CSCI 599: Deep Learning and its Applications

Lecture 5

Spring 2019
Joseph J. Lim

Office Hours

- Instructor OH @ RTH 402
  - Tuesday 2-3:30pm
  - This is NOT for homework related questions.
- TA OH @ SAL 125
  - Monday 5-6pm & Tuesday 12:30-1:30pm
  - Extra OH for assignment due & midterm weeks: Monday 4-5pm & Tuesday 11:30-12:30pm

CSCI 599/699

Communication

- Please use **Piazza** for any general communication including questions
  https://piazza.com/usc/spring2019/csci599/home
- E-mails will be ignored.
- **Register TODAY. Look for your project team mates!**

Assignment 1

- Assignment 1 is released on the website (https://csci599-dl.github.io/)
- DUE February 19th, 2019 (week 7)
- Collaboration is OK! Please list all collaborators' names.

** Date changes **

- Midterm
- Project proposal

Important Dates

- Entrance exam: 1/15
- **Assignment 1: 2/19**
- **Midterm: 2/26**
- Project meeting with Instructor #1: 3/6 — 3/8
- Assignment 2: 3/26
- Project meeting with Instructor #2: 4/1 — 4/3
- Project meeting with TA: 2 times (arranged later)
- Final presentation: 4/23 5-9:00pm **4 hours**

Subject to change!

Course Project

- Computational resource (**be conser**
  $150 Google Cloud credit per student
  $125 Amazon AWS credit per student
- Tentative Schedule for Project
  - **Week 5 (2/5): Course Project Team**
  - Week 8 (2/26): Course Project Proposal
  - Week 13 (4/2): Mid-report
  - Week 16 (4/23): Project Presentation (5-9pm) + Report

Subject to change!

Course Project

- Team-based project (4-5 students per team)
- Each team will have at least 1 dedicated TA
  - 2 Mandatory meetings with TA
  - 2 meetings with me
- Create your own problems (extra points)
  - **Talk and discuss** with your TAs and me!
  - In the worst case, we will give a project idea
    - Less fun, Less points!

Project Evaluation

- Creativity and difficulty of the problem setup
- Novelty of the approach
- Thoroughness of the experiments
- Quality of student's presentation, report, and meetings with TA/instructor

Extra credit for creating your own project (OK to discuss and get help from TAs)

Project

- Report: a blog post (rather than a paper format)
- Lots of “what if we fail” questions

Team Formation

- Due today!

Project meeting with TA #1

- Try to bring something concrete!
- We will post on Piazza your TA assignment
  - Temporary until the proposal due date
- Please sign up for your team on your TA's doodle link.
- Please be available — if not, we may not be able to schedule one with you.

Today's agenda

- Recap: CNNs
- Training Neural Networks
- Recurrent Neural Networks

Today's agenda

- Recap: CNNs
- Training Neural Networks
- Recurrent Neural Networks

Computational Graph

Function: f(x, y, z) = x*y+z

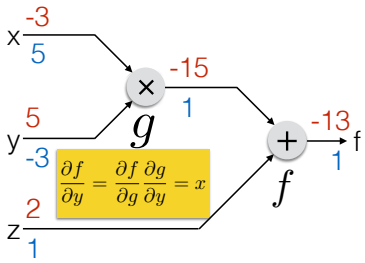
Backpropagation

We get:

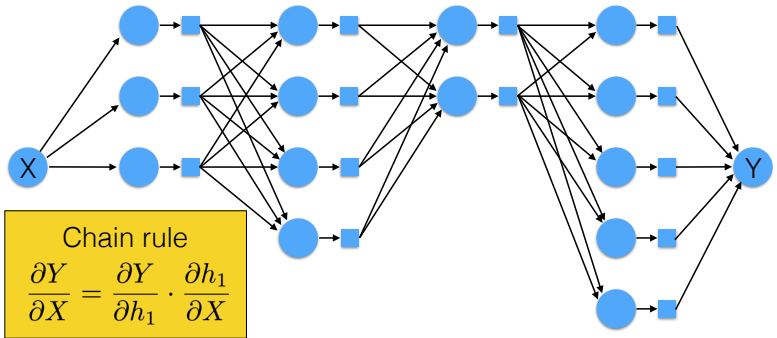
$$\frac{\partial f}{\partial x} = 5$$

$$\frac{\partial f}{\partial y} = -3$$

$$\frac{\partial f}{\partial z} = 1$$



Recap: Backward Propagation

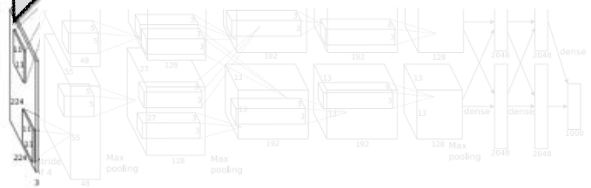


Chain rule

$$\frac{\partial Y}{\partial X} = \frac{\partial Y}{\partial h_1} \cdot \frac{\partial h_1}{\partial X}$$

Backward propagation

AlexNet in details



Image

PC: Daniel Yang

AlexNet in details

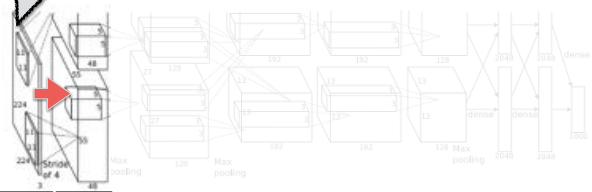


Image Conv

AlexNet in details

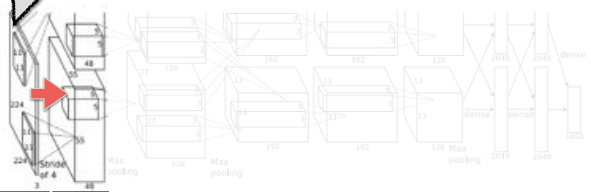


Image Conv Pool

AlexNet in details

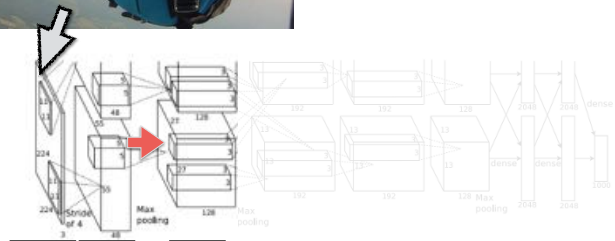


Image | Conv | Pool | Conv

AlexNet in details

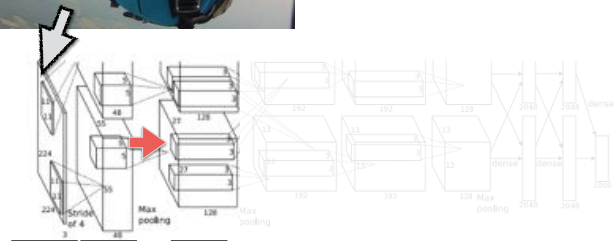


Image | Conv | Pool | Conv | Pool

AlexNet in details

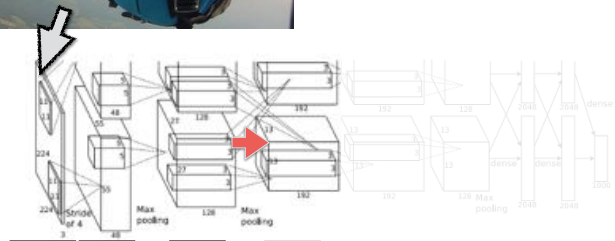


Image | Conv | Pool | Conv | Pool | Conv

AlexNet in details

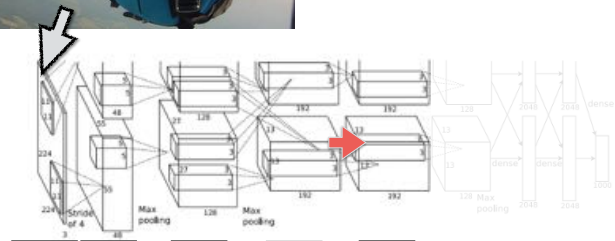


Image | Conv | Pool | Conv | Pool | Conv | Conv

AlexNet in details

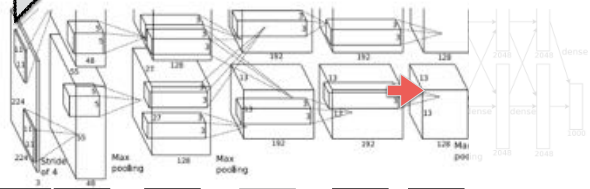


Image | Conv | Pool | Conv | Pool | Conv | Conv | Conv

AlexNet in details

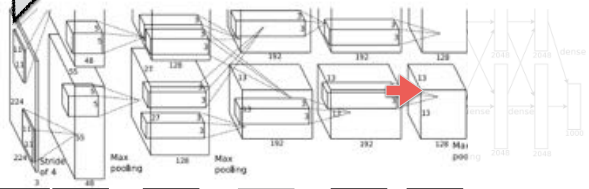


Image | Conv | Pool | Conv | Pool | Conv | Conv | Conv | Pool

AlexNet in details

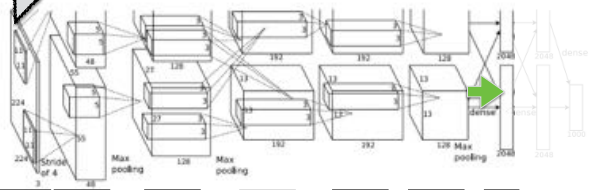


Image | Conv | Pool | Conv | Pool | Conv | Conv | Conv | Pool | FC

AlexNet in details

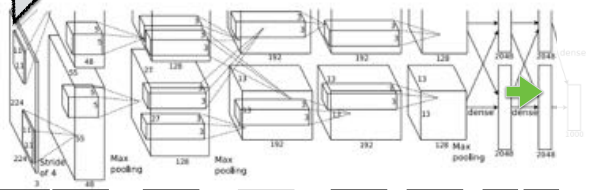
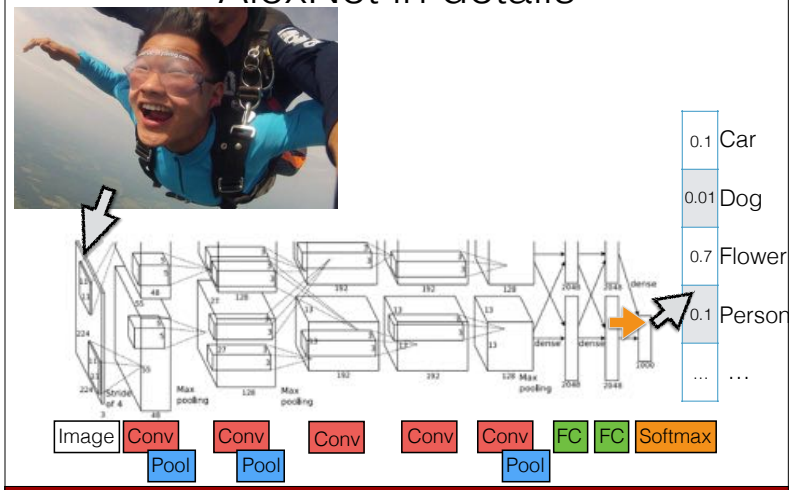
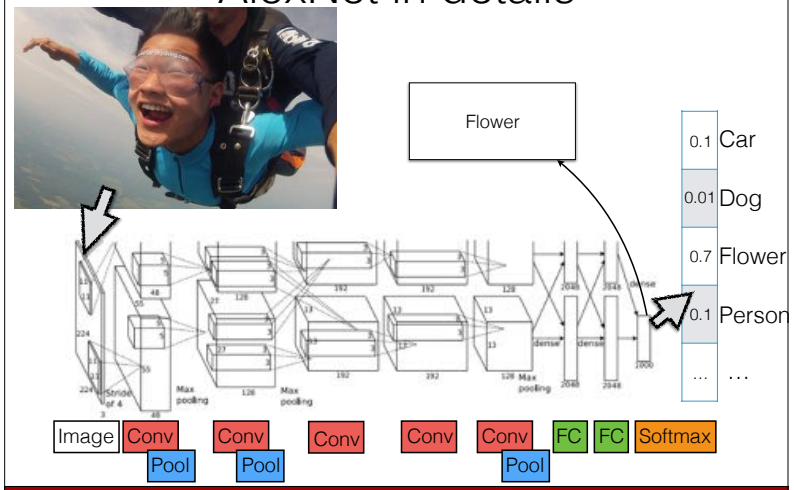


Image | Conv | Pool | Conv | Pool | Conv | Conv | Conv | Pool | FC | FC

AlexNet in details



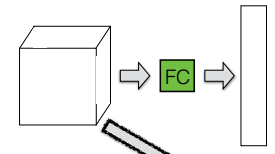
AlexNet in details



What are these layers?

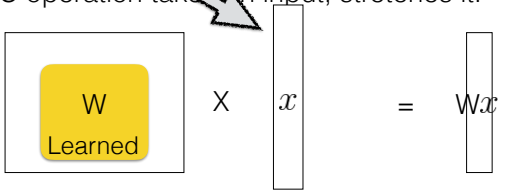
- Convolutional layer Conv
- Pool layer Pool
- Fully connected layer FC
- Softmax layer Softmax

Fully connected layer (FC)



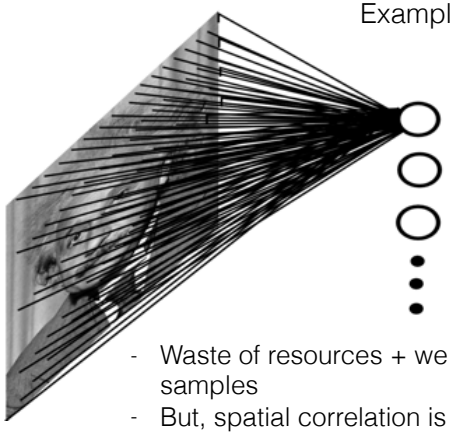
M x N x P Q x 1

FC operation takes an input, stretches it.



Q x (M*N*P) (M*N*P) x 1 Q x 1

Fully connected layer (FC)

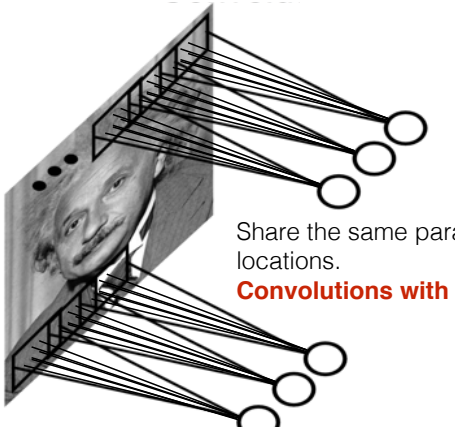


Example: 200x200 image
40k hidden units
~2B parameters!!!

- Waste of resources + we have not enough training samples
- But, spatial correlation is local!

Slide credit: Marc'Aurelio Ranzato

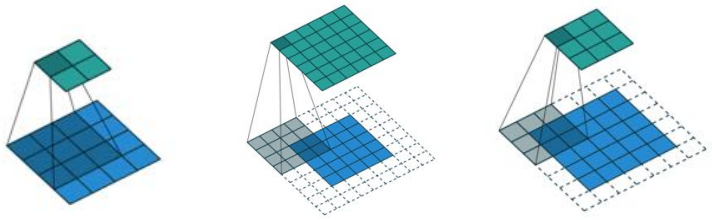
Convolutional layer



Share the same parameters across different locations.
Convolutions with learned kernels (weights)

Slide credit: Marc'Aurelio Ranzato

Convolutional layer

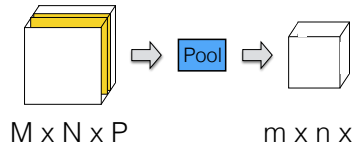


padding 0, stride 1 padding 2, stride 1 padding 1, stride 2

Note: it can have **depth** (3rd dimension).

Image credit: vdumoulin (github): https://github.com/vdumoulin/conv_arithmetic

Max pooling layer



M x N x P m x n x P

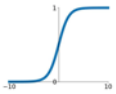
1	3	2	7	2
5	1	3	0	5
4	8	5	0	6
3	2	6	0	8
0	7	4	6	9

max pool with
3x3 filters & stride 2

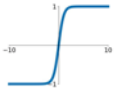
8	7
8	9

Activation functions

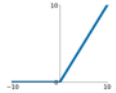
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



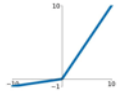
tanh
$\tanh(x)$



ReLU
$\max(0, x)$



Leaky ReLU
$\max(0.1x, x)$



Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

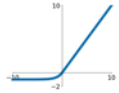


Figure from Stanford cs231n lecture slides

Summary

- Fully connected layer
- Convolutional layer
- Deconvolutional layer
- Pooling layers
- Activation layers (e.g. ReLU)

AlexNet in details

Encoder

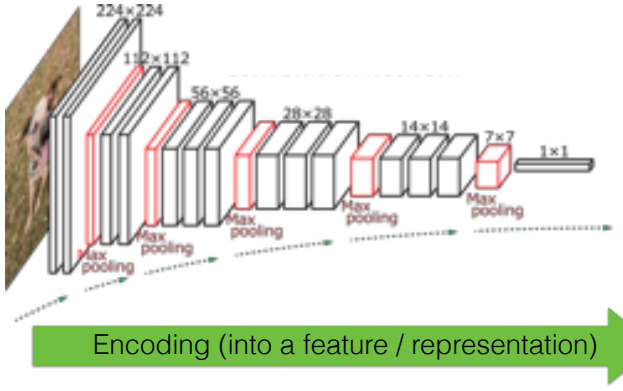


Image credit: Noh, et. al. Learning Deconvolution Network for Semantic Segmentation. ICCV 2015.

Decoder

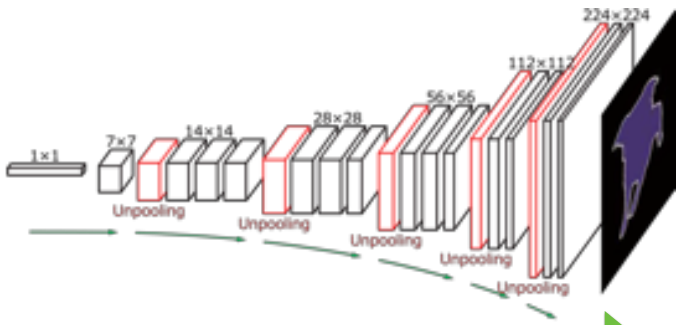


Image credit: Noh, et. al. Learning Deconvolution Network for Semantic Segmentation. ICCV 2015.

Let's revisit CV tasks!

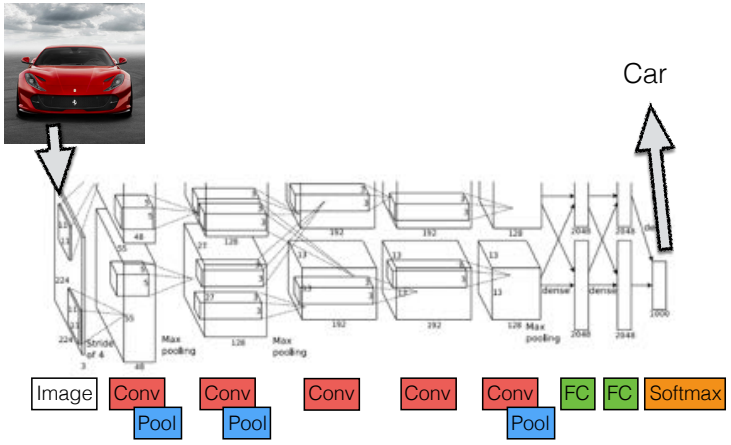
CNN applications

- Image classification
- Object detection
- Object segmentation
- Image resolution
- Human pose detection
- Action classification
- Image captioning

Image Classification

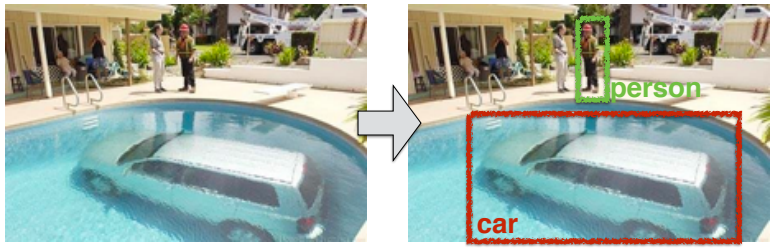
Car

Image Classification



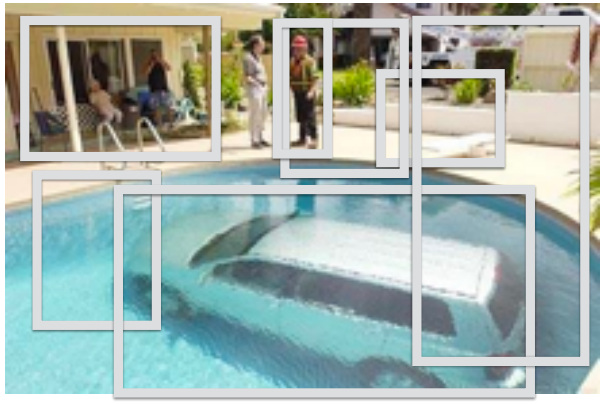
Object Detection

Object Detection



Object Detection

Object Detection

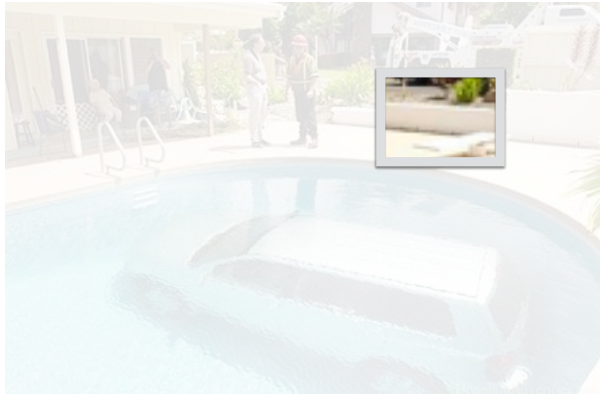


1. Object proposal

Object Detection

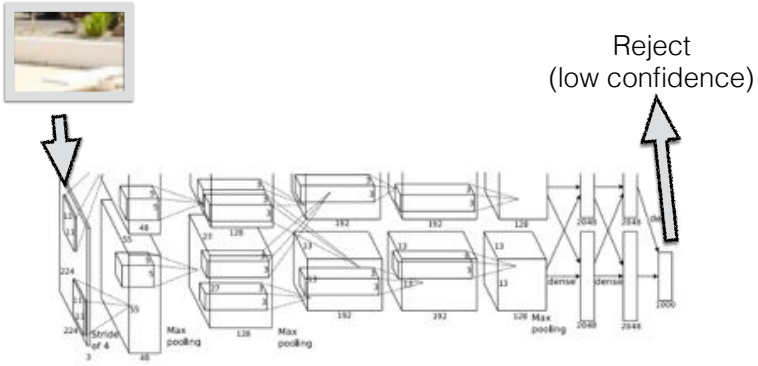
2. Evaluate each proposal

Object Detection



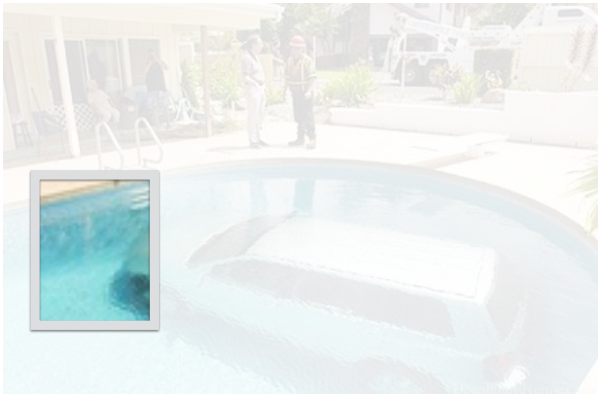
2. Evaluate each proposal

Object Detection



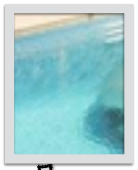
2. Evaluate each proposal

Object Detection

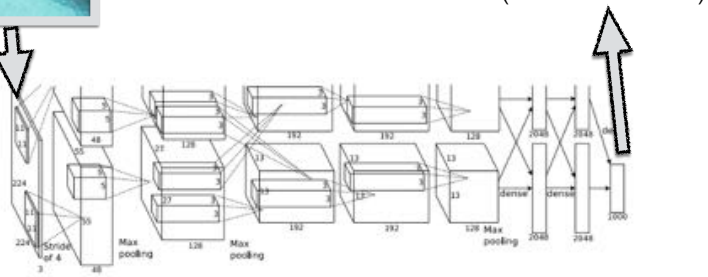


2. Evaluate each proposal

Object Detection



Reject
(low confidence)



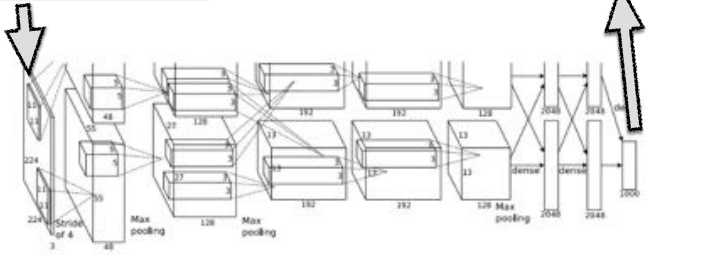
2. Evaluate each proposal

Object Detection

2. Evaluate each proposal

Object Detection

Car



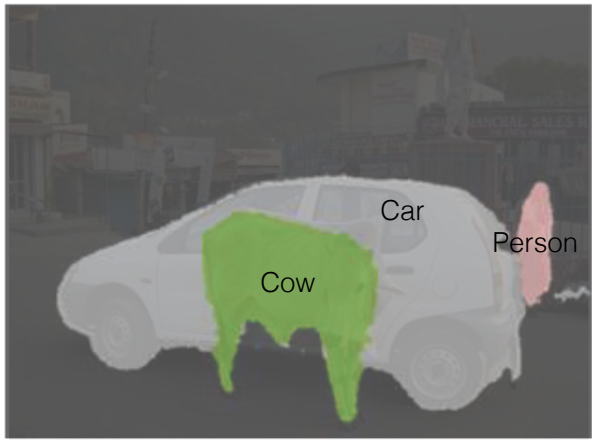
2. Evaluate each proposal

Object Detection

For more:
R-CNN (https://github.com/rbgirshick/rcnn) and Fast R-CNN (https://github.com/rbgirshick/fast-rcnn)

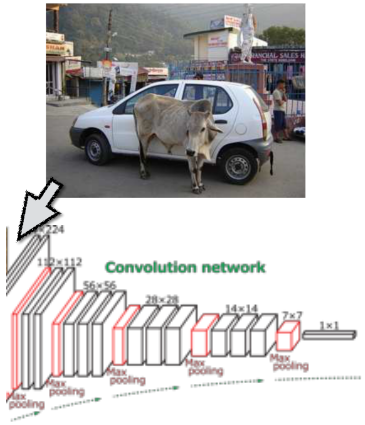
Semantic Segmentation

Semantic Segmentation

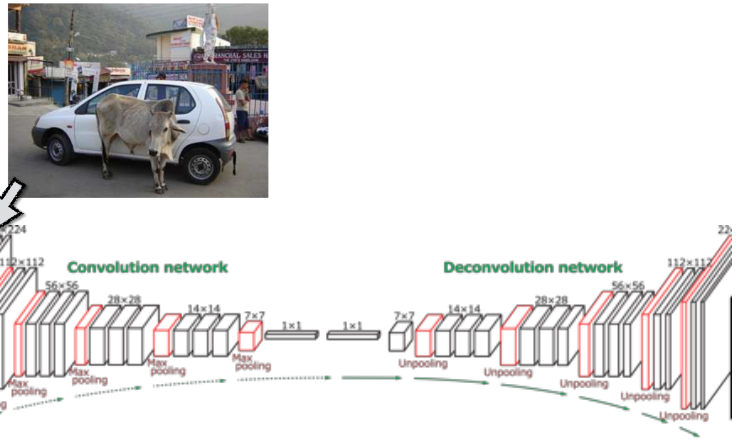


Semantic Segmentation

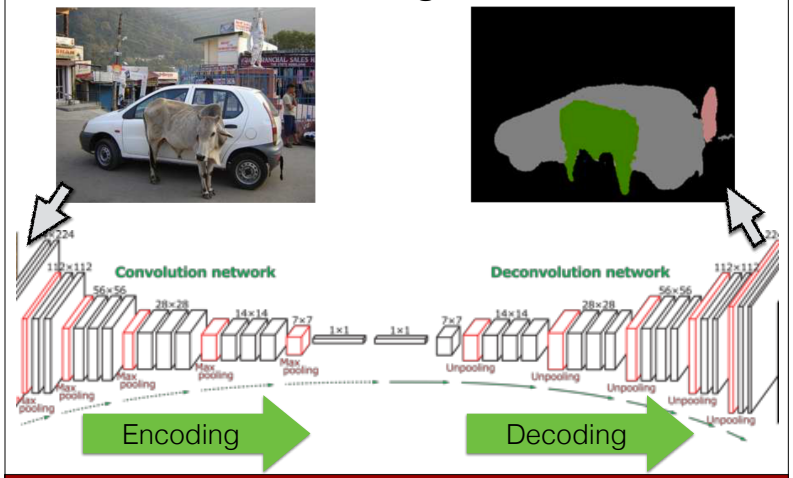
Semantic Segmentation



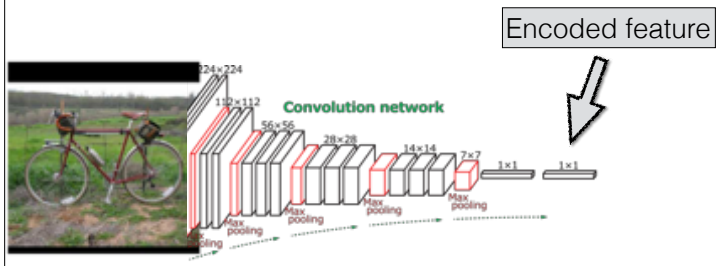
Semantic Segmentation



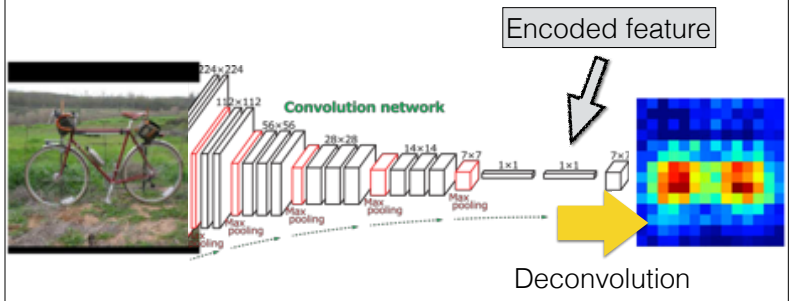
Semantic Segmentation



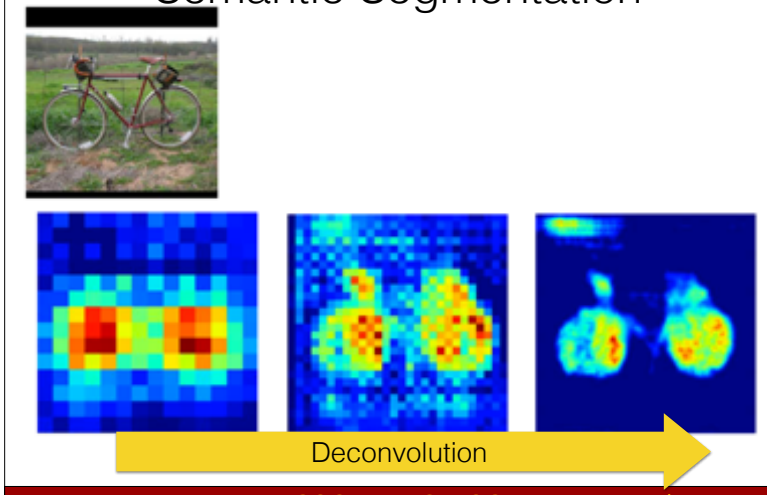
Semantic Segmentation



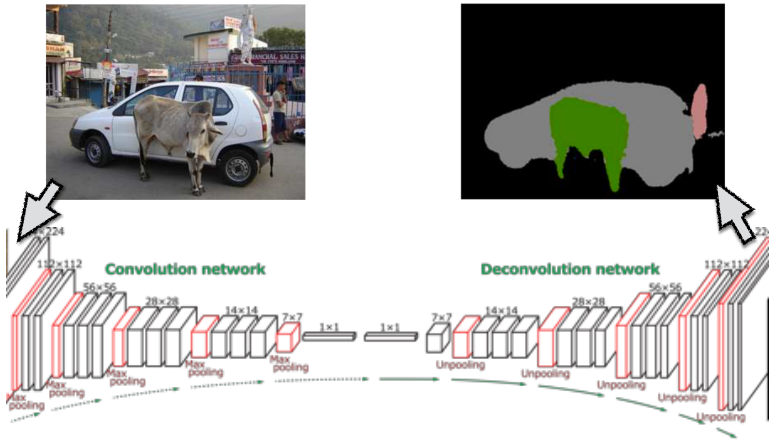
Semantic Segmentation



Semantic Segmentation

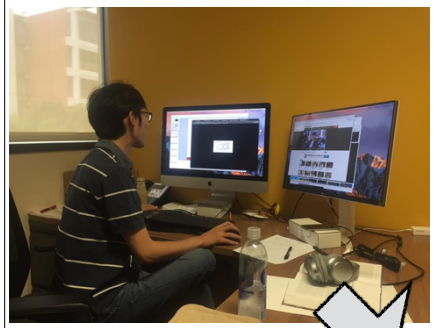


Semantic Segmentation



For more: Noh, et.al. Learning Deconvolution Network for Semantic Segmentation. ICCV 2015.

Image captioning

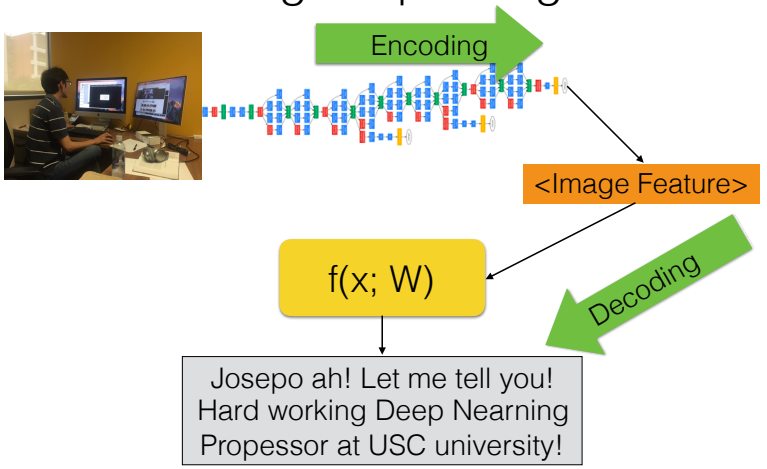


Hao Li (USC prof.) said

Josepo ah! Let me tell you!
Hard working Deep Nearning
Propessor at USC university!

Image captioning

Image captioning



Demo

- 3D Face Reconstruction from a Single Image

  http://www.cs.nott.ac.uk/~psxasj/3dme/

Demo

- 3D Face Reconstruction from a Single Image

  http://www.cs.nott.ac.uk/~psxasj/3dme/

Demo

- 3D Face Reconstruction from a Single Image

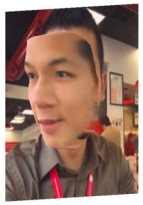
  http://www.cs.nott.ac.uk/~psxasj/3dme/



Demo

- 3D Face Reconstruction from a Single Image

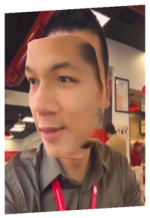
  http://www.cs.nott.ac.uk/~psxasj/3dme/


Demo

- 3D Face Reconstruction from a Single Image

  http://www.cs.nott.ac.uk/~psxasj/3dme/

Today's agenda

- Recap: CNNs
- Training Neural Networks
- Recurrent Neural Networks

All inputs are just numbers

Image

a	b	c	d	e
f	g	h	i	j
k	l	m	n	o
p	q	r	s	t
u	v	w	x	y

Sound

a	b	c	d	e

Data preprocessing

- Zero-center data
- Normalize data
- PCA Whitening

Data preprocessing

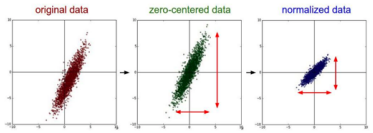
Why do we need preprocessing?

- Zero-center data
- Normalize data
- PCA Whitening

Data preprocessing

Why do we need preprocessing?

- Zero-center data
- Normalize data
- PCA Whitening



Activation functions (ReLU) works around **zero**.

If input data is biased toward positive or negative, randomly initialized layers produce biased output.

Figure from Stanford cs231n lecture slides

Data preprocessing

Why do we need preprocessing?

- Zero-center data
- Normalize data
- PCA Whitening

A feature with small scale has a negligible effect on backprop.

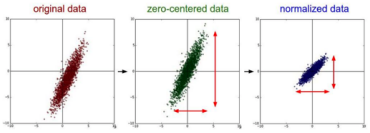


Figure from Stanford cs231n lecture slides

Data preprocessing

Why do we need preprocessing?

- Zero-center data
- Normalize data
- PCA Whitening

Remove correlation between input features

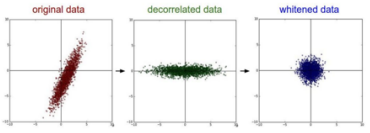
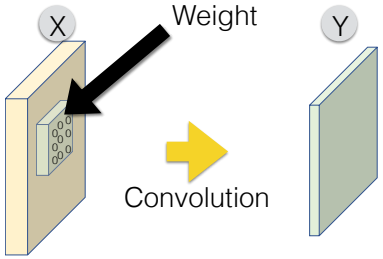


Figure from Stanford cs231n lecture slides

Weight initialization

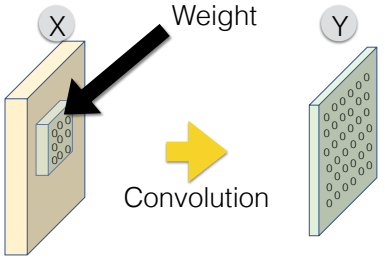
If all weights are 0?



Slide credit: CS 231N

Weight initialization

If all weights are 0?



Gradient vanishing problem

Output **Y** will be always 0 and then gradients will go to 0.

Slide credit: CS 231N

Weight initialization

- Small random numbers
- Xavier
- Xavier / 2 (He et. al.)

Slide credit: CS 231N

Small Random Numbers

- Sample from a gaussian distribution with 0 mean and given standard deviation (such as 0.01)

$$W \sim \mathcal{N}(\mu, \sigma^2)$$

- Doesn't work for deeper networks

Small Random Numbers

If the weights are too small,

the activation signals shrink quickly

If the weights are too large,

the activation signals explode

Xavier Initialization

- Set the variance according to # of input neurons

$$W \sim \mathcal{N}(\mu, \sigma^2) \quad \longleftarrow \quad Var(W) = \frac{1}{n_{in}}$$

- Consider # output neurons for back-prop

$$Var(W) = \frac{2}{n_{in} + n_{out}}$$

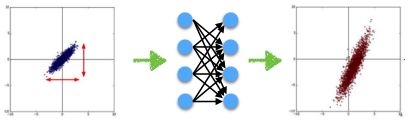
He et. al. Initialization

$$\frac{Random\ Numbers\ of\ dimension\ (D_{in}, D_{out})}{\sqrt{D_{in}\ /\ 2}}$$

- Default choice of training a deep network

Normalization

- After each layer, the distribution of activation signals changes (Internal Covariate Shift)

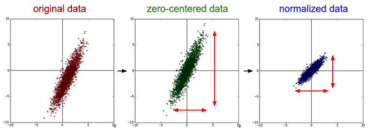


- As a network becomes deeper, distribution shifts more.
- Recall why we need data preprocessing

Figure from Stanford cs231n lecture slides

Normalization

- Normalize the distribution of activations to have zero mean and unit variance



Same as data preprocessing

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

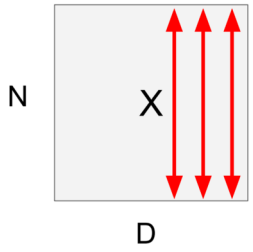
Figure from Stanford cs231n lecture slides

Batch normalization

- Why do we need normalization?
  - Internal Covariate Shift
  - To train deeper networks on larger data
- Make each dimension Unit Gaussian

Batch normalization

- Compute the empirical mean and variance independently for each dimension and normalize.



Batch normalization

- Vanilla Normalization:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch normalization

- Squashing the range:

$$y^{(k)} = \gamma^{(k)}\widehat{x}^{(k)} + \beta^{(k)}$$

- Network can learn:

$$\gamma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$$
$$\beta^{(k)} = \mathrm{E}[x^{(k)}]$$

Batch normalization

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

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

[Ioffe and Szegedy, 2015]

Batch normalization

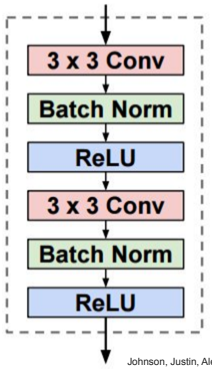
- Why is it good?
  - Improves gradient flow
  - Allows higher learning rates
  - Reduces strong dependence on initialization
  - Regularization

Batch normalization

- At test time
  - The mean and variance are not computed based on the batch.
  - Empirical mean of activations during training is used.

Batch normalization

- Usually added after convolutional layers or fully connected layers
- before non-linearities



Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." *European Conference on Computer Vision*. Springer International Publishing, 2016.

Training details

- Learning rate
- Batch size
- Hyperparameter
- Loss curve

Learning Rate

How **fast** or **slow** we update the model parameters

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$

Learning Rate

How **fast** or **slow** we update the model parameters

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$

Gradients

Learning Rate

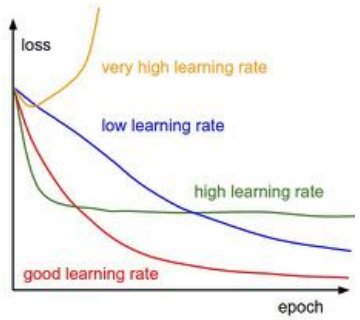
Too large

- Explode

Too small

- Get stuck in local minimum

Learning Rate



ref: http://cs231n.github.io/neural-networks-3/#anneal

Learning Rate

Too large

- Explode

Too small

- Get stuck in local minimum

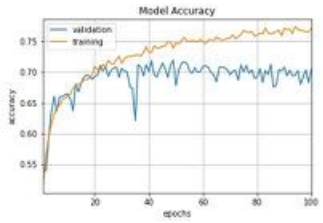
Anneal the learning rate

Learning Rate

- Constant
- Step decay
- Exponential decay
- Time-based decay
- Etc

Learning Rate

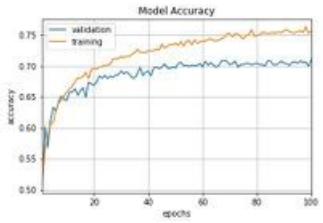
Constant $lr = lr_0$



Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

Time-based decay $lr = \frac{lr_0}{1 + k * iter}$

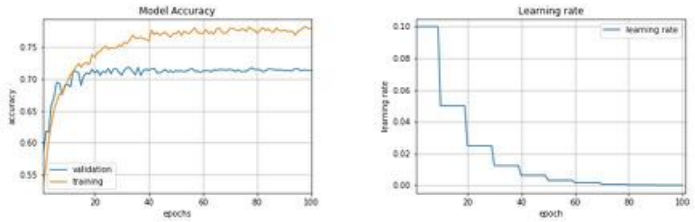


Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

Step decay

$$lr = lr_0 * floor(\frac{epoch}{epoch\_drops})$$

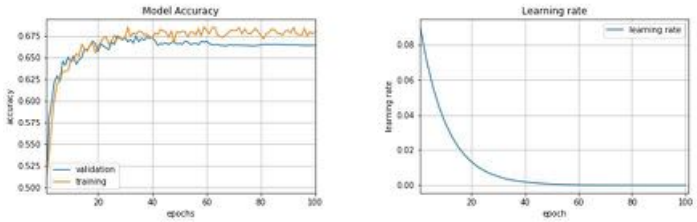


Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

Exponential decay

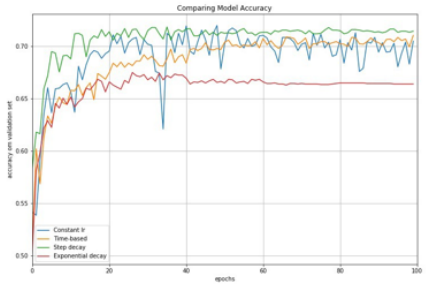
$$lr = \frac{lr_0}{\exp^{k*iter}}$$



Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

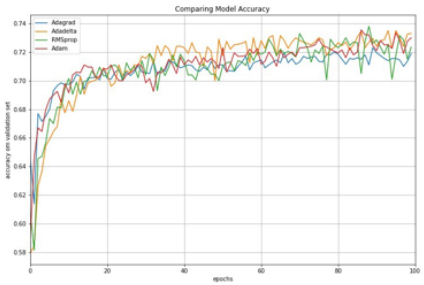
Comparisons



Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Learning Rate

Adaptive learning rate methods



Dataset: CIFAR-10 ref: https://goo.gl/xnrb3N

Batch Size

Number of samples in one forward/backward pass

Batch Size

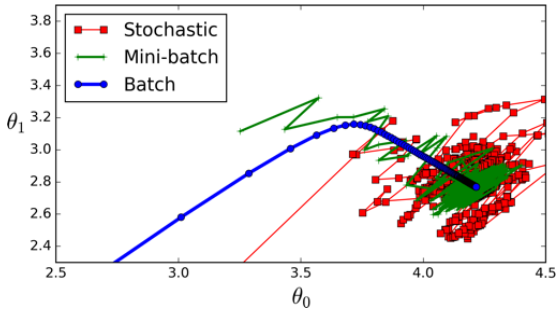
Too large

- Memory inefficient
- Computation inefficient
- Converge to sharp minimizers of the training function

Too small

- Less accurate gradients at each iteration
- Converge to flat minimizers of the training function

Batch Size



Stochastic: batch size is 1

ref: https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network

Optimizers and Algorithms

Gradient Descent

- Gradient descent
- Stochastic gradient descent
- Mini-batch gradient descent
- Momentum
- Nesterov accelerated gradient
- Etc

Gradient Descent

Computes the gradient of the cost function w.r.t. to the parameters θ for **the entire training dataset at each iteration**

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} f(\theta_t)$$

Stochastic Gradient Descent

Computes the gradient of the cost function w.r.t. to the parameters θ for **each data point at each iteration**

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} f(\theta_t; x^{(i)})$$

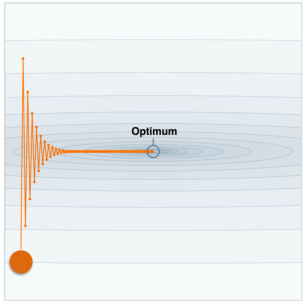
Mini-batch Gradient Descent

Computes the gradient of the cost function w.r.t. to the parameters θ for performs an update for **every mini-batch of n training examples at each iteration**

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} f(\theta_t; x^{(i:i+n)})$$

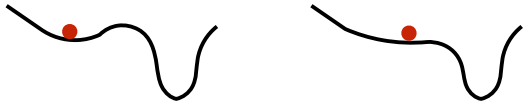
Momentum

GD family updates parameters inefficiency



Momentum

GD family gets stuck at local minimums and saddle points easily



local minimum saddle point

Momentum

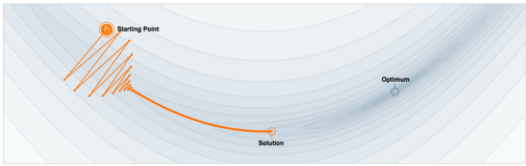
Consider the history:
Adding a fraction γ of **the update vector of the past time step** to the current update vector

$$\nu_t = \gamma \nu_{t-1} + \eta \nabla_{\theta_t} f(\theta_t)$$

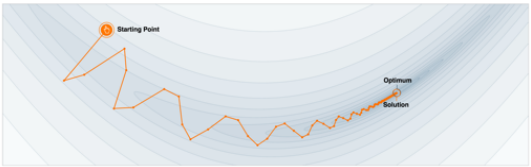
$$\theta_{t+1} = \theta_t - \nu_t$$

Momentum

Without momentum

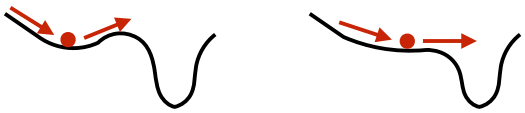


With momentum



Momentum

GD family gets stuck at local minimums and saddle points easily



Nesterov accelerated gradient

Calculate the gradient not w.r.t. to our current parameters but w.r.t. **the approximate future position of our parameters**:

$$\nu_t = \gamma \nu_{t-1} + \eta \nabla_{\theta_t} f(\theta_t - \gamma \nu_{t-1})$$

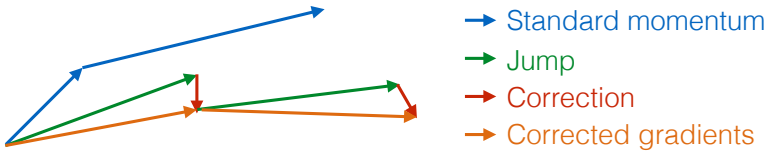
$$\theta_{t+1} = \theta_t - \nu_t$$

Nesterov accelerated gradient

Calculate the gradient not w.r.t. to our current parameters but w.r.t. **the approximate future position of our parameters**:

$$\nu_t = \gamma \nu_{t-1} + \eta \nabla_{\theta_t} f(\theta_t - \gamma \nu_{t-1})$$

$$\theta_{t+1} = \theta_t - \nu_t$$



Regularization

- Extra terms
- Dropout
- Data augmentation

Regularization

$$L = L_{original} + \lambda R(\theta)$$

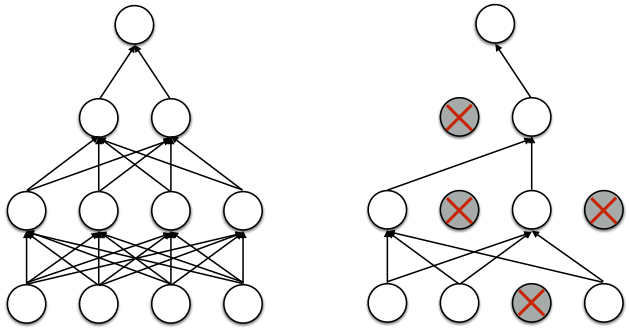
L2 norm regularizer $R(\theta) = \Sigma_i \Sigma_j \theta_{i,j}^2$

L1 norm regularizer $R(\theta) = \Sigma_i \Sigma_j |\theta_{i,j}|$

L1+**L2** norm regularizer $R(\theta) = \Sigma_i \Sigma_j |\theta_{i,j}| + \alpha \theta_{i,j}^2$

Dropout

At each iteration, randomly shutdown some neurons with a certain probability (usually 0.5)



Dropout

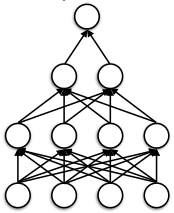
Intuition

- Forces the network to have a redundant representation
- Training a bunches of small models sharing parameters
  - Each binary mask is one model

Dropout

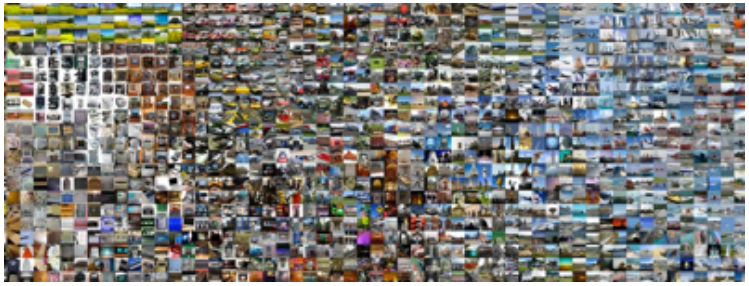
Testing phase

- All neurons are active always
- Scale the activations so that for each neuron
  - Testing output = expected output at training time



Data Augmentation

Deep networks need to be trained on a huge number of training images



What if… we don't have enough amount of data?

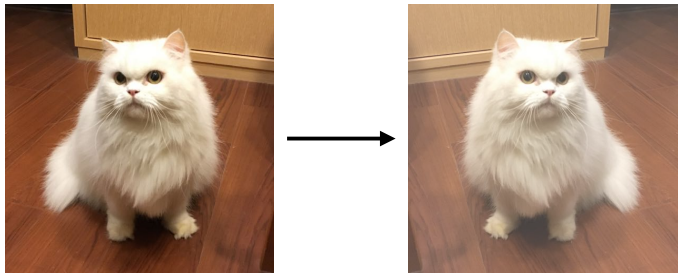
Data Augmentation

Some tricks for image data augmentation

- Flip
- Scale
- Rotate
- Crop
- Color jitter
- etc

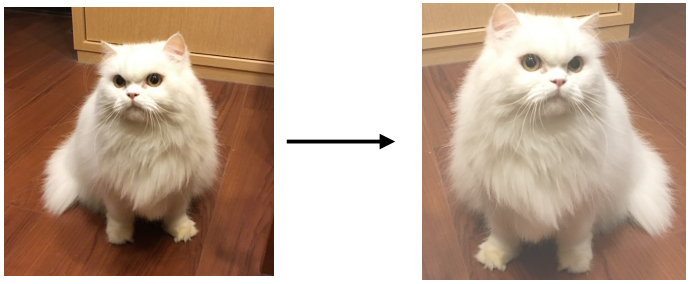
Data Augmentation

Flip



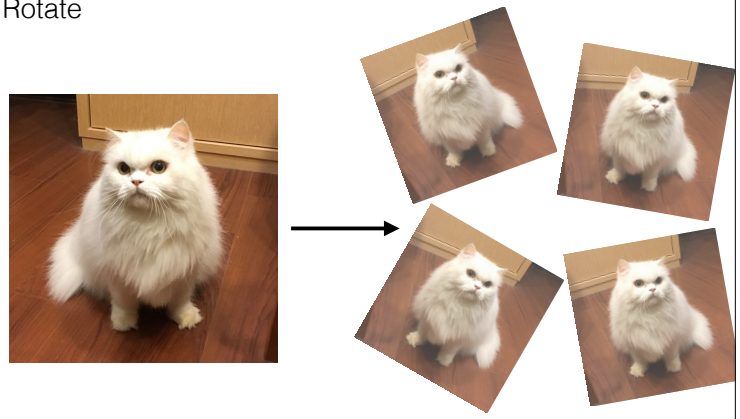
Data Augmentation

Scale



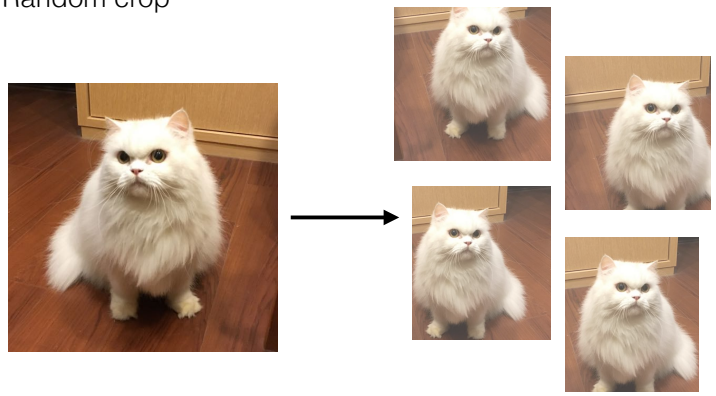
Data Augmentation

Rotate

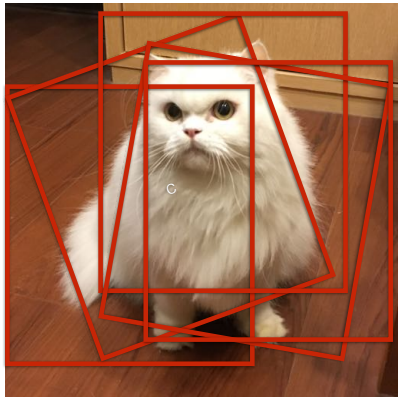


Data Augmentation

Random crop



Data Augmentation



Summary

Deep learning is all about getting everything right…

- Data
- Model
  - Architecture
- Optimizer
- Hyperparameters
- Training tricks
  - Initialization, normalization, regularization, etc

Questions?

Todo

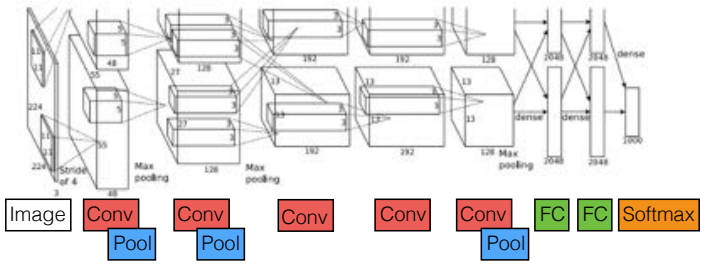
- Find Teammates for your Project
- Assignment 1 will be out next week!

Questions?

Today's agenda

- Recap: CNNs
- Training Neural Networks
- Recurrent Neural Networks

Recap: CNN (AlexNet)

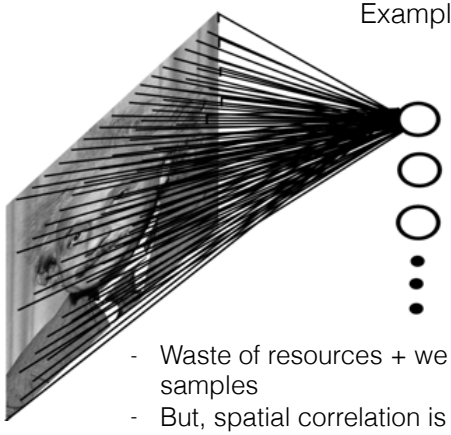


A Krizhevsky, et. al. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012.

Recap: CNNs

- Convolutional layer — Conv
- Pool layer — Pool
- Fully connected layer — FC
- Softmax layer — Softmax
- Activation layer (e.g. ReLU) — ReLU

Recap: Fully connected layer (FC)

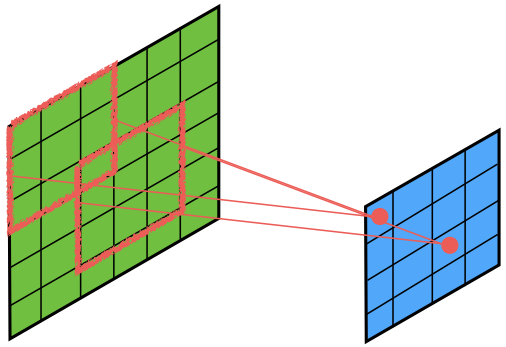


Example: 200x200 image
40k hidden units
~2B parameters!!!

- Waste of resources + we have not enough training samples
- But, spatial correlation is local!

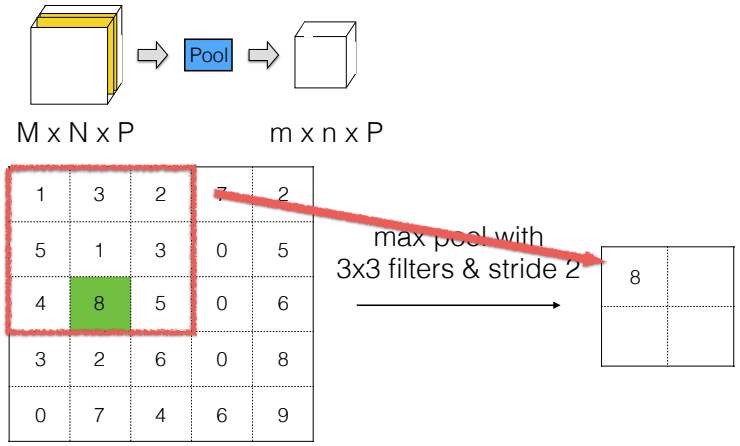
Slide credit: Marc'Aurelio Ranzato

Recap: Convolutional layer



Inspired by Marc'Aurelio Ranzato's slides

Recap: Max pooling layer



Recap: Activation functions

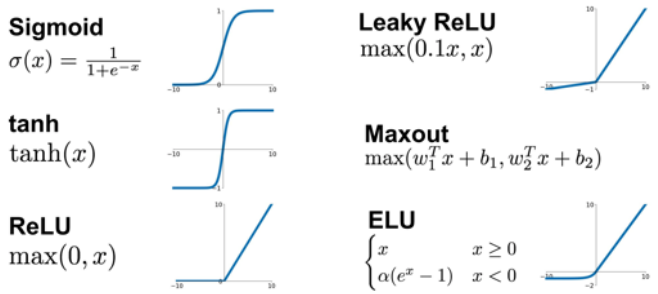


Figure from Stanford cs231n lecture slides

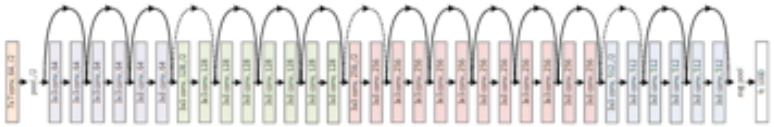
Recap: VGGNet

Simonyan and Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. ICLR 2015.

Recap: GoogLeNet

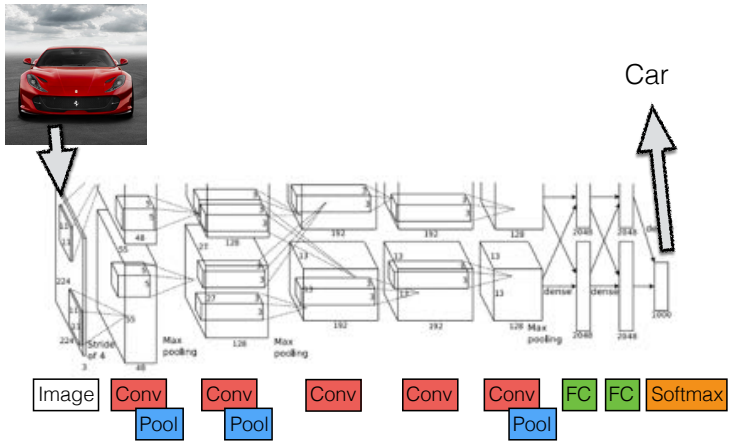
Szegedy, et. al. Going Deeper with Convolution. CVPR 2015.

Recap: ResNet

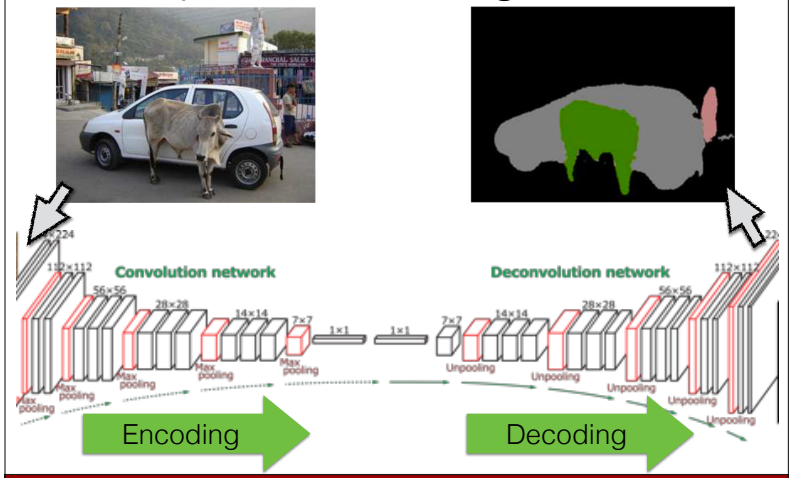


He, et. al. Deep **Residual** Learning for Image Recognition. CVPR 2016.

Recap: Image Classification



Recap: Semantic Segmentation



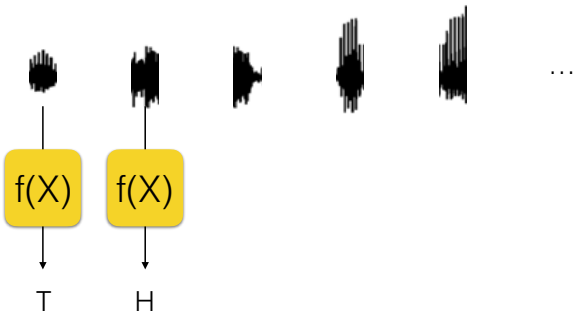
What about speech recognition?

Slide credit: Dhruv Batra
Image credit: Alex Graves and Kevin Gimpel

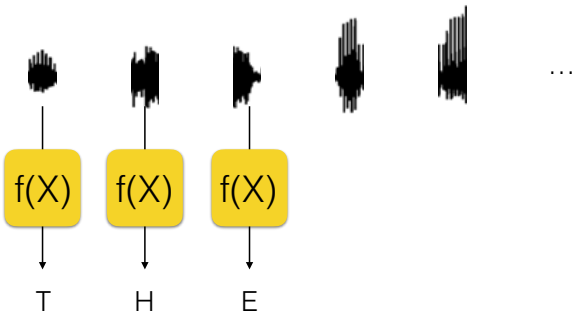
What about speech recognition?

What about speech recognition?

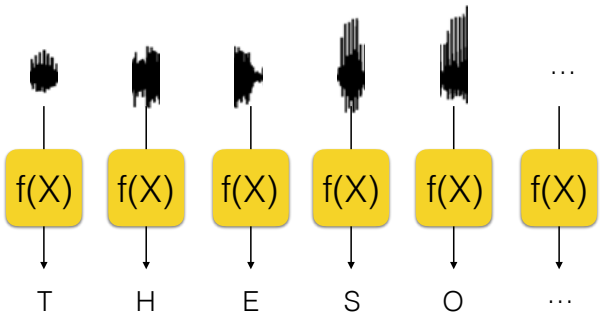
What about speech recognition?



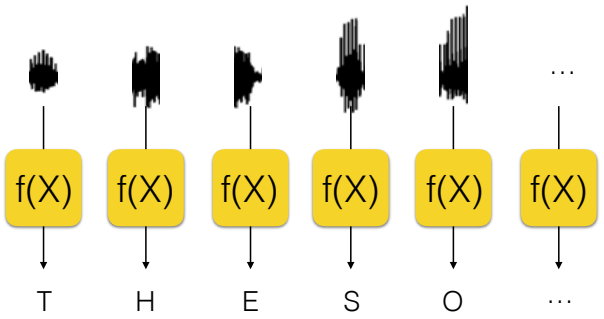
What about speech recognition?



What about speech recognition?

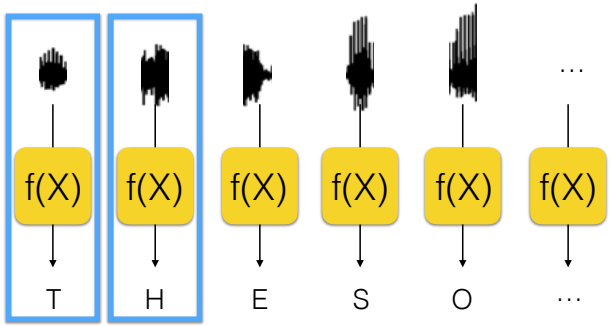


What about speech recognition?

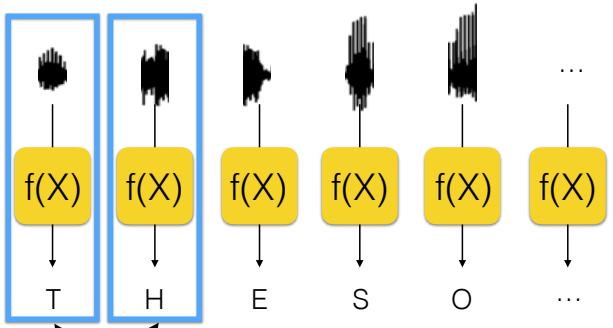


Any limitation on this approach?

A limitation of “normal” CNNs

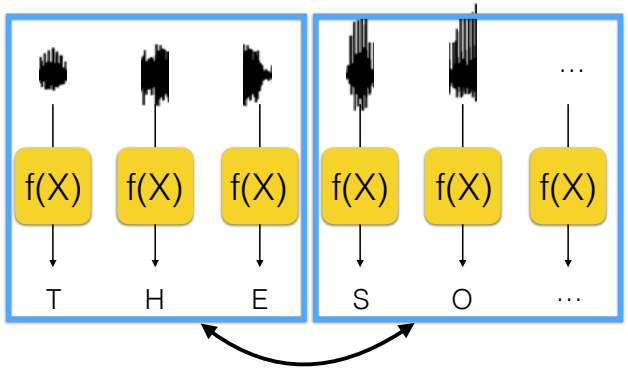


A limitation of “normal” CNNs



Independent

A limitation of “normal” CNNs



Remember Hidden Markov Model?

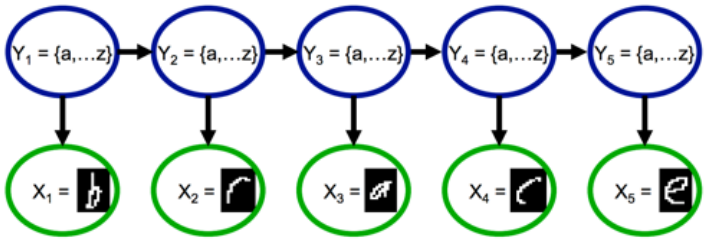
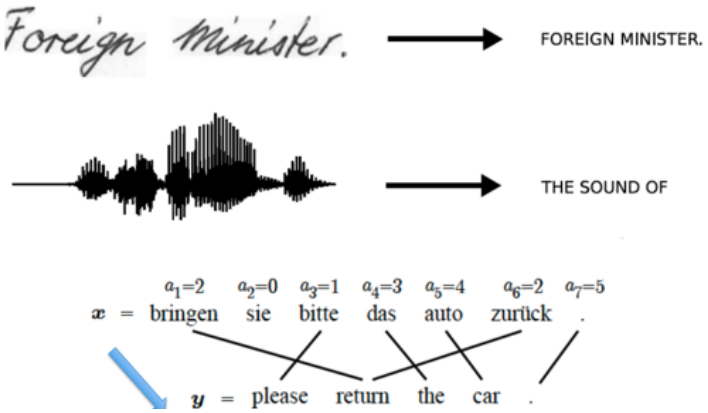


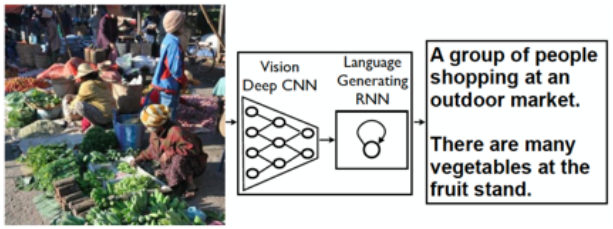
Figure credit: Carlos Guestrin

Sequential data is everywhere



Slide credit: Dhruv Batra
Image credit: Alex Graves and Kevin Gimpel

Sequential data is everywhere



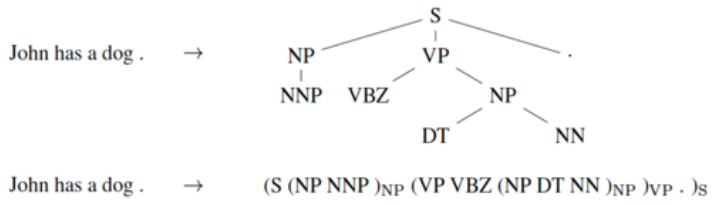
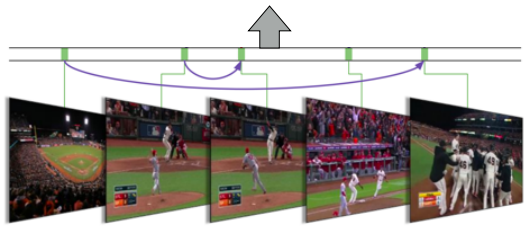


Image credit: Vinyals et. al.

Sequential data is everywhere

Baseball swing



Yeung, et. al. End-to-end Learning of Action Detection from Frame Glimpses in Videos. CVPR 2017.

How do we model sequences?

How do we model sequences?

one-to-one
("normal" CNNs)

Inspired from Stanford CS231n

How do we model sequences?

one-to-one
("normal" CNNs)

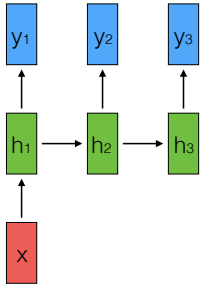
How do we model sequences?

one-to-one
("normal" CNNs)

How do we model sequences?

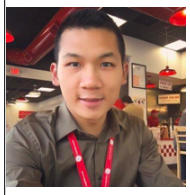
one-to-one
("normal" CNNs)

How do we model sequences?



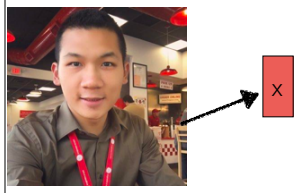
one-to-many
(e.g. image to sentence)

How do we model sequences?



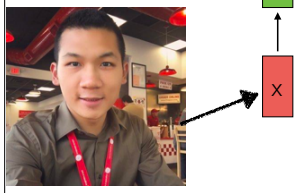
one-to-many

How do we model sequences?



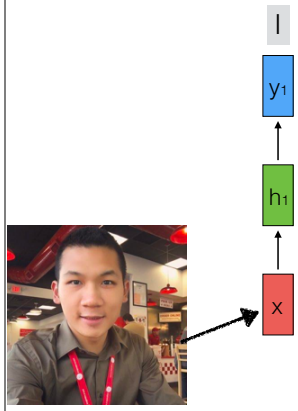
one-to-many

How do we model sequences?



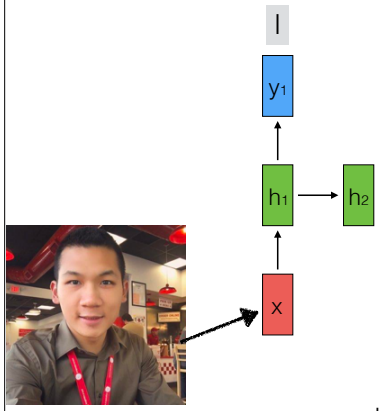
one-to-many

How do we model sequences?



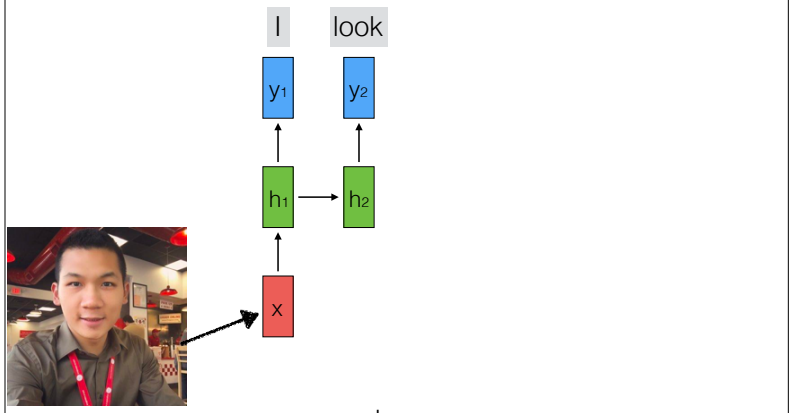
one-to-many

How do we model sequences?



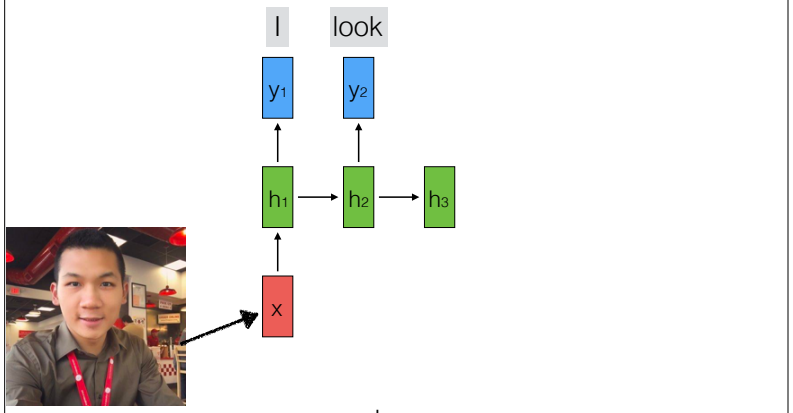
one-to-many

How do we model sequences?



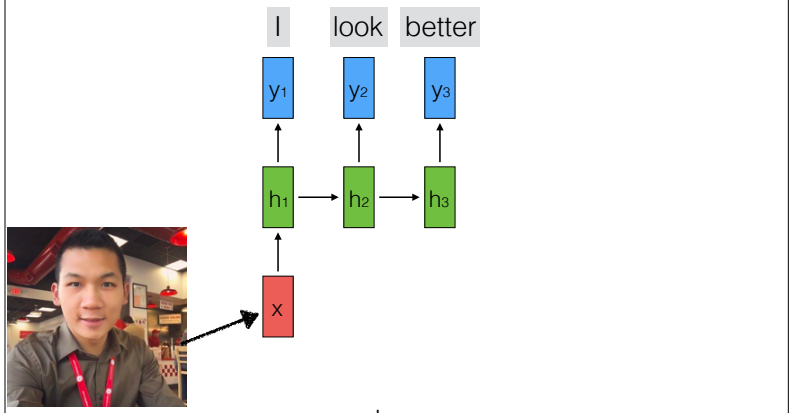
one-to-many

How do we model sequences?



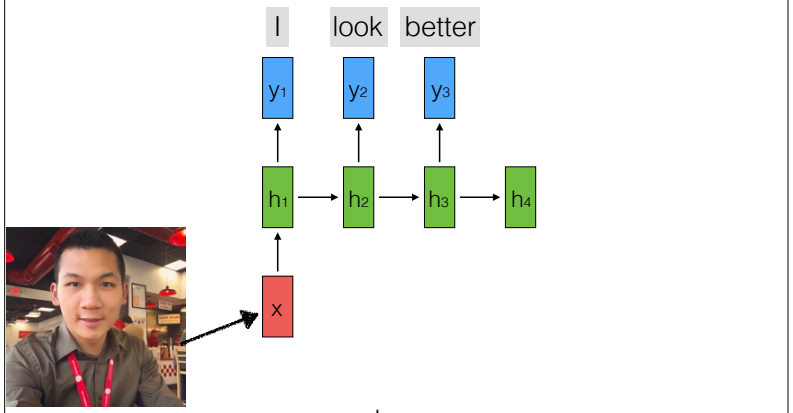
one-to-many

How do we model sequences?



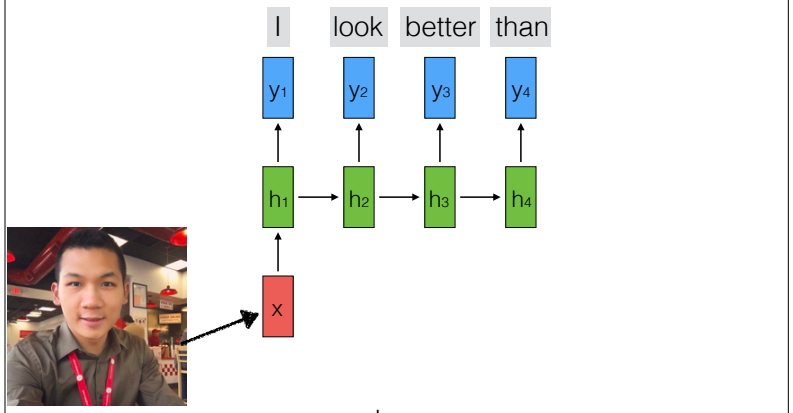
one-to-many

How do we model sequences?



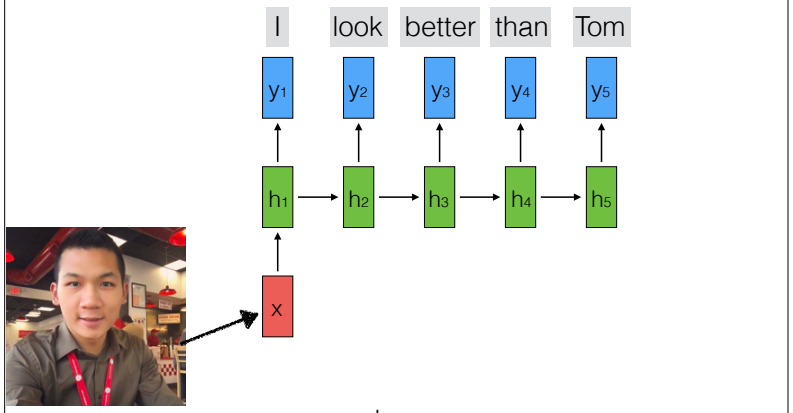
one-to-many

How do we model sequences?



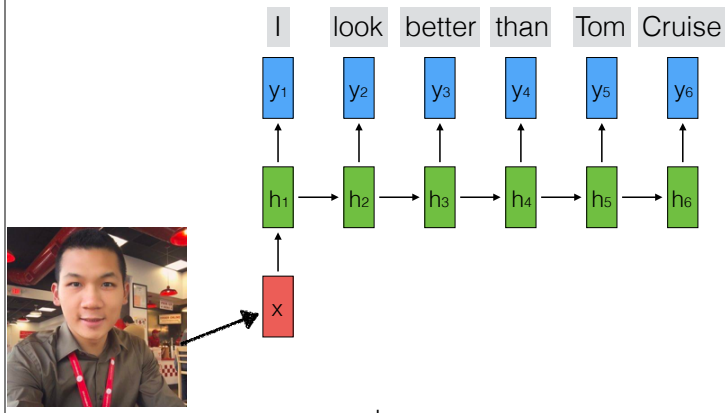
one-to-many

How do we model sequences?



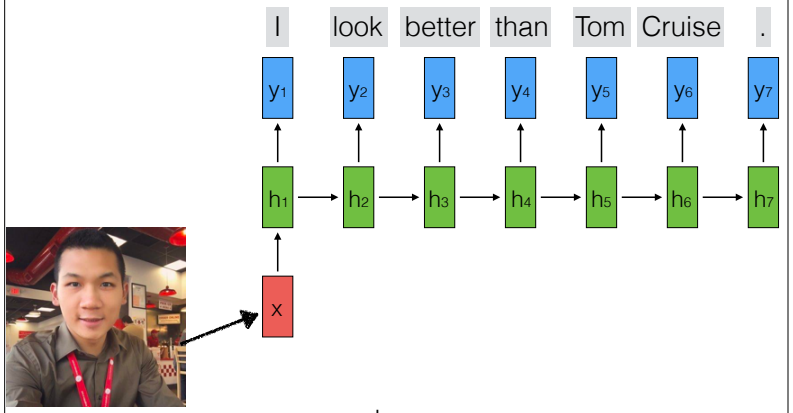
one-to-many

How do we model sequences?



one-to-many

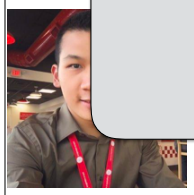
How do we model sequences?



one-to-many

How do we model sequences?

I look better than Tom Cruise .

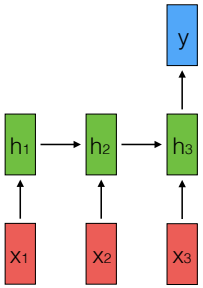


This is why a training data is important!

The model failed…

one-to-many

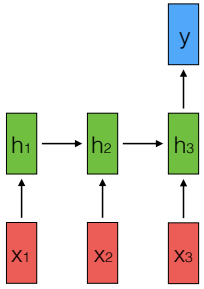
How do we model sequences?



many-to-one

How do we model sequences?

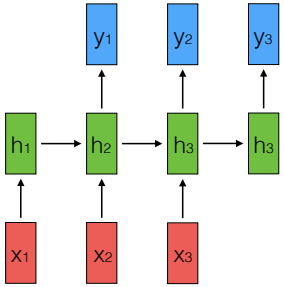
Wrong!



I can fly

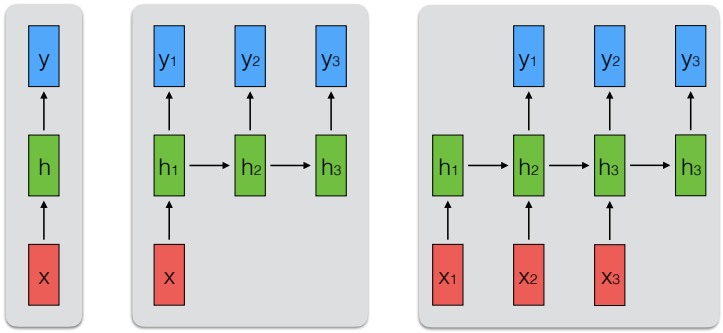
many-to-one

How do we model sequences?



many-to-many
(e.g. machine translation)

Different types of RNNs



Many more variations!

It can get tricky…

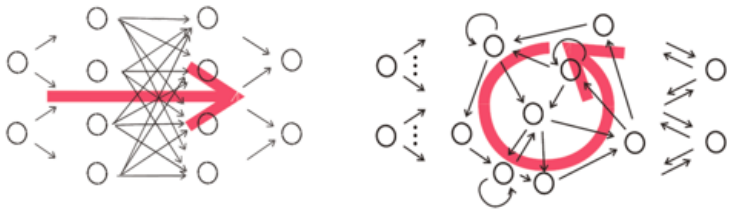
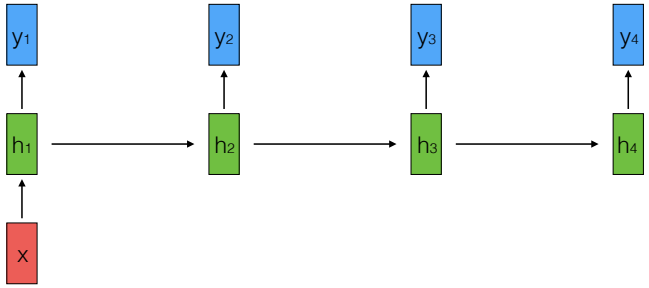
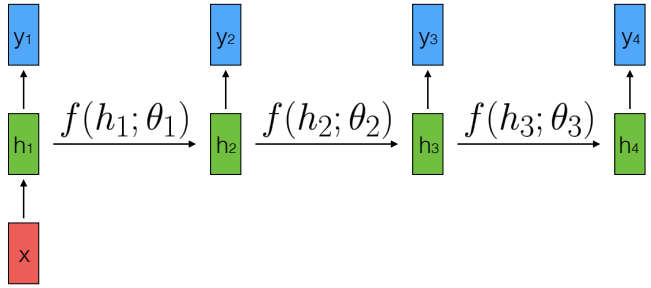


Image credit: Herbert Jaeger

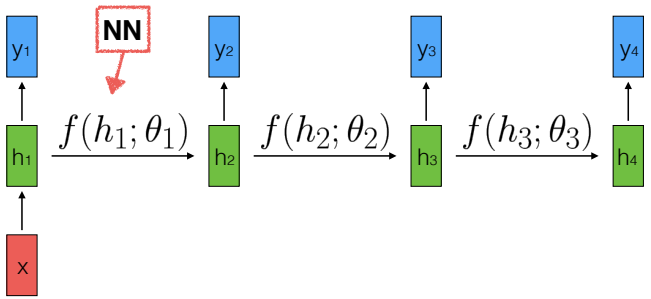
Let's zoom in!



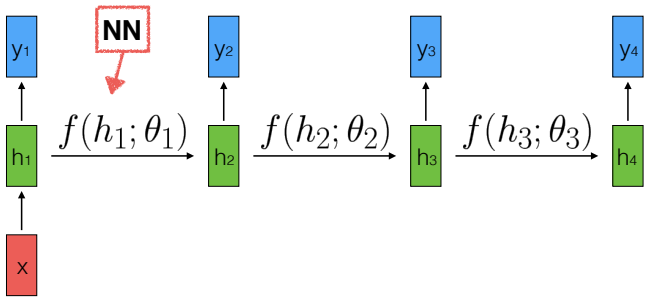
Let's zoom in!



Let's zoom in!



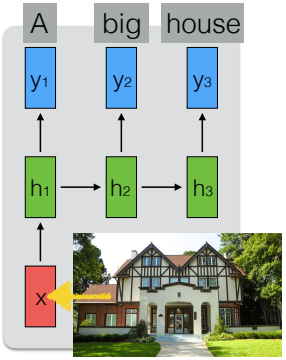
Let's zoom in!



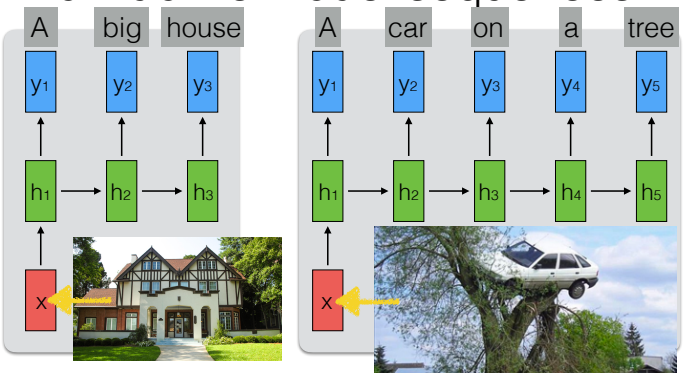
Any issue with this?

How do we model sequences?

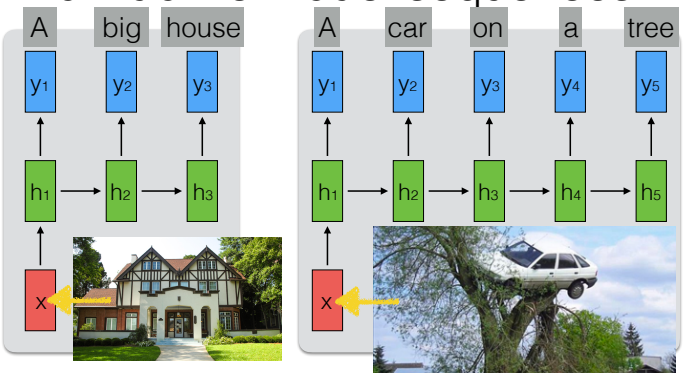
How do we model sequences?



How do we model sequences?

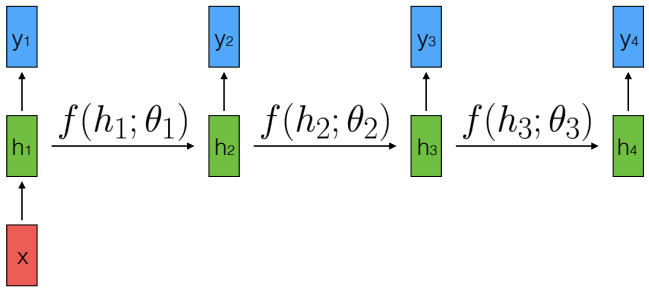


How do we model sequences?

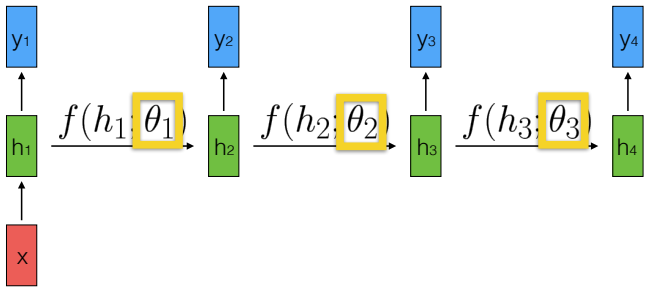


How to cover a variable input / output length?

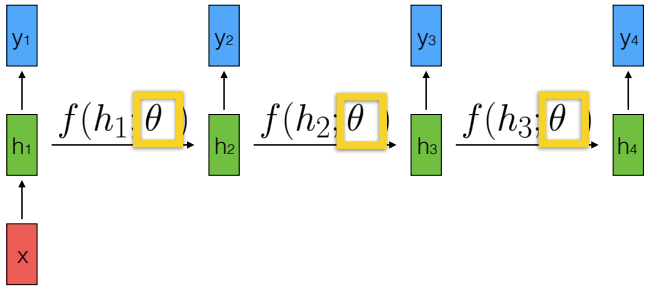
Recurrent Neural Network



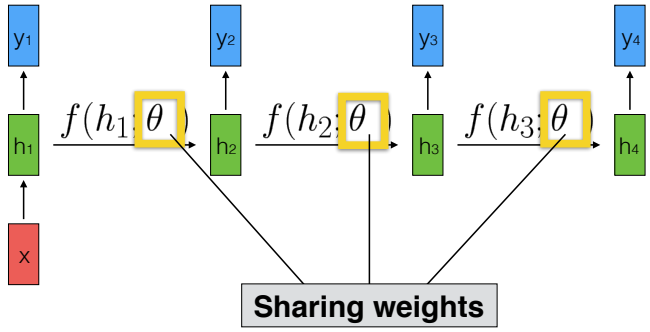
Recurrent Neural Network



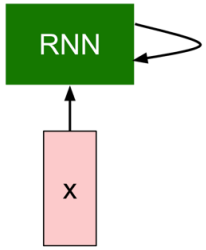
Recurrent Neural Network



Recurrent Neural Network

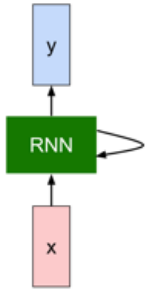


Recurrent Neural Network



Slide credit: Stanford CS231n

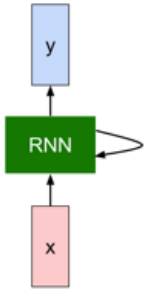
Recurrent Neural Network



Slide credit: Stanford CS231n

Recurrent Neural Network

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

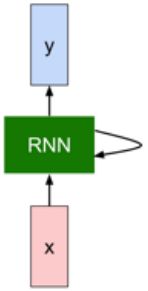


Slide credit: Stanford CS231n

Recurrent Neural Network

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

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



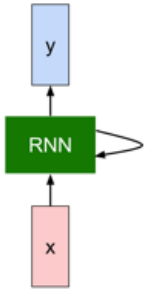
Slide credit: Stanford CS231n

Recurrent Neural Network

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

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

old state



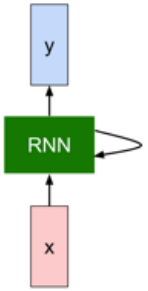
Slide credit: Stanford CS231n

Recurrent Neural Network

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

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

old state input vector at some time step



Slide credit: Stanford CS231n

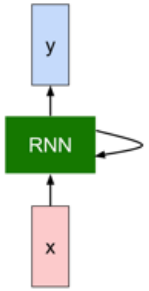
Recurrent Neural Network

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

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

old state — input vector at some time step

some function with parameters W



Slide credit: Stanford CS231n

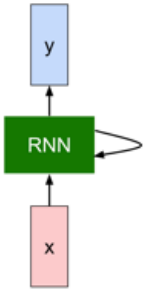
Recurrent Neural Network

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

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

new state — old state — input vector at some time step

some function with parameters W



Slide credit: Stanford CS231n

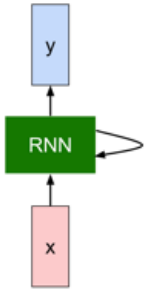
Recurrent Neural Network

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

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

new state — old state — input vector at some time step

some function with parameters W

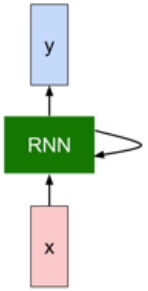


Note: the same function with the same parameters

Slide credit: Stanford CS231n

Recurrent Neural Network

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

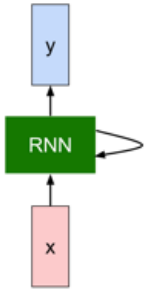


Slide credit: Stanford CS231n

(Vanilla) Recurrent Neural Network

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

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

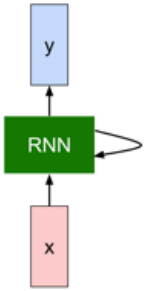


Slide credit: Stanford CS231n

(Vanilla) Recurrent Neural Network

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

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



The state has a single "hidden" vector **h**

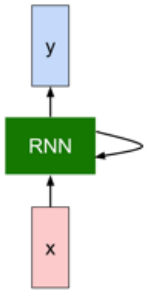
Slide credit: Stanford CS231n

(Vanilla) Recurrent Neural Network

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

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

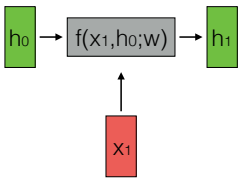
$$y_t = W_{hy}h_t$$



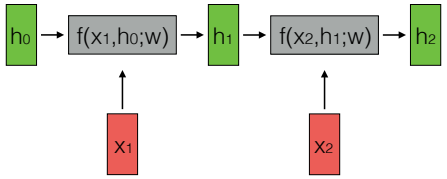
The state has a single "hidden" vector **h**

Slide credit: Stanford CS231n

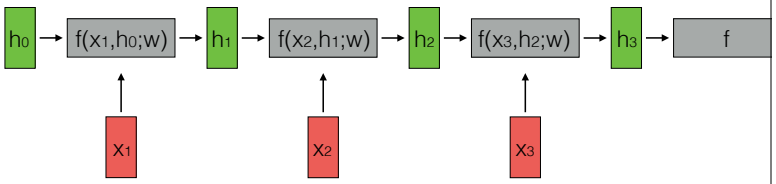
Computational Graph for RNN



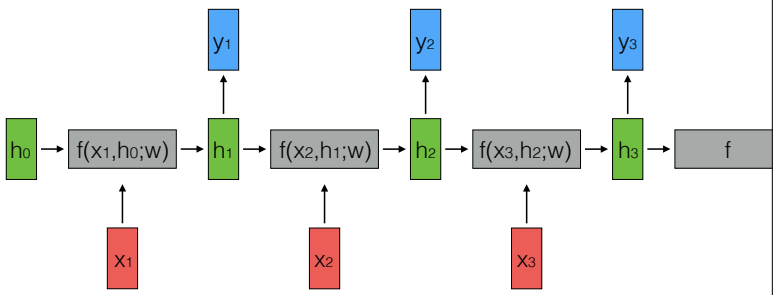
Computational Graph for RNN



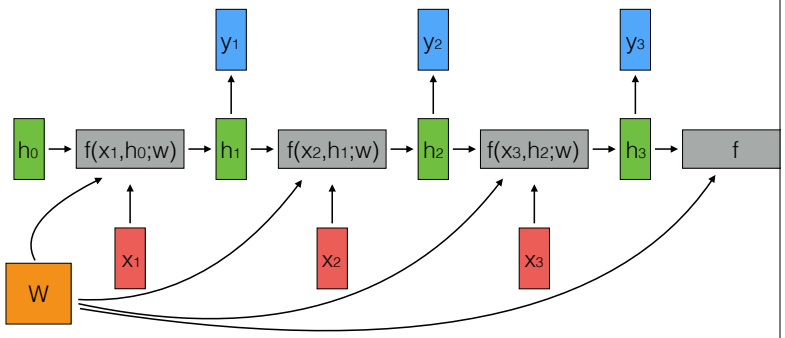
Computational Graph for RNN



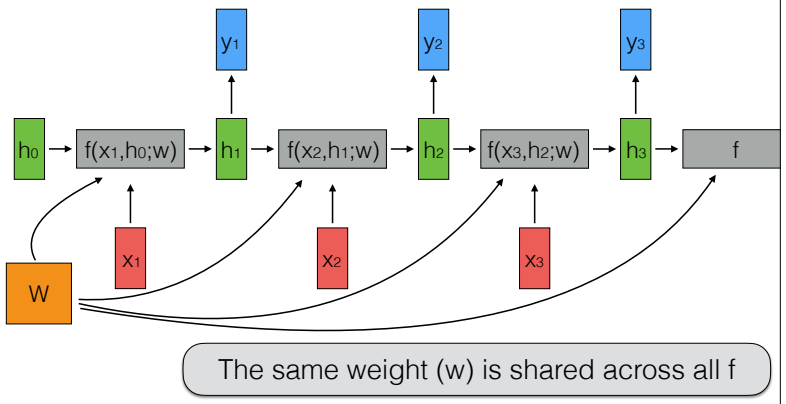
Computational Graph for RNN



Computational Graph for RNN

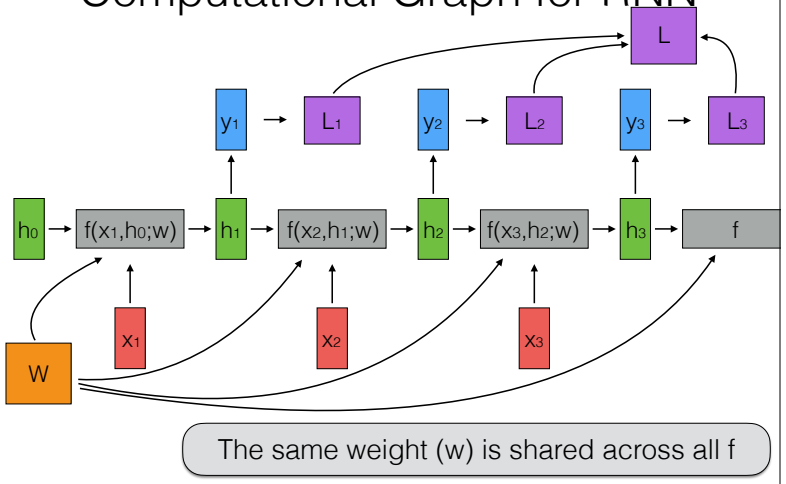


Computational Graph for RNN



The same weight (w) is shared across all f

Computational Graph for RNN



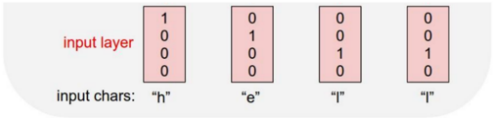
The same weight (w) is shared across all f

An Example

Character-level Language Model

Vocabulary:
[h, e, l, o]

Sample Training Sequence: “hello”



Credits: Stanford CS231n

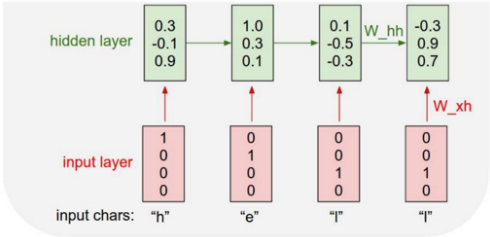
An Example

Character-level Language Model

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

Vocabulary:
[h, e, l, o]

Sample Training Sequence: “hello”



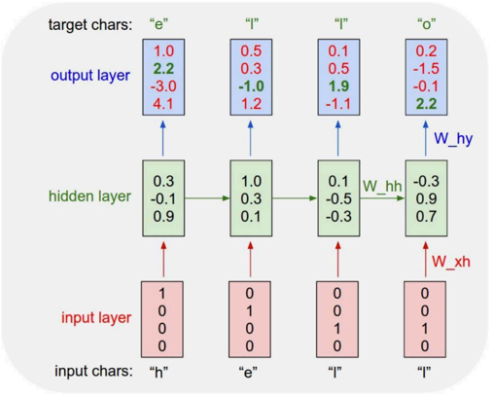
Credits: Stanford CS231n

An Example

Character-level Language Model

Vocabulary:
[h, e, l, o]

Sample Training Sequence: “hello”



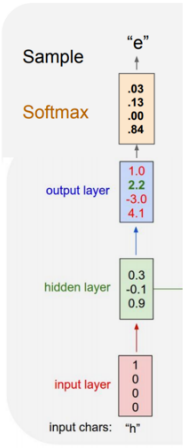
Credits: Stanford CS231n

An Example

Character-level Language Model

Vocabulary:
[h, e, l, o]

At **test-time** sample characters one at a time, and feed it back to the model



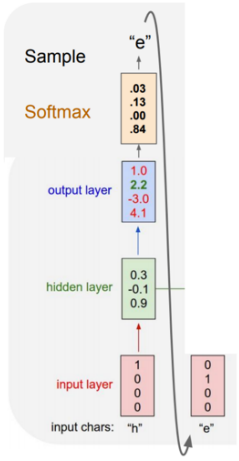
Credits: Stanford CS231n

An Example

Character-level Language Model

Vocabulary:
[h, e, l, o]

At **test-time** sample characters one at a time, and feed it back to the model



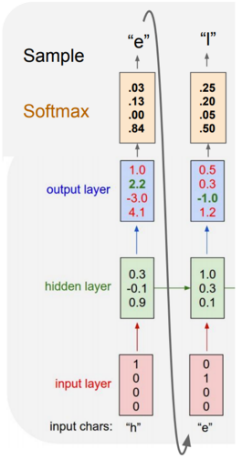
Credits: Stanford CS231n

An Example

Character-level Language Model

Vocabulary:
[h, e, l, o]

At **test-time** sample characters one at a time, and feed it back to the model



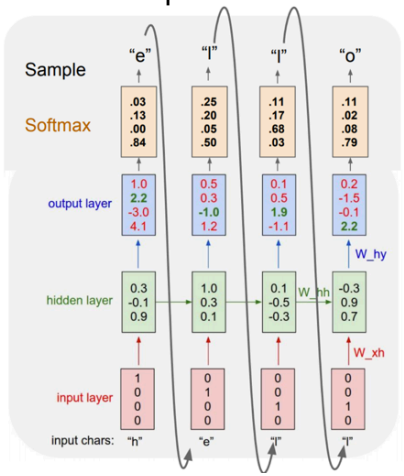
Credits: Stanford CS231n

An Example

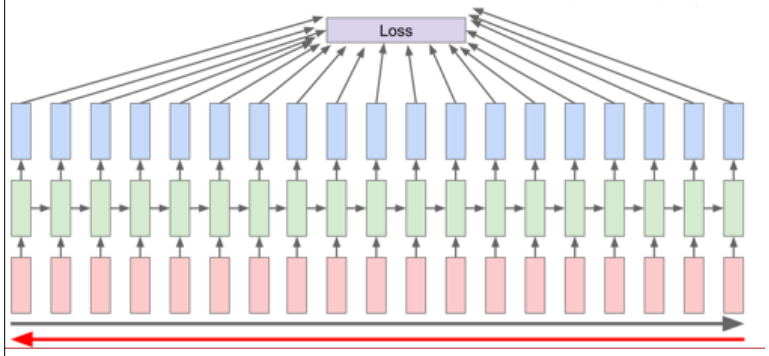
Character-level Language Model

Vocabulary:
[h, e, l, o]

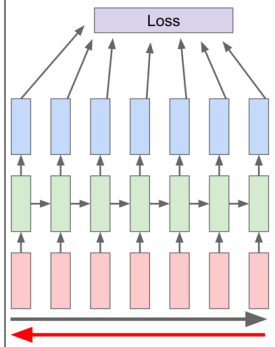
At **test-time** sample characters one at a time, and feed it back to the model



Backpropagation for RNN

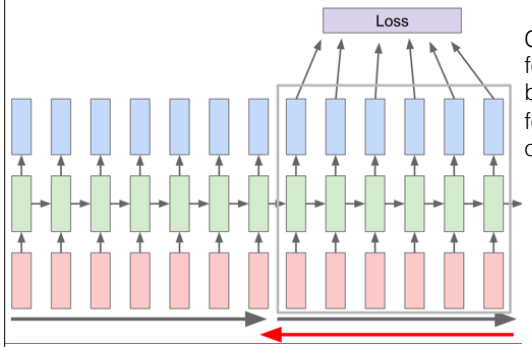


Truncated Backpropagation



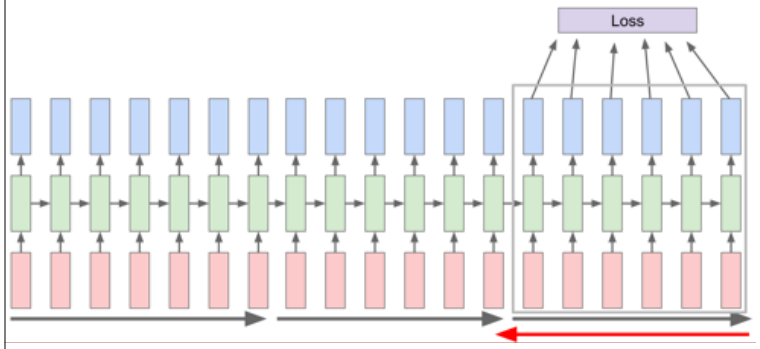
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation



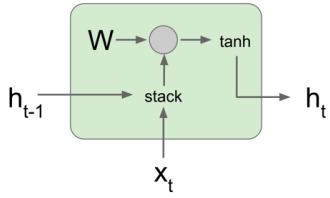
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation



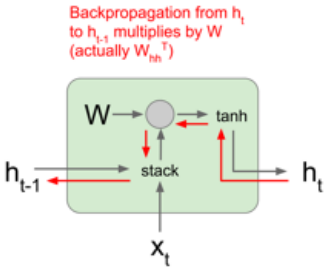
RNNs aren't as stable as CNNs

Vanilla RNN Gradient Flow



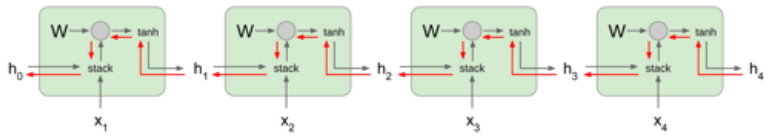
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix}h_{t-1}\\x_t\end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix}h_{t-1}\\x_t\end{pmatrix}\right)$$

Vanilla RNN Gradient Flow



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

Vanilla RNN Gradient Flow

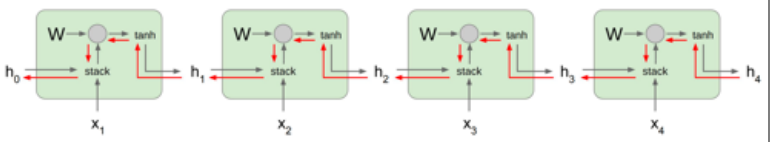


Computing gradient of h_0 involves many factors of W (and repeated tanh)

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

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

Vanilla RNN Gradient Flow



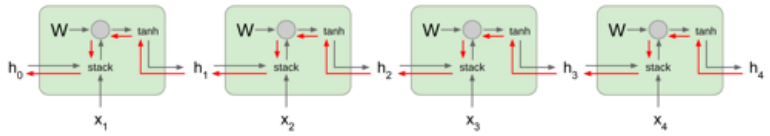
Computing gradient of h_0 involves many factors of W (and repeated tanh)

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

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

Gradient clipping: scale gradient if its norm is too big

Vanilla RNN Gradient Flow



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

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

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

Change RNN architecture

Long Short Term Memory (LSTM)

- Learn **long-term** dependencies
  - Remember information for long periods of time is practically their default behavior

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997
Colah's blog: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)

Vanilla RNN

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

LSTM

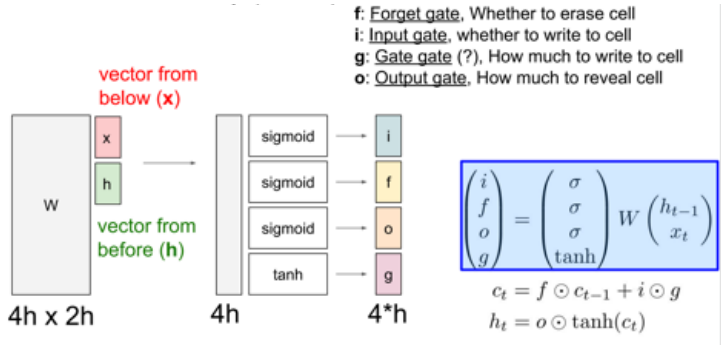
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997

Long Short Term Memory (LSTM)



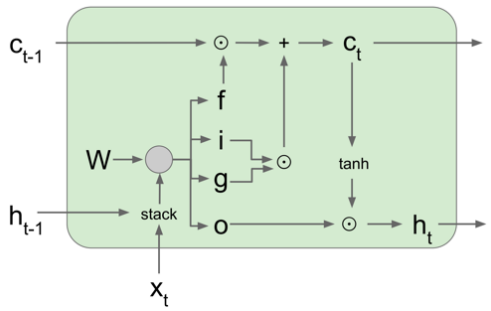
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997

Long Short Term Memory (LSTM)



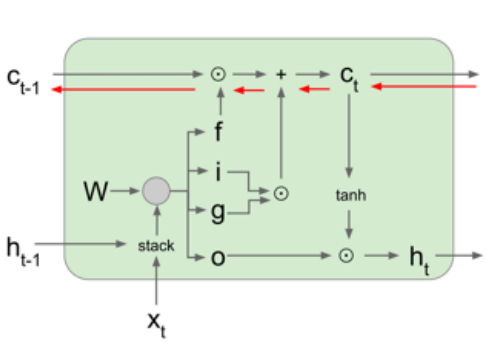
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997

Long Short Term Memory (LSTM)



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

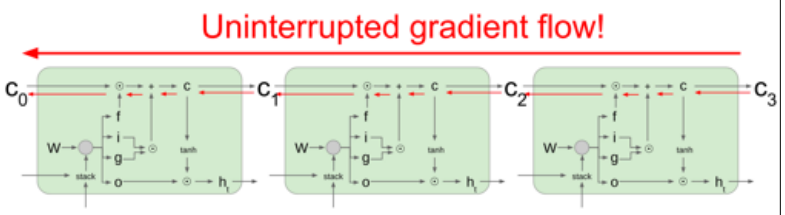
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997

Long Short Term Memory (LSTM)



Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation 1997

Other variants

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

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

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

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

MUT1:

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

MUT2:

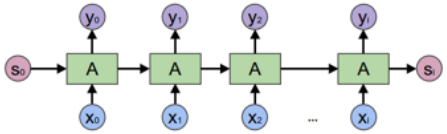
$$\begin{aligned} z &= \text{sigm}(W_{xz}x_t + W_{hz}h_t + b_z) \\ r &= \text{sigm}(x_t + W_{hr}h_t + b_r) \\ h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT3:

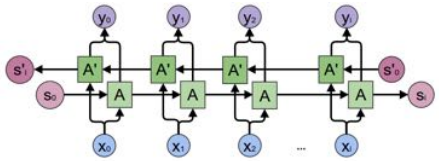
$$\begin{aligned} z &= \text{sigm}(W_{xz}x_t + W_{hz}\tanh(h_t) + b_z) \\ r &= \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r) \\ h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

Other variants

RNN



Bidirectional RNN



Make predictions over a sequence with both **past** and **future** context.

Colah's blog: http://colah.github.io/posts/2015-09-NN-Types-FP/

RNN applications

- Image Captioning
- Image Captioning with Attention
- Question Answering
- Visual Question Answering
- Speech Recognition
- Action Recognition in Videos
- Text Parsing
- Machine Translation

Image Captioning

I look better than Tom Cruise.

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Image Captioning

Image Captioning

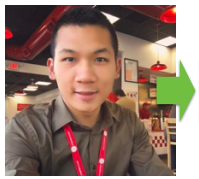
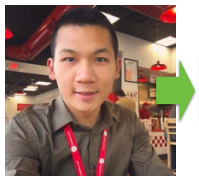


Image Captioning



0

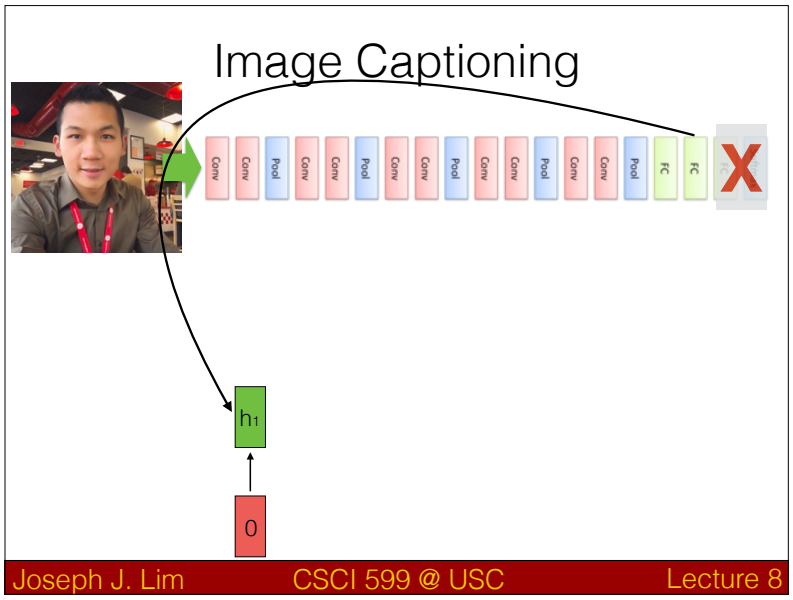


Image Captioning
Conv
Pool
FC
X
h1
0
Joseph J. Lim
CSCI 599 @ USC
Lecture 8

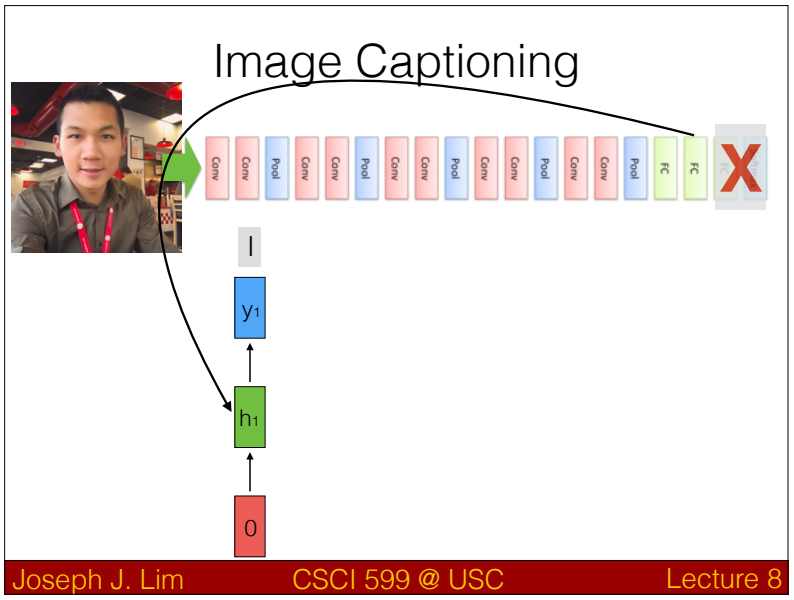


Image Captioning
Conv
Pool
FC
X
I
y1
h1
0
Joseph J. Lim
CSCI 599 @ USC
Lecture 8

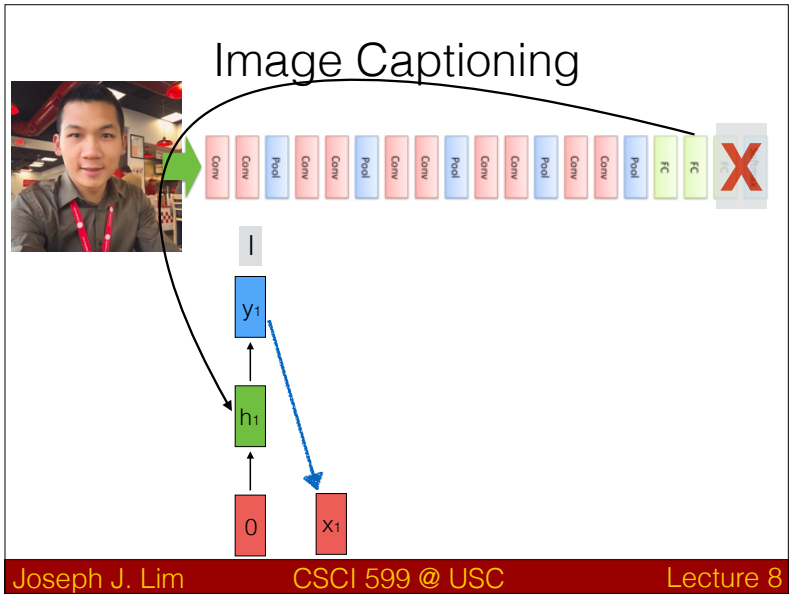


Image Captioning
Conv
Pool
FC
X
I
y1
h1
0
x1
Joseph J. Lim
CSCI 599 @ USC
Lecture 8

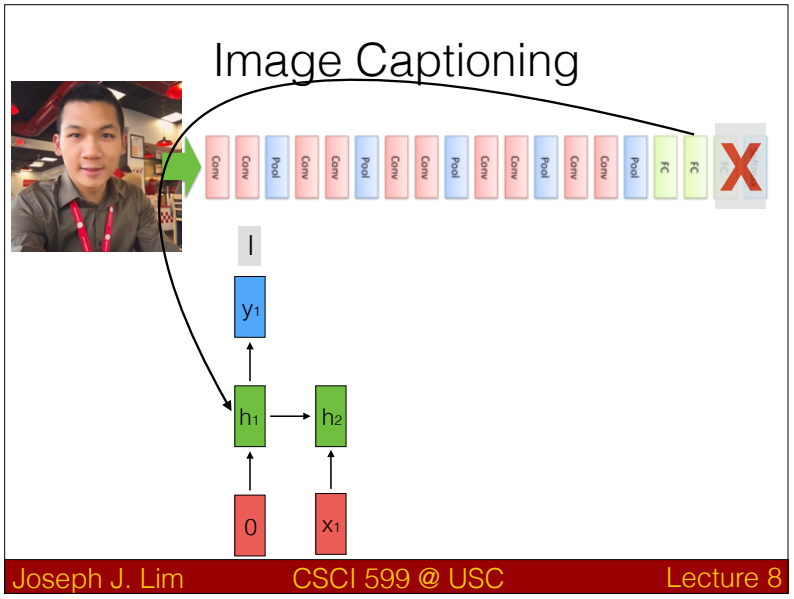


Image Captioning
Conv
Pool
FC
X
I
y1
h1
h2
0
x1
Joseph J. Lim
CSCI 599 @ USC
Lecture 8

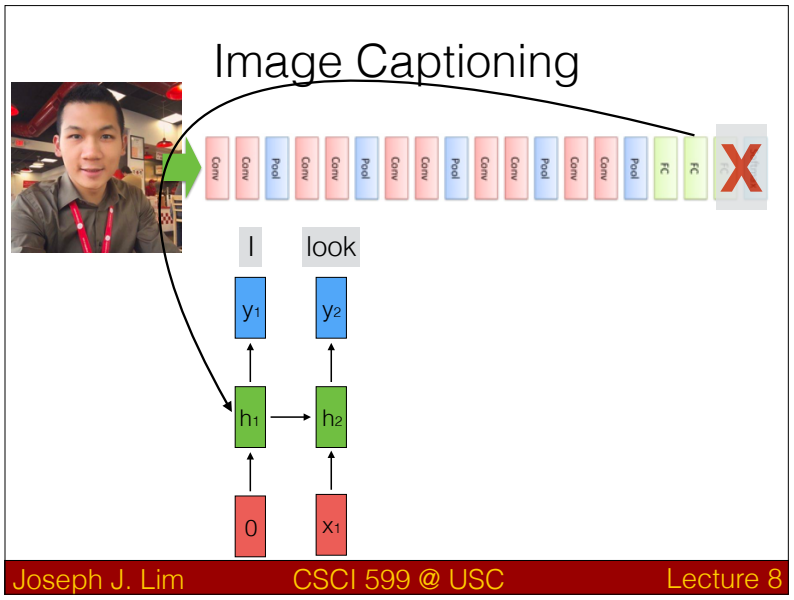


Image Captioning
Conv
Pool
FC
I
look
y1
y2
h1
h2
0
x1
Joseph J. Lim
CSCI 599 @ USC
Lecture 8

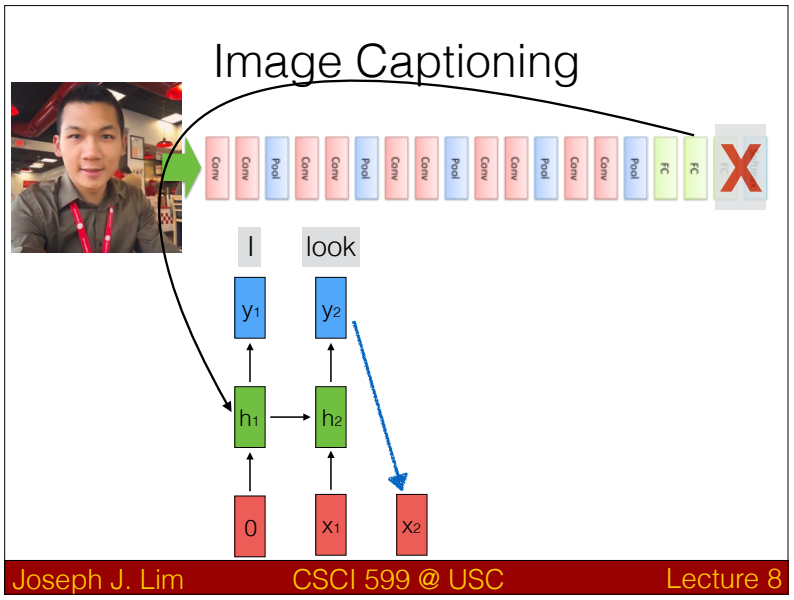


Image Captioning
Conv
Pool
FC
I
look
y1
y2
h1
h2
0
x1
x2
Joseph J. Lim
CSCI 599 @ USC
Lecture 8

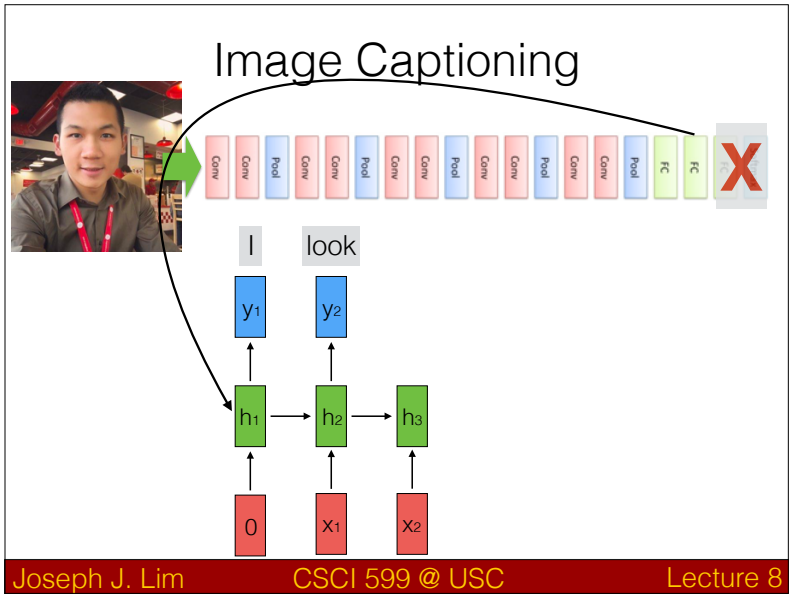


Image Captioning
Conv
Pool
FC
I
look
y1
y2
h1
h2
h3
0
x1
x2
Joseph J. Lim
CSCI 599 @ USC
Lecture 8

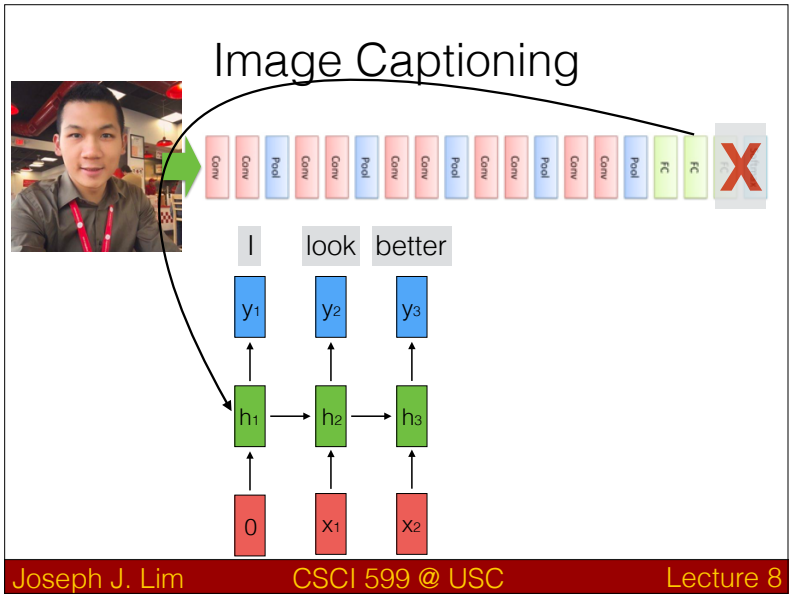


Image Captioning
Conv
Pool
FC
I
look
better
y1
y2
y3
h1
h2
h3
0
x1
x2
Joseph J. Lim
CSCI 599 @ USC
Lecture 8

Image Captioning

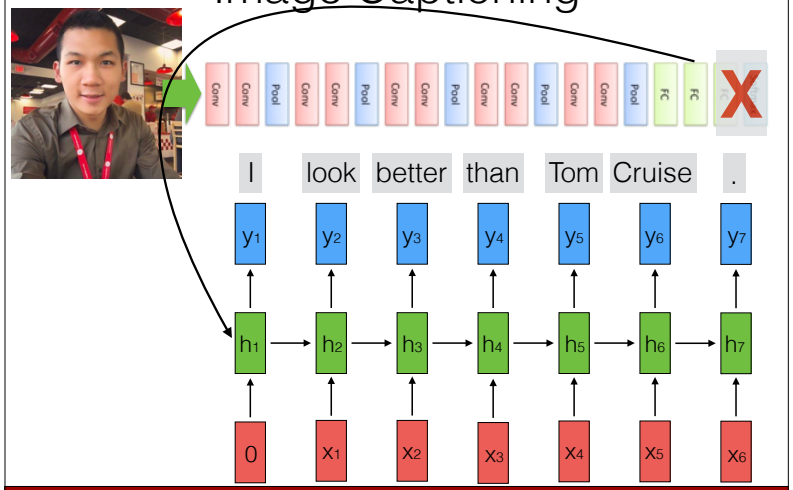
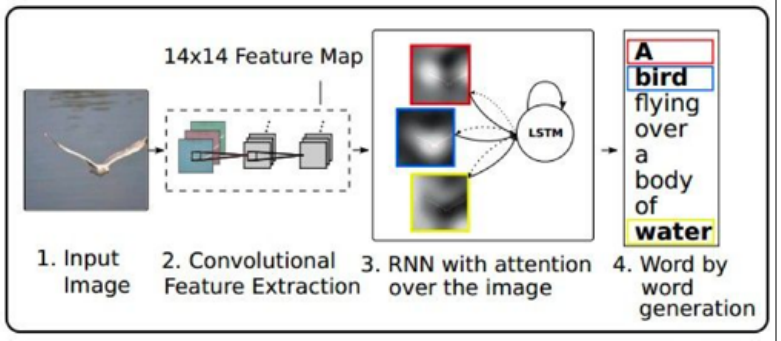


Image Captioning: Examples

Image Captioning: Failures

Image Captioning with Attention



Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015.

Image Captioning with Attention

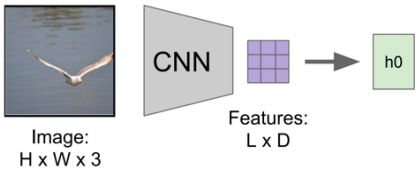


Image Captioning with Attention

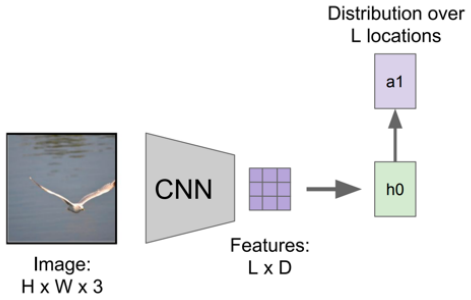


Image Captioning with Attention

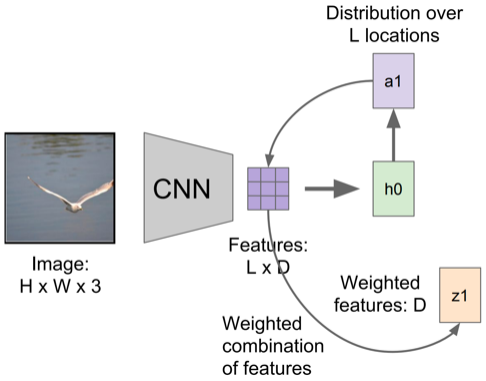


Image Captioning with Attention

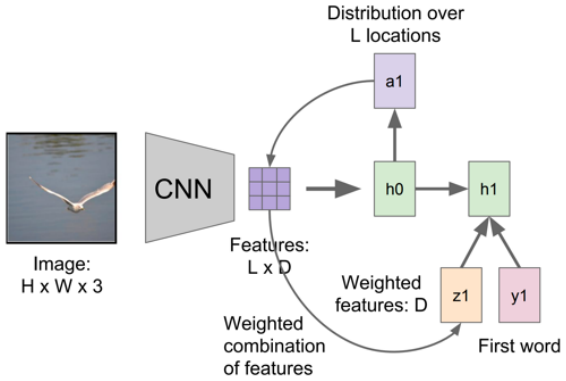


Image Captioning with Attention

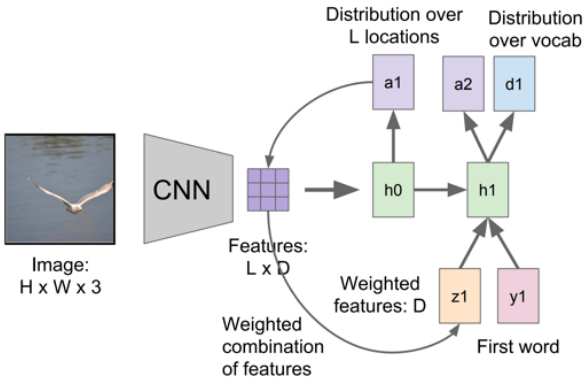


Image Captioning with Attention

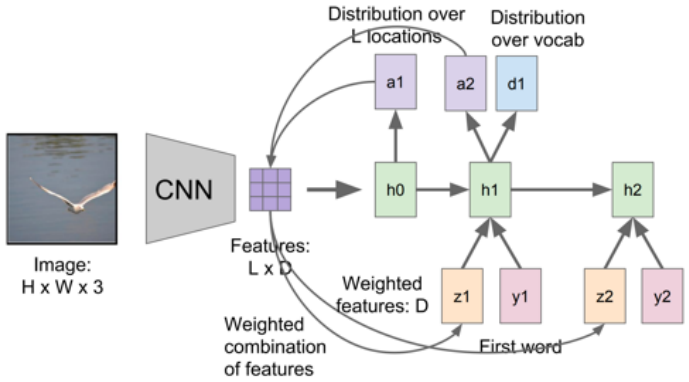


Image Captioning with Attention

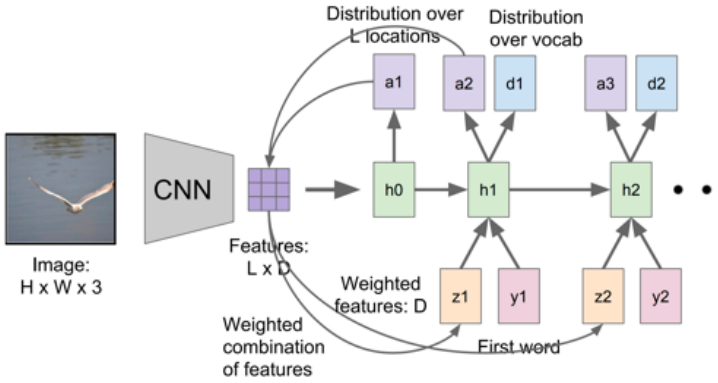
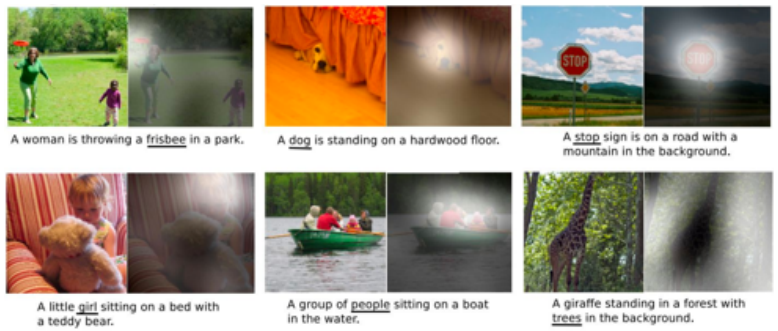
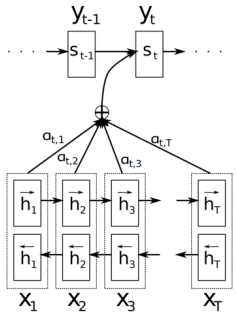


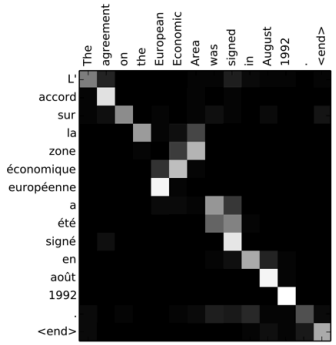
Image Captioning with Attention



Machine Translation with Attention



Self-Attention Mechanism
a: the soft attention coefficients



Visualizing a
sample attention

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

Question Answering

Mary moved to the bathroom.
John went to the hallway.

Where is Mary? bathroom

https://allenai.github.io/bi-att-flow/demo/
Dialogue Learning With Human-in-the-loop, Jiwei et al.

Question Answering

Text

Mary moved to the bathroom.
John went to the hallway.

Question

Where is Mary?

Answer

bathroom

https://allenai.github.io/bi-att-flow/demo/
Dialogue Learning With Human-in-the-loop, Jiwei et al.

Question Answering

Text

Mary moved to the bathroom.
John went to the hallway.

T Text embedding

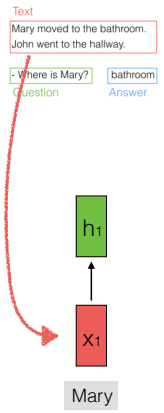
Question

- Where is Mary?

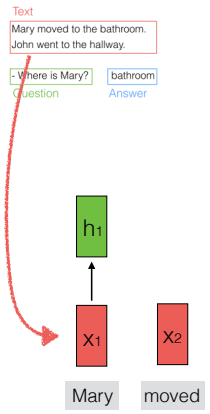
Q Question embedding

Question Answering

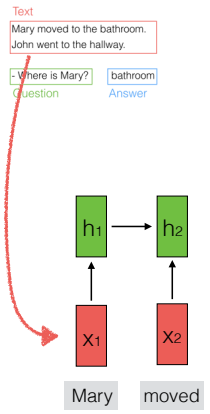
Question Answering



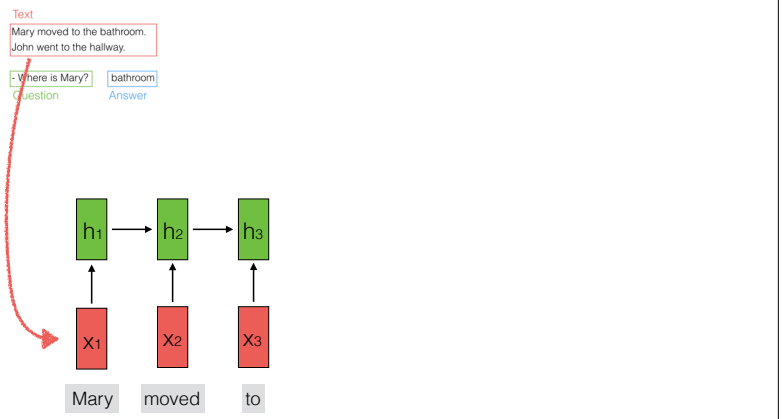
Question Answering



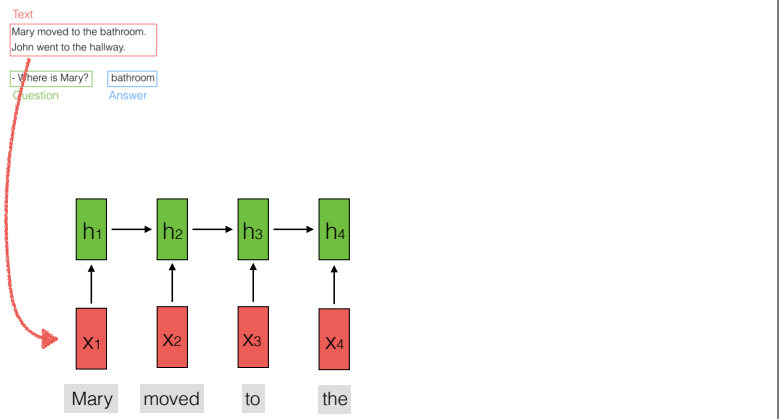
Question Answering



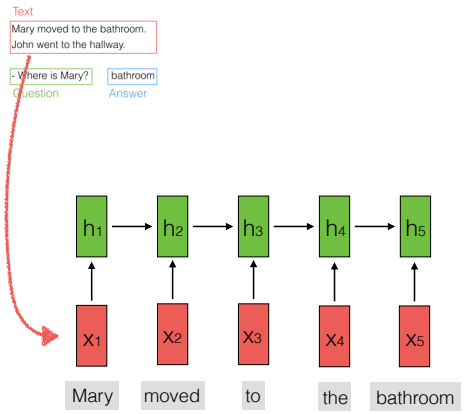
Question Answering



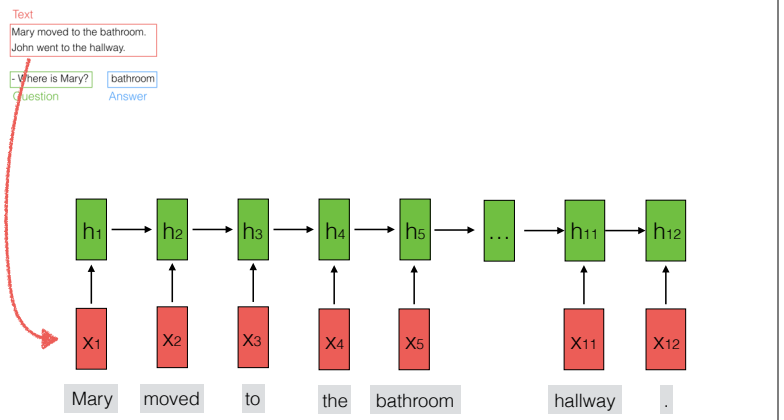
Question Answering



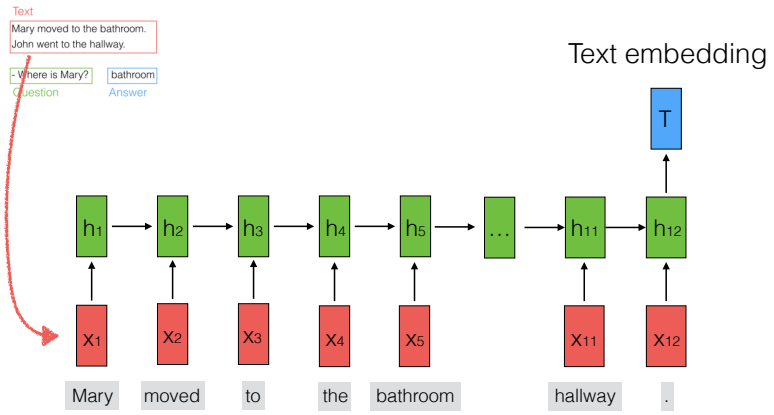
Question Answering



Question Answering

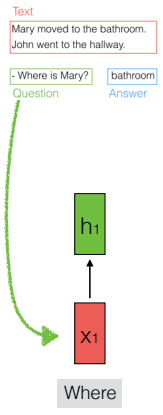


Question Answering

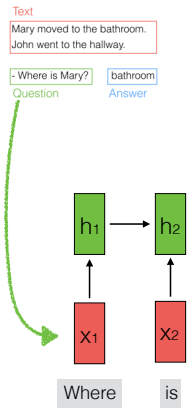


Question Answering

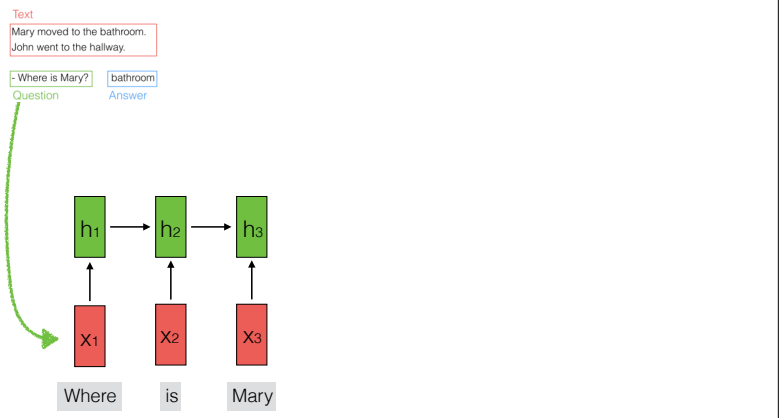
Question Answering



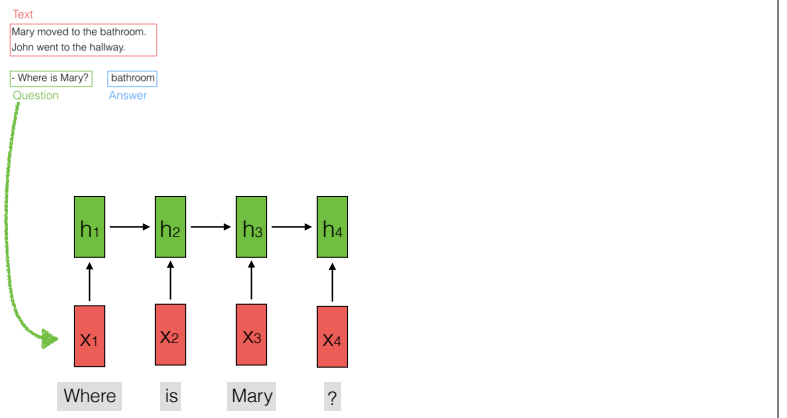
Question Answering



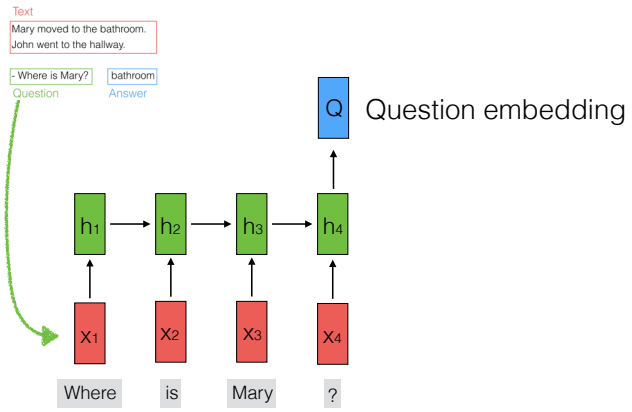
Question Answering



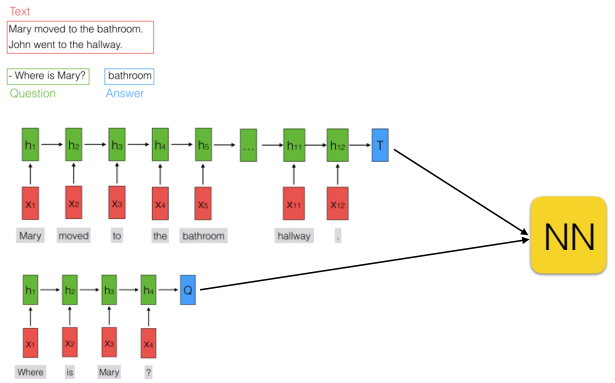
Question Answering



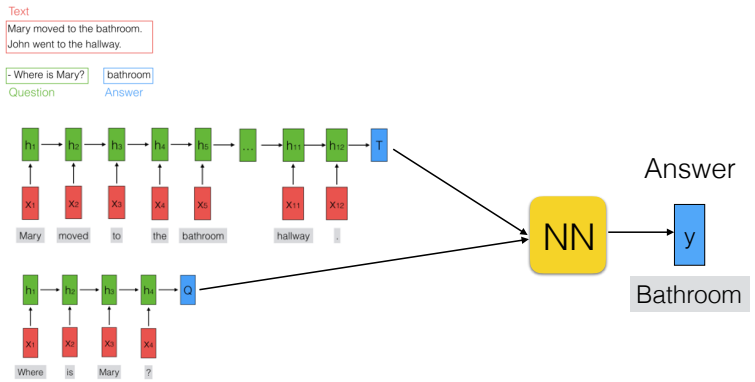
Question Answering



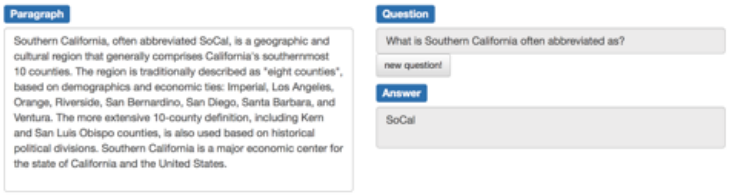
Question Answering



Question Answering



DEMO



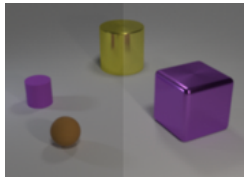
- https://allenai.github.io/bi-att-flow/demo/

Visual Question Answering

VQA: Visual Question Answering, Agrawal et al
A simple neural network module for relational reasoning, Santoro et al

Visual Question Answering

Image



Question

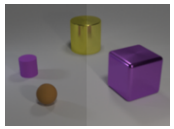
What is the size of the brown sphere?

Answer

Small

Visual Question Answering

Image

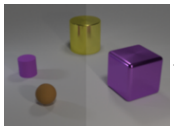


Question

What is the size of the brown sphere?

Visual Question Answering

Image

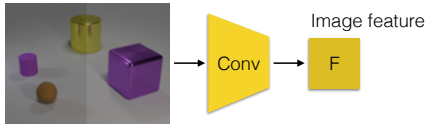


Question

What is the size of the brown sphere?

Visual Question Answering

Image



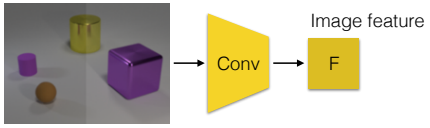
Question

What is the size of the brown sphere?

What

Visual Question Answering

Image

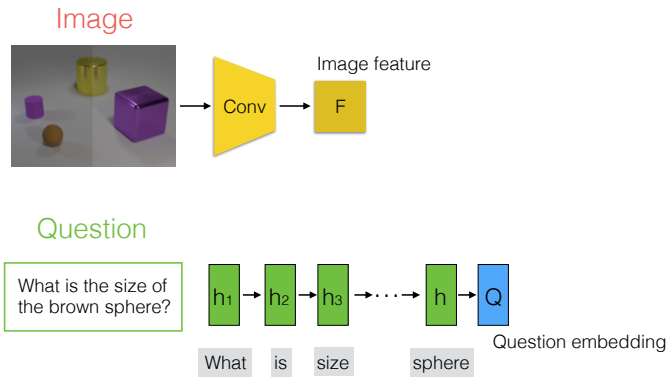


Question

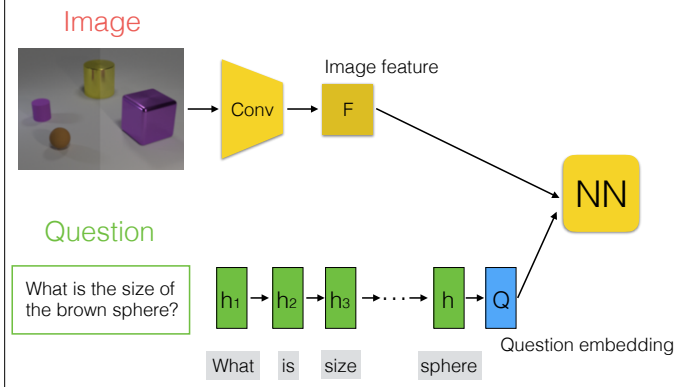
What is the size of the brown sphere?

What is

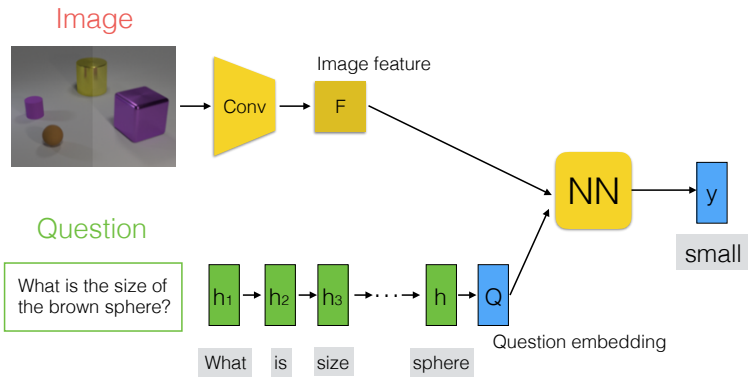
Visual Question Answering



Visual Question Answering



Visual Question Answering



Summary

- RNNs deal with various sequential data or dependencies.
- LSTM or GRU are go-to models (rather than vanilla RNNs)
- RNNs are still under development (relative to CNNs)

Congratulations!

All basic vocabs are covered.

Next week

- Deep Learning Software by TAs

Todo

- Assignment #1 due 2/19/2019
- Project proposal due 2/26/2019
  - 1 page summary
  - N slides describing your project (for 70 seconds)
- Midterm date changed to 2/26/2019

Questions?