High Level Computer Vision

Deep Learning for Computer Vision
Part 4

Bernt Schiele - schiele@mpi-inf.mpg.de
Mario Fritz - mfritz@mpi-inf.mpg.de

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

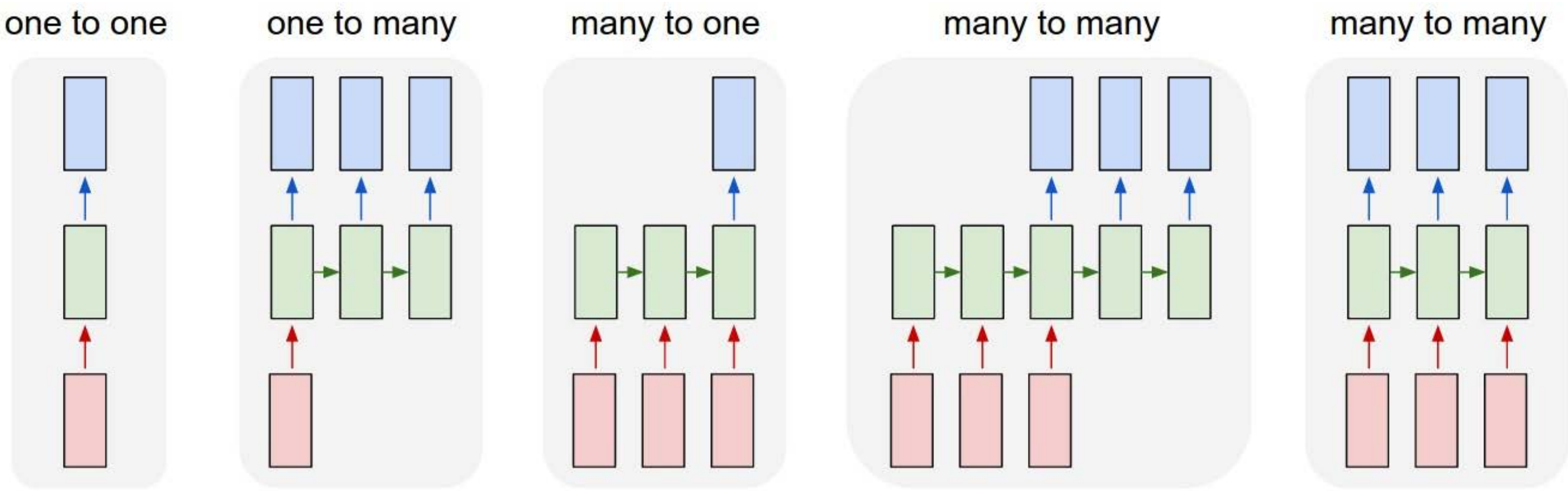
Overview

- Recurrent Neural Networks
  - motivation for recurrent neural networks
  - a particularly successful RNN: Long Short Term Memory (LSTM)
  - slide credit to Andrej Karpathy, Jeff Donahue and Marcus Rohrbach

- Yann LeCun…
  - What's Wrong With Deep Learning (keynote June 2015)
  - slide credit to Yann LeCun (and Xiaogang Wang)

slide credit: Andrej Karpathy

Recurrent Networks offer a lot of flexibility:

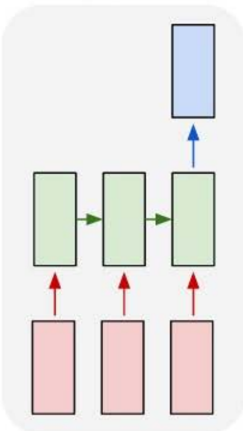


Sequences in Vision

Sequences in the input…
(many-to-one)

- ☐ Running
- ☑ Jumping
- ☐ Dancing
- ☐ Fighting
- ☐ Eating

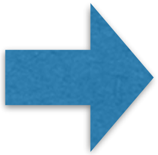
many to one



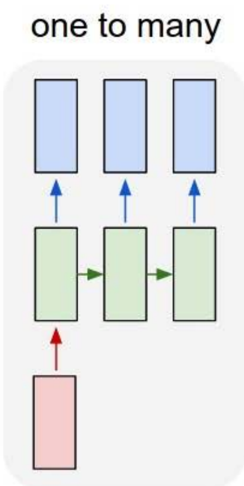
slide credit: Jeff Donahue

Sequences in Vision

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



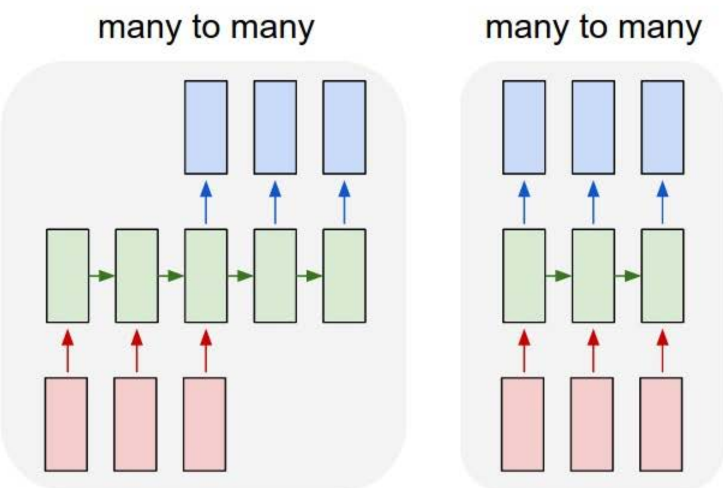
A happy brown dog.



Sequences in Vision

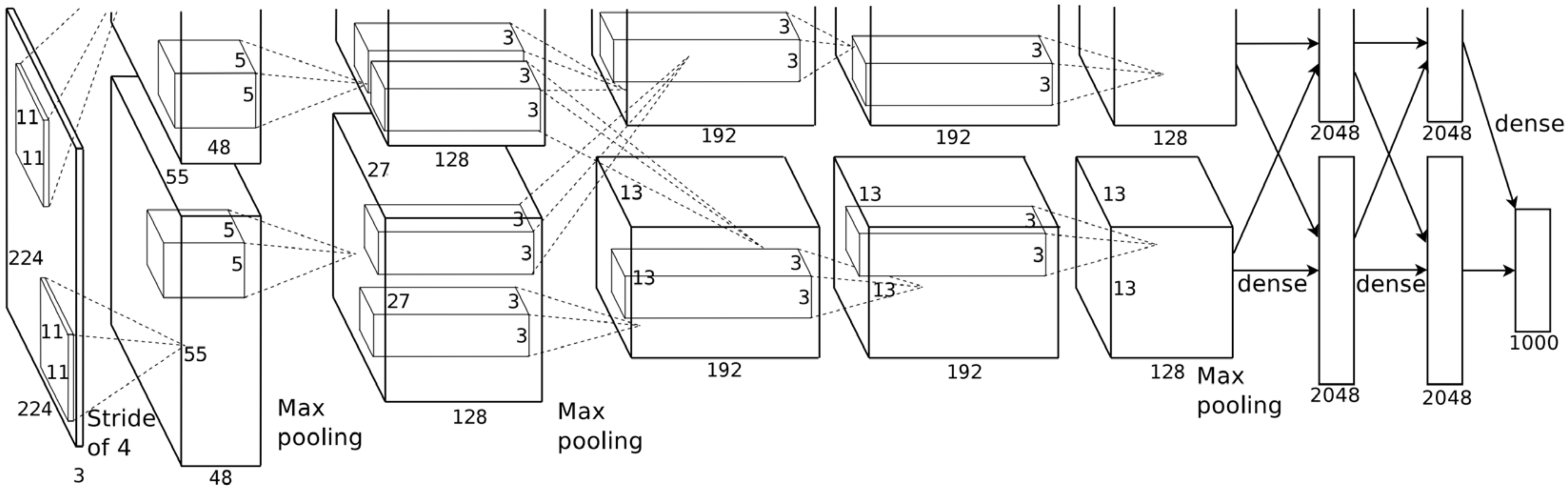
Sequences everywhere!
(many-to-many)

A dog jumps over a hurdle.



slide credit: Jeff Donahue

ConvNets

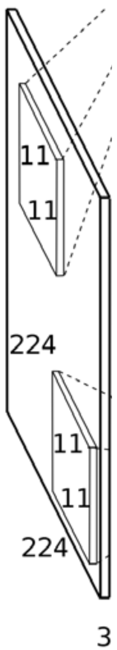


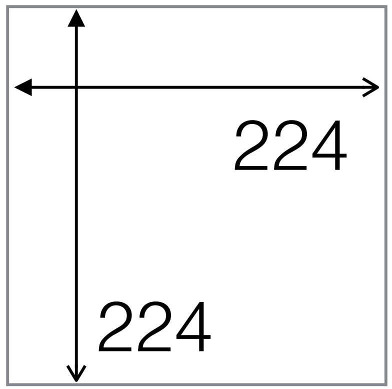
Krizhevsky et al., NIPS 2012

slide credit: Jeff Donahue

Problem #1

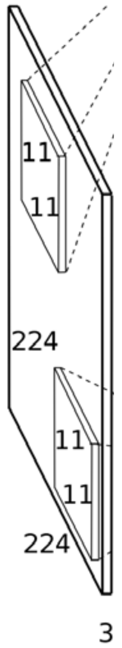
fixed-size, static input





slide credit: Jeff Donahue

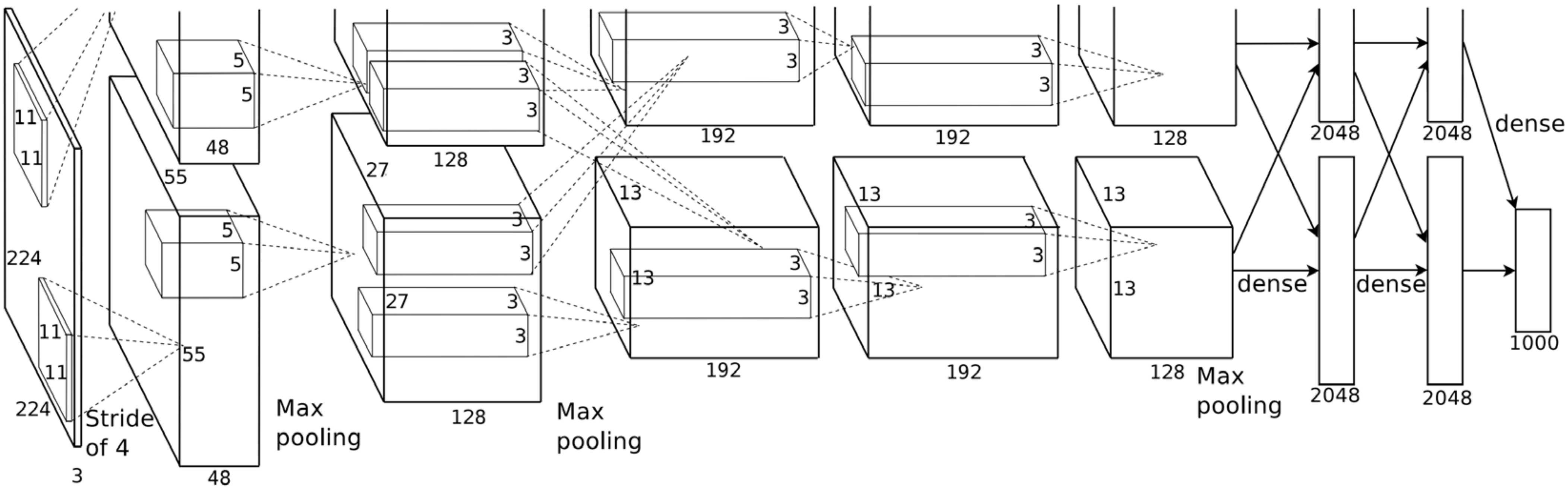
Problem #1



fixed-size, static input

slide credit: Jeff Donahue

Problem #2

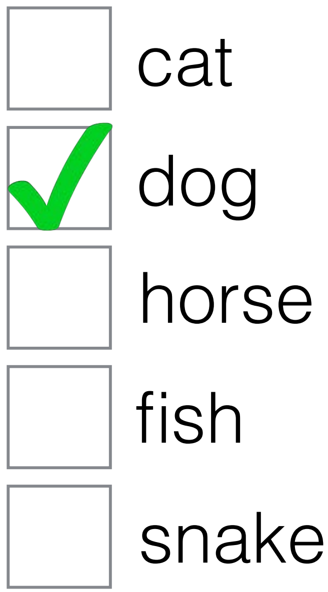


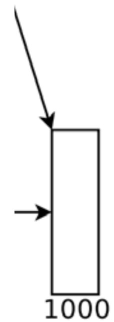
Krizhevsky et al., NIPS 2012

slide credit: Jeff Donahue

Problem #2

output is a single choice from a fixed list of options



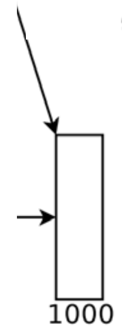


slide credit: Jeff Donahue

Problem #2

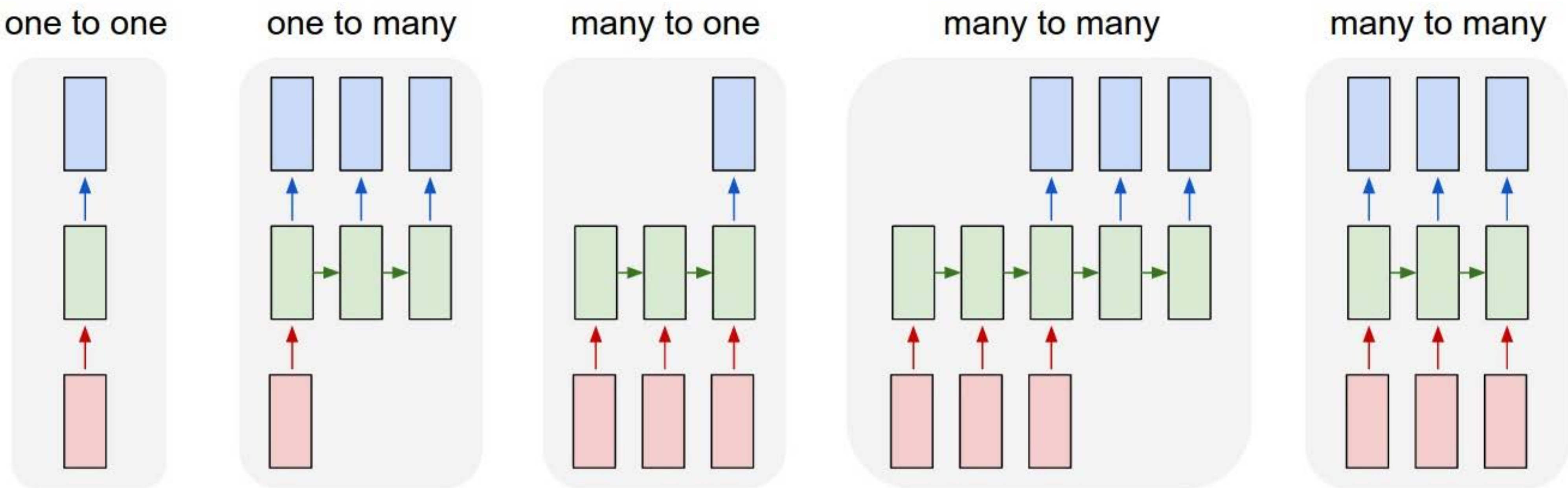
output is a single choice from a fixed list of options

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

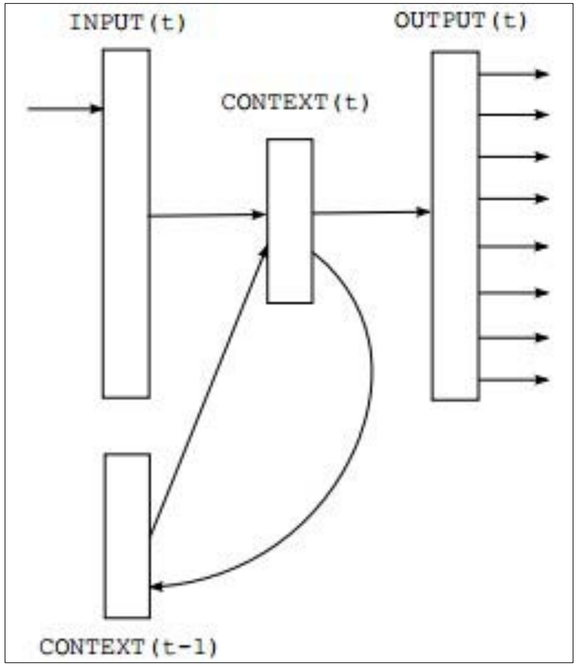


slide credit: Andrej Karpathy

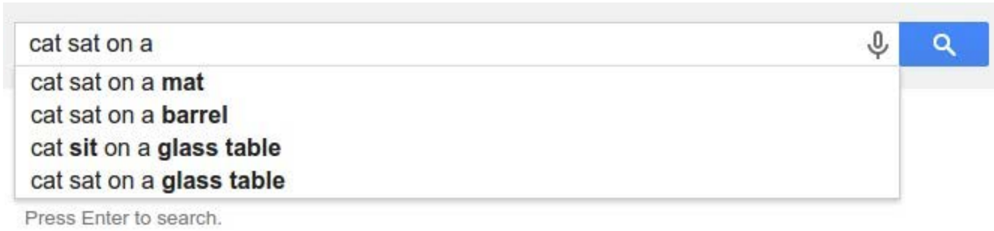
Recurrent Networks offer a lot of flexibility:



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

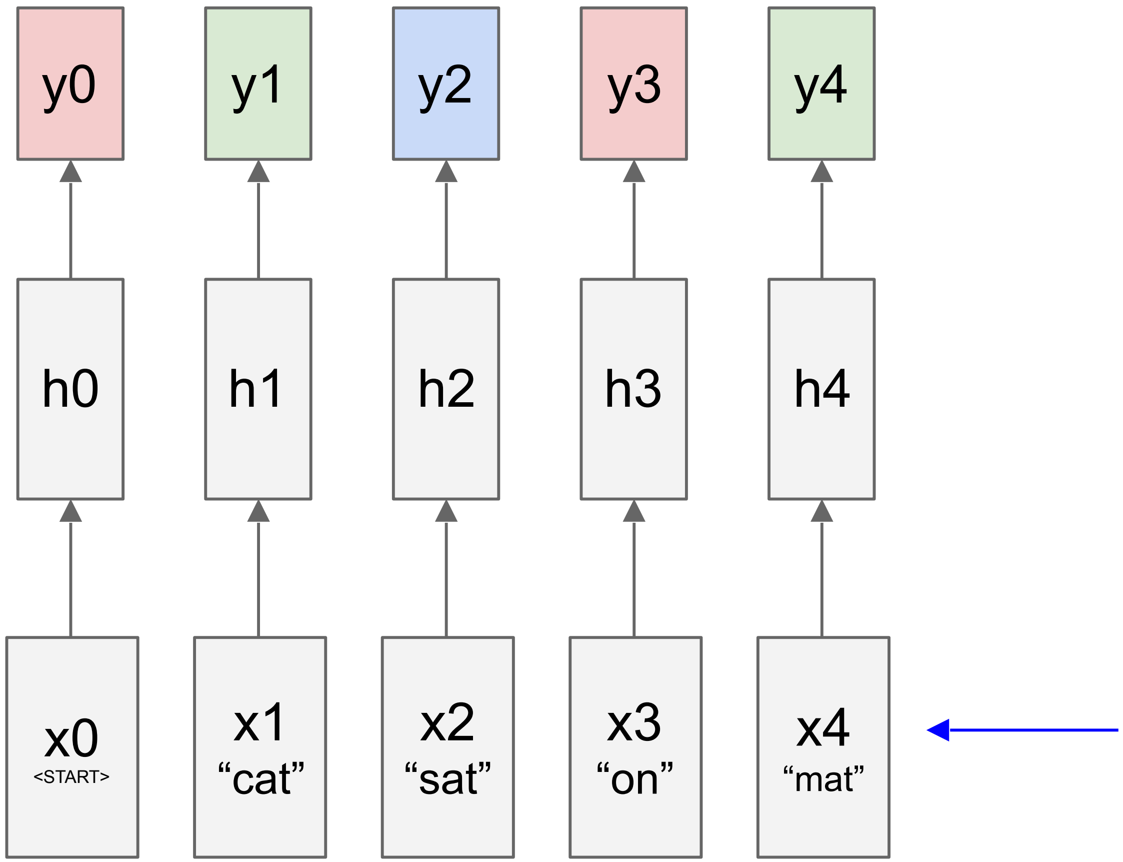
Suppose we had the training sentence “cat sat on mat”

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

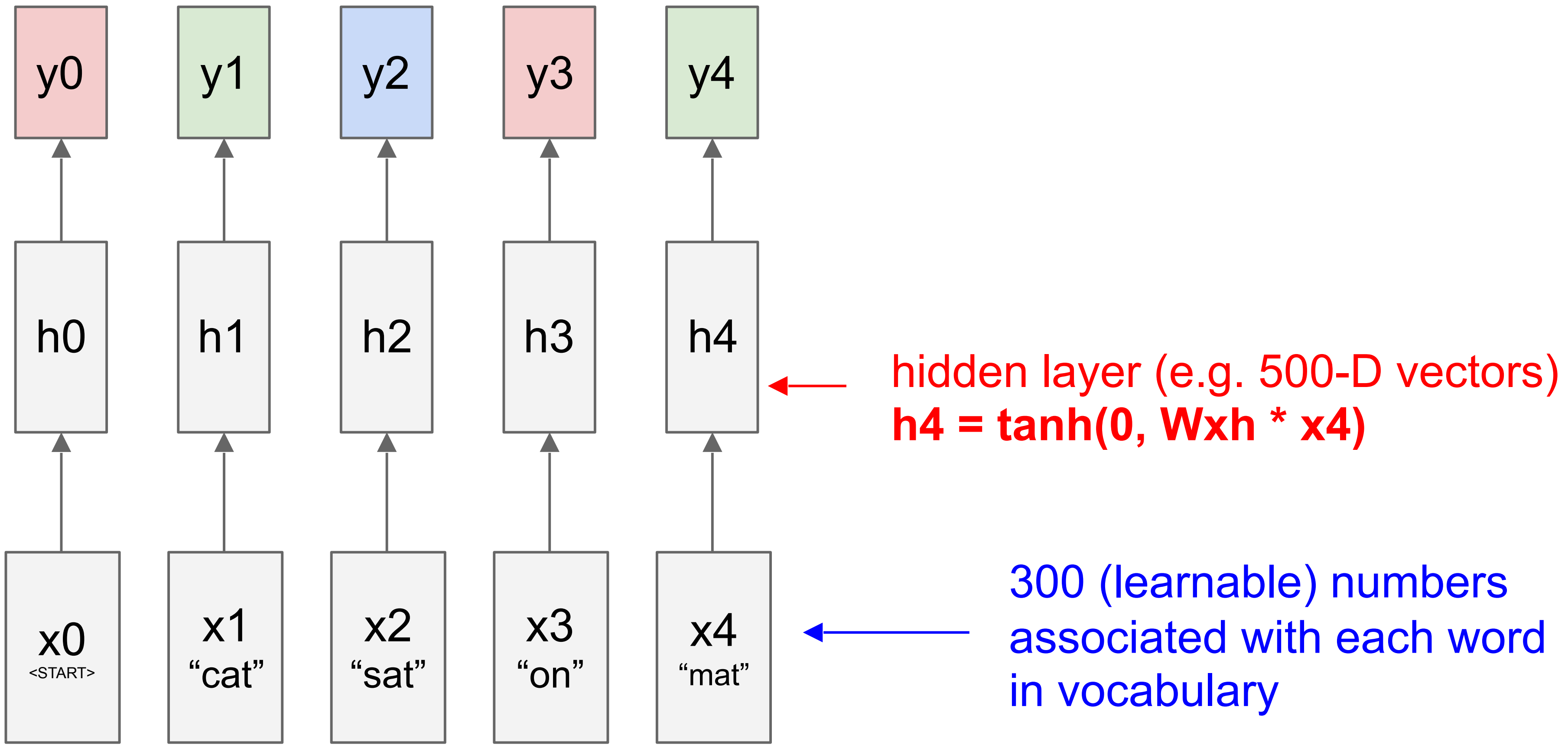
"cat sat on mat"



300 (learnable) numbers associated with each word in vocabulary

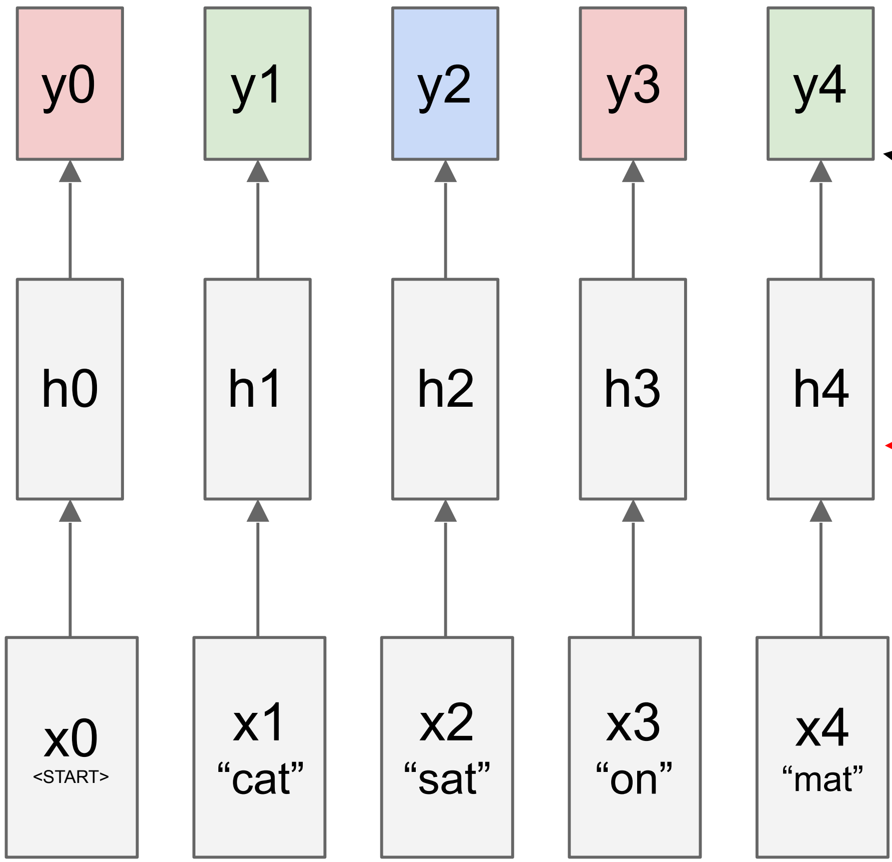
slide credit: Andrej Karpathy

"cat sat on mat"



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

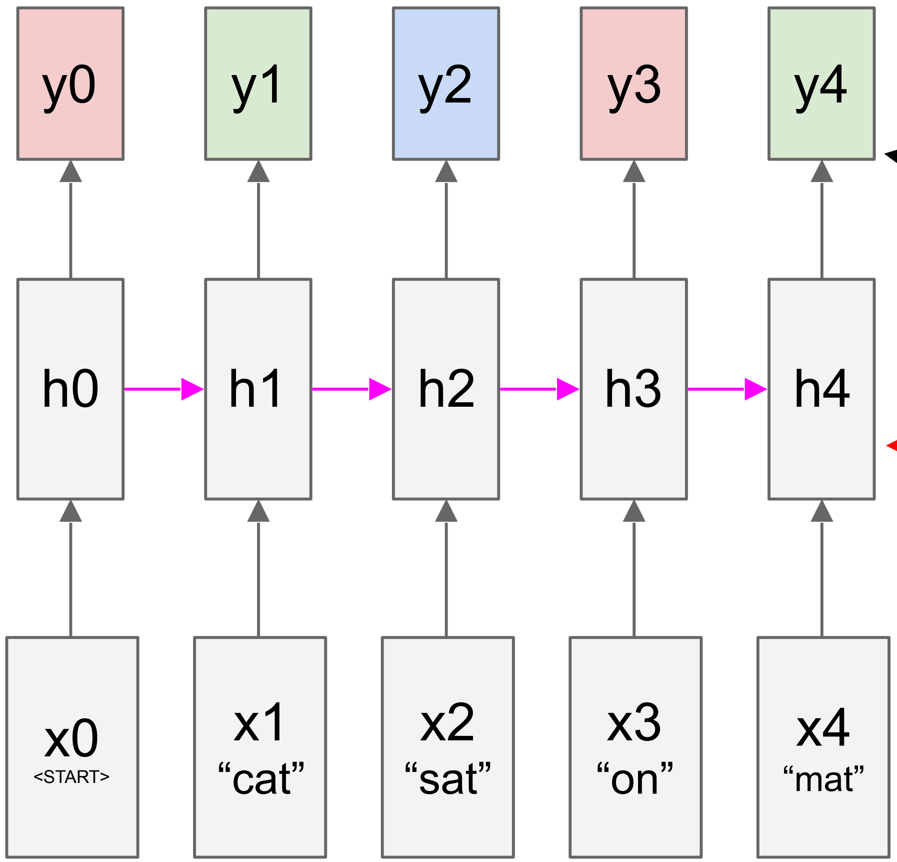
hidden layer (e.g. 500-D vectors)
h4 = tanh(0, Wxh * x4)

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

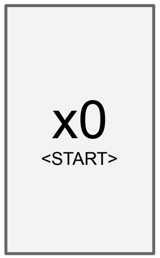
hidden layer (e.g. 500-D vectors)
h4 = tanh(0, Wxh * x4 + Whh * h3)

300 (learnable) numbers
associated with each word
in vocabulary

slide credit: Andrej Karpathy

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

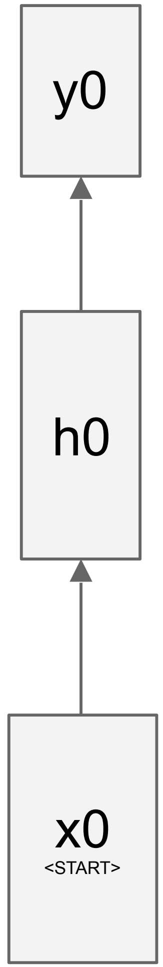
P(next word | previous words)



slide credit: Andrej Karpathy

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

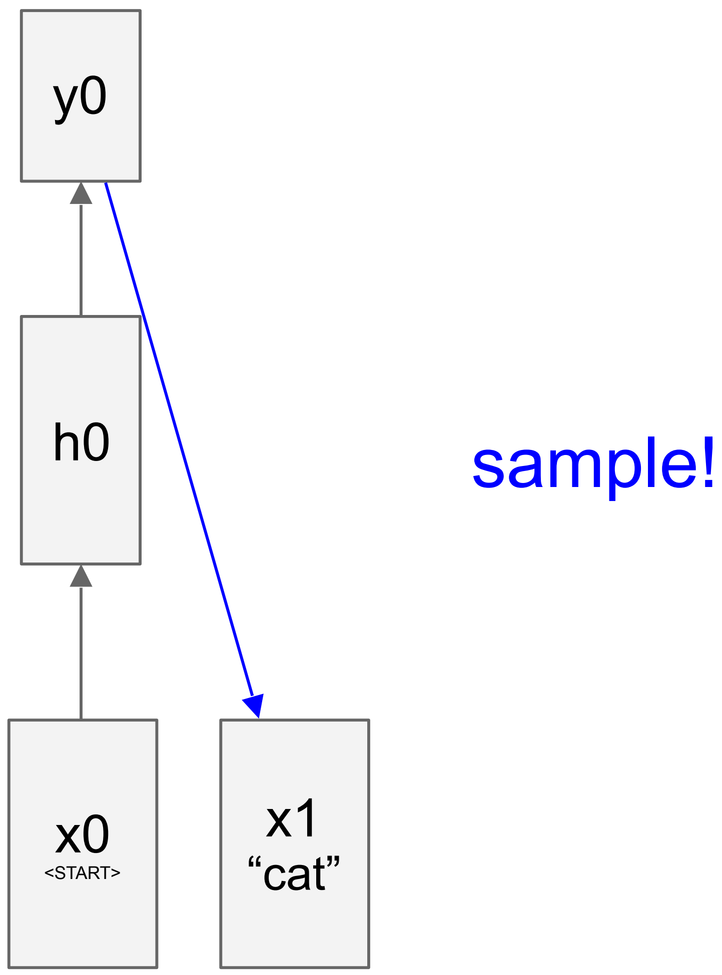
P(next word | previous words)



slide credit: Andrej Karpathy

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

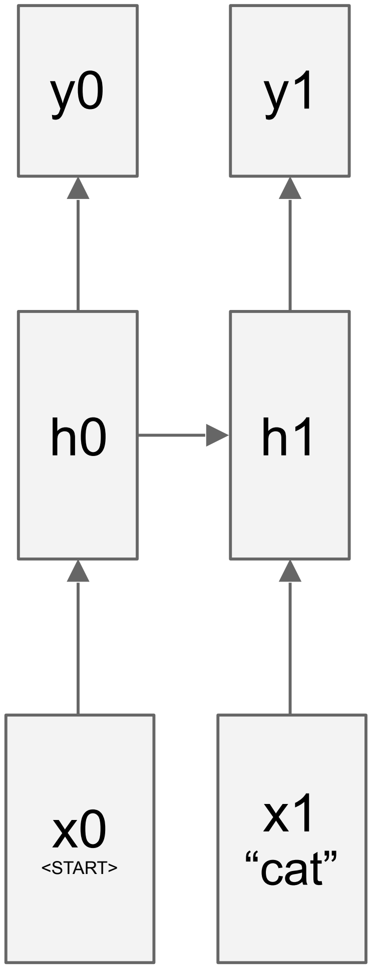
P(next word | previous words)



slide credit: Andrej Karpathy

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

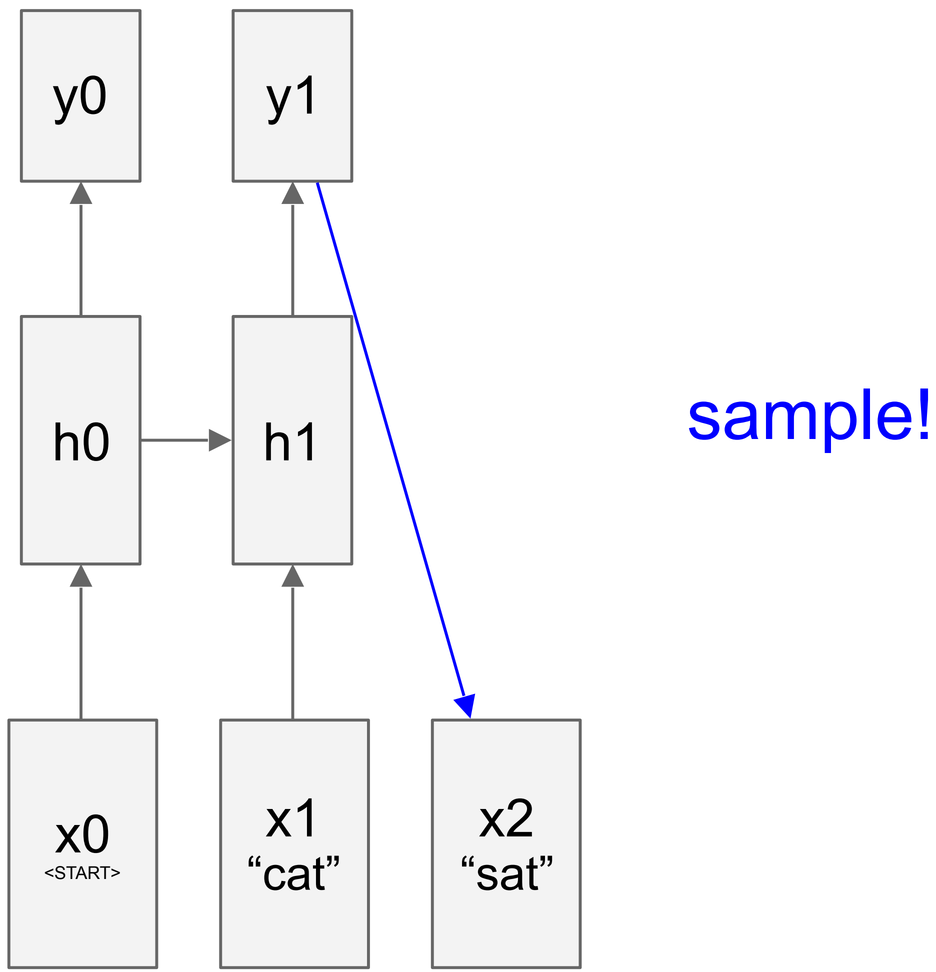
P(next word | previous words)



slide credit: Andrej Karpathy

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

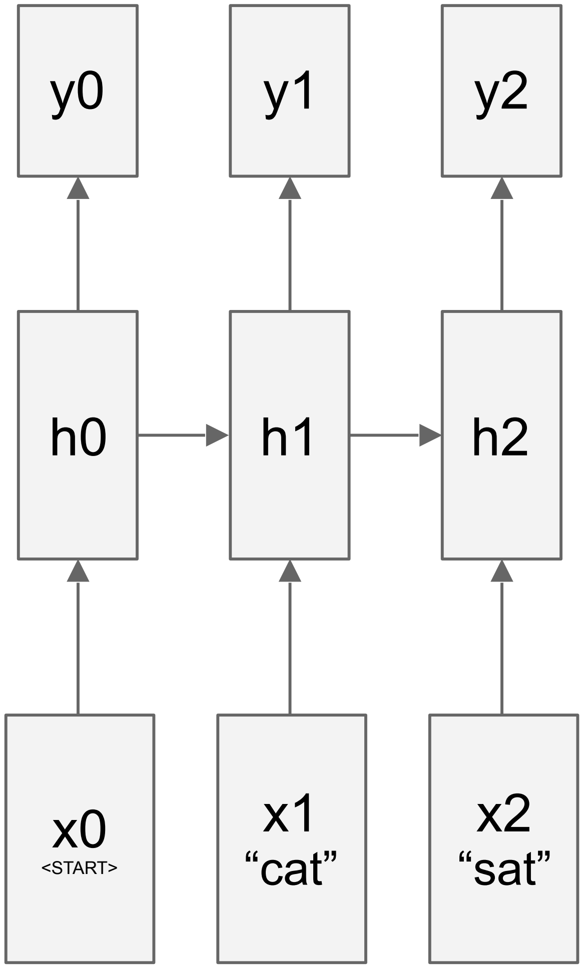
P(next word | previous words)



slide credit: Andrej Karpathy

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

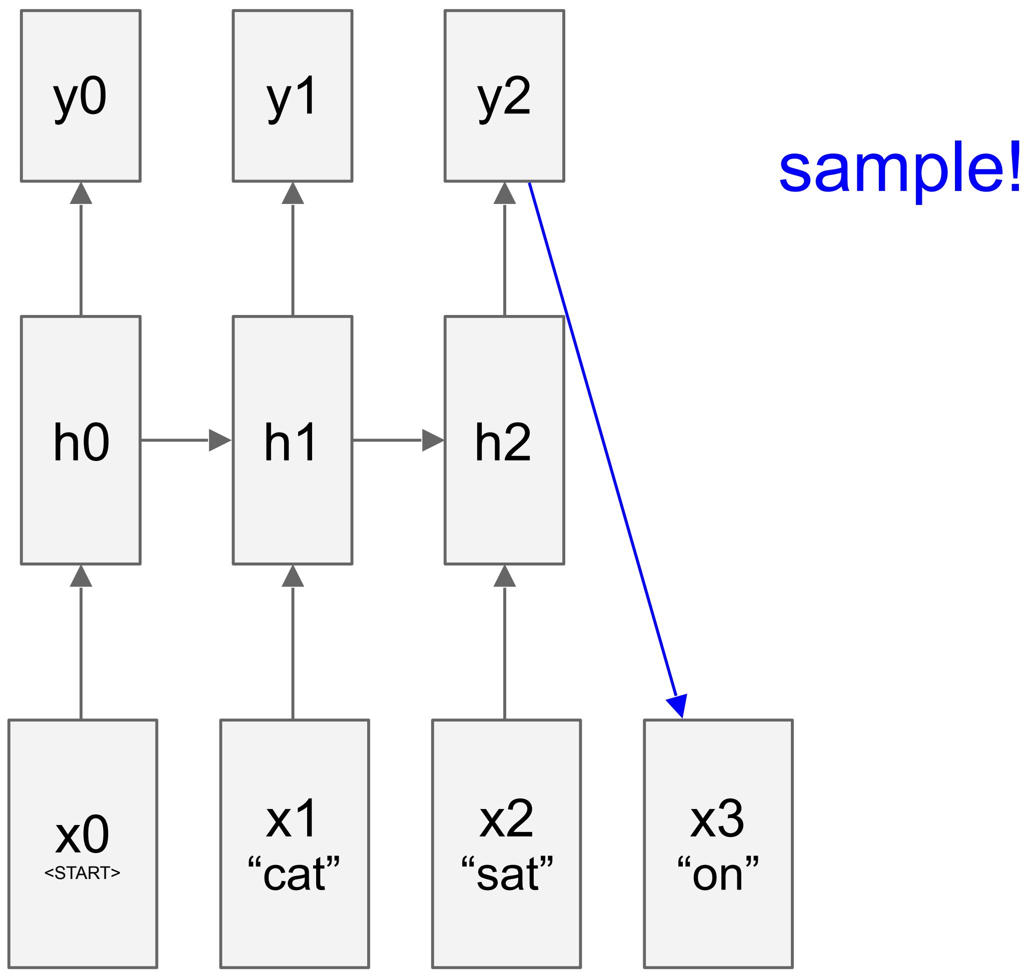
P(next word | previous words)



slide credit: Andrej Karpathy

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

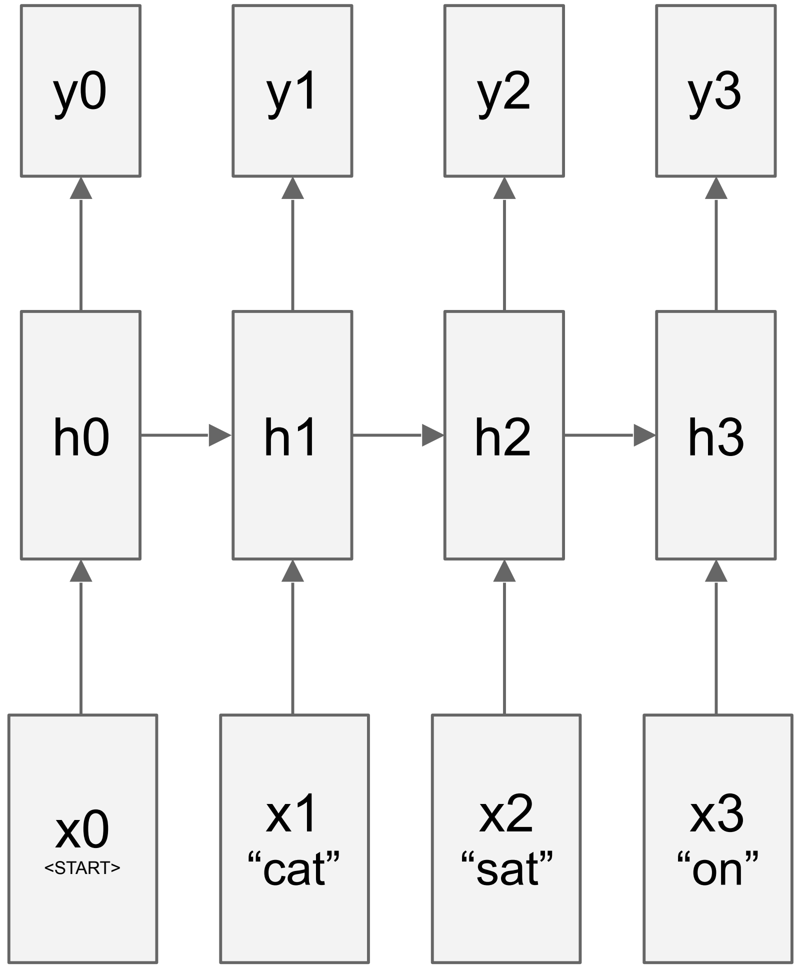
P(next word | previous words)



slide credit: Andrej Karpathy

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

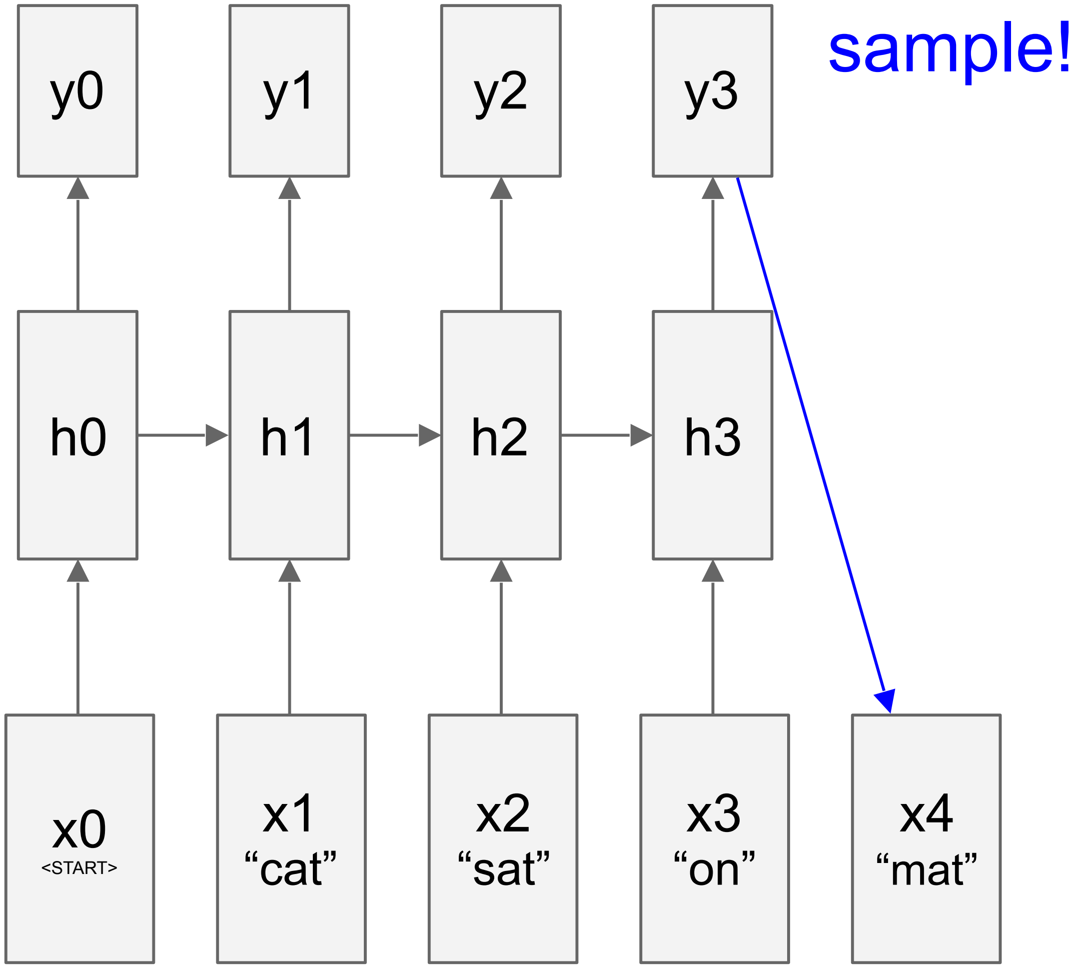
P(next word | previous words)



slide credit: Andrej Karpathy

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

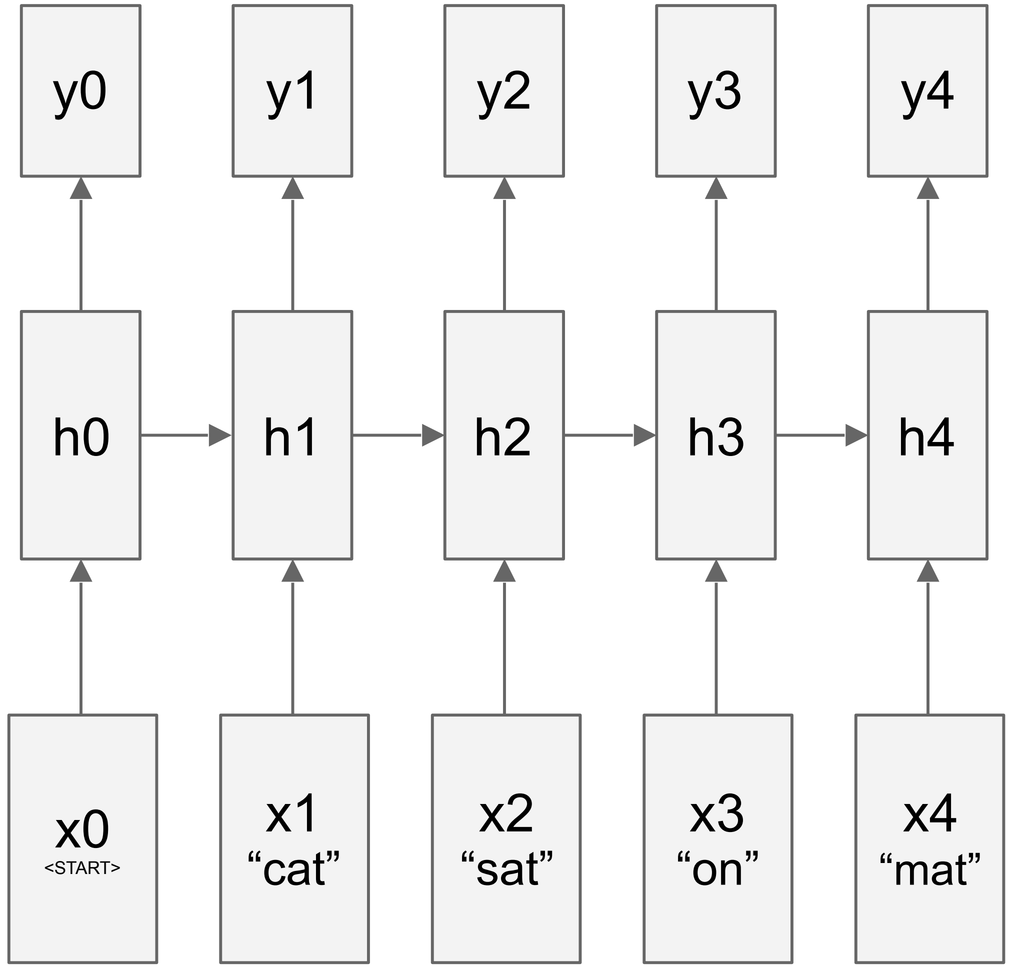
P(next word | previous words)



slide credit: Andrej Karpathy

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

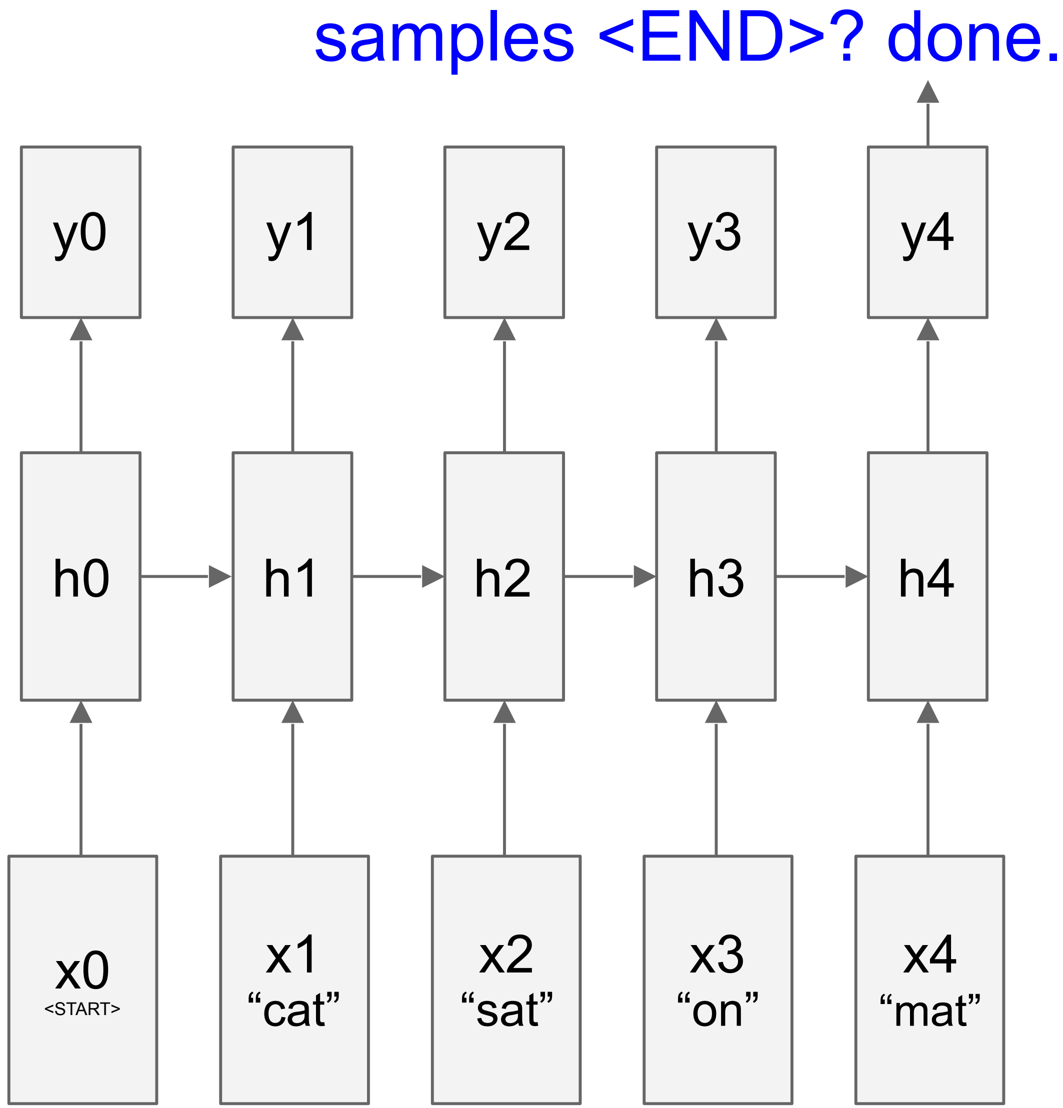
P(next word | previous words)



slide credit: Andrej Karpathy

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

P(next word | previous words)

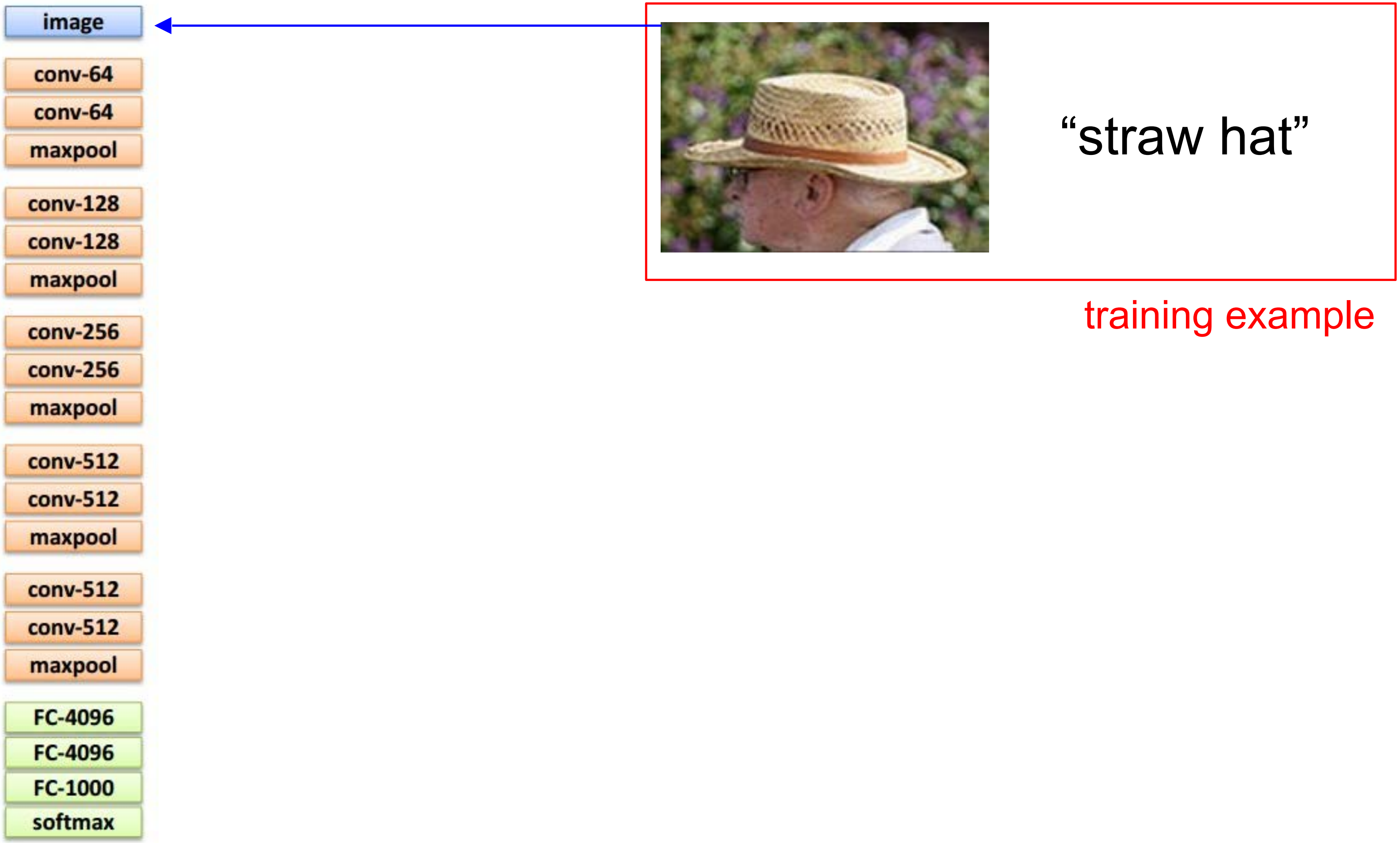


slide credit: Andrej Karpathy

“straw hat”

training example

slide credit: Andrej Karpathy

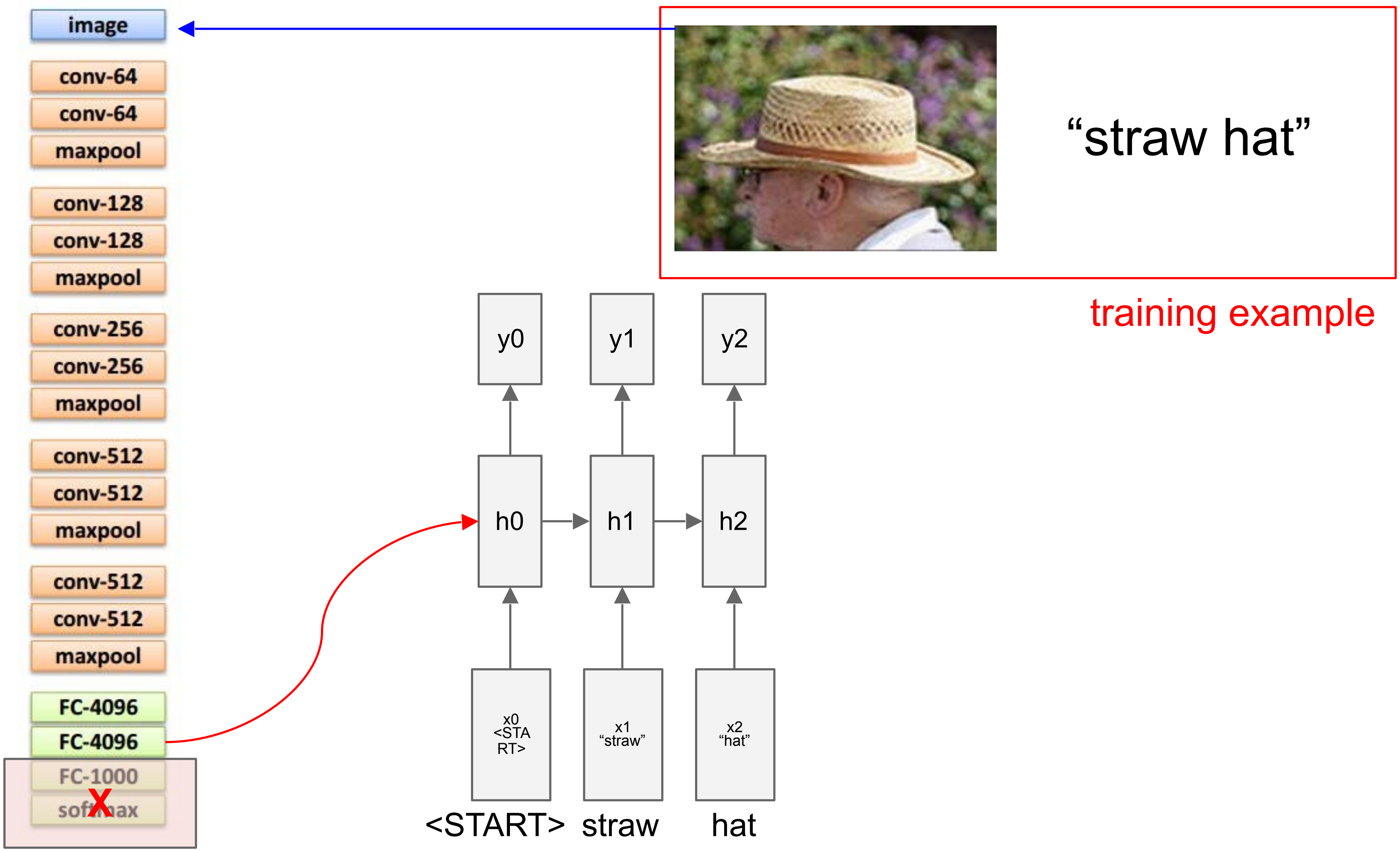


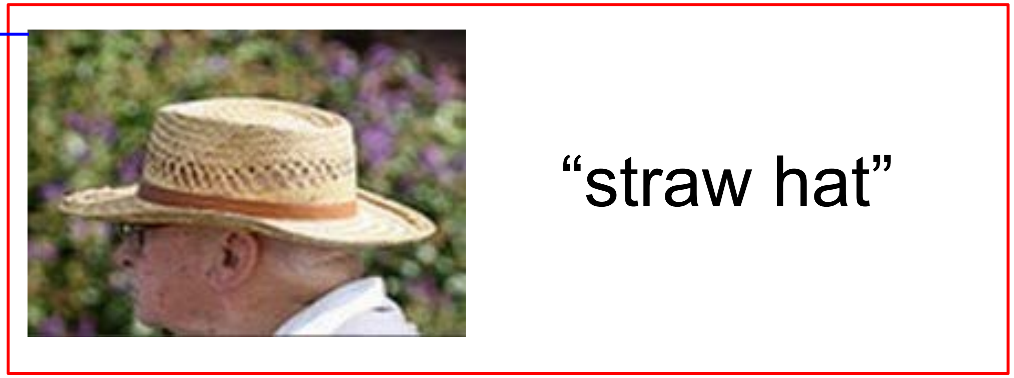
slide credit: Andrej Karpathy

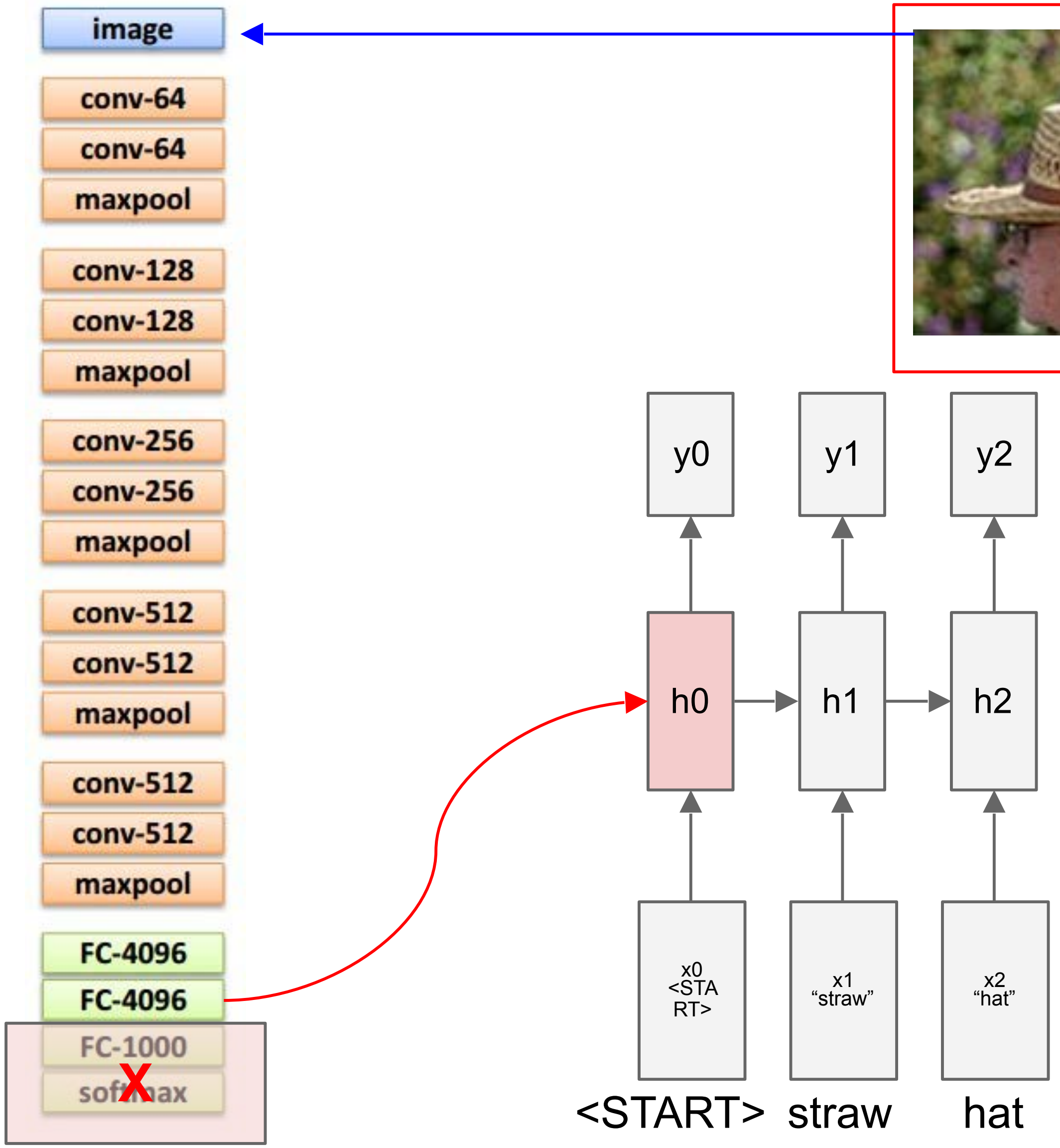
“straw hat”

training example

slide credit: Andrej Karpathy







before:

h0 = tanh(0, Wxh * x0)

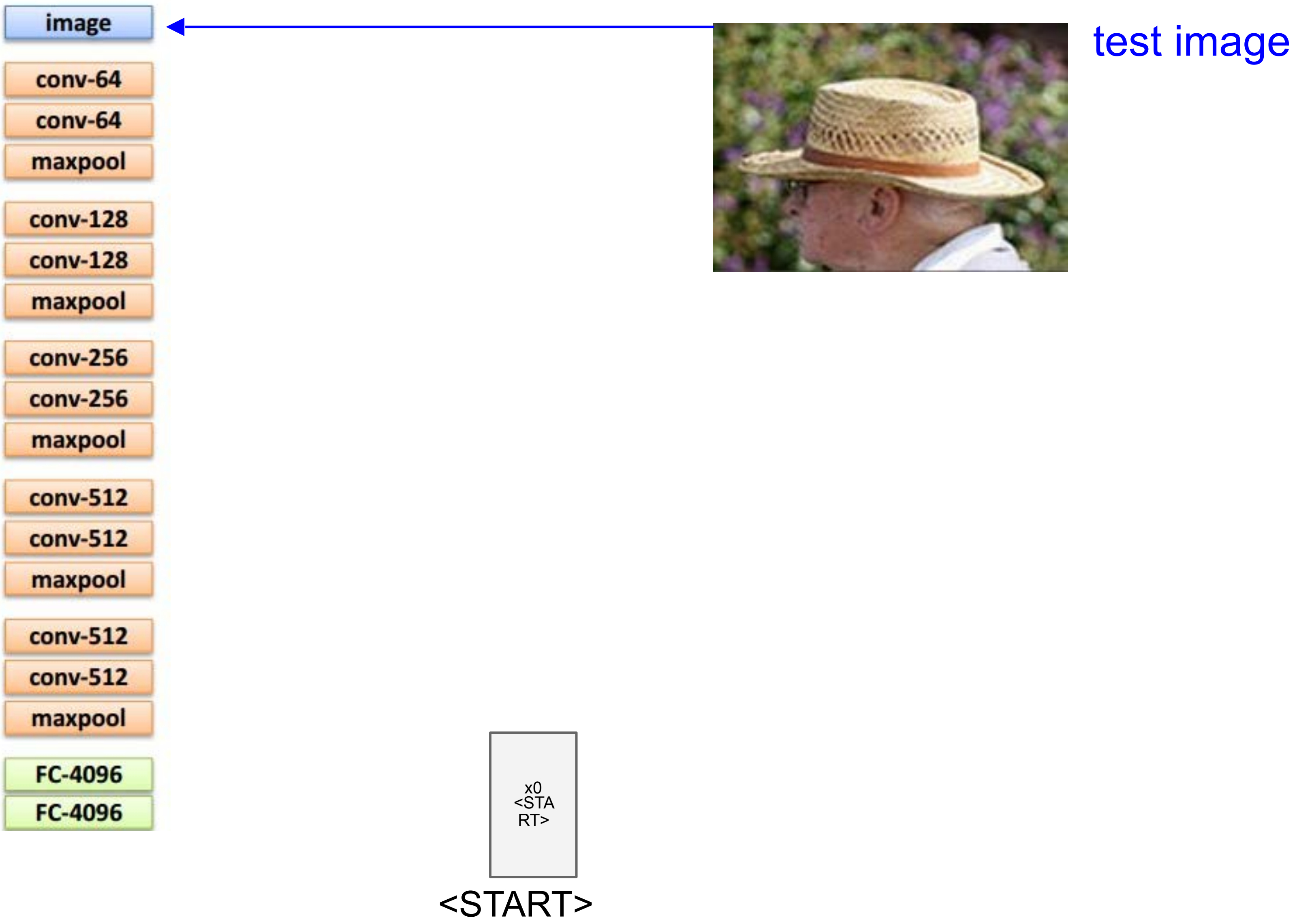
now:

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

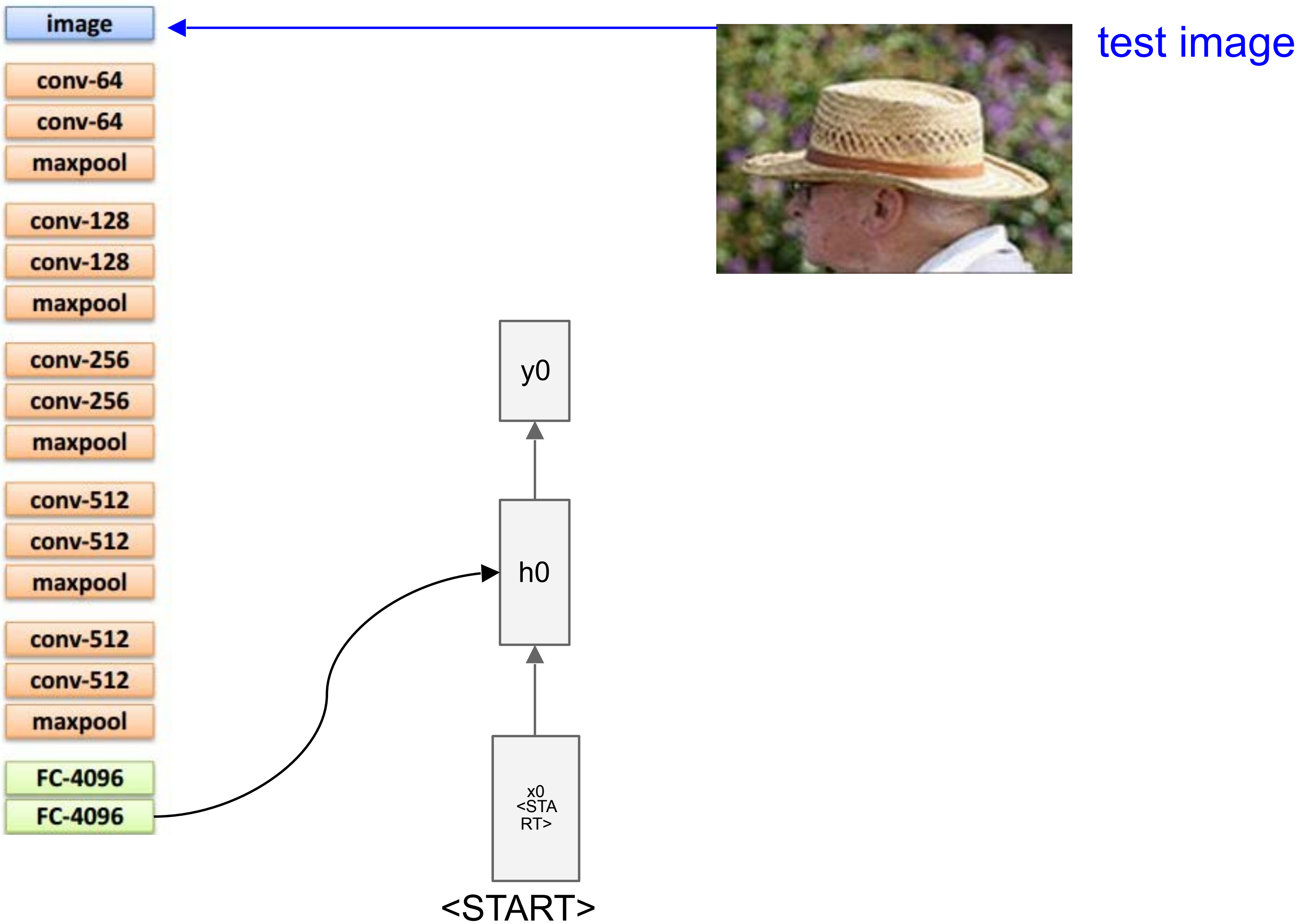
slide credit: Andrej Karpathy

test image

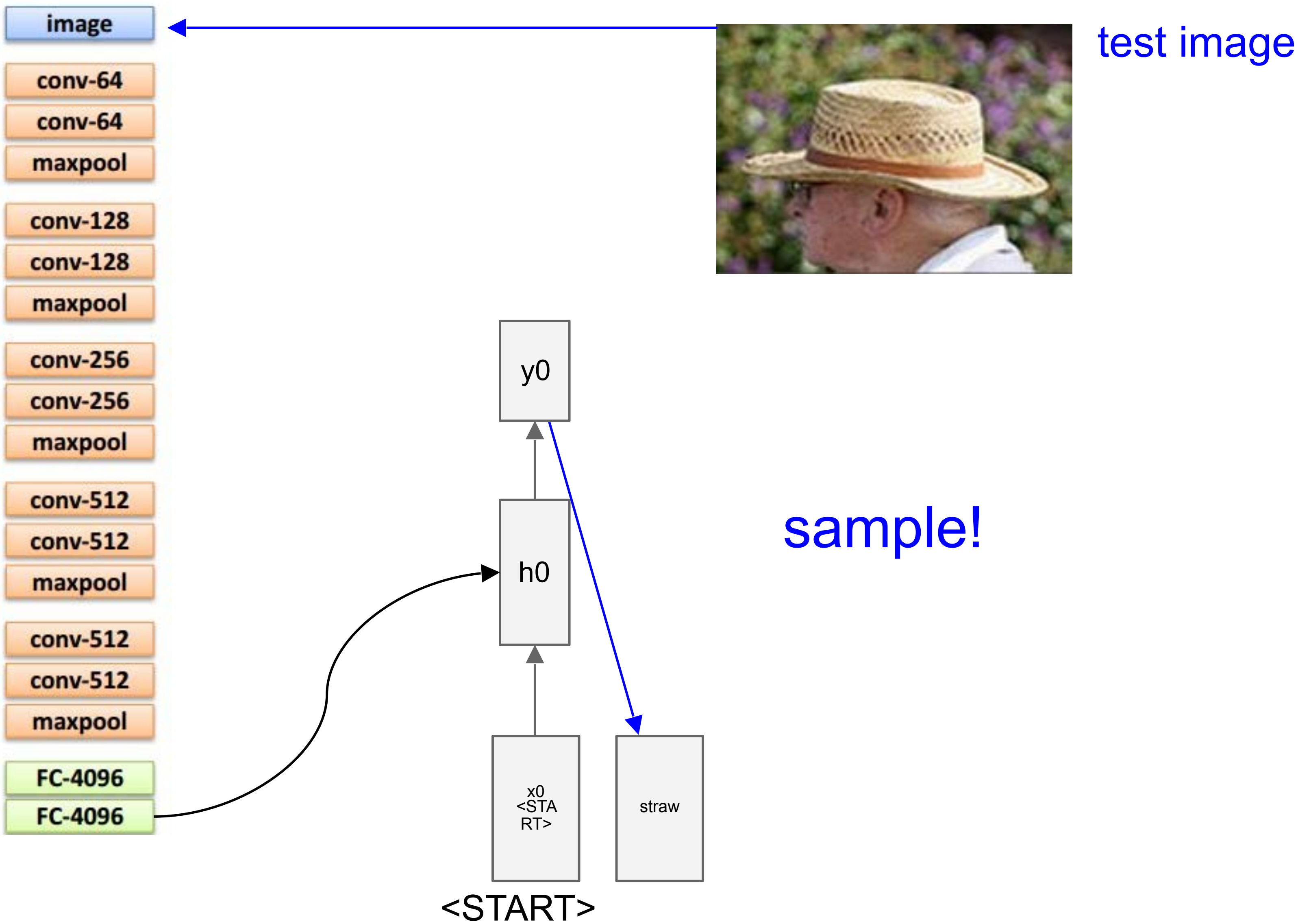
slide credit: Andrej Karpathy



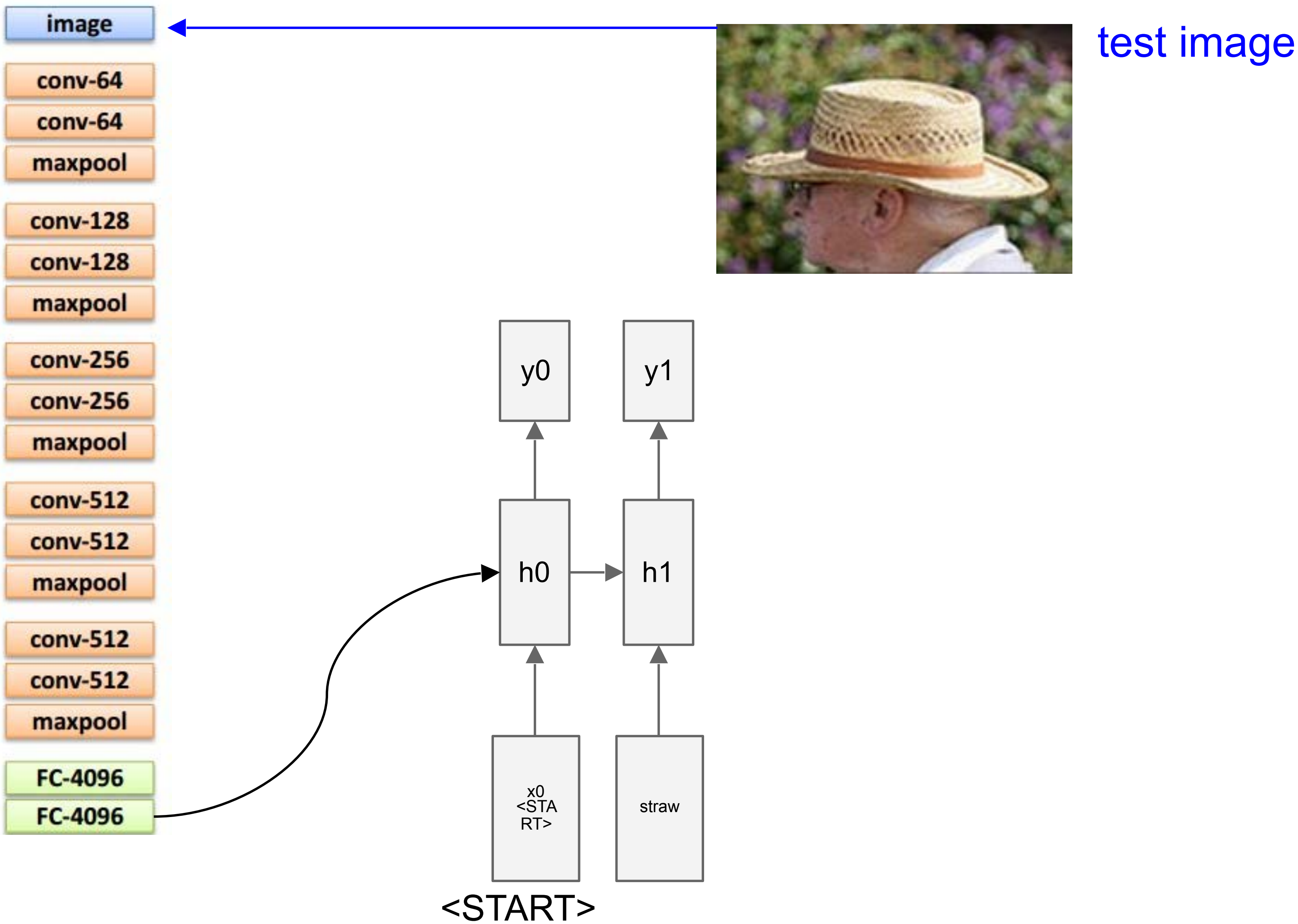
slide credit: Andrej Karpathy



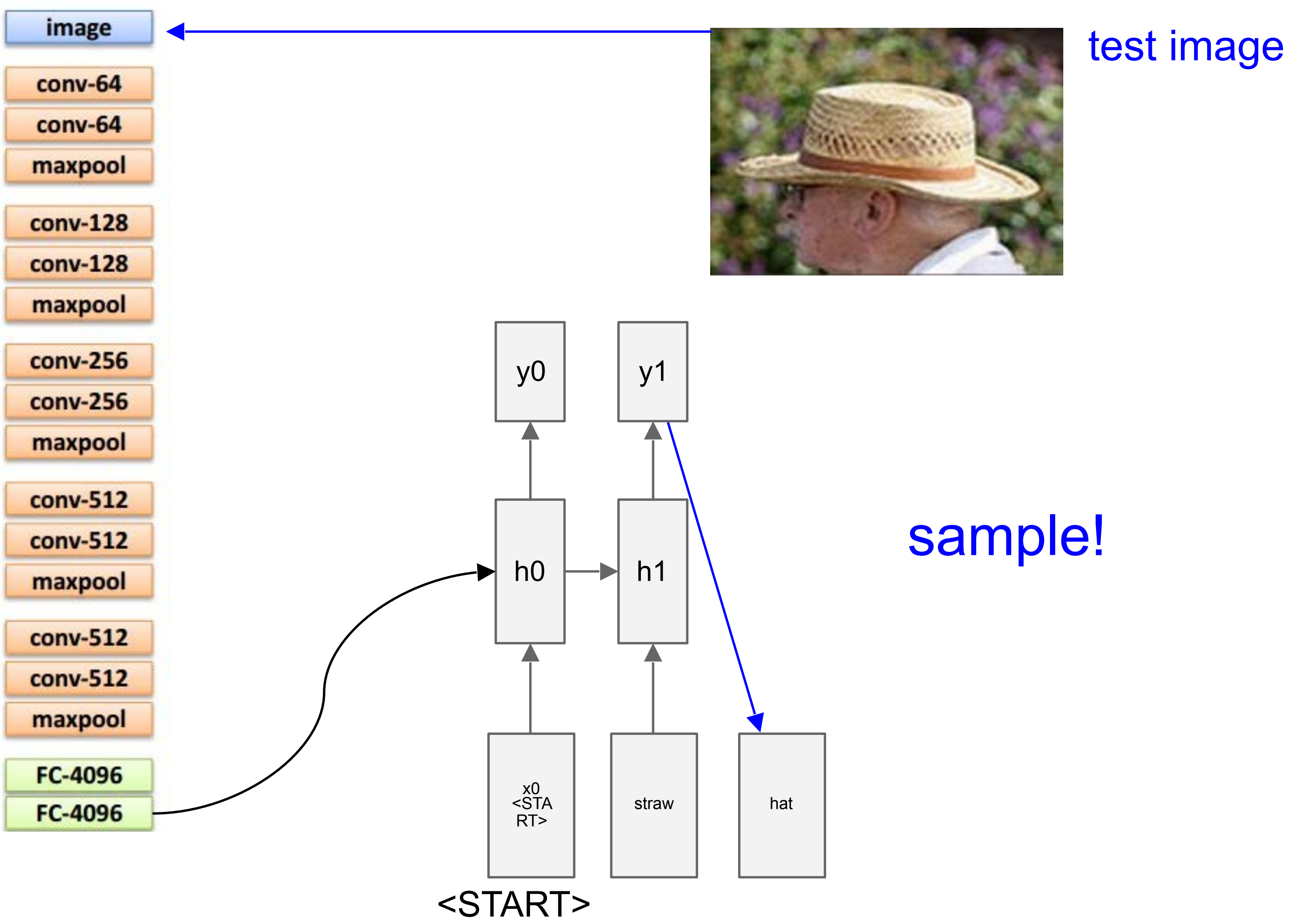
slide credit: Andrej Karpathy



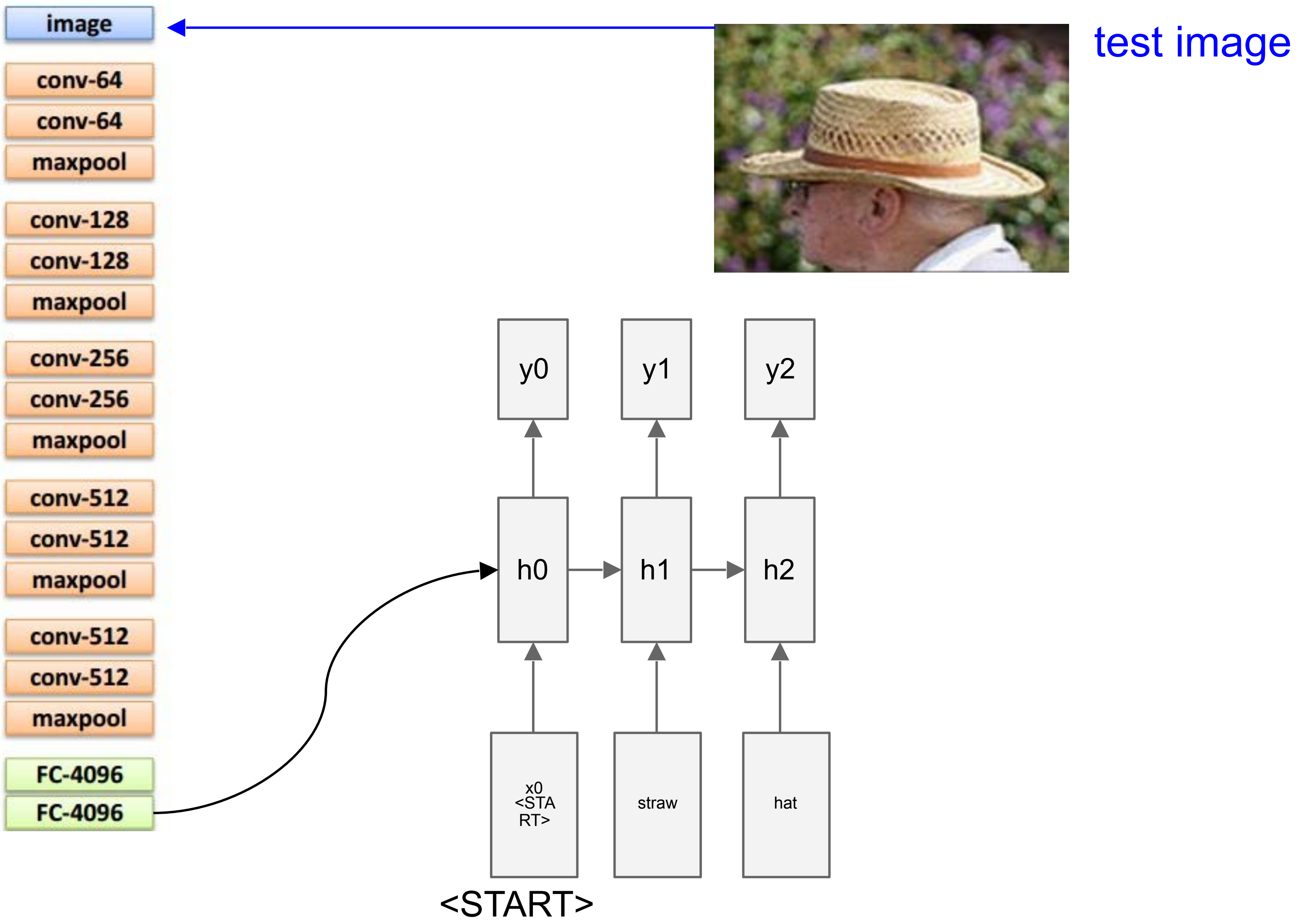
slide credit: Andrej Karpathy



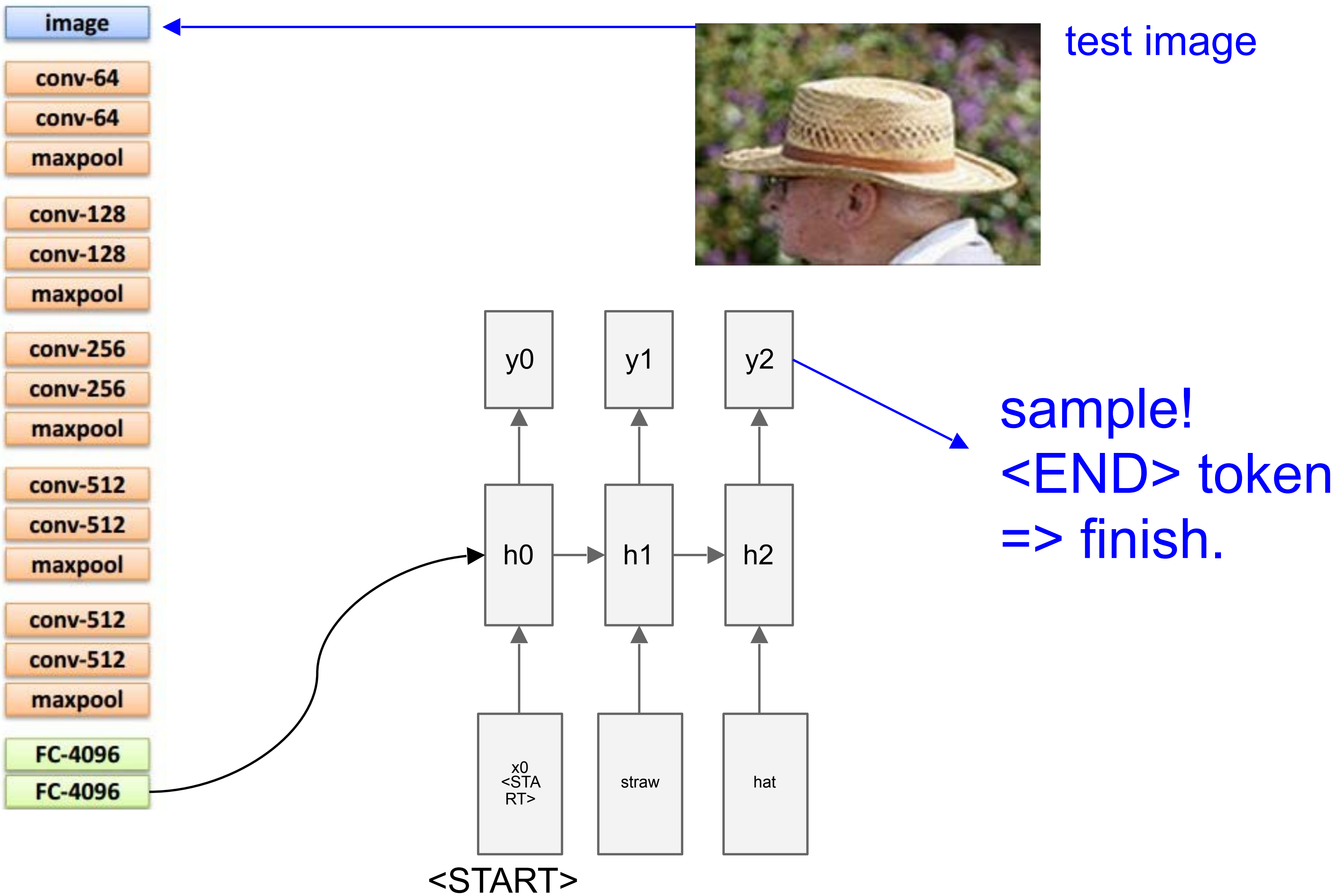
slide credit: Andrej Karpathy



slide credit: Andrej Karpathy



slide credit: Andrej Karpathy



slide credit: Jeff Donahue

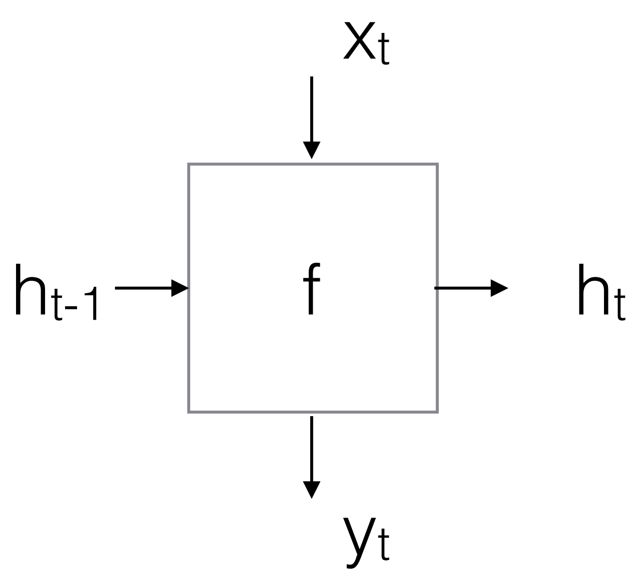
Sequence Learning

- Instances of the form $\mathbf{x} = <x_1, x_2, x_3, \ldots, x_T>$
- Variable sequence length T
- Learn a transition function f with parameters W:
- f should update hidden state h_t and output y_t

$h_0 := 0$

for $t = 1, 2, 3, \ldots, T$:

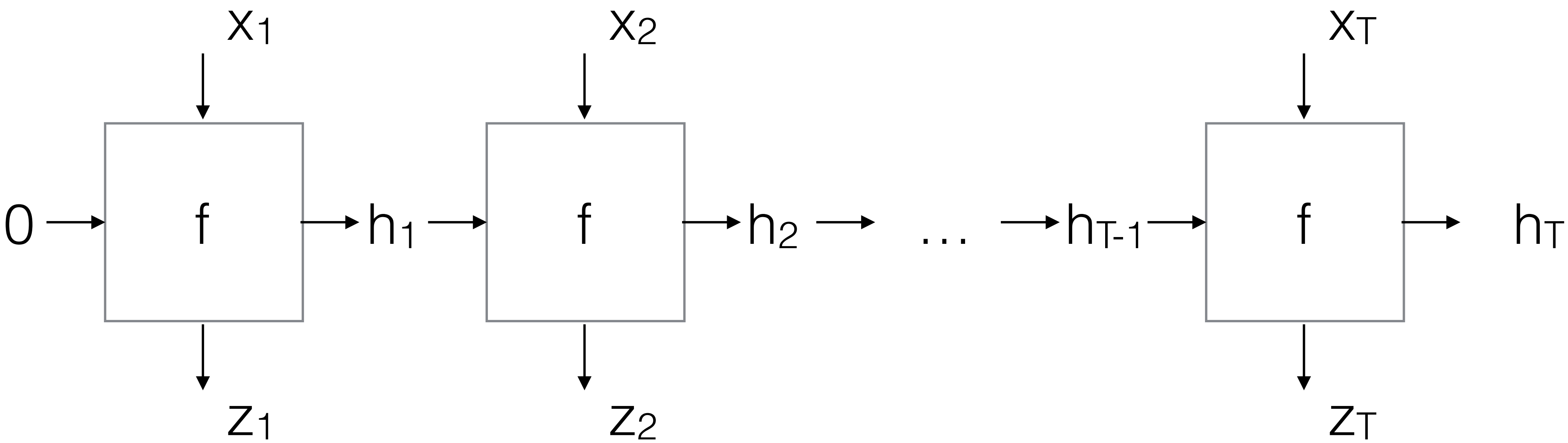
$<y_t, h_t> = f_W(x_t, h_{t-1})$



slide credit: Jeff Donahue

Sequence Learning

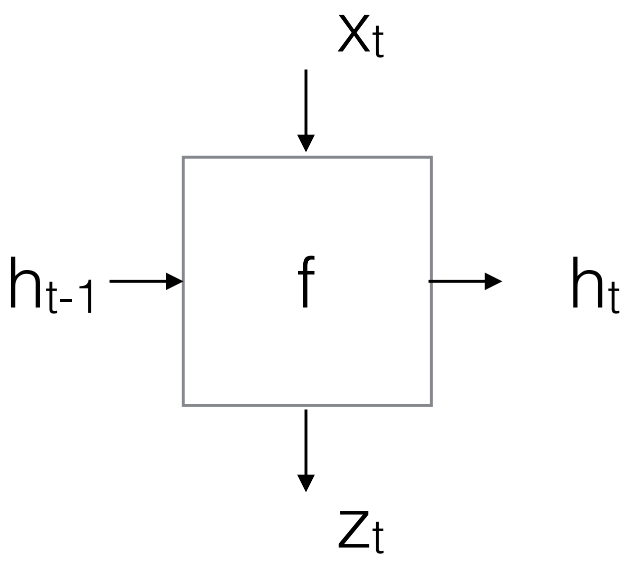
Equivalent to a T-layer deep network, unrolled in time



slide credit: Jeff Donahue

Sequence Learning

- What should the transition function f be?



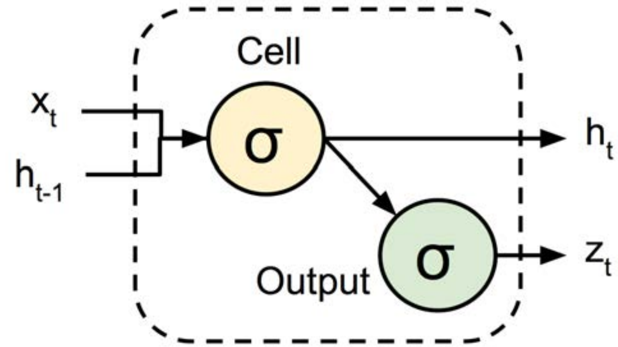
- At a minimum, we want something **non-linear** and **differentiable**

Sequence Learning

- First attempt — a “vanilla” RNN:

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

$$z_t = \sigma(W_{hz}h_t + b_z)$$



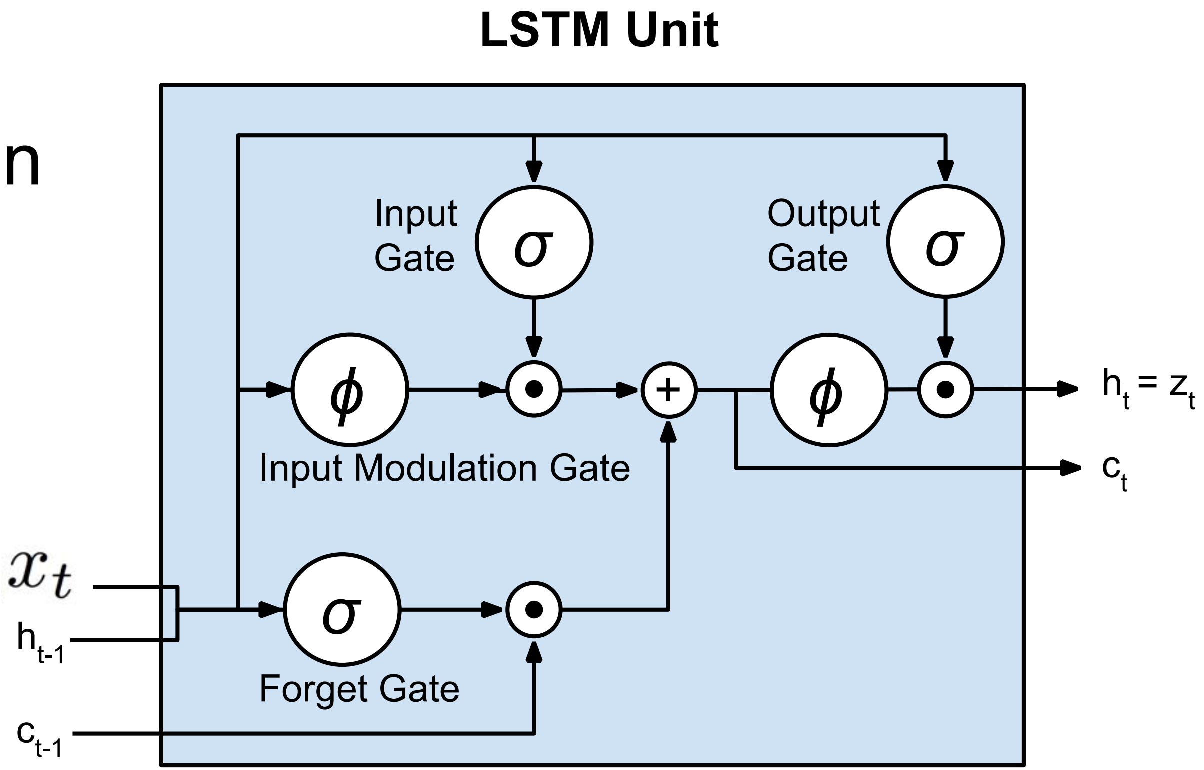
- Problems
  - Difficult to train — vanishing/exploding gradients
  - Unable to “select” inputs, hidden state, outputs

slide credit: Marcus Rohrbach

Sequence Learning

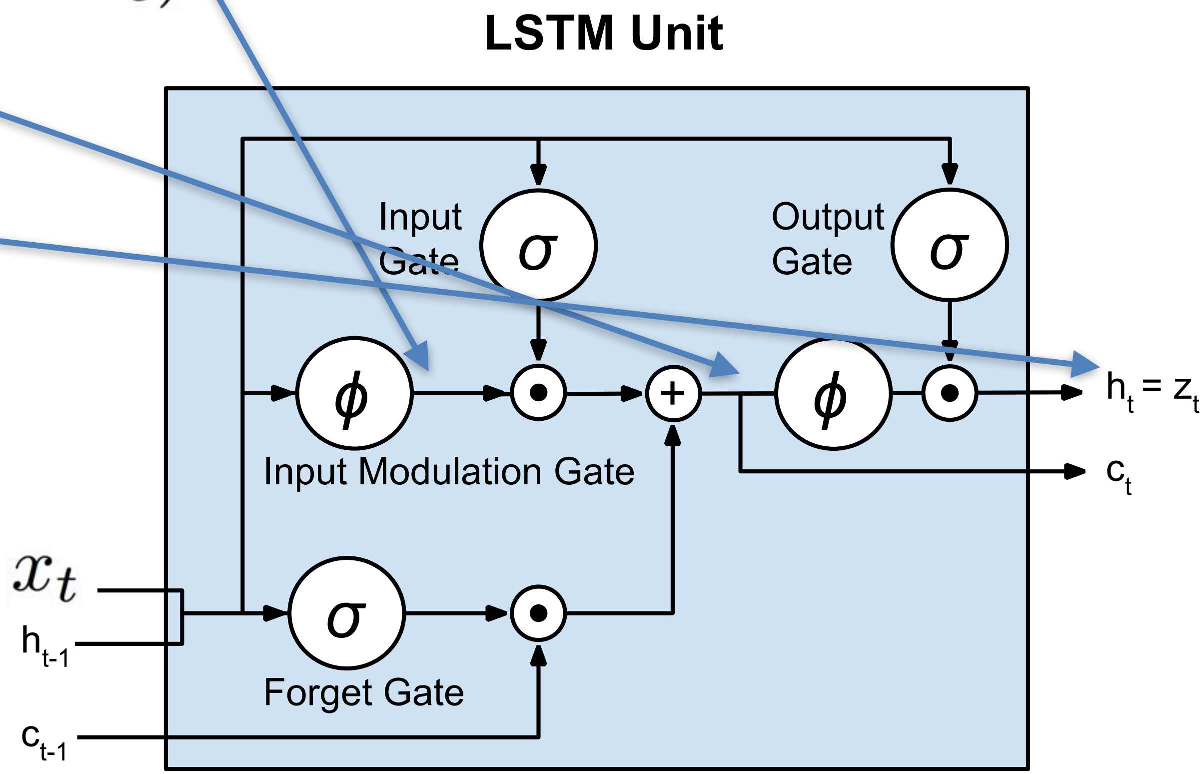
- LSTM - Long Short Term Memory [Hochreiter & Schmidhuber, 1997]
- Selectively propagate or forget hidden state
- Allows long-term dependencies to be learned

- Effective for
  - speech recognition
  - handwriting recognition
  - translation
  - parsing



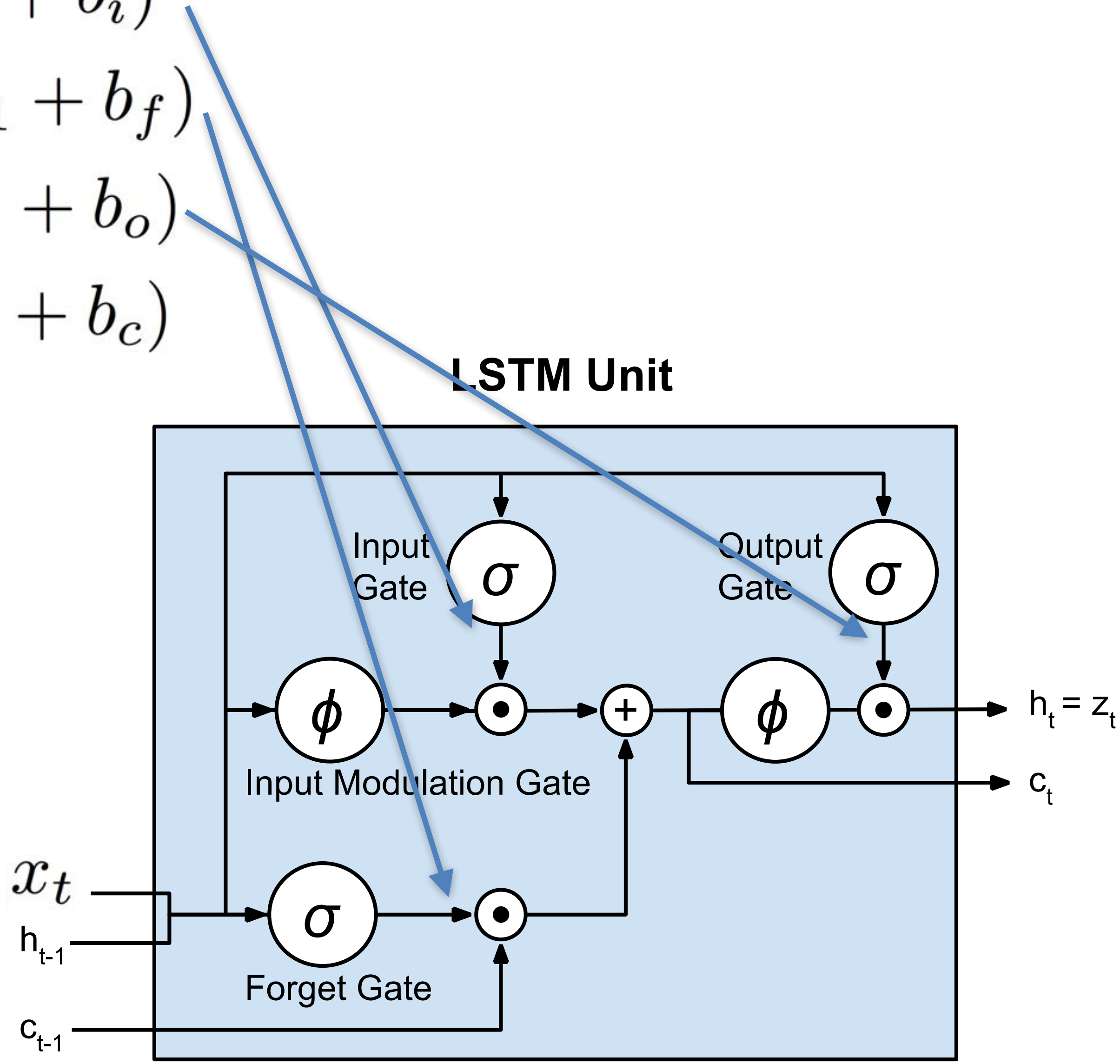
LSTM for sequence modeling

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$g_t = \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \phi(c_t)$$



LSTM for sequence modeling

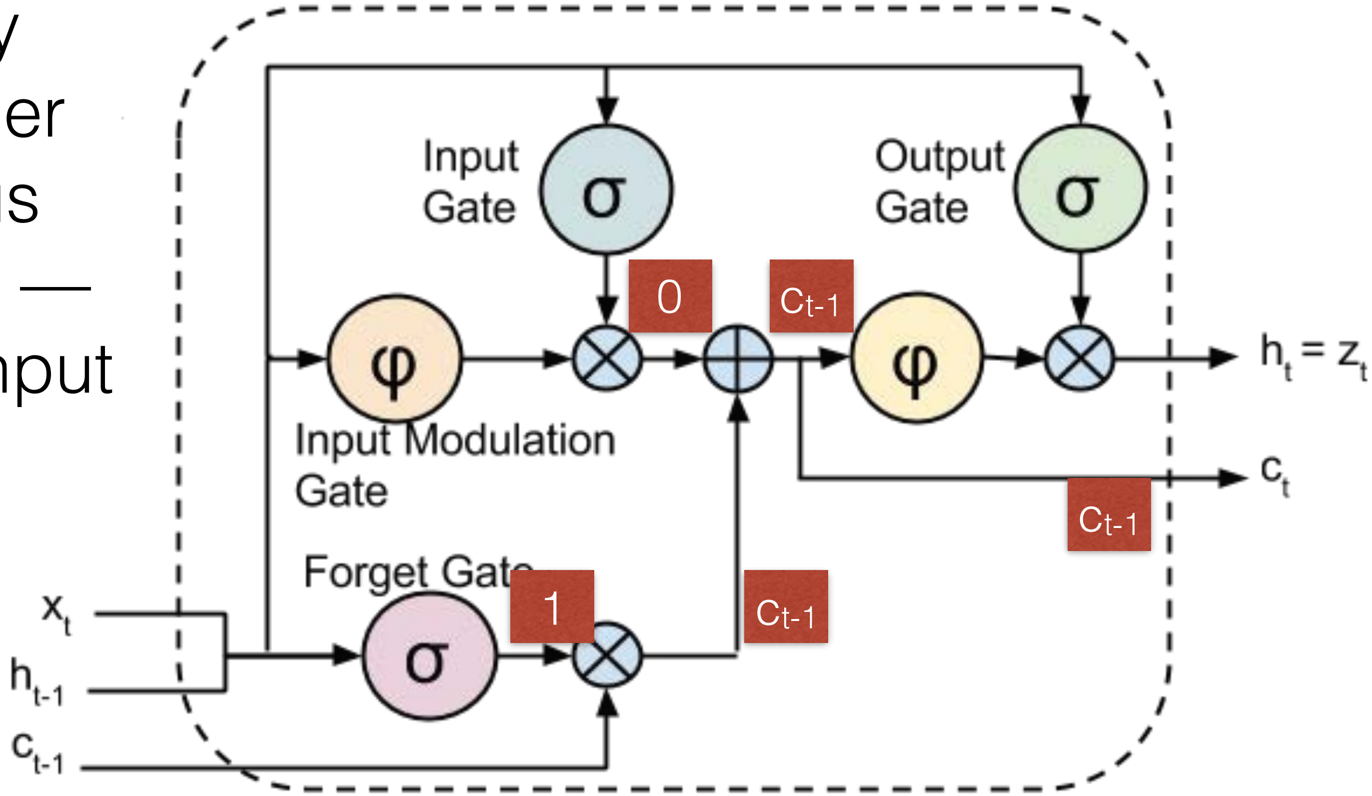
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$g_t = \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \phi(c_t)$$



slide credit: Jeff Donahue

Sequence Learning

Exactly remember previous cell state — discard input

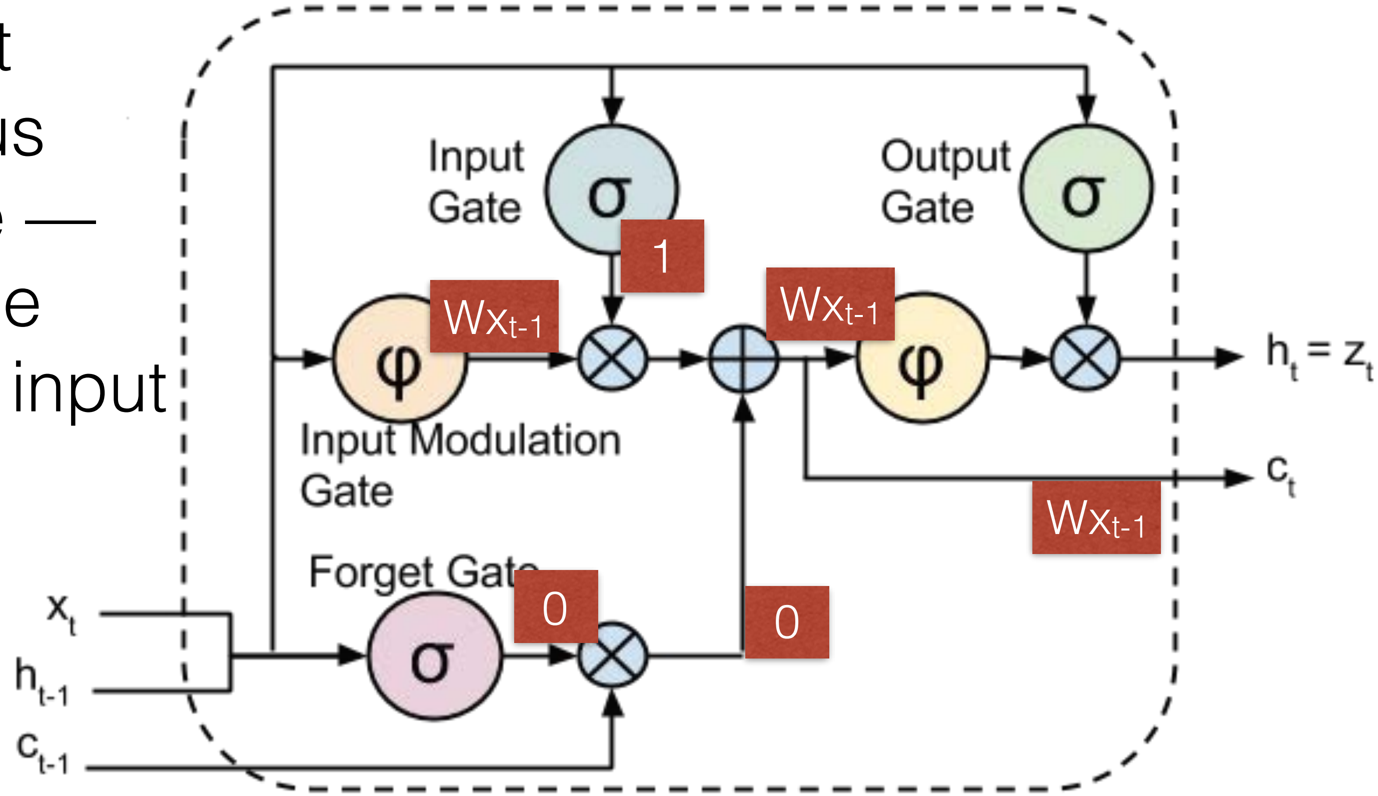


LSTM
(Hochreiter & Schmidhuber, 1997)

slide credit: Jeff Donahue

Sequence Learning

Forget previous cell state — only use modulated input



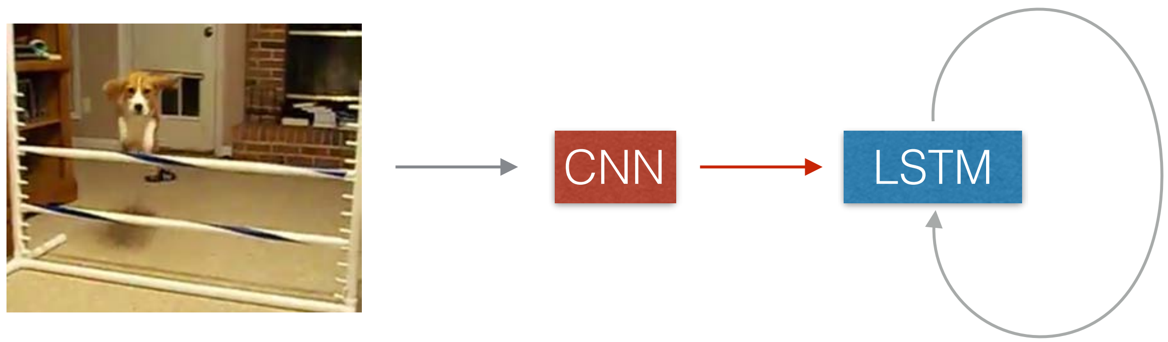
LSTM
(Hochreiter & Schmidhuber, 1997)

slide credit: Jeff Donahue

LRCN

- Long-term Recurrent Convolutional Networks

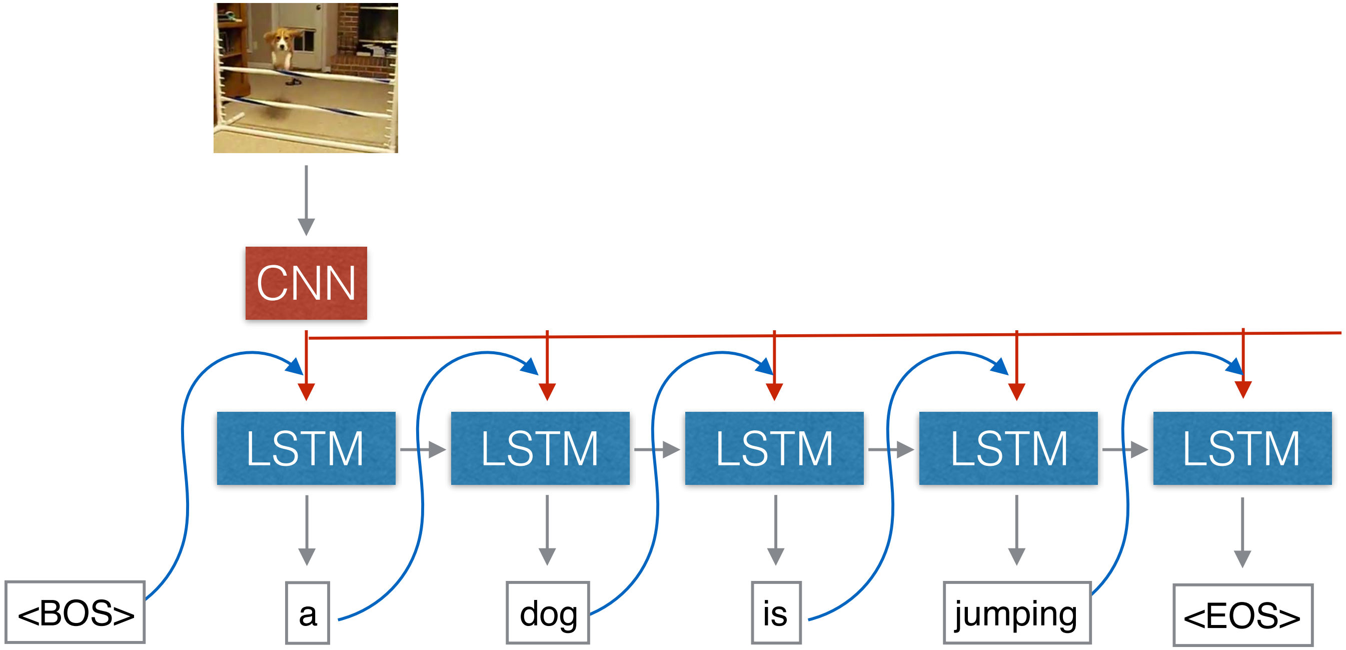
- End-to-end trainable framework for sequence problems in vision



slide credit: Jeff Donahue

Image Description

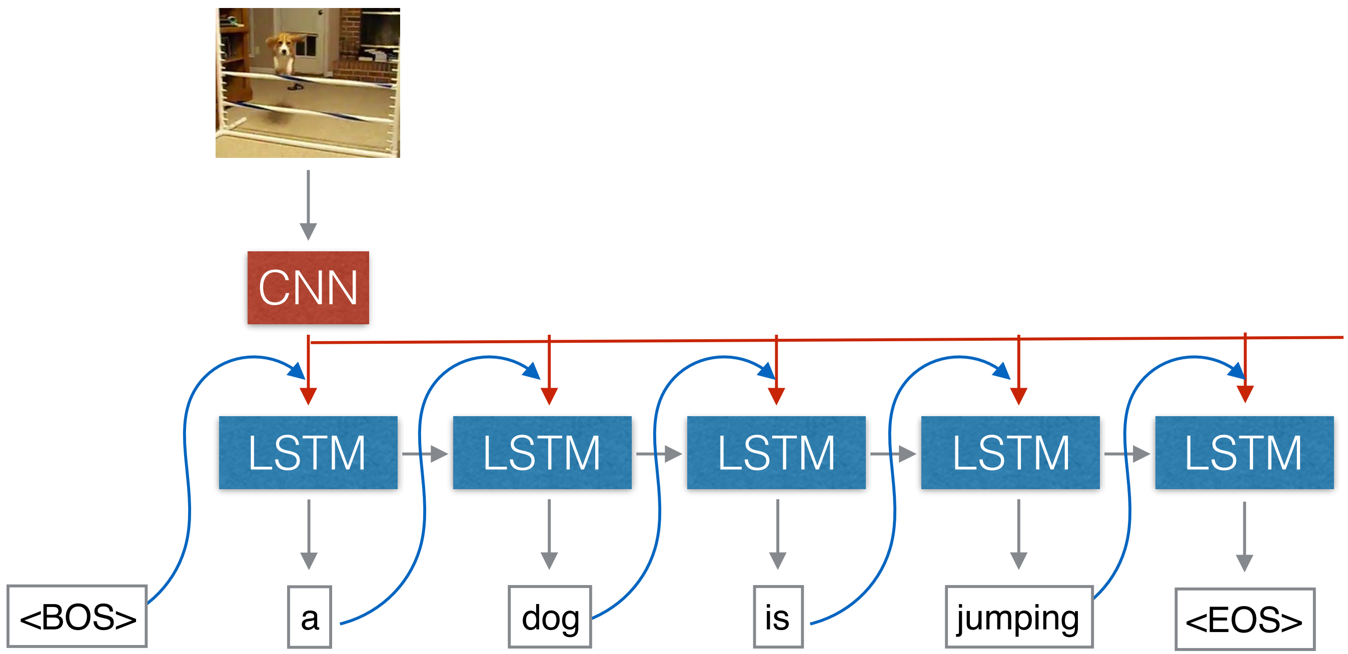
sequential output

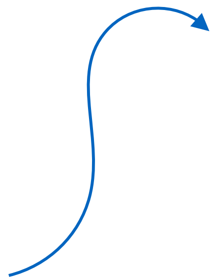


slide credit: Jeff Donahue

Image Description

sequential
output



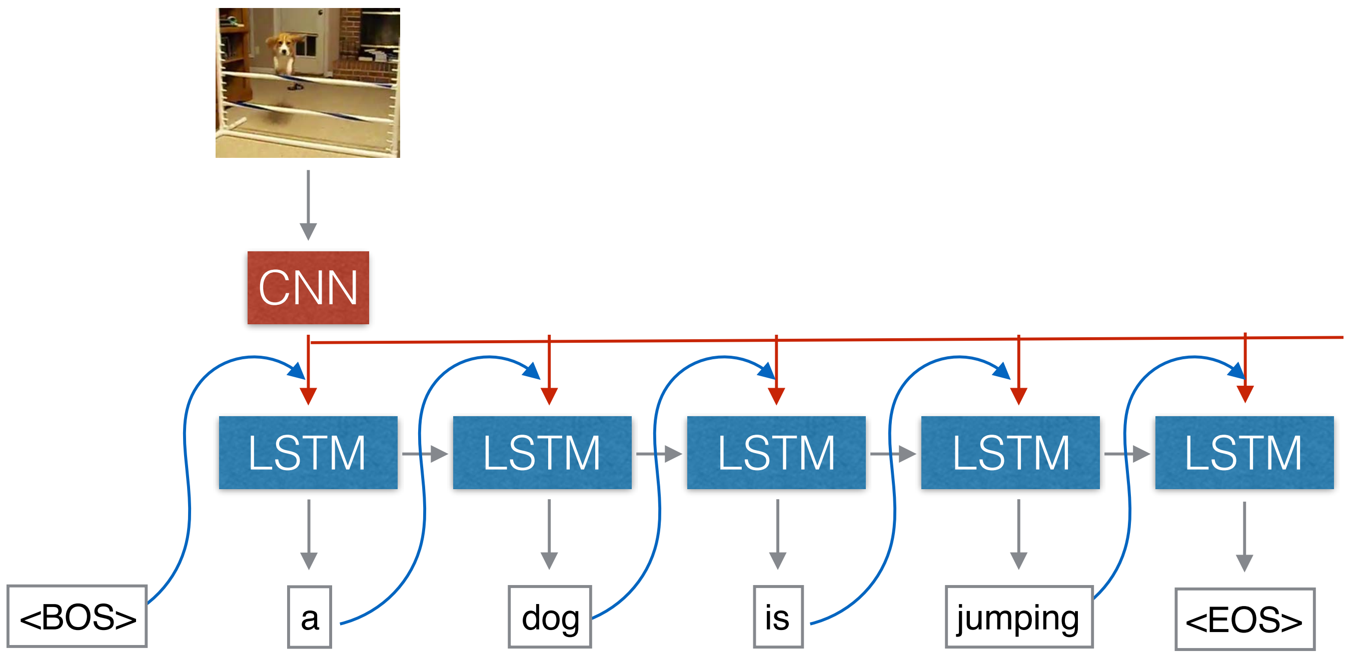


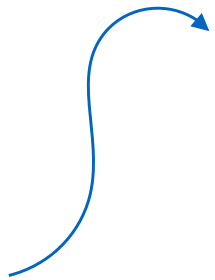
= embed a one-hot vector

slide credit: Jeff Donahue

Image Description

single LSTM layer



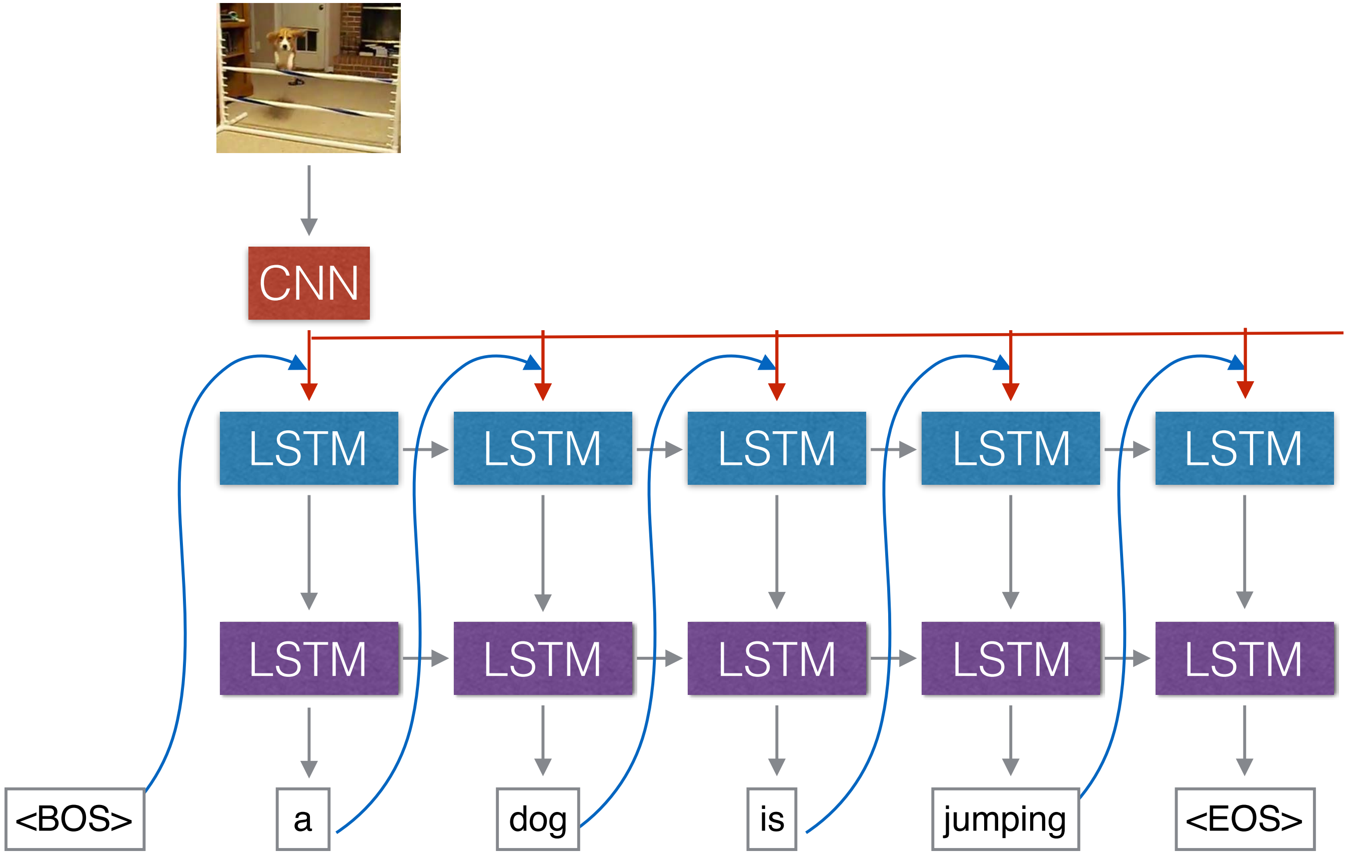


= embed a one-hot vector

slide credit: Jeff Donahue

Image Description

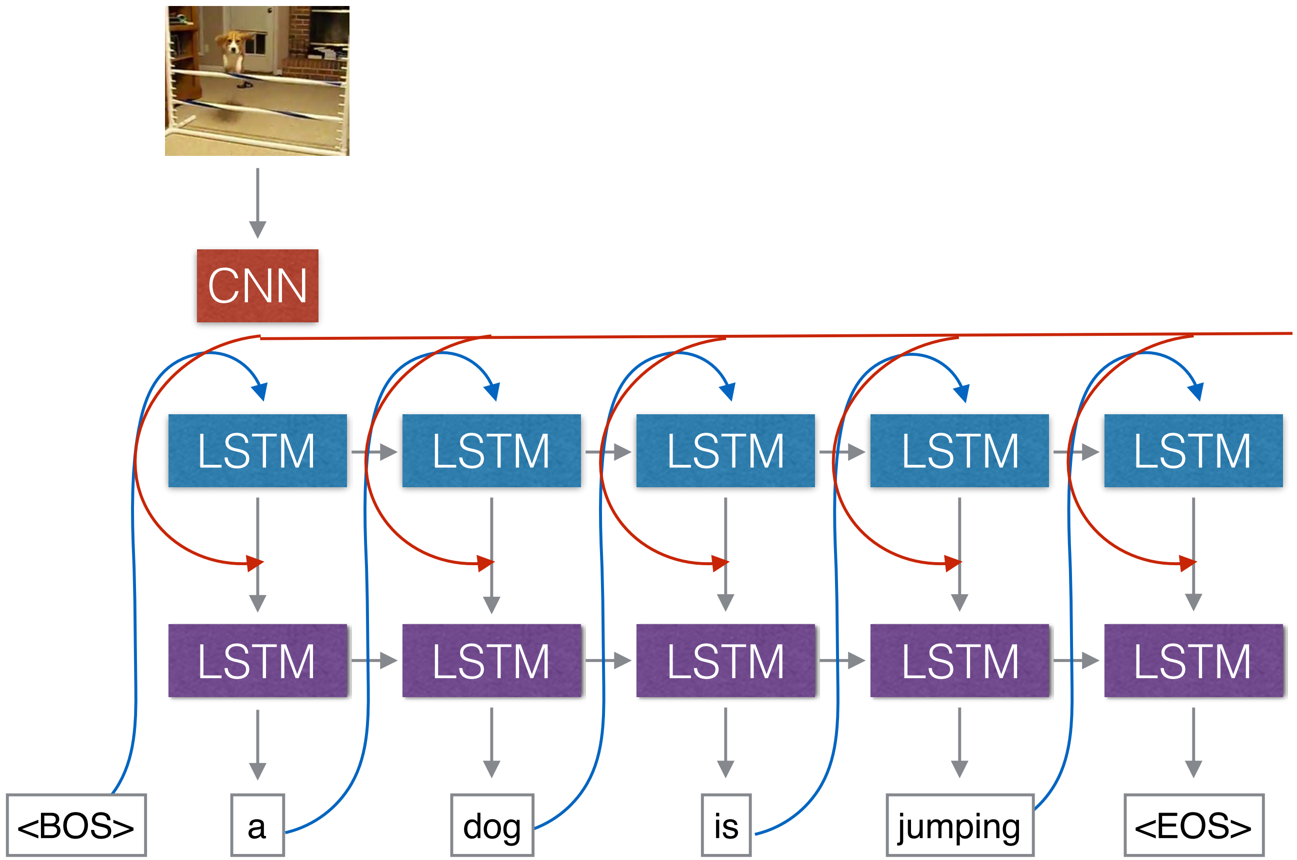
two LSTM layers



slide credit: Jeff Donahue

Image Description

two LSTM
layers, factored



slide credit: Jeff Donahue

Image Description

Architecture	Flickr30k [1] Caption-to-Image Recall@1
Single Layer	14.1%
Two Layer	3.8%
Two Layer, Factored	**17.5%**
Four Layer, Factored	15.8%

[1] P. Hodosh, A. Young, M. Lai, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions.

slide credit: Jeff Donahue

Image Description

COCO [1] CIDEr-D c5 Scores	CaffeNet	VGGNet [2]
Raw	68.8%	77.3%
Finetuned	75.8%	**83.9%**

[1] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Com- mon objects in context. *arXiv preprint arXiv:1405.0312*, 2014.
[2] K. Simonyan & A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". ICLR 2015.

slide credit: Jeff Donahue

Image Description

A female tennis player in action on the court.

A group of young men playing a game of soccer.

A man riding a wave on top of a surfboard.

slide credit: Jeff Donahue

Image Description

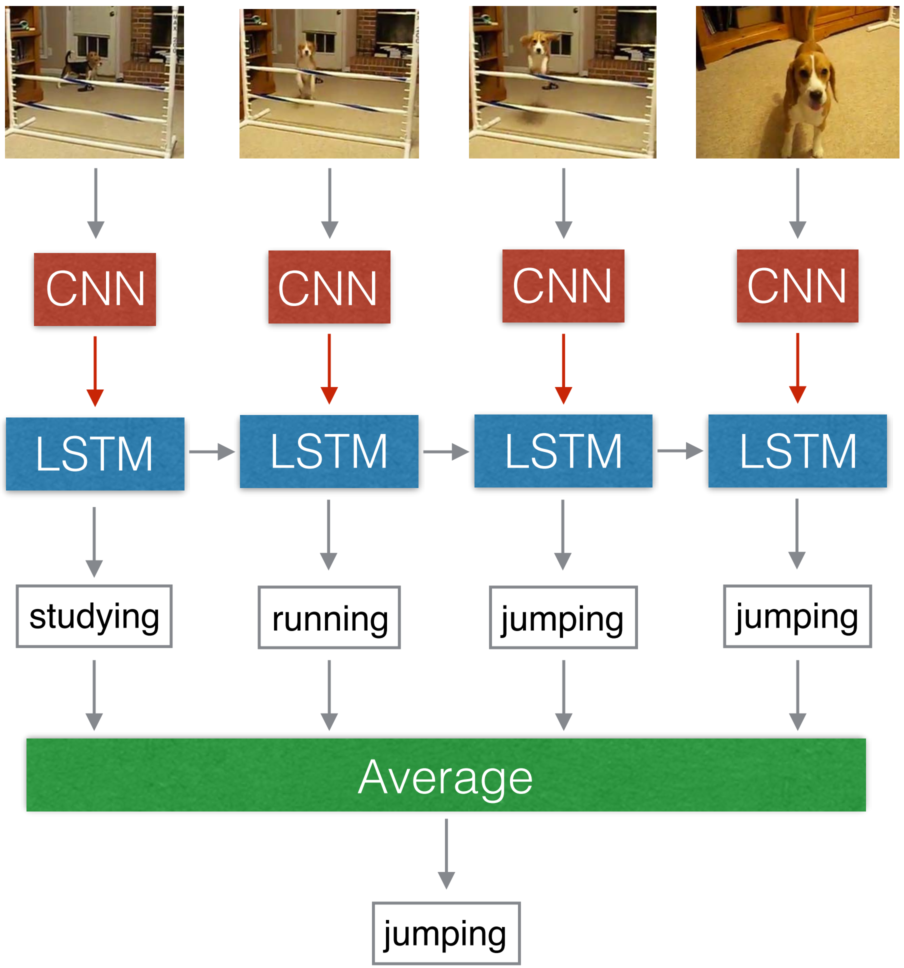
A black and white cat is sitting on a chair.

A large clock mounted to the side of a building.

A bunch of fruit that are sitting on a table.

slide credit: Jeff Donahue

Activity Recognition



sequential
input

slide credit: Jeff Donahue

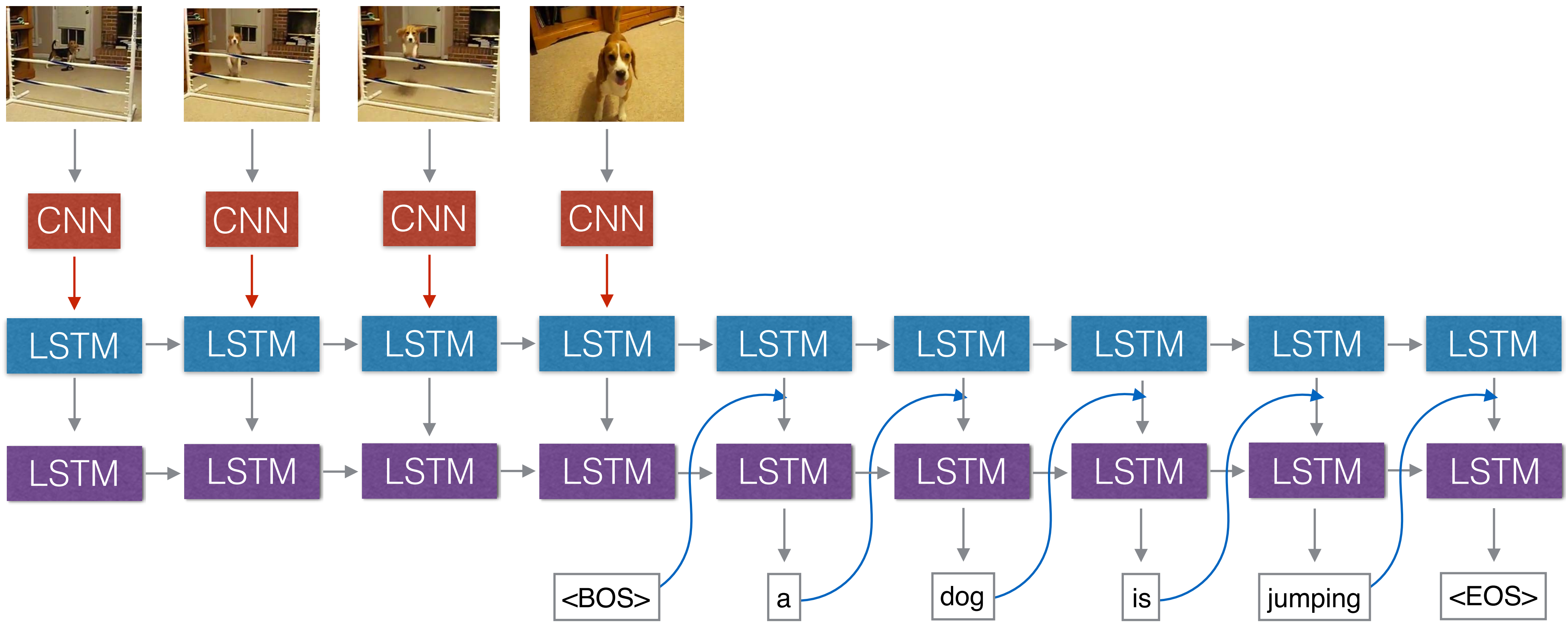
Activity Recognition

UCF101 Class. Acc.	RGB	Optical Flow	RGB+ Flow
Single-Frame CNN	67.7%	72.2%	78.8%
LRCN	**68.2%**	**77.5%**	**82.7%**

Khurram Soomro, Amir Roshan Zamir and Mubarak Shah, UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild., CRCV-TR-12-01, November, 2012.

slide credit: Jeff Donahue

Video Description



sequential input & output

slide credit: Jeff Donahue

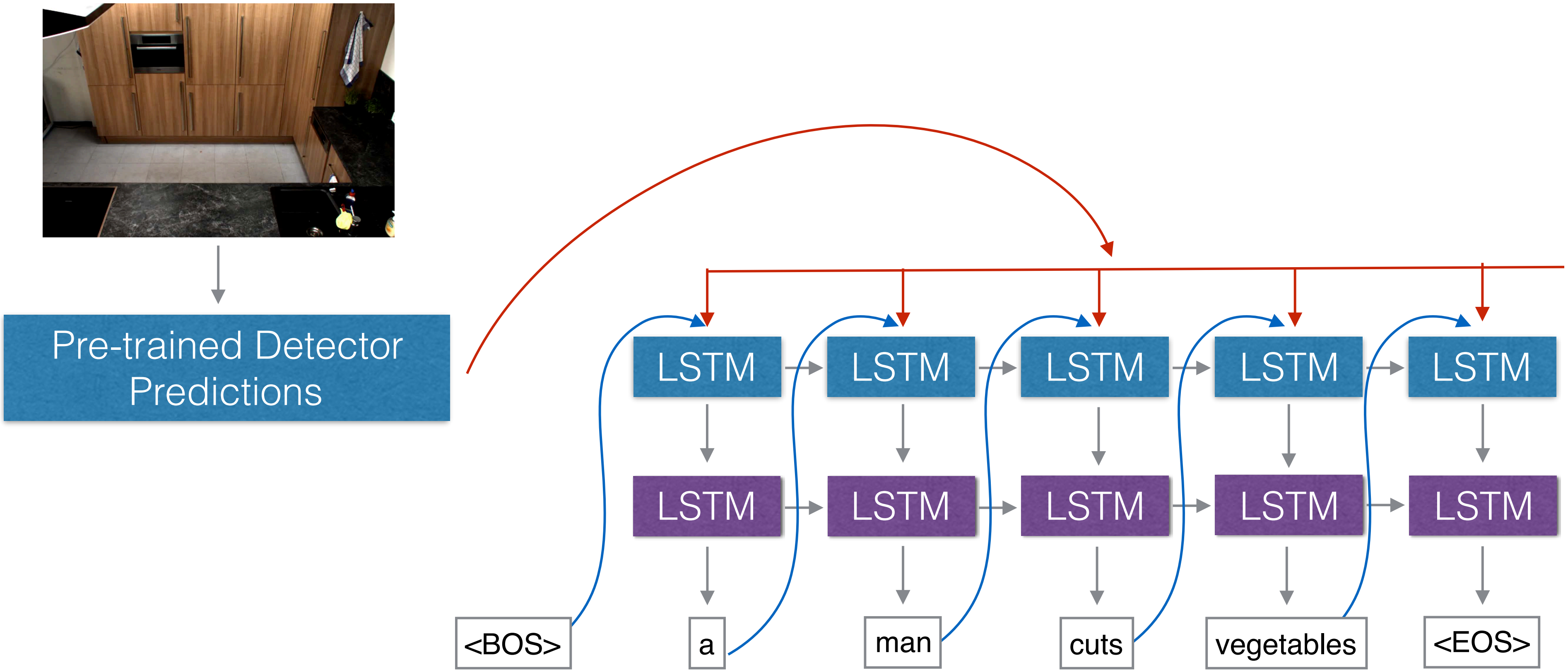
Video Description

MPII TACoS Multi-Level Dataset

Coherent Multi-Sentence Video Description with Variable Level of Detail. A. Rohrbach, M. Rohrbach, W. Qiu, A. Friedrich and M. Pinkal and B. Schiele. GCPR, 2014.

slide credit: Jeff Donahue

Video Description



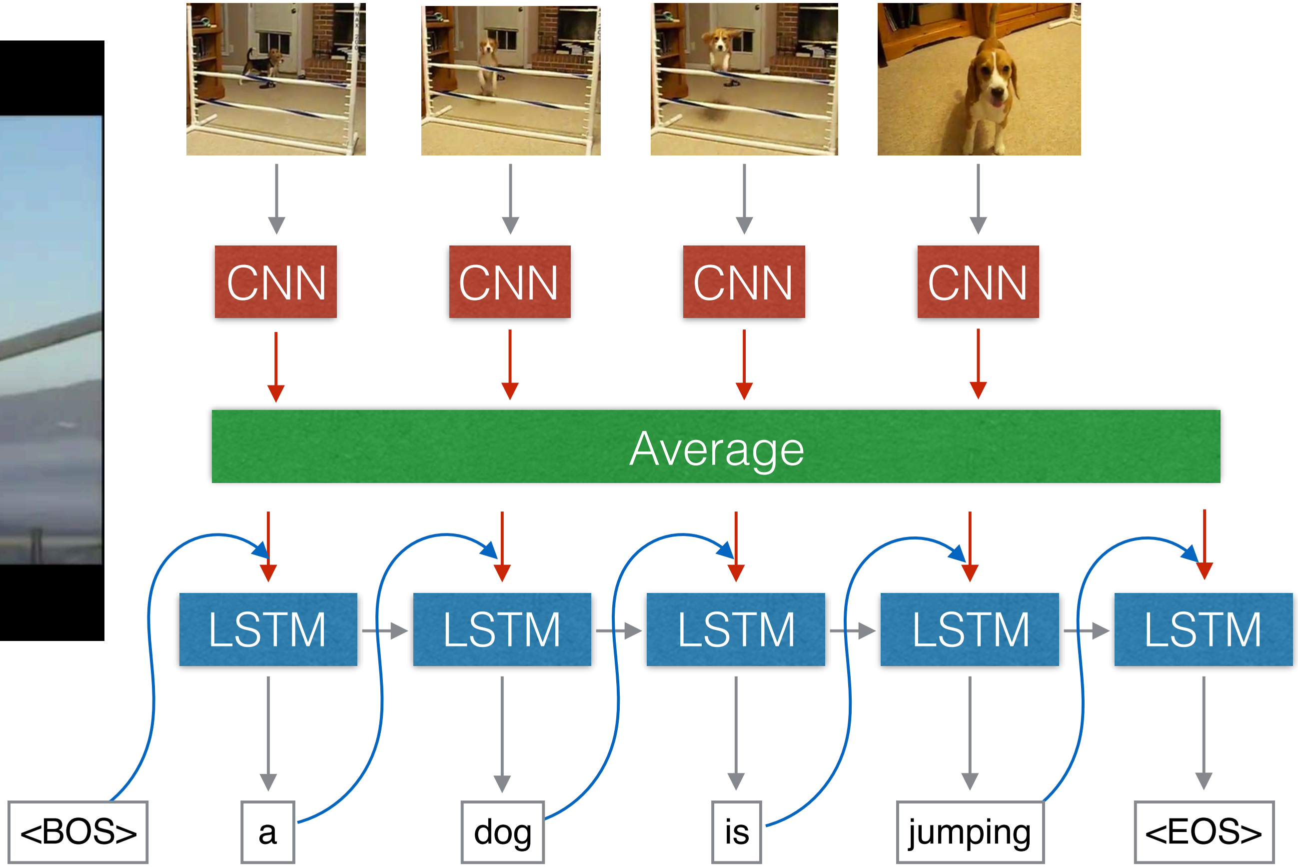
slide credit: Jeff Donahue

Video Description

Approach	Generation Accuracy (BLEU)
SMT	26.9%
LRCN	**28.8%**

slide credit: Jeff Donahue

Video Description



Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, Kate Saenko. "Translating Videos to Natural Language Using Deep Recurrent Neural Networks," NAACL 2015 (oral). http://arxiv.org/abs/1412.4729

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`

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`

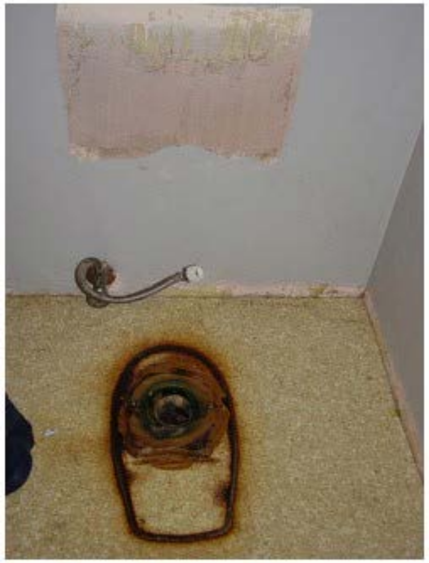
Well, I can kind of see it

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`

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`

slide credit: Andrej Karpathy

"The Unreasonable Effectiveness of Recurrent Neural Networks"

karpathy.github.io

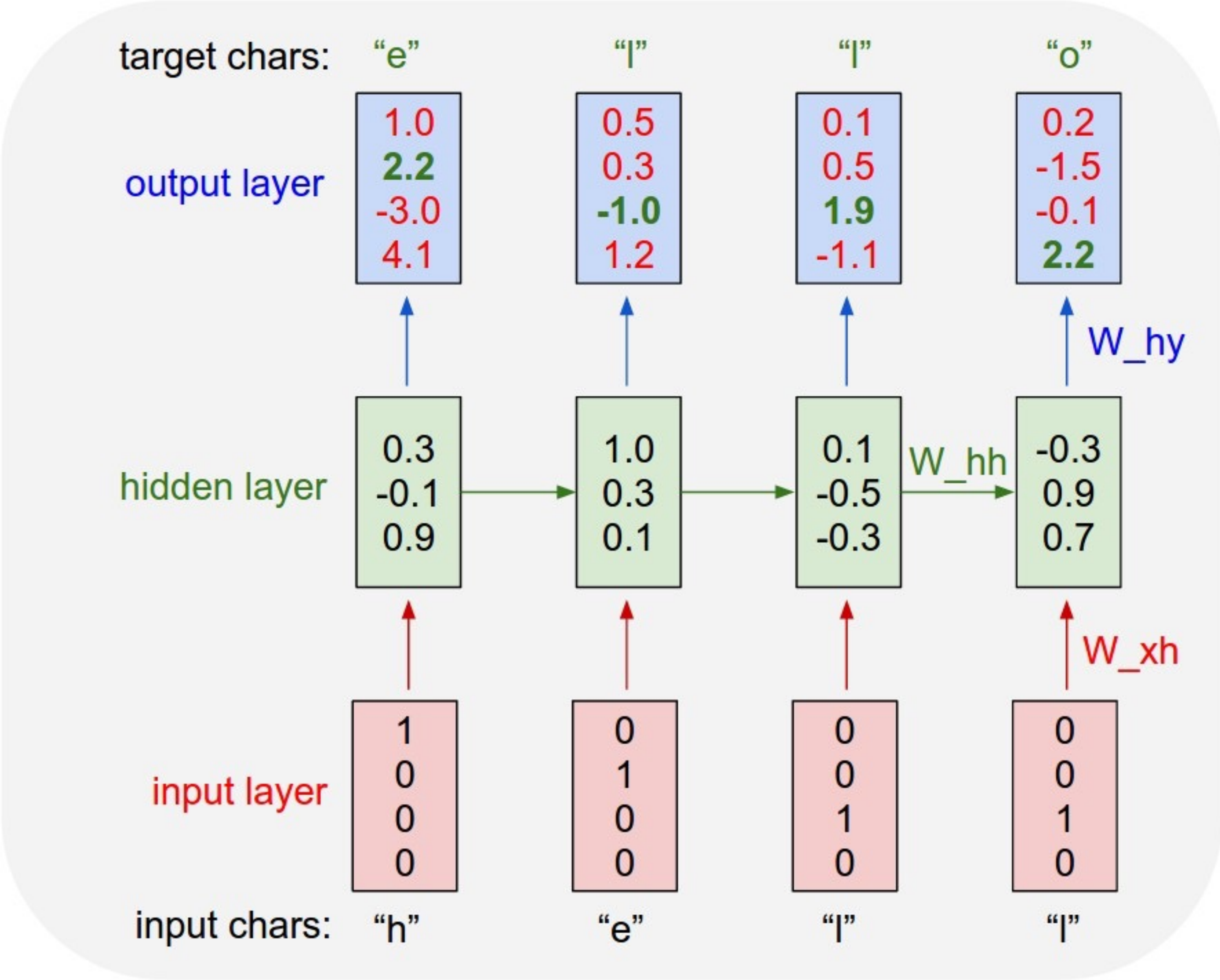
slide credit: Andrej Karpathy

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

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



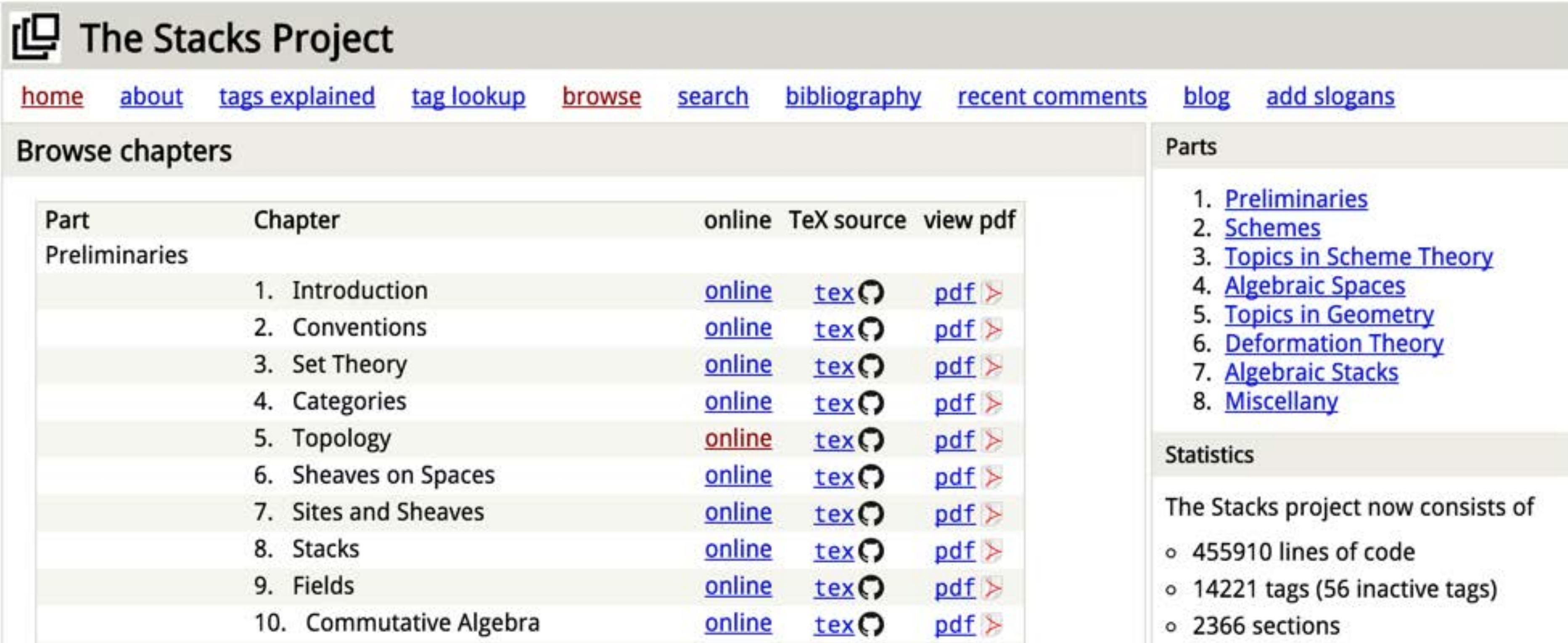
slide credit: Andrej Karpathy

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

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

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 $\operatorname{Spec}(R') \to S$ 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

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

is a unique morphism of algebraic stacks. Note that

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

and

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

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 $\underline{Proj}_X(\mathcal{A}) =$ $\operatorname{Spec}(B)$ 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 $S = \operatorname{Spec}(R)$ and $Y = \operatorname{Spec}(R)$.*

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

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

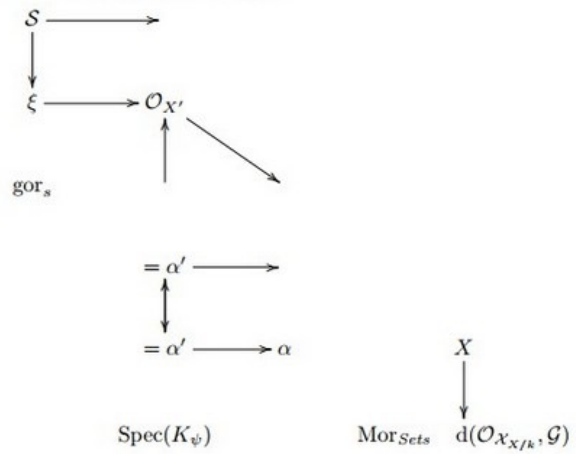
be a morphism of algebraic spaces over S and Y.

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

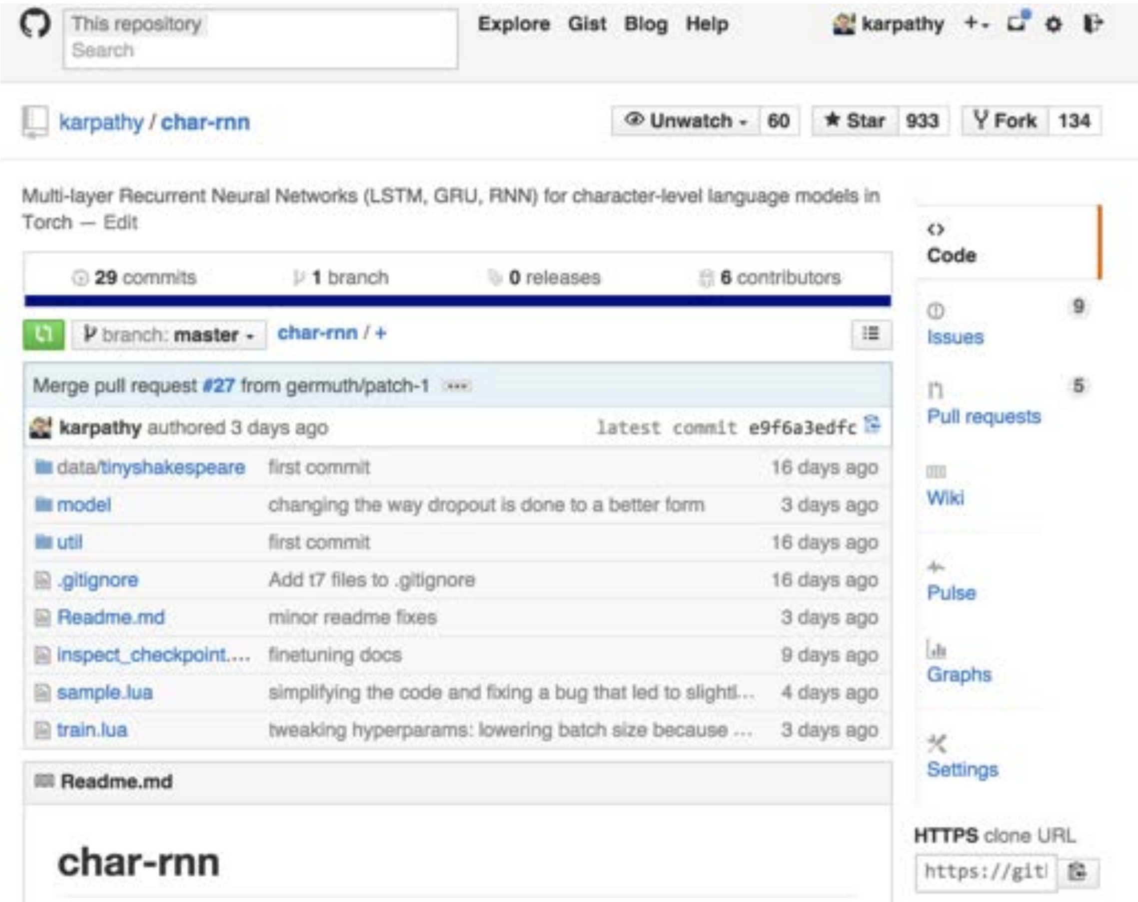
$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

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.

slide credit: Andrej Karpathy

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.
```

slide credit: Andrej Karpathy

Obama Speeches

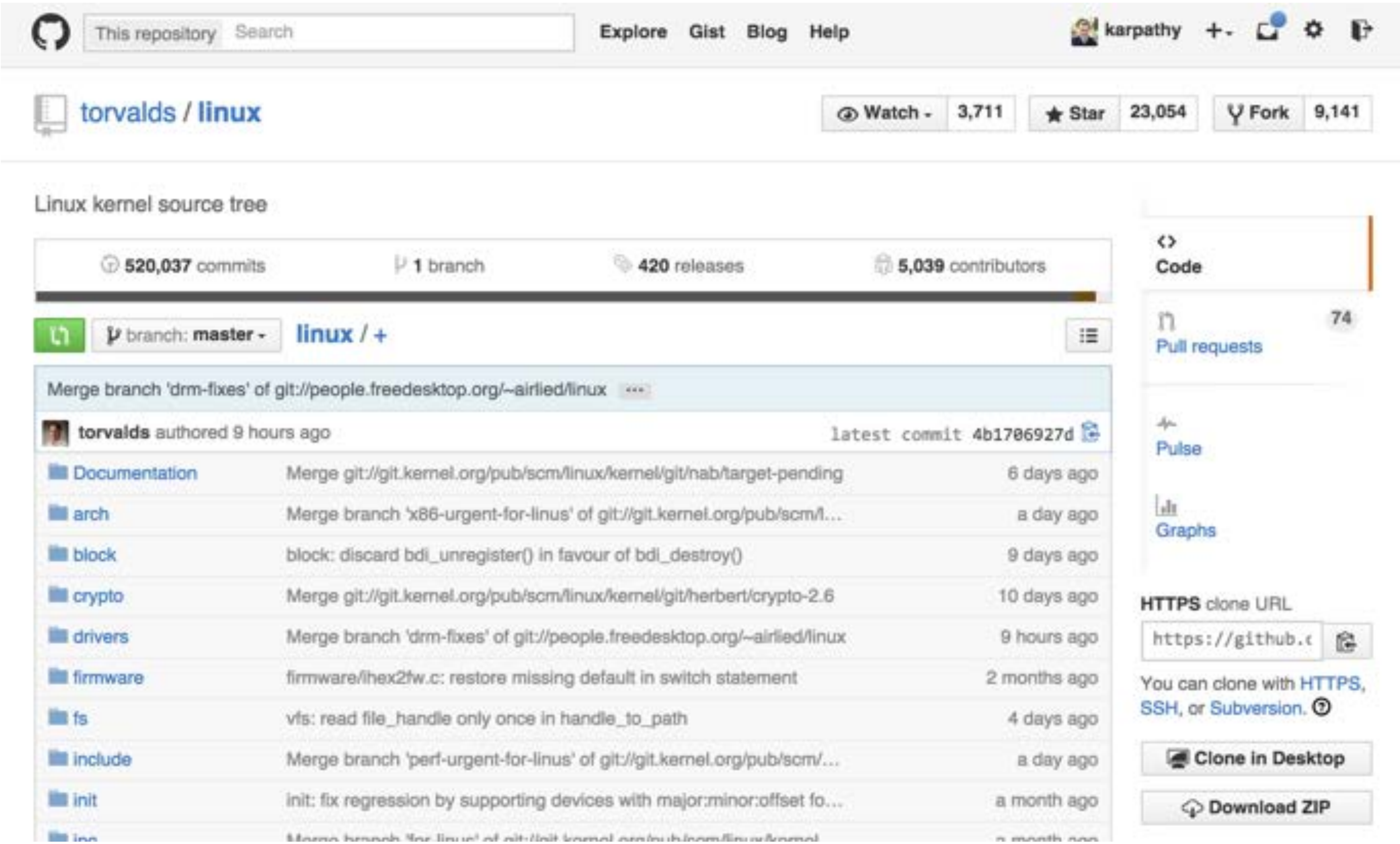
Good afternoon. God bless you.

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

```
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 ");
}
```

Learning from Linux Source Code

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 ");
}
```

Linus Torvalds
Shared publicly - May 24, 2015

I'm not a fan of traditional AI (rule building and LISP/prolog etc), but am just waiting for neural nets to take over.

This is a great example, and training on the Linux kernel sources just explains **so** much.

Sigh.

David Alan Gilbert originally shared:
This is wonderful; neural network setup producing wonderful results; one example is training it on
Read more

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

Visualizing and Understanding Recurrent Networks

Andrej Karpathy*, Justin Johnson*, Li Fei-Fei

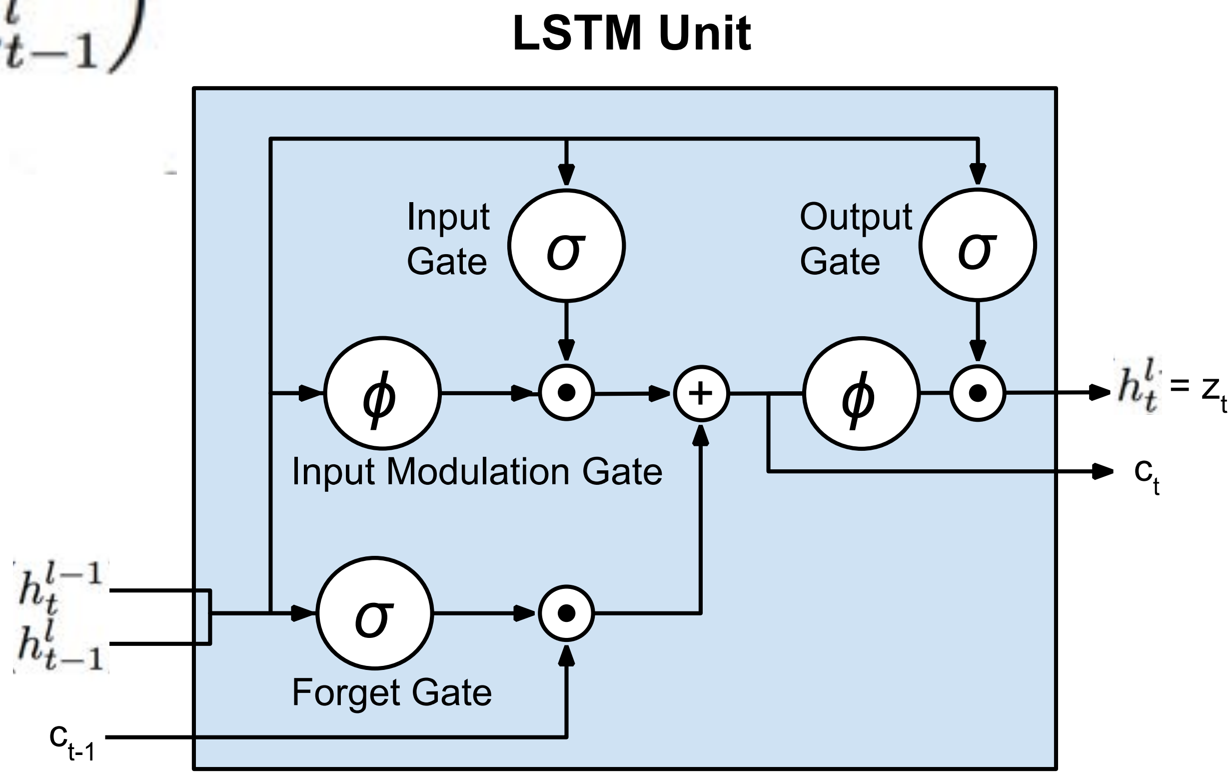
(on arXiv.org as of June 2015)

Hunting interpretable cells

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

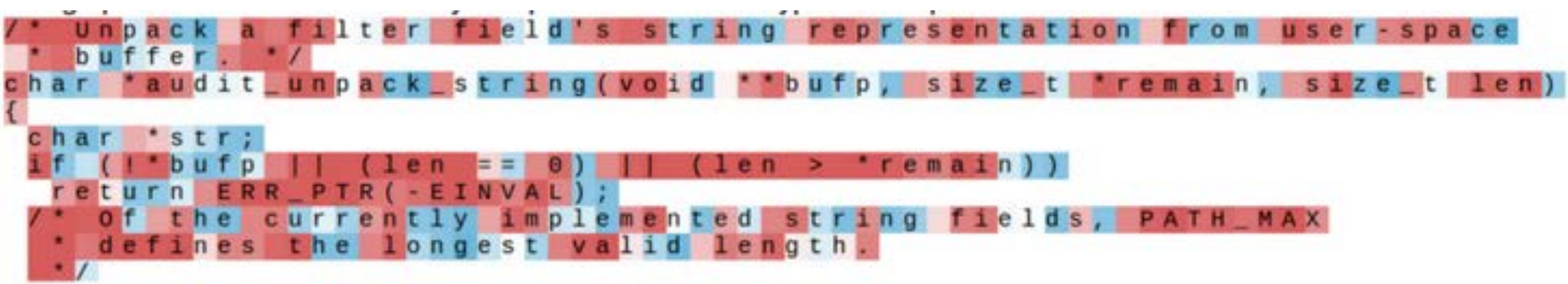
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

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

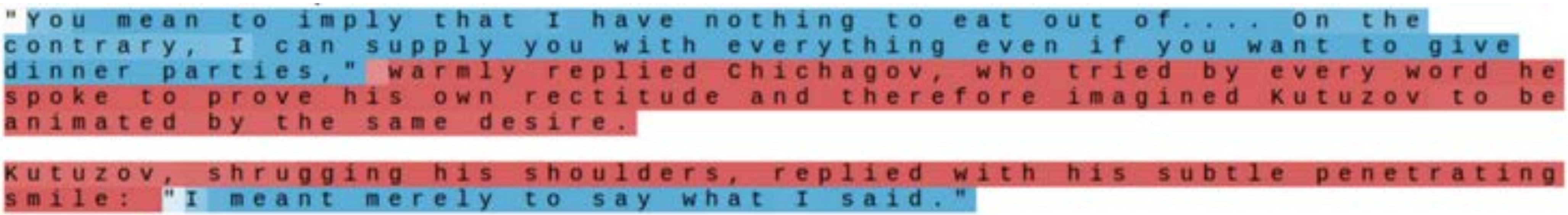


slide credit: Andrej Karpathy

Hunting interpretable cells



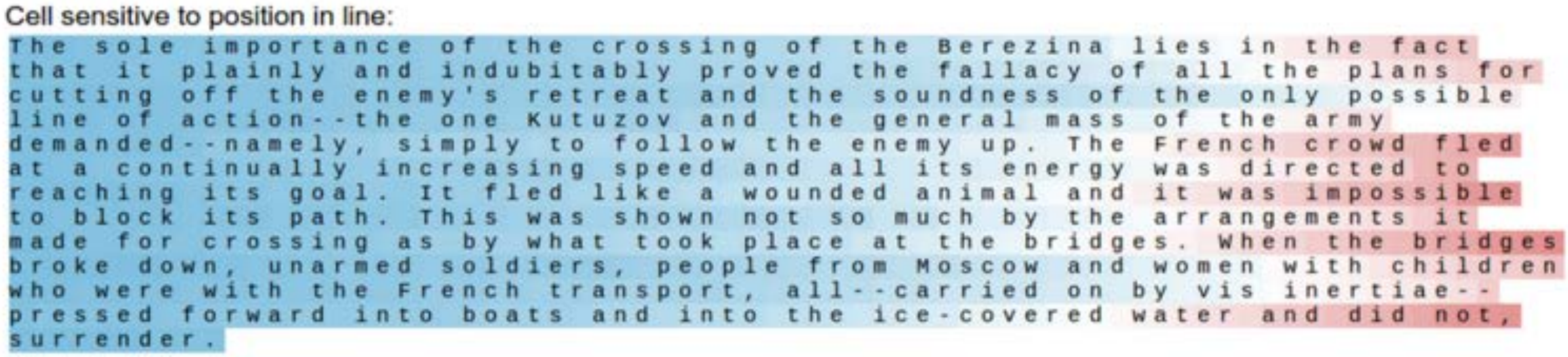
Hunting interpretable cells



quote detection cell

slide credit: Andrej Karpathy

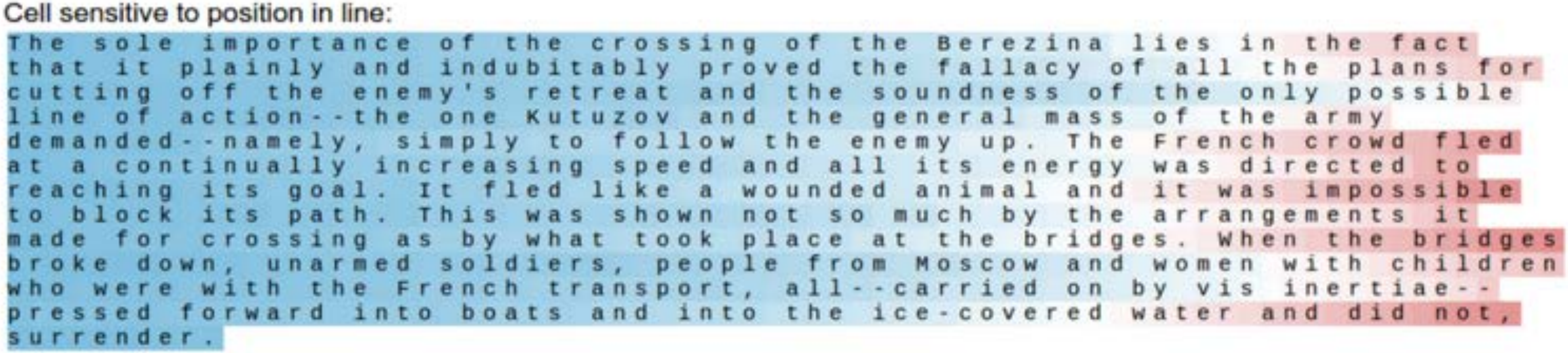
Hunting interpretable cells

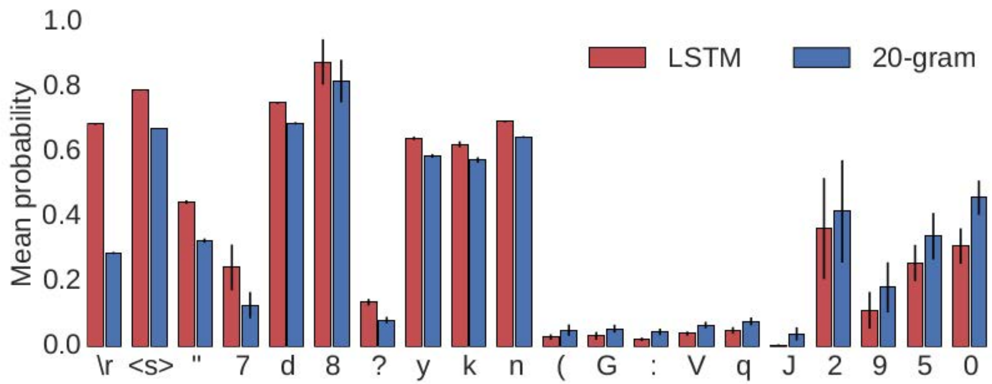


line length tracking cell

slide credit: Andrej Karpathy

Hunting interpretable cells





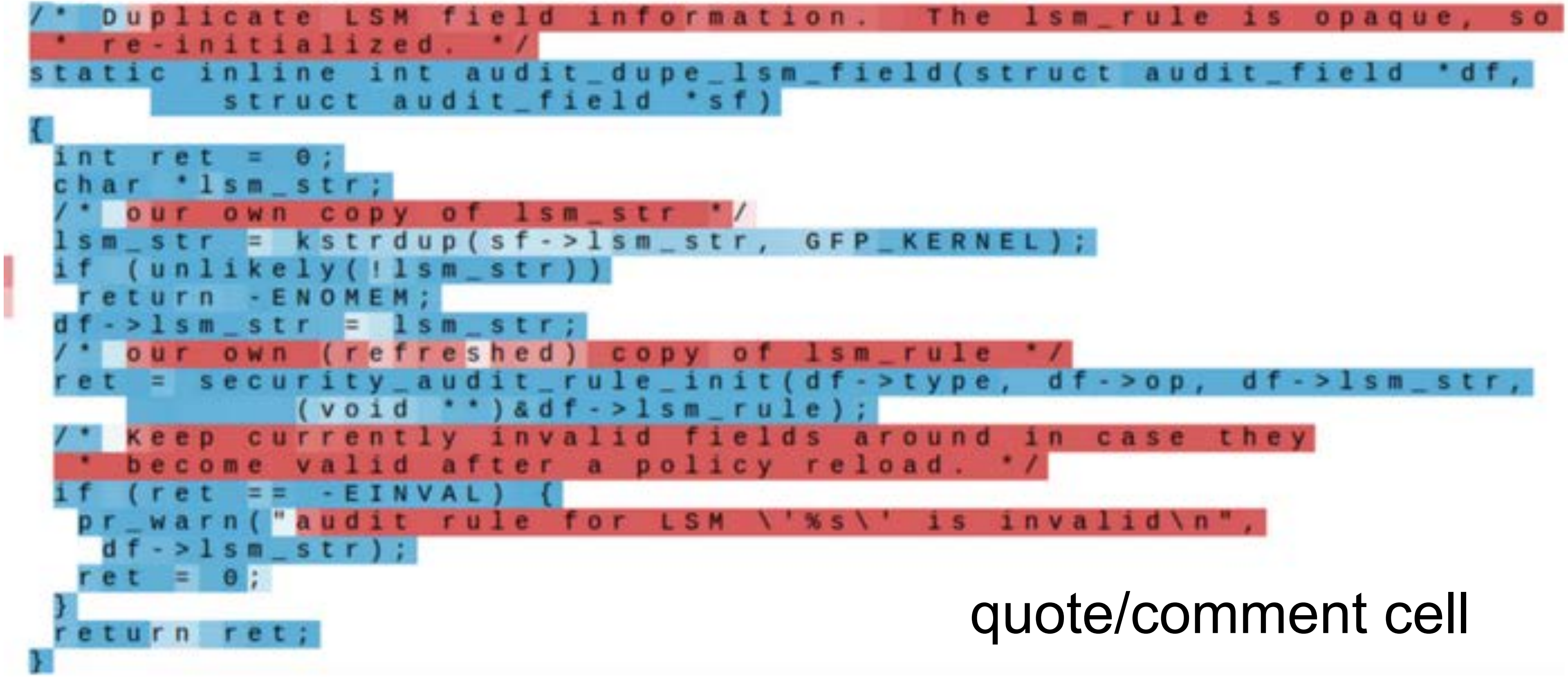
slide credit: Andrej Karpathy

Hunting interpretable cells

if statement cell

slide credit: Andrej Karpathy

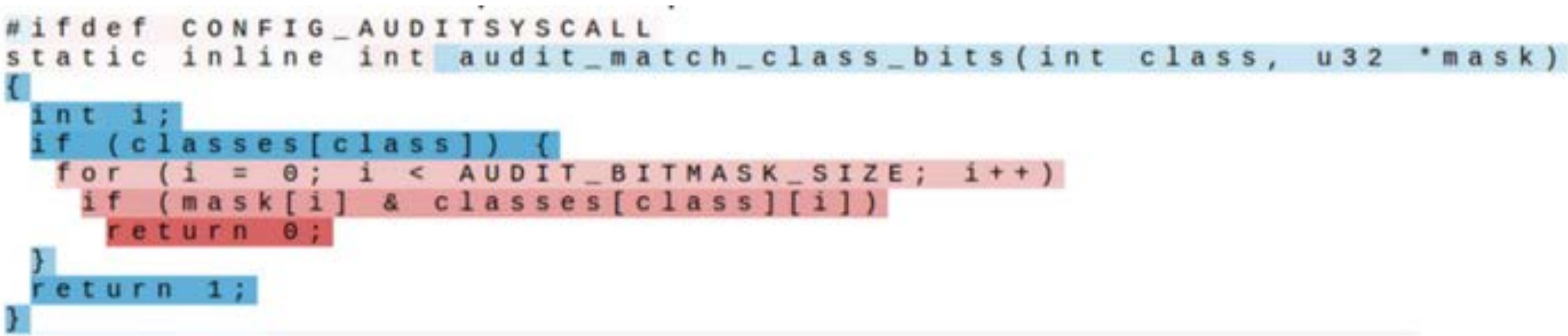
Hunting interpretable cells



quote/comment cell

slide credit: Andrej Karpathy

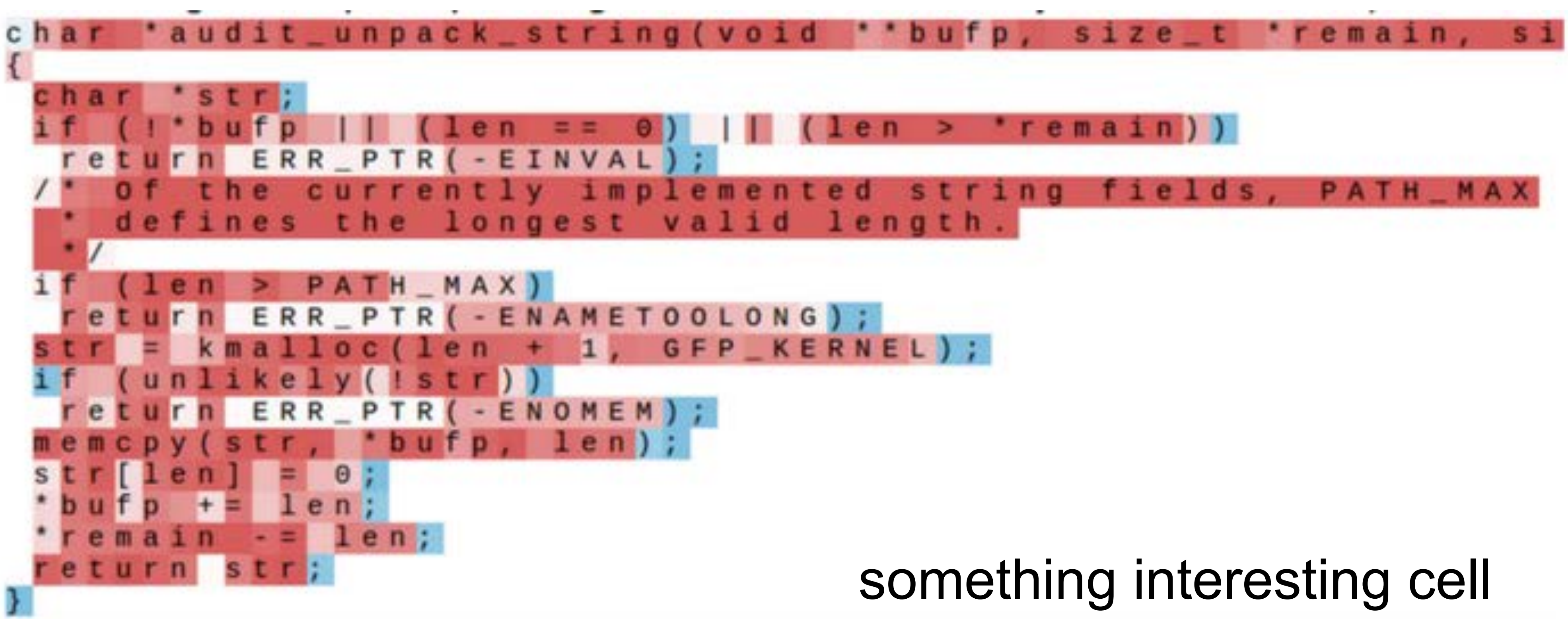
Hunting interpretable cells



code depth cell

slide credit: Andrej Karpathy

Hunting interpretable cells



something interesting cell
(not quite sure what)

What's Wrong With Deep Learning?

Yann LeCun

Facebook AI Research &

Center for Data Science, NYU

yann@cs.nyu.edu

http://yann.lecun.com

slide credit: Yan LeCun

Missing Some Theory

Theory

Why are ConvNets a good architecture?

- Scattering transform
- Mark Tygert's "complex ConvNet"

How many layers do we really need?

- Really?

How many effective free parameters are there in a large ConvNet

- The weights seem to be awfully redundant

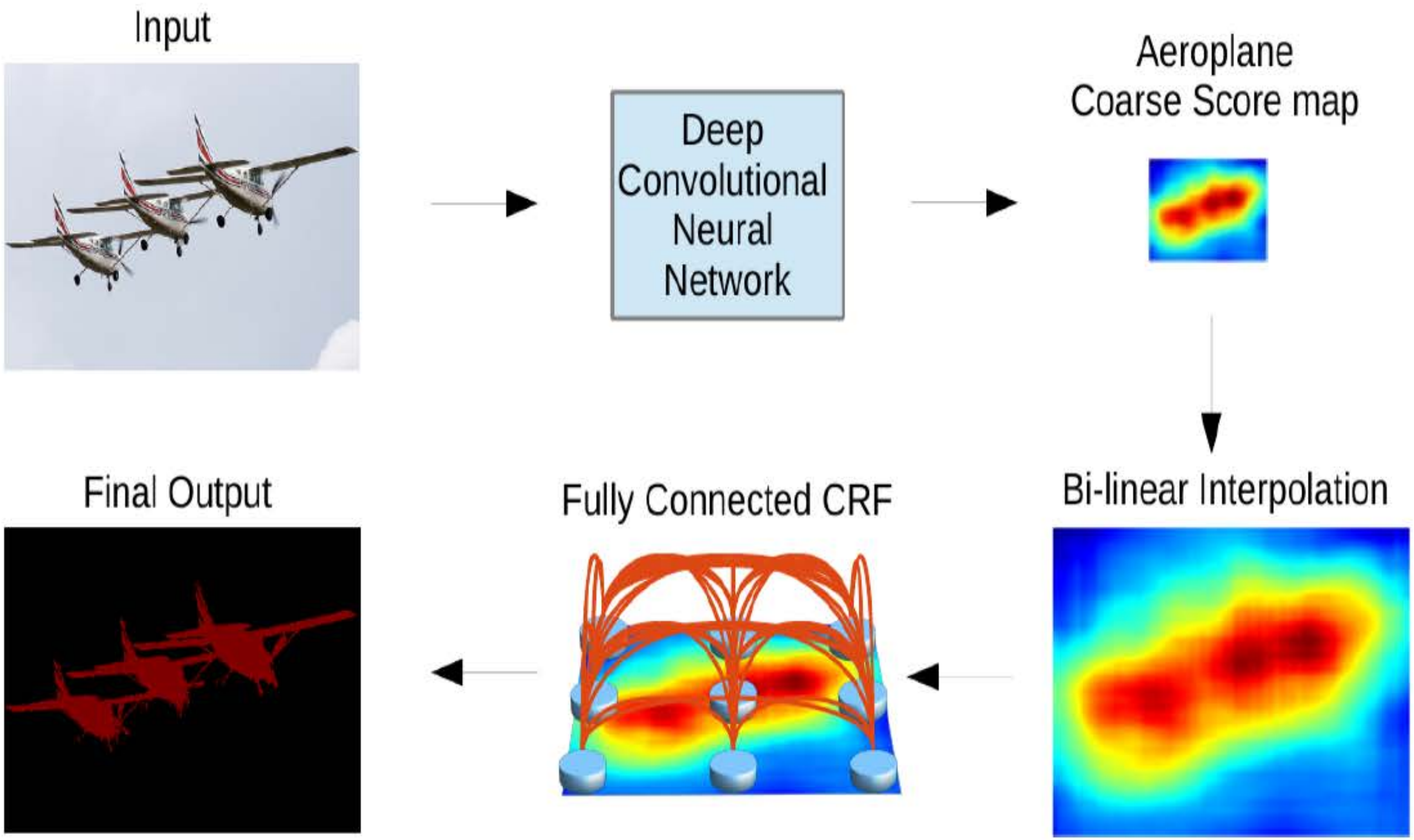
What about Local Minima?

- Turns out almost all the local minima are equivalent
- Local minima are degenerate (very flat in most directions)
- Random matrix / spin glass theory comes to the rescue
- [Choromanska, Henaff, Mathieu, Ben Arous, LeCun AI-stats 2015]

slide credit: Yann LeCun

Missing: Reasoning

System outline



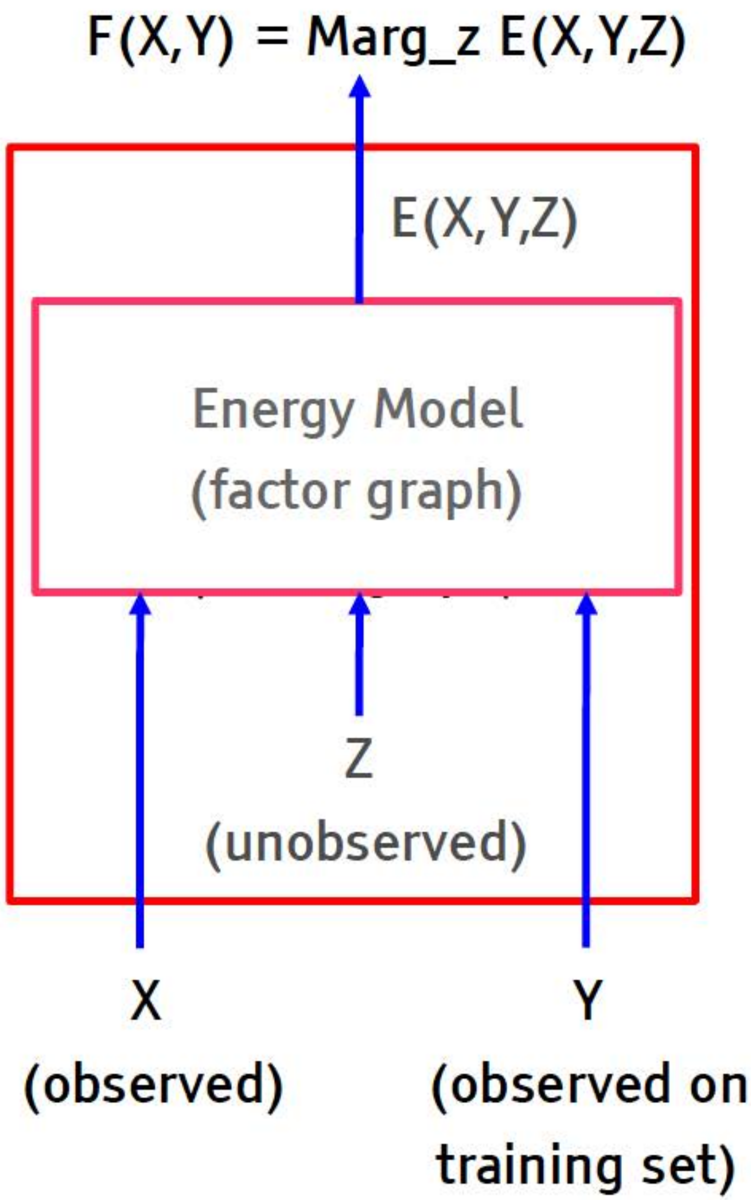
J. Long, E. Shelhamer, T. Darrell, FCNNs for Semantic Segmentation, CVPR 15

P. Krähenbühl and V. Koltun, Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials, NIPS 2011

Reasoning as Energy Minimization (structured prediction++)

Y LeCun

- **Deep Learning systems can be assembled into energy models AKA factor graphs**
  - Energy function is a sum of factors
  - Factors can embed whole deep learning systems
  - X: observed variables (inputs)
  - Z: never observed (latent variables)
  - Y: observed on training set (output variables)
- **Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X**
  - F(X,Y) = MIN_z E(X,Y,Z)
  - F(X,Y) = -log SUM_z exp[-E(X,Y,Z)]

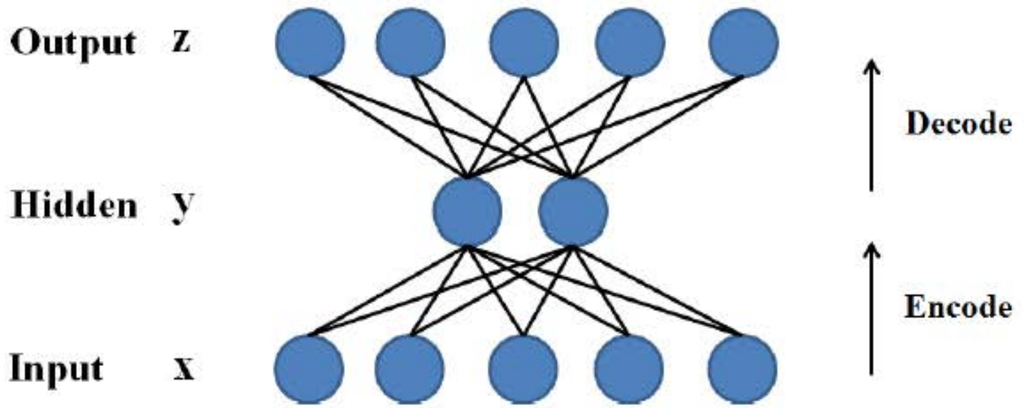


slide credit: Yann LeCun

Missing: Unsupervised Learning

Autoencoder

- An autoencoder takes an input $\mathbf{x} \in [0, 1]^d$ and first maps it (with an encoder) to a hidden representation $\mathbf{y} \in [0, 1]^{d'}$ through a deterministic mapping
$$\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b})$$
where s is a non-linear activation function (such as sigmoid).
- $\mathbf{y}$ is mapped back (with a decoder) into a reconstruction $\mathbf{z}$ of the same shape as $\mathbf{x}$,
$$\mathbf{z} = s(\mathbf{W}'y + \mathbf{b}')$$
$\mathbf{z}$ is seen as a prediction of $\mathbf{x}$.



slide credit: Xiaogang Wang

Autoencoder

- Encoder

$$\mathbf{y} = f_\theta(\mathbf{x})$$

- Decoder

$$\mathbf{z} = g_\theta(\mathbf{y})$$

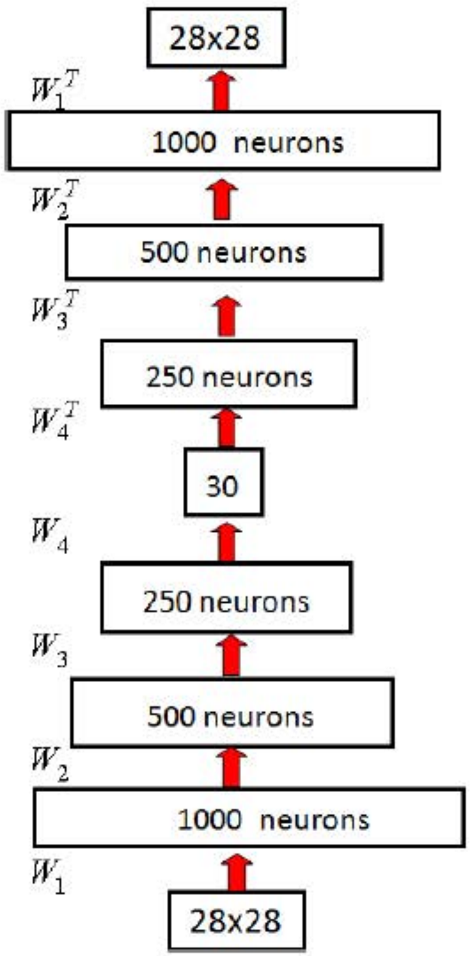
$$\theta = \{\mathbf{W}, \mathbf{W}', \mathbf{b}, \mathbf{b}'\}$$

- It is important to add regularization in the training criterion or the parametrization to prevent the auto-encoder from learning the identity function, which would lead zero reconstruction error everywhere
- A particular form of regularization consists in constraining the code to have a low dimension, and this is what the classical auto-encoder or PCA do.

slide credit: Xiaogang Wang

Deep autoencoder

- Stack multiple encoders (and their corresponding decoders)



Deep autoencoder

- Very difficult to optimize deep autoencoders using backpropagation
- Pre-training + fine-tuning
  - First train a stack of RBMs
  - Then "unroll" them
  - Then fine-tune with backpropagation

Comparison of methods of compressing images