# CS 4803 / 7643: Deep Learning

Topics:

- (Finish) Transfer Learning
- Recurrent Neural Networks

Zsolt Kira

Georgia Tech

# Administrative

- HW3 will be out today
  - Due date pushed back to March 15th to allow you more time

- Next Thursday: FB lecture on attention/transformers

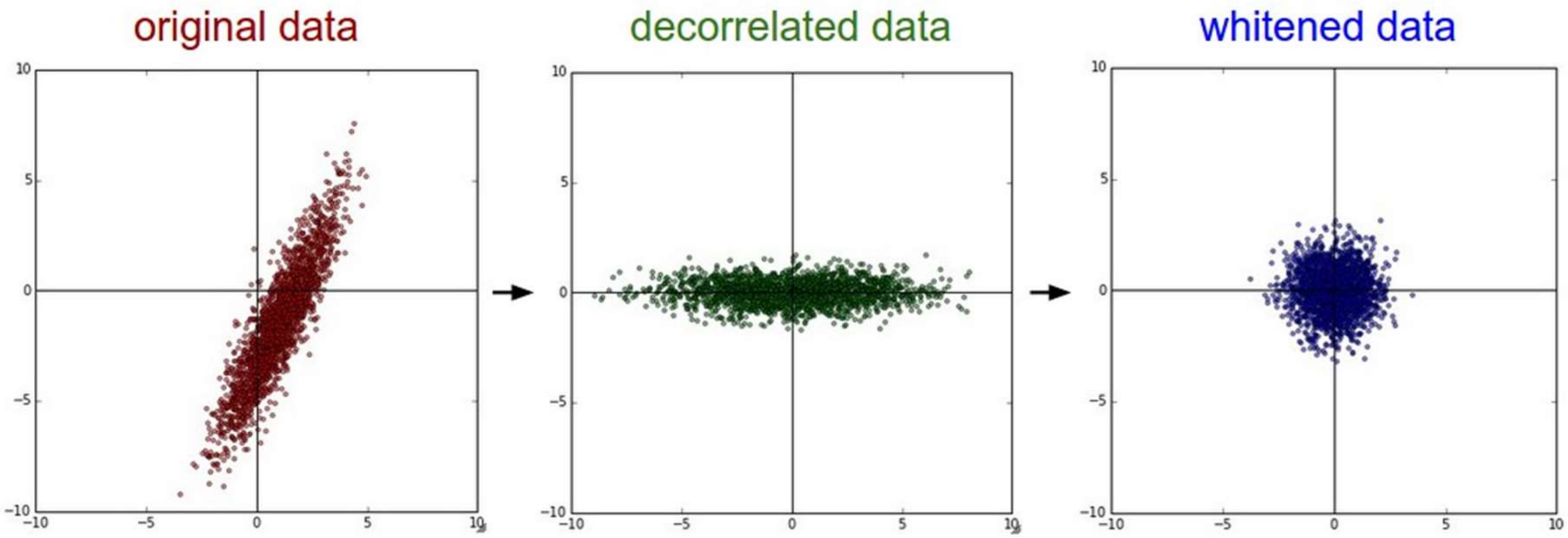
# Part 1

- Activation Functions
- Data Preprocessing
- Weight Initialization
- Batch Normalization
- Babysitting the Learning Process
- Hyperparameter Optimization

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Step 1: Preprocess the data

In practice, you may also see **PCA** and **Whitening** of the data



(data has diagonal covariance matrix)

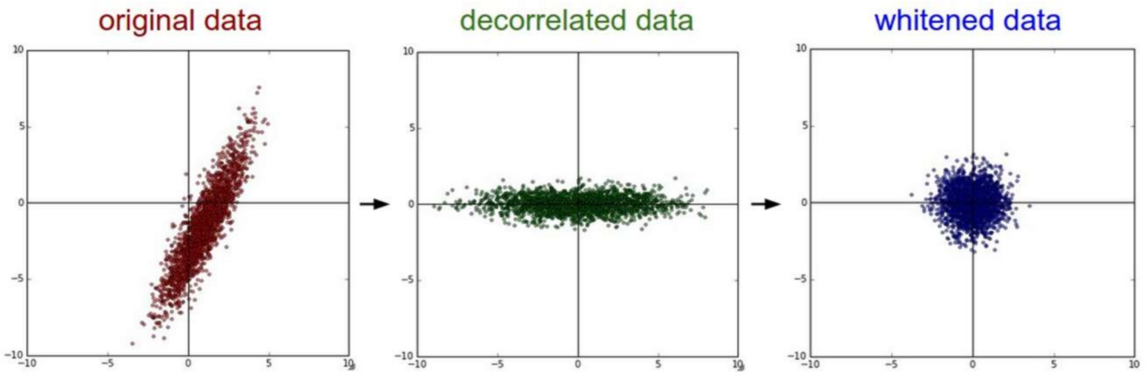
(covariance matrix is the identity matrix)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# **TLDR: In practice for Images:** center only

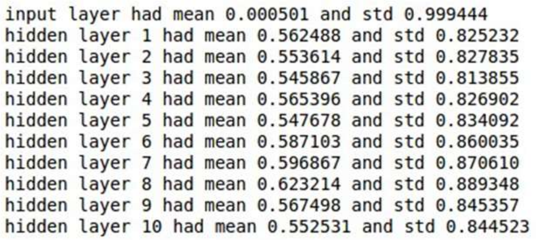
e.g. consider CIFAR-10 example with [32,32,3] images

- Subtract the mean image (e.g. AlexNet)
  (mean image = [32,32,3] array)
- Subtract per-channel mean (e.g. VGGNet)
  (mean along each channel = 3 numbers)

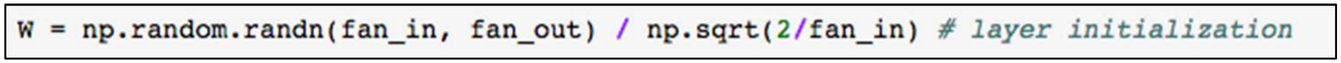


Not common to normalize variance, to do PCA or whitening

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

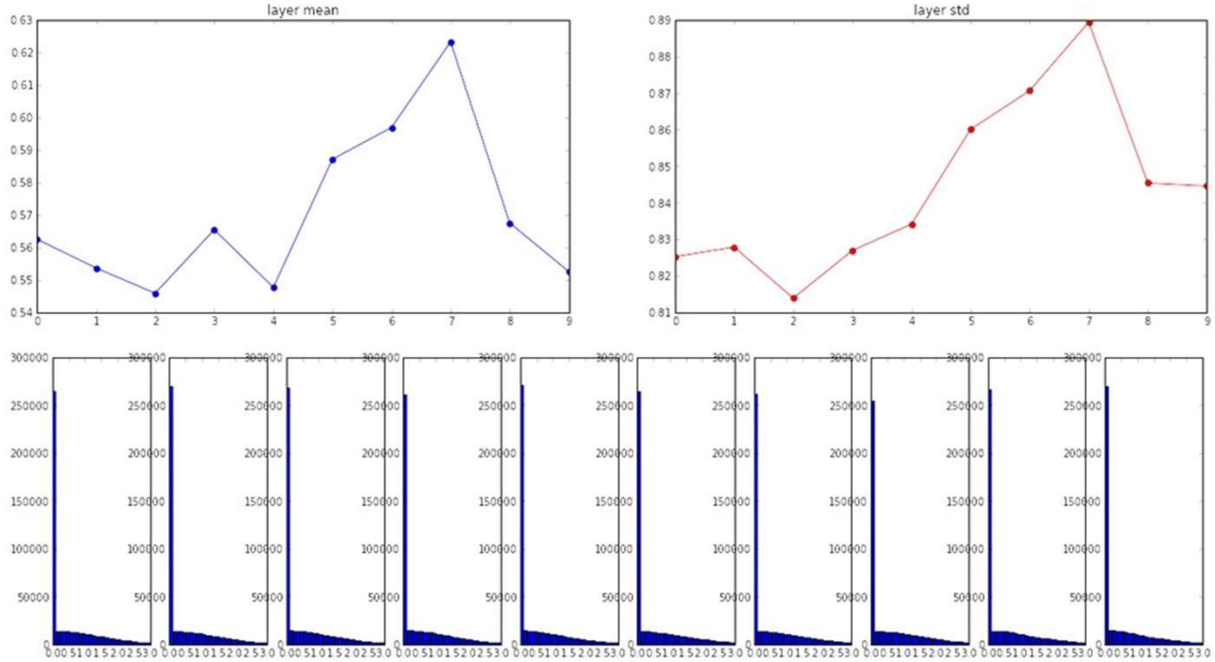


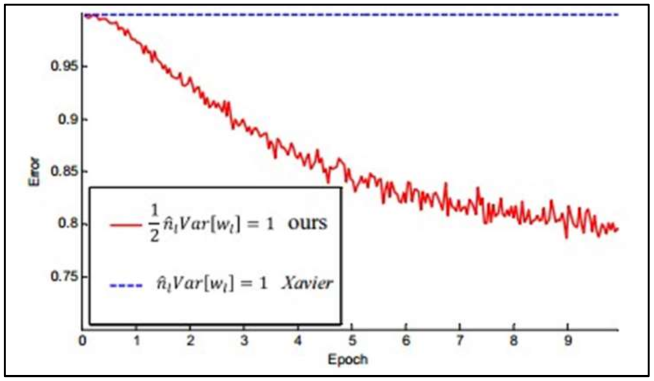
```
input layer had mean 0.000501 and std 0.999444
hidden layer 1 had mean 0.562488 and std 0.825232
hidden layer 2 had mean 0.553614 and std 0.827835
hidden layer 3 had mean 0.545867 and std 0.813855
hidden layer 4 had mean 0.565396 and std 0.826902
hidden layer 5 had mean 0.547678 and std 0.834092
hidden layer 6 had mean 0.587103 and std 0.860035
hidden layer 7 had mean 0.596867 and std 0.870610
hidden layer 8 had mean 0.623214 and std 0.889348
hidden layer 9 had mean 0.567498 and std 0.845357
hidden layer 10 had mean 0.552531 and std 0.844523
```



```
W = np.random.randn(fan_in, fan_out) / np.sqrt(2/fan_in) # layer initialization
```

He et al., 2015
(note additional 2/)

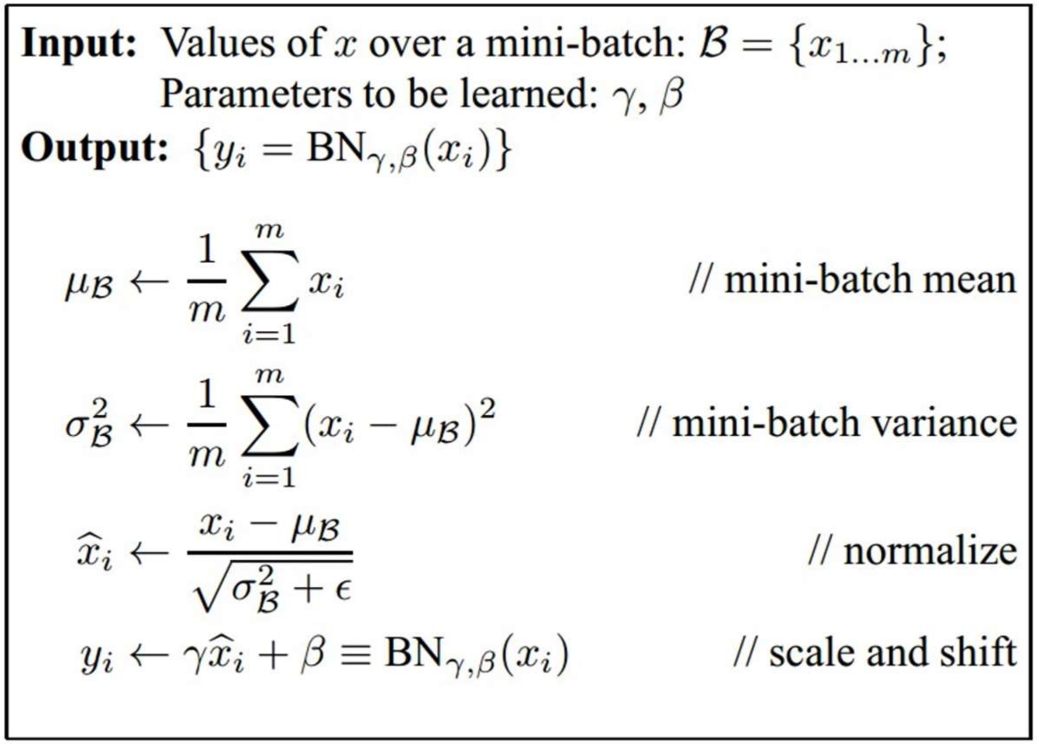




Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Batch Normalization

[Ioffe and Szegedy, 2015]



```
Input: Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
       Parameters to be learned: $\gamma$, $\beta$
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

    $\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$    // mini-batch mean
    $\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2$    // mini-batch variance
    $\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}}$    // normalize
    $y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$    // scale and shift
```

- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Acts as a form of regularization in a funny way, and slightly reduces the need for dropout, maybe

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Summary

TLDRs

We looked in detail at:

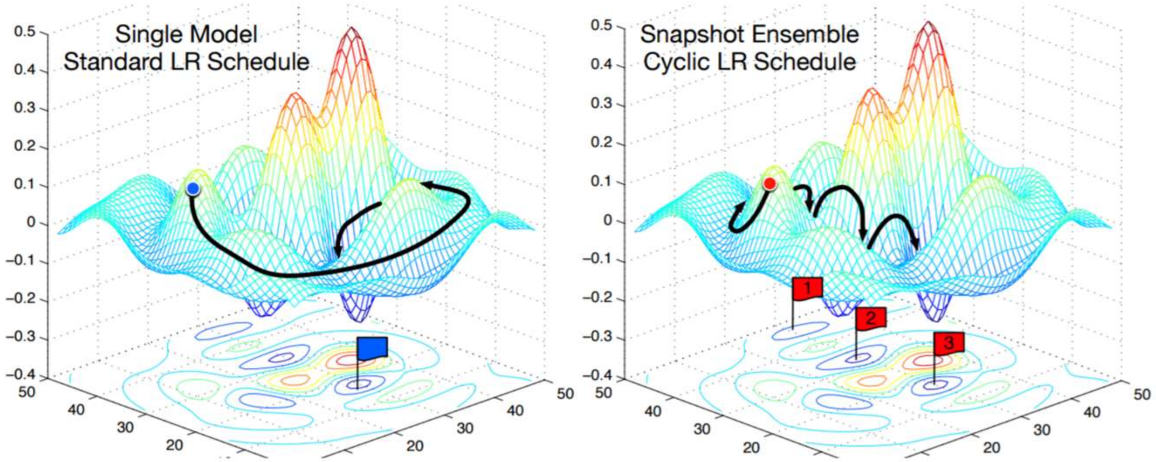
- Activation Functions (use ReLU)
- Data Preprocessing (images: subtract mean)
- Weight Initialization (use Xavier/He init)
- Batch Normalization (use)
- Babysitting the Learning process
- Hyperparameter Optimization

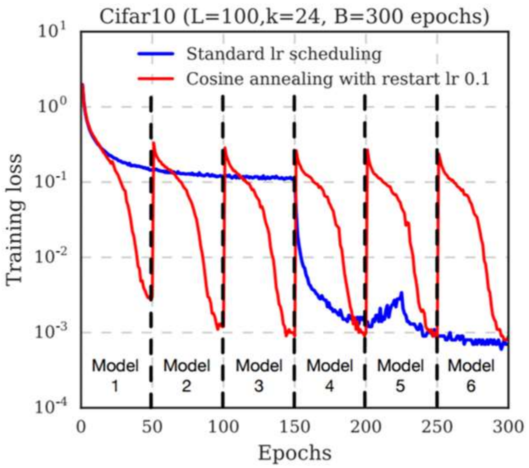
(random sample hyperparams, in log space when appropriate)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Model Ensembles: Tips and Tricks

Instead of training independent models, use multiple snapshots of a single model during training!



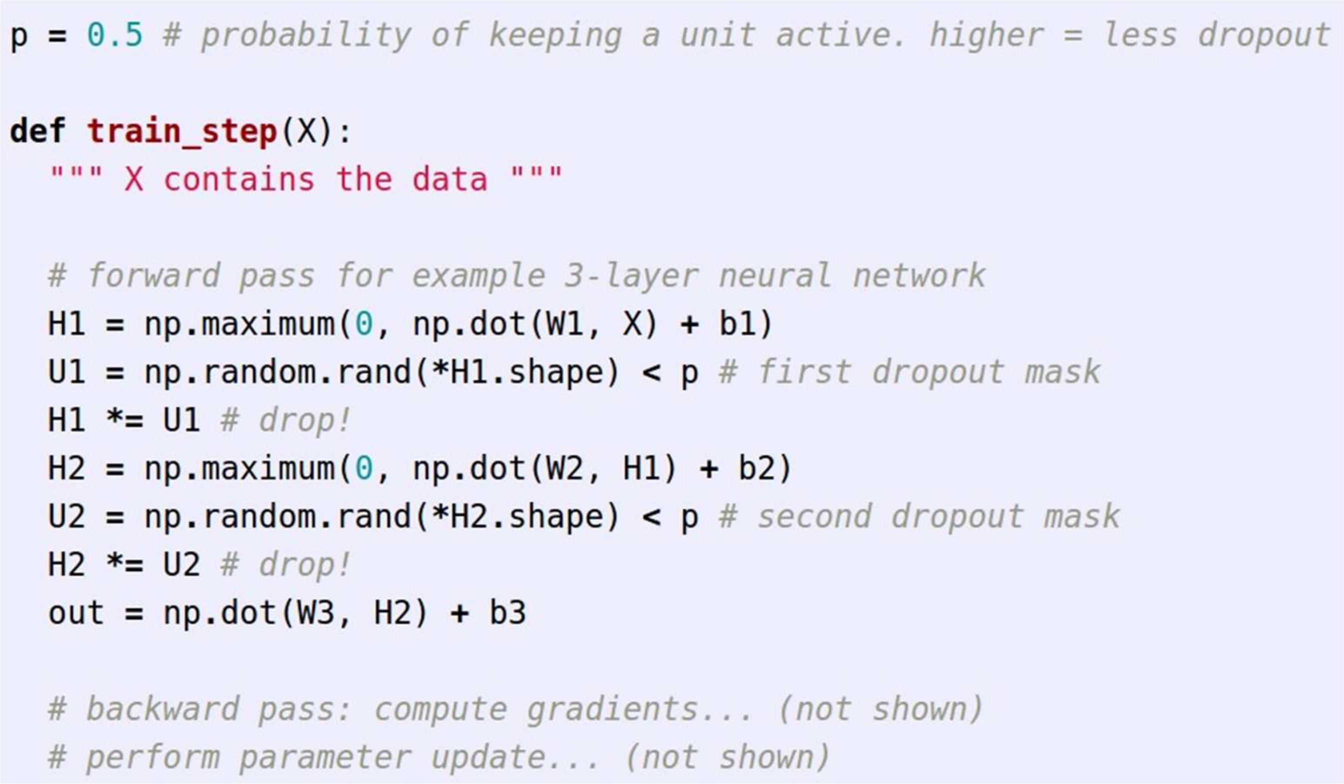


Loshchilov and Hutter, “SGDR: Stochastic gradient descent with restarts”, arXiv 2016
Huang et al, “Snapshot ensembles: train 1, get M for free”, ICLR 2017
Figures copyright Yixuan Li and Geoff Pleiss, 2017. Reproduced with permission.

Cyclic learning rate schedules can make this work even better!

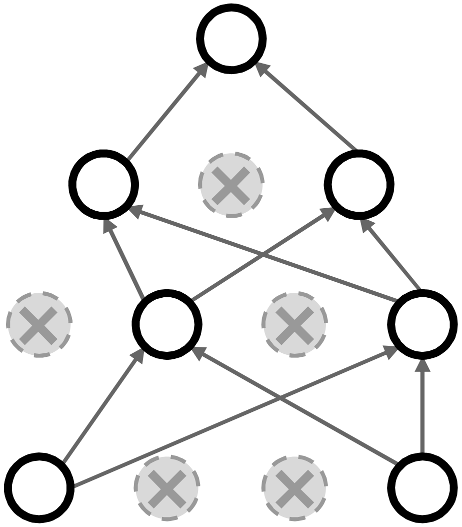
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Regularization: Dropout



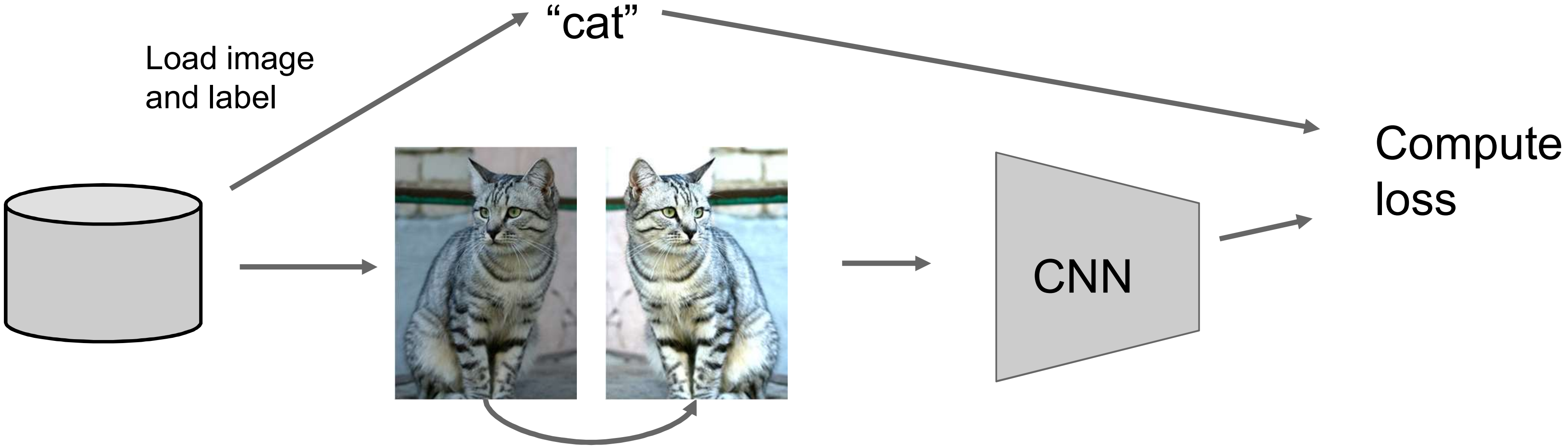
```
p = 0.5 # probability of keeping a unit active. higher = less dropout

def train_step(X):
  """ X contains the data """

  # forward pass for example 3-layer neural network
  H1 = np.maximum(0, np.dot(W1, X) + b1)
  U1 = np.random.rand(*H1.shape) < p # first dropout mask
  H1 *= U1 # drop!
  H2 = np.maximum(0, np.dot(W2, H1) + b2)
  U2 = np.random.rand(*H2.shape) < p # second dropout mask
  H2 *= U2 # drop!
  out = np.dot(W3, H2) + b3

  # backward pass: compute gradients... (not shown)
  # perform parameter update... (not shown)
```

Example forward pass with a 3-layer network using dropout



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Regularization: Data Augmentation



“cat”

Load image and label

CNN

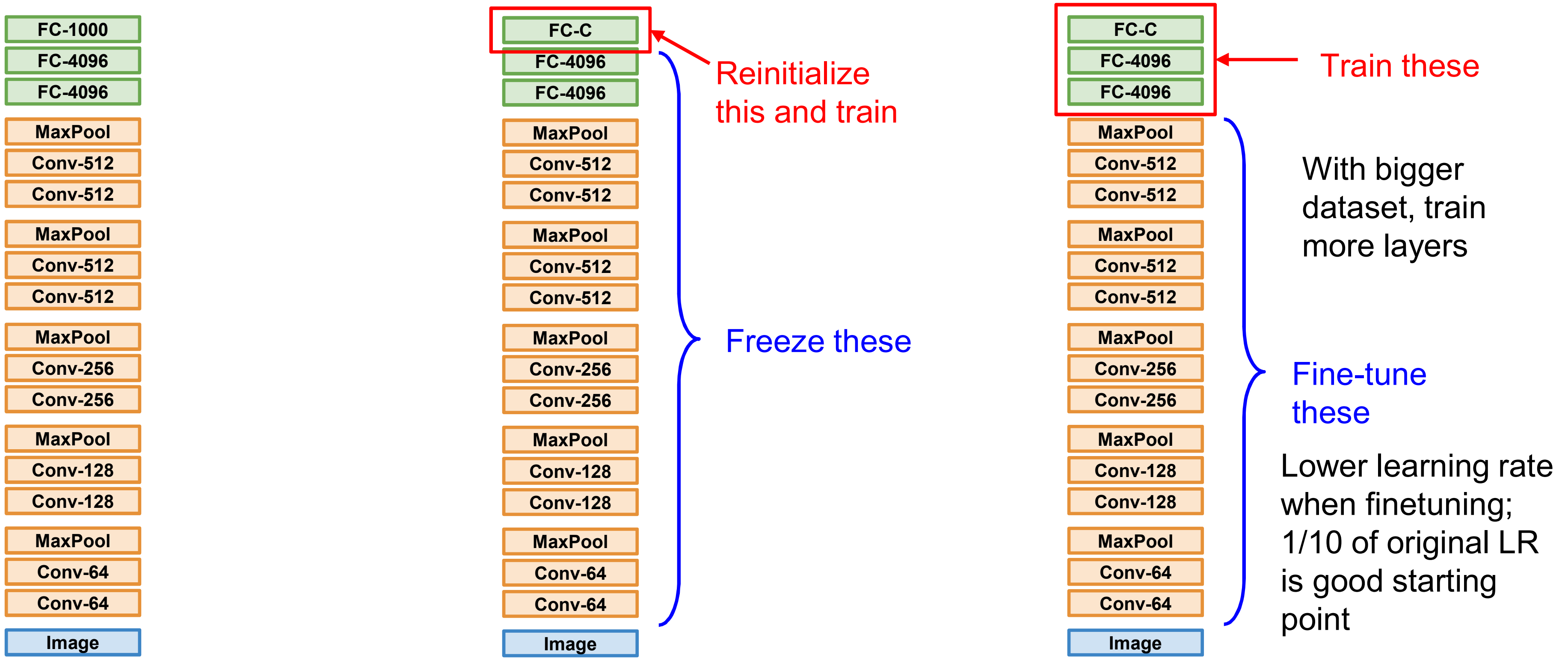
Compute loss

Transform image

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014
Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

# Transfer Learning with CNNs



1. Train on Imagenet

FC-1000
FC-4096
FC-4096
MaxPool
Conv-512
Conv-512
MaxPool
Conv-512
Conv-512
MaxPool
Conv-256
Conv-256
MaxPool
Conv-128
Conv-128
MaxPool
Conv-64
Conv-64
Image

2. Small Dataset (C classes)

FC-C
FC-4096
FC-4096
MaxPool
Conv-512
Conv-512
MaxPool
Conv-512
Conv-512
MaxPool
Conv-256
Conv-256
MaxPool
Conv-128
Conv-128
MaxPool
Conv-64
Conv-64
Image

Reinitialize this and train

Freeze these

3. Bigger dataset

FC-C
FC-4096
FC-4096
MaxPool
Conv-512
Conv-512
MaxPool
Conv-512
Conv-512
MaxPool
Conv-256
Conv-256
MaxPool
Conv-128
Conv-128
MaxPool
Conv-64
Conv-64
Image

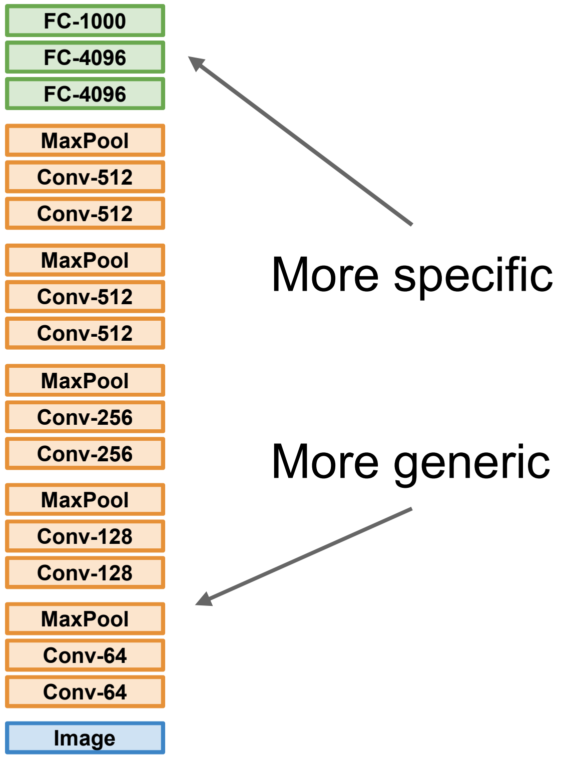
Train these

With bigger dataset, train more layers

Fine-tune these

Lower learning rate when finetuning; 1/10 of original LR is good starting point

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



FC-1000

FC-4096

FC-4096

MaxPool

Conv-512

Conv-512

MaxPool

Conv-512

Conv-512

MaxPool

Conv-256

Conv-256

MaxPool

Conv-128

Conv-128

MaxPool

Conv-64

Conv-64

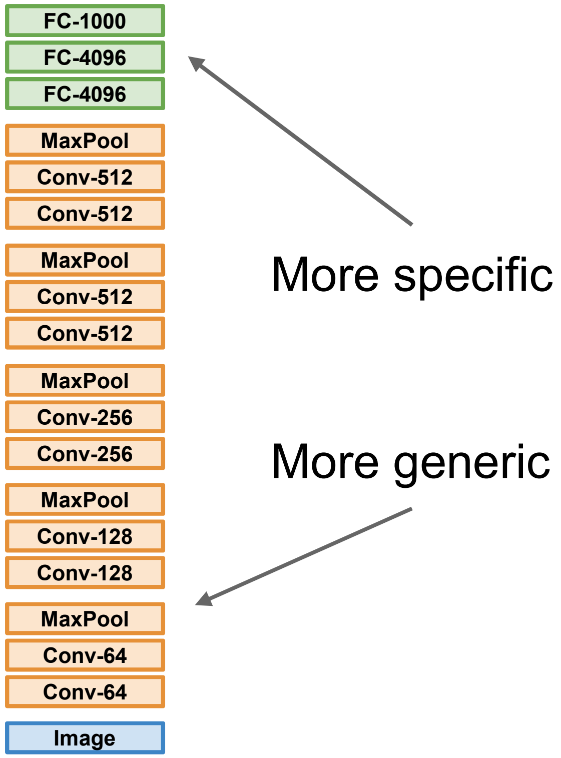
Image

More specific

More generic

| | **very similar dataset** | **very different dataset** |
|---|---|---|
| **very little data** | ? | ? |
| **quite a lot of data** | ? | ? |

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



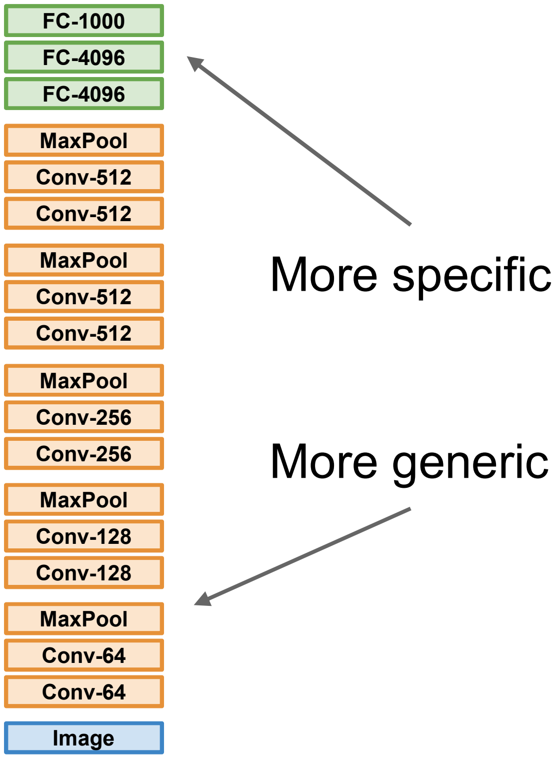
FC-1000<br>
FC-4096<br>
FC-4096<br>
MaxPool<br>
Conv-512<br>
Conv-512<br>
MaxPool<br>
Conv-512<br>
Conv-512<br>
MaxPool<br>
Conv-256<br>
Conv-256<br>
MaxPool<br>
Conv-128<br>
Conv-128<br>
MaxPool<br>
Conv-64<br>
Conv-64<br>
Image

More specific

More generic

| | **very similar dataset** | **very different dataset** |
|---|---|---|
| **very little data** | Use Linear Classifier on top layer | ? |
| **quite a lot of data** | Finetune a few layers | ? |

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



FC-1000
FC-4096
FC-4096
MaxPool
Conv-512
Conv-512
MaxPool
Conv-512
Conv-512
MaxPool
Conv-256
Conv-256
MaxPool
Conv-128
Conv-128
MaxPool
Conv-64
Conv-64
Image

More specific

More generic

| | **very similar dataset** | **very different dataset** |
|---|---|---|
| **very little data** | Use Linear Classifier on top layer | You’re in trouble… Try linear classifier from different stages |
| **quite a lot of data** | Finetune a few layers | Finetune a larger number of layers |

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Transfer learning with CNNs is pervasive…
## (it’s the norm, not an exception)

Object Detection
(Fast R-CNN)

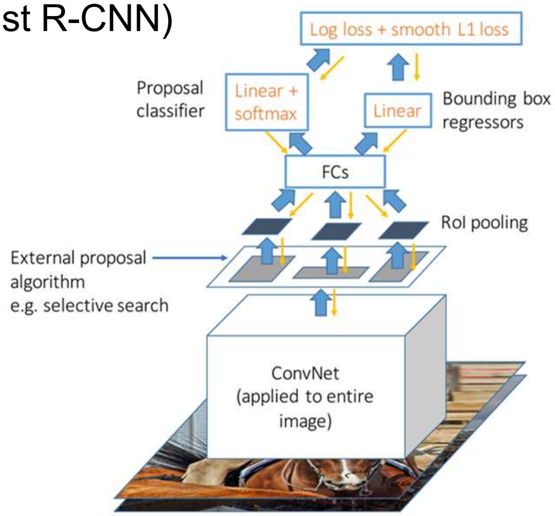
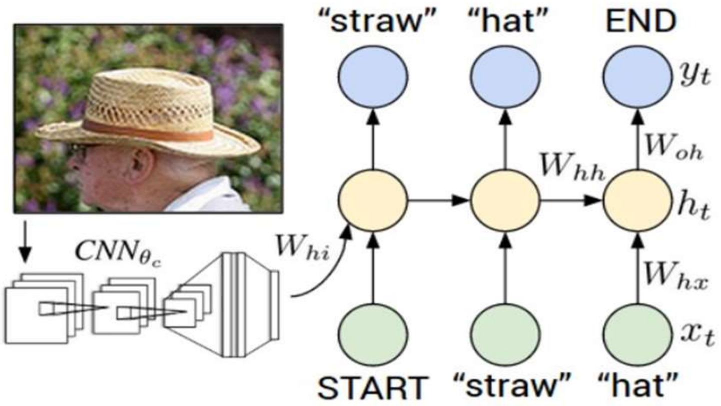

Log loss + smooth L1 loss
Proposal classifier
Linear + softmax
Linear
Bounding box regressors
FCs
RoI pooling
External proposal algorithm e.g. selective search
ConvNet (applied to entire image)

Image Captioning: CNN + RNN



Girshick, “Fast R-CNN”, ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

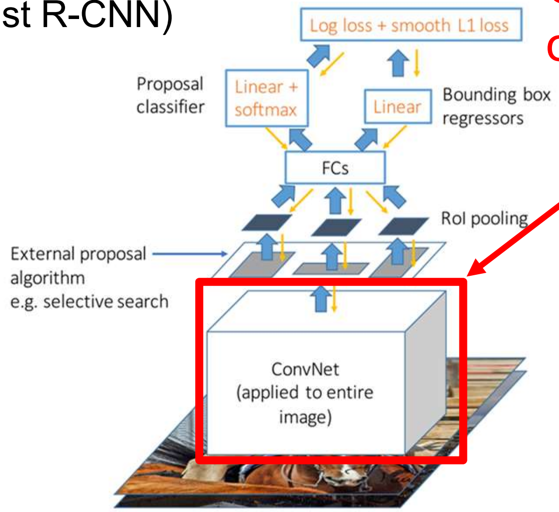
Karpathy and Fei-Fei, “Deep Visual-Semantic Alignments for Generating Image Descriptions”, CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Transfer learning with CNNs is pervasive…

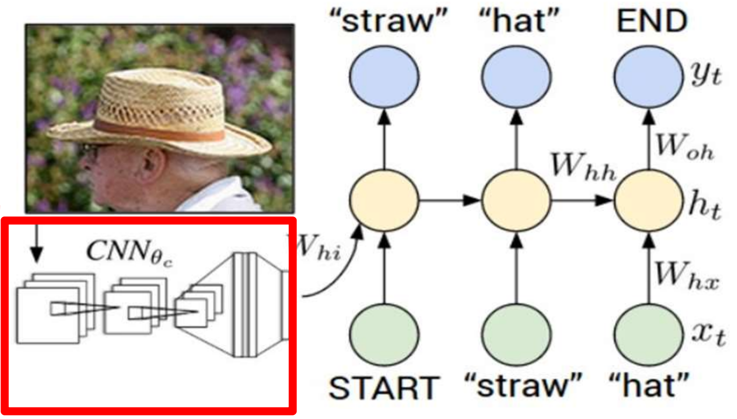
(it’s the norm, not an exception)

Object Detection
(Fast R-CNN)


Log loss + smooth L1 loss
Proposal classifier
Linear + softmax
Linear
Bounding box regressors
FCs
RoI pooling
External proposal algorithm e.g. selective search
ConvNet (applied to entire image)

CNN pretrained on ImageNet

Image Captioning: CNN + RNN



Girshick, “Fast R-CNN”, ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

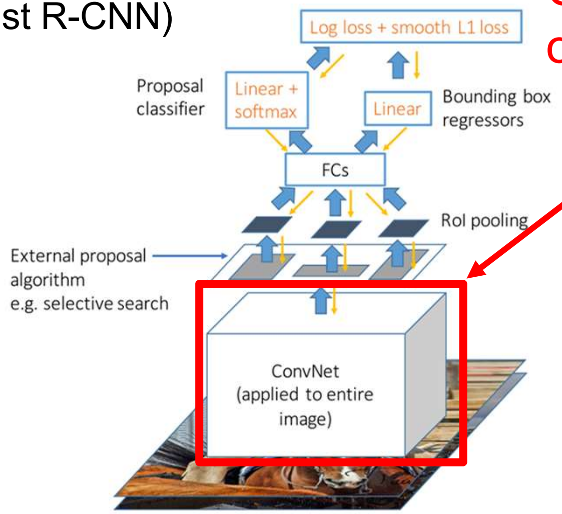
Karpathy and Fei-Fei, “Deep Visual-Semantic Alignments for Generating Image Descriptions”, CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Transfer learning with CNNs is pervasive…

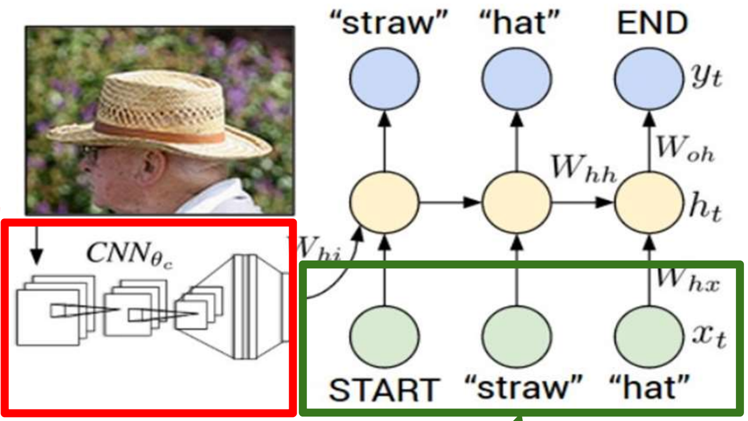
(it's the norm, not an exception)

Object Detection
(Fast R-CNN)


Log loss + smooth L1 loss
Proposal classifier
Linear + softmax
Linear
Bounding box regressors
FCs
RoI pooling
External proposal algorithm e.g. selective search
ConvNet (applied to entire image)

CNN pretrained
on ImageNet

Image Captioning: CNN + RNN



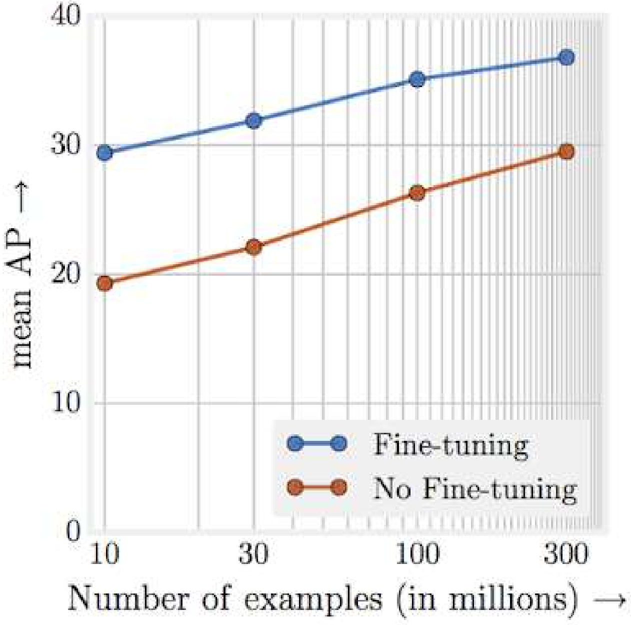
Word vectors pretrained
with word2vec

Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# The most effective method: Gather more data!



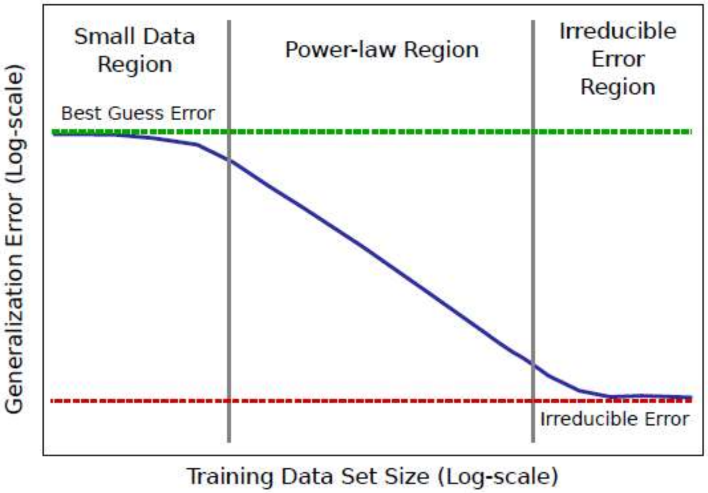


Figure 6: Sketch of power-law learning curves

**Revisiting the Unreasonable Effectiveness of Data**
https://ai.googleblog.com/2017/07/revisiting-unreasonable-effectiveness.html

DEEP LEARNING SCALING IS PREDICTABLE, EMPIRICALLY
Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, Yanqi Zhou

## Takeaway for your projects and beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

Deep learning frameworks provide a “Model Zoo” of pretrained models so you don’t need to train your own

Caffe: https://github.com/BVLC/caffe/wiki/Model-Zoo
TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Life is never so simple

There are several areas being researched

- Batch size
- Regularization and generalization
- Overparameterization and why SGD is so good

Why is this still not understood?

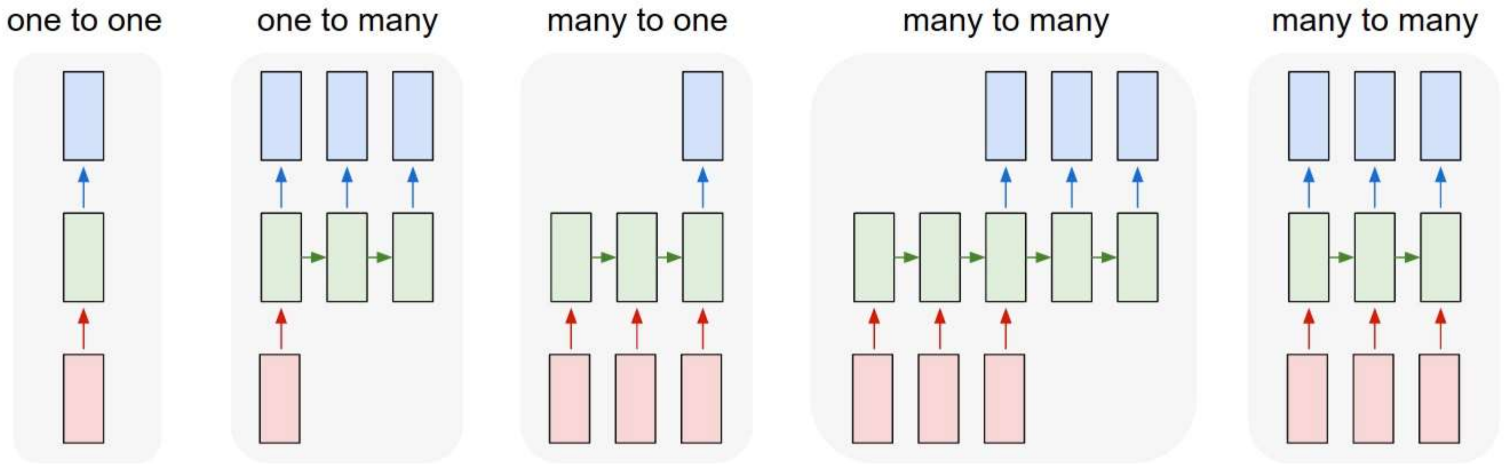
- Our understanding comes from built-in intuition that is repeated but not always tested
- Difficult to apply theory being developed

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Plan for Today

- Recurrent Neural Networks (RNNs)
  - Example Problem: (Character-level) Language modeling
  - Learning: (Truncated) BackProp Through Time (BPTT)
  - Visualizing RNNs
  - Example: Image Captioning
  - Inference: Beam Search
  - Multilayer RNNs
  - Problems with gradients in “vanilla” RNNs
  - LSTMs (and other RNN variants)

# New Topic: RNNs



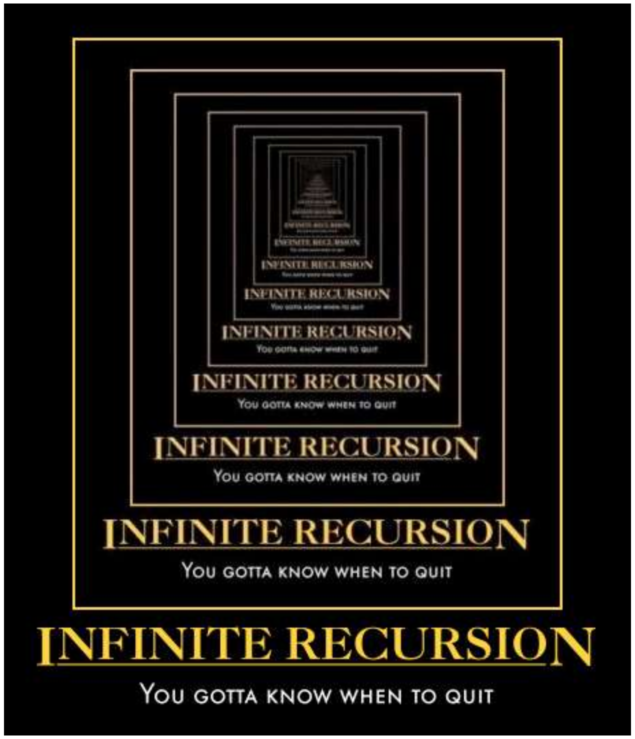


Image Credit: Andrej Karpathy

# Why model sequences?

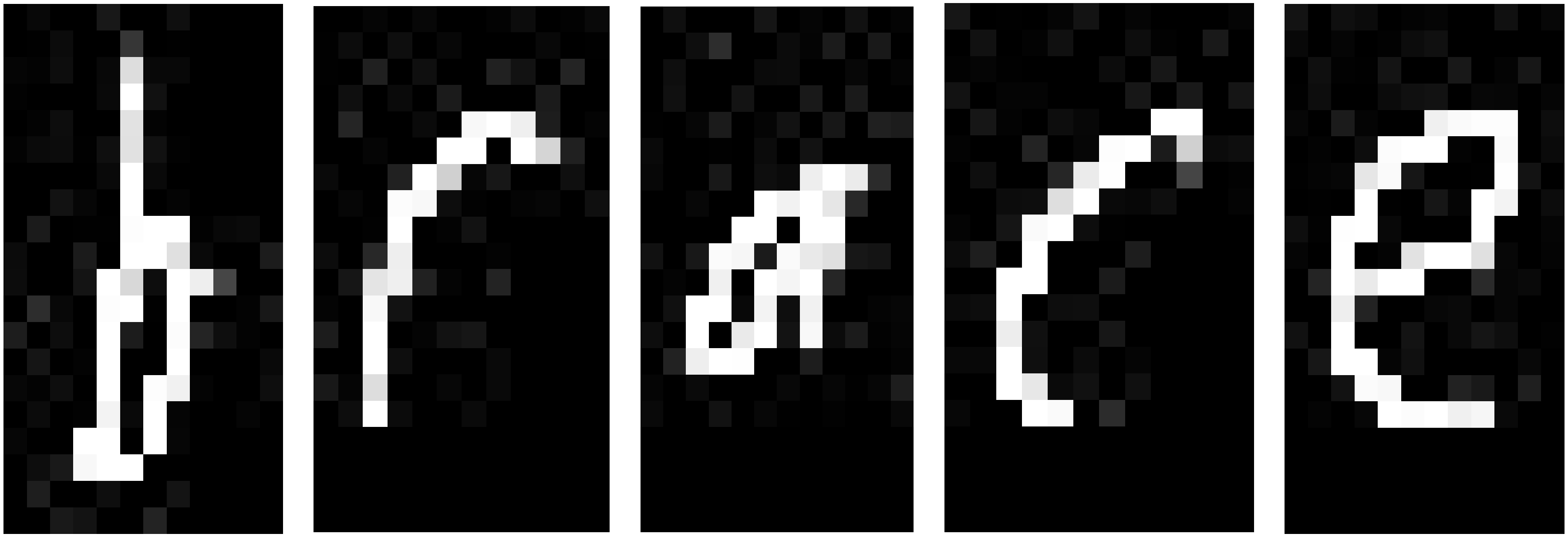


Figure Credit: Carlos Guestrin

# Why model sequences?

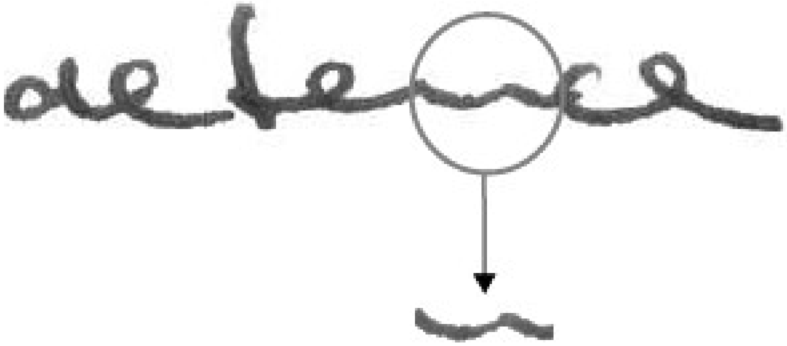


Image Credit: Alex Graves

# Sequences are everywhere…

Foreign Minister.

→ FOREIGN MINISTER.

→ THE SOUND OF

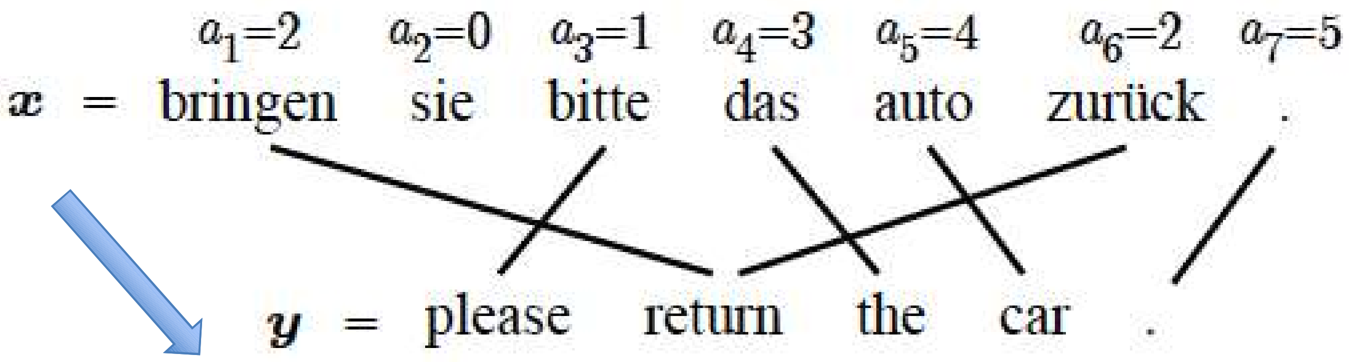

$a_1$=2
$a_2$=0
$a_3$=1
$a_4$=3
$a_5$=4
$a_6$=2
$a_7$=5
$\boldsymbol{x}$ = bringen sie bitte das auto zurück .
$\boldsymbol{y}$ = please return the car .

Image Credit: Alex Graves and Kevin Gimpel

# Even where you might not expect a sequence…

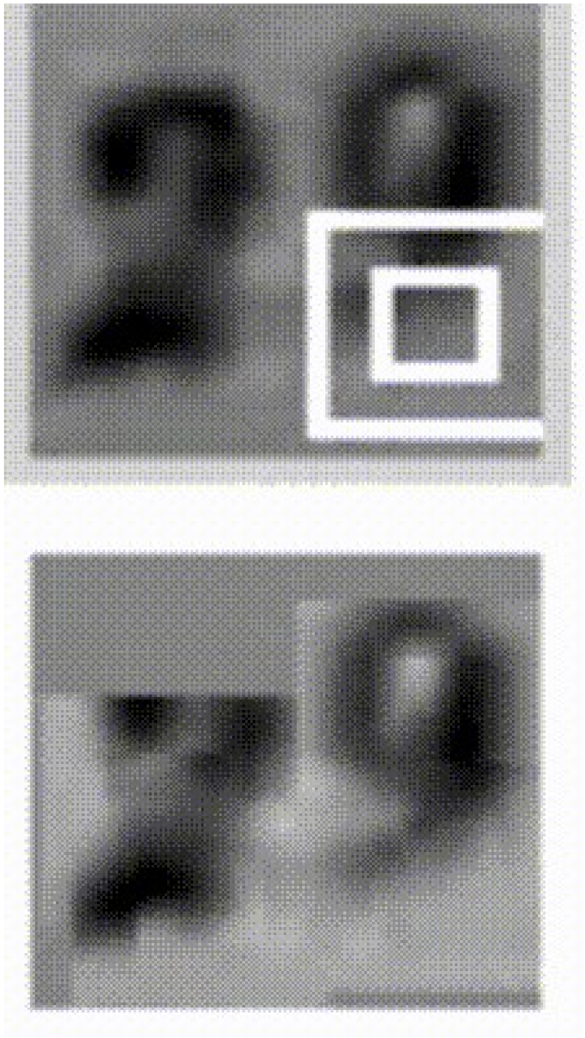
Classify images by taking a series of “glimpses”

Ba, Mnih, and Kavukcuoglu, “Multiple Object Recognition with Visual Attention”, ICLR 2015.
Gregor et al, “DRAW: A Recurrent Neural Network For Image Generation”, ICML 2015
Figure copyright Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra, 2015. Reproduced with permission.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Even where you might not expect a sequence…

- Output ordering = sequence



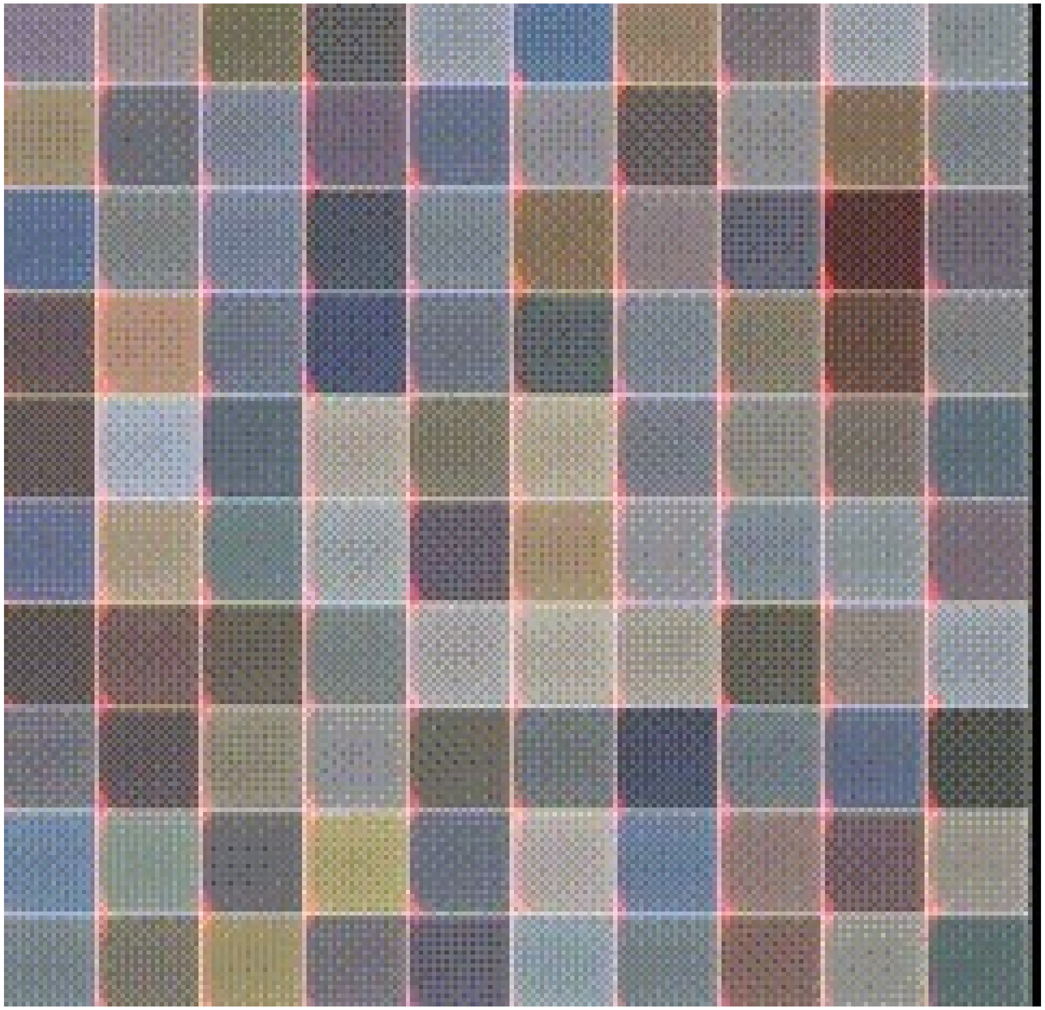
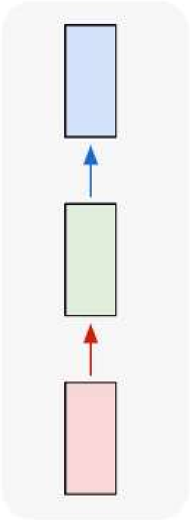


Image Credit: Ba et al.; Gregor et al

# Sequences in Input or Output?

- It’s a spectrum…

one to one



Input: No sequence

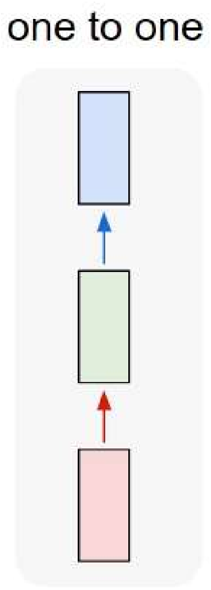
Output: No sequence

Example: “standard” classification / regression problems

Image Credit: Andrej Karpathy

# Sequences in Input or Output?

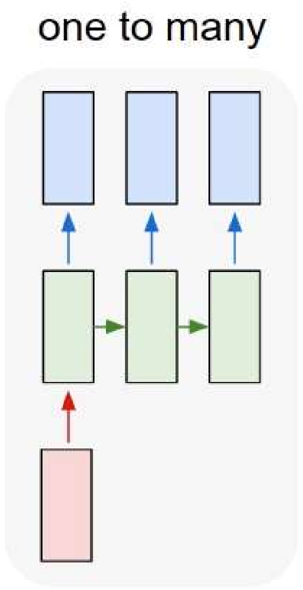
- It's a spectrum…



Input: No sequence

Output: No sequence

Example: "standard" classification / regression problems



Input: No sequence

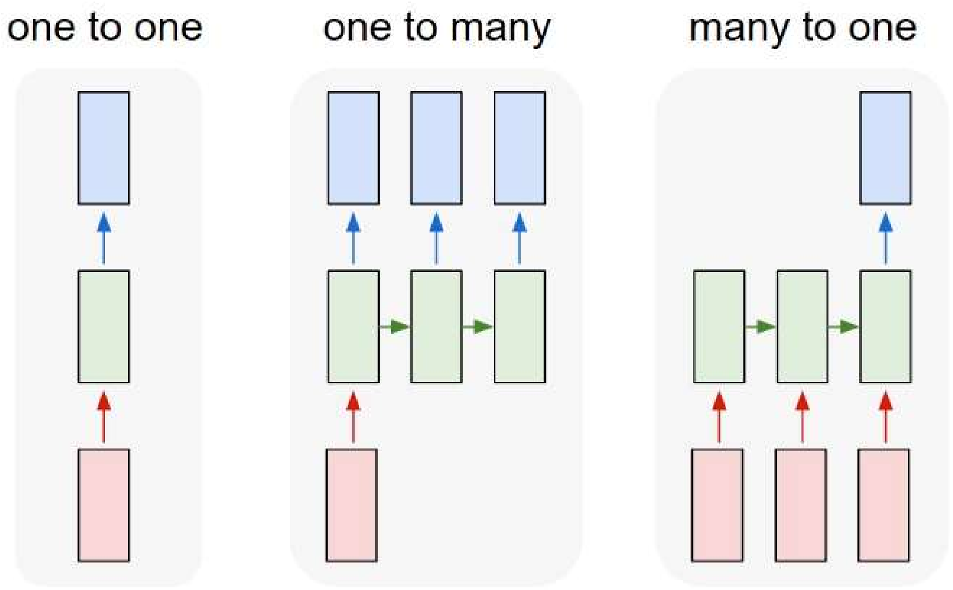
Output: Sequence

Example: Im2Caption

Image Credit: Andrej Karpathy

# Sequences in Input or Output?

- It’s a spectrum…



Input: No sequence

Output: No sequence

Example: “standard” classification / regression problems

Input: No sequence

Output: Sequence

Example: Im2Caption

Input: Sequence

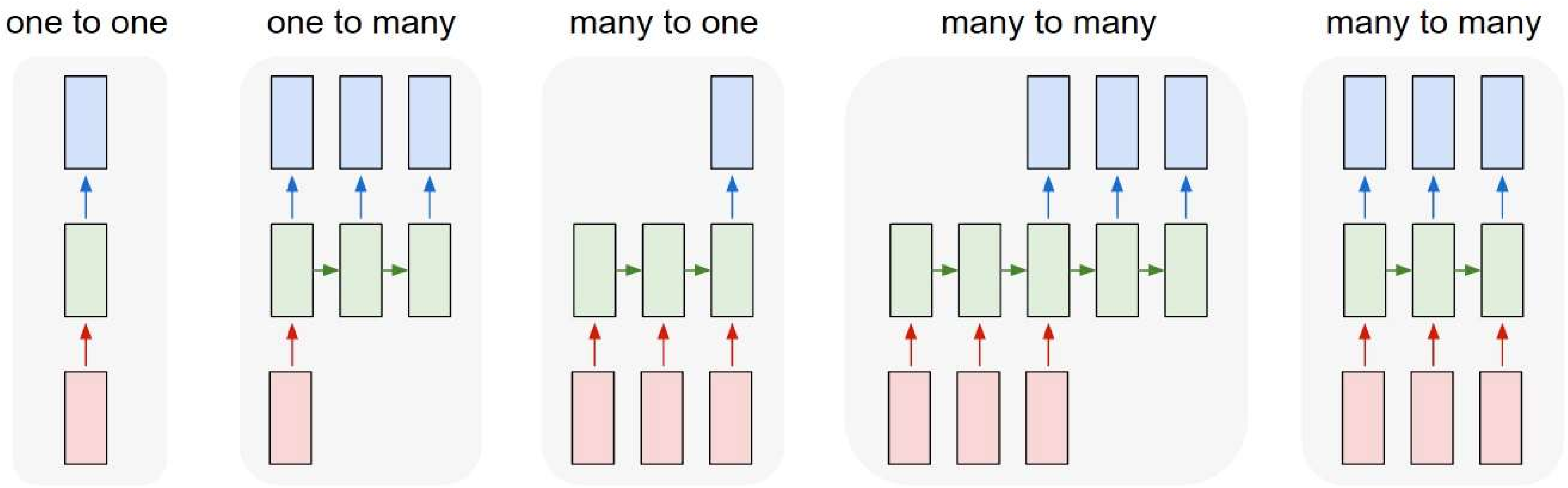
Output: No sequence

Example: sentence classification, multiple-choice question answering

Image Credit: Andrej Karpathy

# Sequences in Input or Output?

- It’s a spectrum…



Input: No sequence

Output: No sequence

Example: “standard” classification / regression problems

Input: No sequence

Output: Sequence

Example: Im2Caption

Input: Sequence

Output: No sequence

Example: sentence classification, multiple-choice question answering

Input: Sequence

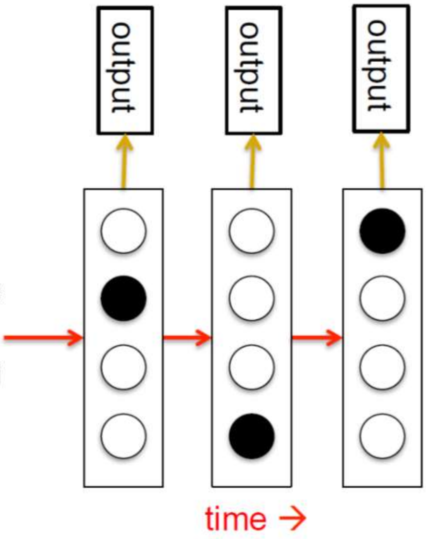
Output: Sequence

Example: machine translation, video classification, video captioning, open-ended question answering

Image Credit: Andrej Karpathy

# (Non-Deep) Ways to deal with sequence labelling

- **Autoregressive models**
  - Predict the next term in a sequence from a fixed number of previous terms using delay taps.
  - 1st-order Autoregressive model, AR(1): $y_t = w_0 + w_1 y_{t-1} + \epsilon_t$
  - 2nd-order Autoregressive model, AR(2): $y_t = w_0 + w_1 y_{t-1} + w_2 y_{t-2} + \epsilon_t$
  - And so on.
- **Hidden Markov Model, HMM**
  - HMMs have a discrete one-of-N hidden state. Transitions between states are stochastic and controlled by a transition probability matrix. Also, the outputs produced by a state are also stochastic, and are controlled by emission probabilities.
    - We can not be sure which state produced a given output. So, the state is "hidden"
    - It is easy to represent a probability distribution across the N states with N probabilities.
  - To predict the next output we need to infer the probability distribution over hidden states. HMMs have efficient algorithm for inference and learning.



# What’s wrong with MLPs?

- Problem 1: Can’t model sequences
  - Fixed-sized Inputs & Outputs
  - No temporal structure

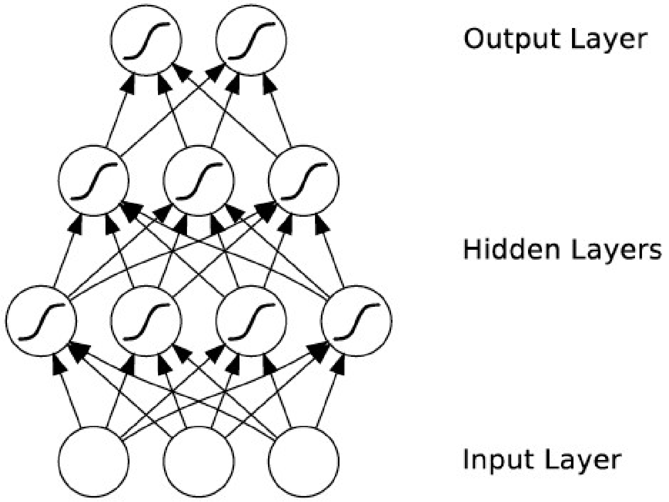


Image Credit: Alex Graves, book

# What's wrong with MLPs?

- Problem 1: Can't model sequences
  - Fixed-sized Inputs & Outputs
  - No temporal structure

- Problem 2: Pure feed-forward processing
  - No "memory", no feedback

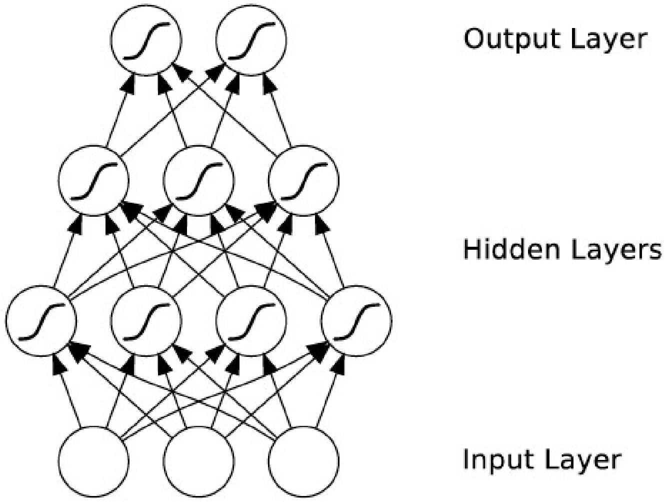


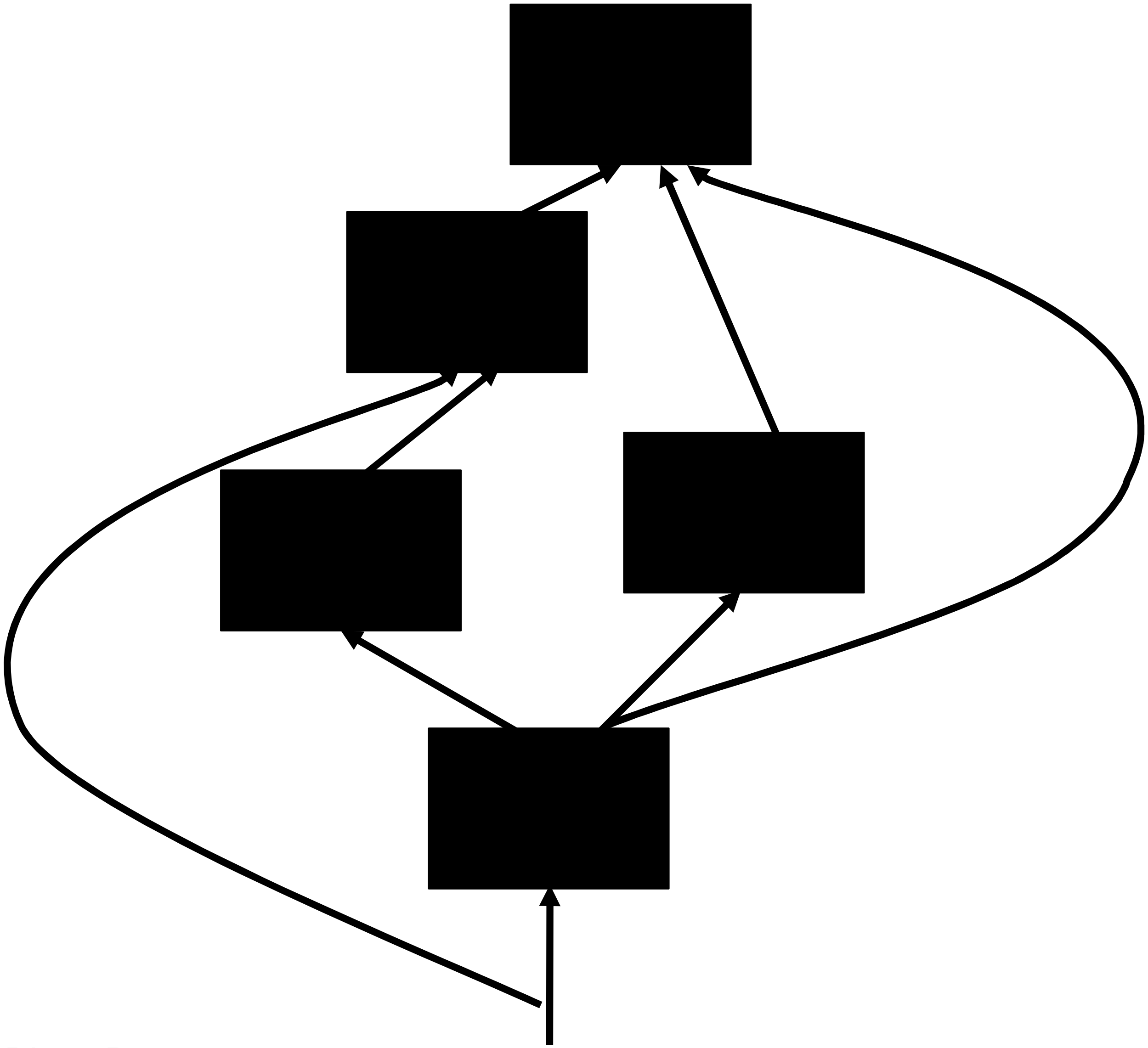
Image Credit: Alex Graves, book

# A Naïve Way to Do This

# 2 Key Ideas

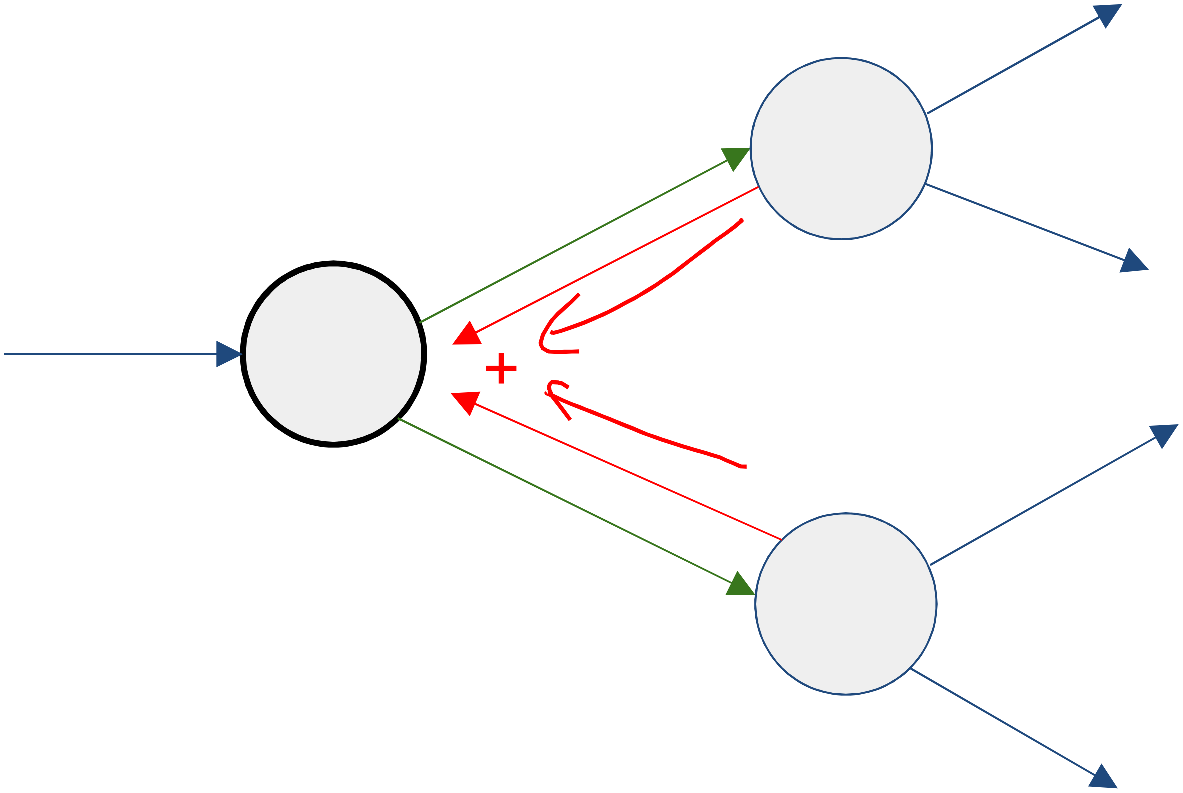
- Parameter Sharing
  - in computation graphs = adding gradients

# Computational Graph



Slide Credit: Marc'Aurelio Ranzato

# Gradients add at branches



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# 2 Key Ideas

- Parameter Sharing
  - in computation graphs = adding gradients

- “Unrolling”
  - in computation graphs with parameter sharing

# How do we model sequences?

- No input

$$\boldsymbol{s}_t = f_\theta(\boldsymbol{s}_{t-1})$$

Image Credit: Bengio, Goodfellow, Courville

# How do we model sequences?

- With inputs

$$s_t = f_\theta(s_{t-1}, x_t)$$

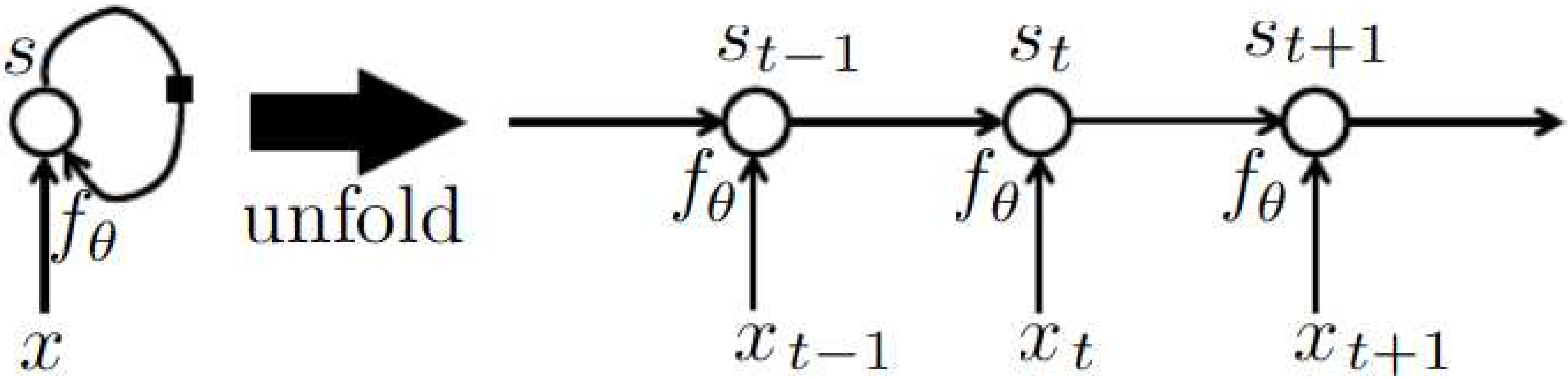


Image Credit: Bengio, Goodfellow, Courville

# 2 Key Ideas

- Parameter Sharing
  - in computation graphs = adding gradients

- “Unrolling”
  - in computation graphs with parameter sharing

- Parameter sharing + Unrolling
  - Allows modeling arbitrary sequence lengths!
  - Keeps numbers of parameters in check

# New Words

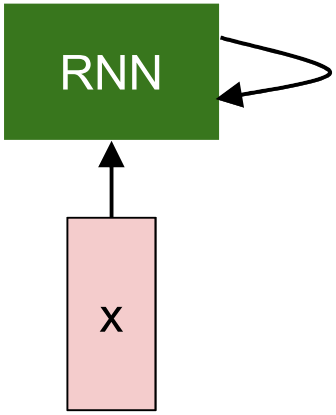
- Recurrent Neural Networks (RNNs)

- Recursive Neural Networks
  - General family; think graphs instead of chains

- Types:
  - "Vanilla" RNNs (Elman Networks)
  - Long Short Term Memory (LSTMs)
  - Gated Recurrent Units (GRUs)
  - …

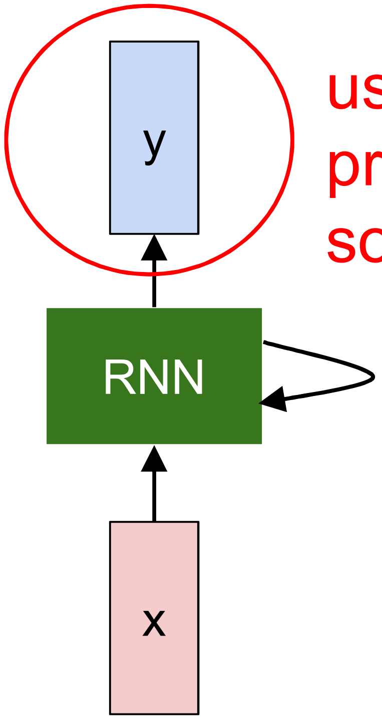
- Algorithms
  - BackProp Through Time (BPTT)
  - BackProp Through Structure (BPTS)

# Recurrent Neural Network



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Recurrent Neural Network



usually want to predict a vector at some time steps

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Recurrent Neural Network

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

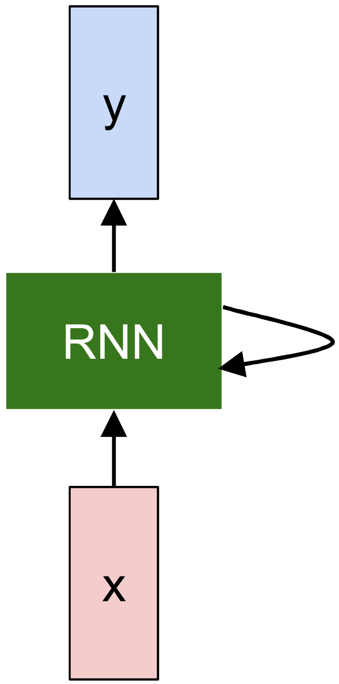
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W

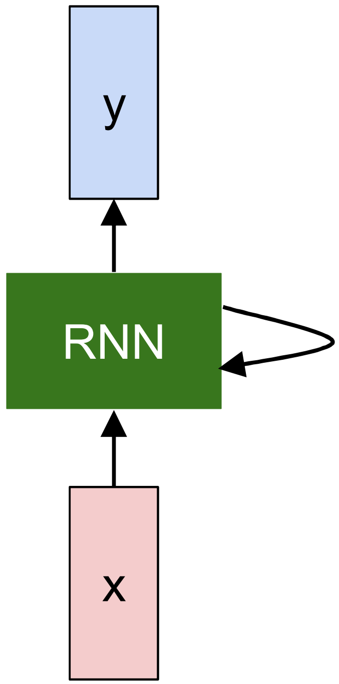


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Recurrent Neural Network

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

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

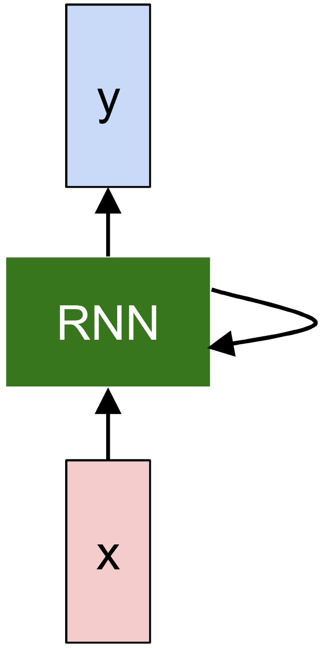


Notice: the same function and the same set of parameters are used at every time step.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# (Vanilla) Recurrent Neural Network

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



$$y_t = W_{hy}h_t + b_y$$

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

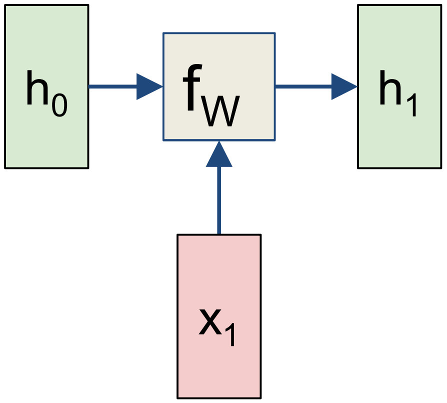
$$\downarrow$$

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

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

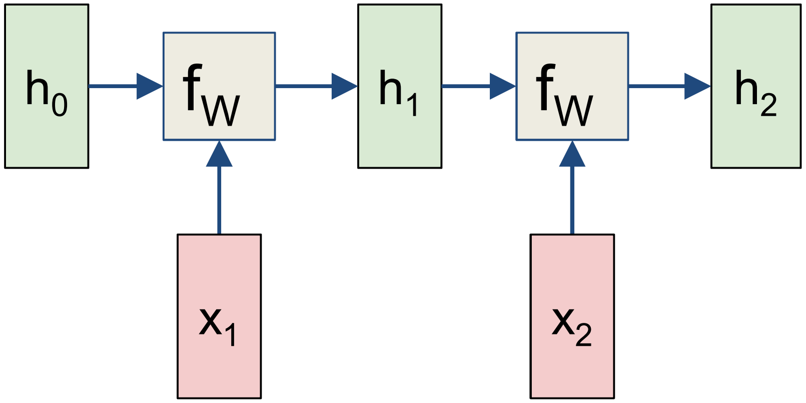
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# RNN: Computational Graph



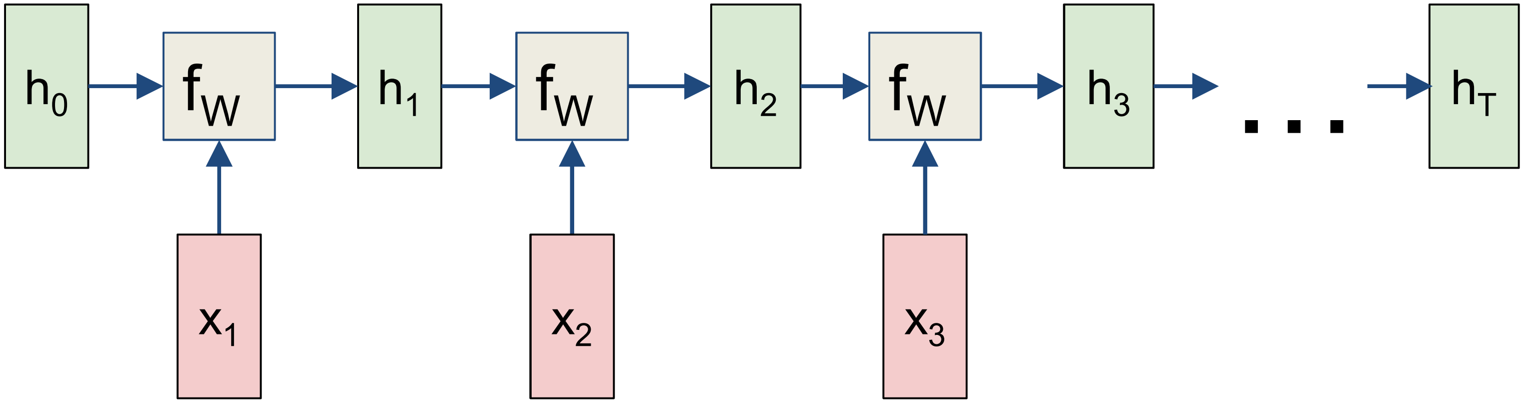
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# RNN: Computational Graph



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

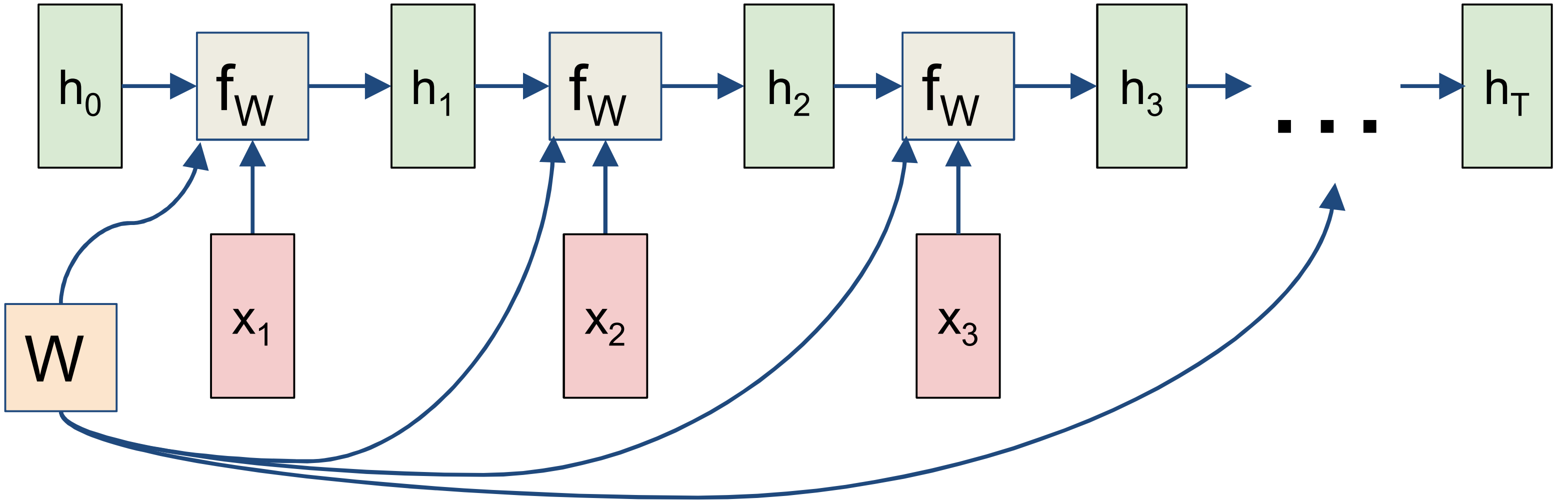
# RNN: Computational Graph



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

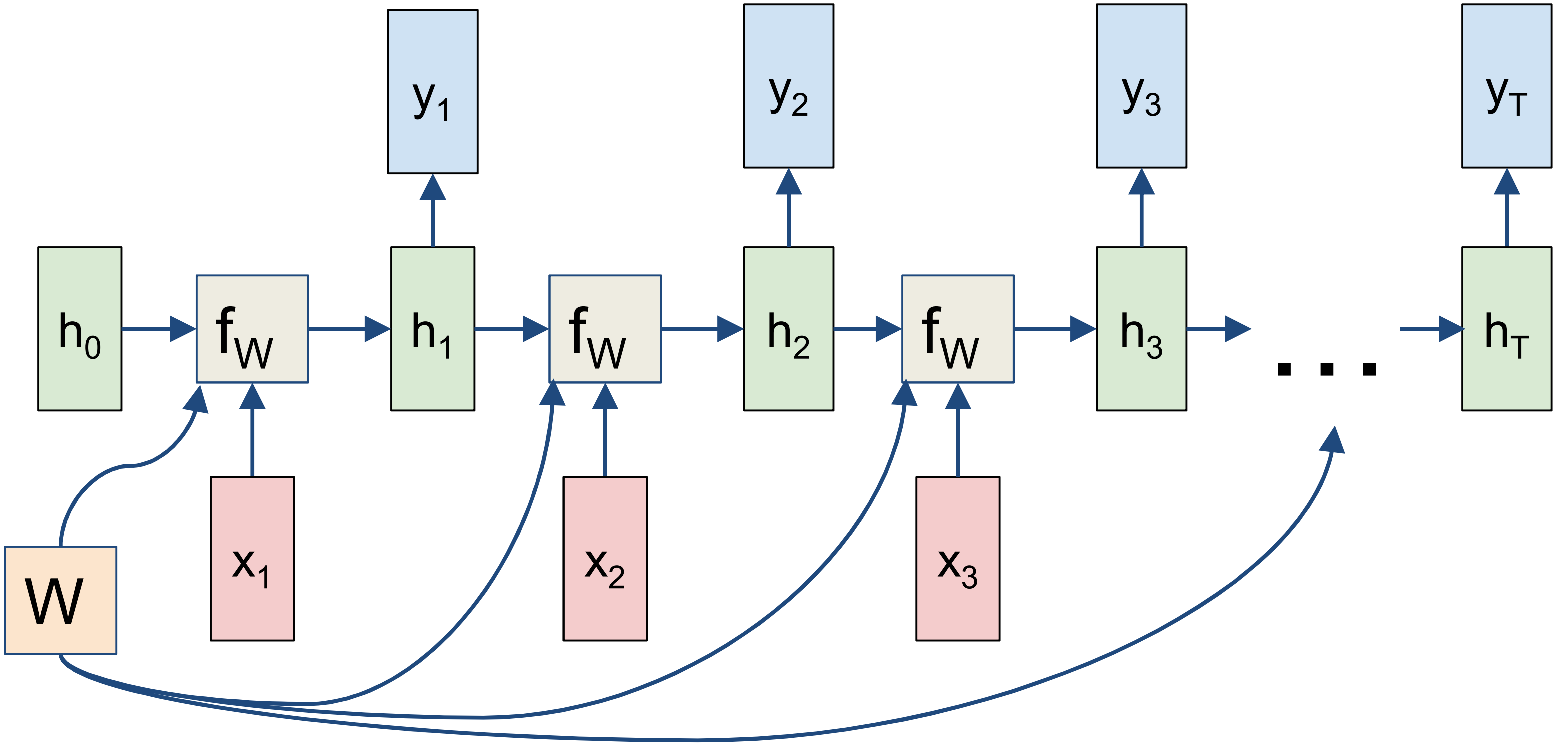
# RNN: Computational Graph

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



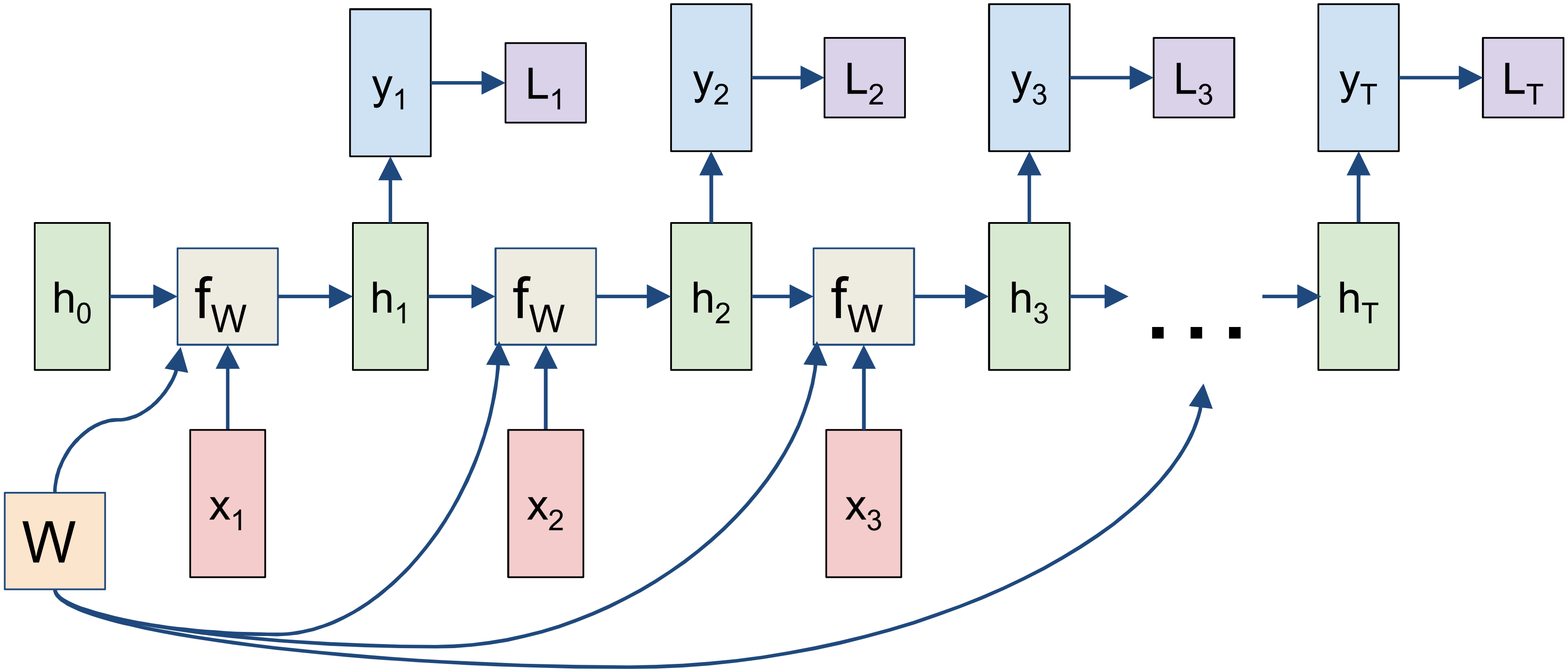
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# RNN: Computational Graph: Many to Many



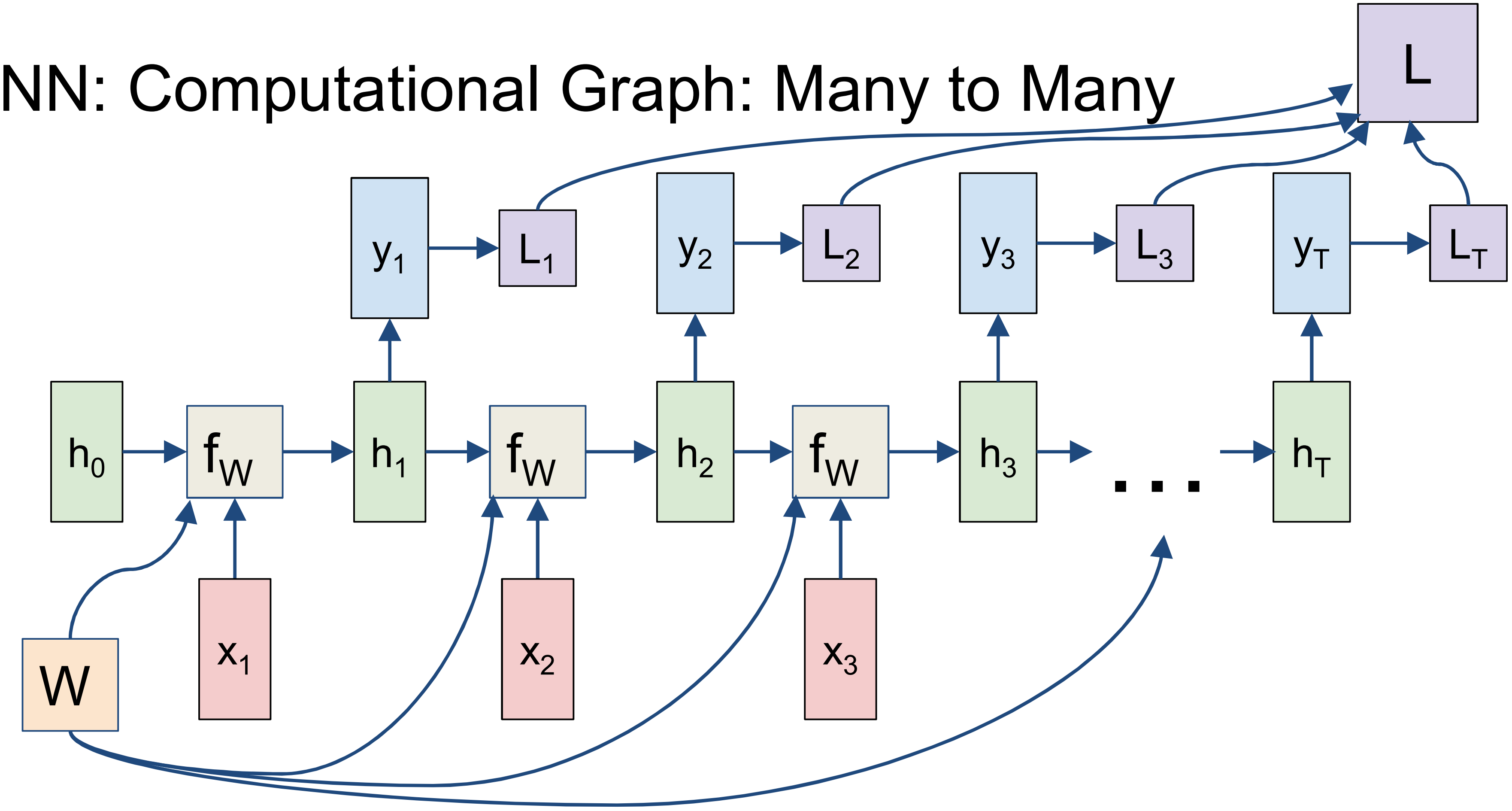
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# RNN: Computational Graph: Many to Many



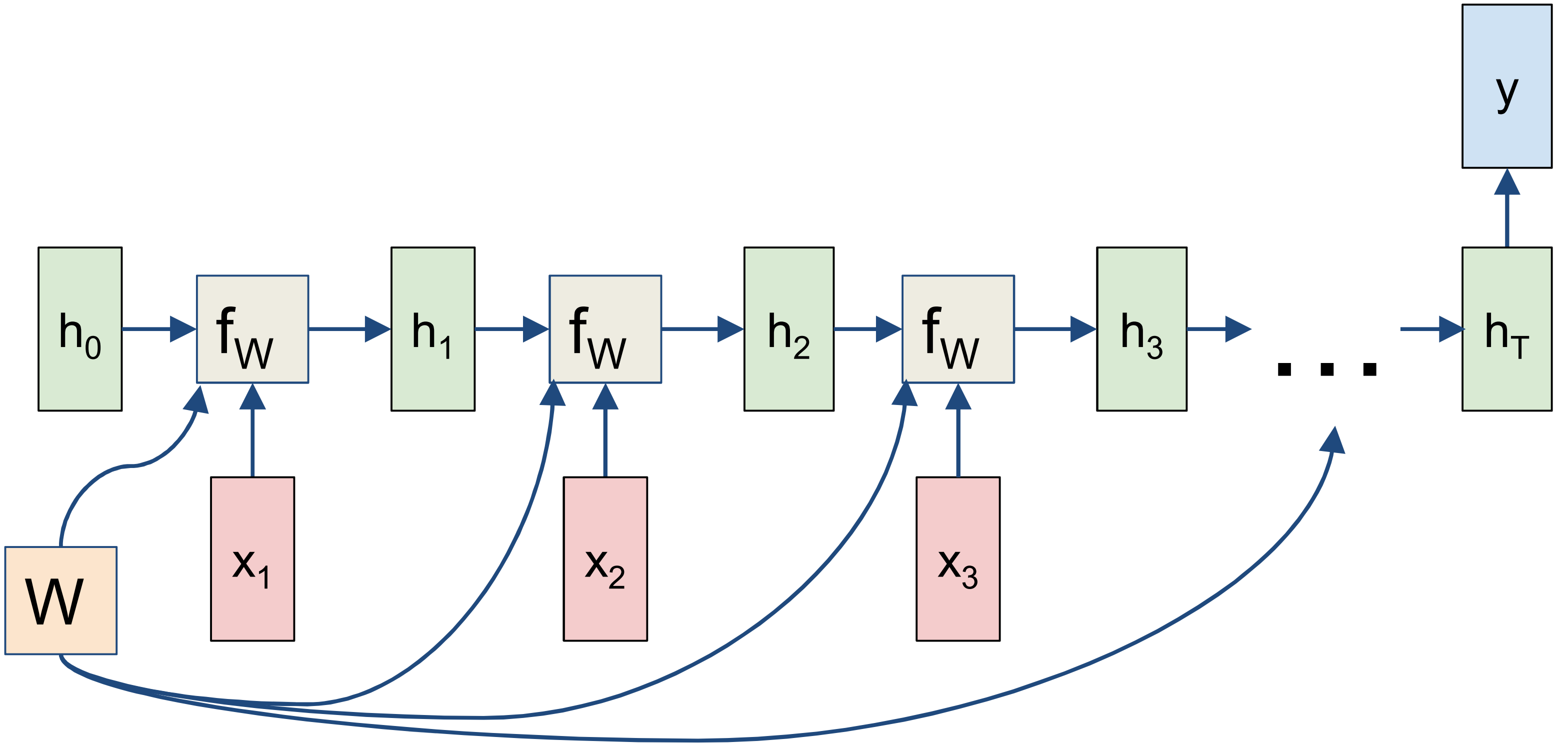
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# RNN: Computational Graph: Many to Many



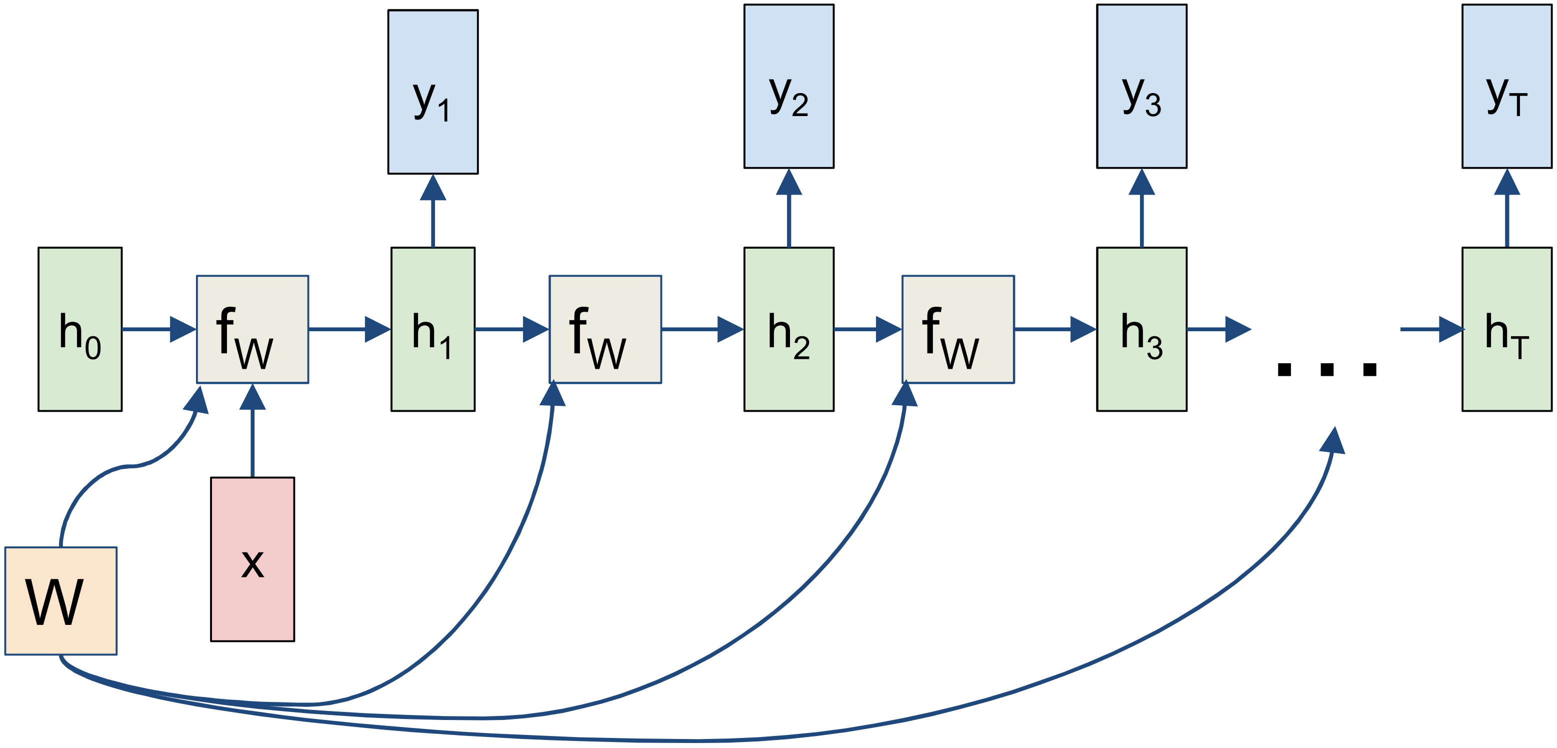
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# RNN: Computational Graph: Many to One



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

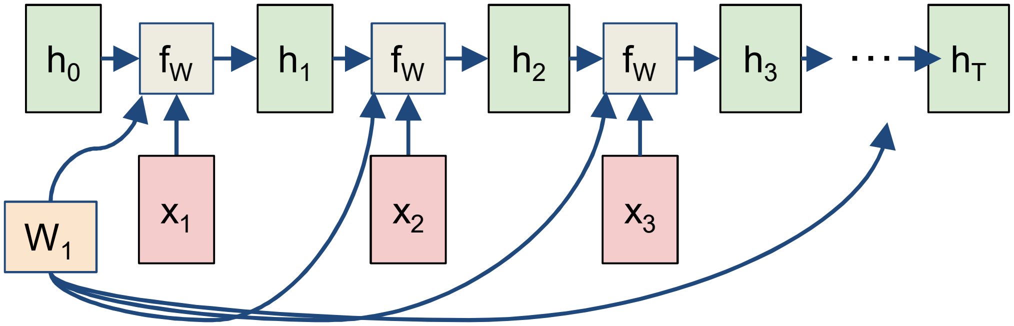
# RNN: Computational Graph: One to Many



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

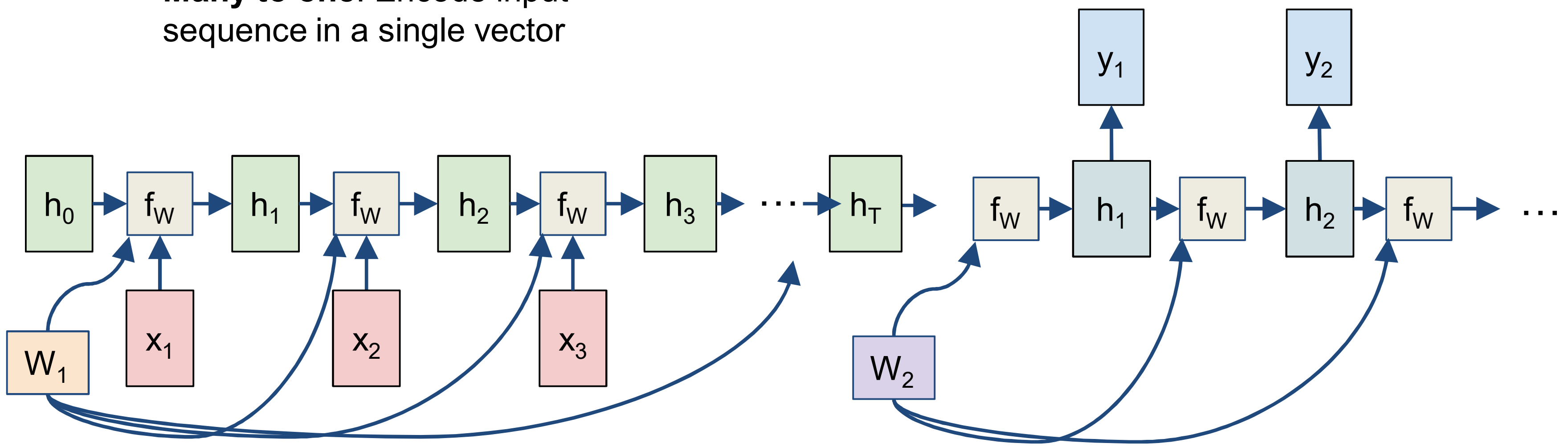
# Sequence to Sequence: Many-to-one + one-to-many

**Many to one**: Encode input sequence in a single vector



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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



**Many to one**: Encode input sequence in a single vector

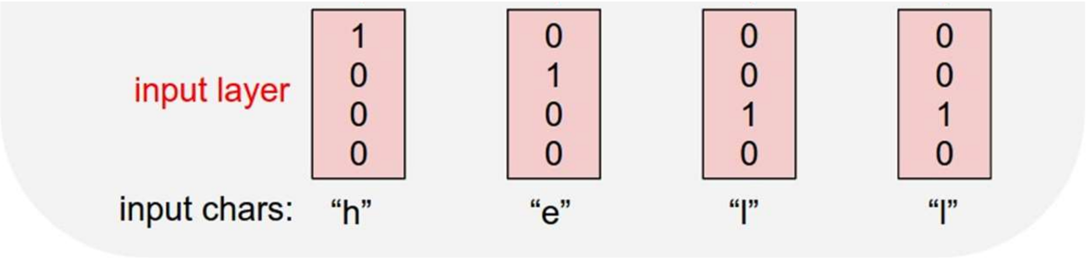
**One to many**: Produce output sequence from single input vector

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

## Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
**“hello”**



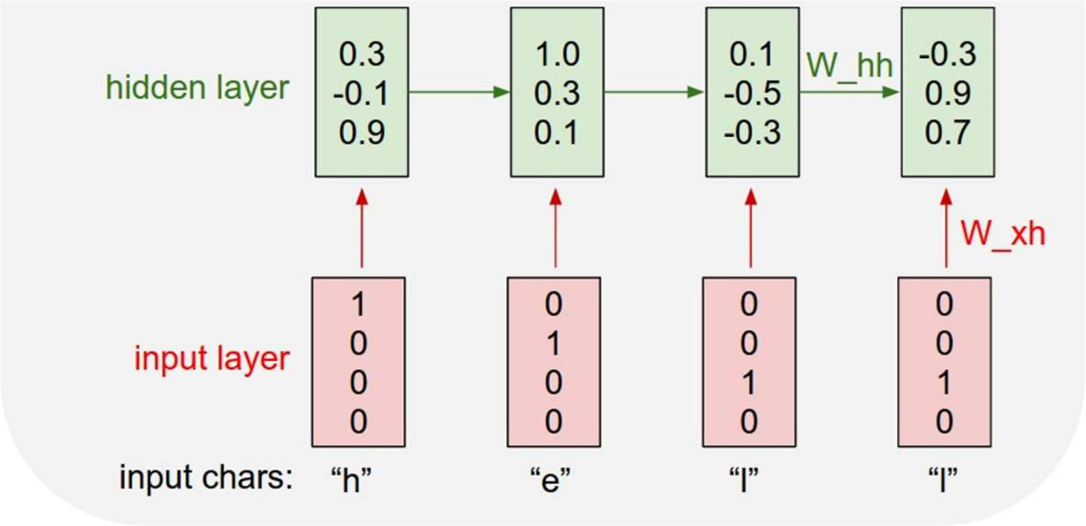
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

**Example: Character-level Language Model**

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

Vocabulary: [h,e,l,o]

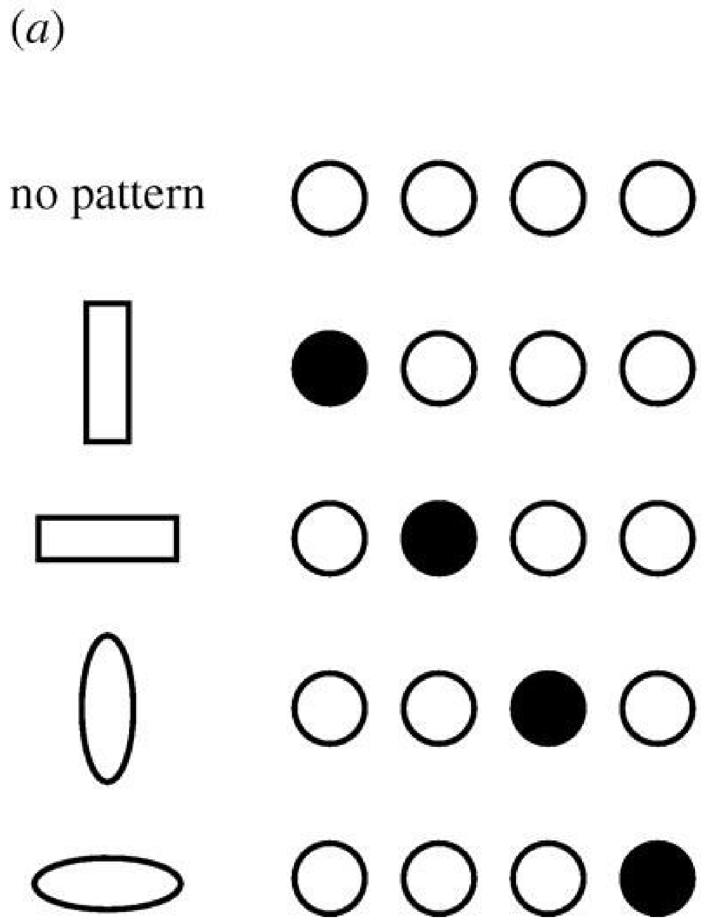
Example training sequence: **“hello”**



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Distributed Representations Toy Example

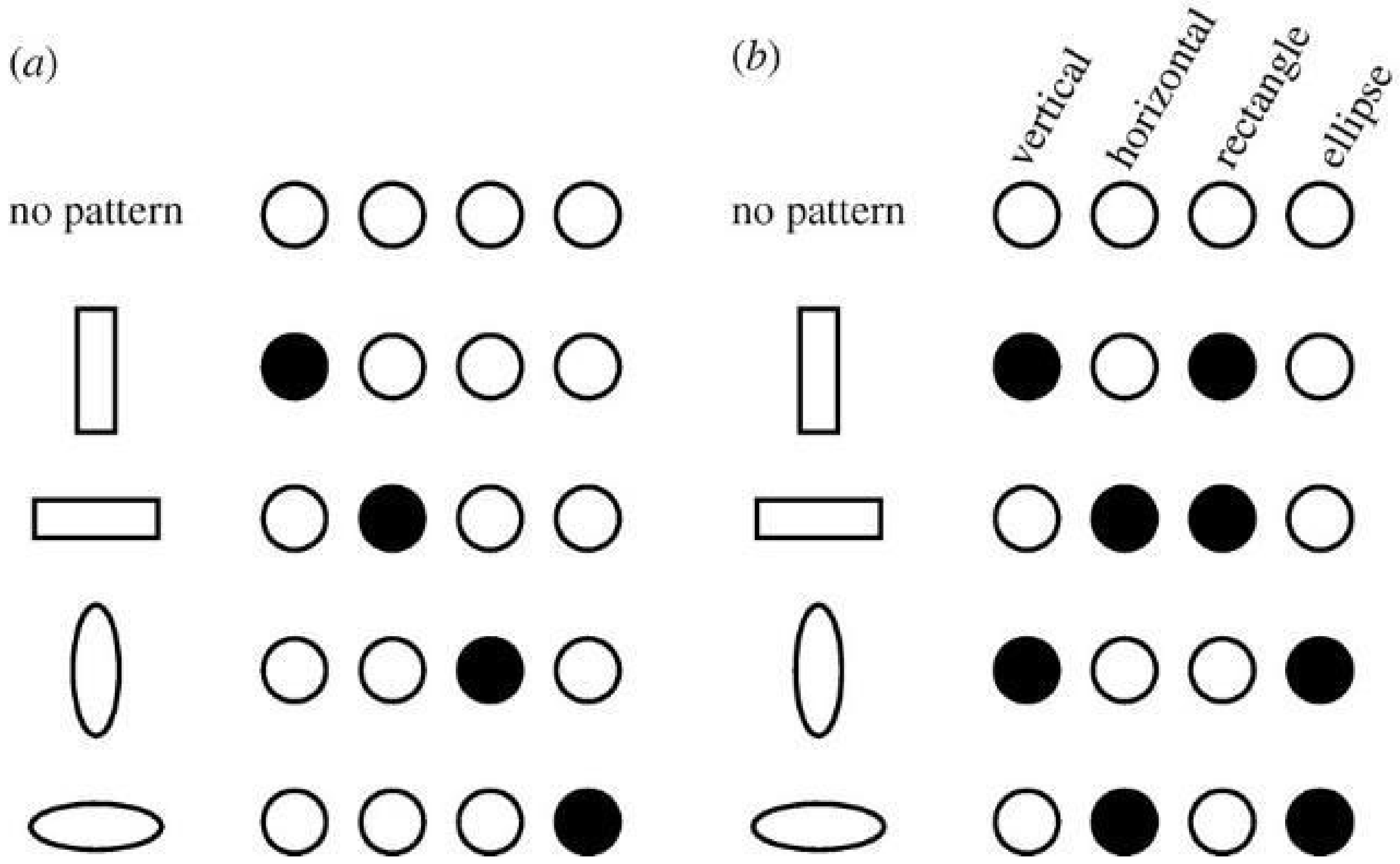
- Local vs Distributed



Slide Credit: Moontae Lee

# Distributed Representations Toy Example

- Can we interpret each dimension?



Slide Credit: Moontae Lee

# Power of distributed representations!

Local ● ● ○ ● = VR + HR + HE = ?

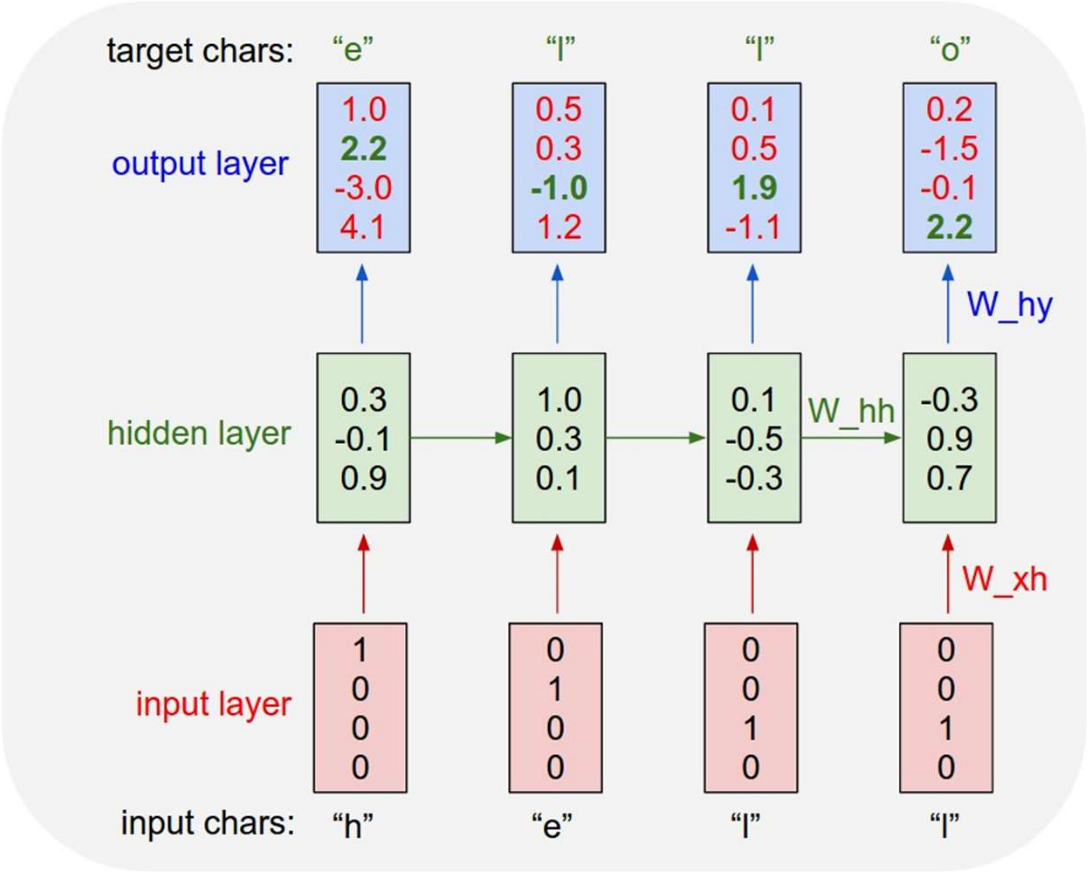
Distributed ● ● ○ ● = V + H + E ≈ ○

Slide Credit: Moontae Lee

**Example: Character-level Language Model**

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



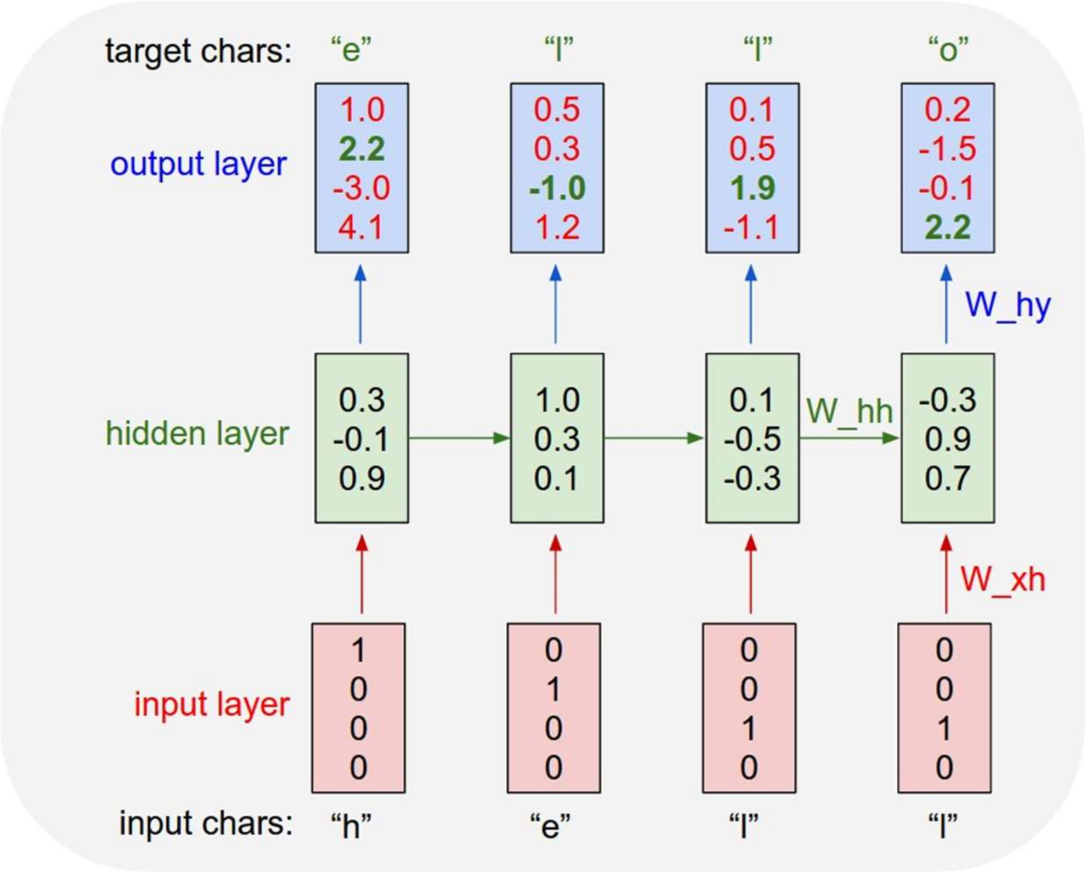
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Training Time: MLE / “Teacher Forcing”

**Example: Character-level Language Model**

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**



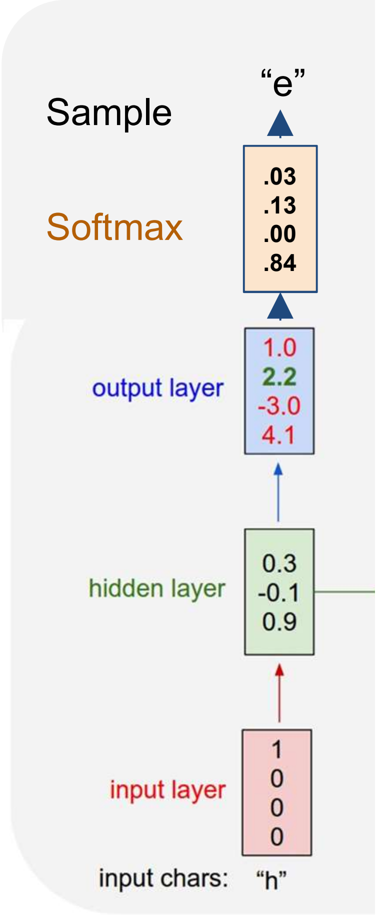
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Test Time: Sample / Argmax / Beam Search

**Example:
Character-level
Language Model
Sampling**

Vocabulary:
[h,e,l,o]

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



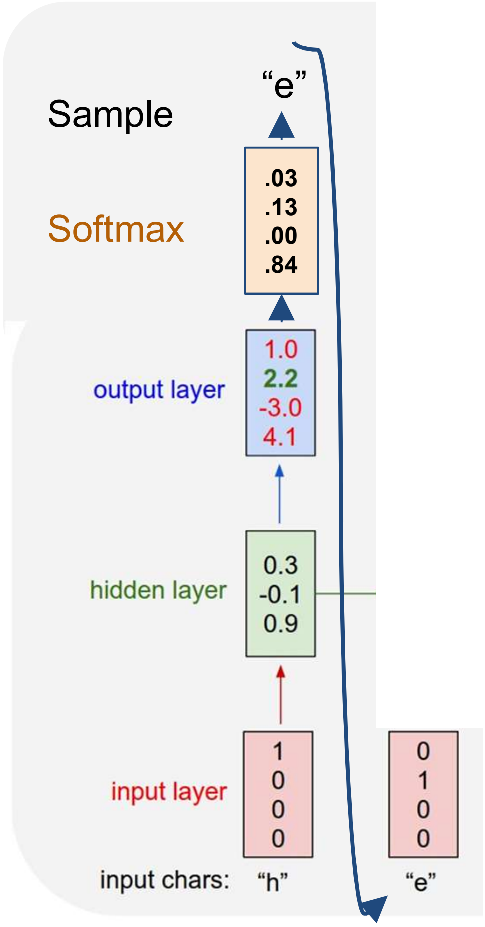
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Test Time: Sample / Argmax / Beam Search

**Example: Character-level Language Model Sampling**

Vocabulary: [h,e,l,o]

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



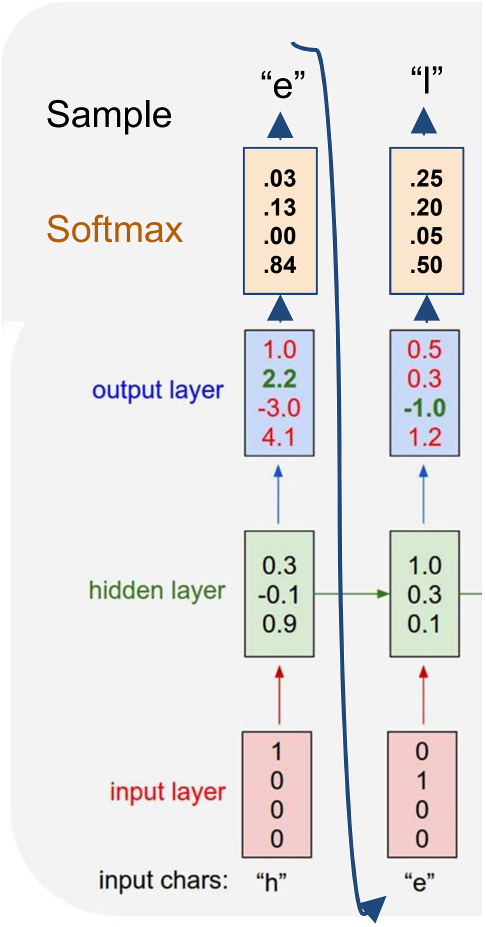
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Test Time: Sample / Argmax / Beam Search

**Example: Character-level Language Model Sampling**

Vocabulary: [h,e,l,o]

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



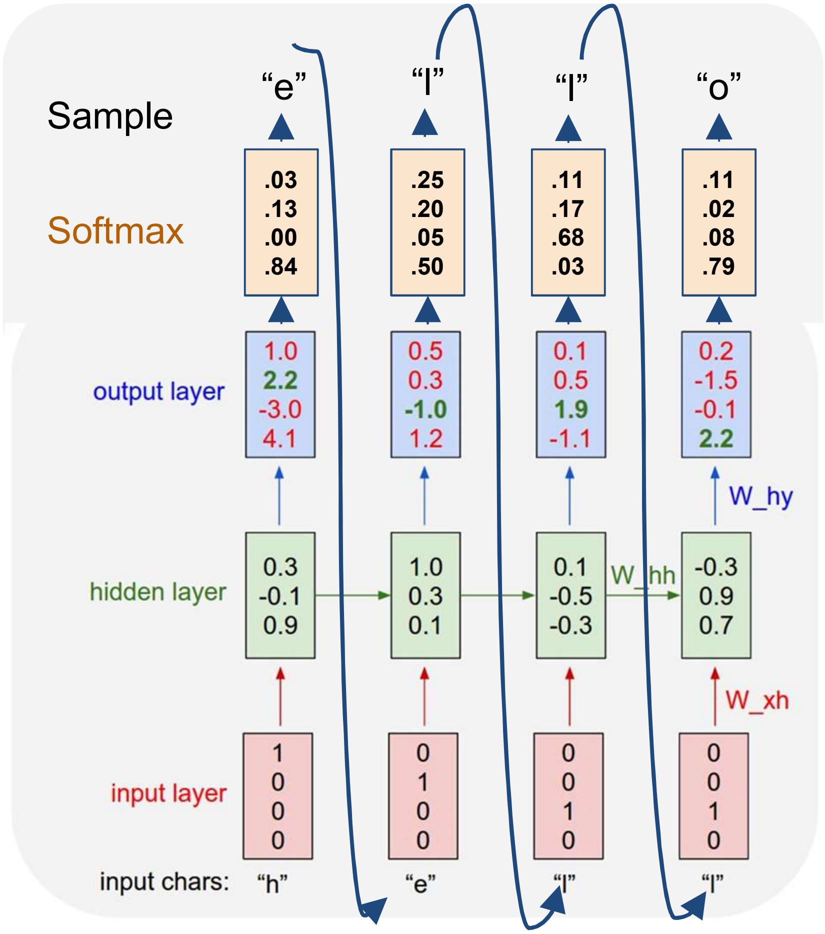
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Test Time: Sample / Argmax / Beam Search

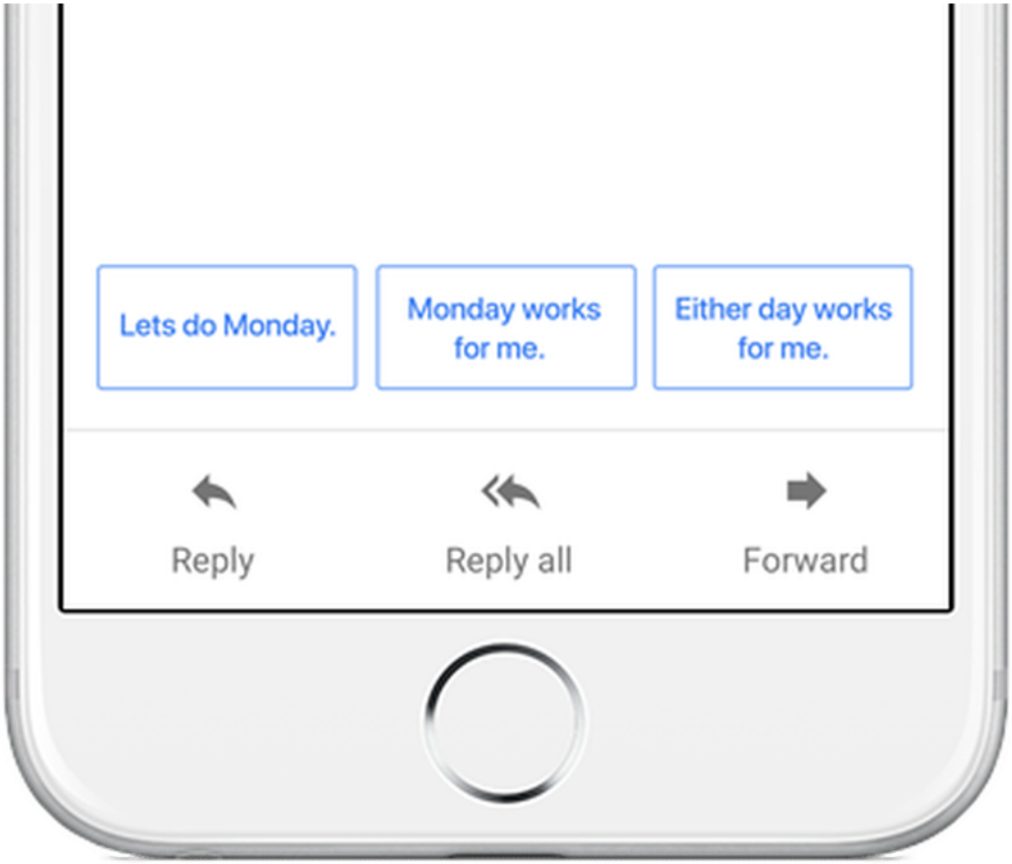
**Example: Character-level Language Model Sampling**

Vocabulary:
[h,e,l,o]

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



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



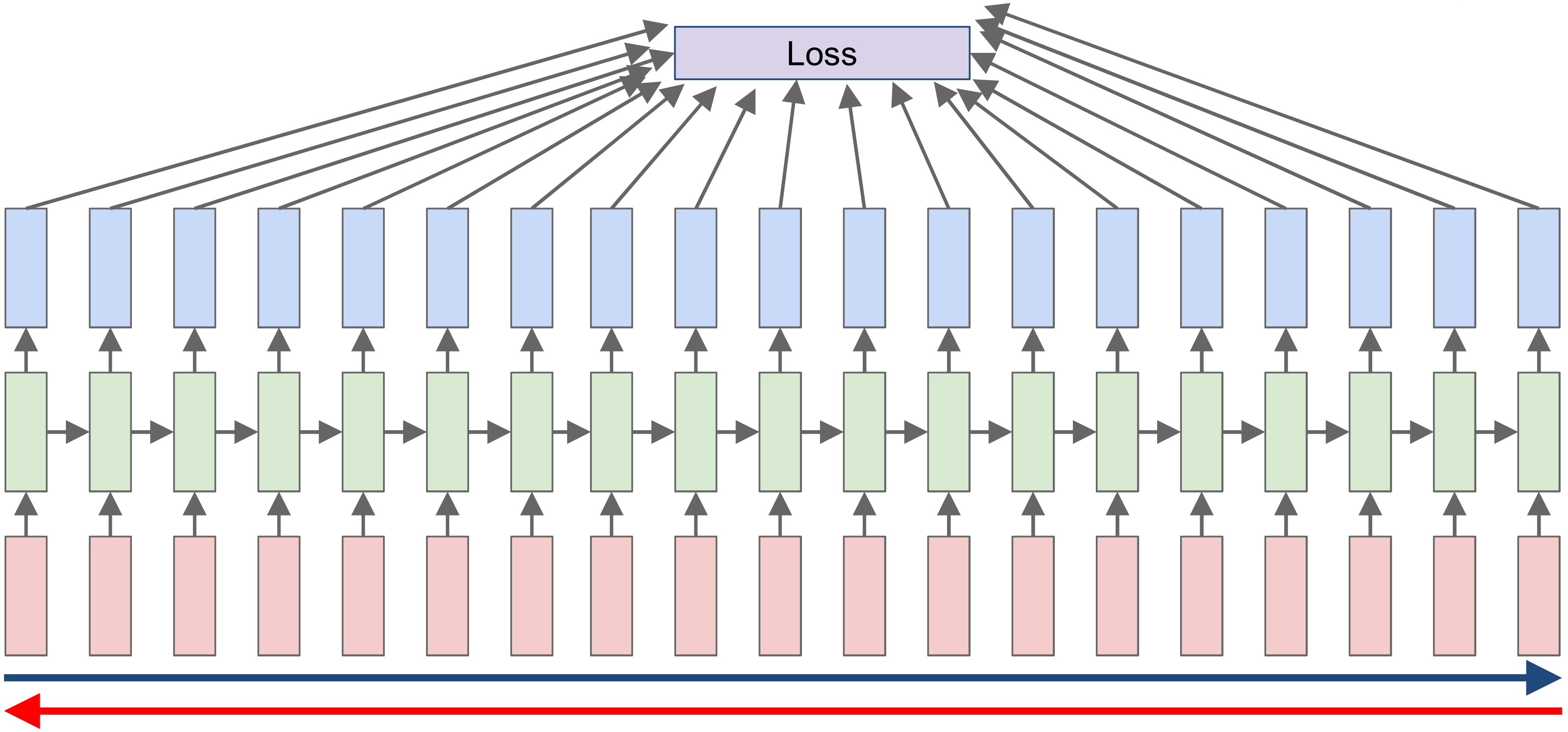
Lets do Monday.

Monday works for me.

Either day works for me.

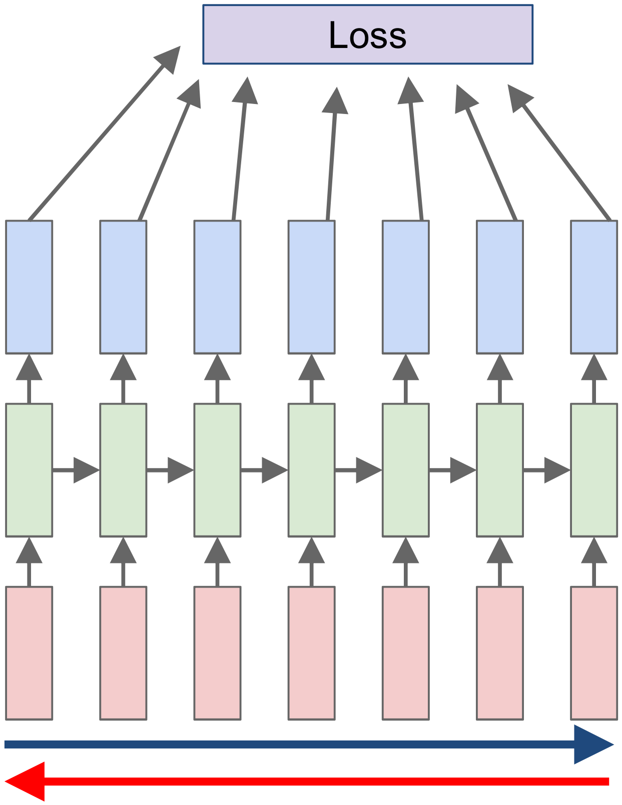
# Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

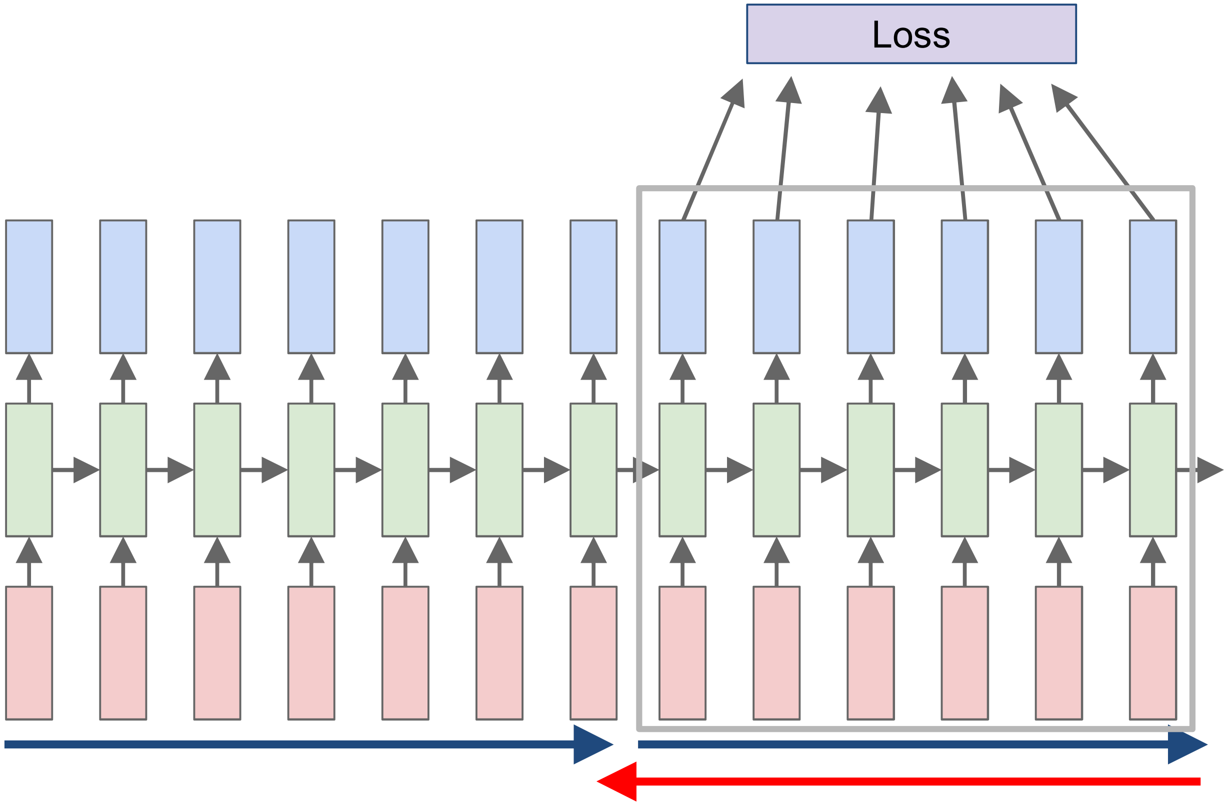
# **Truncated** Backpropagation through time



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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

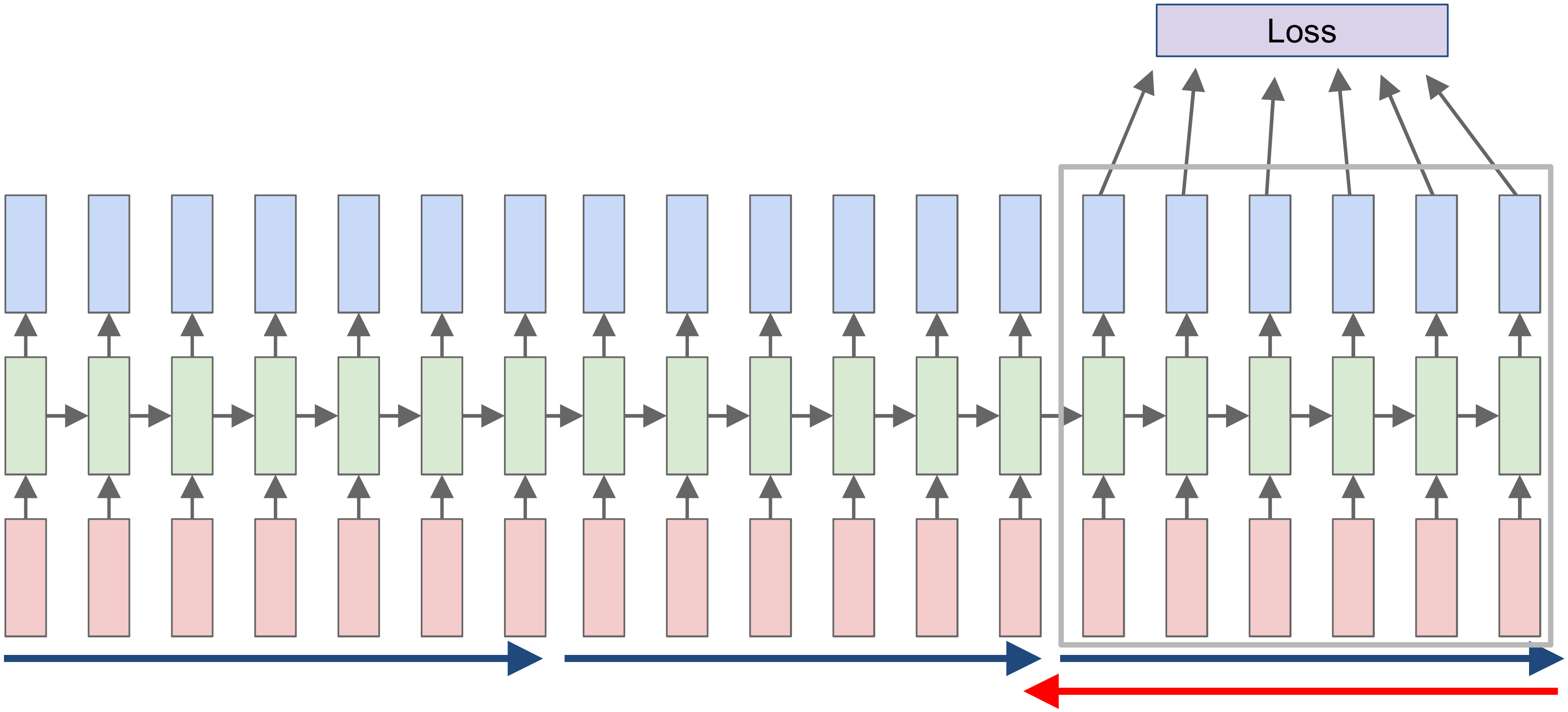
# **Truncated** Backpropagation through time



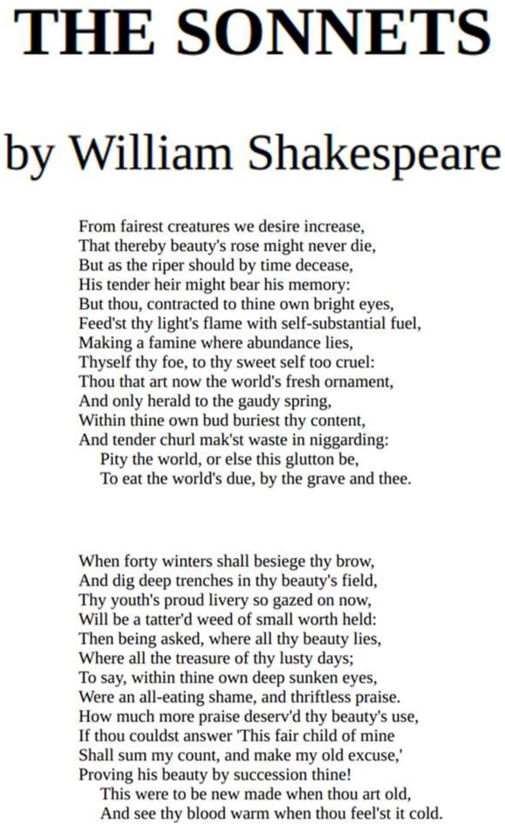
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# **Truncated** Backpropagation through time



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

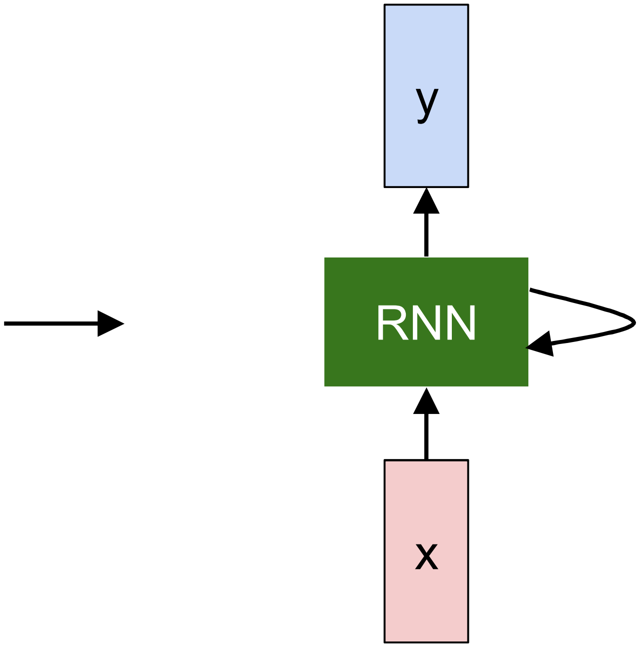


# THE SONNETS

## by William Shakespeare

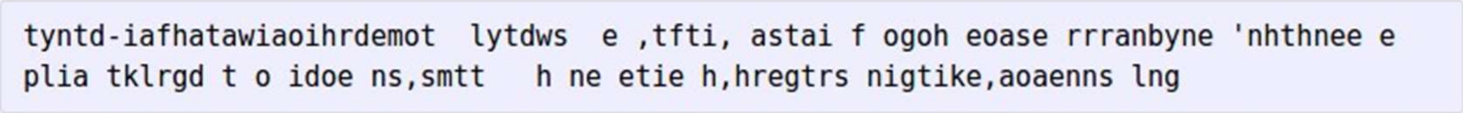
From fairest creatures we desire increase,
That thereby beauty's rose might never die,
But as the riper should by time decease,
His tender heir might bear his memory:
But thou, contracted to thine own bright eyes,
Feed'st thy light's flame with self-substantial fuel,
Making a famine where abundance lies,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

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



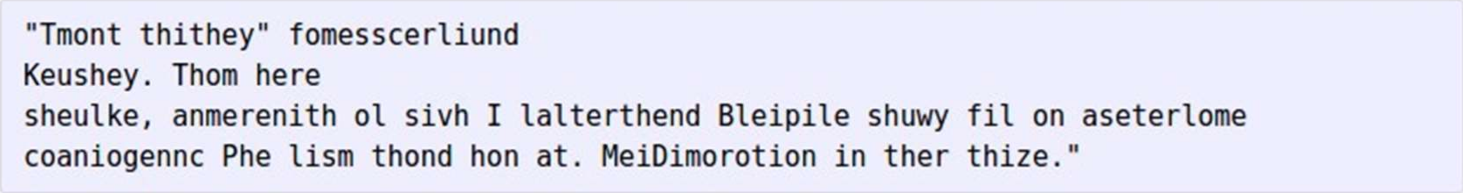
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

at first:



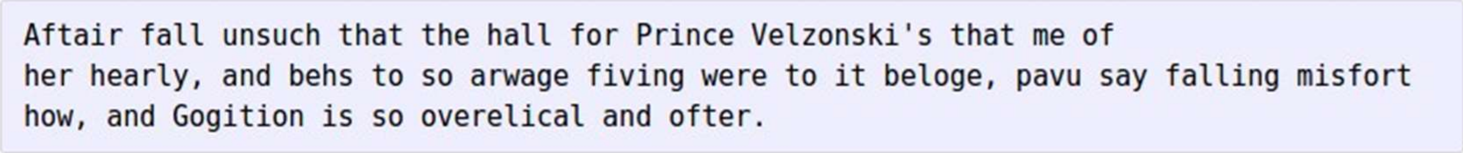
```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more



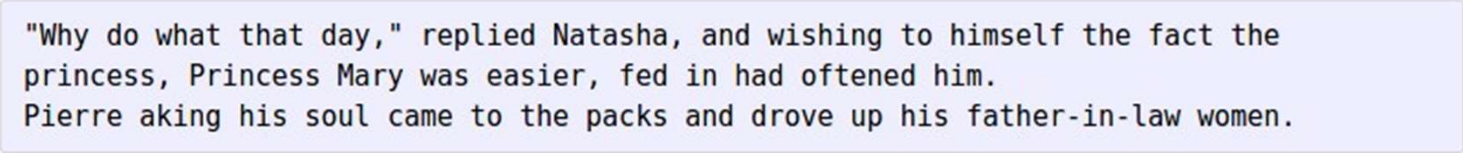
```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more



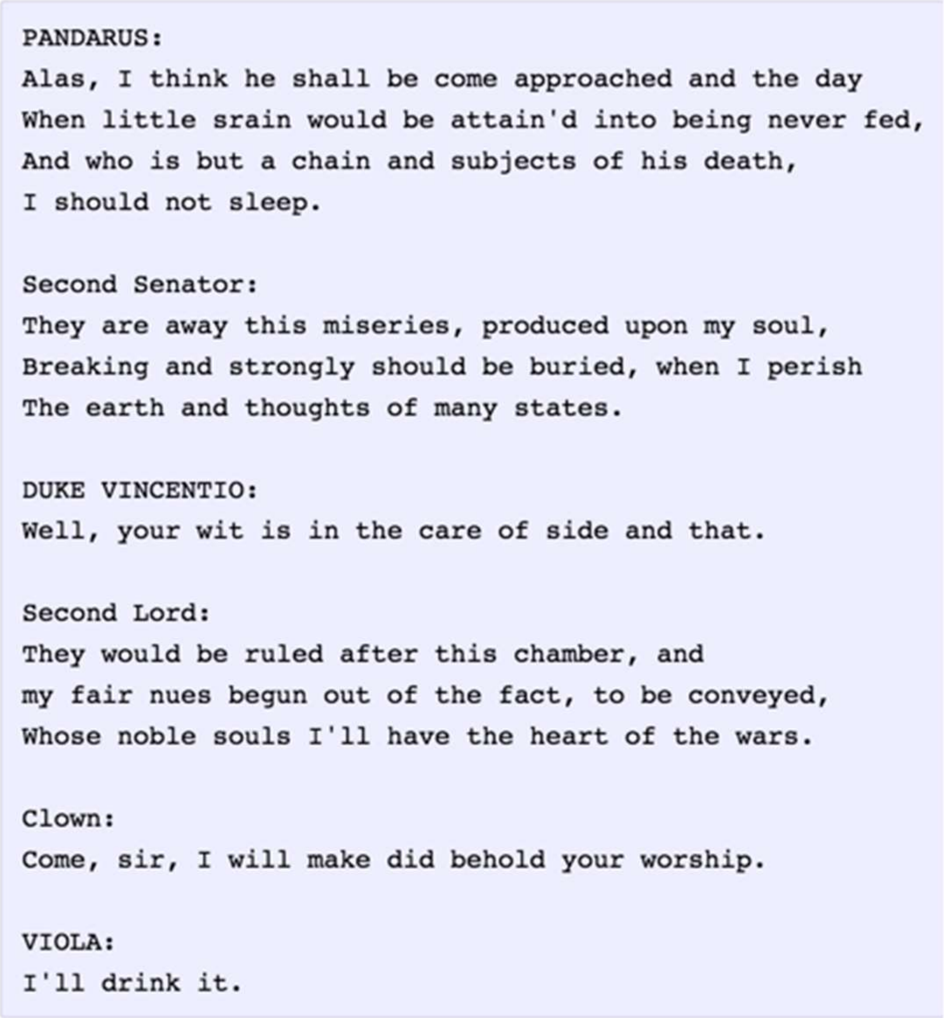
```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

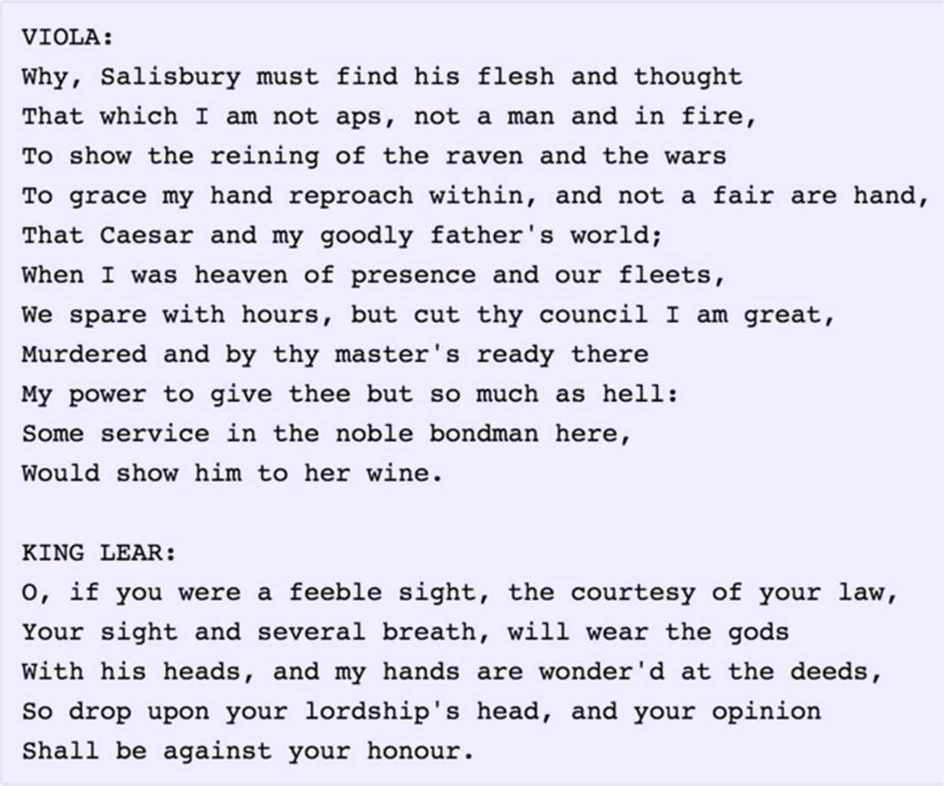


```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



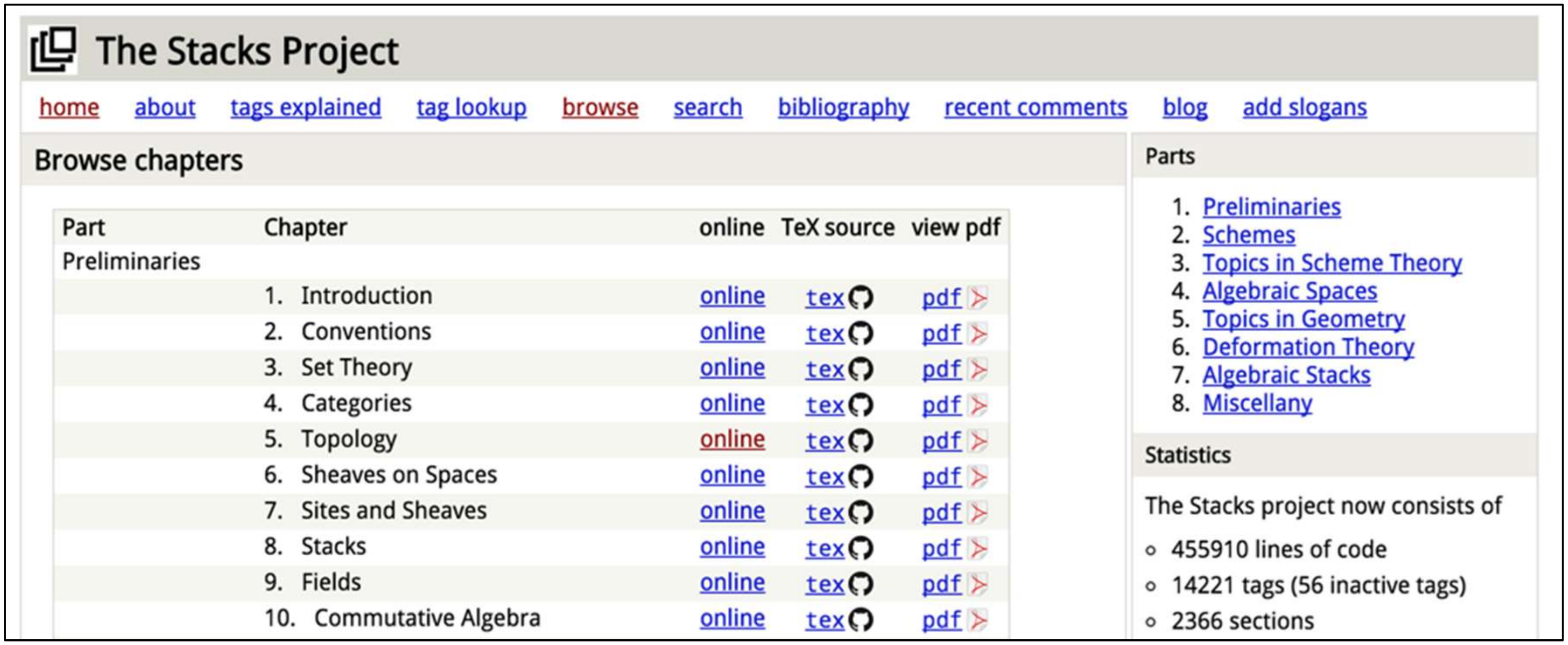
```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

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

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

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

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

VIOLA:
I'll drink it.
```



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

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# The Stacks Project: open source algebraic geometry textbook



## 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

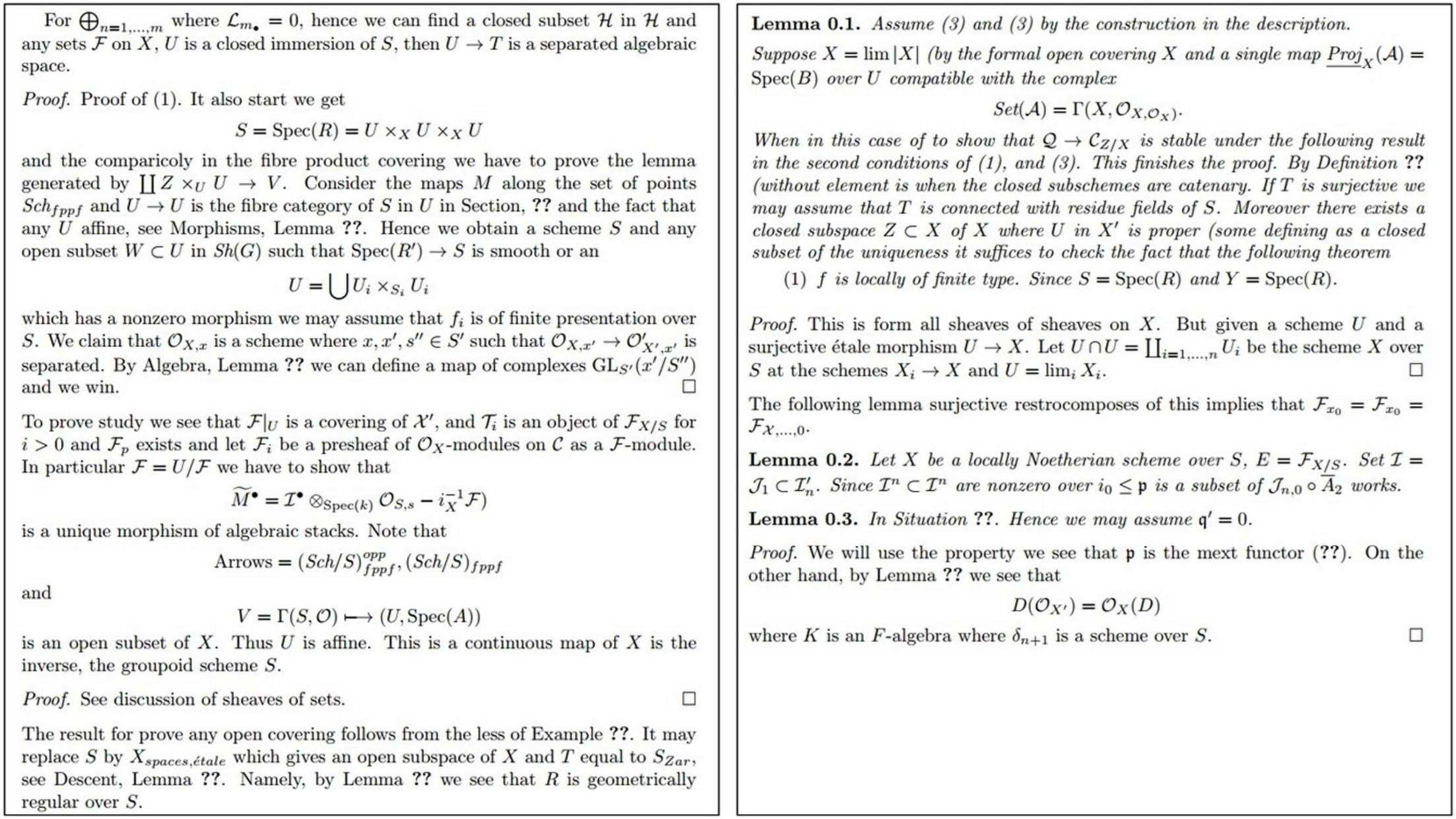
The Stacks project now consists of

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

Latex source

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



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.*

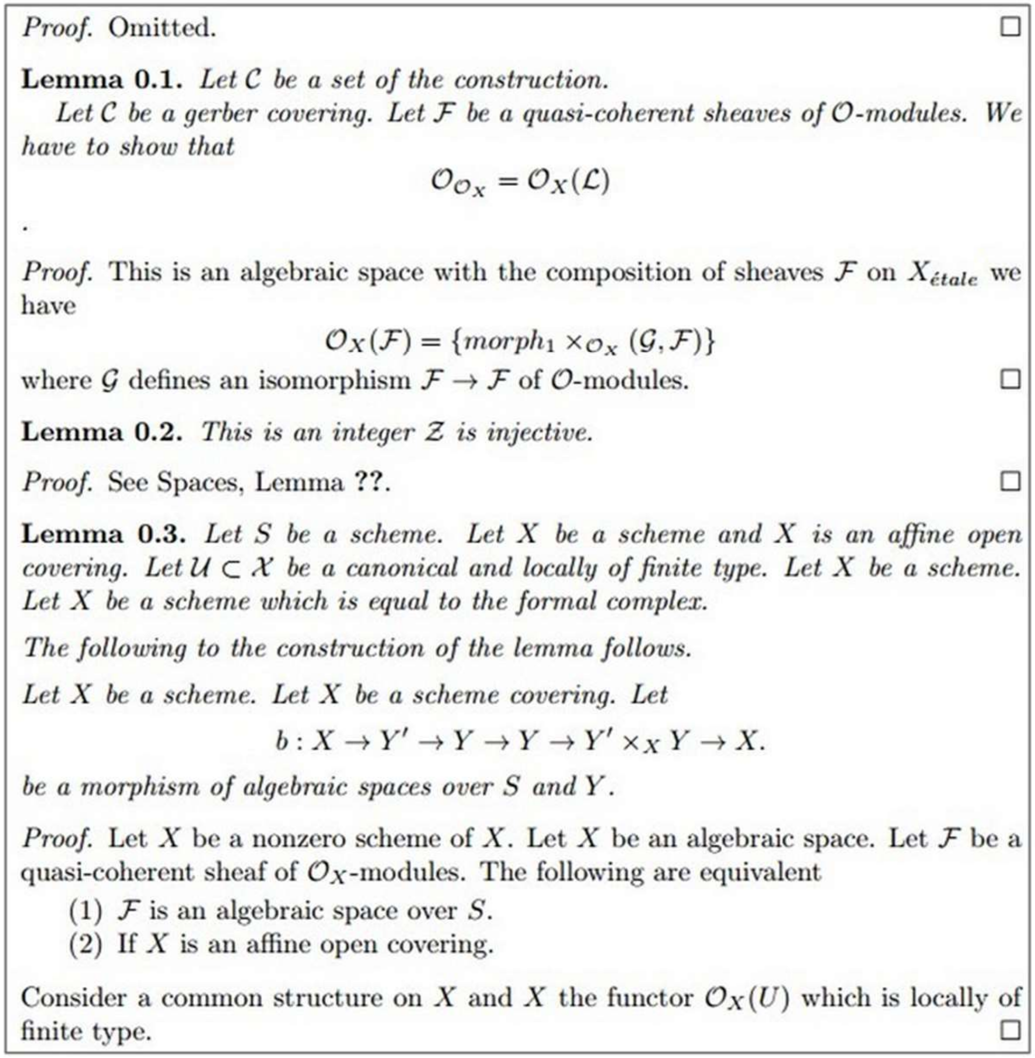
**Lemma 0.3.** *In Situation* **??**. *Hence we may assume $\mathfrak{q}' = 0$.*

*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 $\delta_{n+1}$ is a scheme over $S$. □

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



*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*

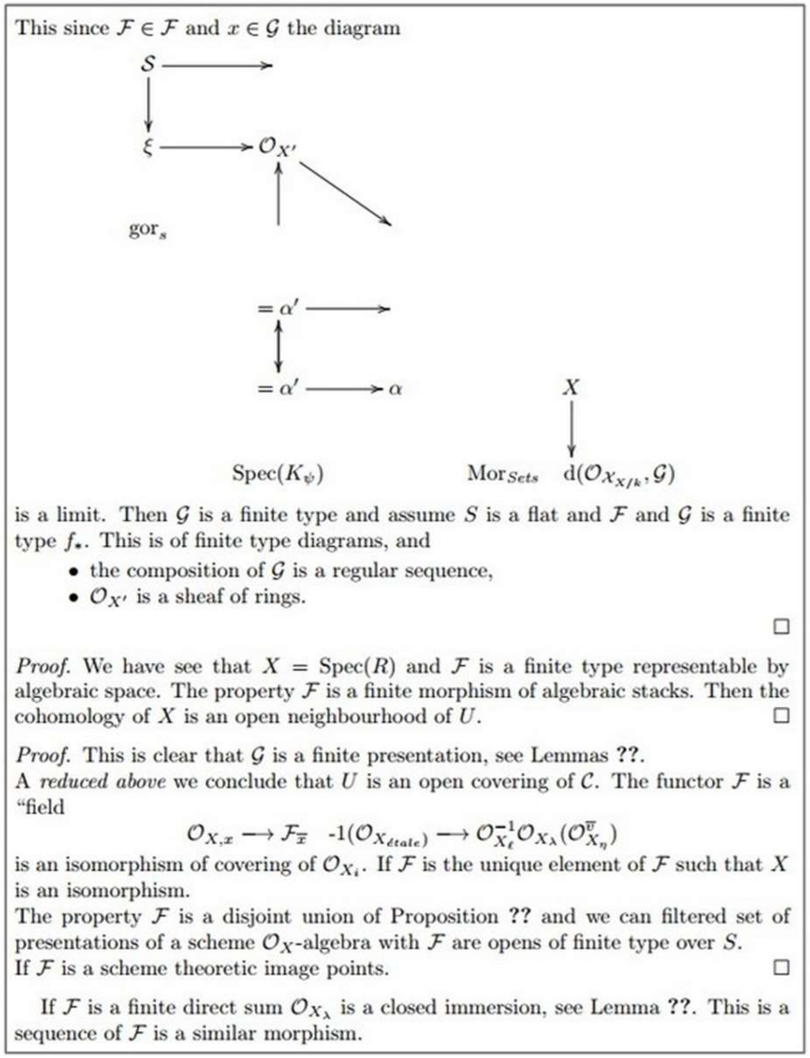
$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

*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

$$\begin{tikzcd}
\mathcal{S} \arrow[r] \arrow[d] & {} & & & \\
\xi \arrow[r] & \mathcal{O}_{X'} \arrow[dr] & & & \\
\mathrm{gor}_s & {} \arrow[u] & {} & & \\
& =\alpha' \arrow[r] \arrow[d, leftrightarrow] & {} & & \\
& =\alpha' \arrow[r] & \alpha & & X \arrow[d] \\
& \mathrm{Spec}(K_\psi) & & \mathrm{Mor}_{Sets} & \mathrm{d}(\mathcal{O}_{\mathcal{X}_{X/k}}, \mathcal{G})
\end{tikzcd}$$

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 $U$. □

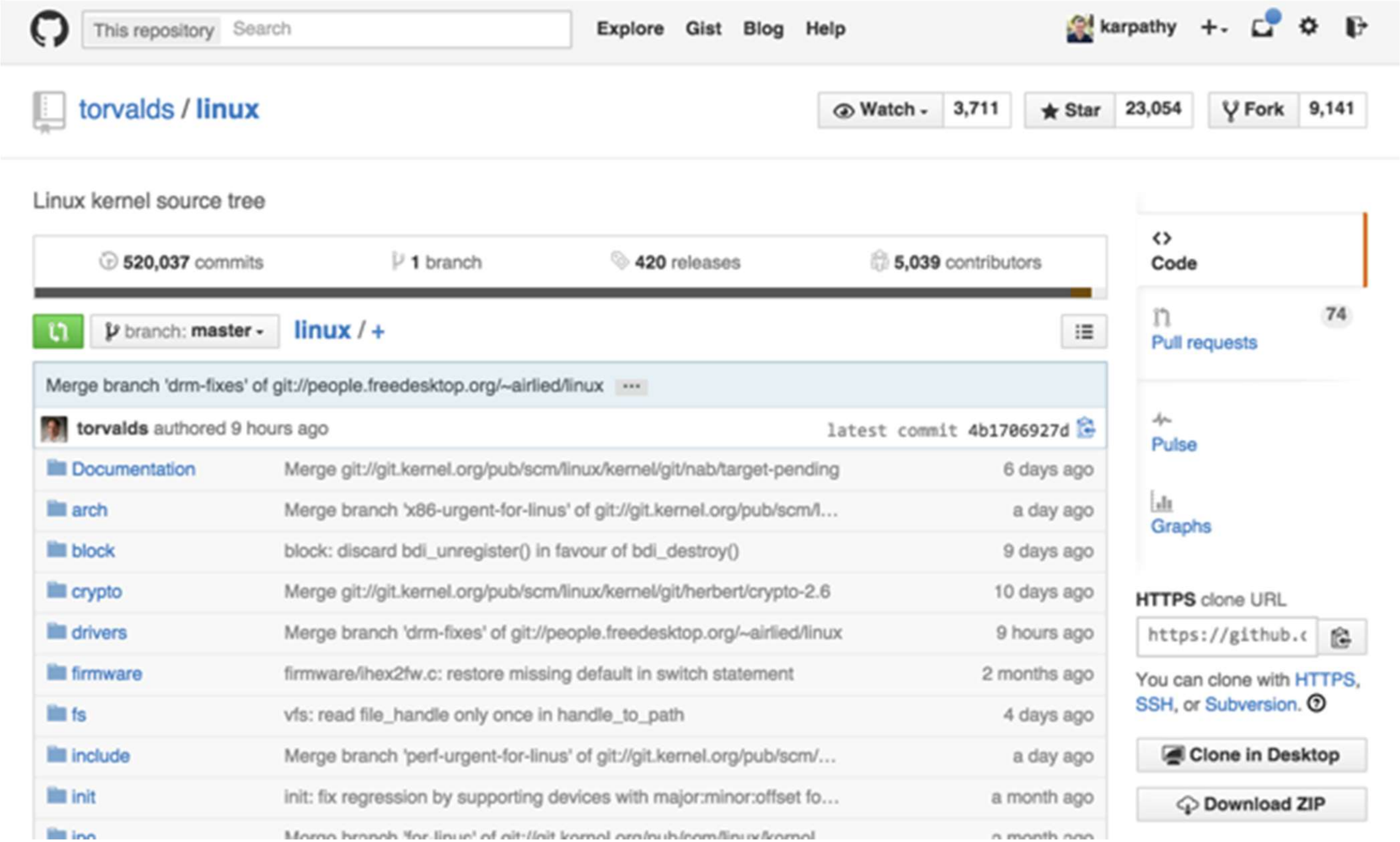
*Proof.* This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that $U$ is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

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

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: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



Linux kernel source tree

520,037 commits · 1 branch · 420 releases · 5,039 contributors

Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux ···

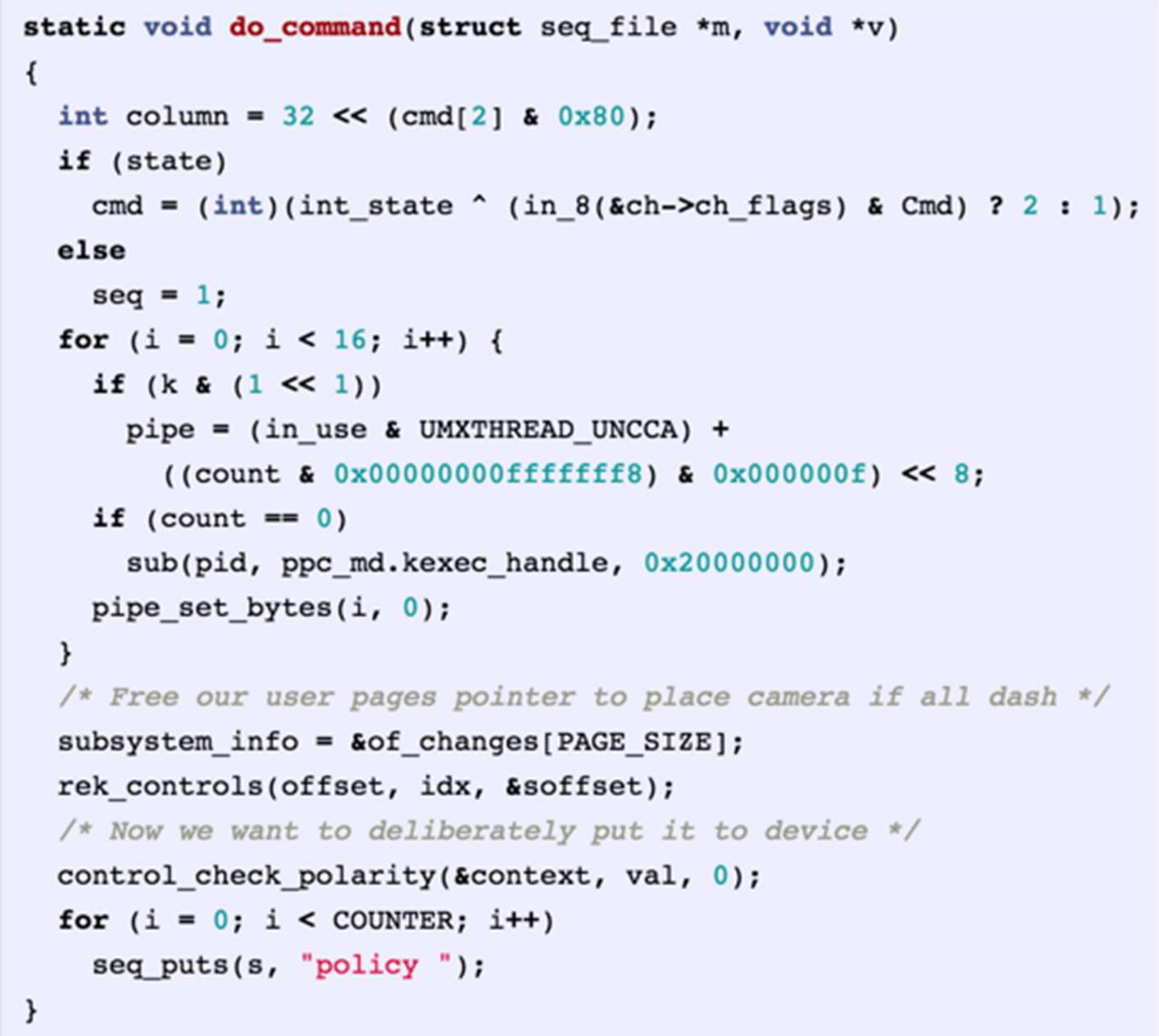
torvalds authored 9 hours ago · latest commit 4b1706927d

| | | |
|---|---|---|
| Documentation | Merge git://git.kernel.org/pub/scm/linux/kernel/git/nab/target-pending | 6 days ago |
| arch | Merge branch 'x86-urgent-for-linus' of git://git.kernel.org/pub/scm/l… | a day ago |
| block | block: discard bdi_unregister() in favour of bdi_destroy() | 9 days ago |
| crypto | Merge git://git.kernel.org/pub/scm/linux/kernel/git/herbert/crypto-2.6 | 10 days ago |
| drivers | Merge branch 'drm-fixes' of git://people.freedesktop.org/~airlied/linux | 9 hours ago |
| firmware | firmware/ihex2fw.c: restore missing default in switch statement | 2 months ago |
| fs | vfs: read file_handle only once in handle_to_path | 4 days ago |
| include | Merge branch 'perf-urgent-for-linus' of git://git.kernel.org/pub/scm/… | a day ago |
| init | init: fix regression by supporting devices with major:minor:offset fo… | a month ago |
| ipc | Merge branch 'for-linus' of git://git.kernel.org/pub/scm/linux/kernel | a month ago |

You can clone with HTTPS, SSH, or Subversion.

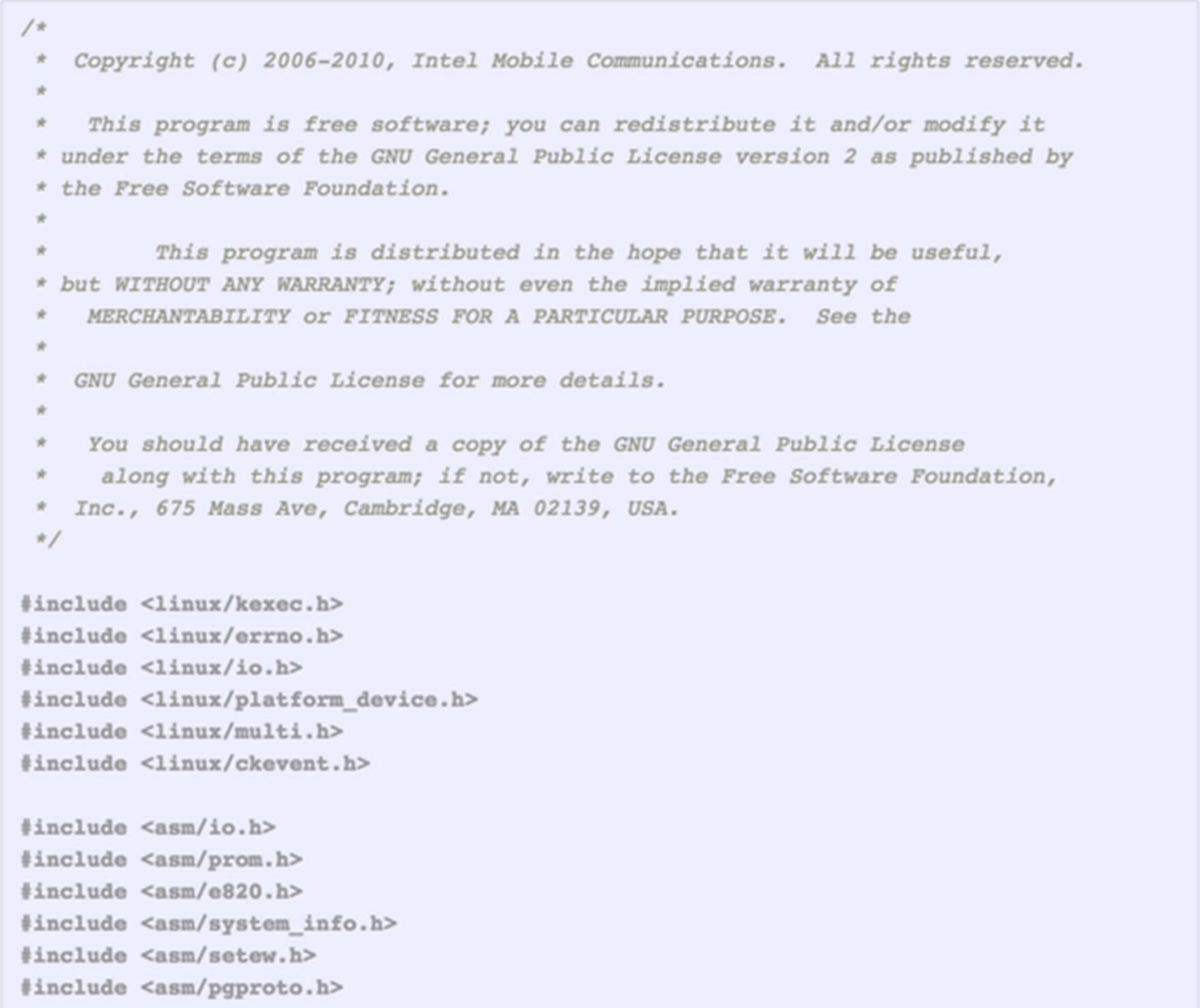
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Generated C code



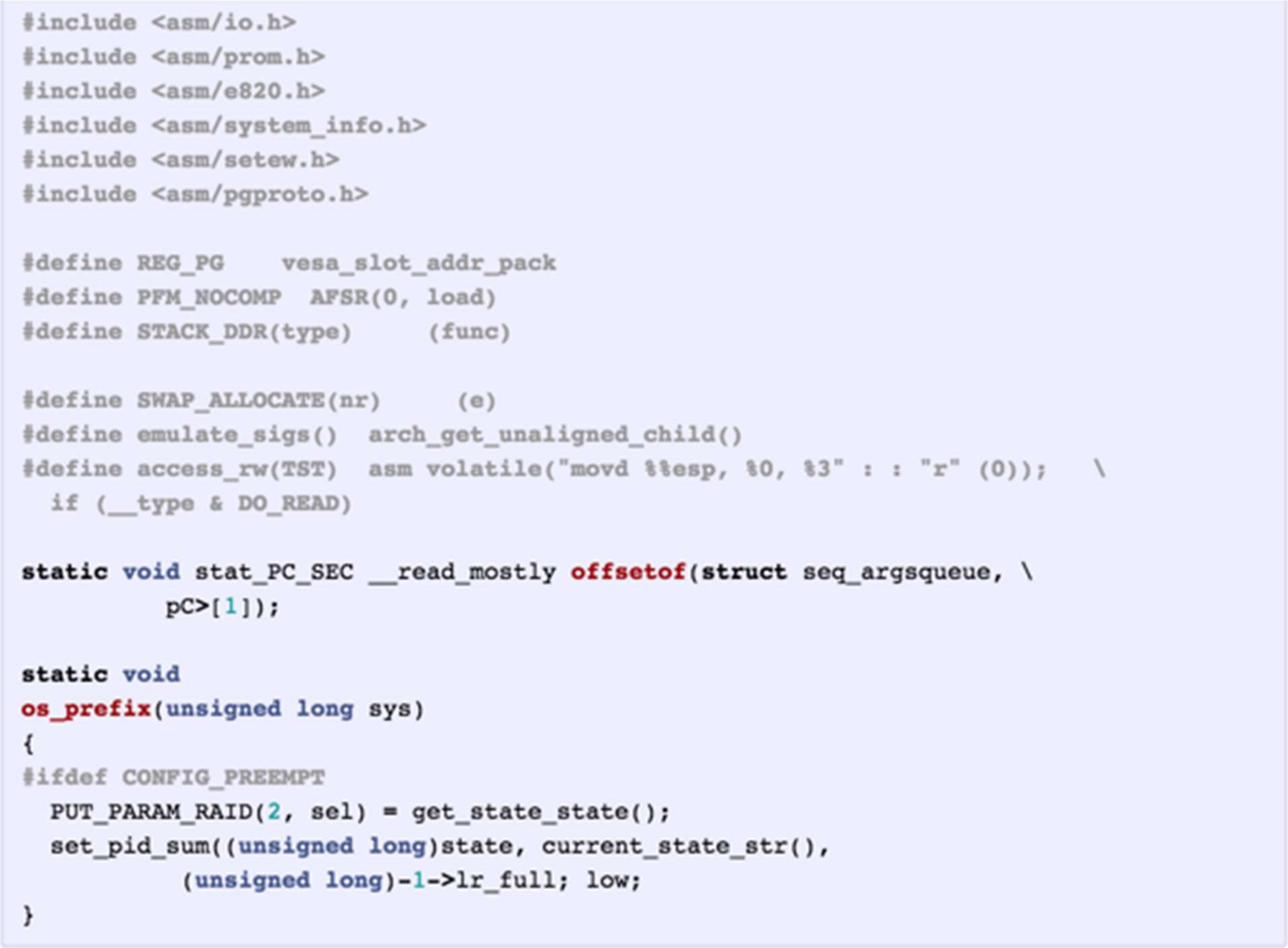
```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 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: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



```
/*
 *  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>
```

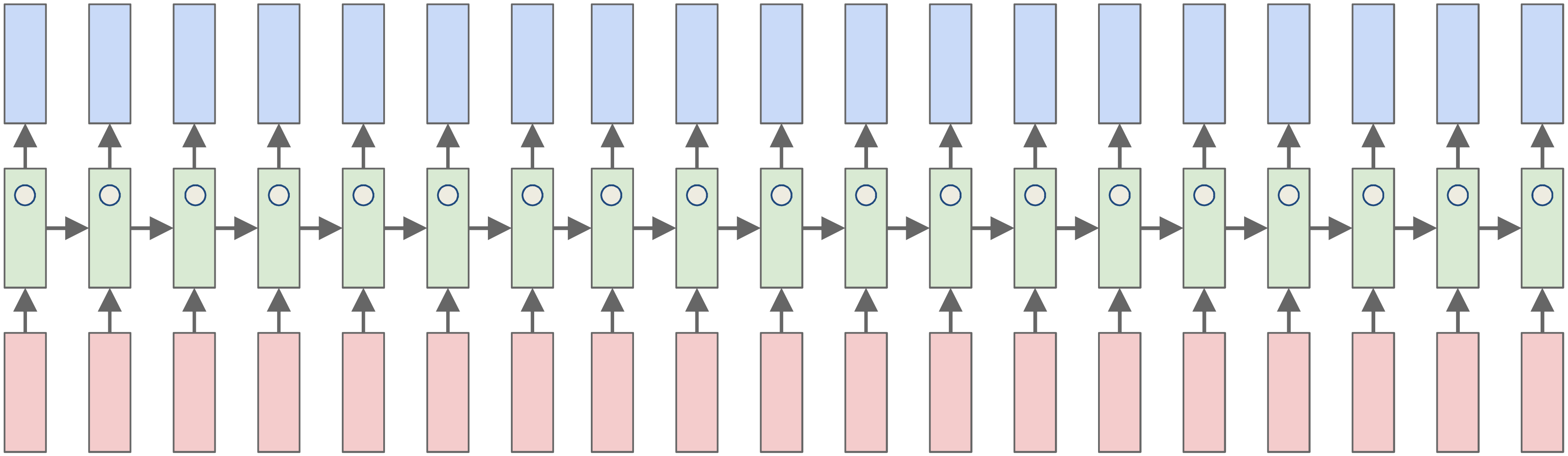
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



```
#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: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

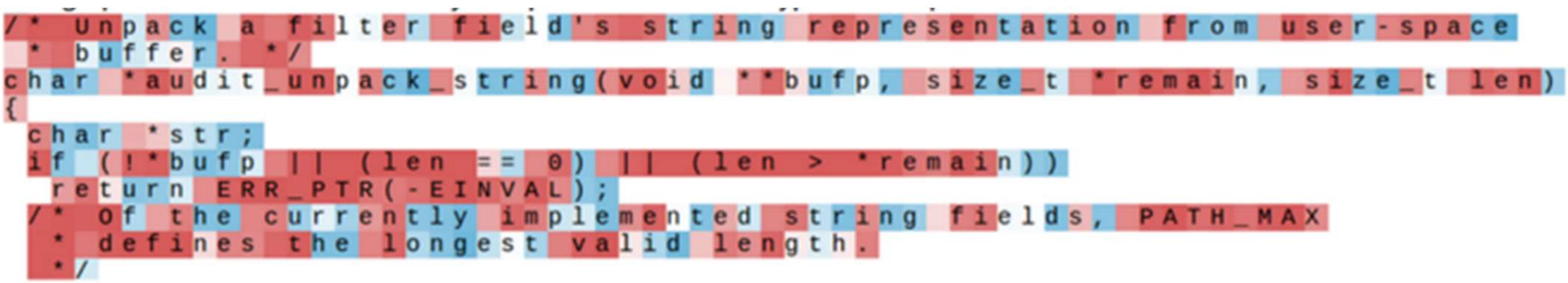
# Searching for  interpretable cells



Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Searching for  interpretable cells



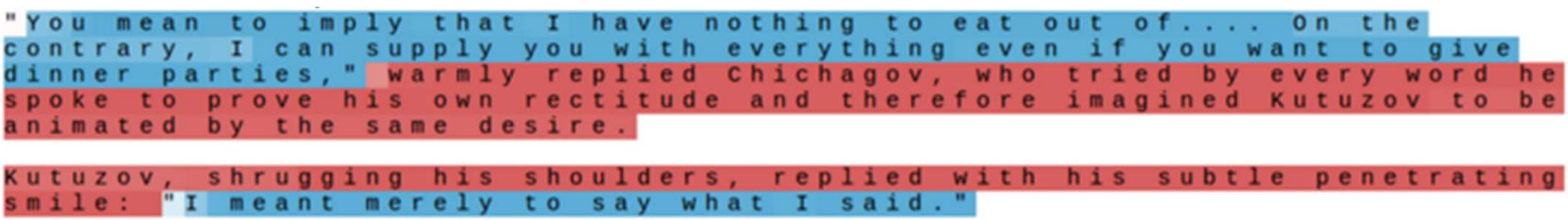
```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
 char *str;
 if (!*bufp || (len == 0) || (len > *remain))
  return ERR_PTR(-EINVAL);
 /* Of the currently implemented string fields, PATH_MAX
  * defines the longest valid length.
  */
```

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Searching for  interpretable cells



"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

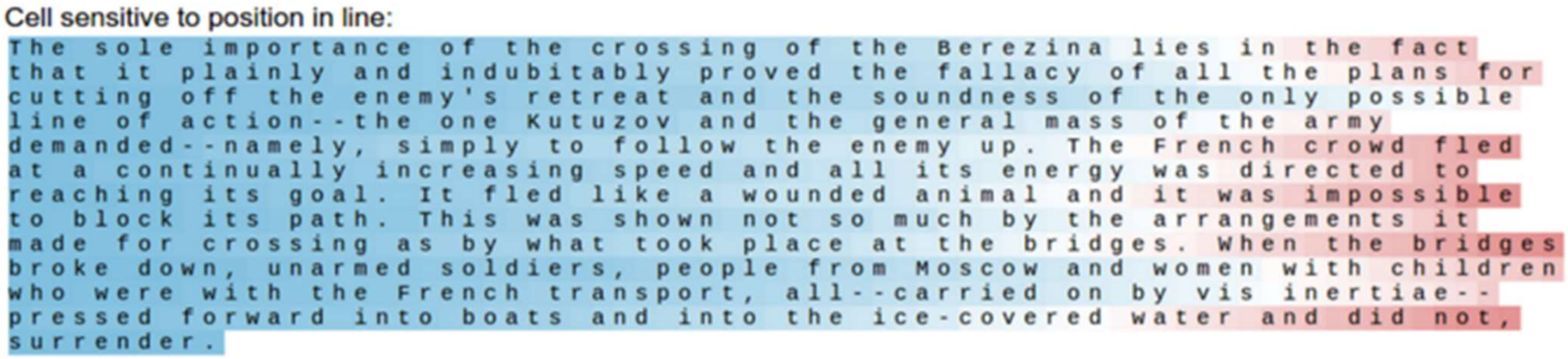
quote detection cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Searching for interpretable cells



Cell sensitive to position in line:

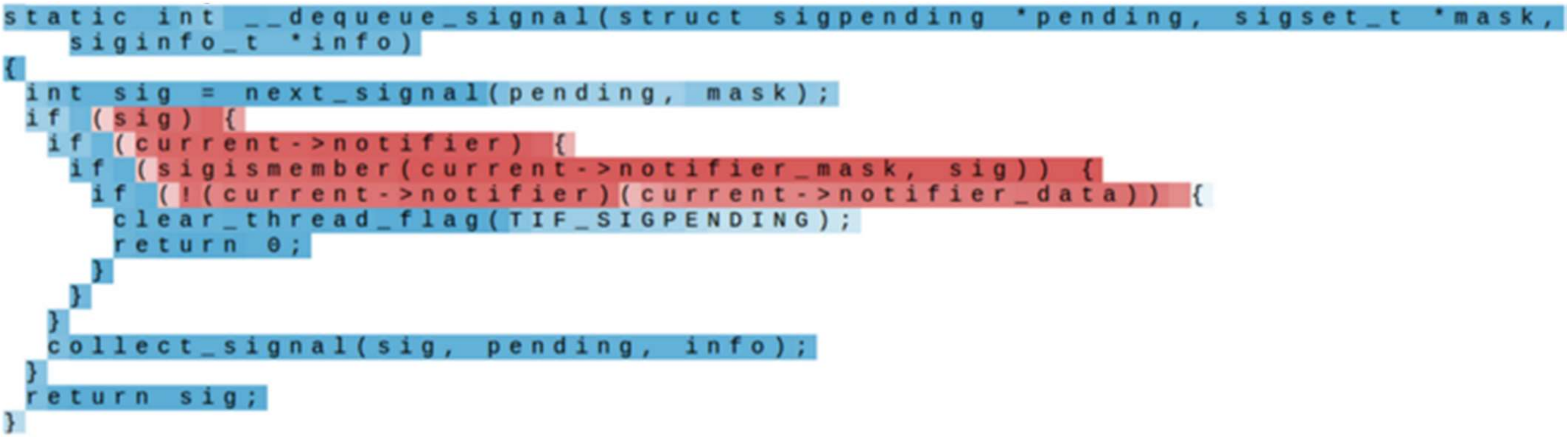
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.

line length tracking cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Searching for  interpretable cells



```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
   siginfo_t *info)
{
 int sig = next_signal(pending, mask);
 if (sig) {
  if (current->notifier) {
   if (sigismember(current->notifier_mask, sig)) {
    if (!(current->notifier)(current->notifier_data)) {
     clear_thread_flag(TIF_SIGPENDING);
     return 0;
    }
   }
  }
  collect_signal(sig, pending, info);
 }
 return sig;
}
```

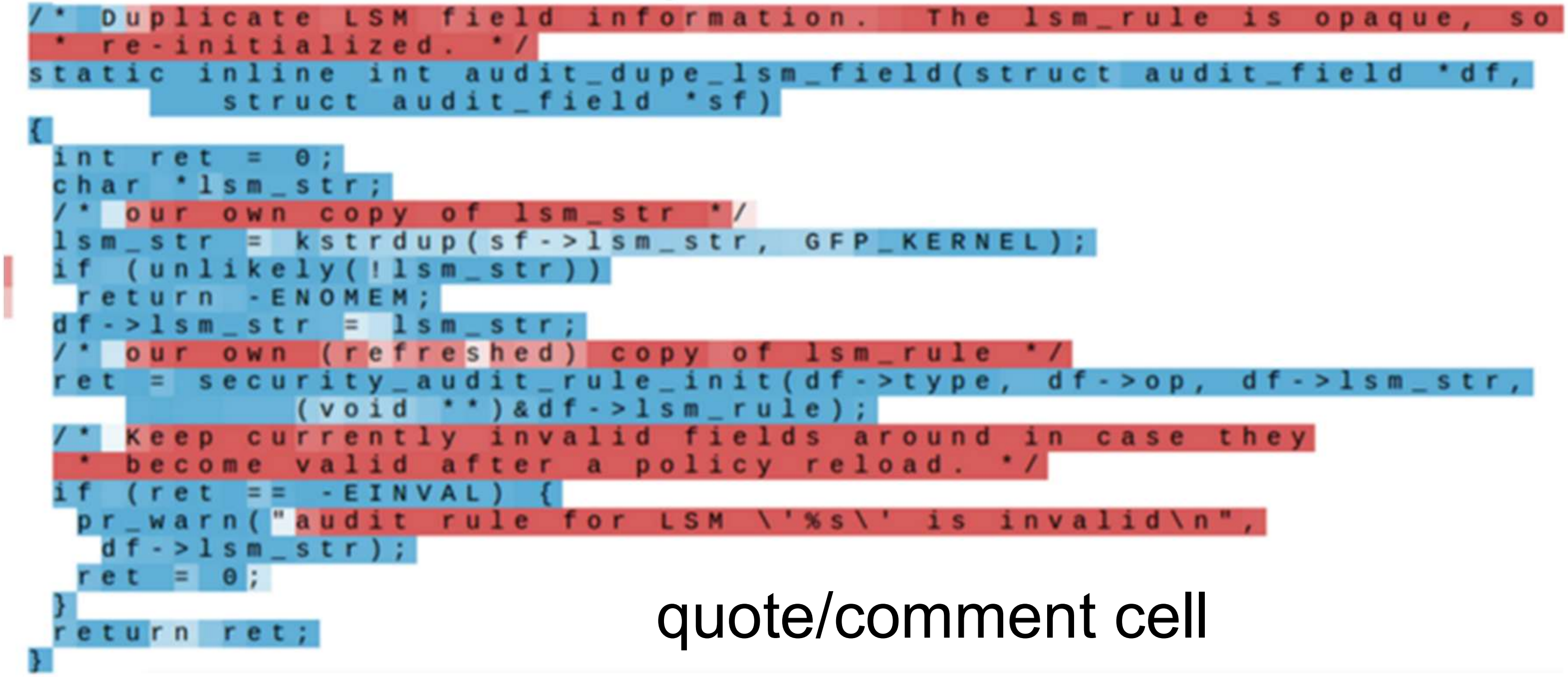
if statement cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Searching for interpretable cells

Cell that turns on inside comments and quotes:



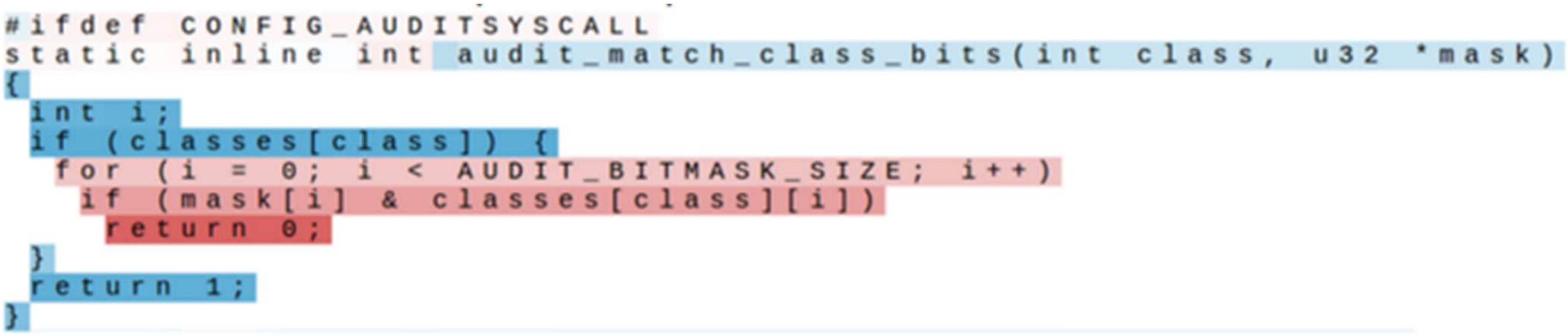
```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
     struct audit_field *sf)
{
 int ret = 0;
 char *lsm_str;
 /* our own copy of lsm_str */
 lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
 if (unlikely(!lsm_str))
  return -ENOMEM;
 df->lsm_str = lsm_str;
 /* our own (refreshed) copy of lsm_rule */
 ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
        (void **)&df->lsm_rule);
 /* Keep currently invalid fields around in case they
  * become valid after a policy reload. */
 if (ret == -EINVAL) {
  pr_warn("audit rule for LSM \'%s\' is invalid\n",
   df->lsm_str);
  ret = 0;
 }
 return ret;
}
```

quote/comment cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

# Searching for interpretable cells



```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```

code depth cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n