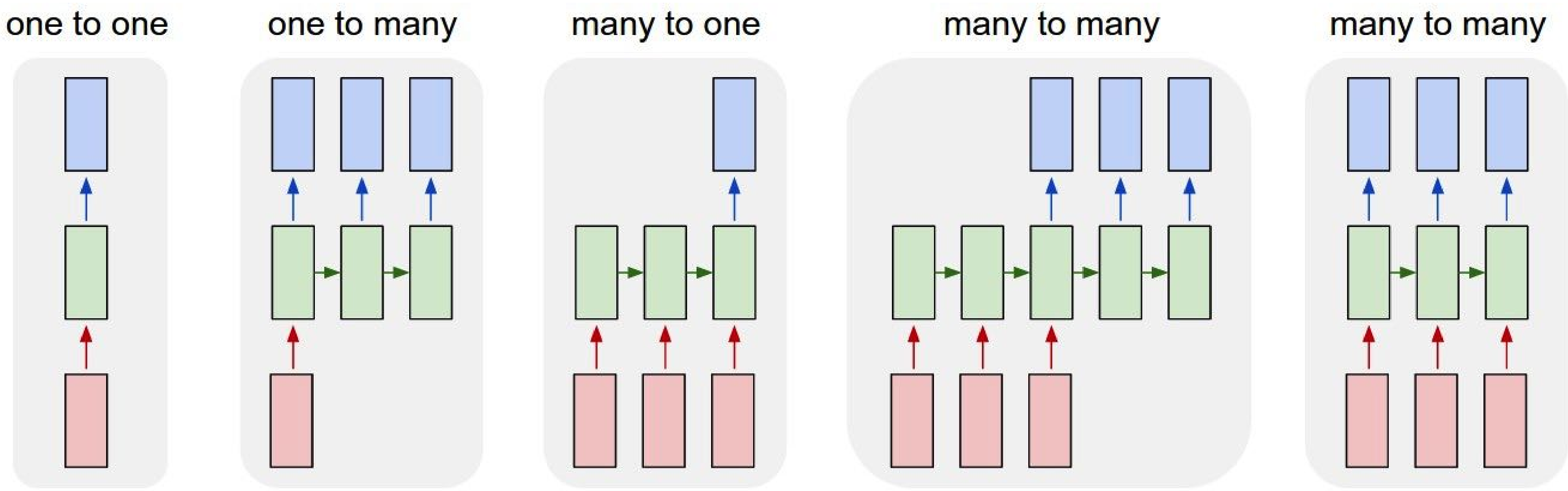
output is a single choice from a fixed list of options

- [ ] a happy brown dog
- [x] a big brown dog
- [ ] a happy red dog
- [ ] a big red dog
- [ ] …

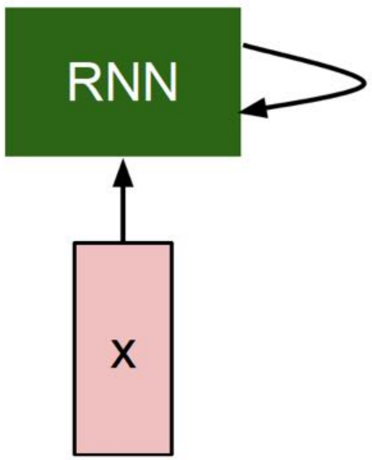


slide credit: Andrej Karpathy

Recurrent Networks offer a lot of flexibility:

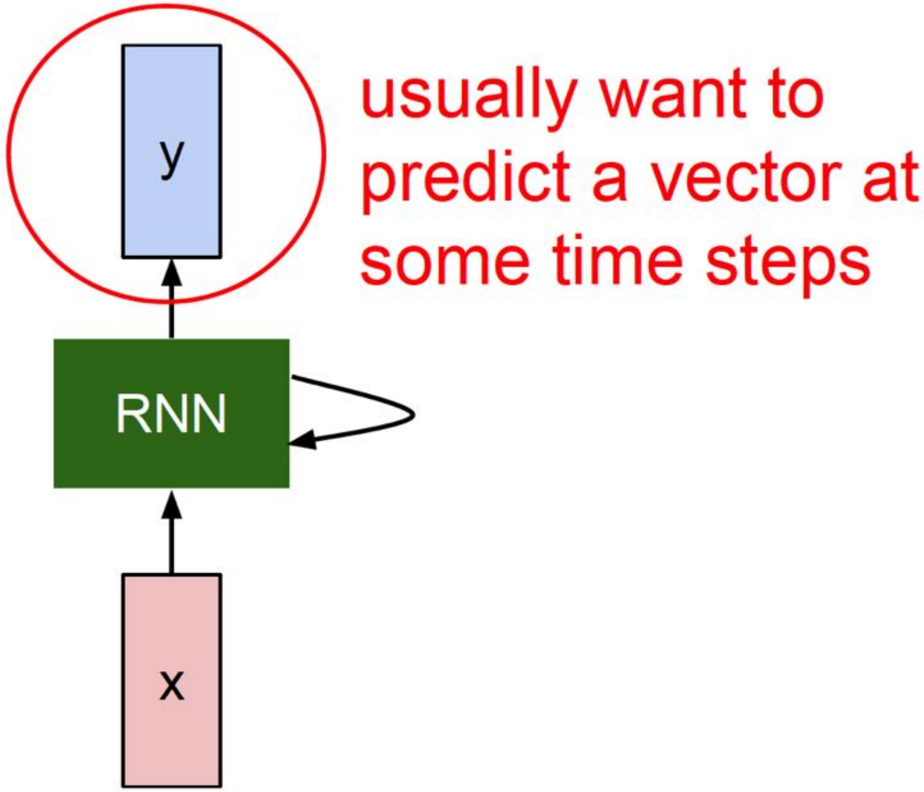


Recurrent Neural Network (RNN)



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Recurrent Neural Network (RNN)



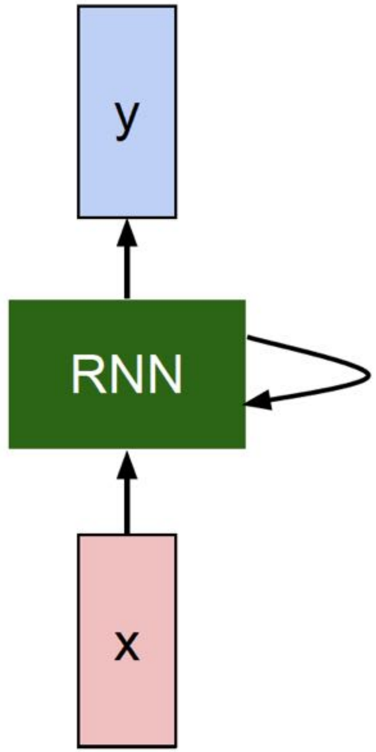
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Recurrent Neural Network (RNN)

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state — some function with parameters W — old state — input vector at some time step



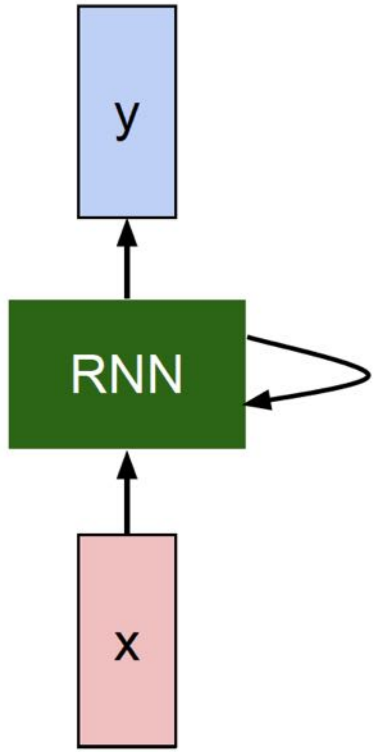
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Recurrent Neural Network (RNN)

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

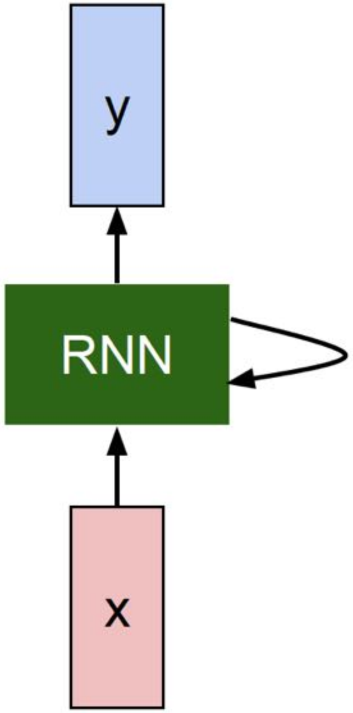
Notice: the same function and the same set of parameters are used at every time step.



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

(Simple) Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:



$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

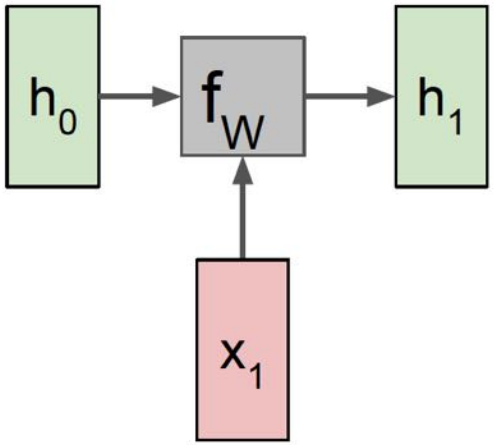
$$y_t = W_{hy}h_t$$

Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

RNN: Computational Graph

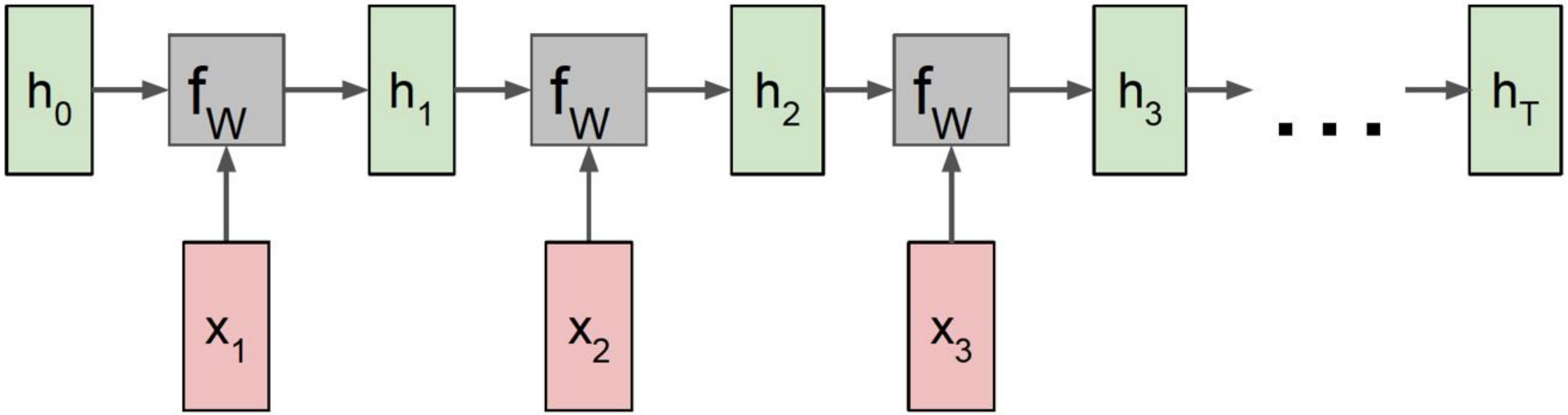
$$h_t = f_W(h_{t-1}, x_t)$$



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

RNN: Computational Graph

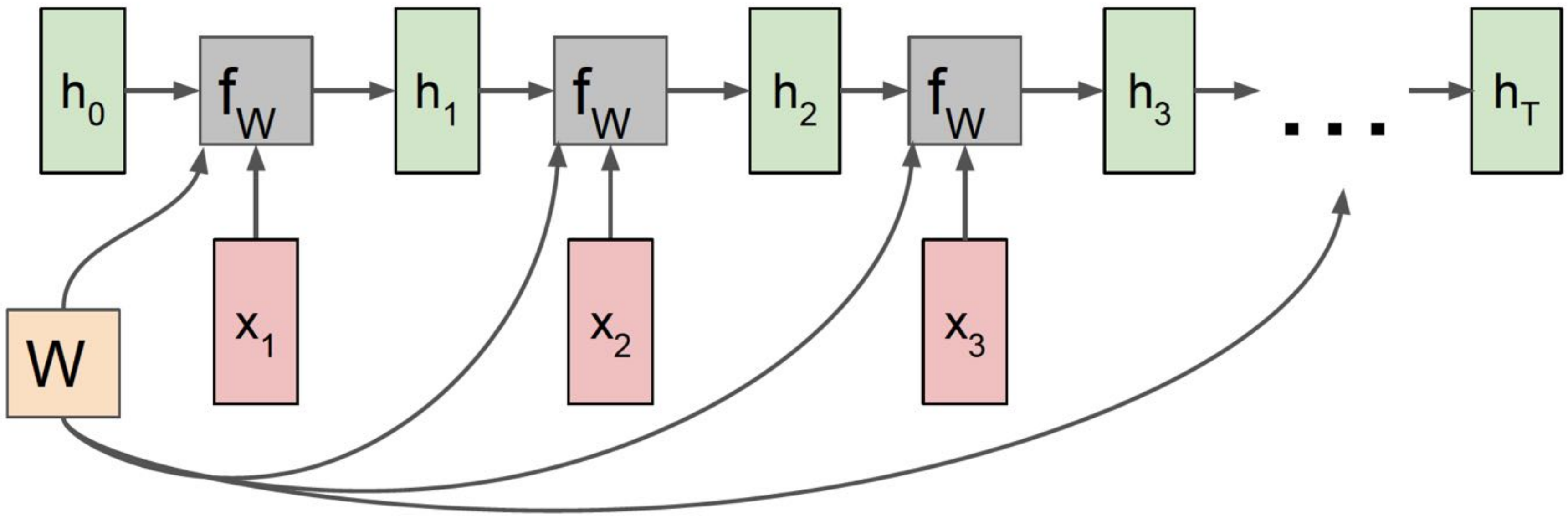
$$h_t = f_W(h_{t-1}, x_t)$$



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

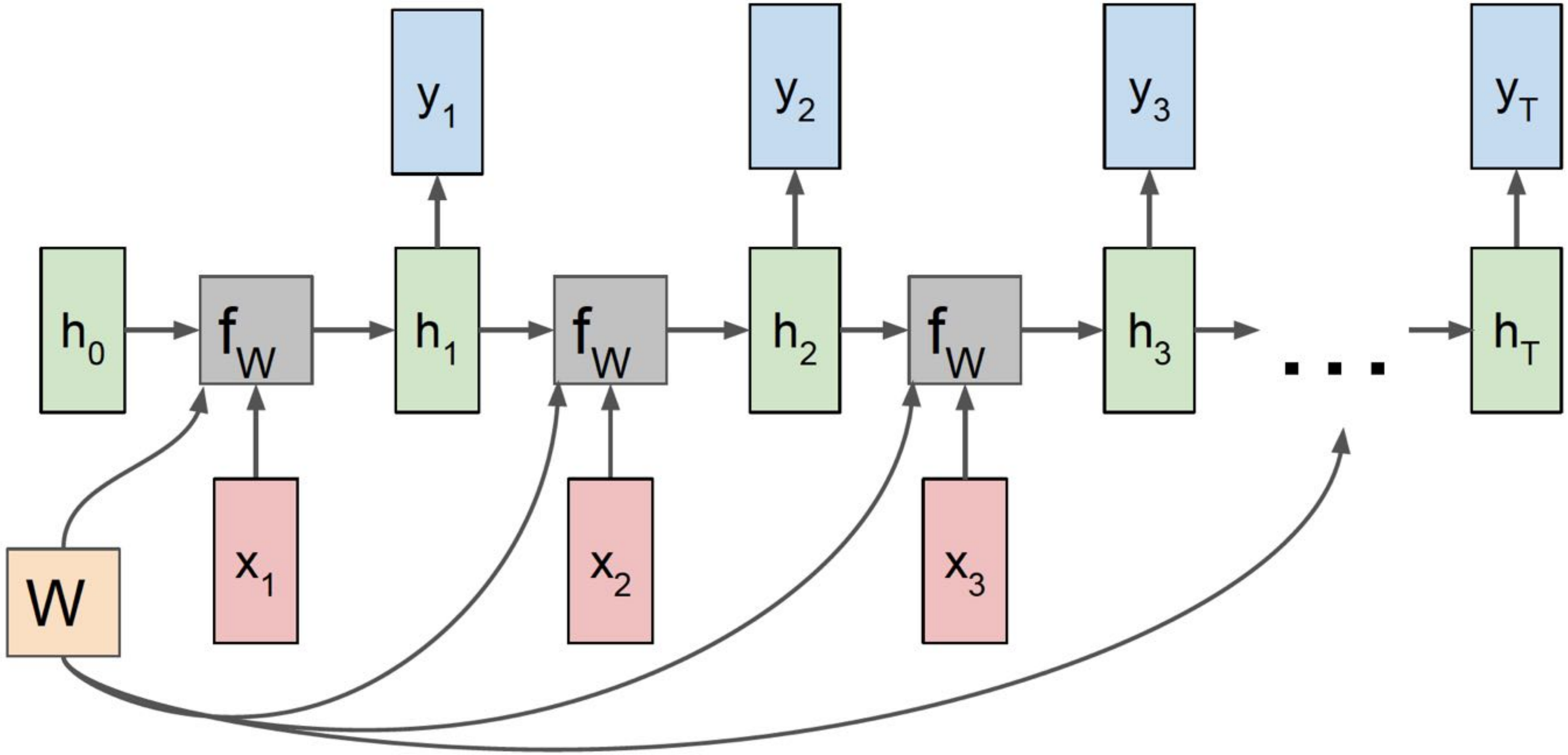
RNN: Computational Graph

Re-use the same weight matrix at every time-step



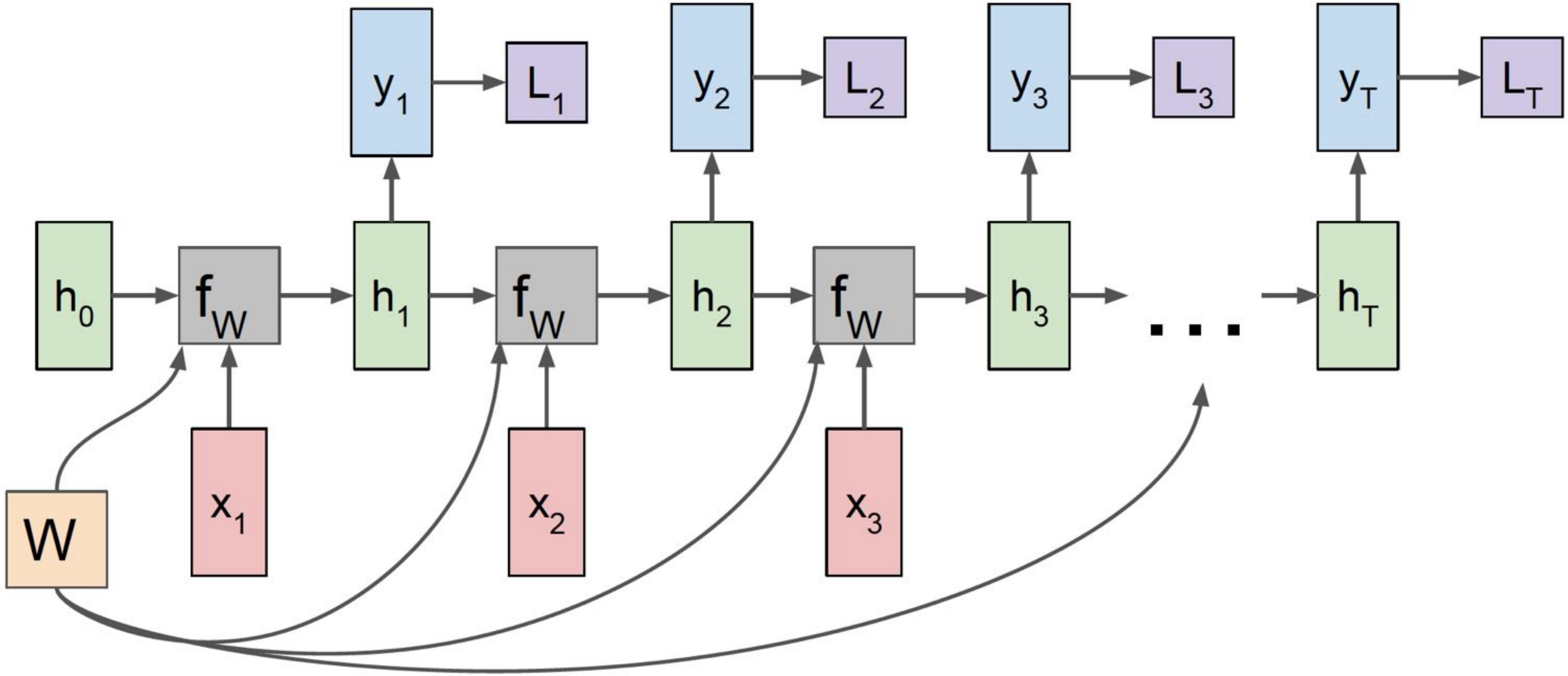
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

RNN: Computational Graph: Many to Many



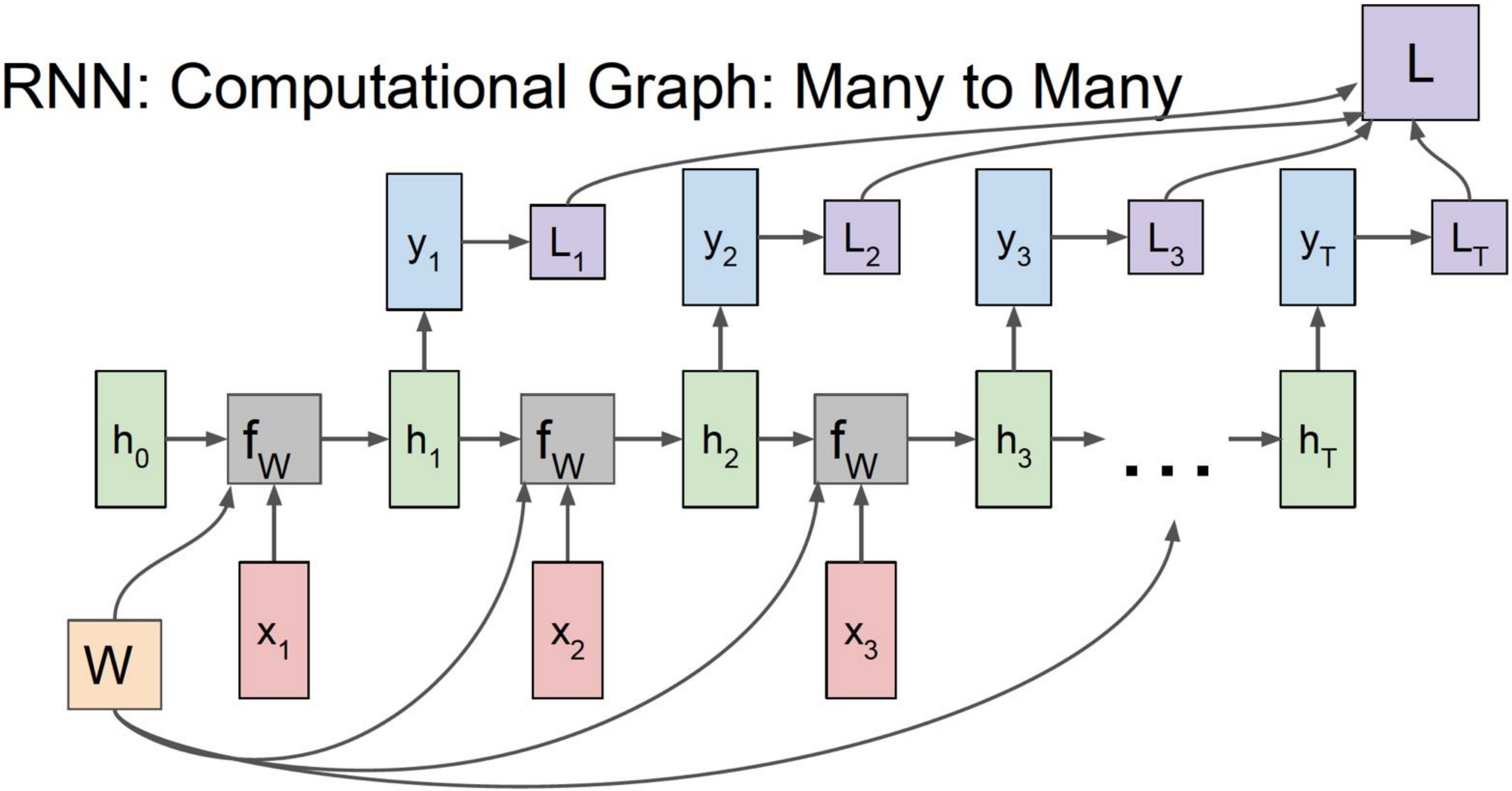
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

RNN: Computational Graph: Many to Many



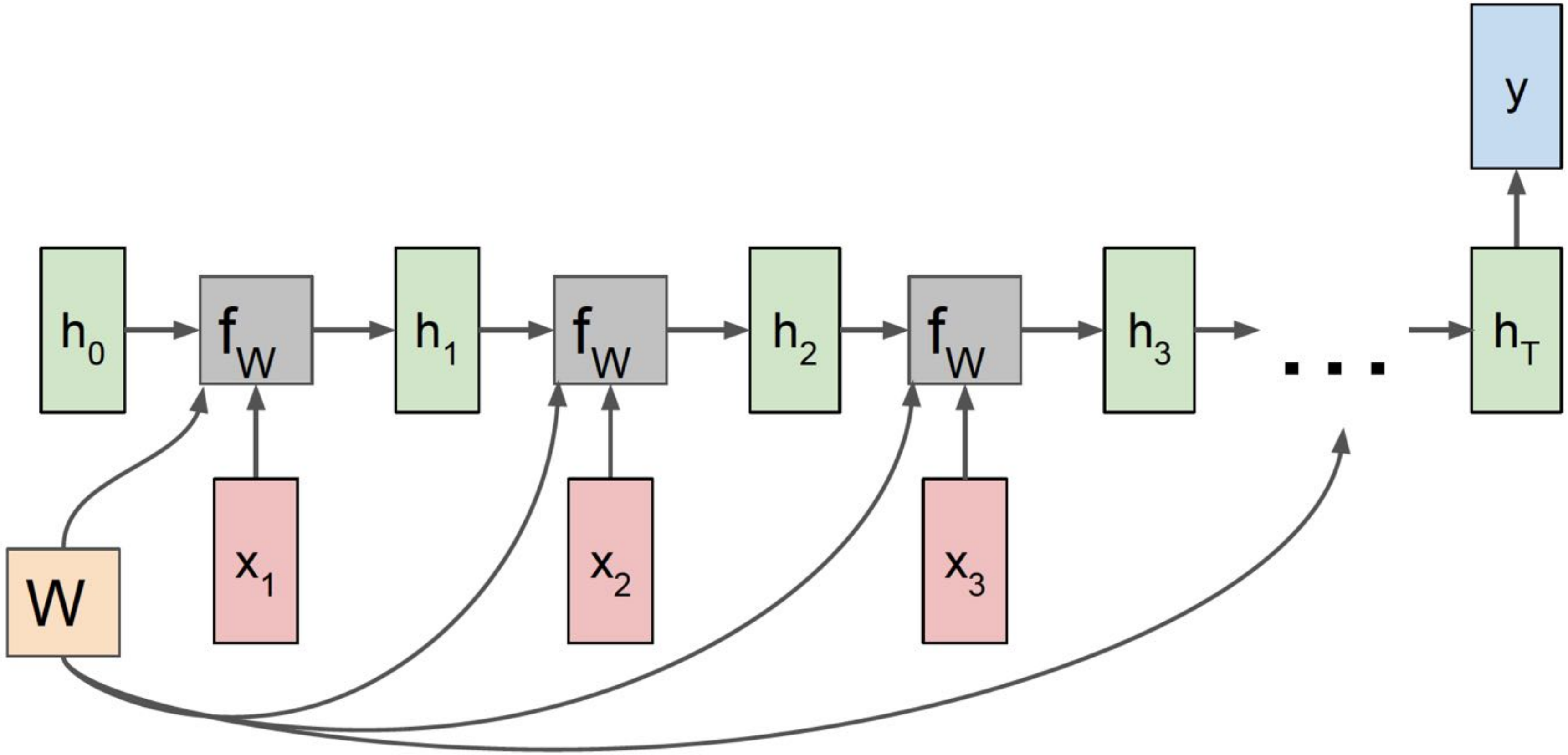
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

RNN: Computational Graph: Many to Many



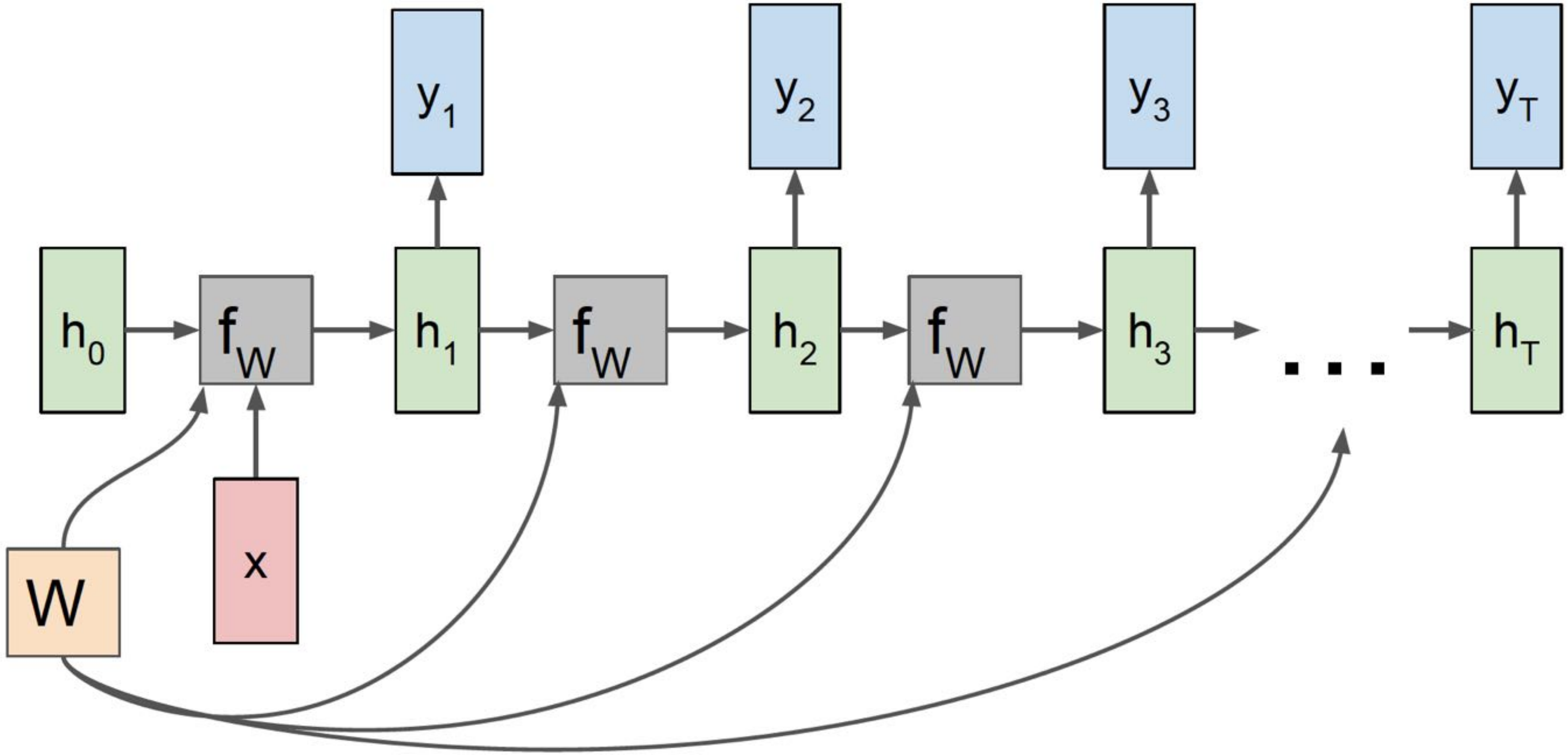
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

RNN: Computational Graph: Many to One



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

RNN: Computational Graph: One to Many

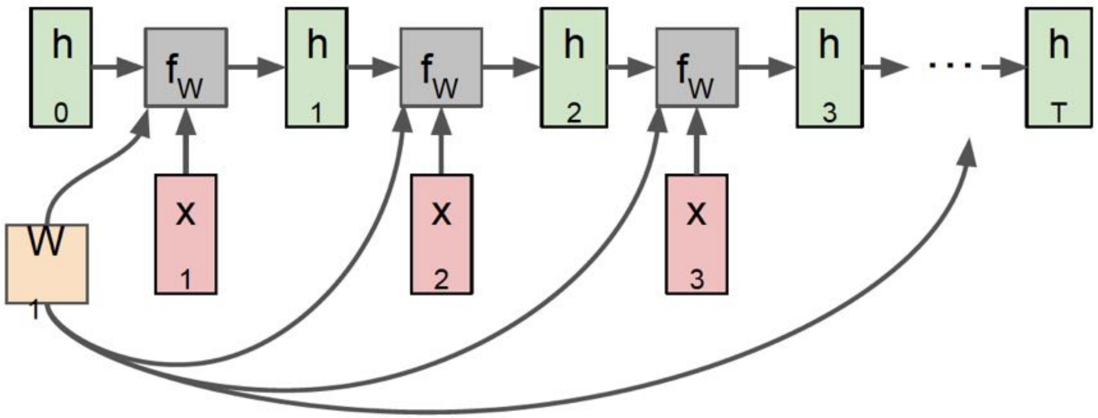


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Sequence to Sequence

Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector



Sutskever et al, “Sequence to Sequence Learning with Neural Networks”, NIPS 2014

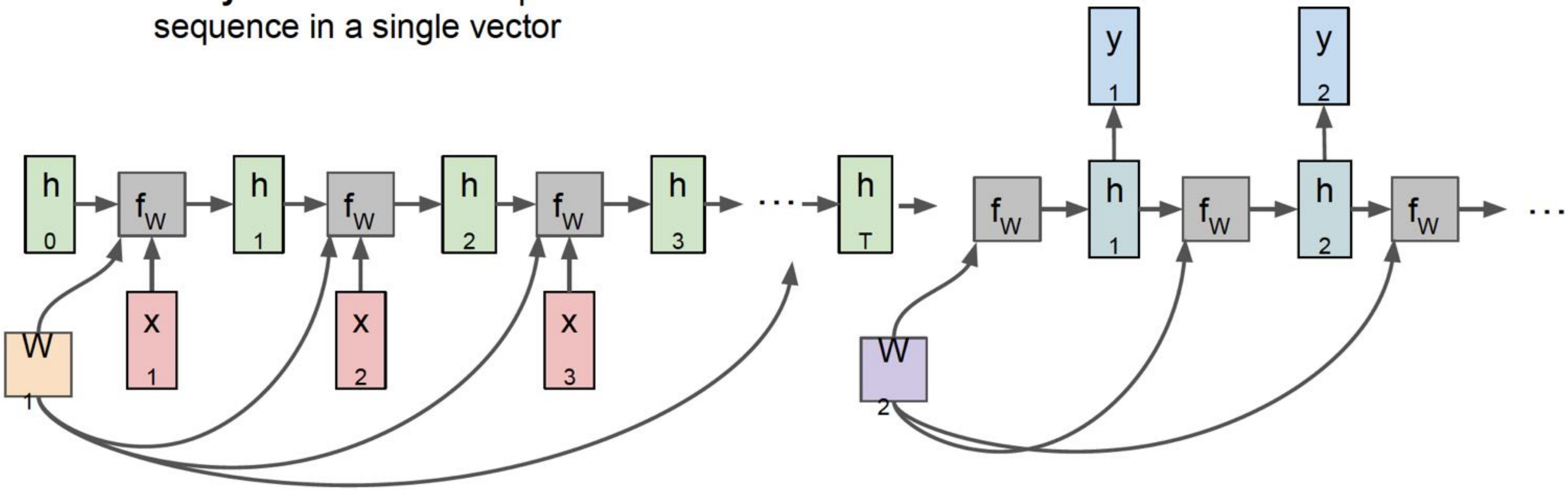
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Sequence to Sequence

Sequence to Sequence: Many-to-one + one-to-many

One to many: Produce output sequence from single input vector

Many to one: Encode input sequence in a single vector

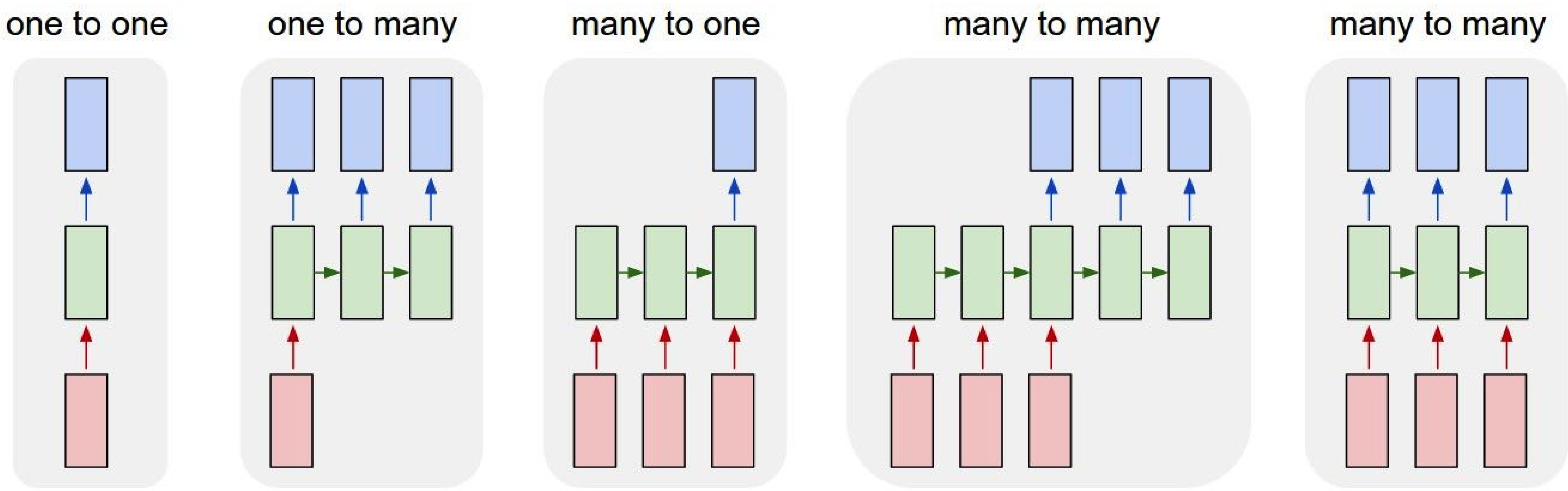


Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

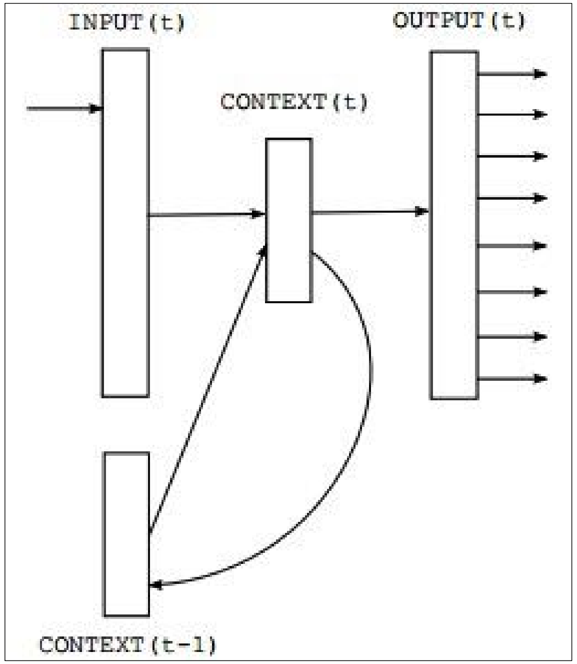
slide credit: Andrej Karpathy

Recurrent Networks offer a lot of flexibility:

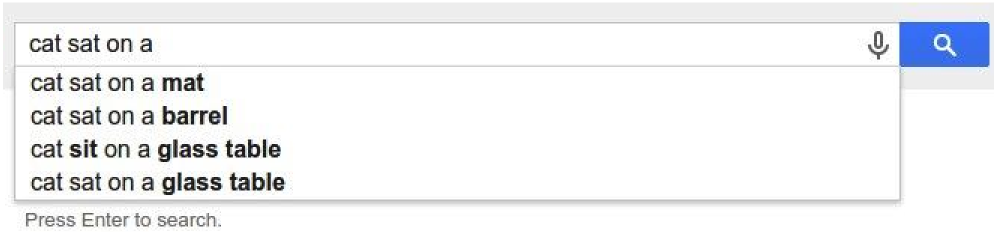


slide credit: Andrej Karpathy

Language Models



Word-level language model. Similar to:



Recurrent Neural Network Based Language Model
[Tomas Mikolov, 2010]

slide credit: Andrej Karpathy

Suppose we had the training sentence “cat sat on mat”

We want to train a **language model**:
P(next word | previous words)

i.e. want these to be high:
P(cat | [<S>])
P(sat | [<S>, cat])
P(on | [<S>, cat, sat])
P(mat | [<S>, cat, sat, on])
P(<E>| [<S>, cat, sat, on, mat])

slide credit: Andrej Karpathy

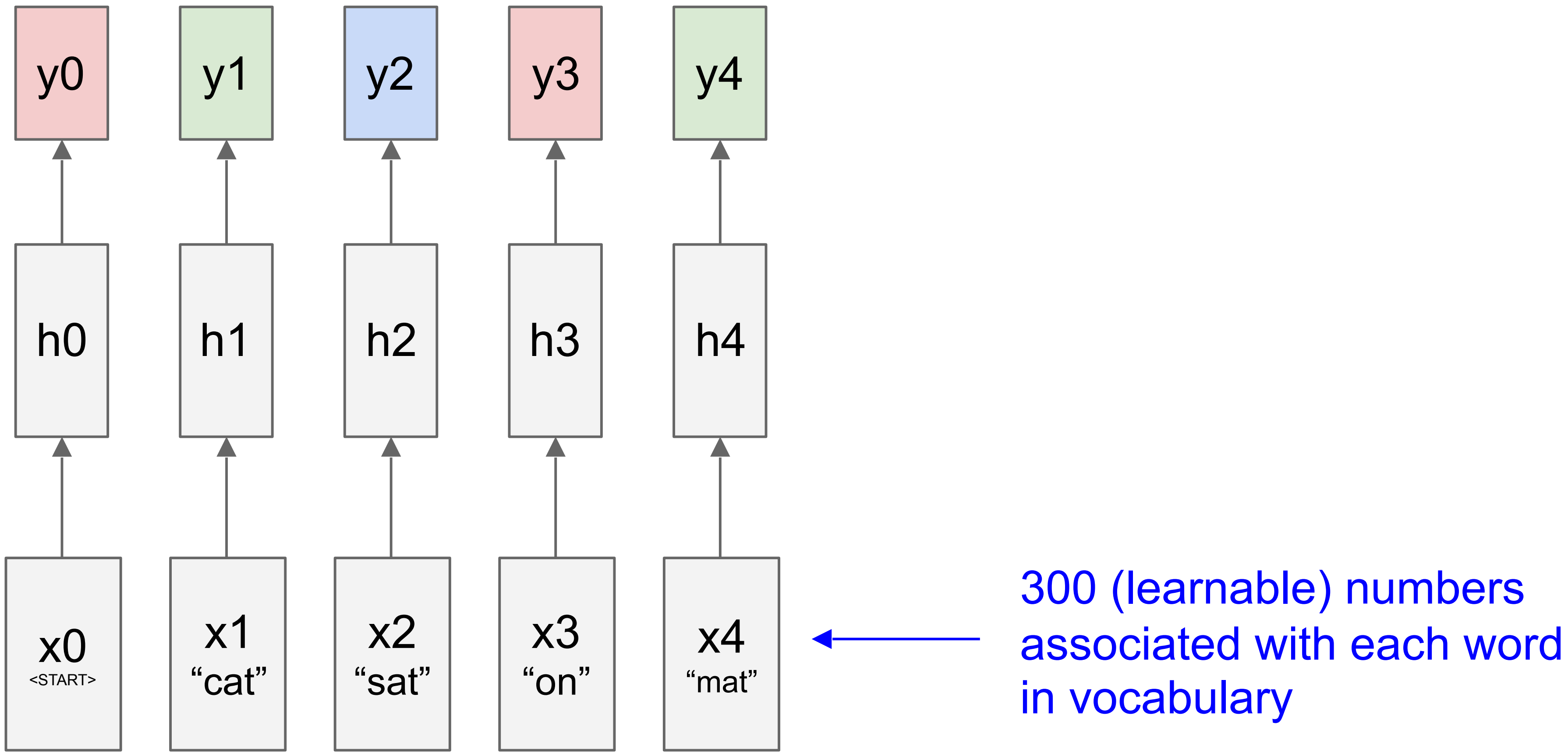
Suppose we had the training sentence "cat sat on mat"

We want to train a **language model**:
P(next word | previous words)

First, suppose we had only a finite, 1-word history:
i.e. want these to be high:
P(cat | <S>)
P(sat | cat)
P(on | sat)
P(mat | on)
P(<E>| mat)

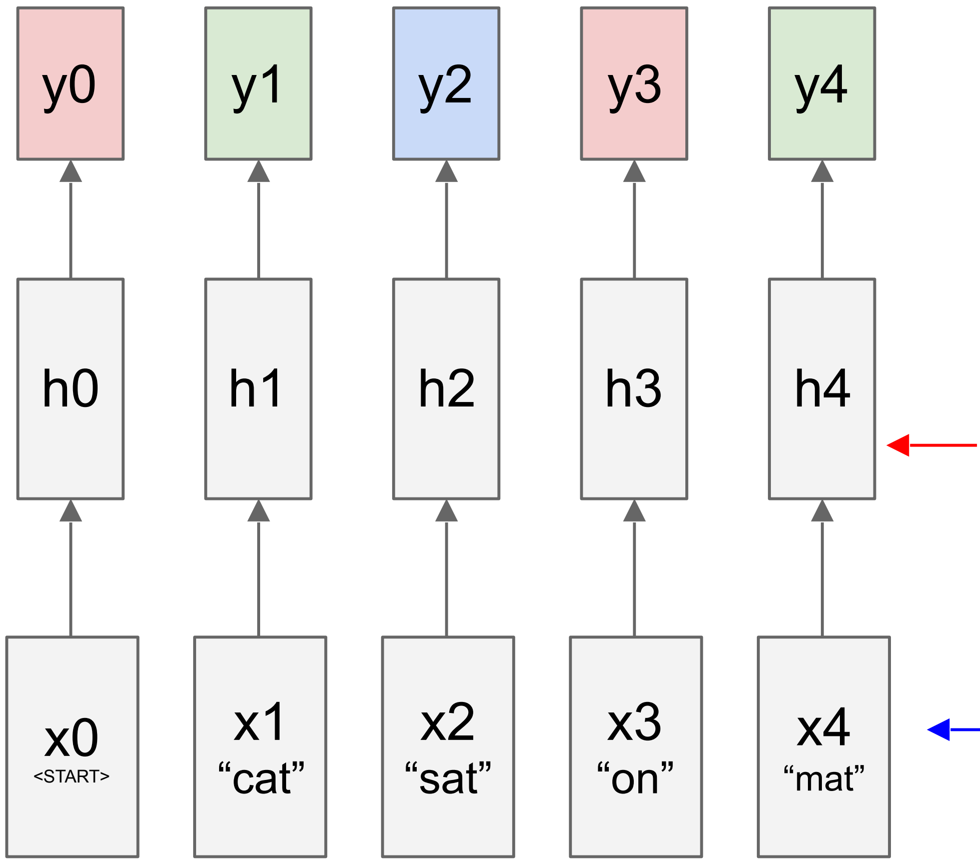
slide credit: Andrej Karpathy

"cat sat on mat"



slide credit: Andrej Karpathy

“cat sat on mat”

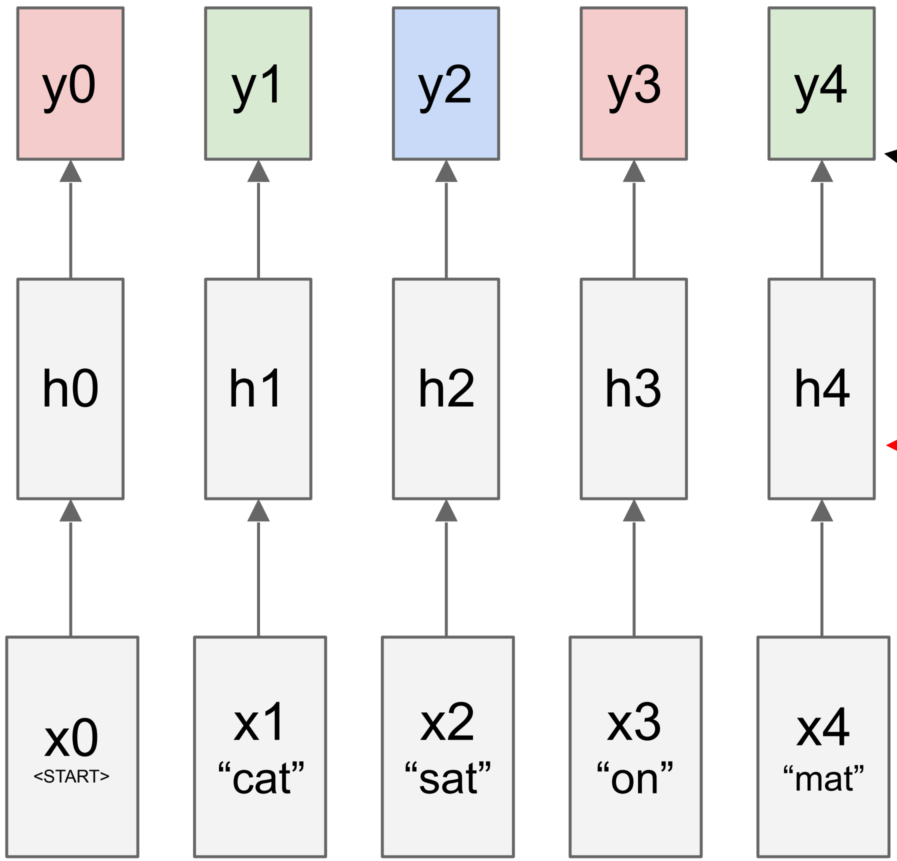


hidden layer (e.g. 500-D vectors)
h4 = tanh(0, Wxh * x4)

300 (learnable) numbers associated with each word in vocabulary

slide credit: Andrej Karpathy

“cat sat on mat”



10,001-D class scores:
Softmax over 10,000 words and
a special <END> token.
y4 = Why * h4

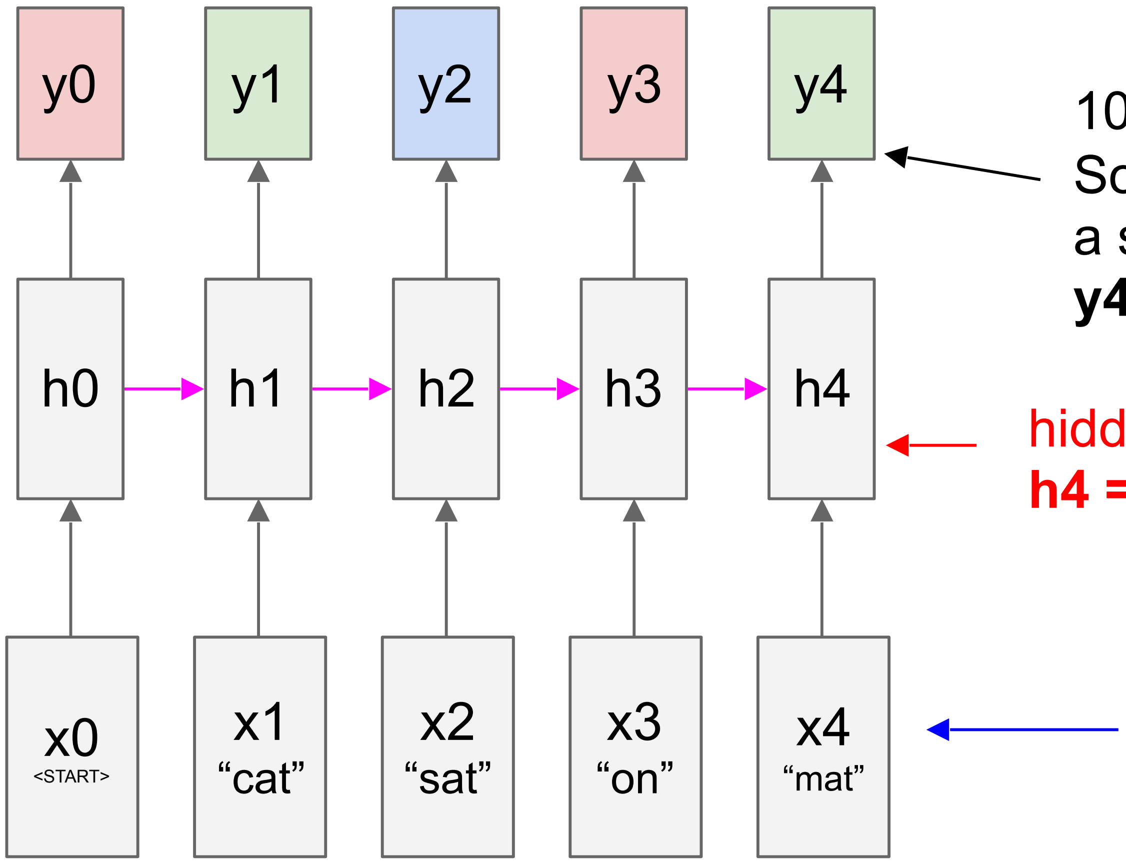
hidden layer (e.g. 500-D vectors)
h4 = tanh(0, Wxh * x4)

300 (learnable) numbers
associated with each word
in vocabulary

slide credit: Andrej Karpathy

Recurrent Neural Network:

"cat sat on mat"



10,001-D class scores:
Softmax over 10,000 words and a special <END> token.
y4 = Why * h4

hidden layer (e.g. 500-D vectors)
h4 = tanh(0, Wxh * x4 + Whh * h3)

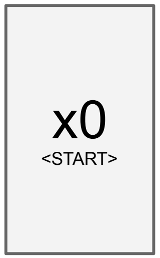
300 (learnable) numbers associated with each word in vocabulary

slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

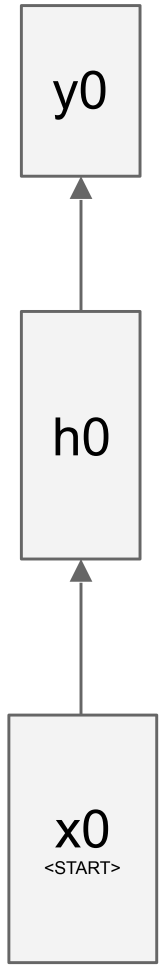


slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

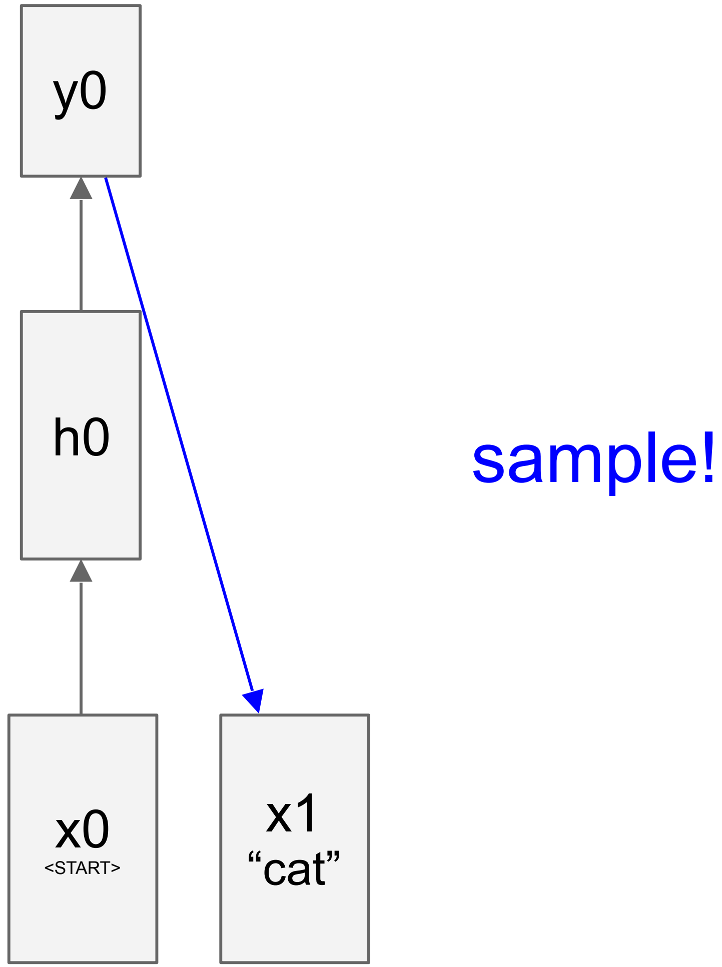


slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

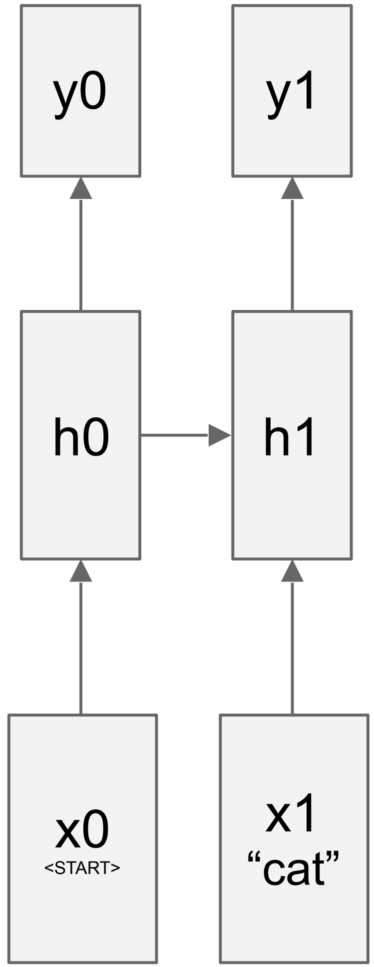


slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

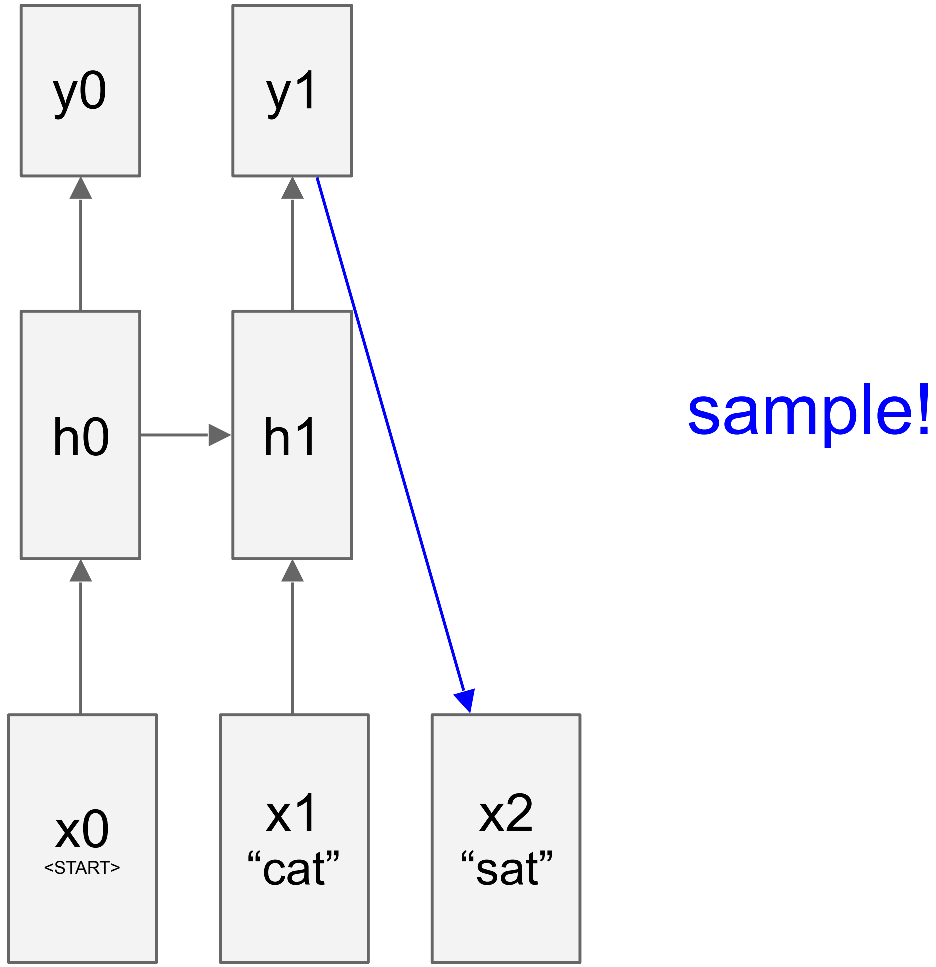
P(next word | previous words)



Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

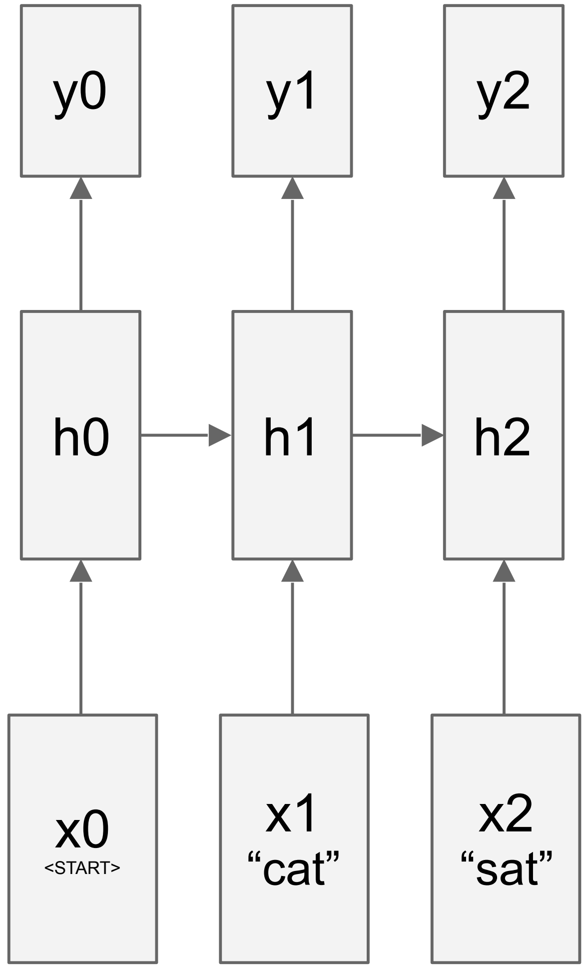


slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

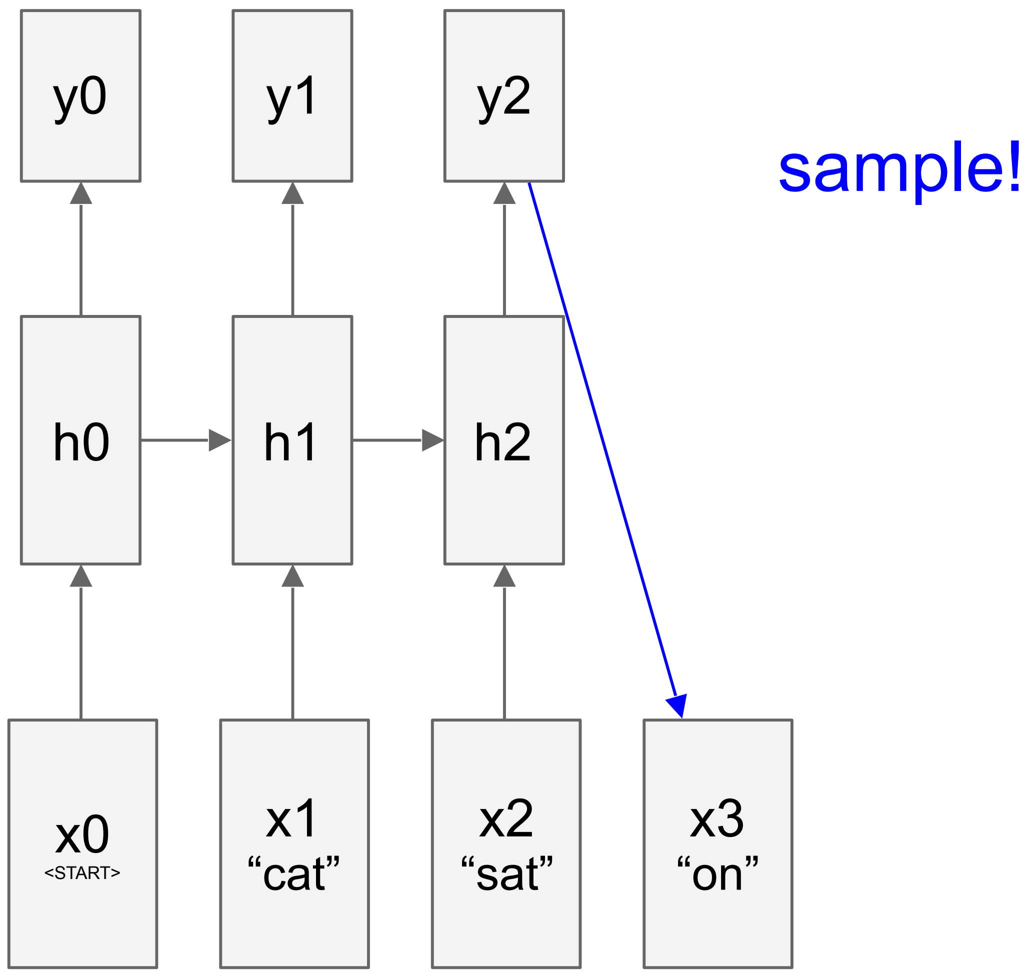


slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

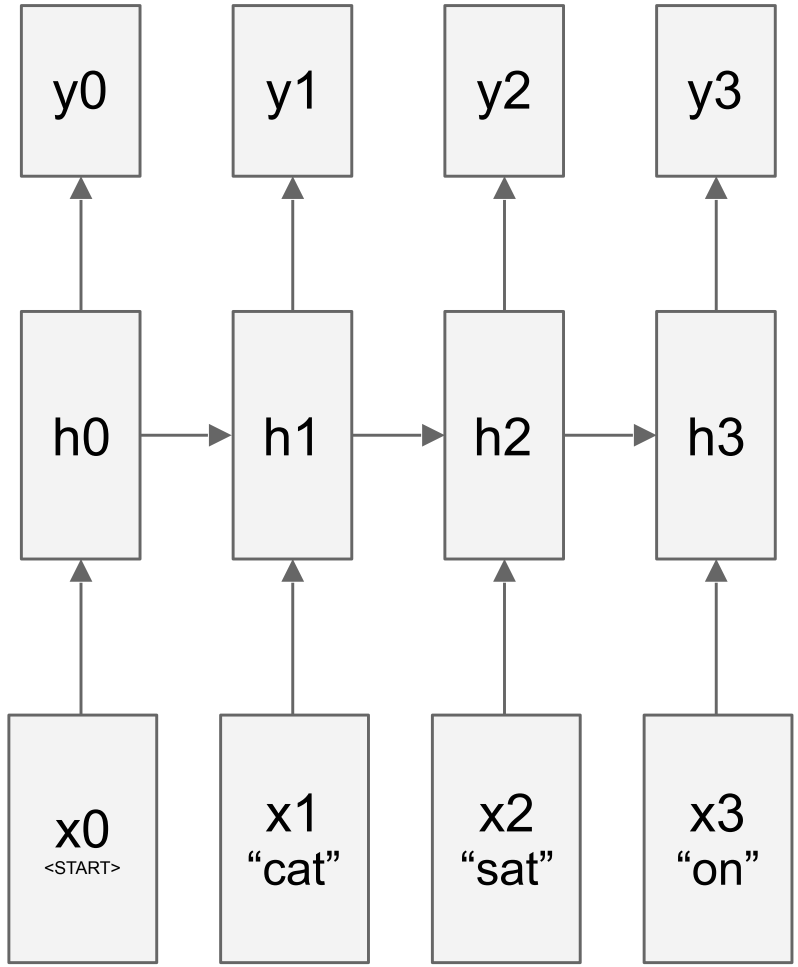


slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

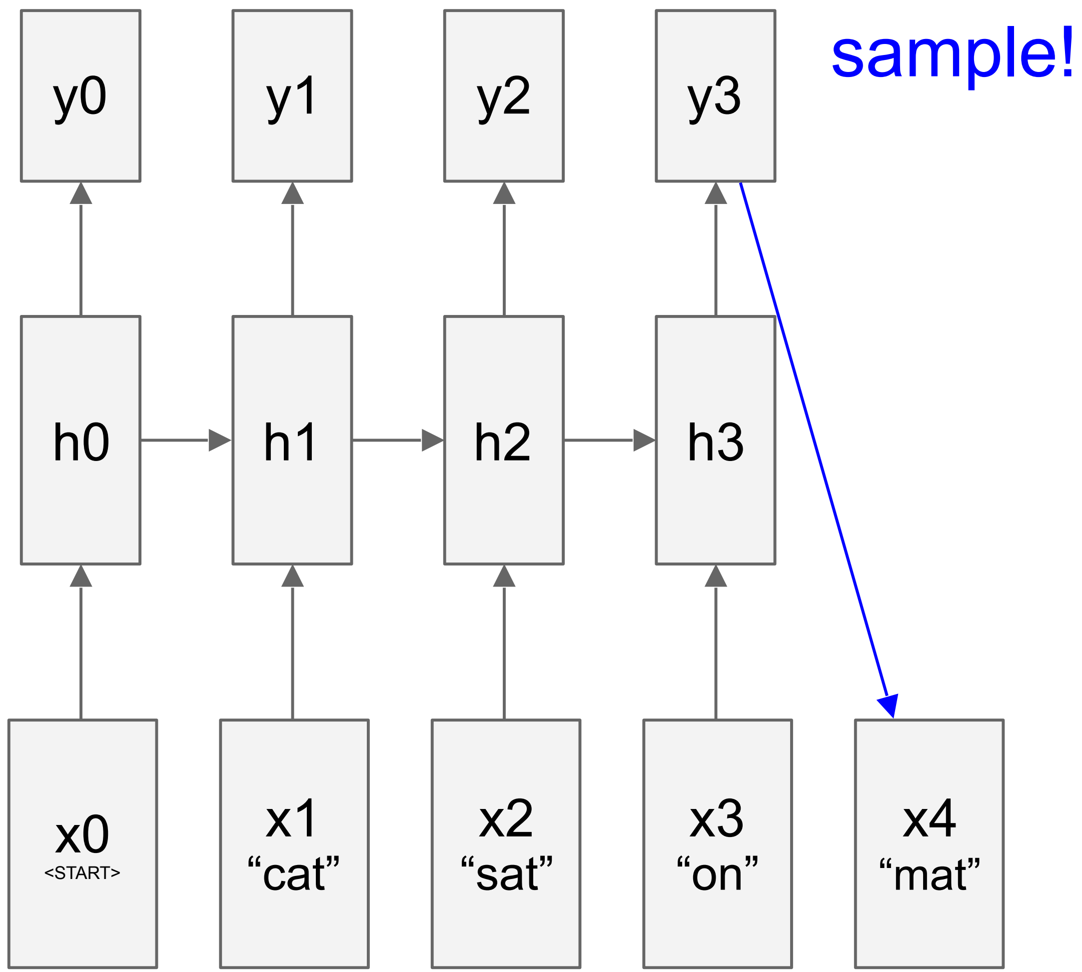


slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

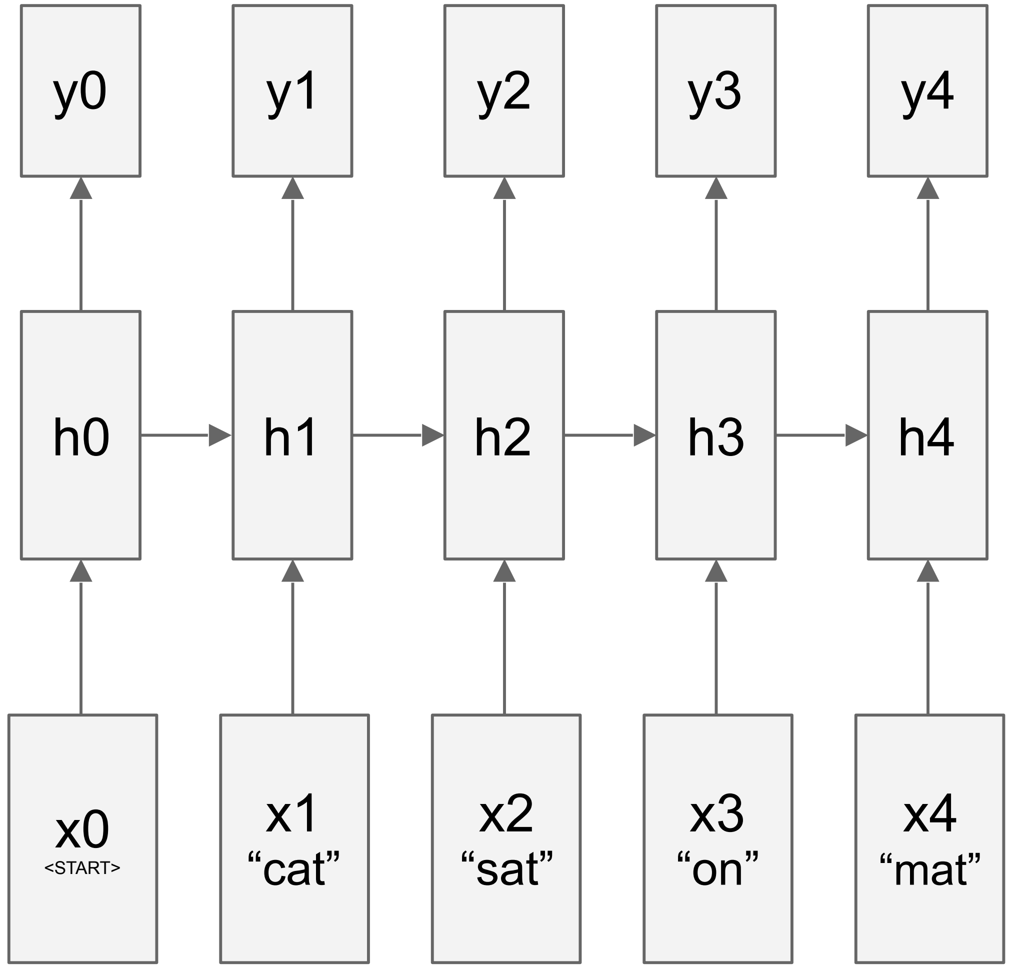


slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

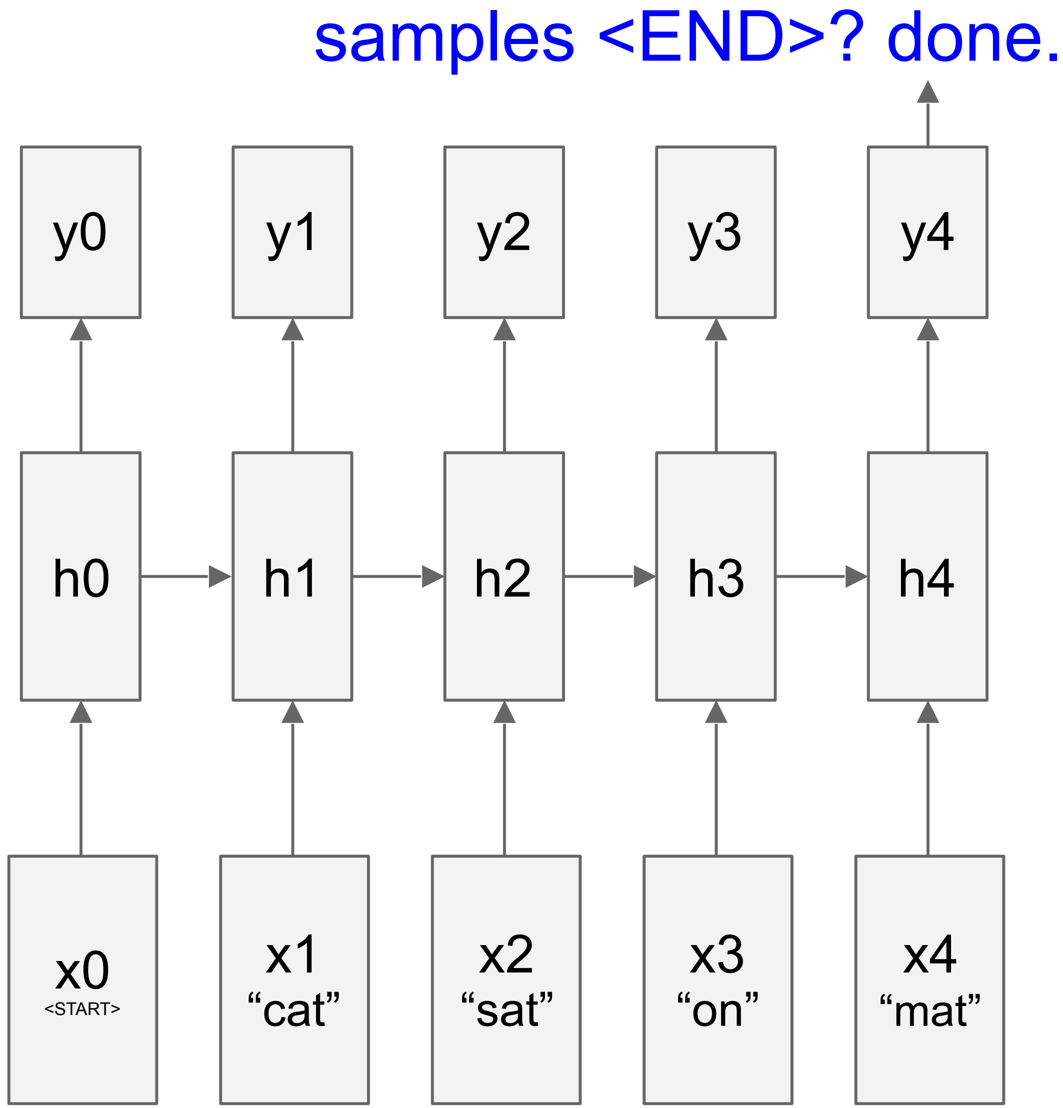


slide credit: Andrej Karpathy

Generating Sentences...

Training this on a lot of sentences would give us a language model. A way to predict

P(next word | previous words)

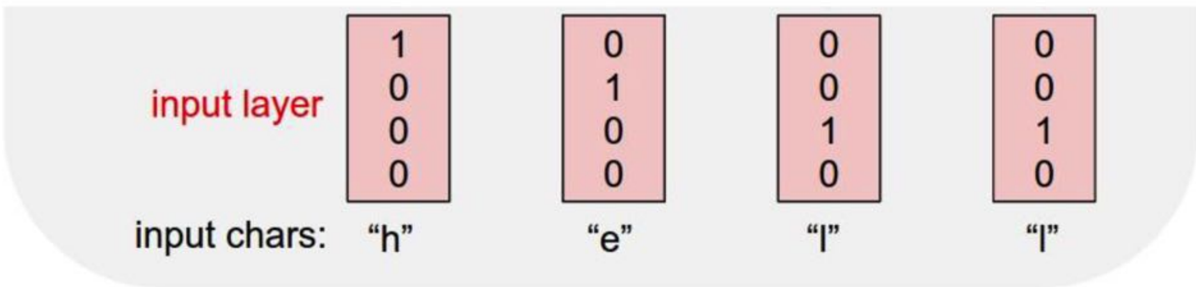


Example…

Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

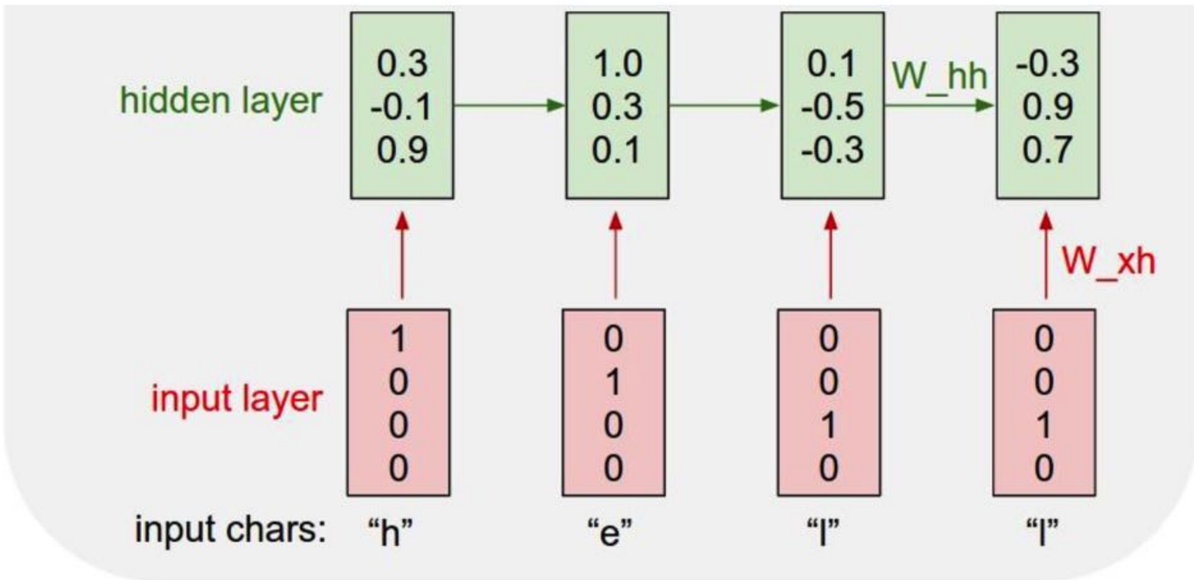
Example…

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



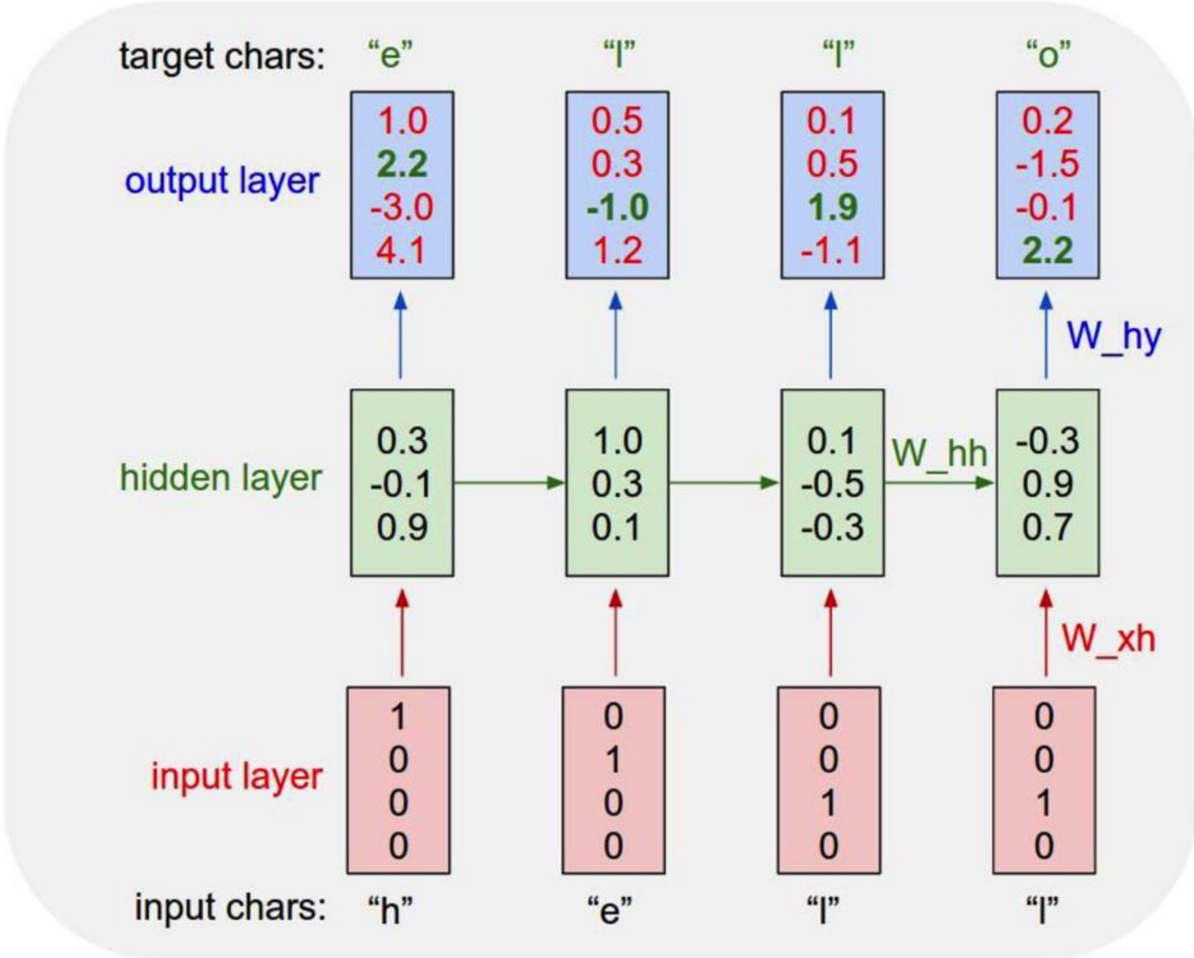
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Example…

Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



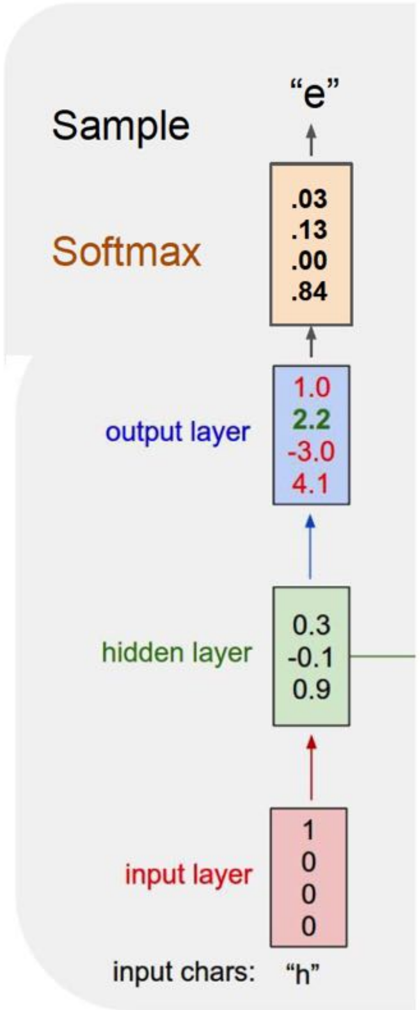
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Example…

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model



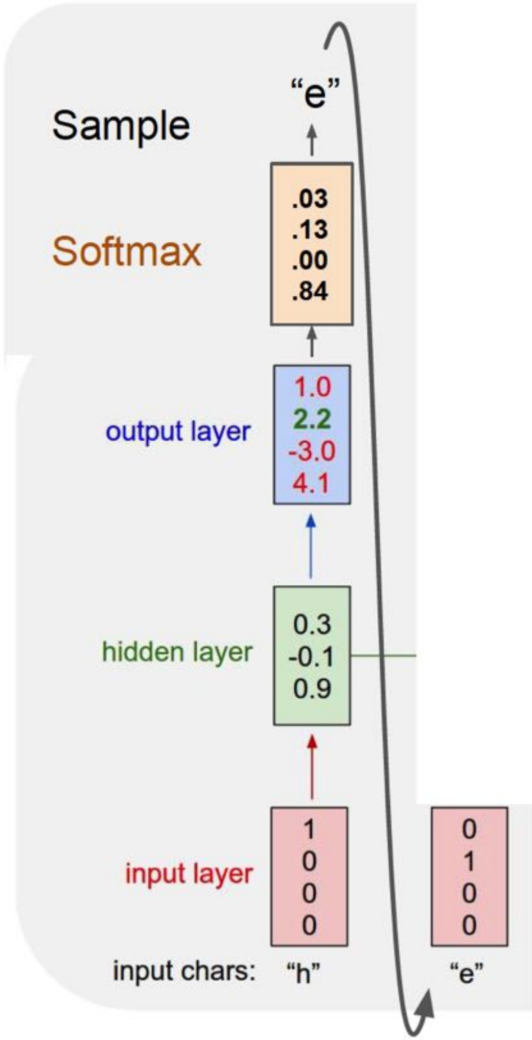
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Example…

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model



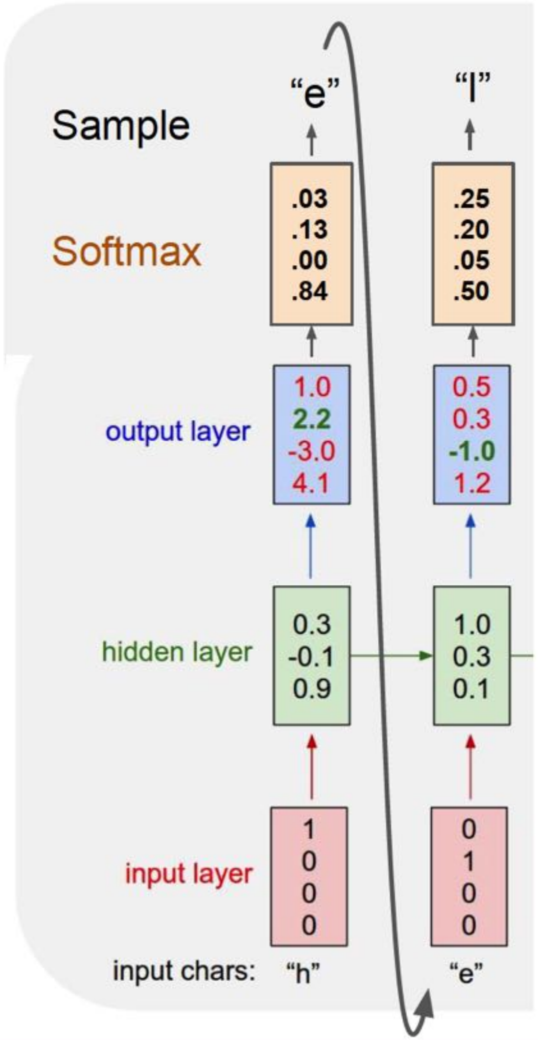
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Example…

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model



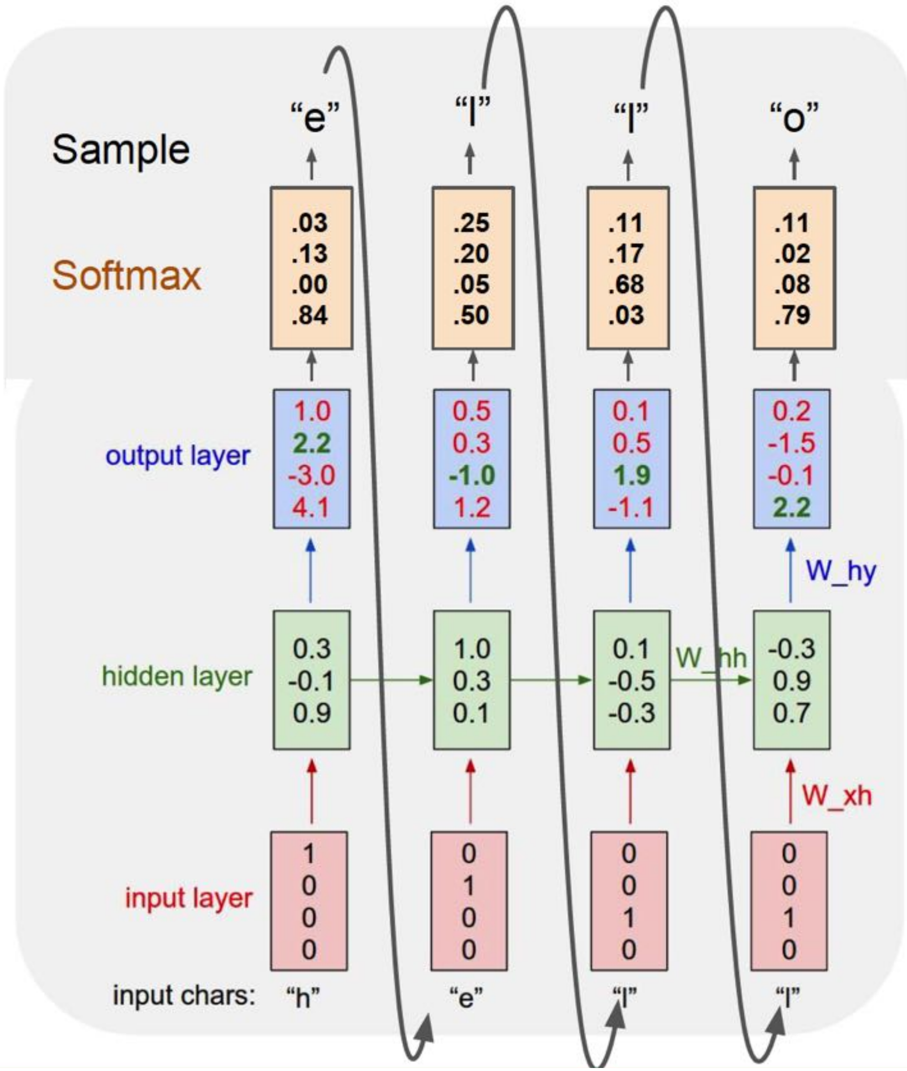
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Example…

Example: Character-level Language Model Sampling

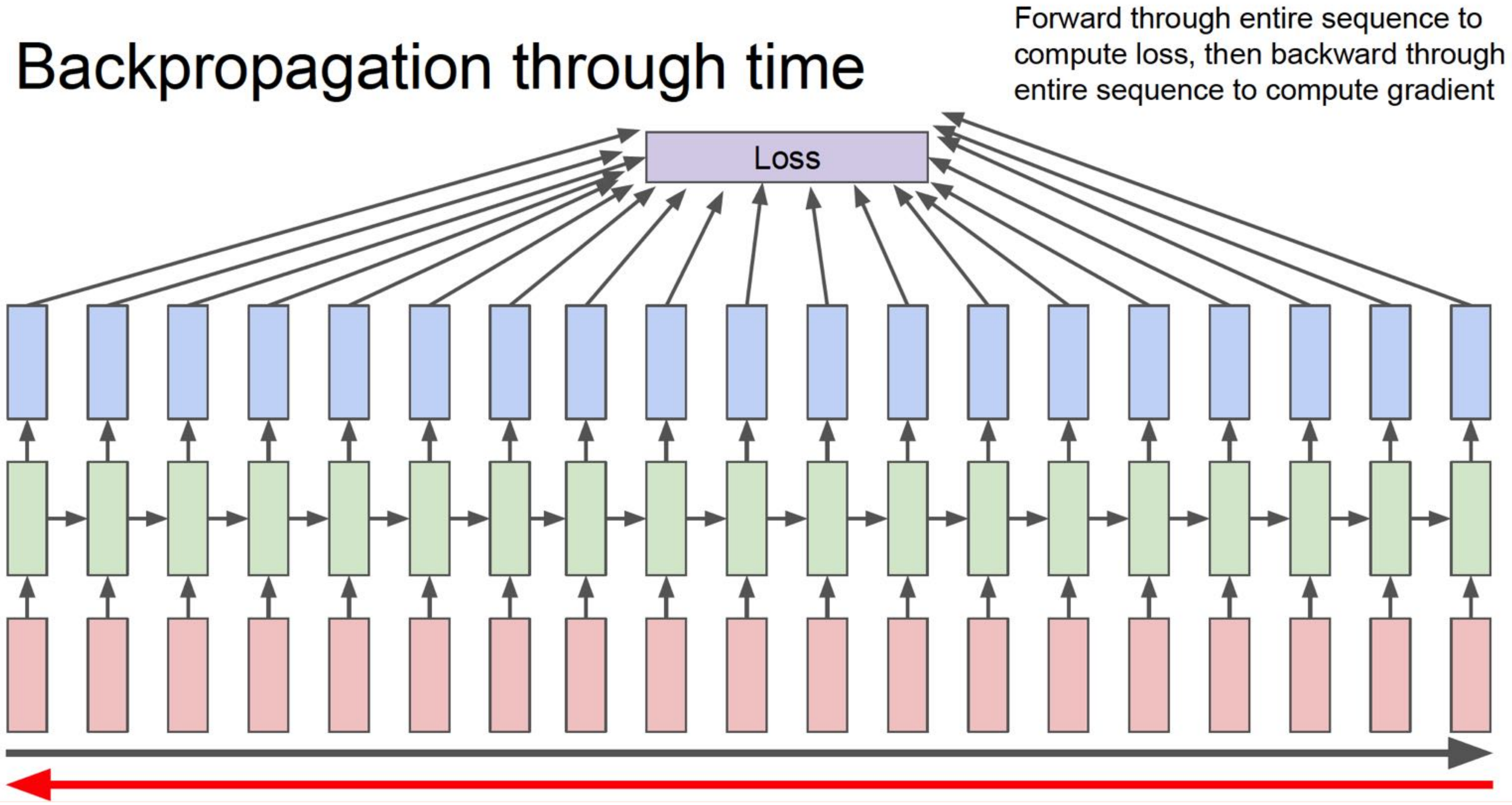
Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

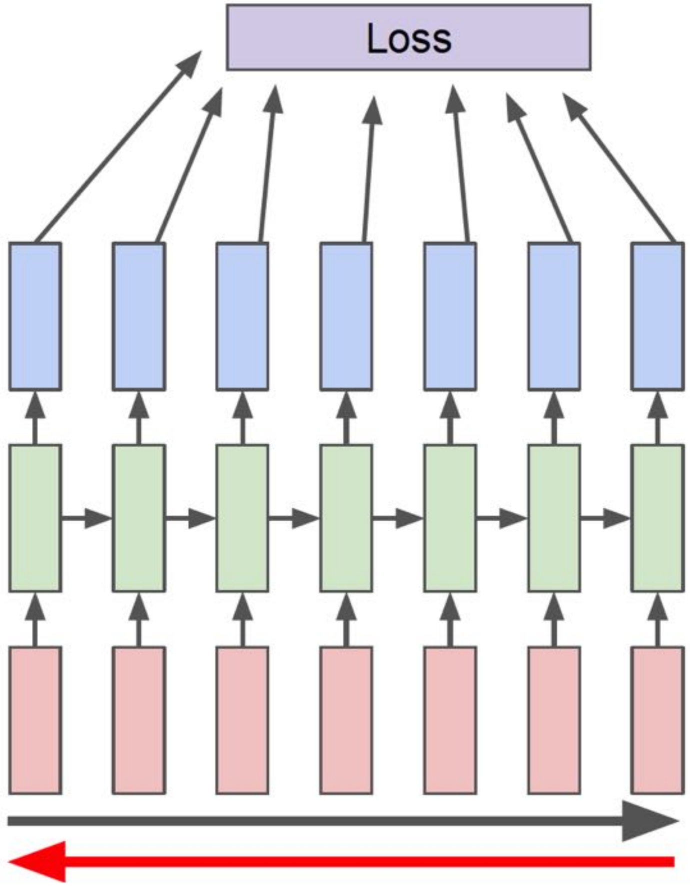
Learning via Backpropagation…



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Learning via Backpropagation…

Truncated Backpropagation through time

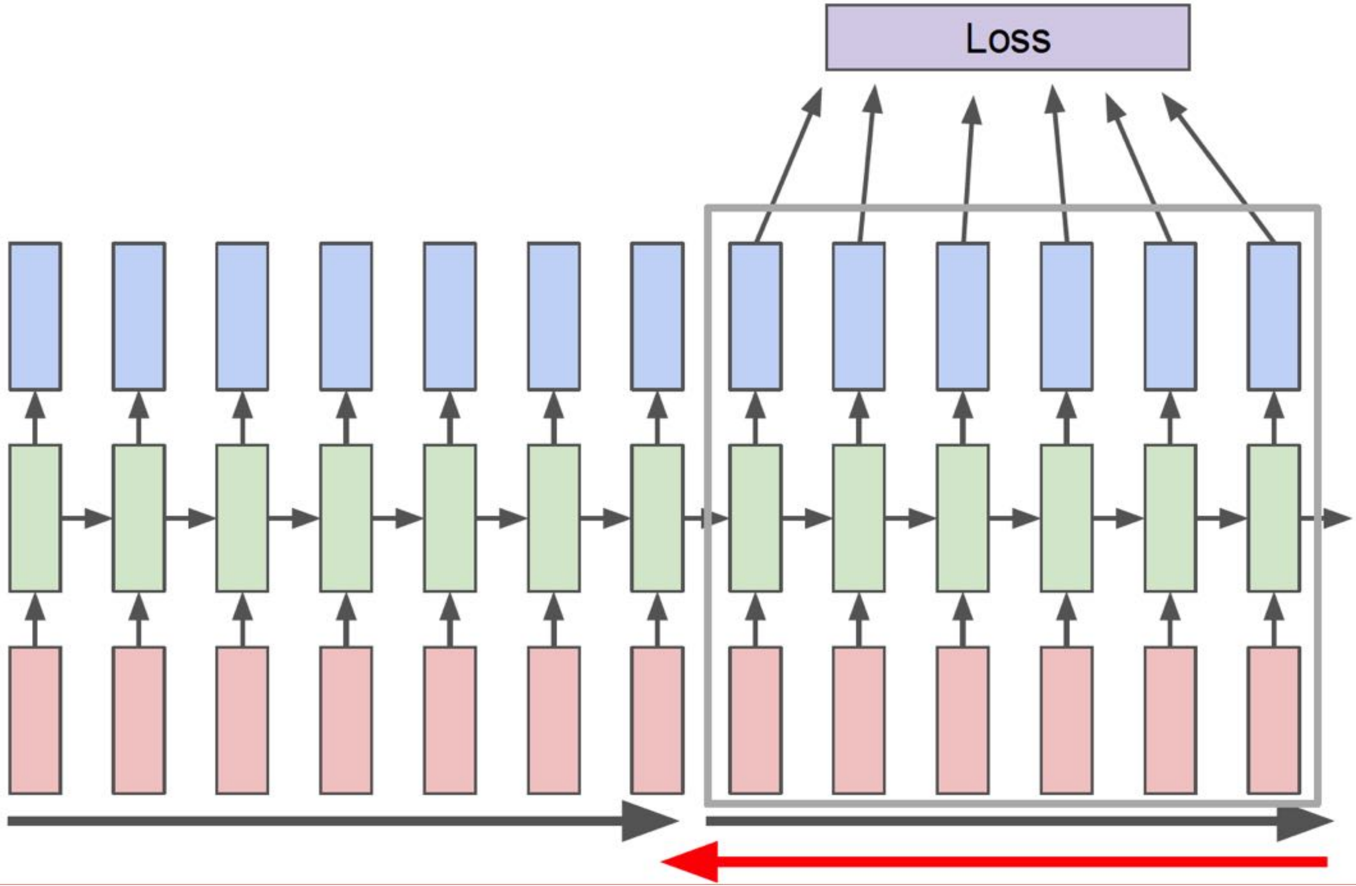


Run forward and backward through chunks of the sequence instead of whole sequence

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Learning via Backpropagation…

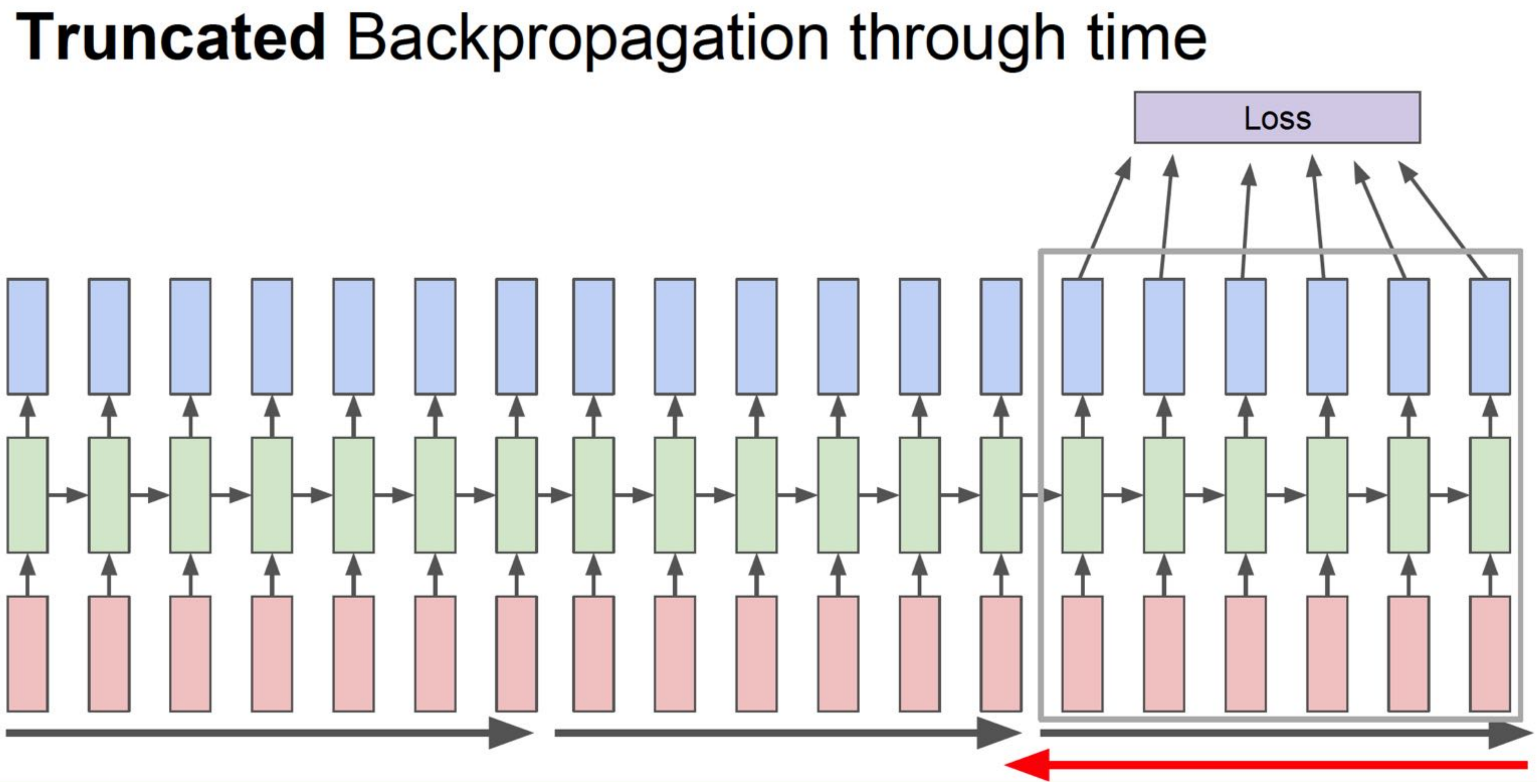
Truncated Backpropagation through time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Learning via Backpropagation…



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

“The Unreasonable Effectiveness of Recurrent Neural Networks”

karpathy.github.io

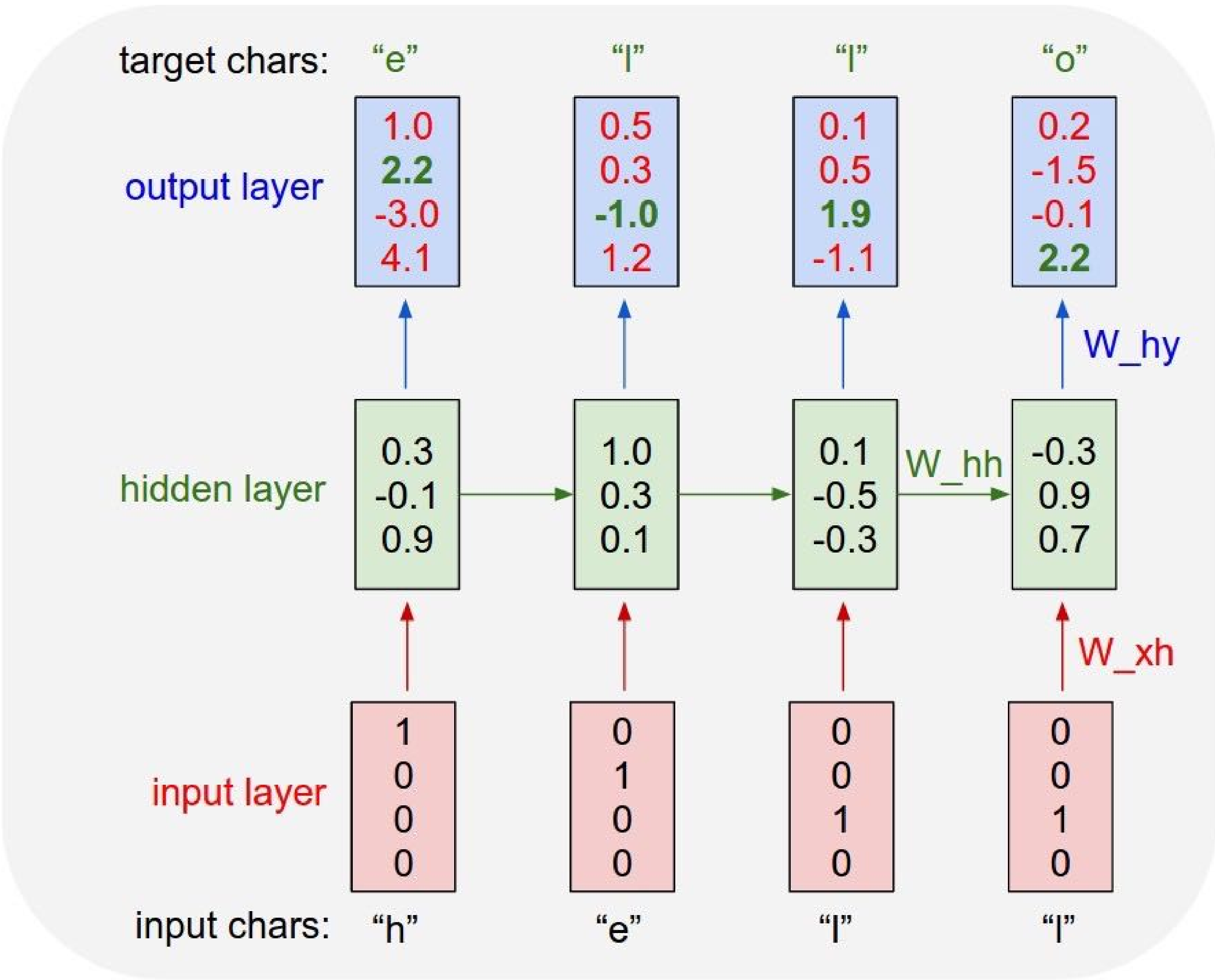
slide credit: Andrej Karpathy

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
"hello"

$$h_{t+1} = \tanh(W_{hh}h_t + W_{xh}x_t)$$



slide credit: Andrej Karpathy

Sonnet 116 – Let me not ...

by William Shakespeare

Let me not to the marriage of true minds
        Admit impediments. Love is not love
Which alters when it alteration finds,
        Or bends with the remover to remove:
O no! it is an ever-fixed mark
        That looks on tempests and is never shaken;
It is the star to every wandering bark,
        Whose worth's unknown, although his height be taken.
Love's not Time's fool, though rosy lips and cheeks
        Within his bending sickle's compass come:
Love alters not with his brief hours and weeks,
        But bears it out even to the edge of doom.
If this be error and upon me proved,
        I never writ, nor no man ever loved.

slide credit: Andrej Karpathy

The Stacks Project

home about tags explained tag lookup browse search bibliography recent comments blog add slogans

Browse chapters

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

slide credit: Andrej Karpathy

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, 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 $U \to T$ is a separated algebraic space.

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

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

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ 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 $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ 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 $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

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

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

is a unique morphism of algebraic stacks. Note that

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

and

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

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. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,étale}$ 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 $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

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

*When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ 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 $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

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

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

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 $\mathcal{J}_{n,0} \circ \overline{A}_2$ 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 mext 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. □

slide credit: Andrej Karpathy

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

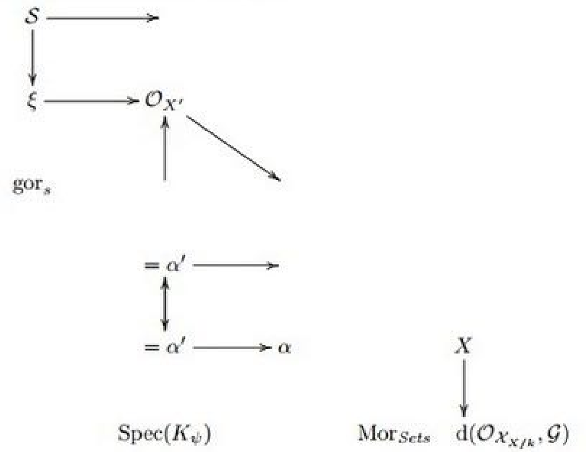
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

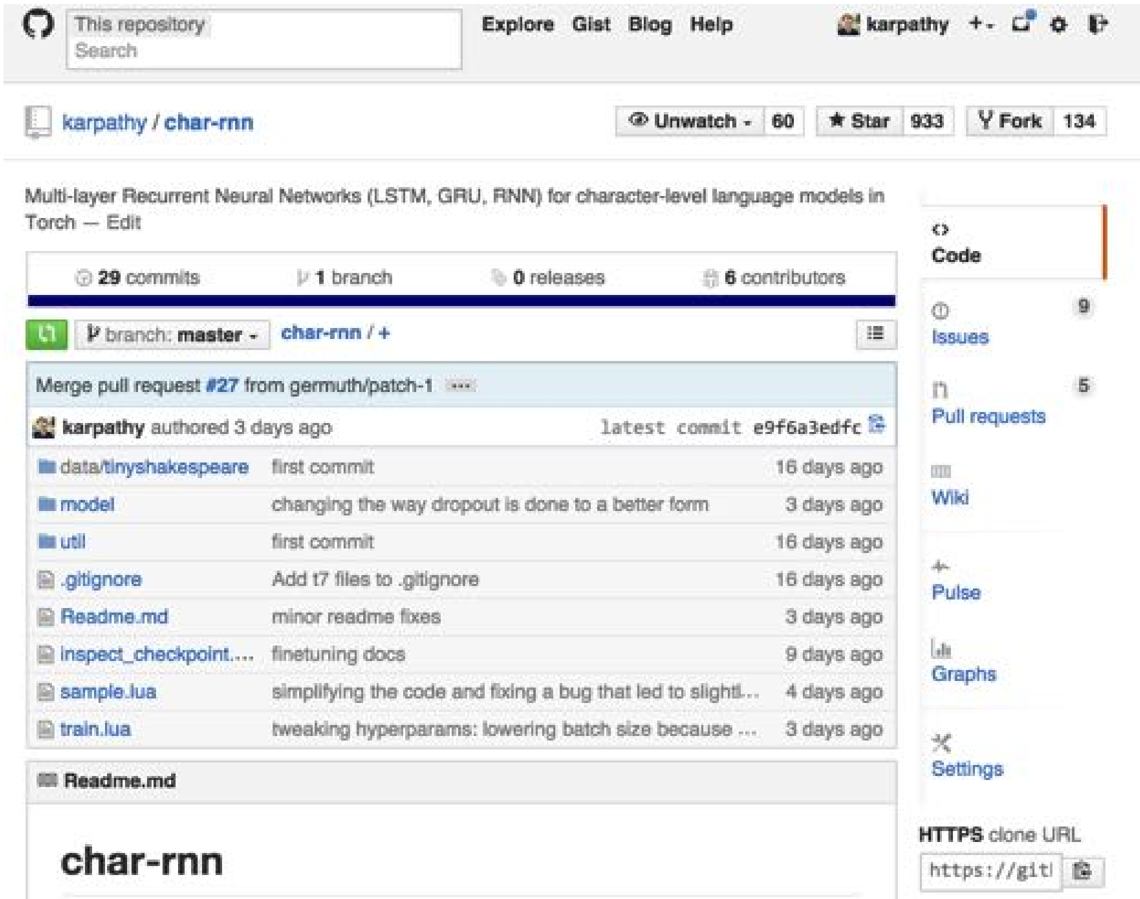
Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Try it yourself: **char-rnn** on Github (uses Torch7)



slide credit: Andrej Karpathy

Cooking Recipes

```
     Title: BASIC CHEESE WINGS:
Categories: Desserts
     Yield: 6 Servings

     3    Eggs
     2 tb Chopped fresh curry
          -or cooking spray
     1 c  Water; cooked
     2    Lemons minced mushrooms
     3 oz Sweet cooked rice
   1/2    Onion; chopped
     3 c  Butter, melted
     2 ts Soy sauce
     1 ts Cinnamon
     2 md Sugar or food coloring;
          -stems cored bowl
     2 tb Salt and freshly grated
   1/4 ts Ground ginger
   1/2 c  Flour
     1 tb Water; fresh parsley
     1 c  Water (or or)
     1    Clove garlic, minced

 Preheat oven to 350F. Combine sugar, salt, baking soda, celery and sugar. Add
 the chicken broth well. Add the cornstarch to the pan; cool. Add the olive
 oil, oil, and basil or cooking spray. Pour the onions until melted.
```

Obama Speeches

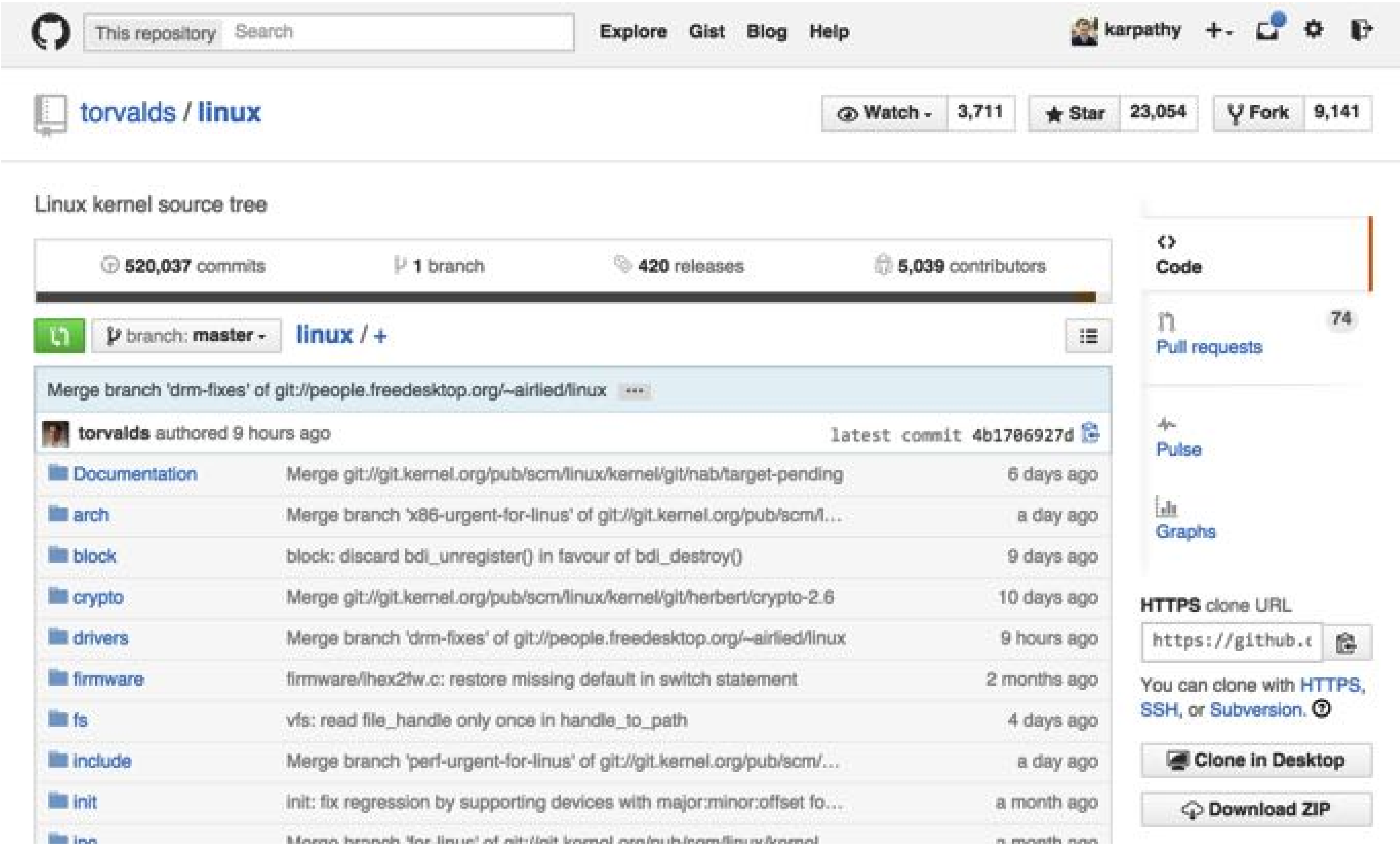
Good afternoon. God bless you.

The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women who were still going to take out the fact that the American people have fought to make sure that they have to be able to protect our part. It was a chance to stand together to completely look for the commitment to borrow from the American people. And the fact is the men and women in uniform and the millions of our country with the law system that we should be a strong stretcks of the forces that we can afford to increase our spirit of the American people and the leadership of our country who are on the Internet of American lives.

Thank you very much. God bless you, and God bless the United States of America.

slide credit: Andrej Karpathy

slide credit: Andrej Karpathy



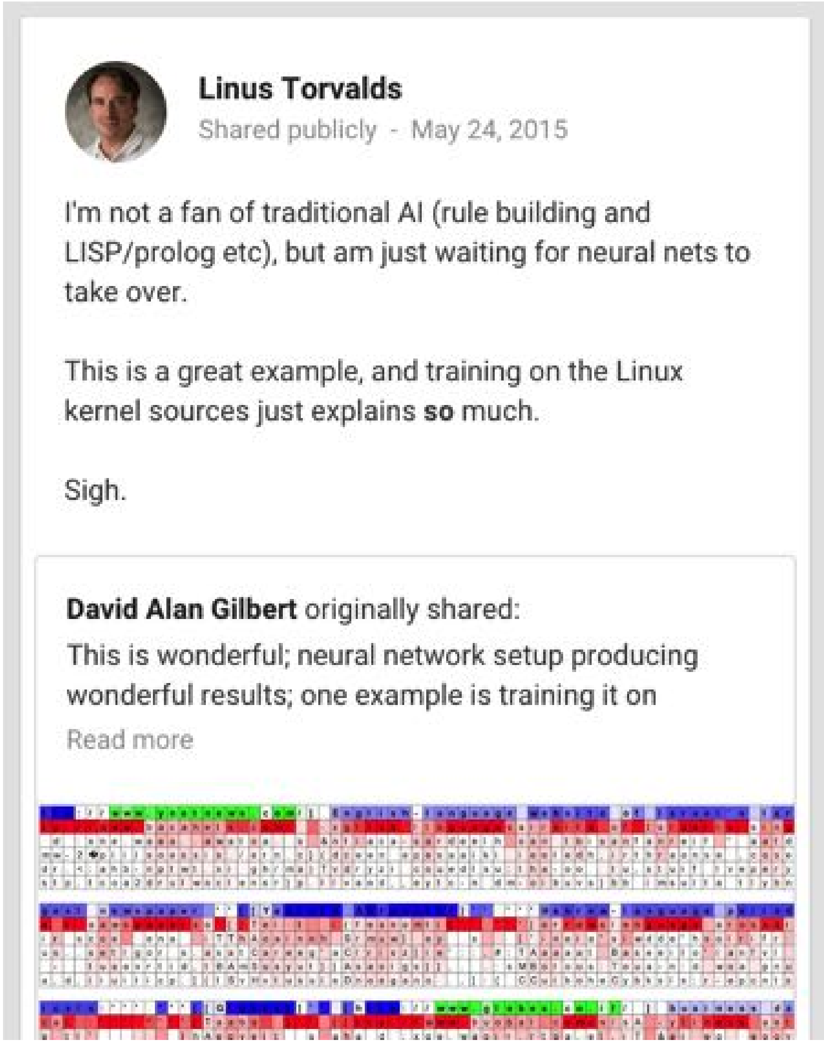
slide credit: Andrej Karpathy

Learning from Linux Source Code

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

slide credit: Andrej Karpathy

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```



slide credit: Andrej Karpathy

```
/*
 *  Copyright (c) 2006-2010, Intel Mobile Communications.  All rights reserved.
 *
 *   This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License version 2 as published by
 * the Free Software Foundation.
 *
 *        This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *
 *  GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *    along with this program; if not, write to the Free Software Foundation,
 *  Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

#include <linux/kexec.h>
#include <linux/errno.h>
#include <linux/io.h>
#include <linux/platform_device.h>
#include <linux/multi.h>
#include <linux/ckevent.h>

#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>
```

slide credit: Andrej Karpathy

```
#include <asm/io.h>
#include <asm/prom.h>
#include <asm/e820.h>
#include <asm/system_info.h>
#include <asm/setew.h>
#include <asm/pgproto.h>

#define REG_PG    vesa_slot_addr_pack
#define PFM_NOCOMP  AFSR(0, load)
#define STACK_DDR(type)     (func)

#define SWAP_ALLOCATE(nr)     (e)
#define emulate_sigs()  arch_get_unaligned_child()
#define access_rw(TST)  asm volatile("movd %%esp, %0, %3" : : "r" (0));   \
  if (__type & DO_READ)

static void stat_PC_SEC __read_mostly offsetof(struct seq_argsqueue, \
          pC>[1]);

static void
os_prefix(unsigned long sys)
{
#ifdef CONFIG_PREEMPT
  PUT_PARAM_RAID(2, sel) = get_state_state();
  set_pid_sum((unsigned long)state, current_state_str(),
           (unsigned long)-1->lr_full; low;
}
```

slide credit: Andrej Karpathy

Yoav Goldberg n-gram experiments

Order 10 ngram model on Shakespeare:

```
APEMANTUS:
Ho, ho! I laugh to see your beard!

BOYET:
Madam, in great extremes of passion as she
discovers it.

PAROLLES:
By my white head and her wit
Values itself: to the sepulchre!'
With this, my lord,
That I have some business: let's away.

First Keeper:
Forbear to murder: and wilt thou not say he lies,
And lies, and let the devil would have said, sir, their speed
Hath been balm to heal their woes,
```

slide credit: Andrej Karpathy

But on Linux:

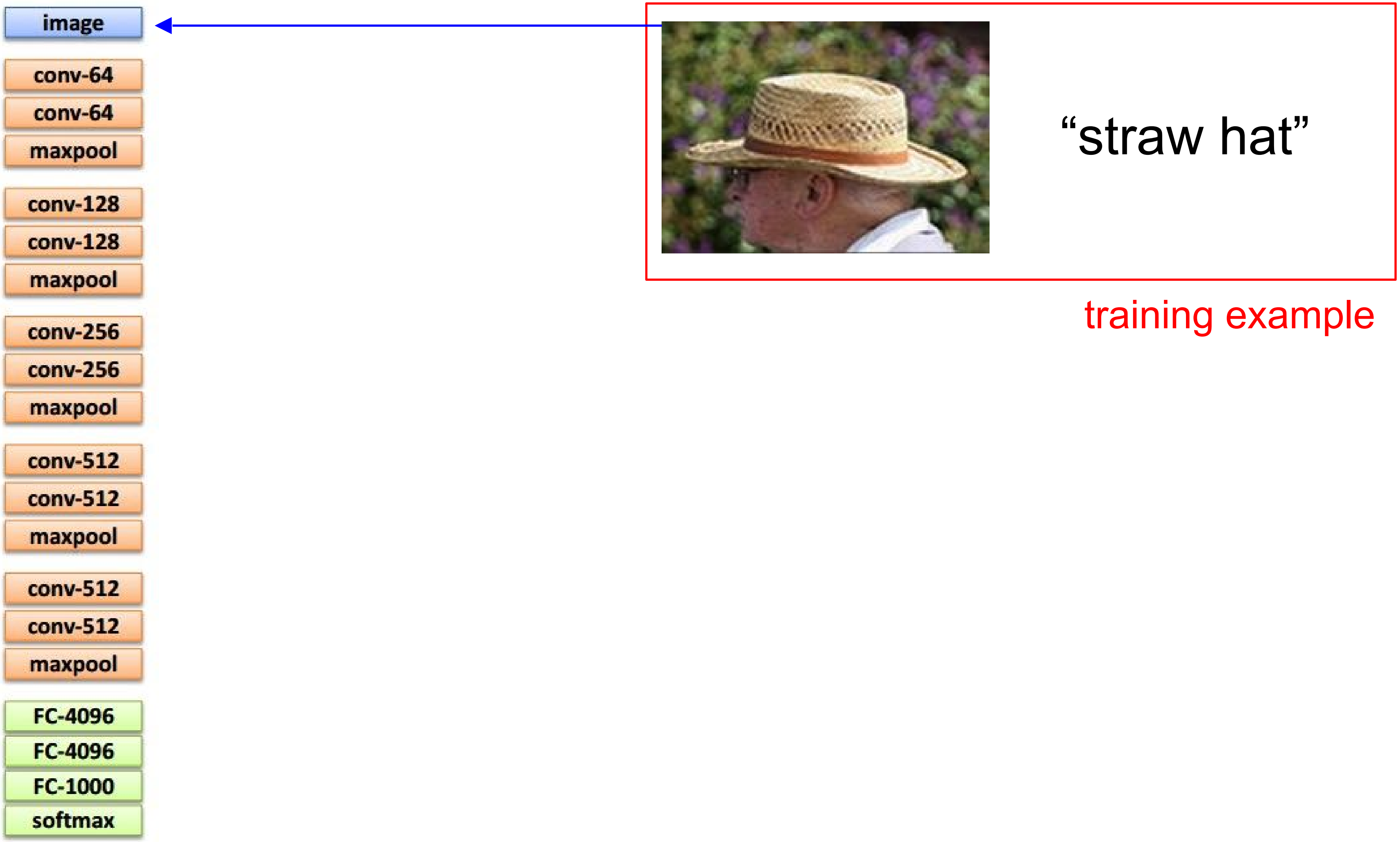
```
~~/*
 * linux/kernel/time.c
 * Please report this on hardware.
 */
void irq_mark_irq(unsigned long old_entries, eval);

                /*
                 * Divide only 1000 for ns^2 -> us^2 conversion values
don't overflow:
                seq_puts(m, "\ttramp: %pS",
                                        (void *)class->contending_point
]++;
        if (likely(t->flags & WQ_UNBOUND)) {
                /*
                 * Update inode information. If the
                 * slowpath and sleep time (abs or rel)
 * @rmtp: remaining (either due
 * to consume the state of ring buffer size. */
        header_size - size, in bytes, of the chain.
                 */
                BUG_ON(!error);
                } while (cgrp) {
                if (old) {
                if (kdb_continue_catastrophic;
#endif
```

slide credit: Andrej Karpathy

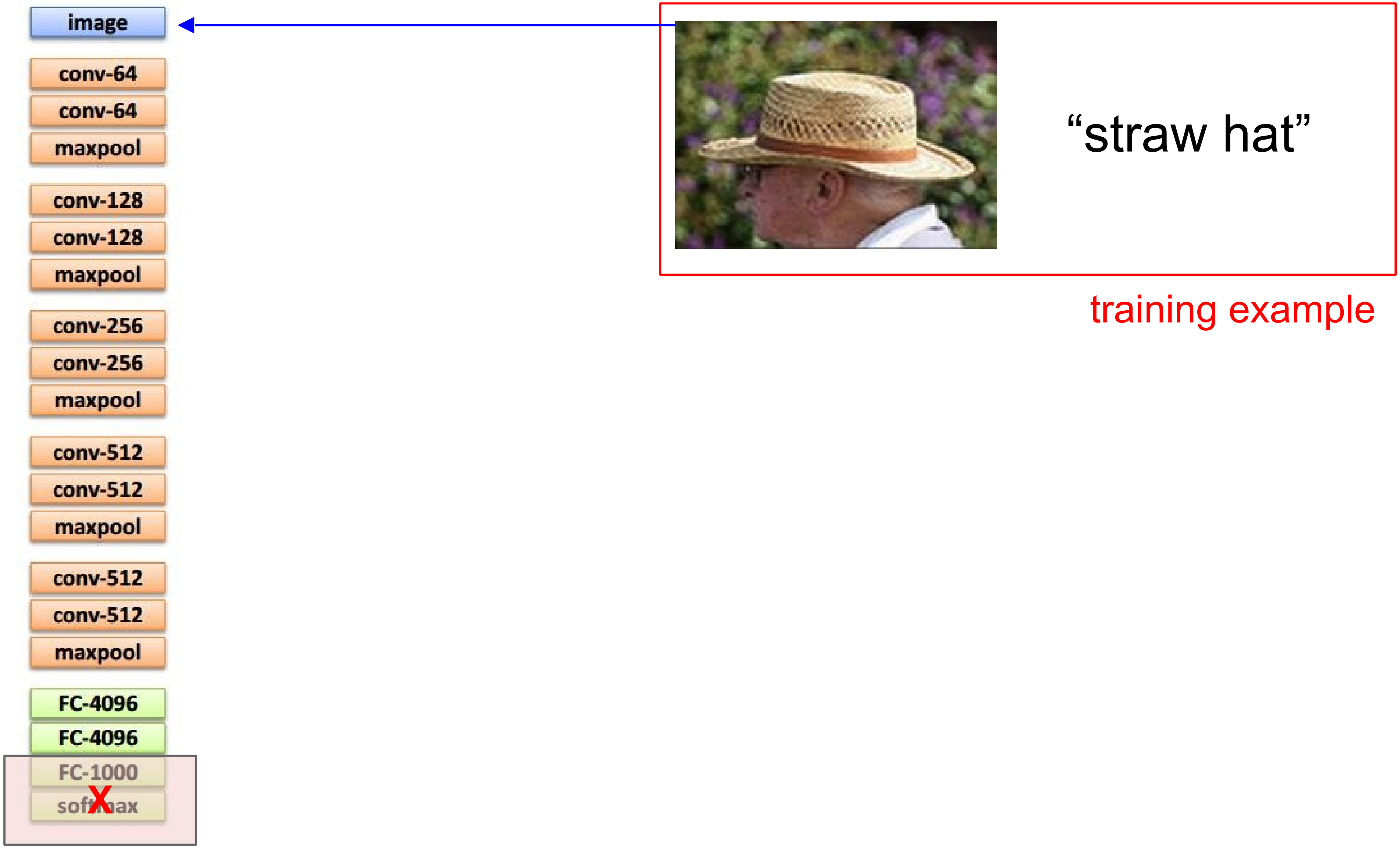
“straw hat”

training example

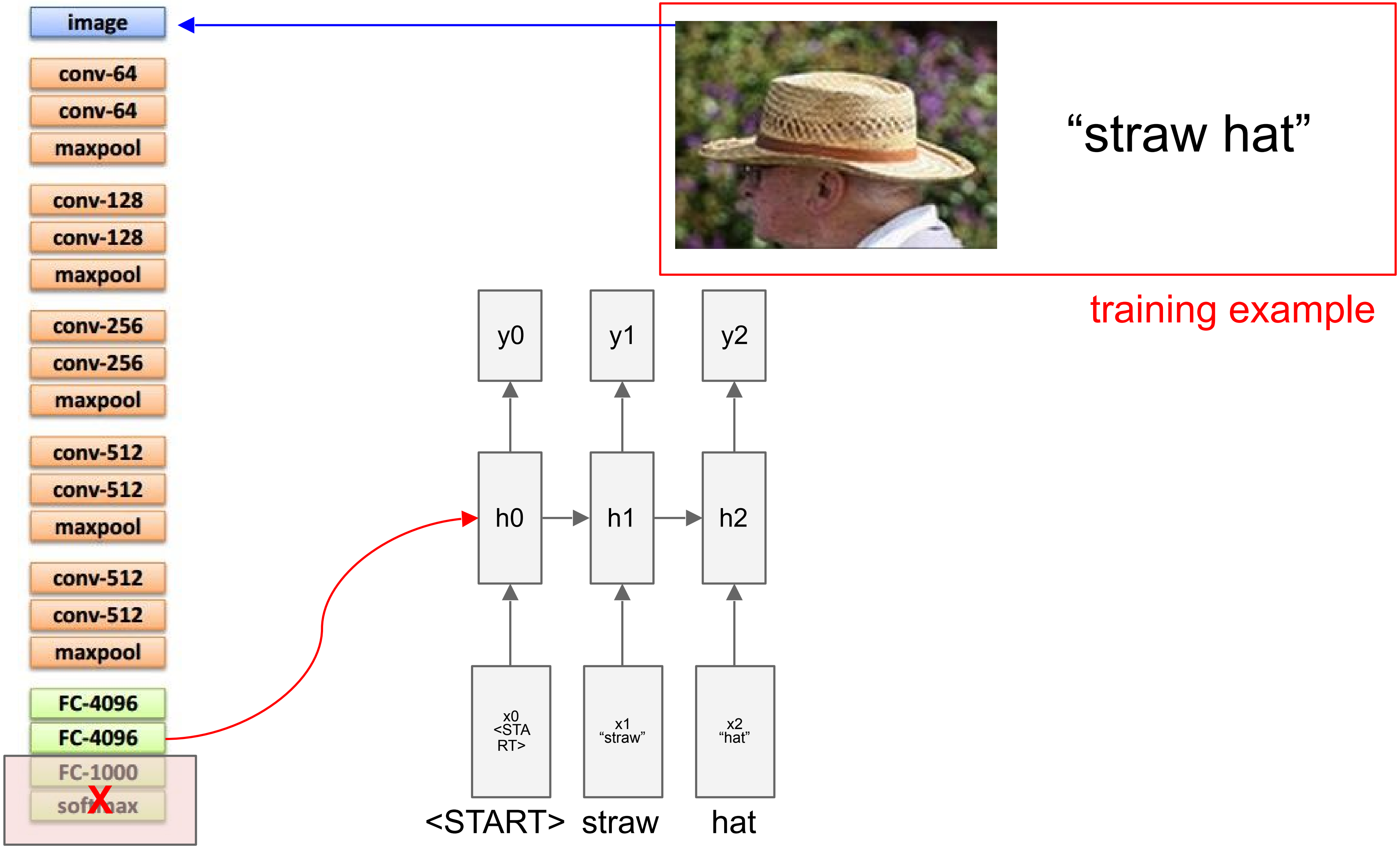
slide credit: Andrej Karpathy

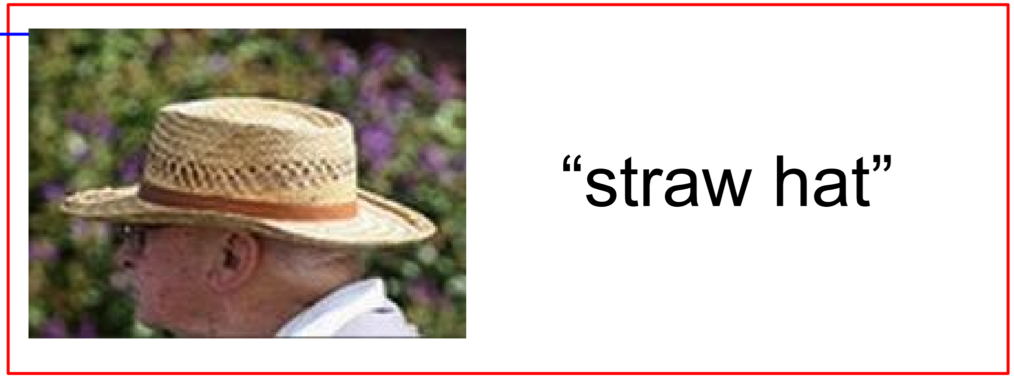


slide credit: Andrej Karpathy

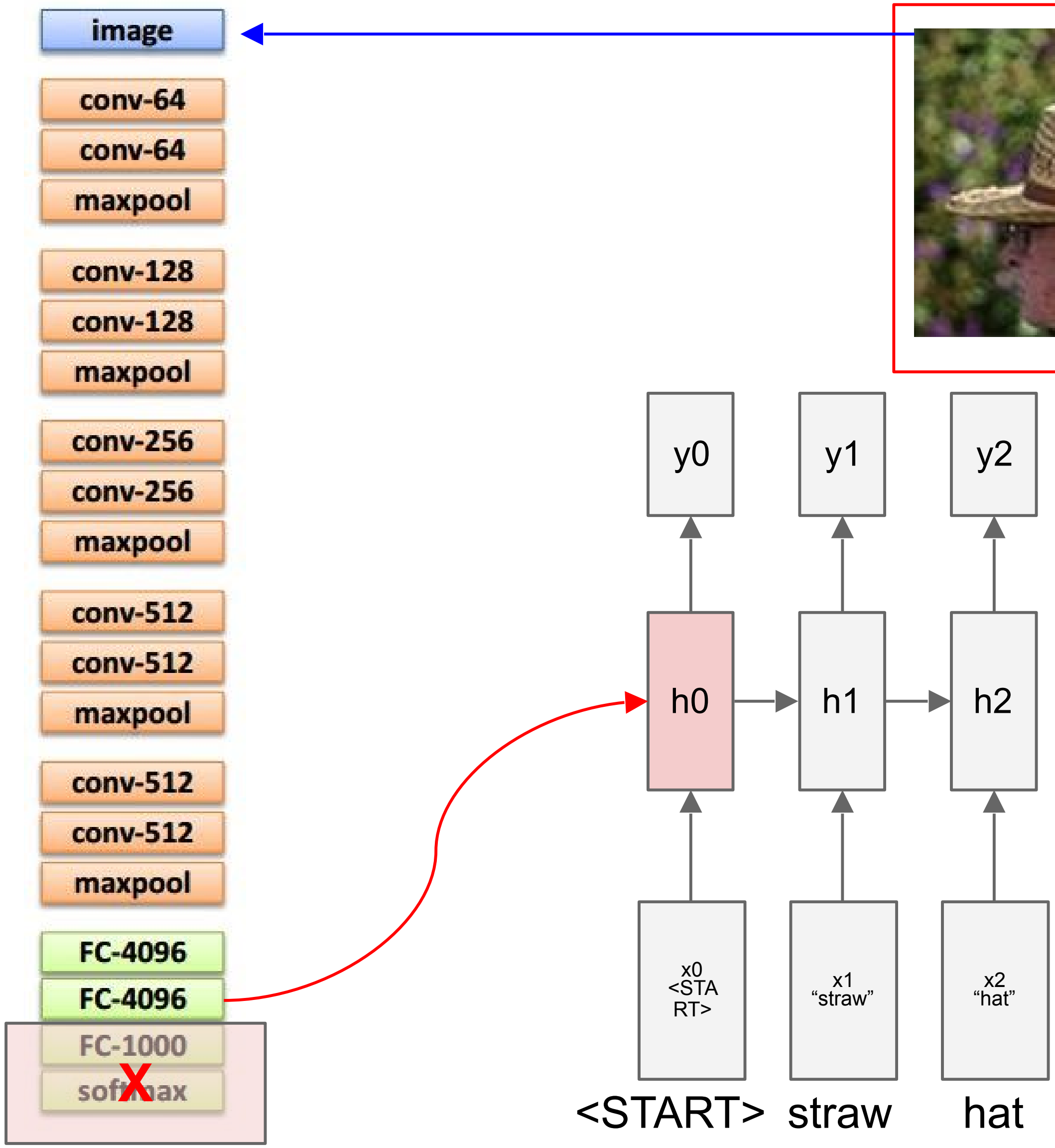


slide credit: Andrej Karpathy





training example



before:

h0 = tanh(0, Wxh * x0)

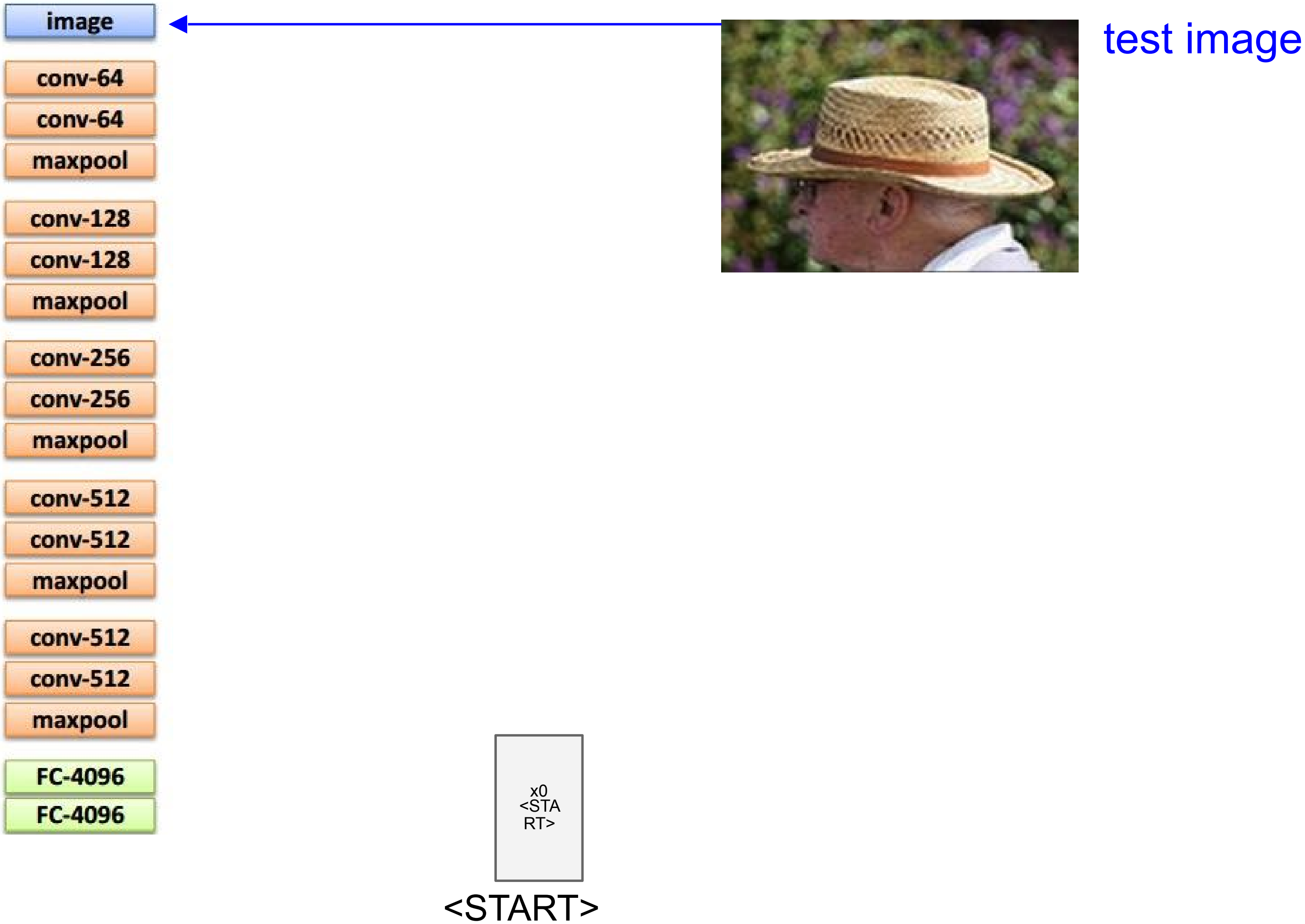
now:

h0 = tanh(0, Wxh * x0 + Wih * v)

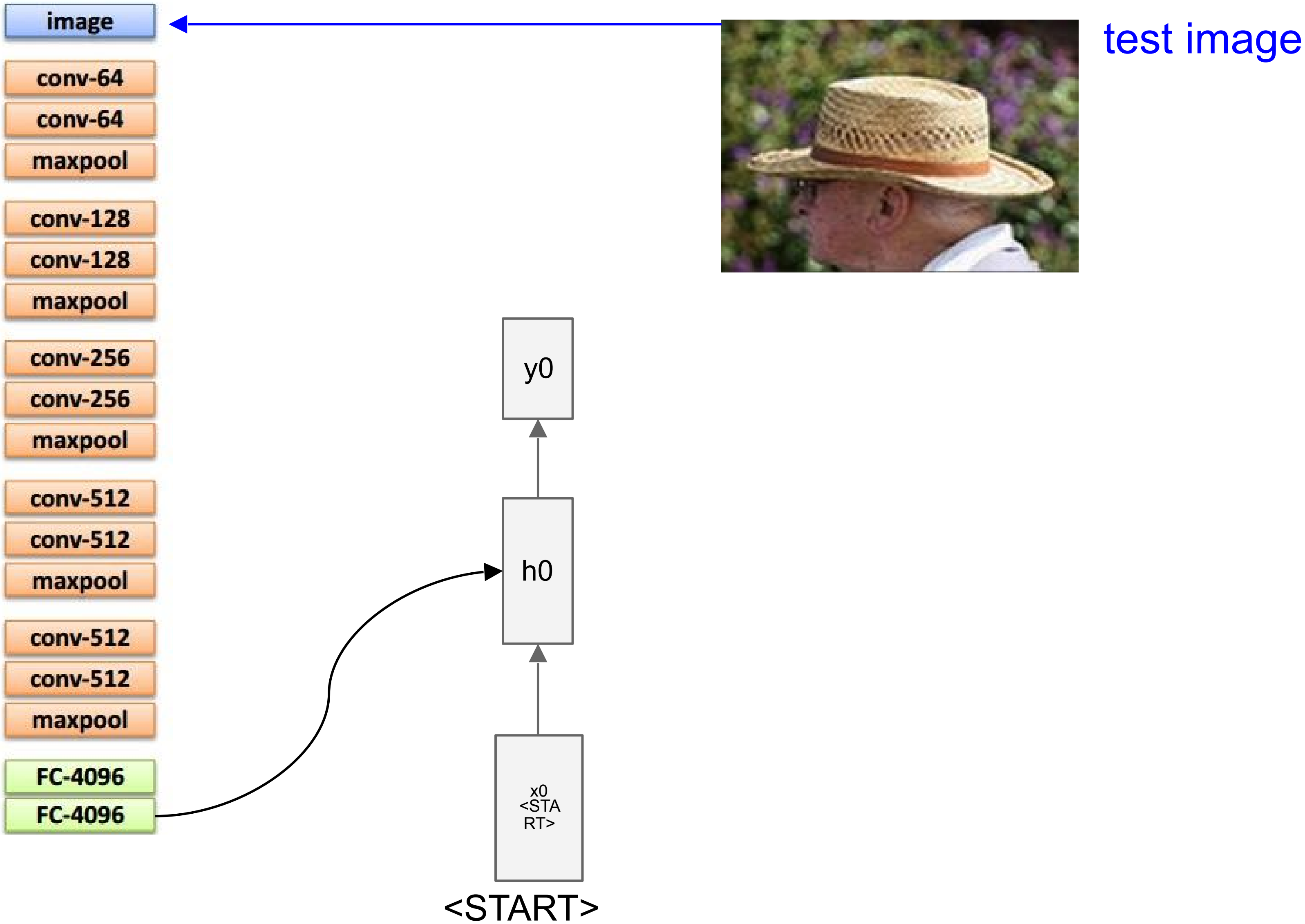
slide credit: Andrej Karpathy

test image

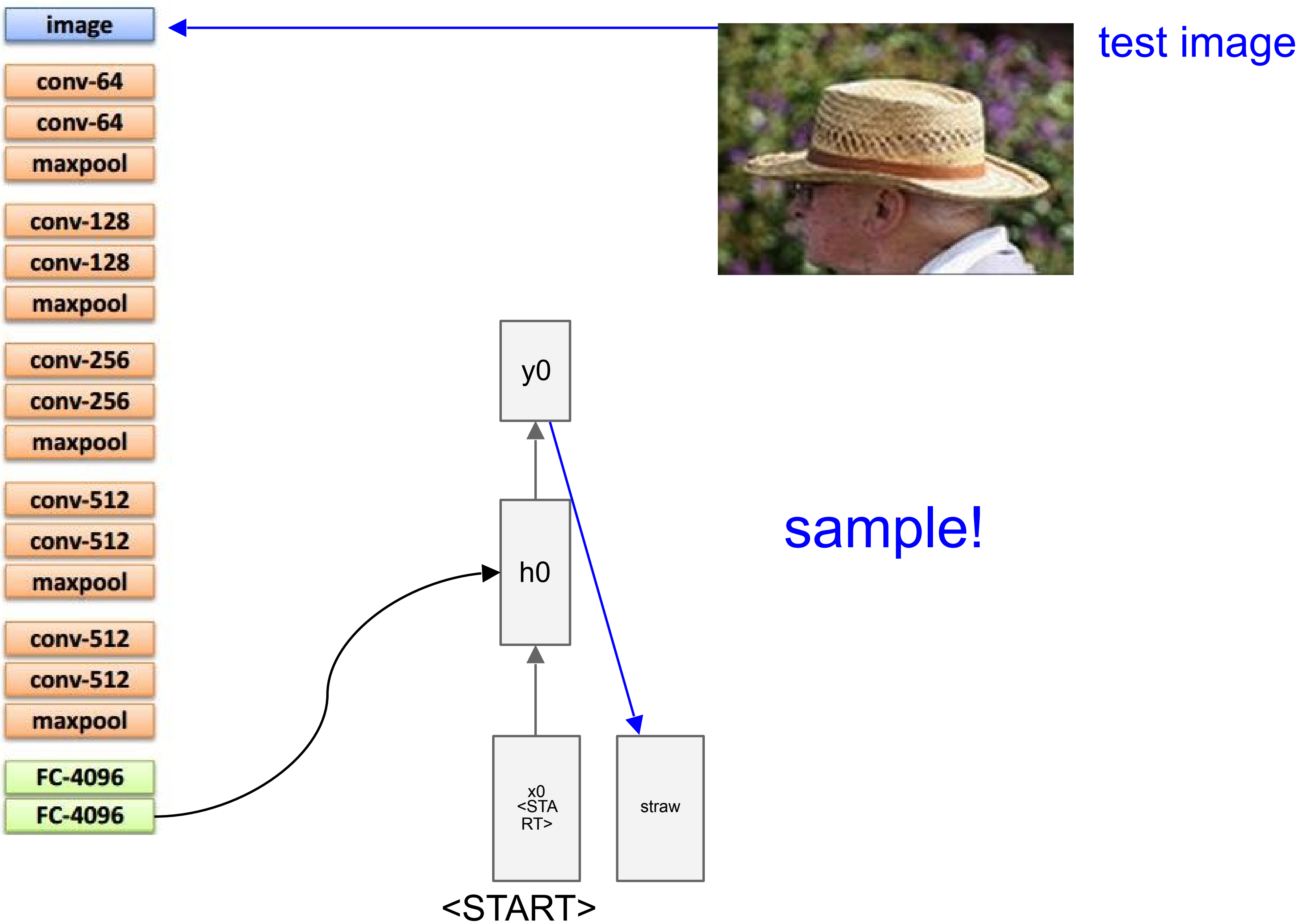
slide credit: Andrej Karpathy



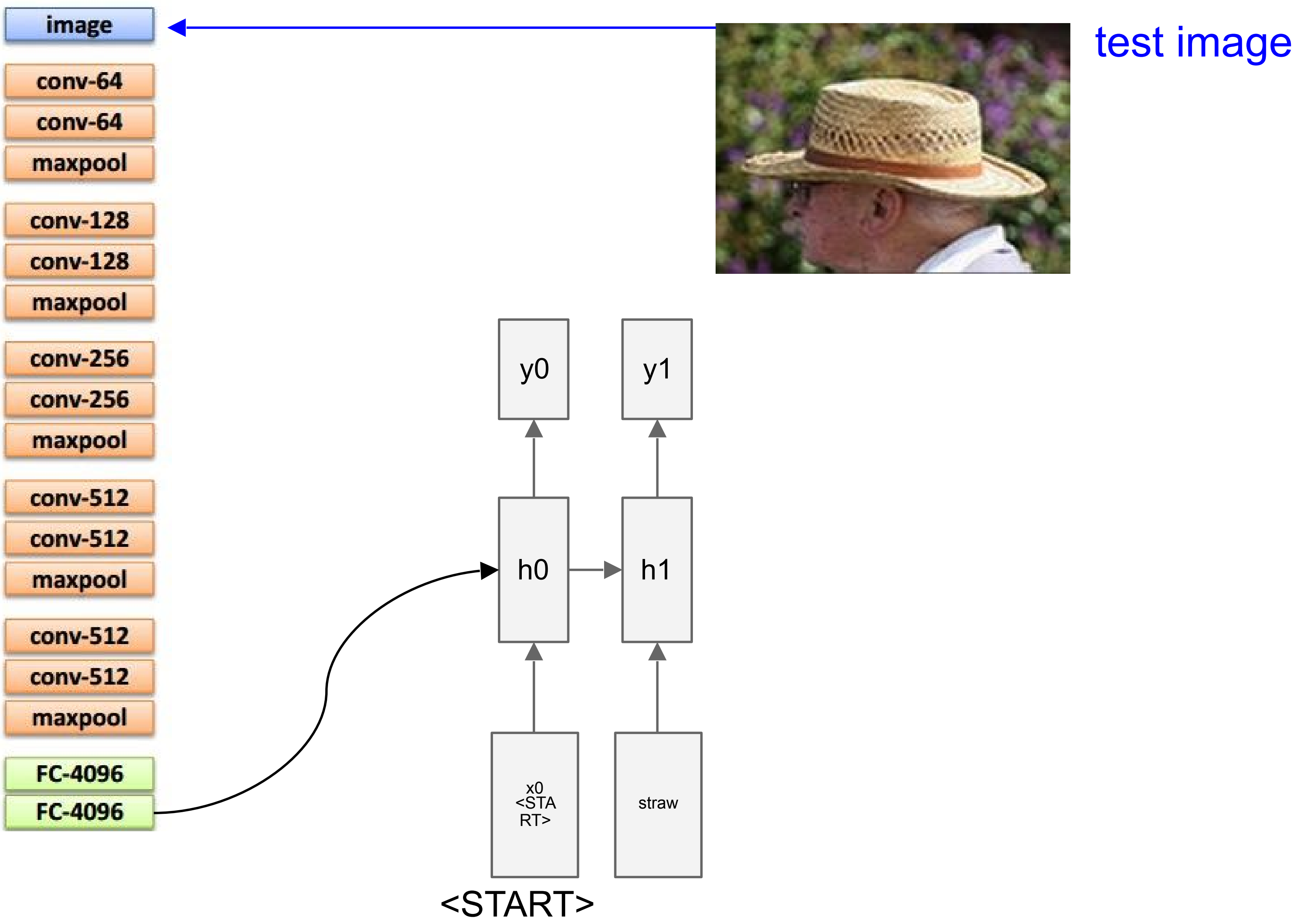
slide credit: Andrej Karpathy



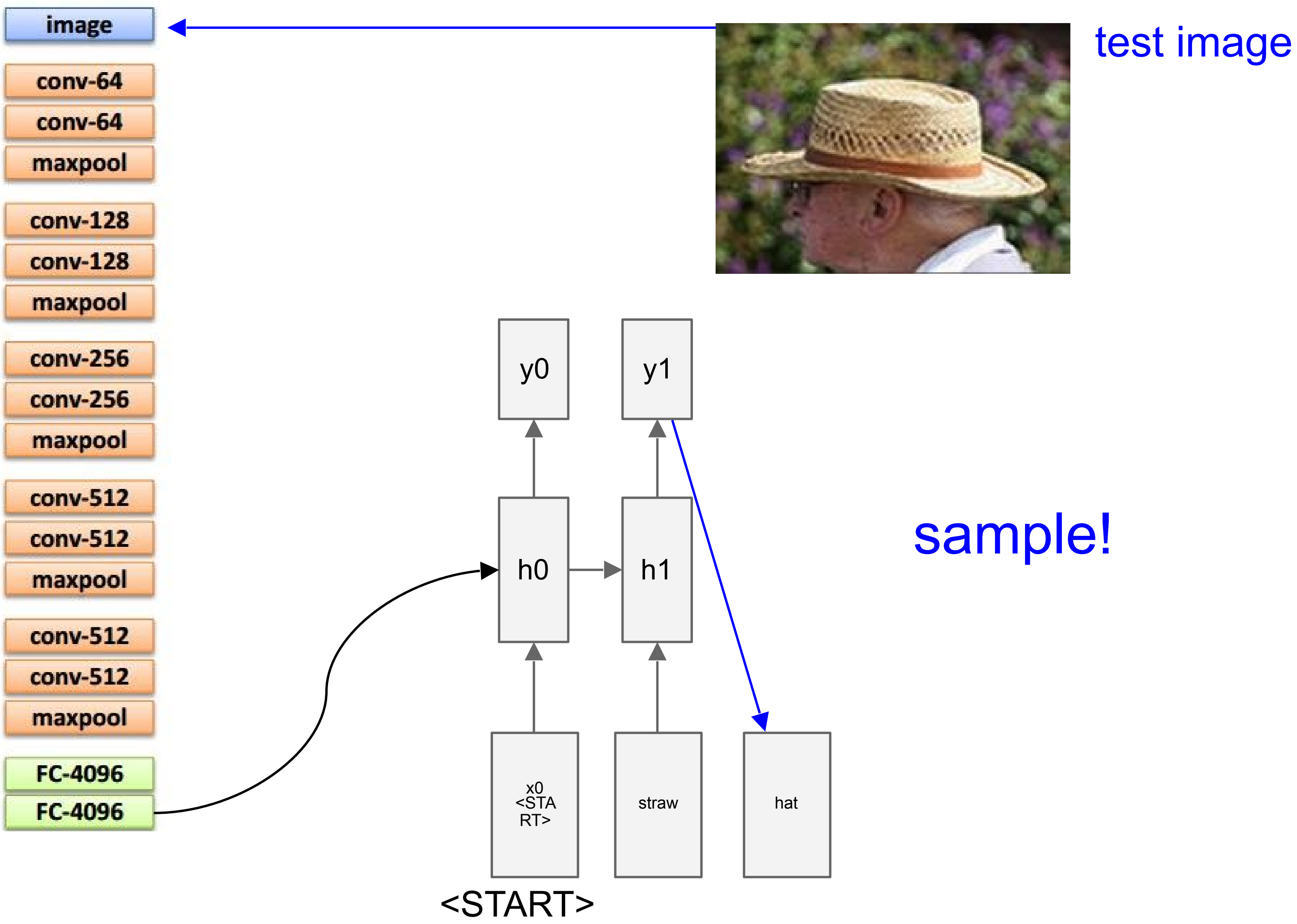
slide credit: Andrej Karpathy



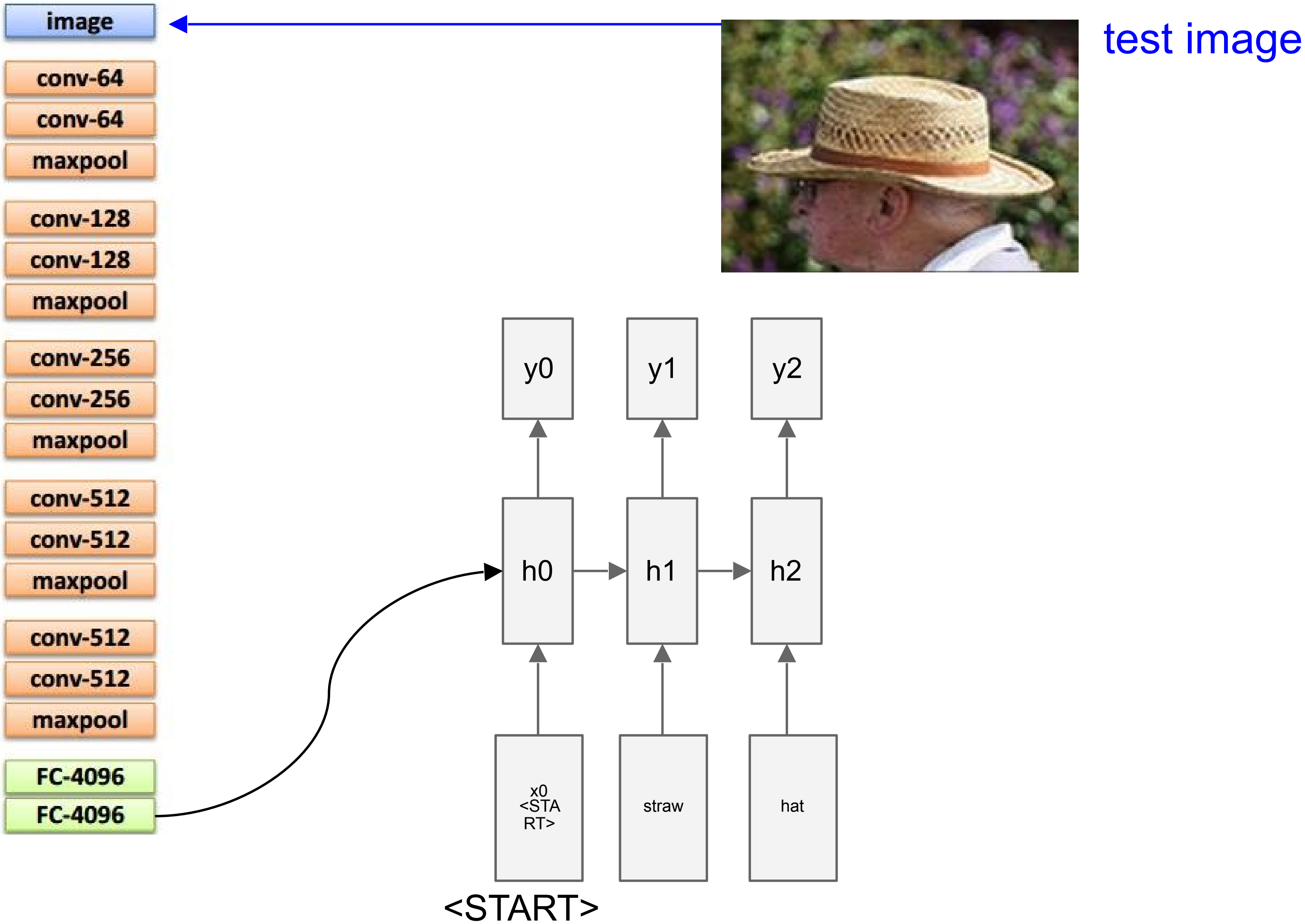
slide credit: Andrej Karpathy



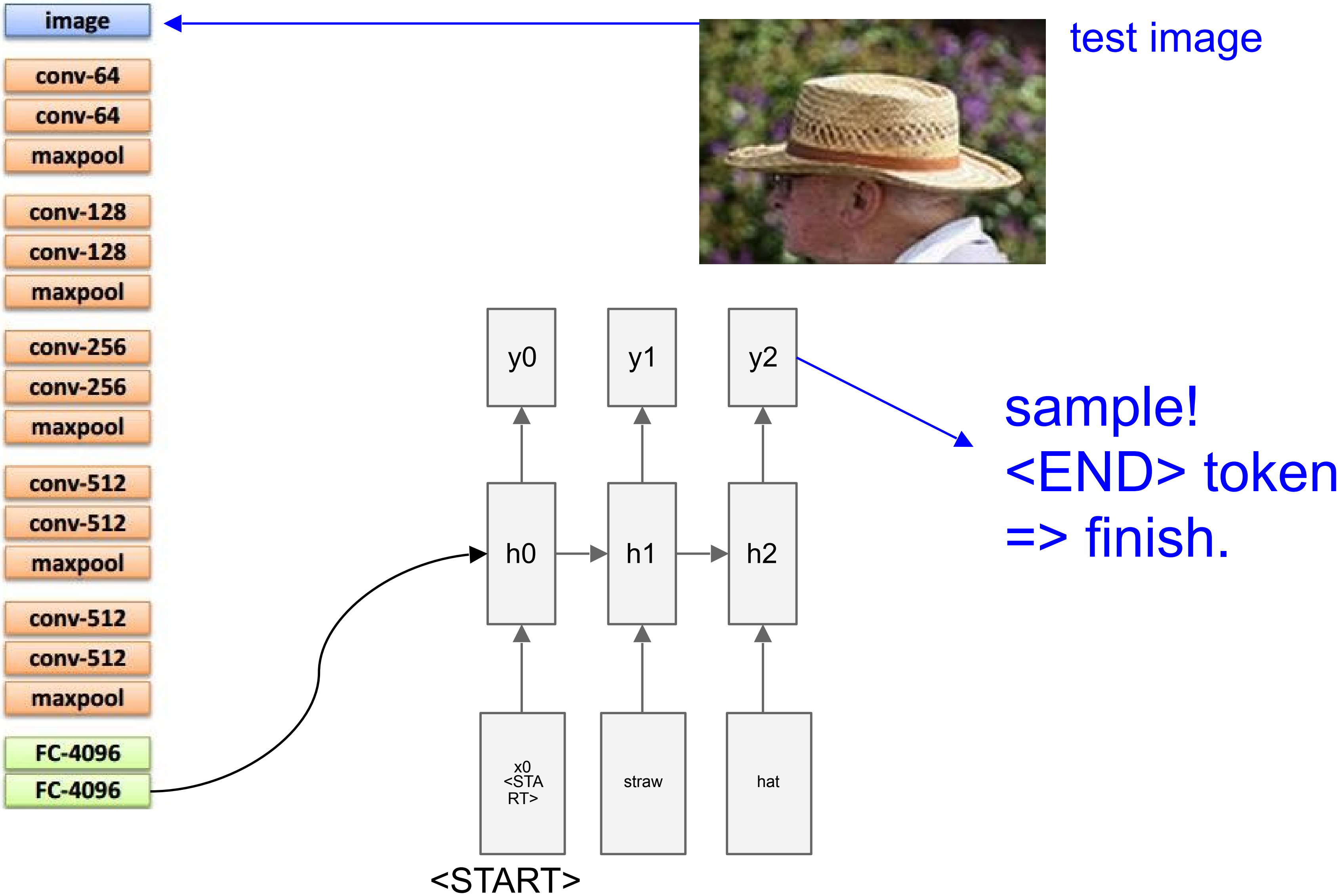
slide credit: Andrej Karpathy



slide credit: Andrej Karpathy



slide credit: Andrej Karpathy



slide credit: Jeff Donahue

Wow I can’t believe that worked

a group of people standing around a room with remotes
`logprob: -9.17`

a young boy is holding a baseball bat
`logprob: -7.61`

a cow is standing in the middle of a street
`logprob: -8.84`

slide credit: Jeff Donahue

Wow I can't believe that worked


a cat is sitting on a toilet seat
logprob: -7.79


a display case filled with lots of different types of donuts
logprob: -7.78


a group of people sitting at a table with wine glasses
logprob: -6.71

Well, I can kind of see it

a man standing next to a clock on a wall
`logprob: -10.08`

a young boy is holding a baseball bat
`logprob: -7.65`

a cat is sitting on a couch with a remote control
`logprob: -12.45`

slide credit: Jeff Donahue

Well, I can kind of see it

a baby laying on a bed with a stuffed bear
logprob: -8.66

a table with a plate of food and a cup of coffee
logprob: -9.93

a young boy is playing frisbee in the park
logprob: -9.52

slide credit: Jeff Donahue

Not sure what happened there...

a toilet with a seat up in a bathroom
logprob: -13.44

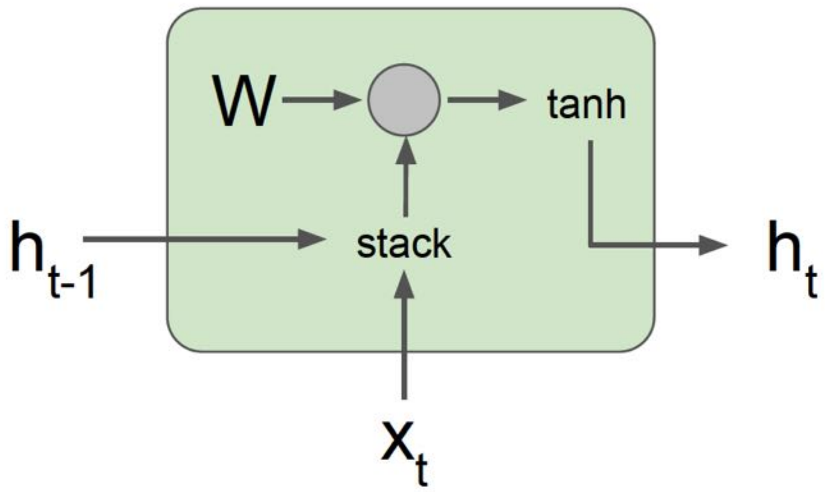
a woman holding a teddy bear in front of a mirror
logprob: -9.65

a horse is standing in the middle of a road
logprob: -10.34

Vanilla RNN...

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



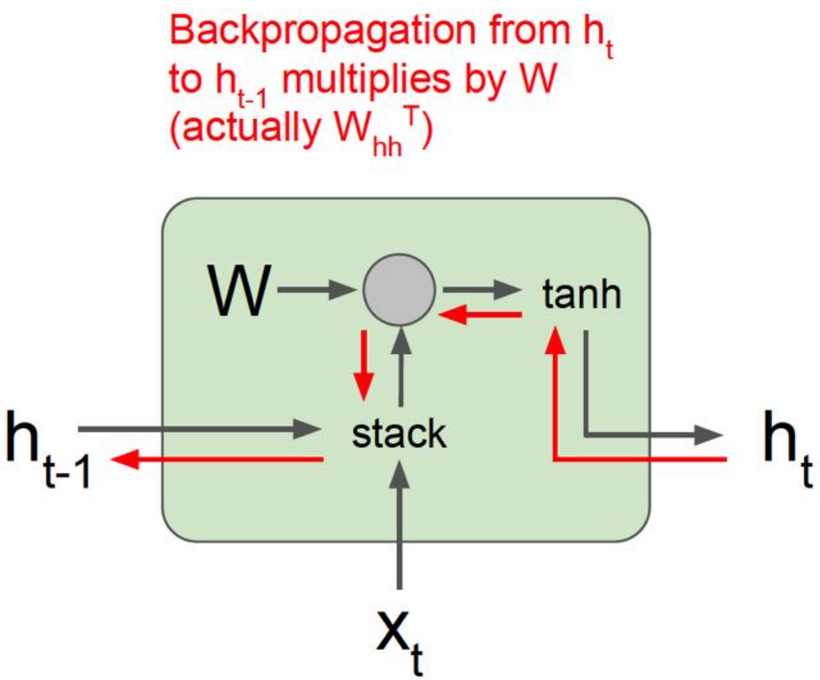
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Vanilla RNN...

Vanilla RNN Gradient Flow

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013



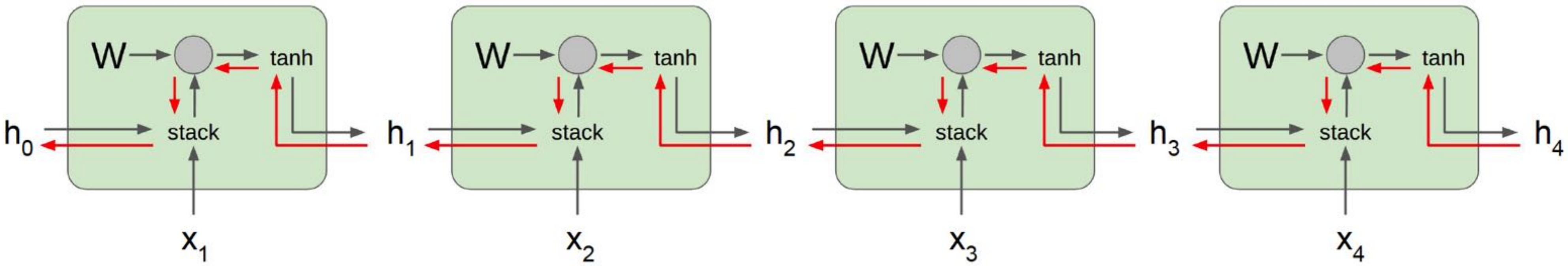
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Vanilla RNN...

Vanilla RNN Gradient Flow

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013



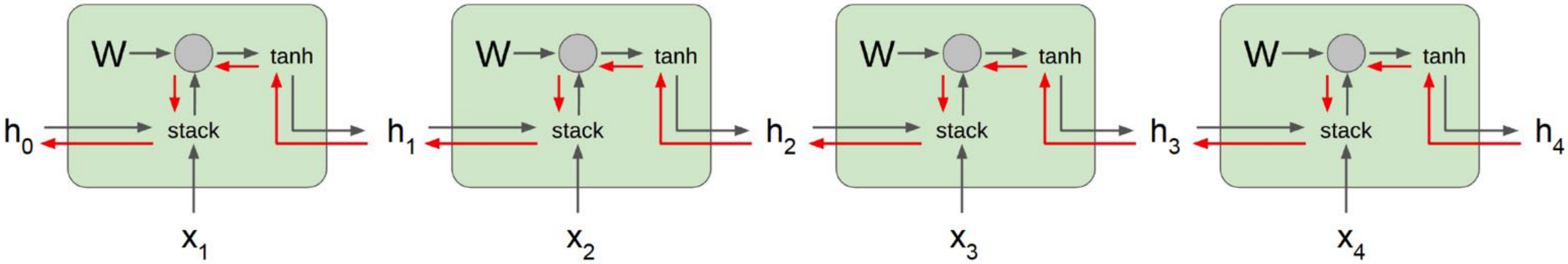
Computing gradient of h_0 involves many factors of W (and repeated tanh)

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Vanilla RNN...

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

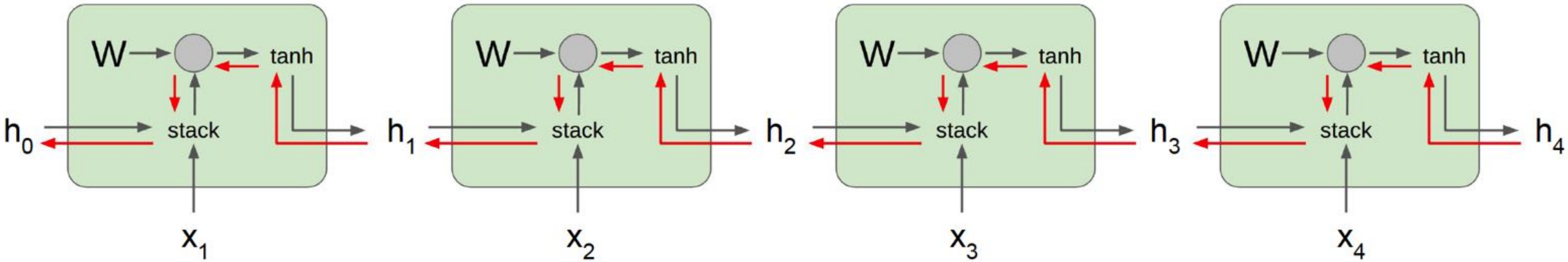
Largest singular value < 1:
Vanishing gradients

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Vanilla RNN...

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1: **Exploding gradients**

Largest singular value < 1: **Vanishing gradients**

→ **Gradient clipping**: Scale gradient if its norm is too big

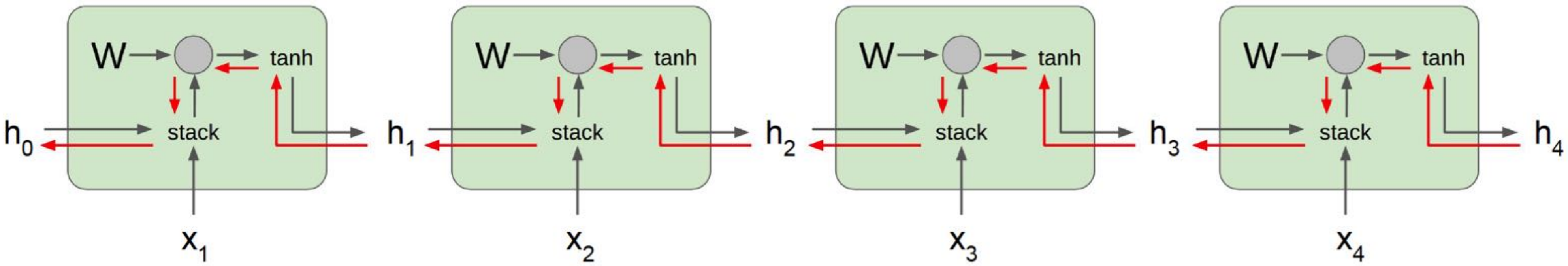
```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Vanilla RNN...

Vanilla RNN Gradient Flow

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

→ Change RNN architecture

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

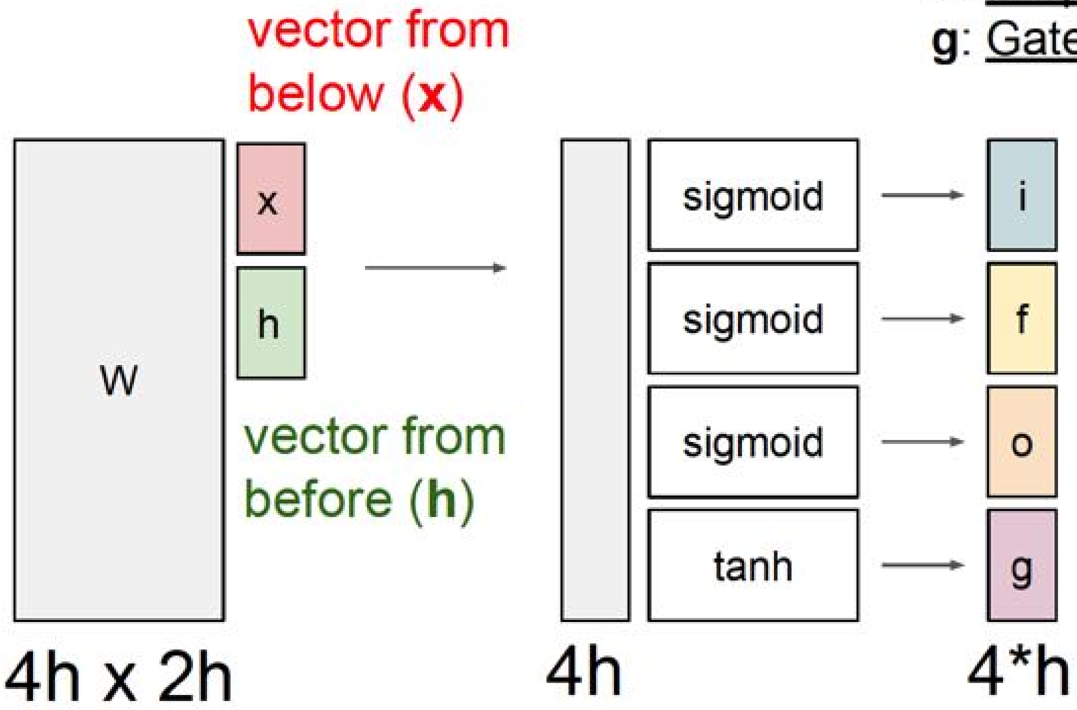
Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]

i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell
o: Output gate, How much to reveal cell
g: Gate gate (?), How much to write to cell



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

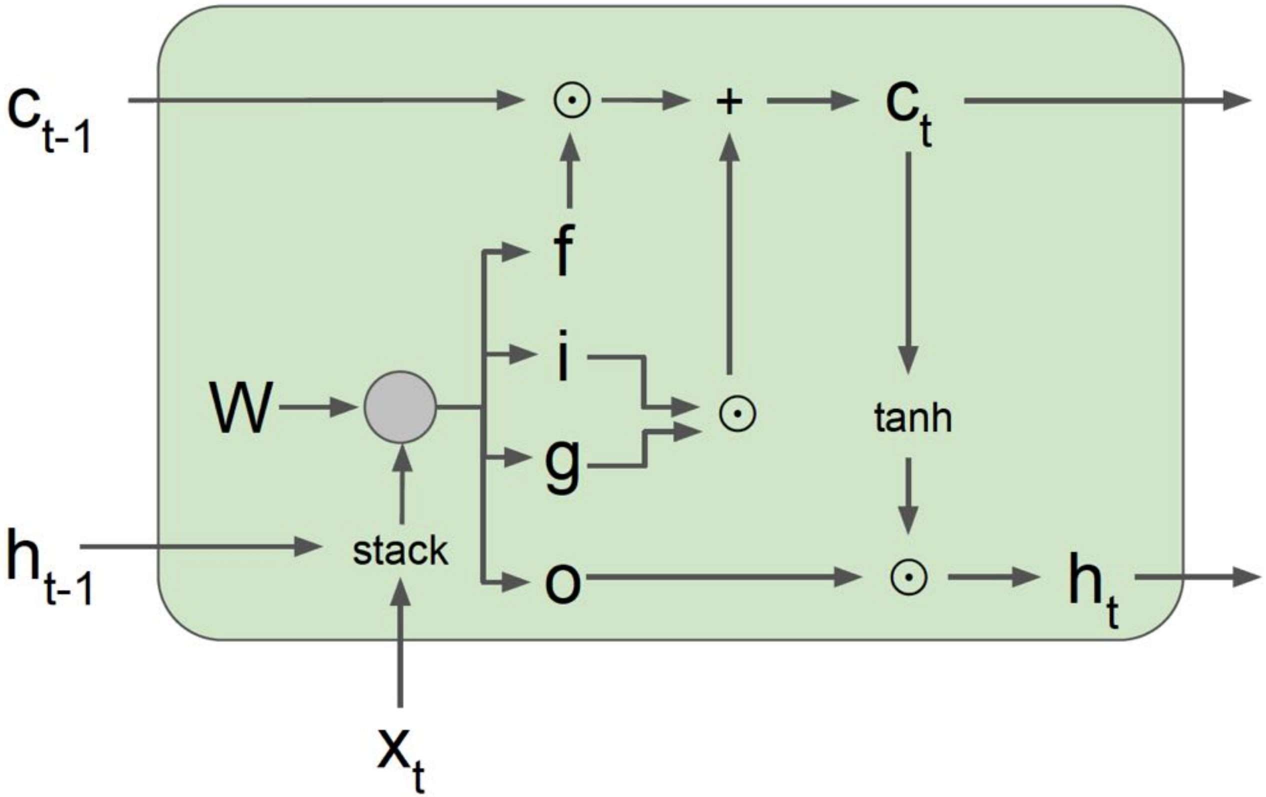
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

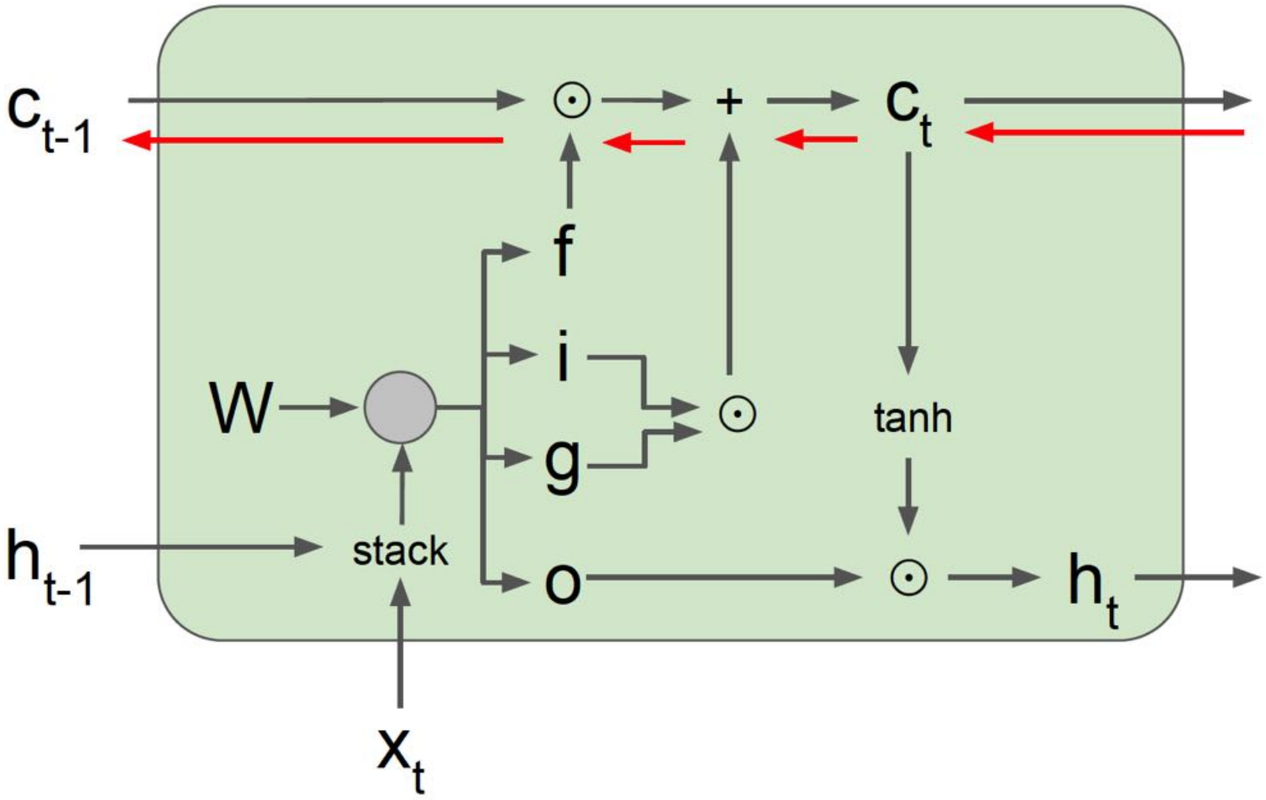
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Long Short Term Memory (LSTM): Gradient Flow

Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

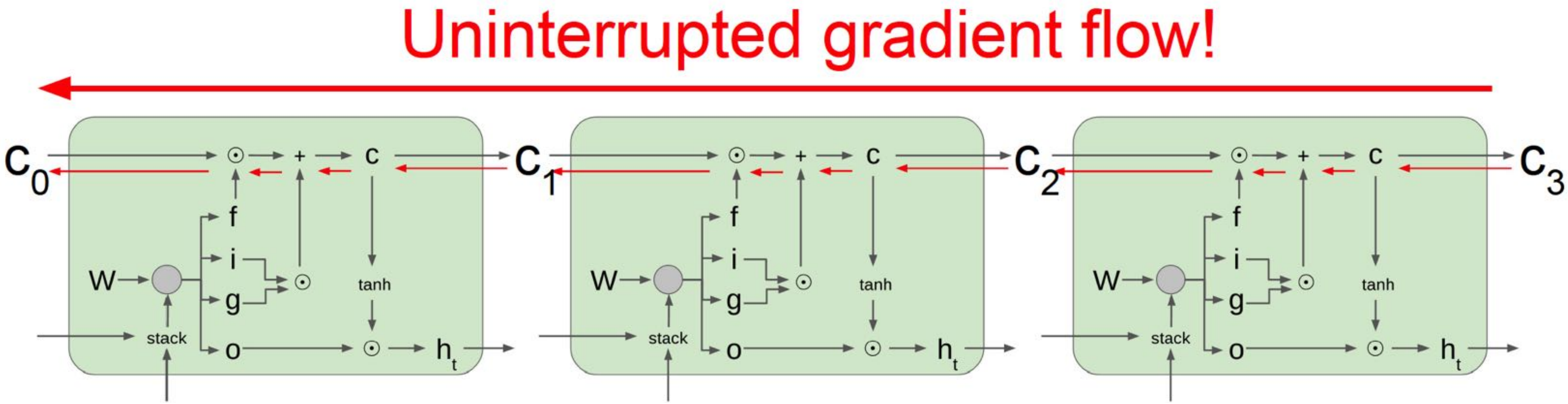
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Long Short Term Memory (LSTM): Gradient Flow

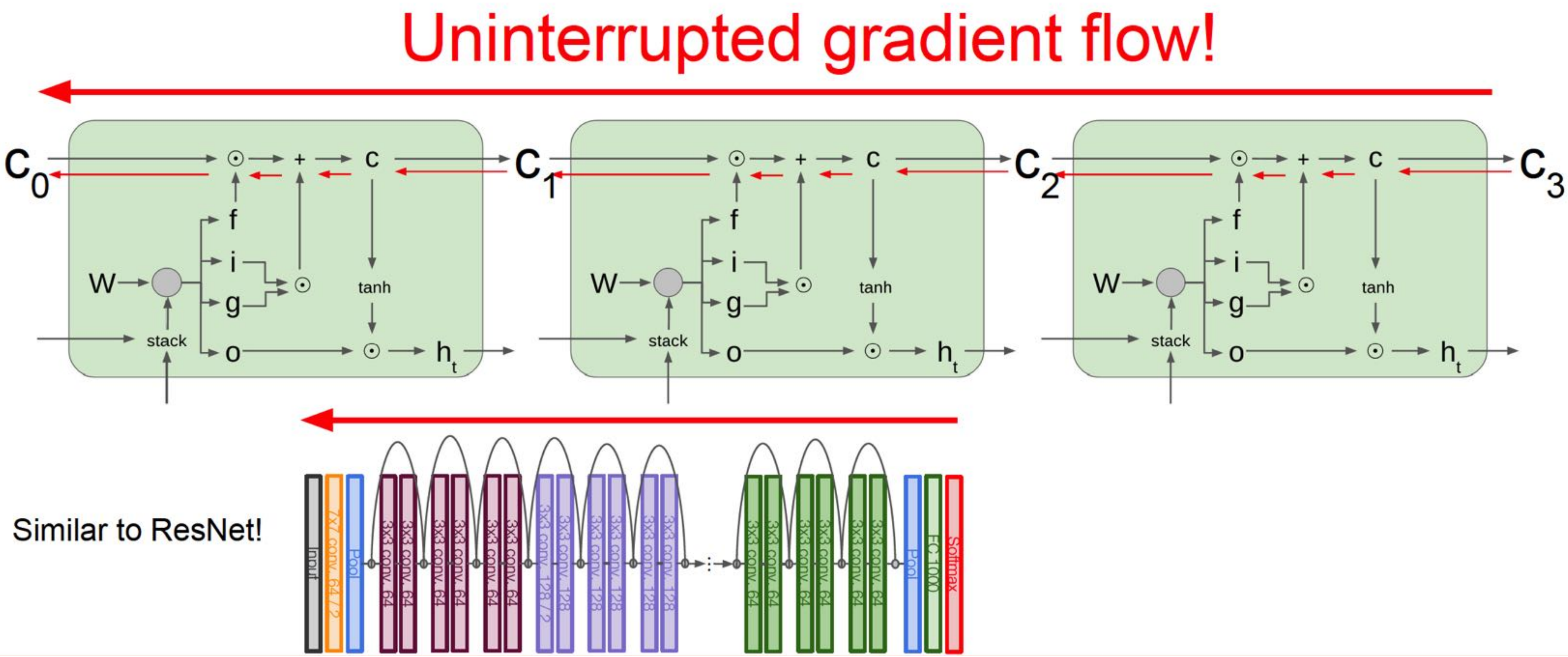
Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Long Short Term Memory (LSTM): Gradient Flow

Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]



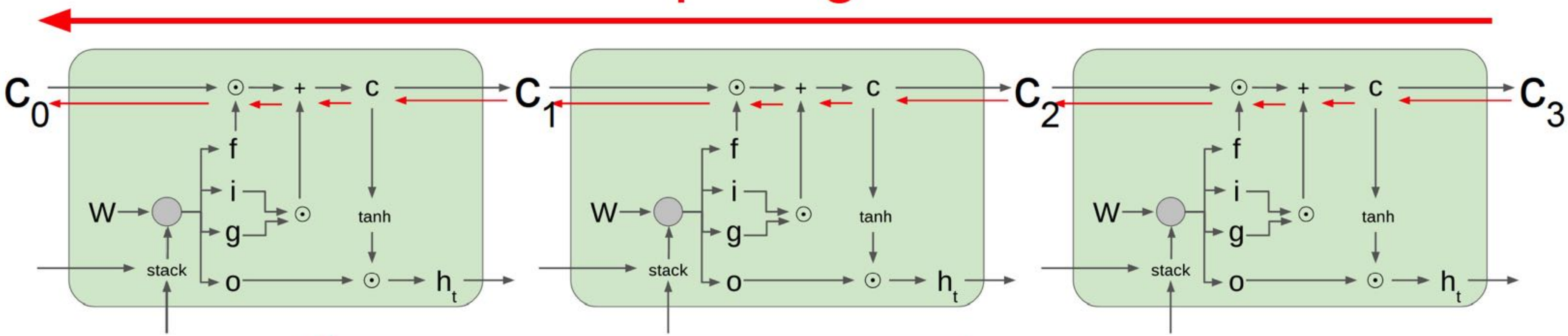
Similar to ResNet!

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

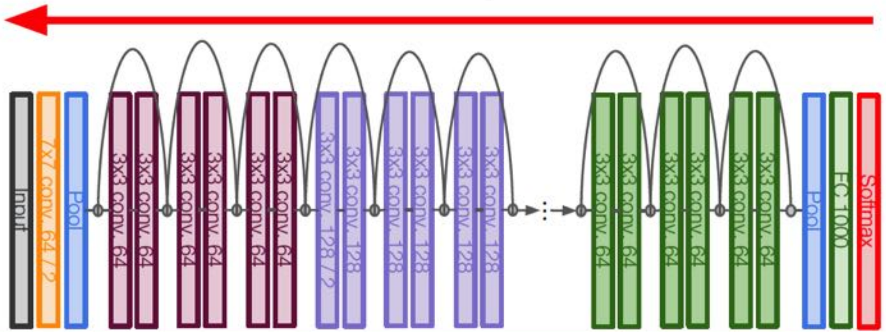
Long Short Term Memory (LSTM): Gradient Flow

Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]

Uninterrupted gradient flow!



Similar to ResNet!



In between:
Highway Networks

$$g = T(x, W_T)$$

$$y = g \odot H(x, W_H) + (1 - g) \odot x$$

Srivastava et al, "Highway Networks", ICML DL Workshop 2015

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Alternatives

Other RNN Variants

GRU [*Learning phrase representations using rnn encoder-decoder for statistical machine translation*, Cho et al. 2014]

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

[*LSTM: A Search Space Odyssey*, Greff et al., 2015]

[*An Empirical Exploration of Recurrent Network Architectures*, Jozefowicz et al., 2015]

MUT1:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}}x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + \tanh(x_t) + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT2:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + W_{\mathrm{hz}}h_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh}x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT3:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + W_{\mathrm{hz}}\tanh(h_t) + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}}x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh}x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Summary

- RNNs allow a lot of flexibility in architecture design
- Vanilla RNNs are simple but don't work very well
- Common to use LSTM or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish. Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)
- Better/simpler architectures are a hot topic of current research
- Better understanding (both theoretical and empirical) is needed.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

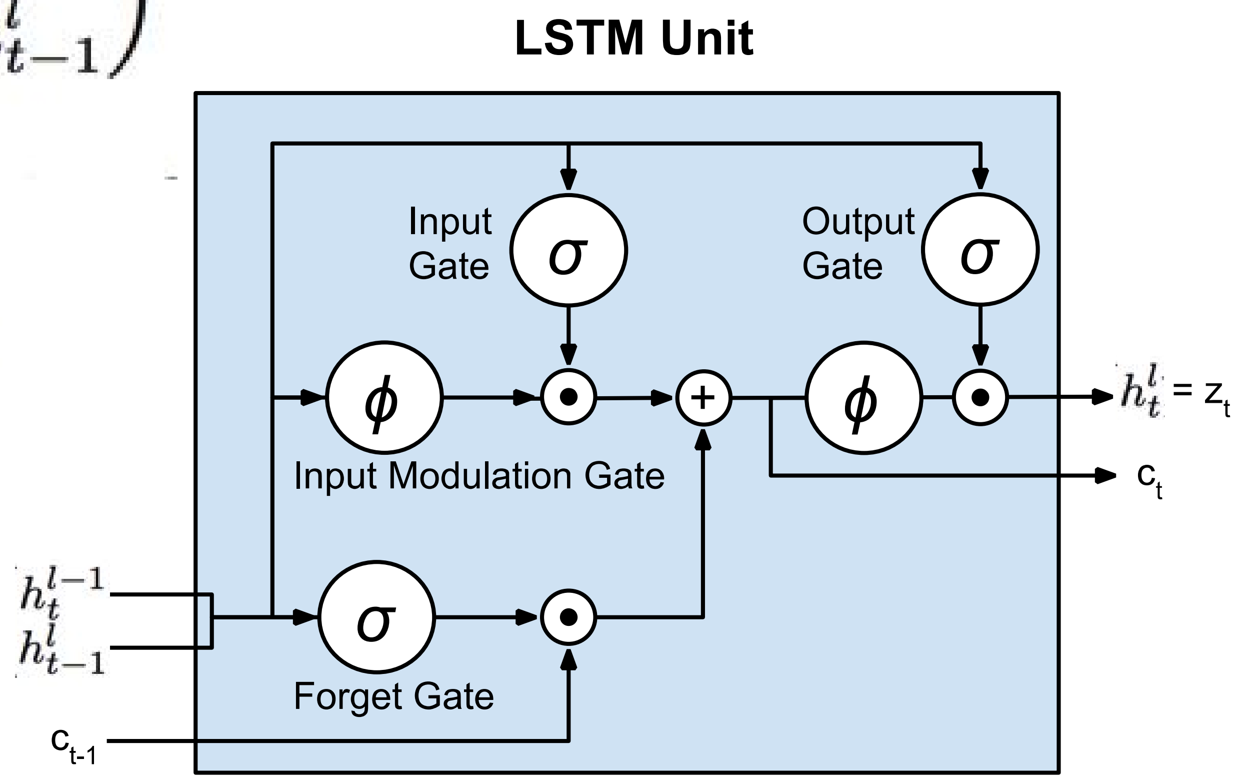
Visualizing and Understanding Recurrent Networks
Andrej Karpathy*, Justin Johnson*, Li Fei-Fei
(on arXiv.org)

Hunting interpretable cells

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

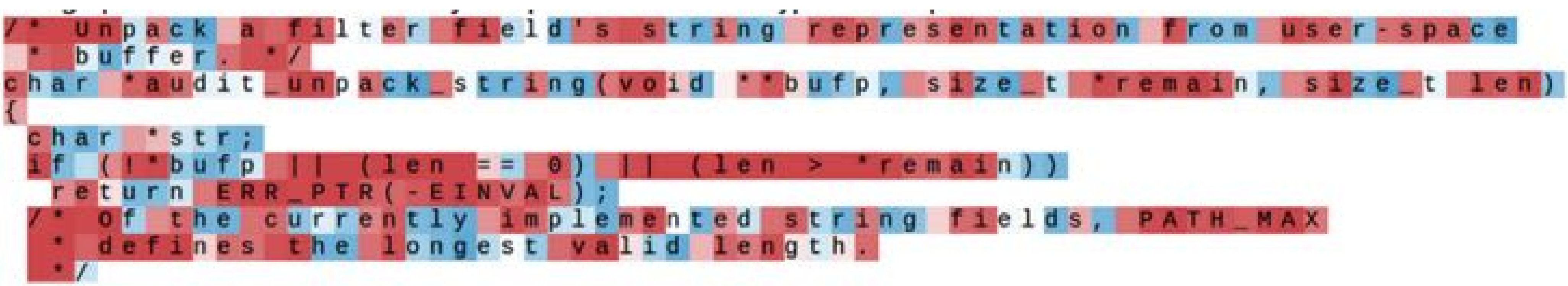
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$



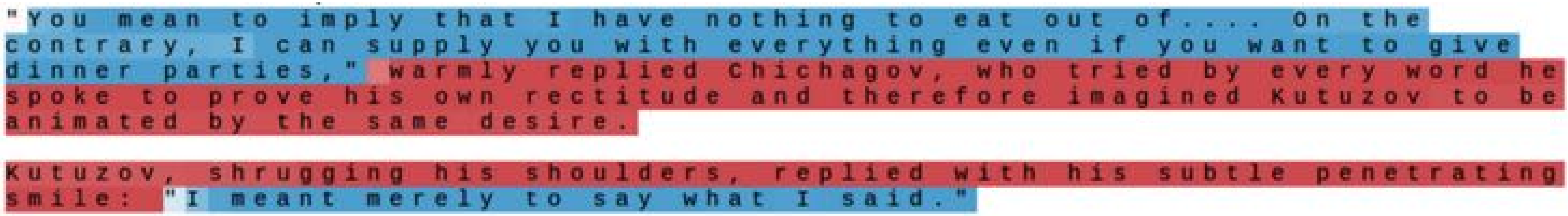
slide credit: Andrej Karpathy

Hunting interpretable cells



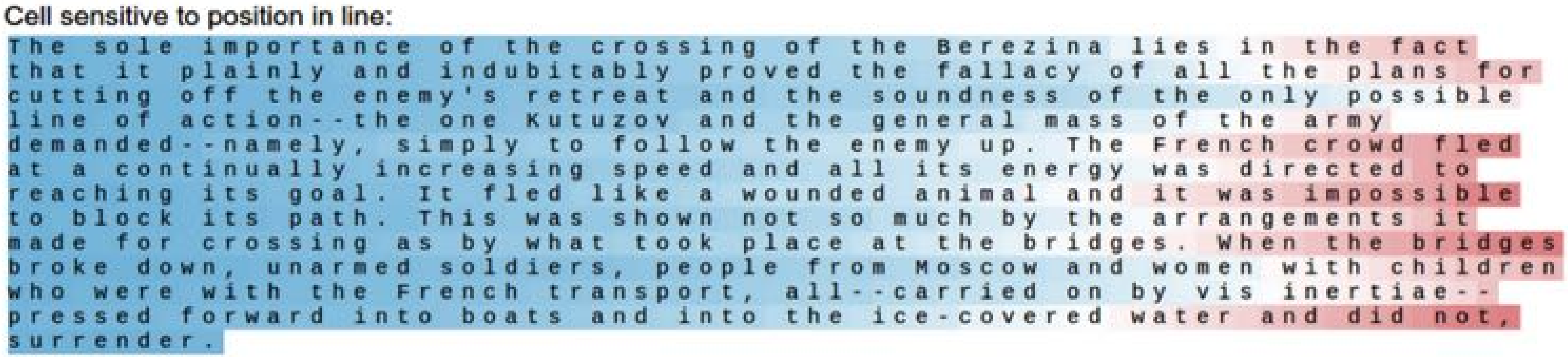
slide credit: Andrej Karpathy

Hunting interpretable cells



quote detection cell

Hunting interpretable cells



line length tracking cell

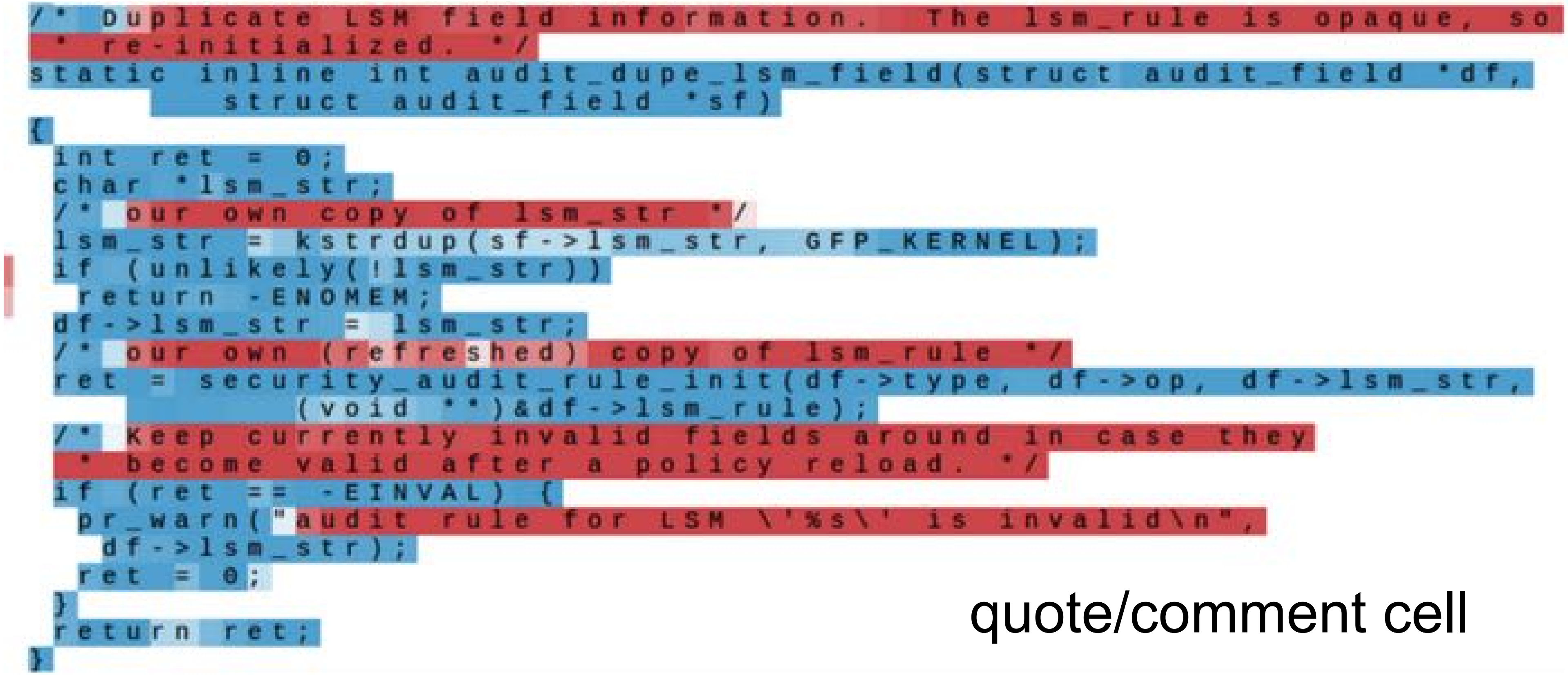
slide credit: Andrej Karpathy

Hunting interpretable cells

if statement cell

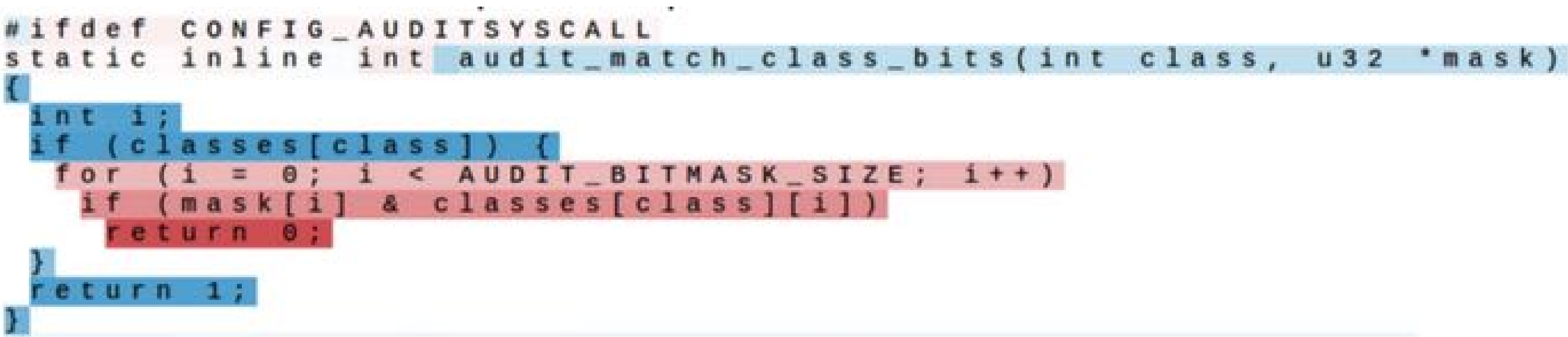
slide credit: Andrej Karpathy

Hunting interpretable cells



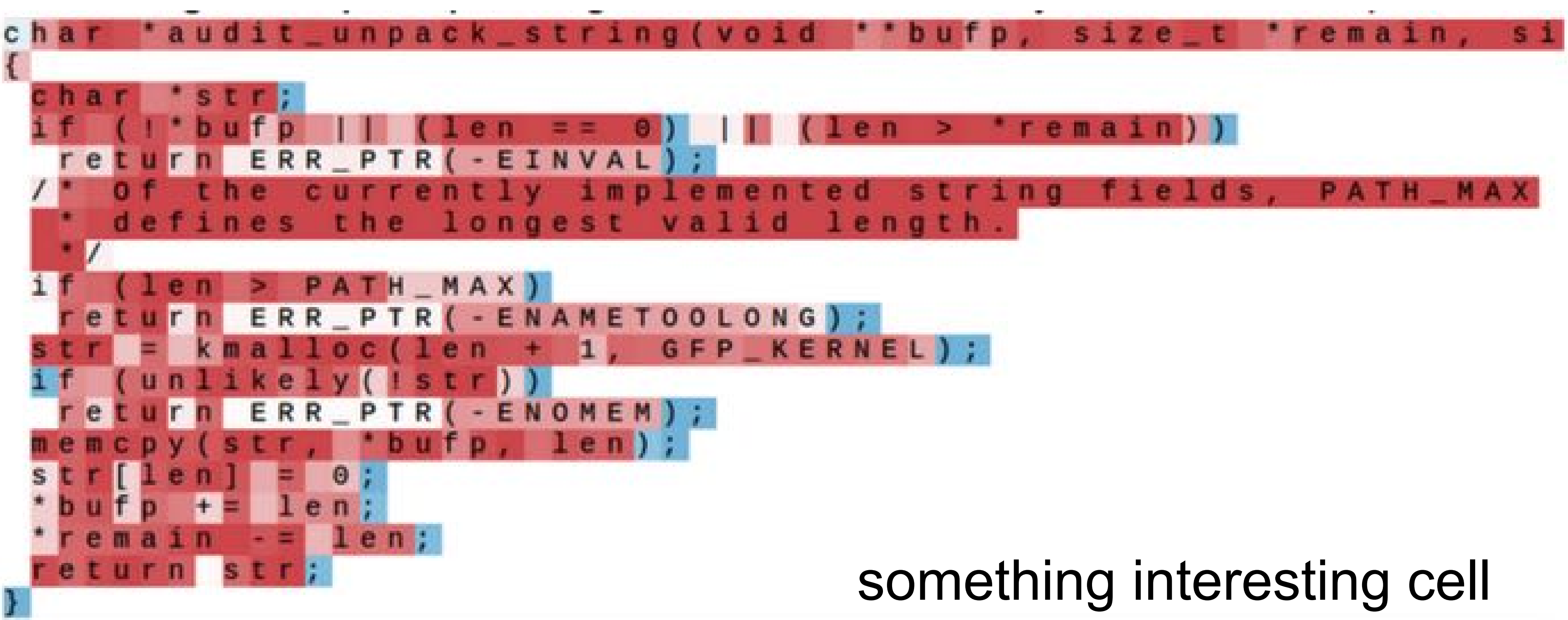
quote/comment cell

Hunting interpretable cells



code depth cell

Hunting interpretable cells



something interesting cell
(not quite sure what)