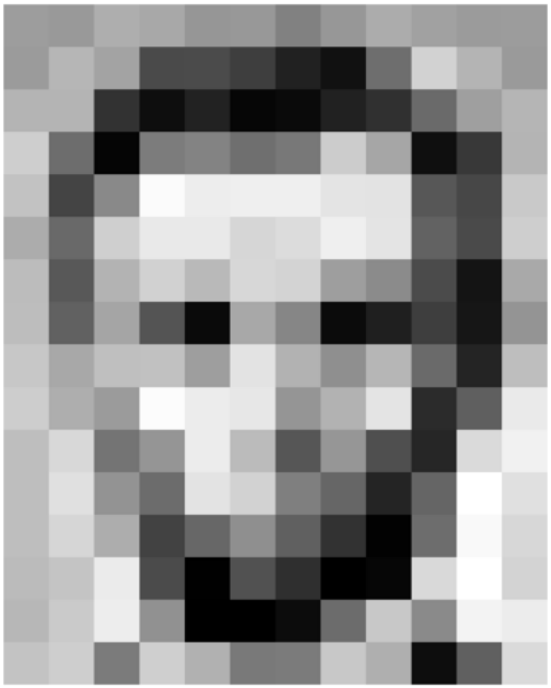
Lecture 10 (CNN,RNN)


CNN

What computers ‘see’: Images as Numbers

What you see



Input Image

What you both see

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Input Image + values

What the computer "sees"

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Pixel intensity values
(“pix-el”=picture-element)

Levin *Image Processing & Computer Vision*

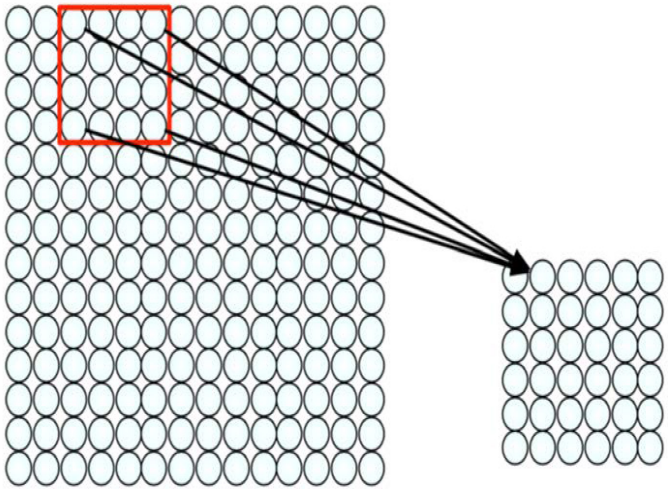
An image is just a matrix of numbers [0,255]. i.e., 1080x1080x3 for an RGB image.

Can I just do classification on the 1,166400-long image vector directly?

No. Instead: exploit image spatial structure. Learn patches. Build them up

This slide is taken from Manolis Kellis

Feature Extraction with Convolution



- Filter of size 4x4 : 16 different weights
- Apply this same filter to 4x4 patches in input
- Shift by 2 pixels for next patch

This "patchy" operation is **convolution**

1) Apply a set of weights – a filter – to extract **local features**

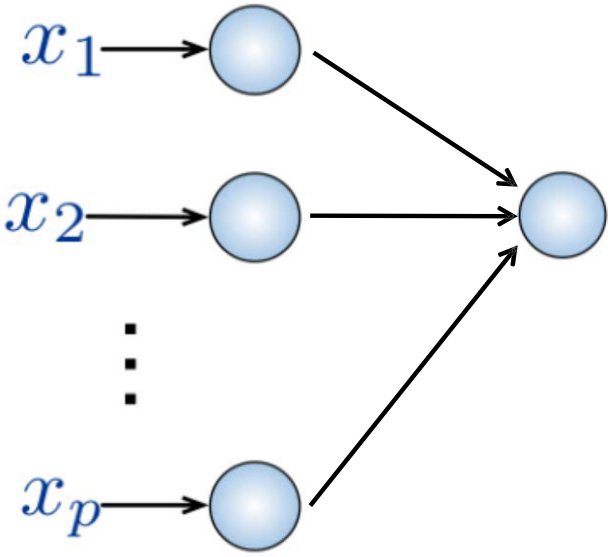
2) Use **multiple filters** to extract different features

3) Spatially share parameters of each filter

This slide is taken from Manolis Kellis

Fully Connected Neural Network

Input:
- 2D image
- Vector of pixel values

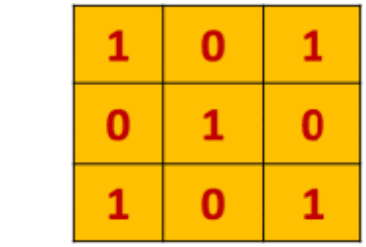


Fully Connected:
- Each neuron in hidden layer connected to all neurons in input layer
- No spatial information
- Many, many parameters

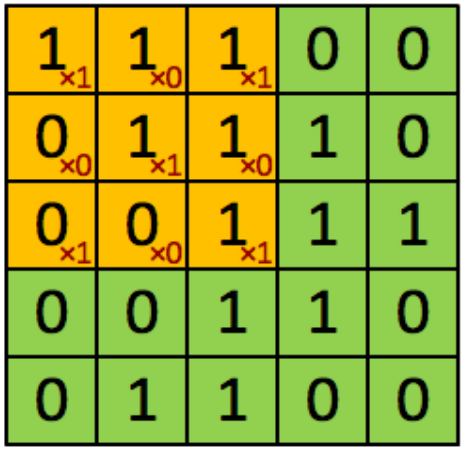
Key idea: Use **spatial structure** in input to inform architecture of the network

This slide is taken from Manolis Kellis

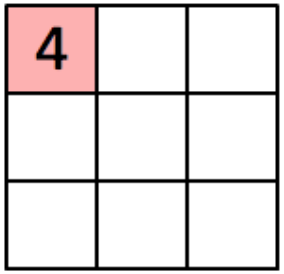
Convolution operation is element wise multiply and add



Filter / Kernel



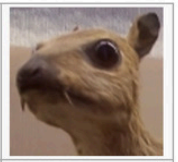
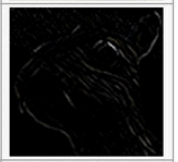
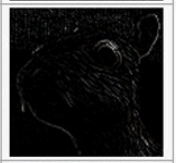
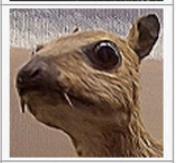
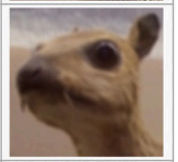
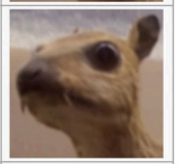
Image



Convolved Feature

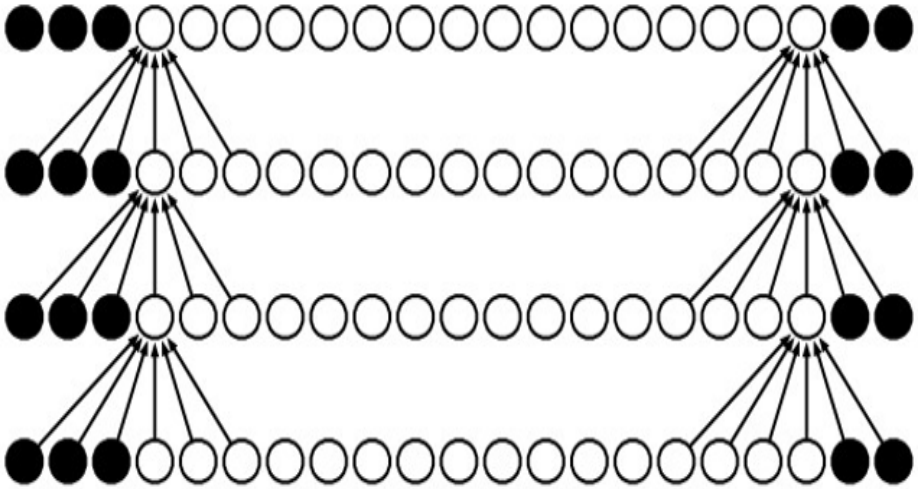
This slide is taken from Manolis Kellis

Simple Kernels / Filters

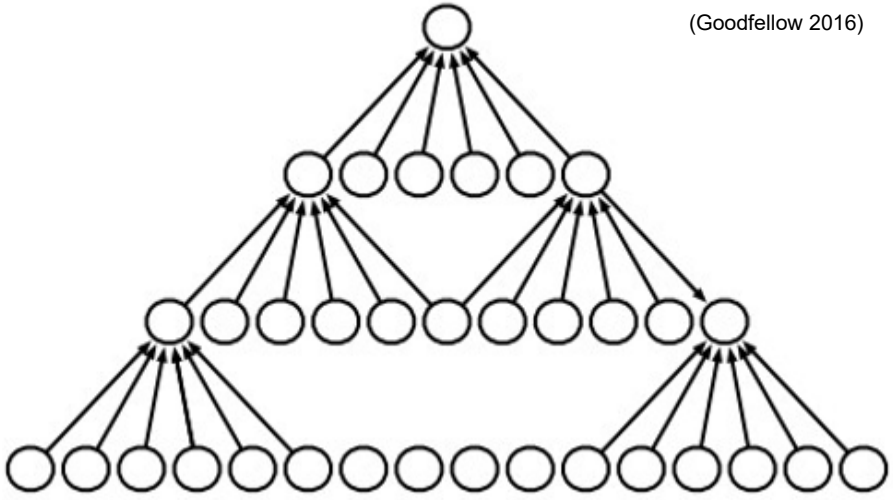
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

This slide is taken from Manolis Kellis

Zero Padding Controls Output Size



(Goodfellow 2016)



• **Same convolution**: zero pad input so output is same size as input dimensions

• **Valid-only convolution**: output only when entire kernel contained in input (shrinks output)

This slide is taken from Manolis Kellis

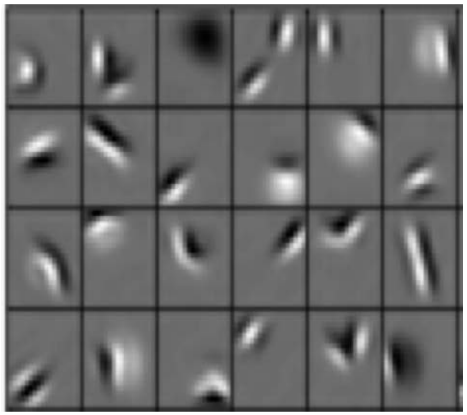
Key idea:

learn hierarchy of features

directly from the data

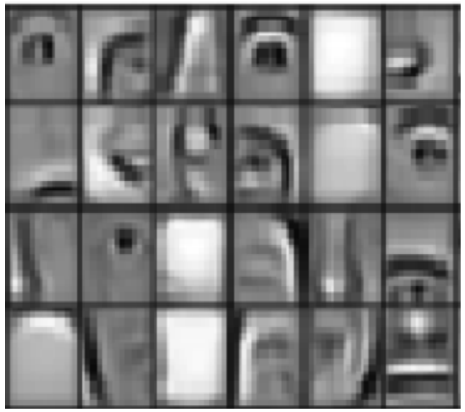
(rather than hand-engineering them)

Low level features



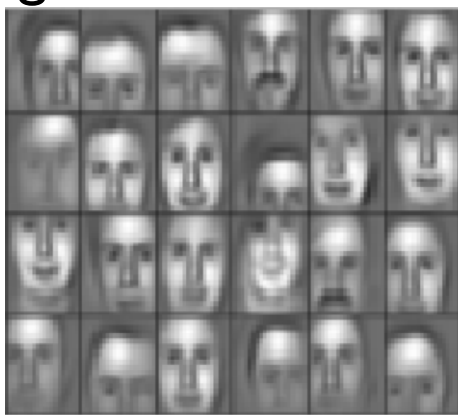
Edges, dark spots

Mid level features



Eyes, ears,nose

High level features



Facial structure

This slide is taken from Manolis Kellis

Lee+ *ICML* 2009

LeNet-5

- *Gradient Based Learning Applied To Document Recognition - Y. Lecun, L. Bottou, Y. Bengio, P. Haffner; 1998*
- Helped establish how we use CNNs today
- Replaced manual feature extraction

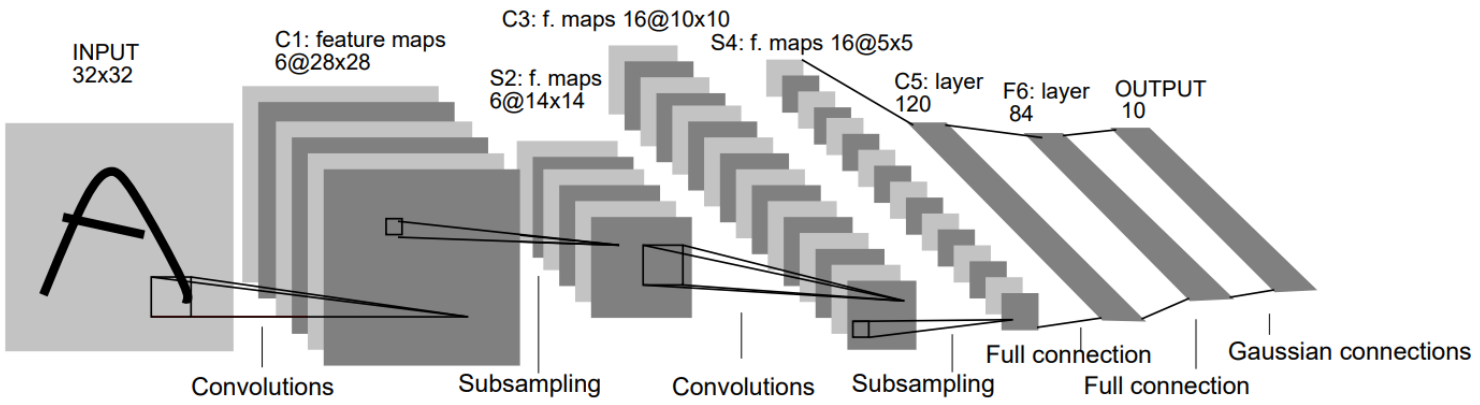
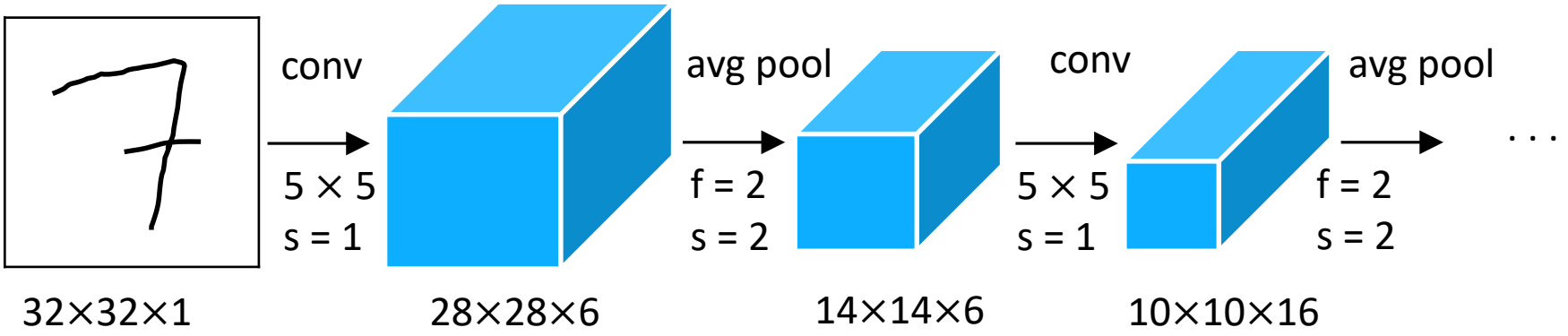


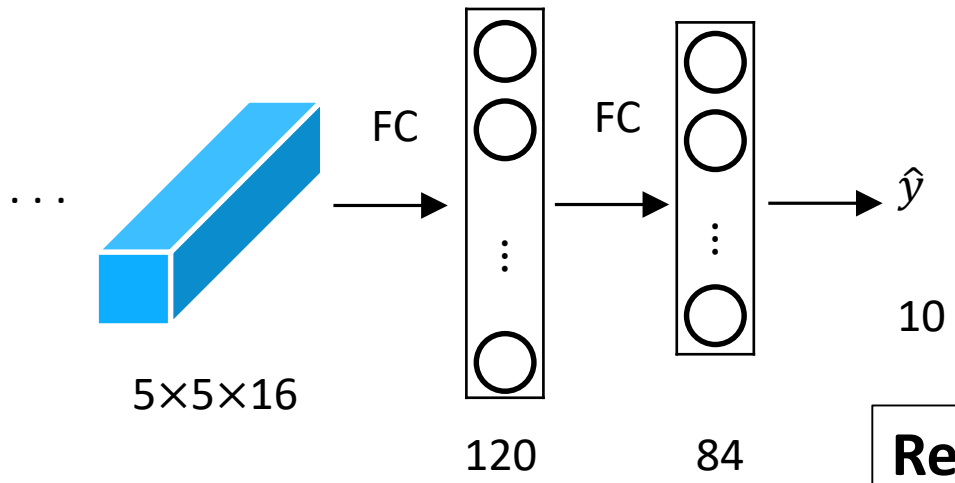
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

This slide is taken from Manolis Kellis

[LeCun et al., 1998]

LeNet-5



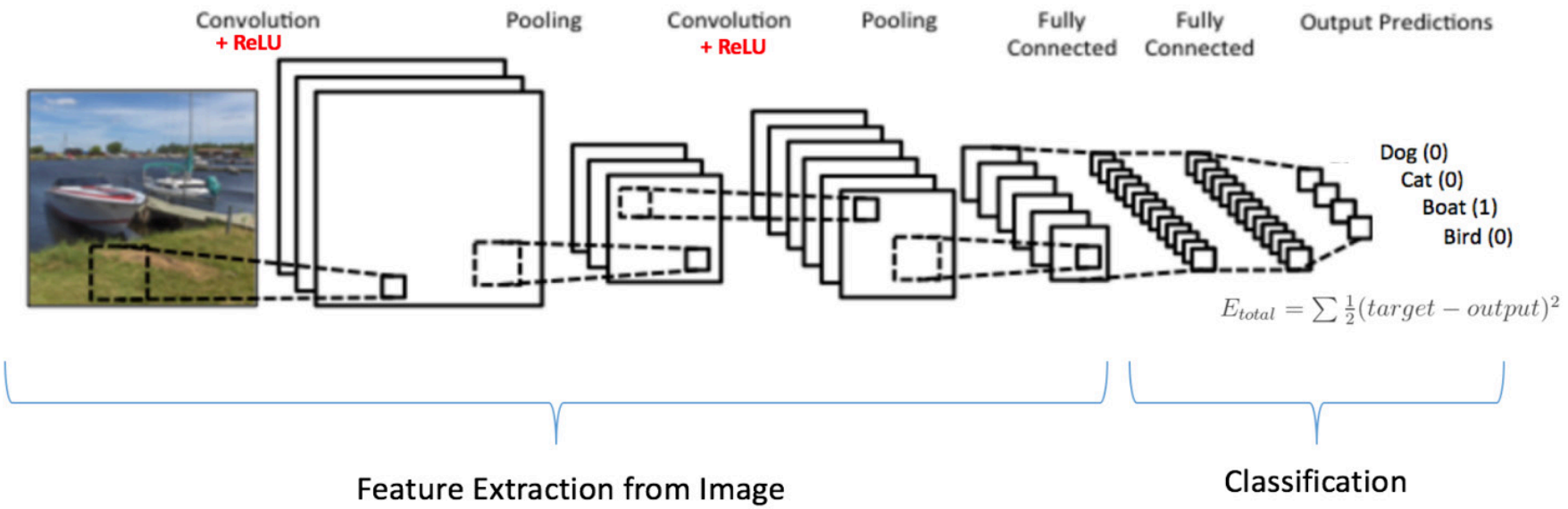


Reminder:
Output size = (N+2P-F)/stride + 1

This slide is taken from Andrew Ng

[LeCun et al., 1998]

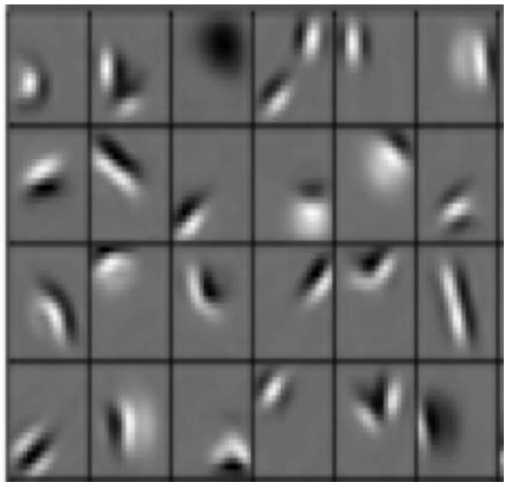
An image classification CNN



This slide is taken from Manolis Kellis

Representation Learning in Deep CNNs

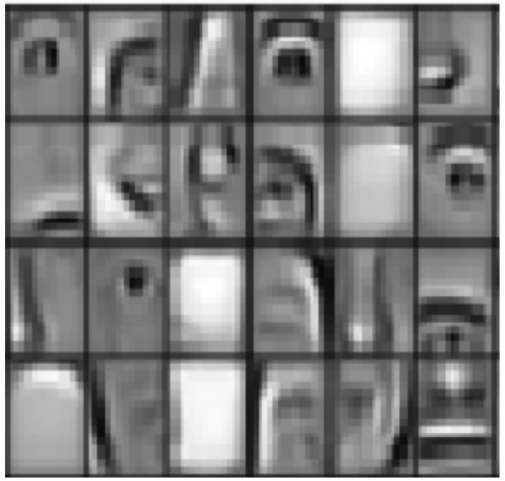
Low level features



Edges, dark spots

Conv Layer 1

Mid level features



Eyes, ears,nose

Conv Layer 2

High level features

Facial structure

Conv Layer 3

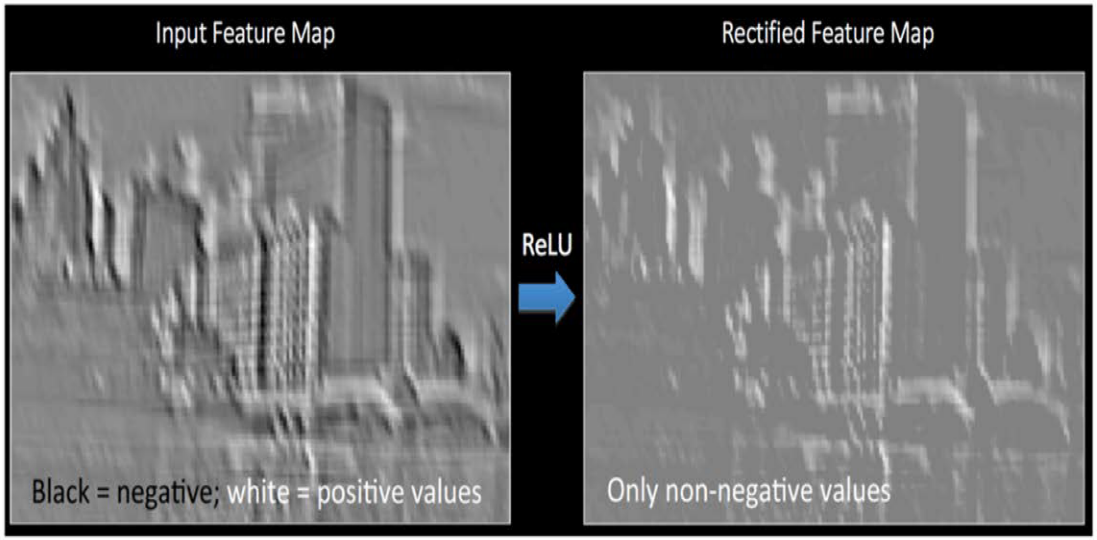
This slide is taken from Manolis Kellis

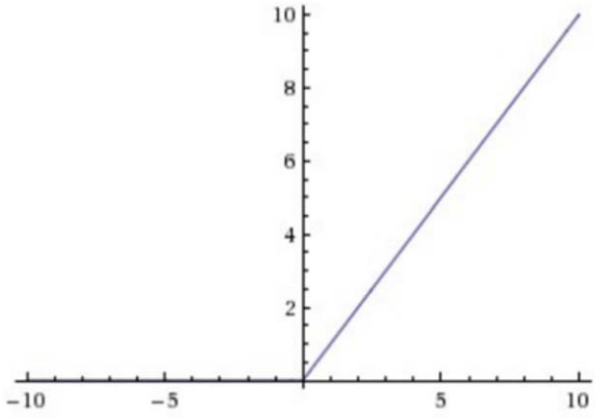
Lee+ *ICML* 2009

Introducing Non-Linearity

- Apply after every convolution operation (i.e., after convolutional layers)
- ReLU: pixel-by-pixel operation that replaces all negative values by zero.
- **Non-linear operation**

Rectified Linear Unit (ReLU)





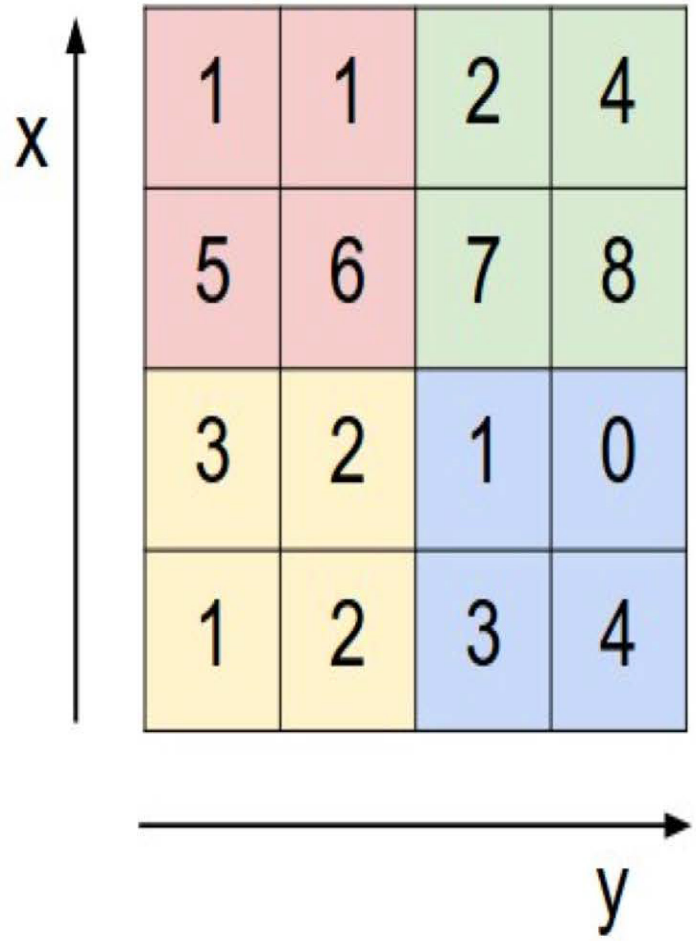
$$g(z) = \max(0, z)$$

`tf.keras.layers.ReLU`

This slide is taken from Manolis Kellis

Karn *Intuitive CNNs*

Pooling



max pool with 2x2 filters and stride 2

```
tf.keras.layers.MaxPool2D(
    pool_size=(2,2),
    strides=2
)
```

6	8
3	4

1) Reduced dimensionality
2) Spatial invariance

Max Pooling, average pooling

This slide is taken from Manolis Kellis

How can computers recognize objects?

Challenge:

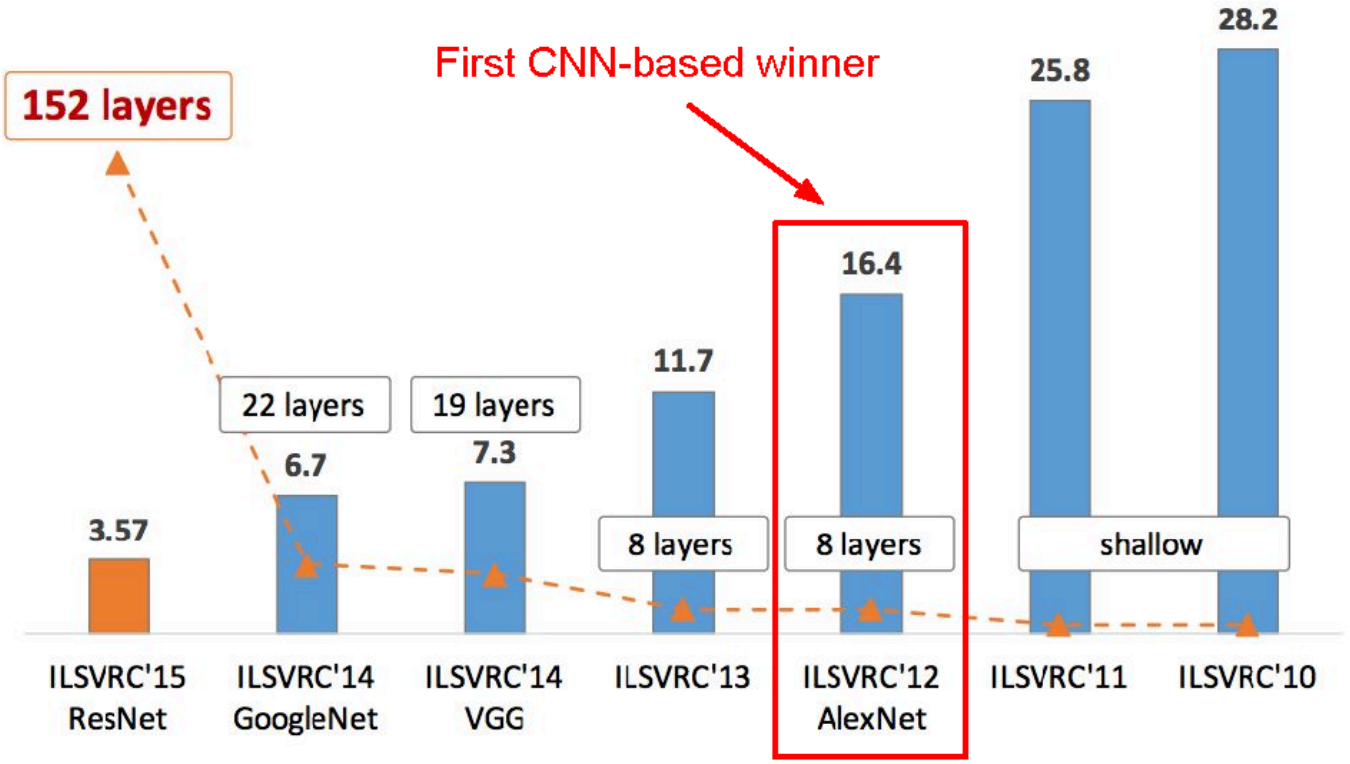
- Objects can be anywhere in the scene, in any orientation, rotation, color hue, etc.
- How can we overcome this challenge?

Answer:

- Learn a ton of features (millions) from the bottom up
- Learn the convolutional filters, rather than pre-computing them

This slide is taken from Manolis Kellis

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

AlexNet

- *ImageNet Classification with Deep Convolutional Neural Networks - Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton; 2012*
- Facilitated by GPUs, highly optimized convolution implementation and large datasets (ImageNet)
- Large CNN
- Has 60 Million parameter compared to 60k parameter of LeNet-5

This slide is taken from Manolis Kellis

[**Krizhevsky et al.**, **2012**]

AlexNet

Architecture
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

- Input: 227x227x3 images (224x224 before padding)
- First layer: 96 11x11 filters applied at stride 4

- **Output volume size?**

  (N-F)/s+1 = (227-11)/4+1 = 55 -> [55x55x96]

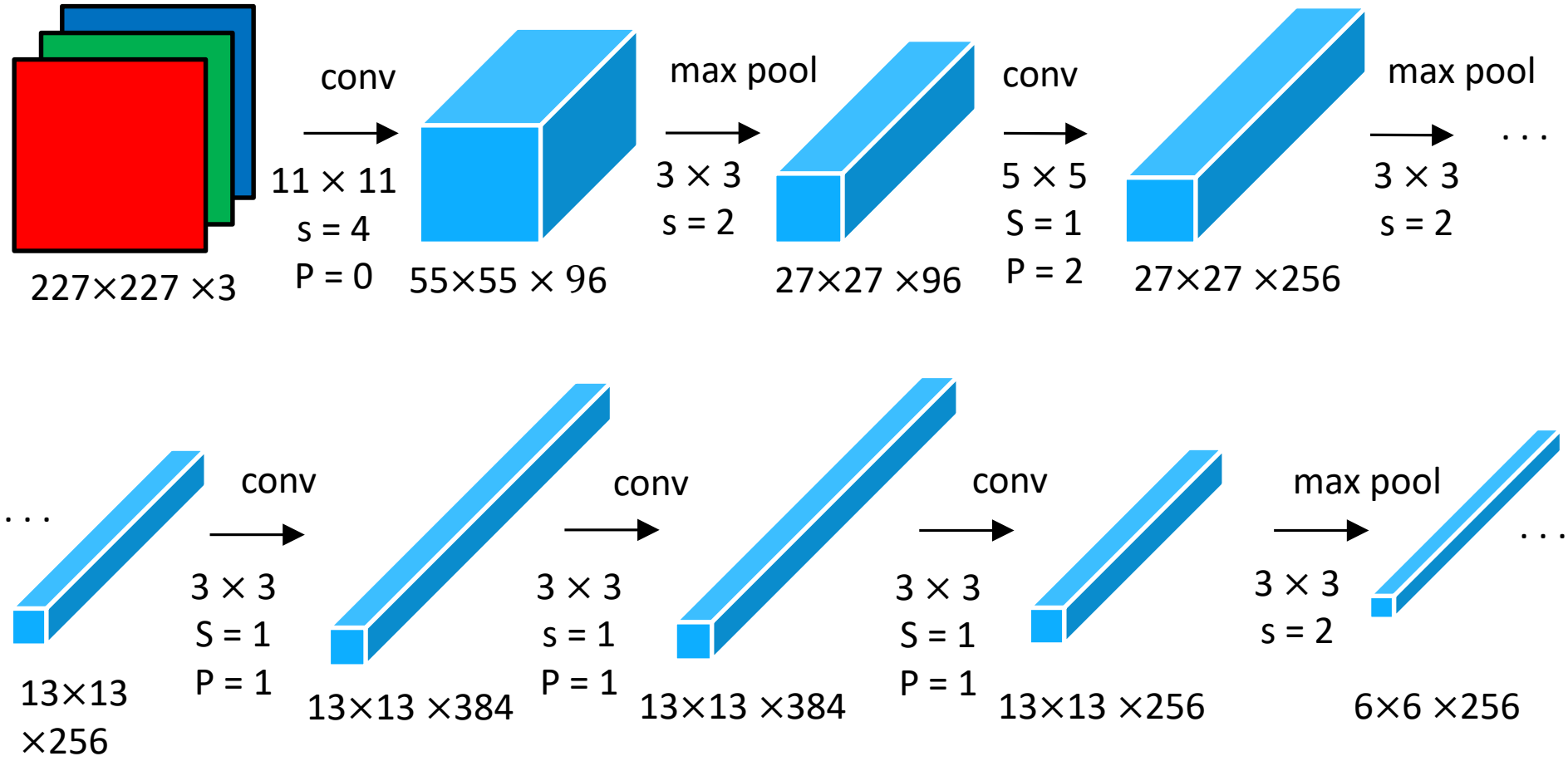
- **Number of parameters in this layer?**

  (11*11*3)*96 = 35K

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Krizhevsky et al., 2012]

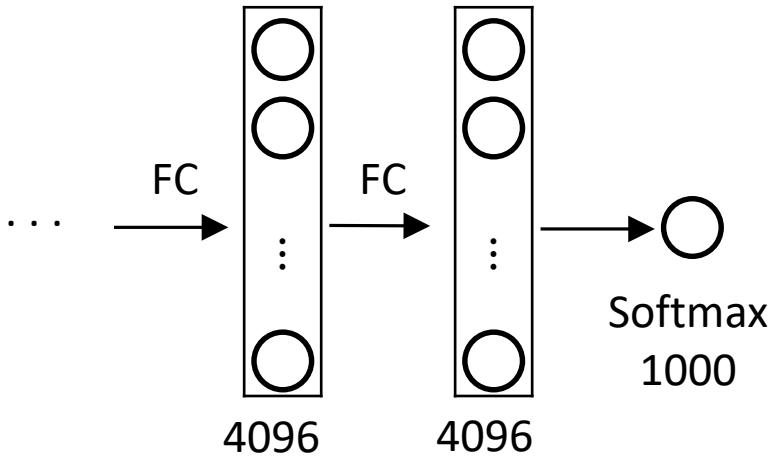
AlexNet



This slide is taken from Andrew Ng

[Krizhevsky et al., 2012]

AlexNet



This slide is taken from Andrew Ng

[**Krizhevsky et al.**, **2012**]

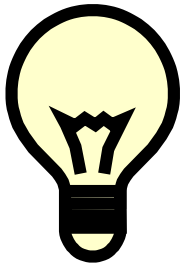
AlexNet

Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- 7 CNN ensemble

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[**Krizhevsky et al., 2012**]



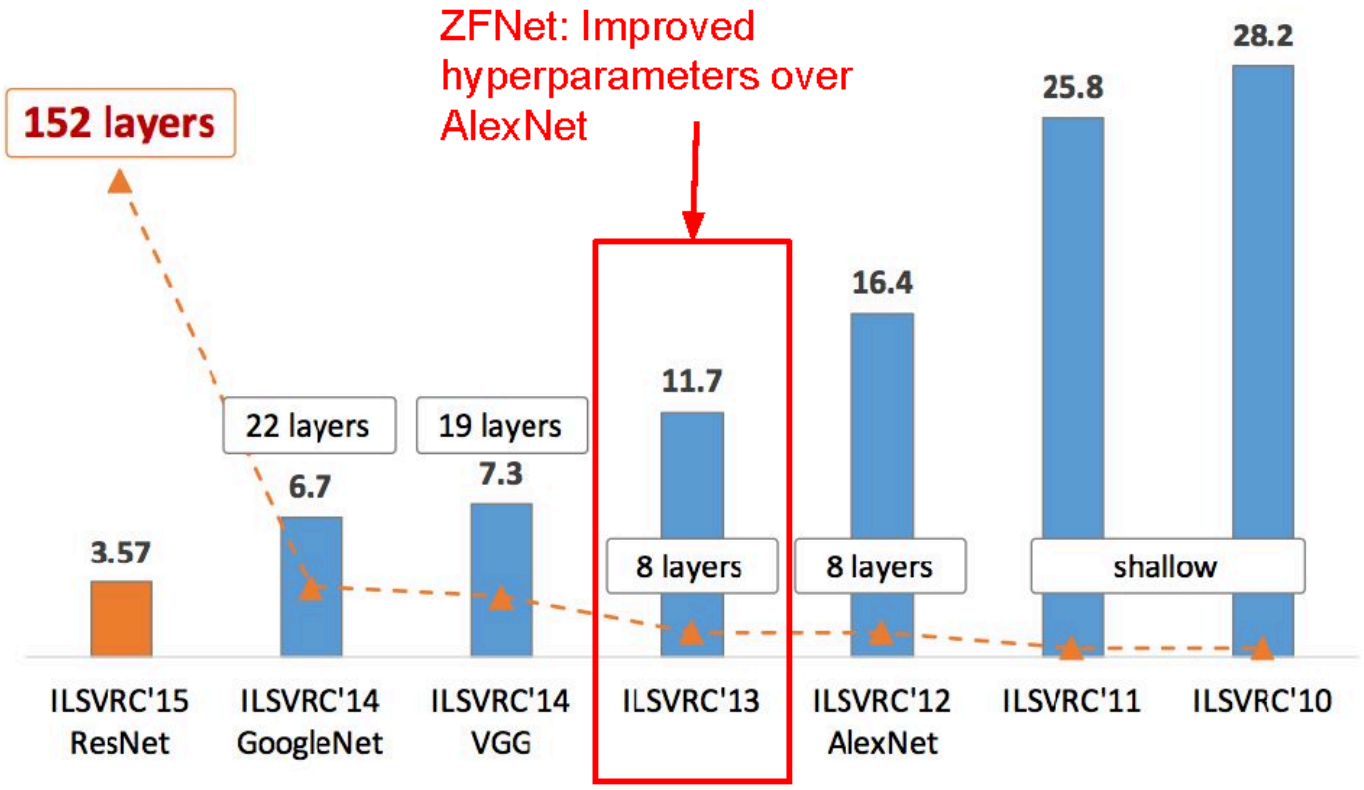
AlexNet

AlexNet was the coming out party for CNNs in the computer vision community. This was **the first time a model performed so well on a historically difficult ImageNet dataset**. This paper illustrated the benefits of CNNs and backed them up with record breaking performance in the competition.

This slide is taken from Manolis Kellis

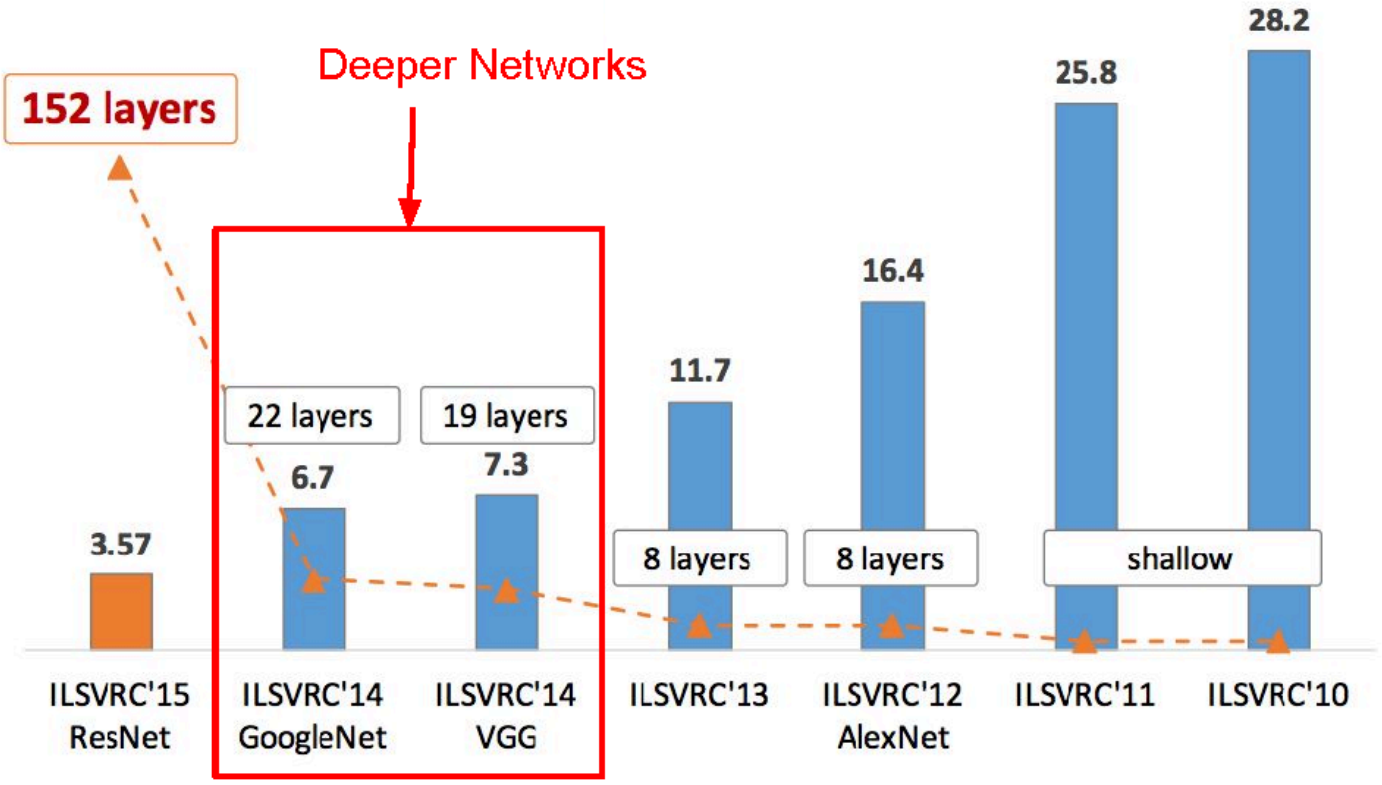
[**Krizhevsky et al., 2012**]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

Input
3x3 conv, 64
3x3 conv, 64
Pool 1/2
3x3 conv, 128
3x3 conv, 128
Pool 1/2
3x3 conv, 256
3x3 conv, 256
Pool 1/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool 1/2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
Pool 1/2
FC 4096
FC 4096
FC 1000
Softmax

VGGNet

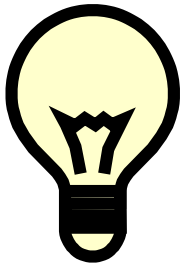
- **Smaller filters**
  Only 3x3 CONV filters, stride 1, pad 1
  and 2x2 MAX POOL , stride 2

- **Deeper network**
  AlexNet: 8 layers
  VGGNet: 16 - 19 layers

- ZFNet: 11.7% top 5 error in ILSVRC'13
- VGGNet: 7.3% top 5 error in ILSVRC'14

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Simonyan and Zisserman, 2014]



VGGNet

VGG Net reinforced the notion that **convolutional neural networks have to have a deep network of layers in order for this hierarchical representation of visual data to work**.

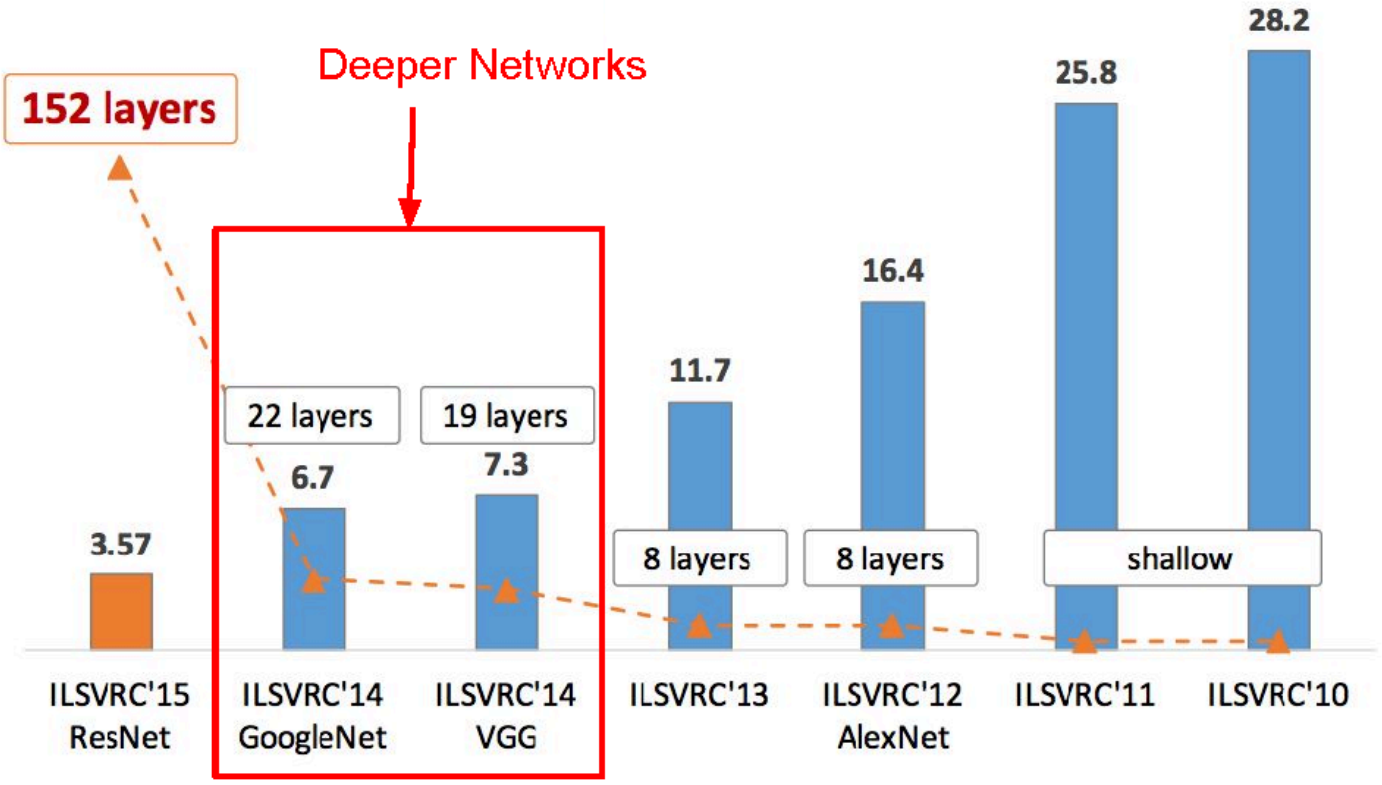
Keep it deep.

Keep it simple.

This slide is taken from Manolis Kellis

[Simonyan and Zisserman, 2014]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



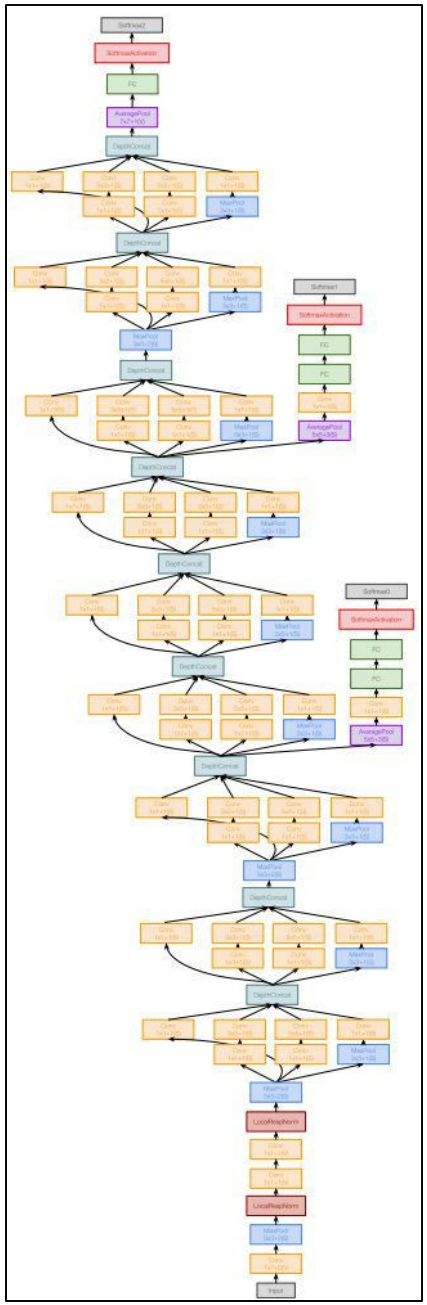
Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

GoogleNet

- *Going Deeper with Convolutions - Christian Szegedy et al.; 2015*
- ILSVRC 2014 competition winner
- Also significantly deeper than AlexNet
- x12 less parameters than AlexNet
- Focused on computational efficiency

This slide is taken from Manolis Kellis

[Szegedy et al., 2014]



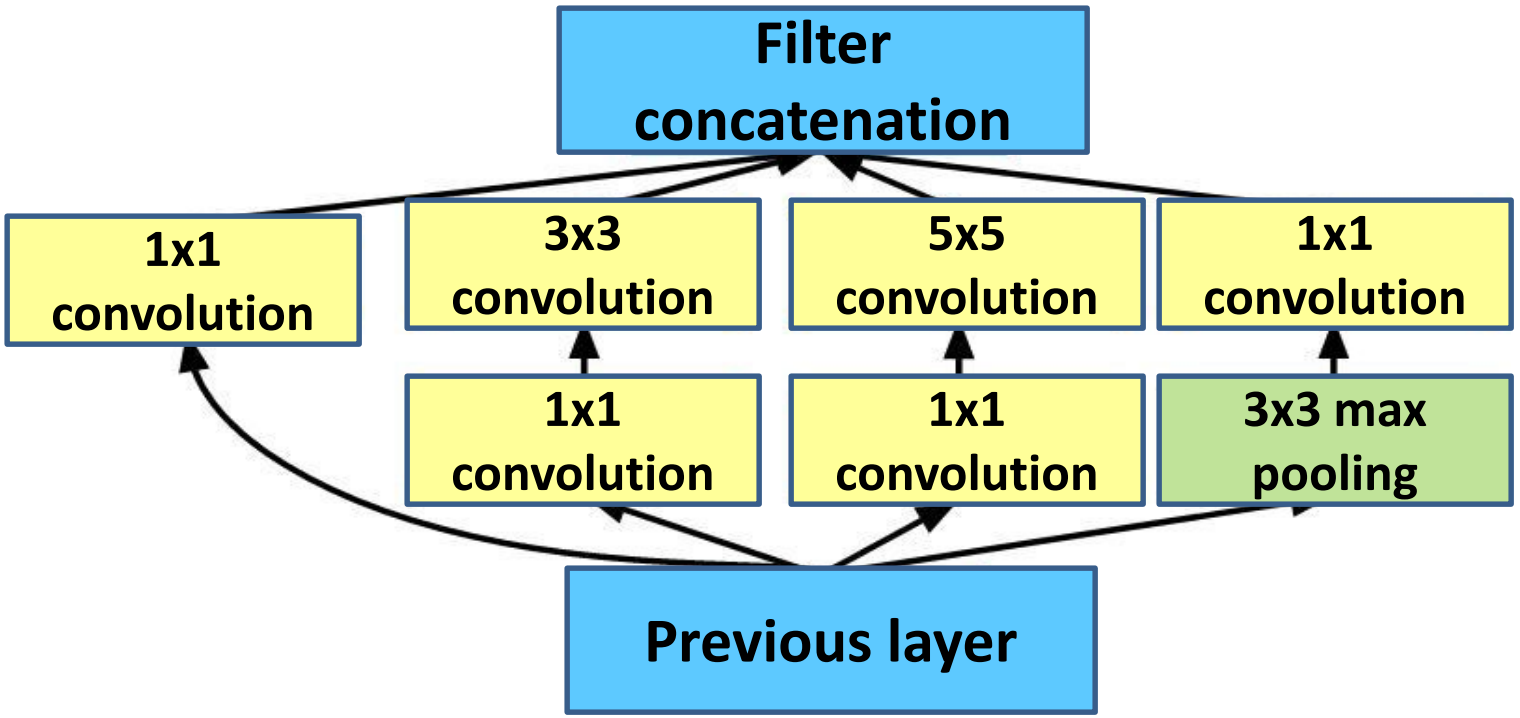
GoogleNet

- 22 layers
- Efficient **“Inception” module** - strayed from the general approach of simply stacking conv and pooling layers on top of each other in a sequential structure
- No FC layers
- Only 5 million parameters!
- ILSVRC’14 classification winner (6.7% top 5 error)

[Szegedy et al., 2014]

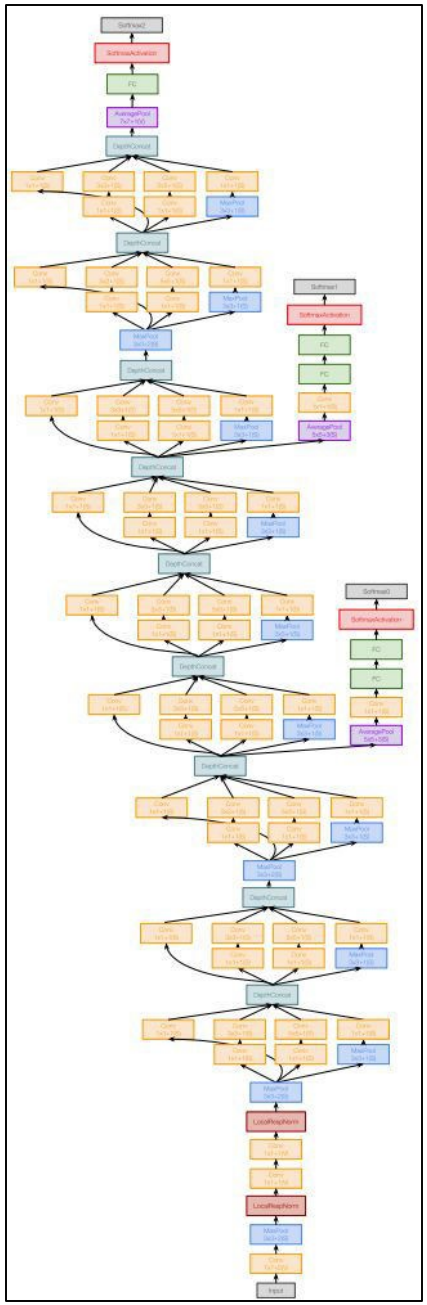
GoogleNet

"**Inception module":** design a good local network topology (network within a network) and then stack these modules on top of each other



Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Szegedy et al., 2014]



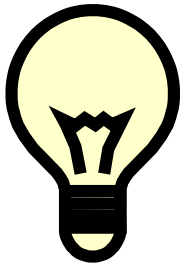
GoogleNet

Details/Retrospectives :

- Deeper networks, with computational efficiency
- 22 layers
- Efficient “Inception” module
- No FC layers
- 12x less params than AlexNet
- ILSVRC’14 classification winner (6.7% top 5 error)

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[Szegedy et al., 2014]



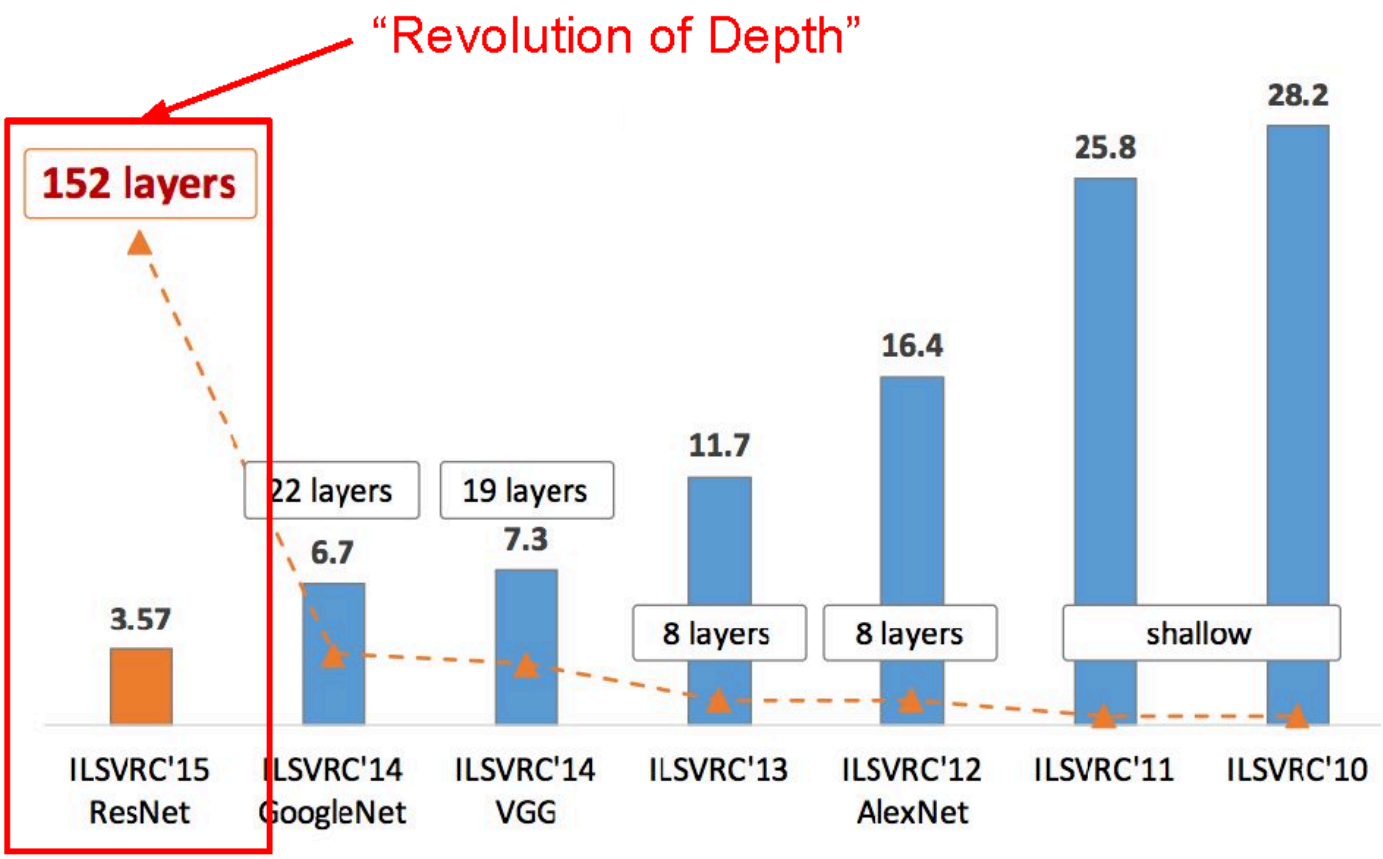
GoogleNet

Introduced the idea that CNN layers **didn't always have to be stacked up sequentially**. Coming up with the Inception module, the authors showed that a creative structuring of layers can lead to improved performance and **computationally efficiency**.

This slide is taken from Manolis Kellis

[Szegedy et al., 2014]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



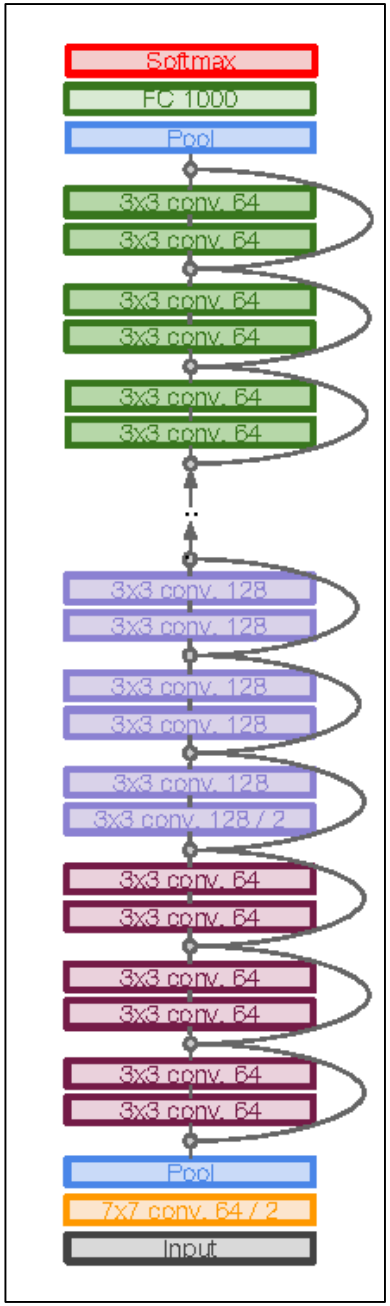
Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

ResNet

- *Deep Residual Learning for Image Recognition - Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; 2015*
- Extremely deep network – 152 layers
- Deeper neural networks are more difficult to train.
- Deep networks suffer from vanishing and exploding gradients.
- Present a residual learning framework to ease the training of networks that are substantially deeper than those used previously.

This slide is taken from Manolis Kellis

[He et al., 2015]



ResNet

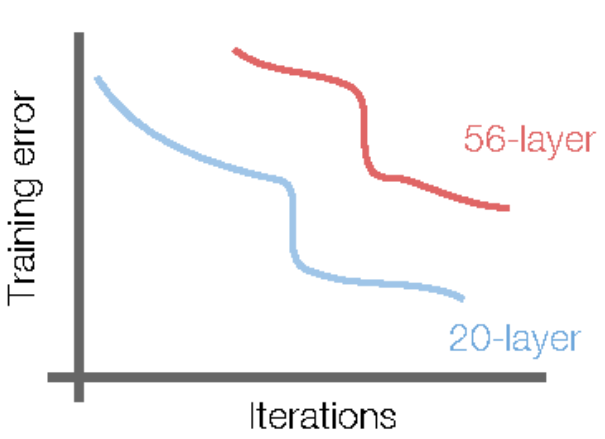
- ILSVRC'15 classification winner (3.57% top 5 error, humans generally hover around a 5-10% error rate)
  Swept all classification and detection competitions in ILSVRC'15 and COCO'15!

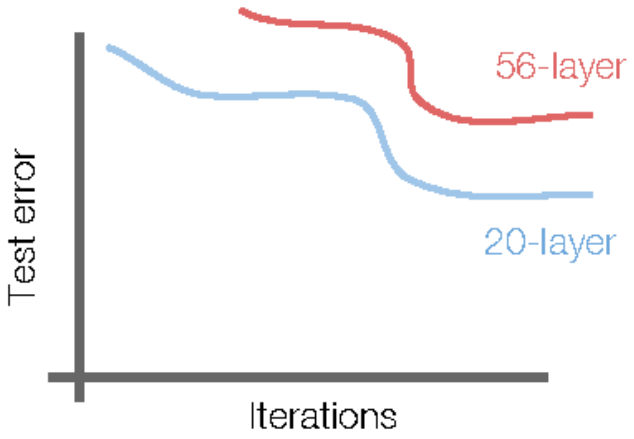
Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[He et al., 2015]

ResNet

- What happens when we continue stacking deeper layers on a convolutional neural network?





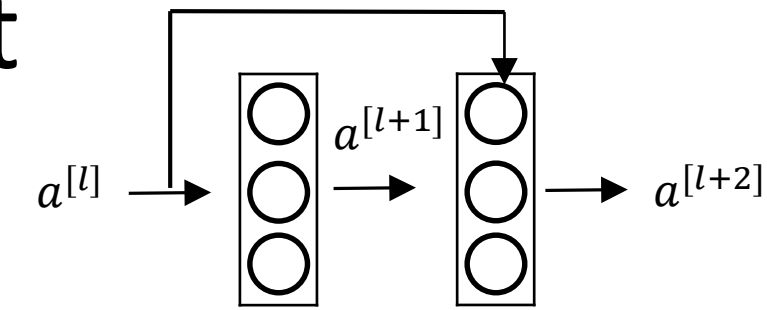
- 56-layer model performs worse on both training and test error

-> The deeper model performs worse (not caused by overfitting)!

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[He et al., 2015]

ResNet



Short cut/ skip connection

$a^{[l]}$ → **Linear** → **ReLU** → **Linear** → **ReLU** → $a^{[l+2]}$

$a^{[l+1]}$

$$\mathbf{z}^{[l+1]} = \mathbf{W}^{[l+1]}\,\mathbf{a}^{[l]} + \mathbf{b}^{[l+1]}$$

$$\mathbf{a}^{[l+1]} = \mathbf{g}(\mathbf{z}^{[l+1]})$$

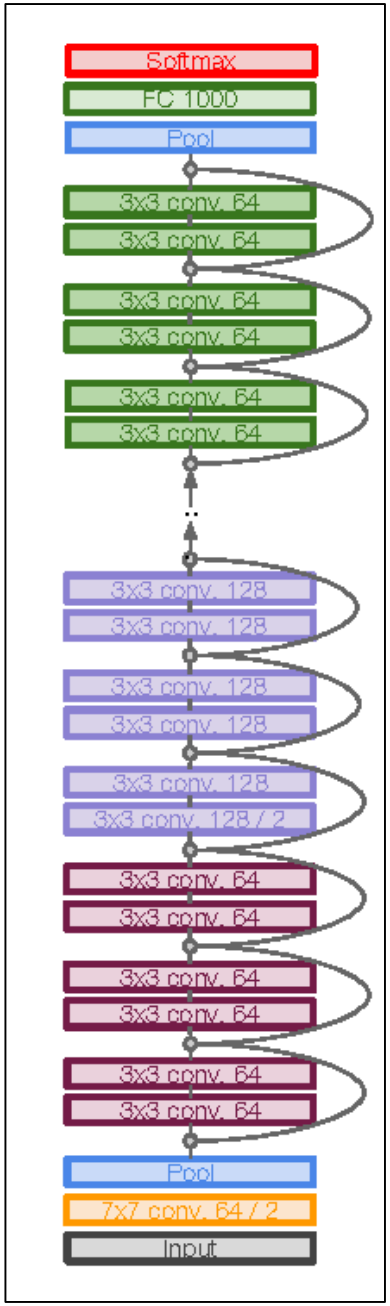
$$\mathbf{z}^{[l+2]} = \mathbf{W}^{[l+2]}\mathbf{a}^{[l+1]} + \mathbf{b}^{[l+2]}$$

$$\mathbf{a}^{[l+2]} = \mathbf{g}(\mathbf{z}^{[l+2]})$$

$$\mathbf{a}^{[l+2]} = \mathbf{g}\left(\mathbf{z}^{[l+2]} + \mathbf{a}^{[l]}\right) = \mathbf{g}(\mathbf{W}^{[l+2]}\mathbf{a}^{[l+1]} + \mathbf{b}^{[l+2]} + \mathbf{a}^{[l]})$$

This slide is taken from Manolis Kellis

[He et al., 2015]



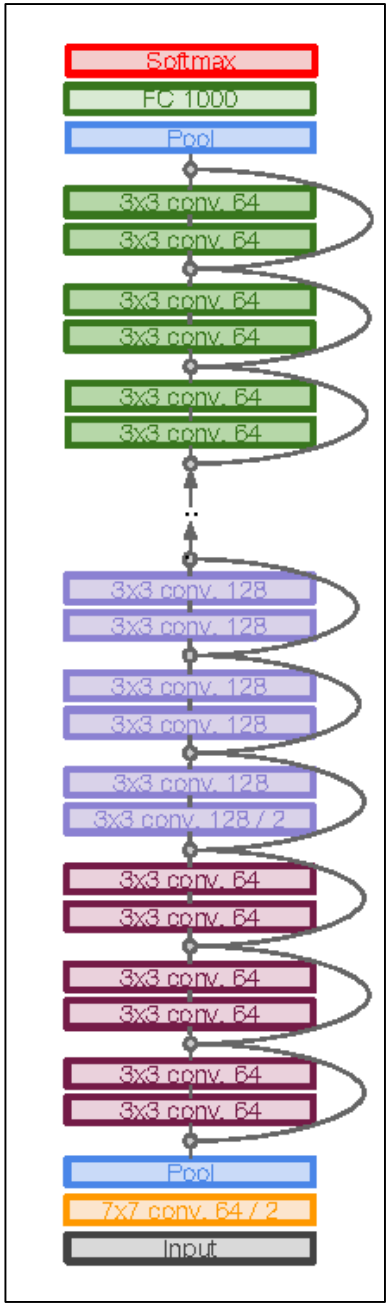
ResNet

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[He et al., 2015]



ResNet

- Total depths of 34, 50, 101, or 152 layers for ImageNet
- For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[He et al., 2015]

ResNet

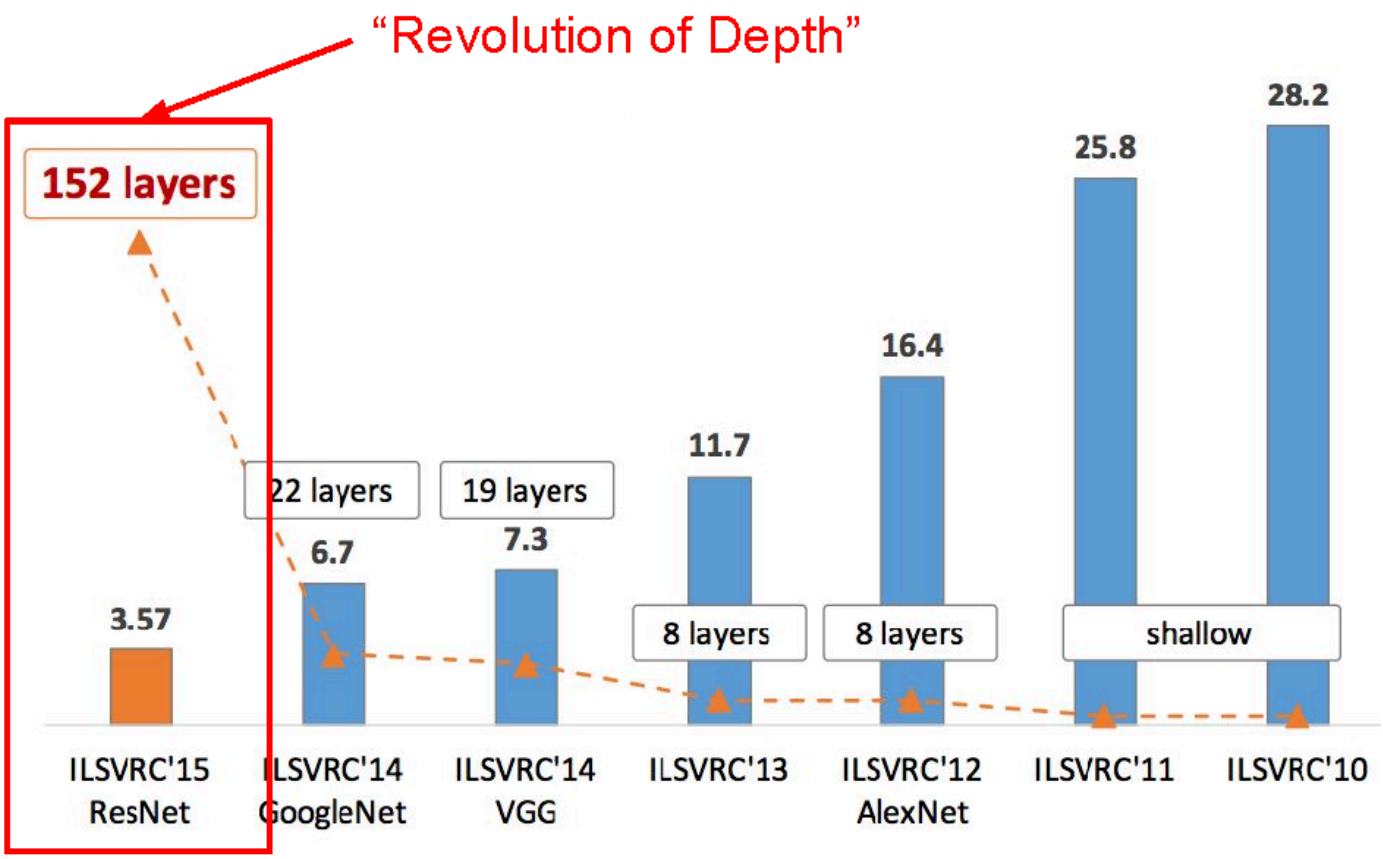
Experimental Results:

- Able to train very deep networks without degrading
- Deeper networks now achieve lower training errors as expected

Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

[He et al., 2015]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

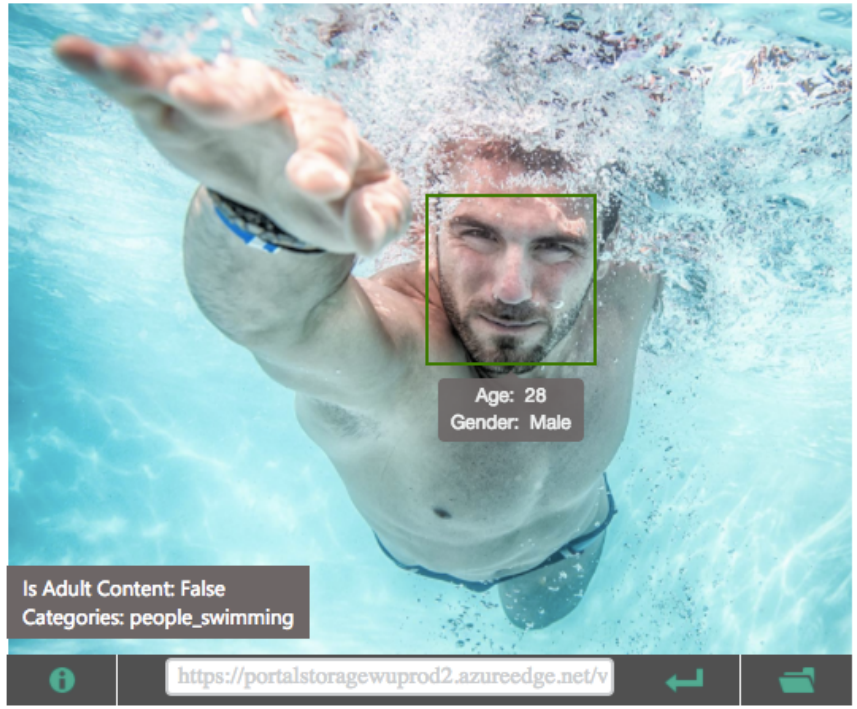


Slide taken from Fei-Fei & Justin Johnson & Serena Yeung. Lecture 9.

RNN

Sequence Applications: One-to-Many

- **Input**: fixed-size
- **Output**: sequence
- e.g., image captioning



Features:

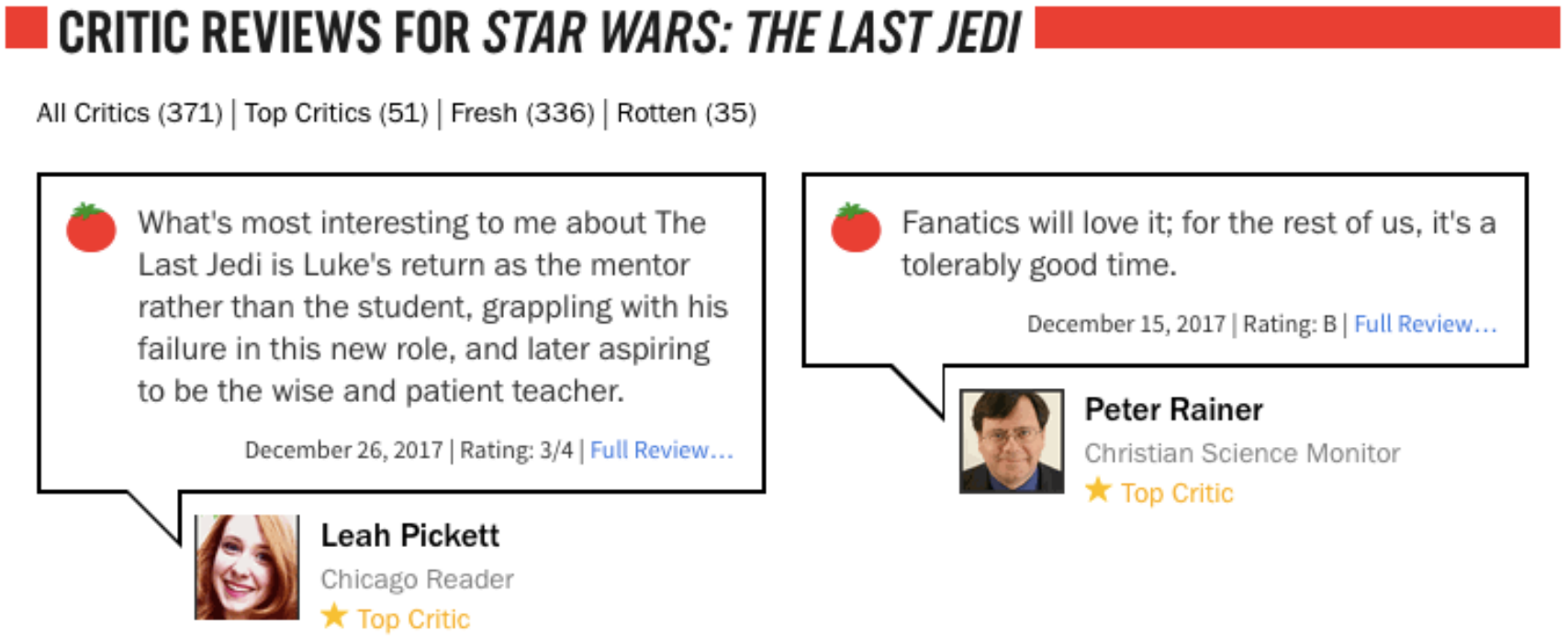
Feature Name	Value
Description	{ "type": 0, "captions": [{ "text": "a man swimming in a pool of water", "confidence": 0.7850108693093019 }] }
Tags	[{ "name": "water", "confidence": 0.9996442794799805 }, { "name": "sport", "confidence": 0.9504992365837097 }, { "name": "swimming", "confidence": 0.9062818288803101, "hint": "sport" }, { "name": "pool", "confidence": 0.8787588477134705 }, { "name": "water sport", "confidence": 0.631849467754364, "hint": "sport" }]
Image Format	jpeg
Image Dimensions	1500 x 1155
Clip Art Type	0 Non-clipart
Line Drawing Type	0 Non-LineDrawing
Black & White Image	False

Captions: https://www.microsoft.com/cognitive-services/en-us/computer-vision-api

This slide is taken from Danna Gurari

Sequence Applications: Many-to-One

- **Input**: sequence
- **Output**: fixed-size
- e.g., sentiment analysis (hate? love?, etc)



https://www.rottentomatoes.com/m/star_wars_the_last_jedi

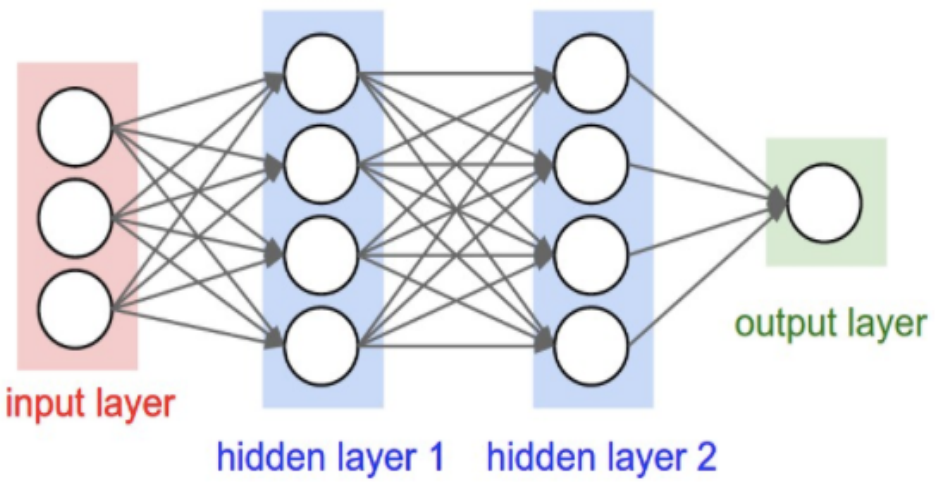
This slide is taken from Danna Gurari

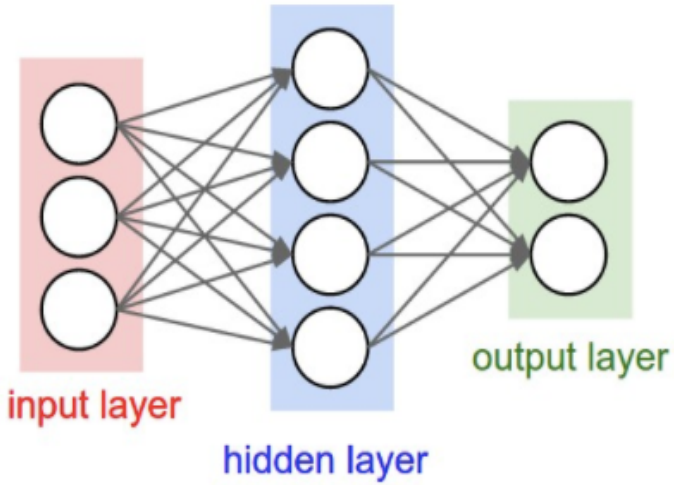
Sequence Applications: Many-to-Many

- **Input**: sequence
- **Output**: sequence
- e.g., language translation

This slide is taken from Danna Gurari

Recall: Feedforward Neural Networks





Problem: many model parameters!

Problem: no memory of past since weights learned independently

Each layer serves as input to the next layer with no loops

This slide is taken from Danna Gurari

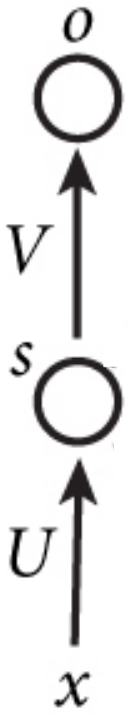
Figure Source: http://cs231n.github.io/neural-networks-1/

Recurrent Neural Networks (RNNs)

- Main idea: use hidden state to capture information about the past

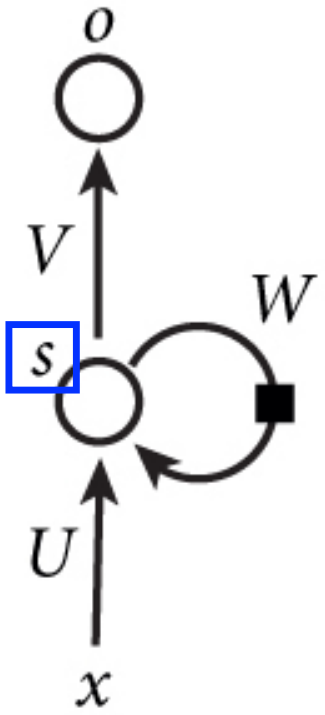
Feedforward Network

Each layer receives input from the previous layer with no loops



Recurrent Network

Each layer receives input from the previous layer and the output from the previous time step

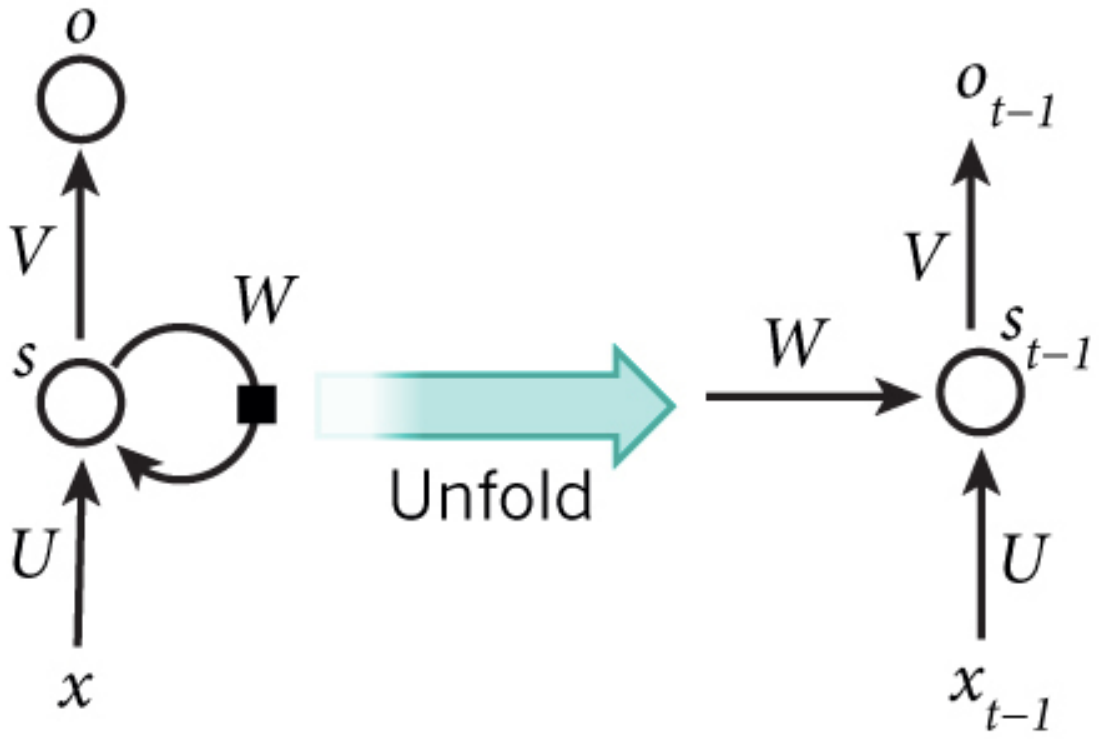


This slide is taken from Danna Gurari

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

RNN: Time Step 1

- Main idea: use hidden state to capture information about the past

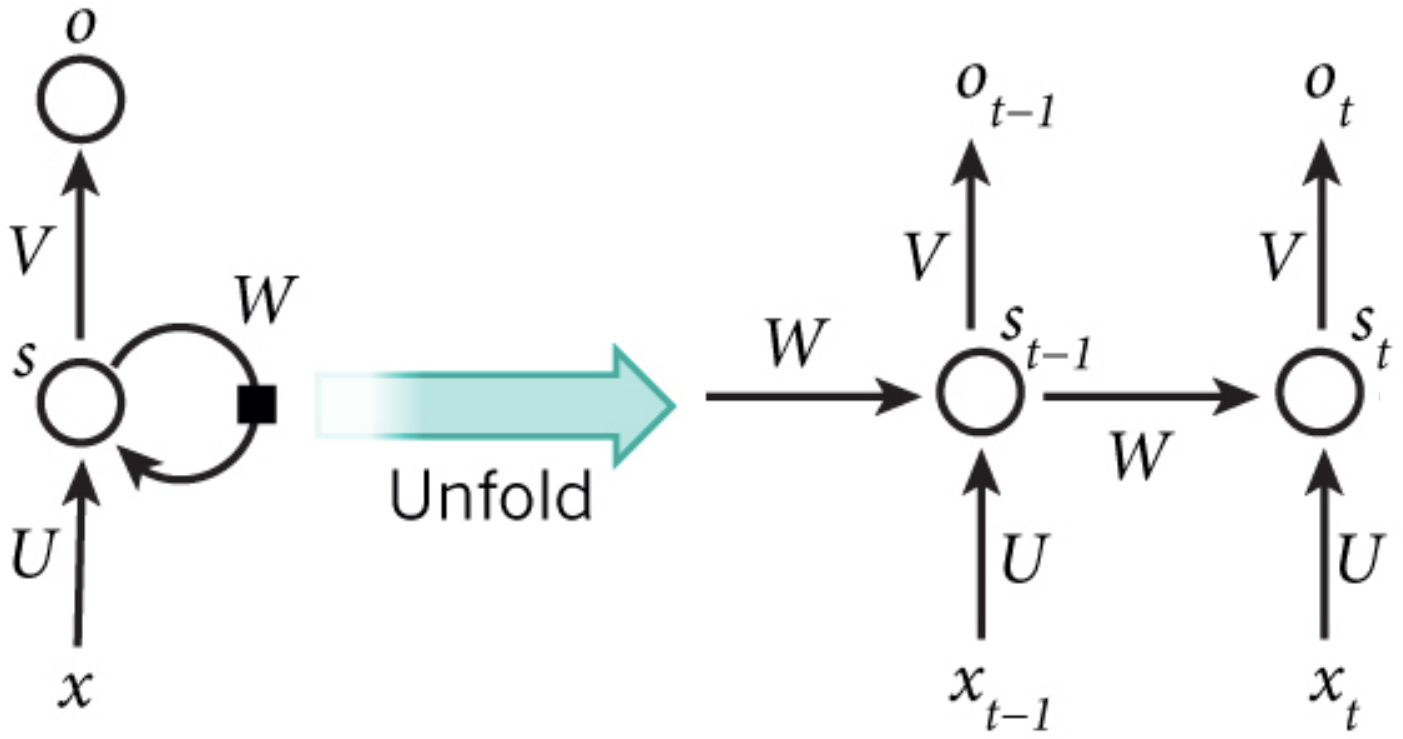


This slide is taken from Danna Gurari

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

RNN: Time Step 2

- Main idea: use hidden state to capture information about the past

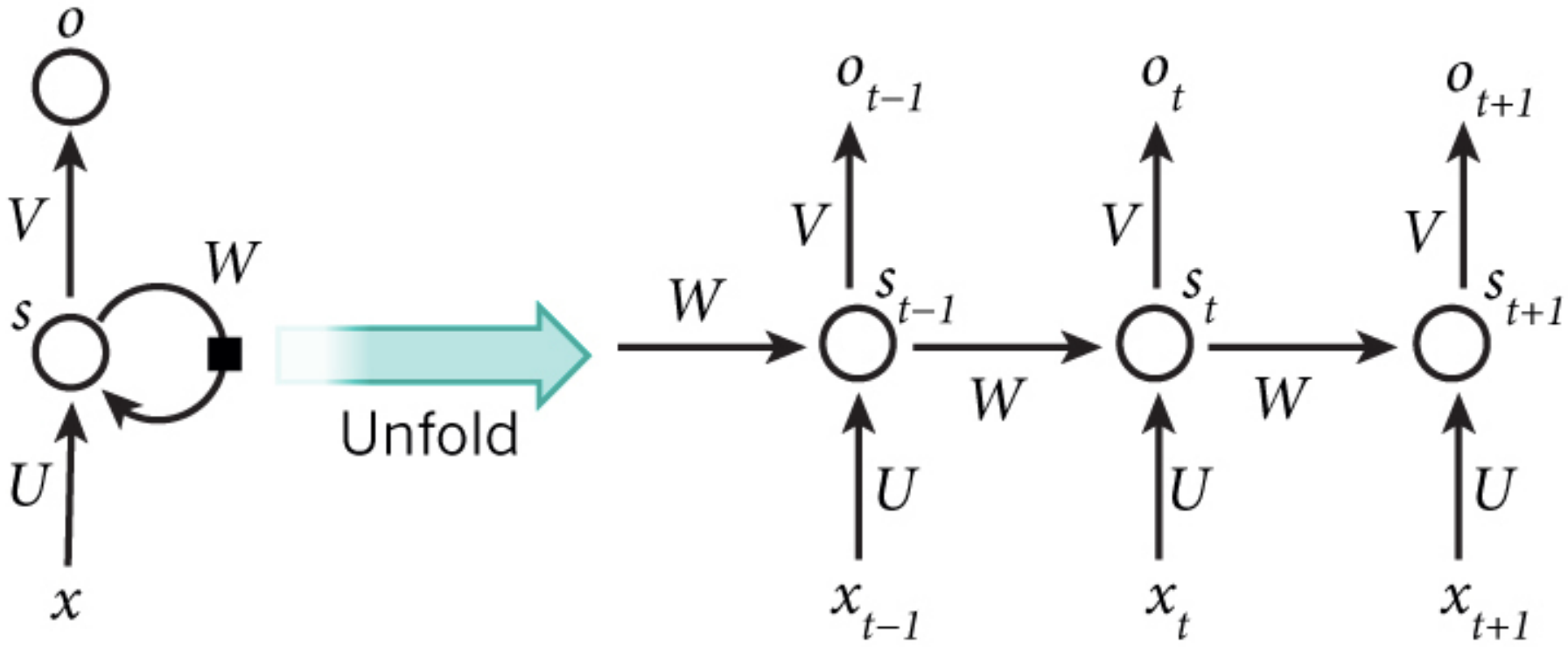


This slide is taken from Danna Gurari

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

RNN: Time Step 3

- Main idea: use hidden state to capture information about the past

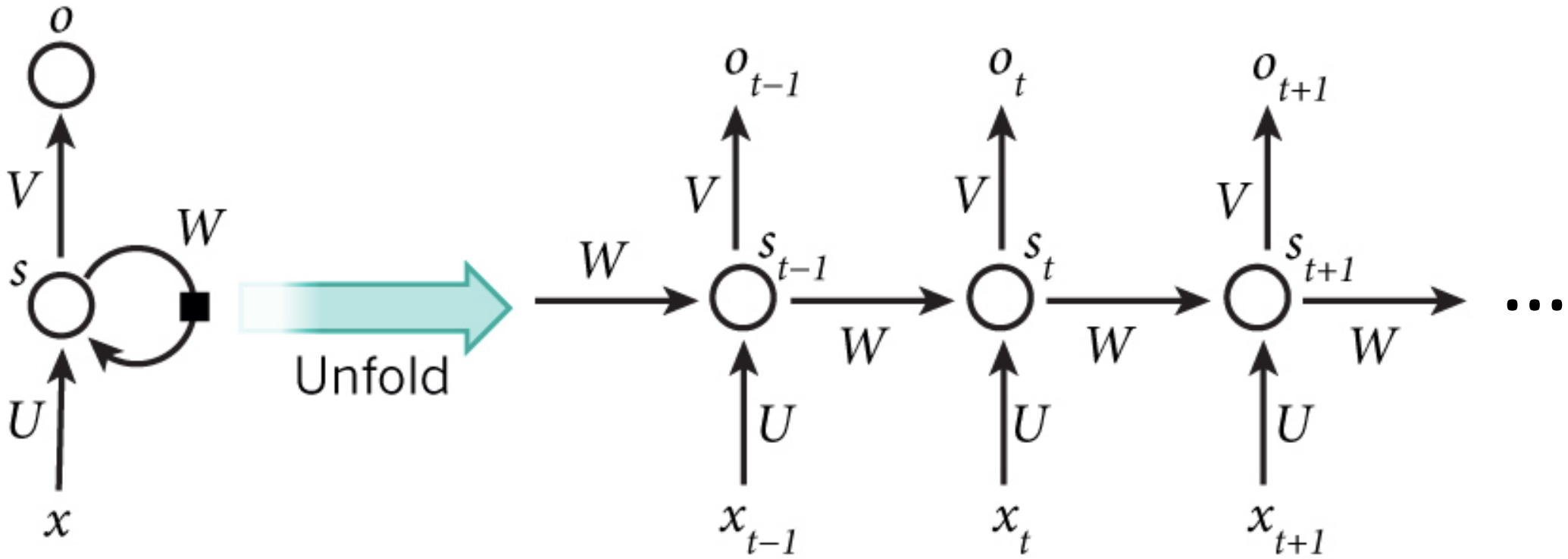


This slide is taken from Danna Gurari

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

RNN: And So On…

- Main idea: use hidden state to capture information about the past

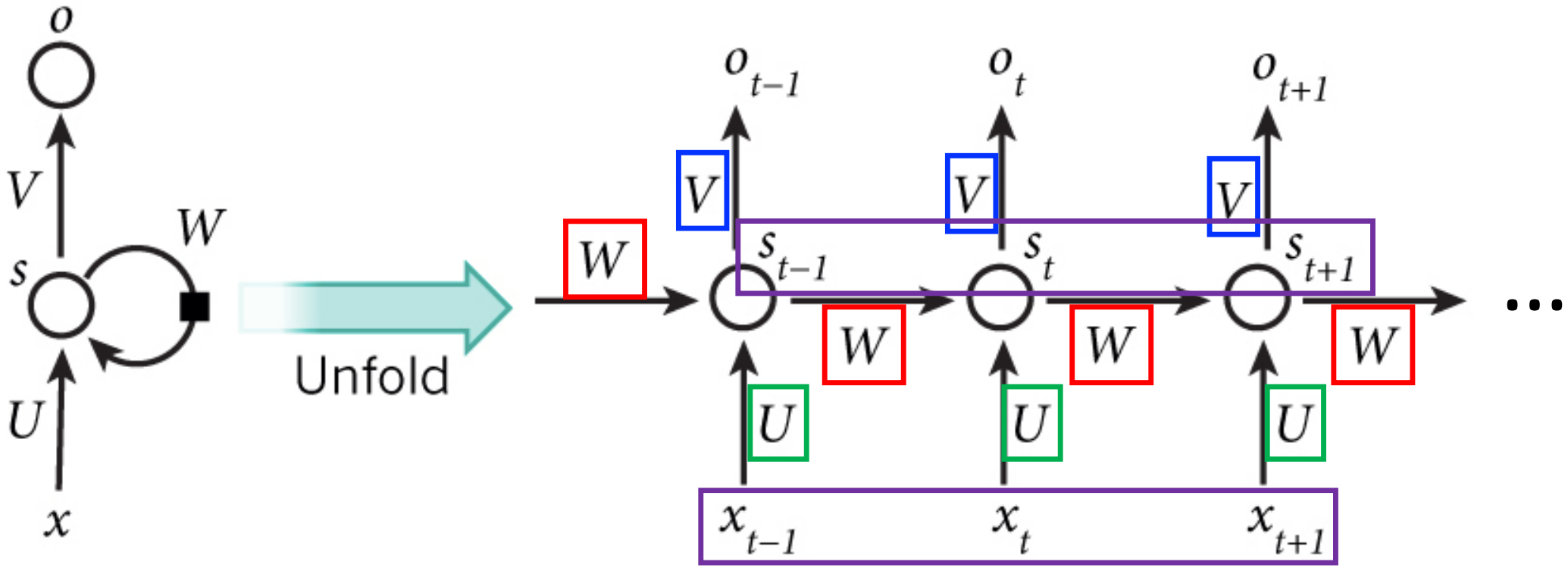


This slide is taken from Danna Gurari

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

RNN: Model Parameters and Inputs

- All layers share the same model parameters (U, V, W)
  - What is different between the layers?

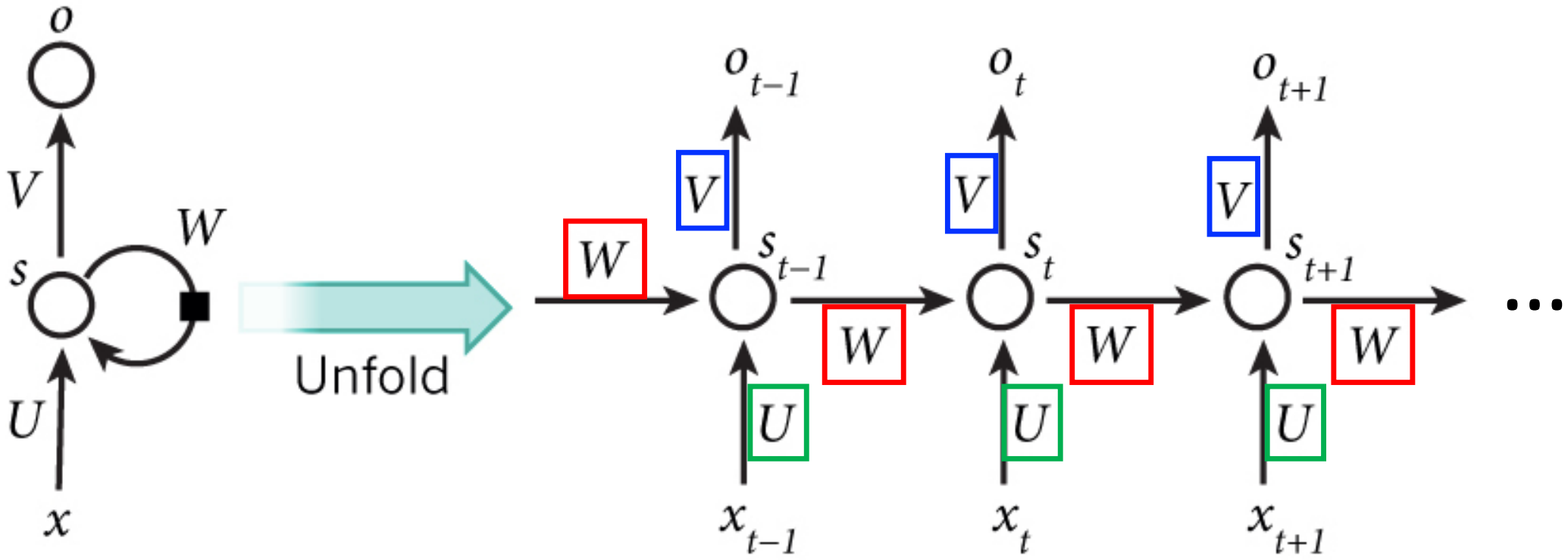


This slide is taken from Danna Gurari

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

RNN: Model Parameters and Inputs

- When unfolded, a RNN is a deep feedforward network with shared weights!

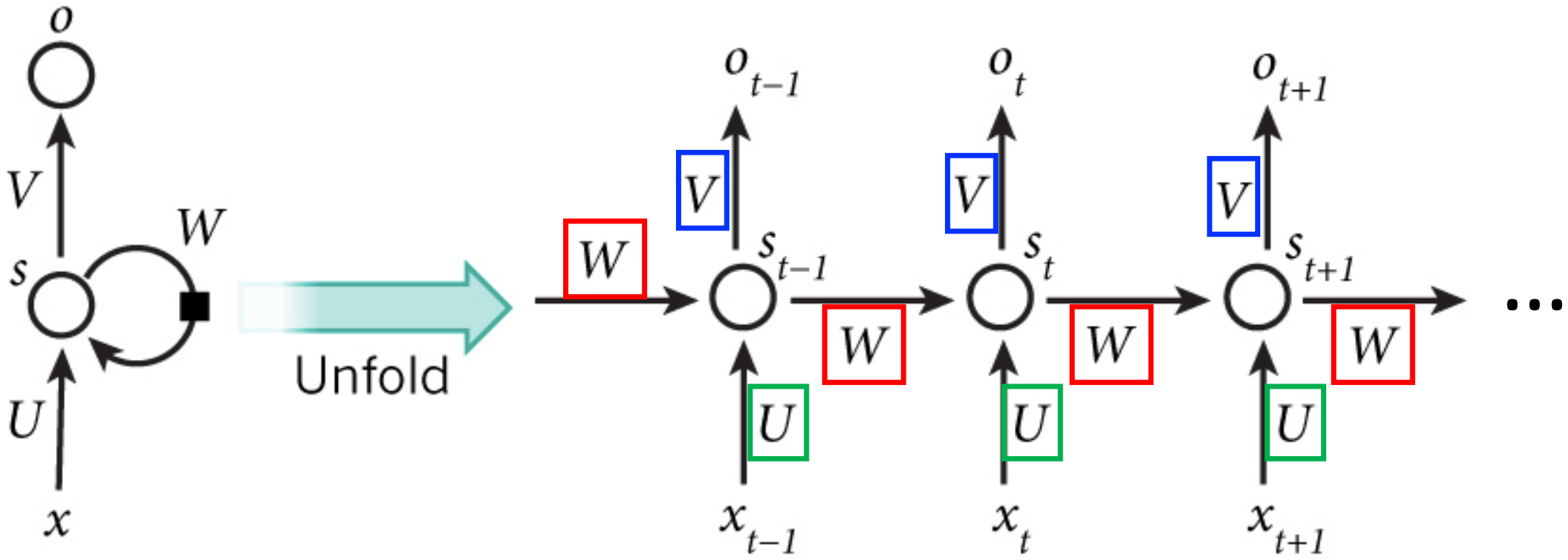


This slide is taken from Danna Gurari

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

RNN: Advantages

- Overcomes problem that weights of each layer are learned **independently** by using previous hidden state
- Overcomes problem that model has many parameters since weights are shared across layers

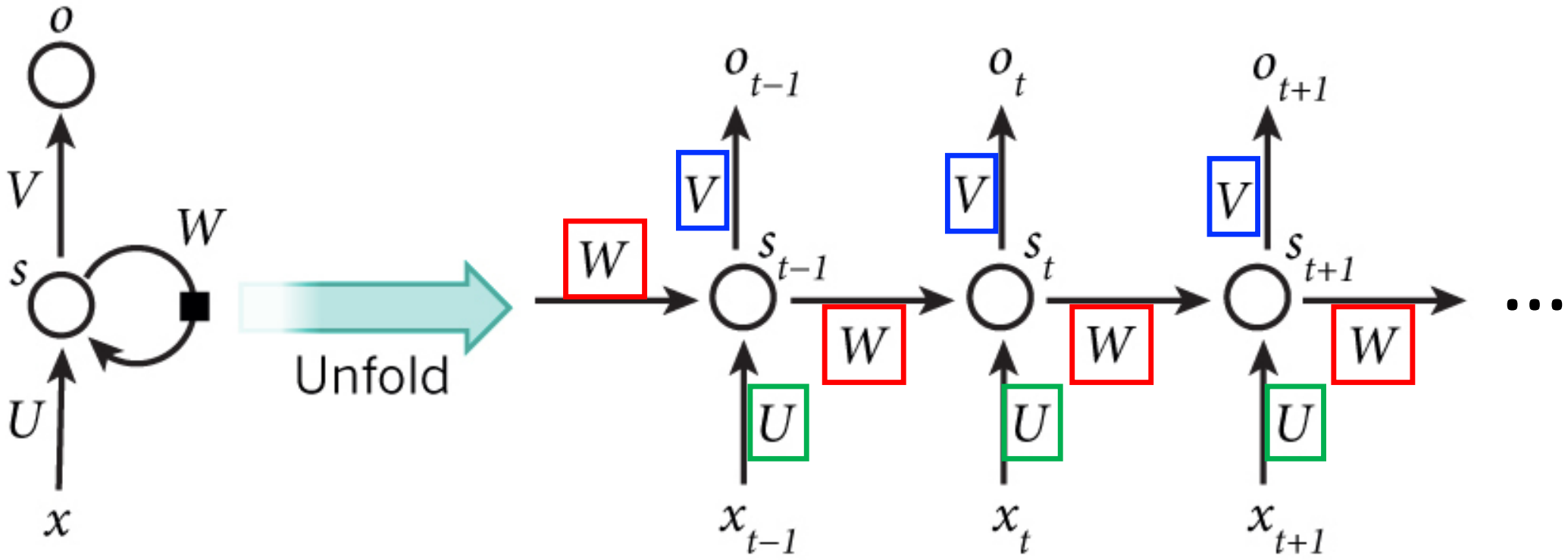


This slide is taken from Danna Gurari

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

RNN: Advantages

- Retains information about past inputs for an amount of time that depends on the model's weights and input data rather than a fixed duration selected a priori



This slide is taken from Danna Gurari

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

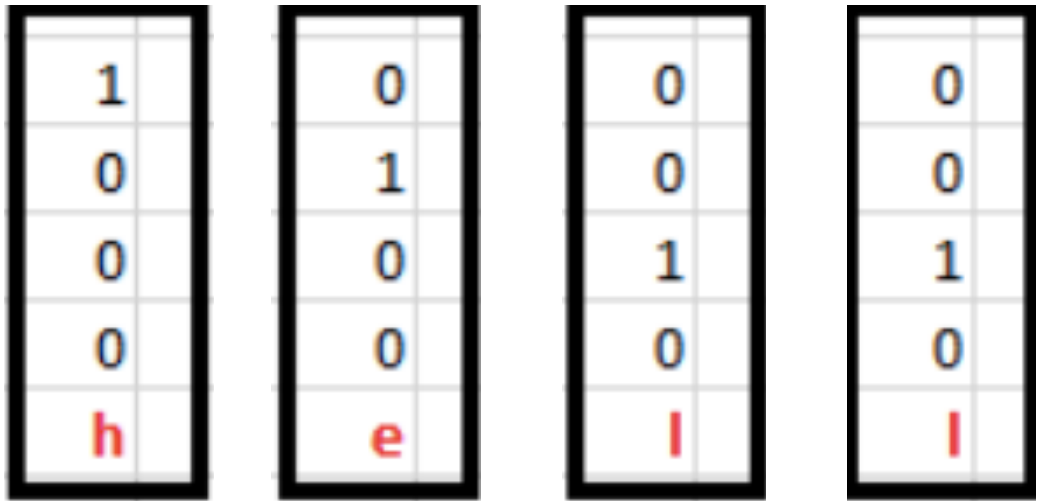
RNN Example: Predict Sequence of Characters

- Goal: predict next character in text
- Training Data: sequence of characters represented as one-hot vectors

This slide is taken from Danna Gurari

Example: Predict Sequence of Characters

- Goal: predict next character in text
- Prediction: feed training sequence of one-hot encoded characters; e.g., “hello”
  - For simplicity, assume the following vocabulary (i.e., character set): {h, e, l, o}
  - What is our input at time step 1?
  - What is our input at time step 2?
  - What is our input at time step 3?
  - What is our input at time step 4?
  - And so on…



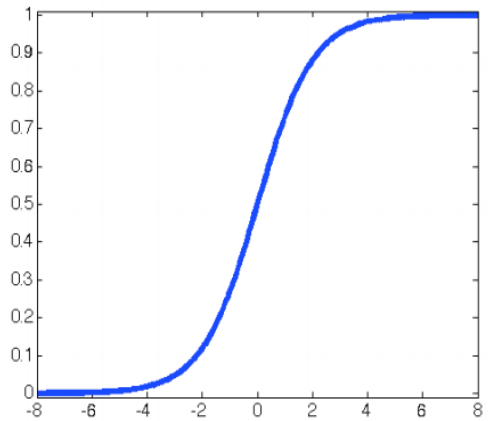
This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

Example: Predict Sequence of Characters

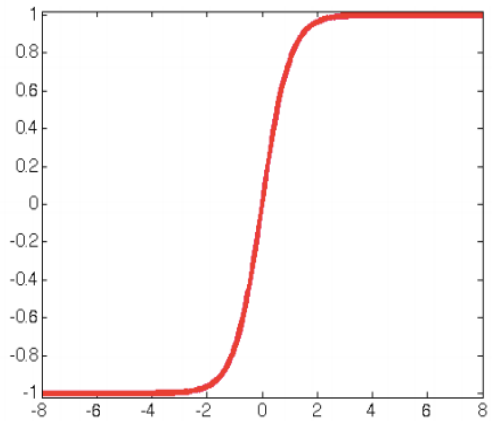
Recall activation functions: use tanh as activation function

Sigmoid



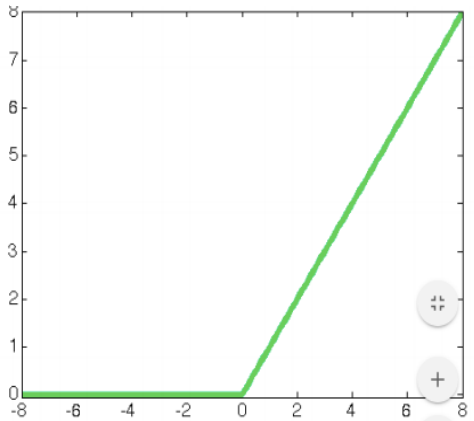
$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

Tanh



$$\tanh(z) = \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$$

ReLU



$$\mathrm{ReLU}(z) = \max(0, z)$$

This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

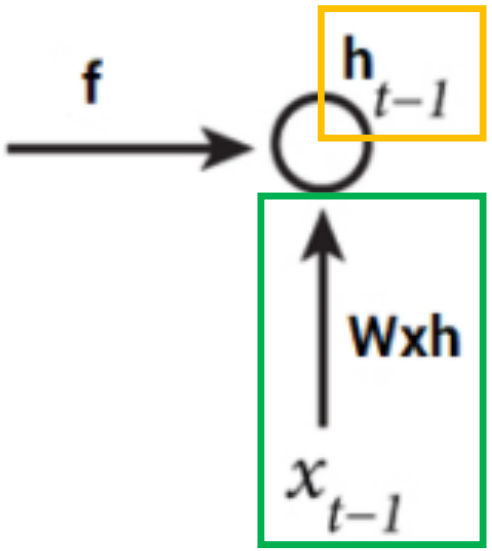
Example: Predict Sequence of Characters

Initialize to random value: 0.567001

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

Initialize to 0

Input at next time step



wxh			
0.287027	0.84606	0.572392	0.486813
0.902874	0.871522	0.691079	0.18998
0.537524	0.09224	0.558159	0.491528

× [1, 0, 0, 0, h] = [0.287027, 0.902874, 0.537524]

H_t = TANH { [0.287027359, 0.902874425, 0.537523791] + [0.567001, 0.567001, 0.567001] }

= [0.693168, 0.899554, 0.802118]

This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

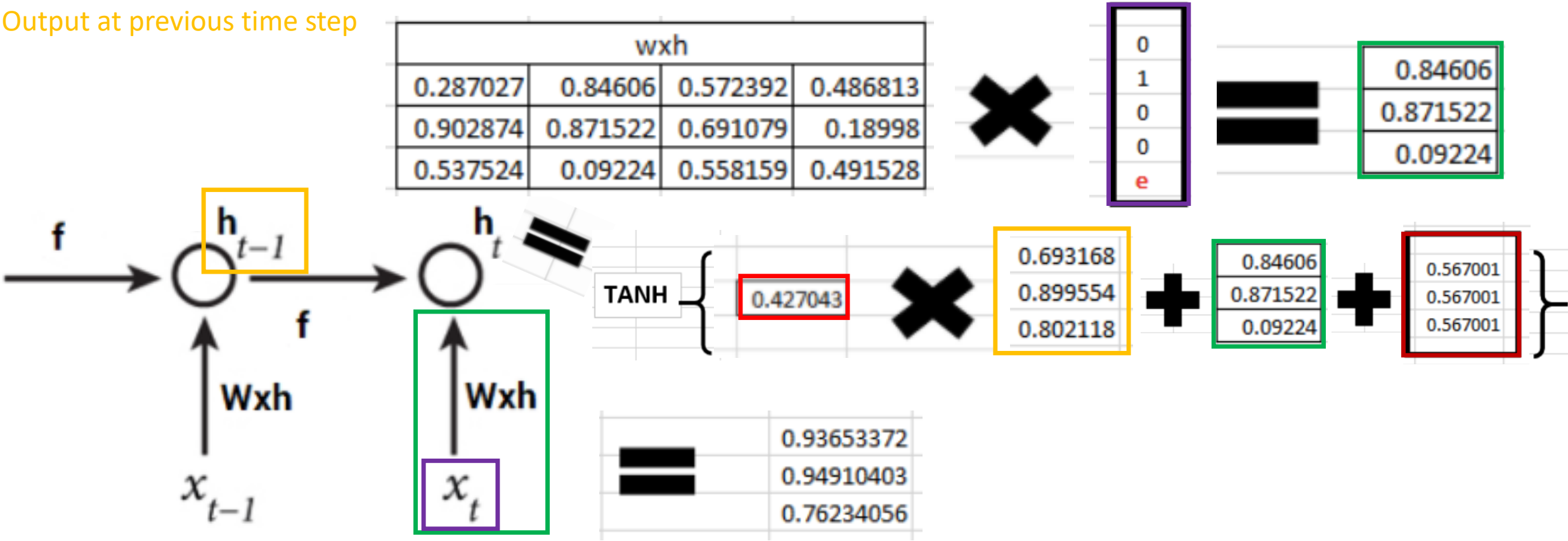
Example: Predict Sequence of Characters

Initialize to random value: 0.427043

Initialized to random value: 0.567001

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

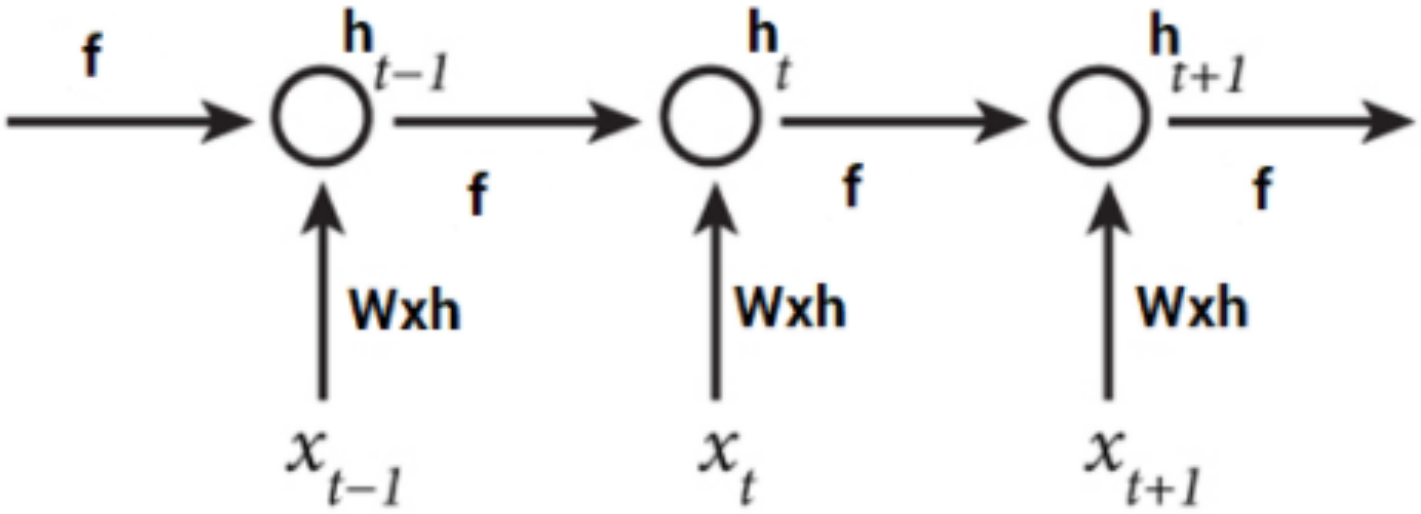
Output at previous time step



This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

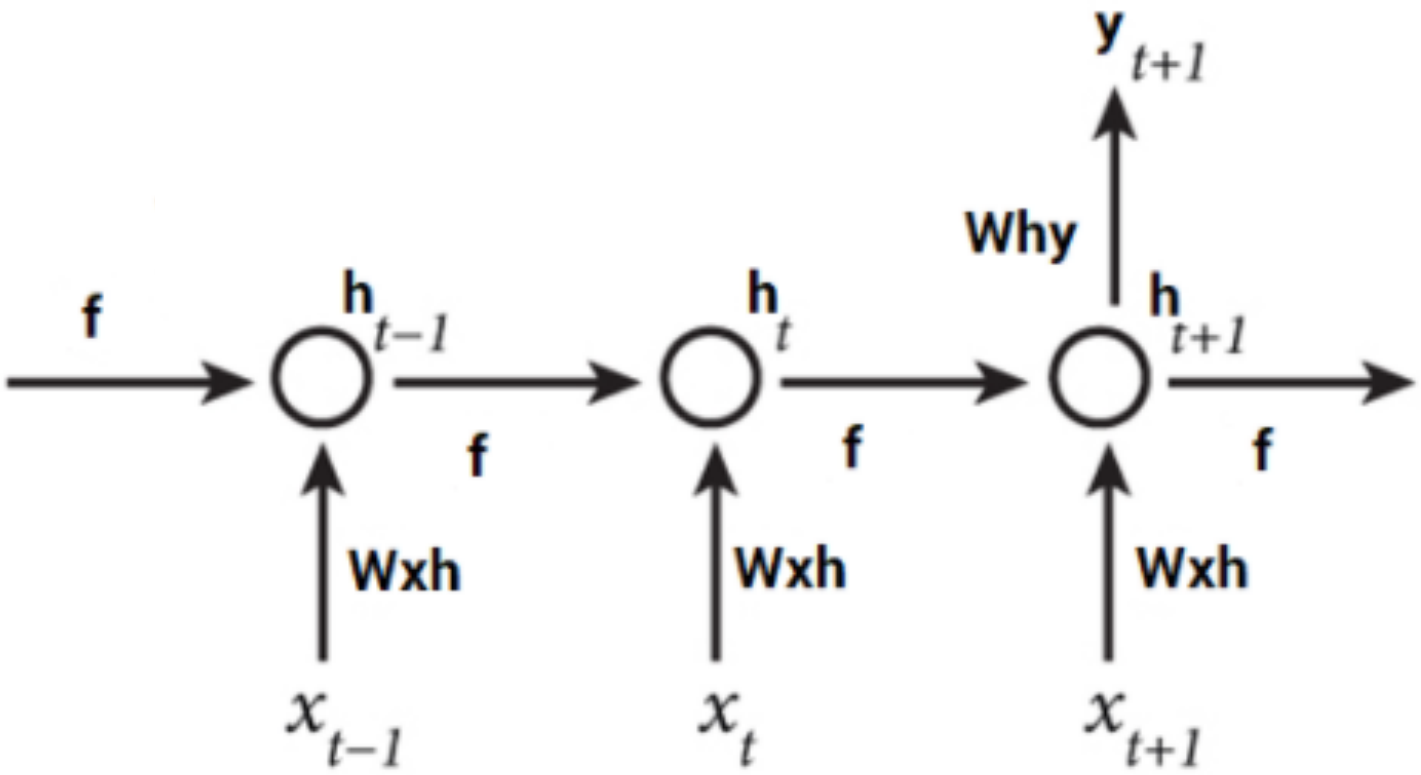
Example: Predict Sequence of Characters



This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

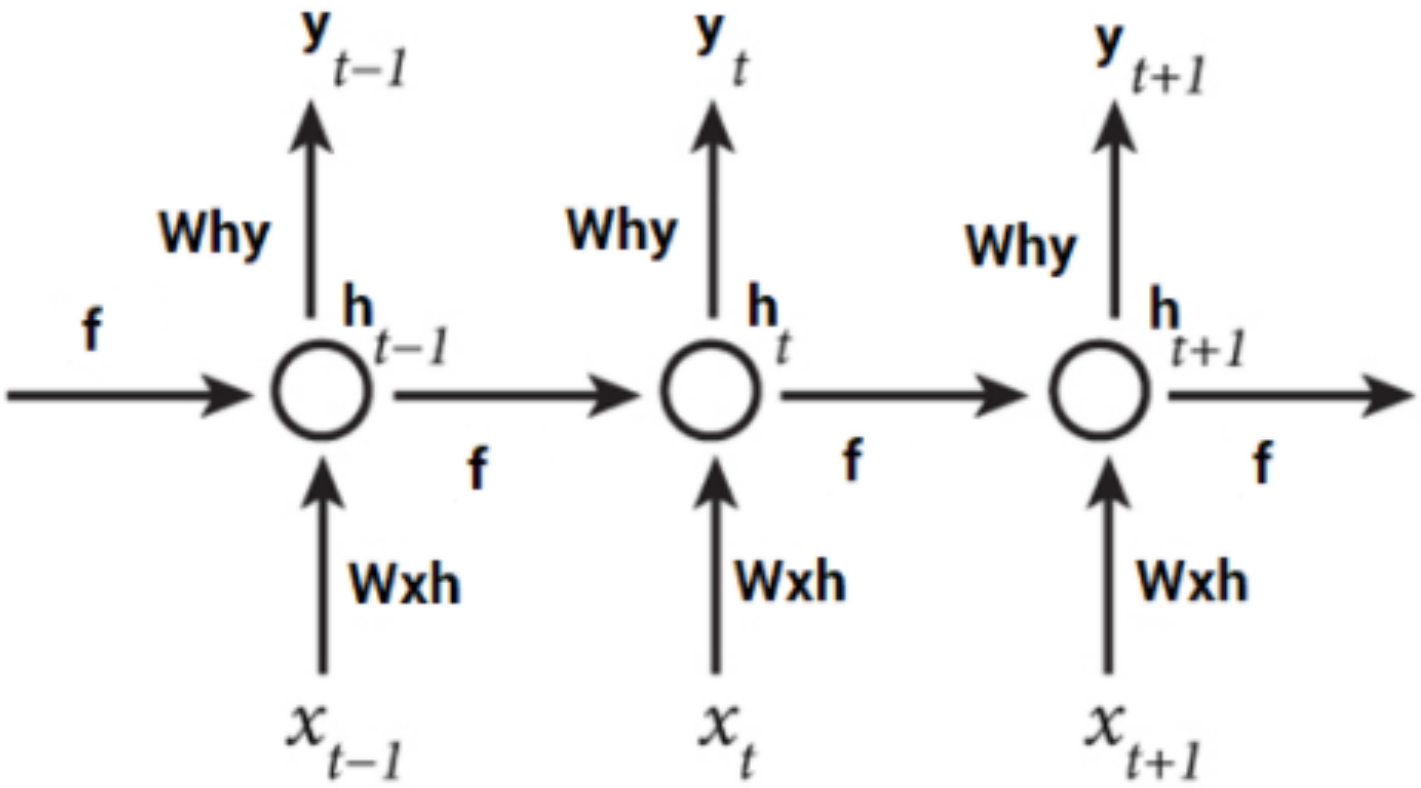
Example: Prediction (Many-To-One)



This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

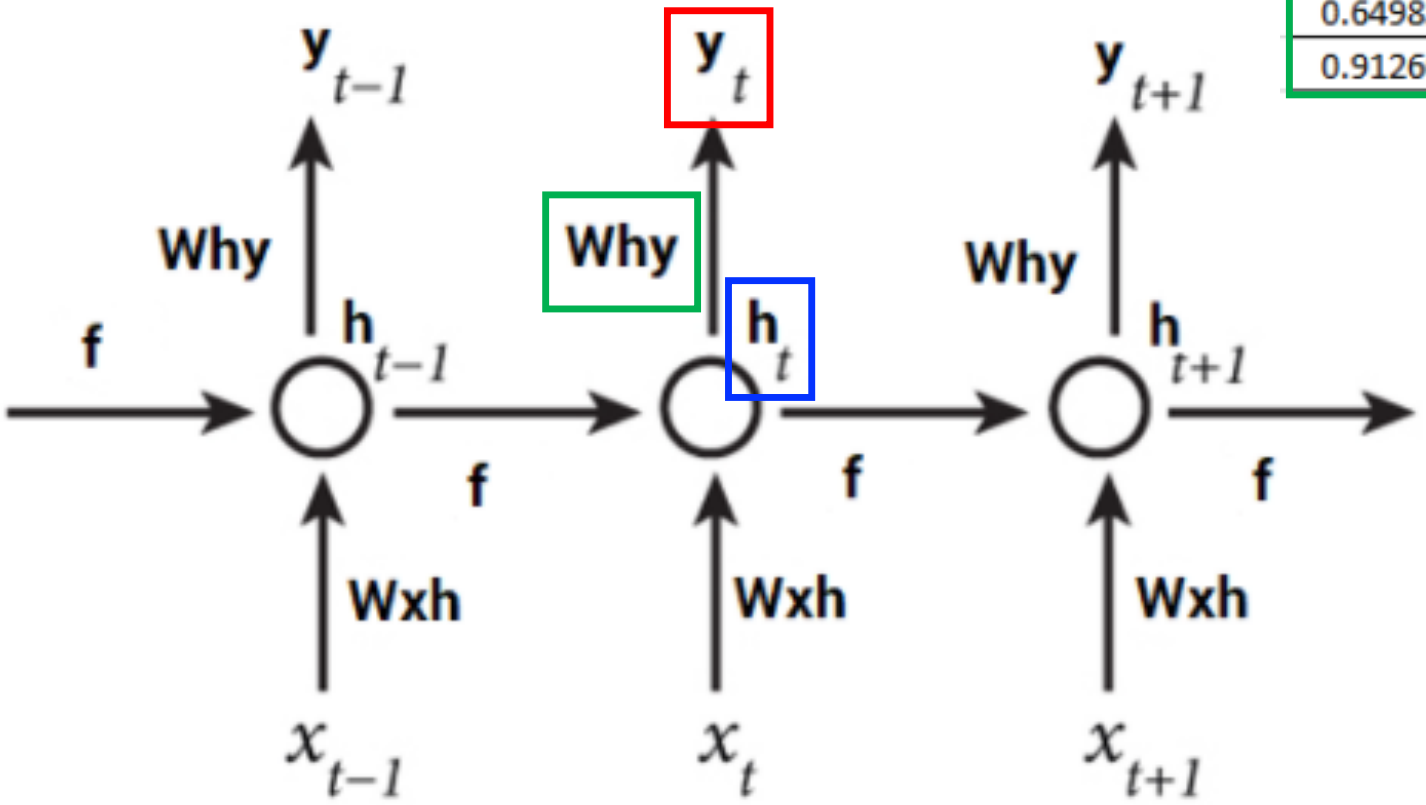
Example: Prediction (Many-To-Many)



This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

Example: Prediction for Time Step 2



why		
0.37168	0.974829459	0.830034886
0.39141	0.282585823	0.659835709
0.64985	0.09821557	0.334287084
0.91266	0.32581642	0.144630018

×

Ht
0.936534
0.949104
0.762341

=

yt
1.90607732
1.13779113
0.95666016
1.27422602

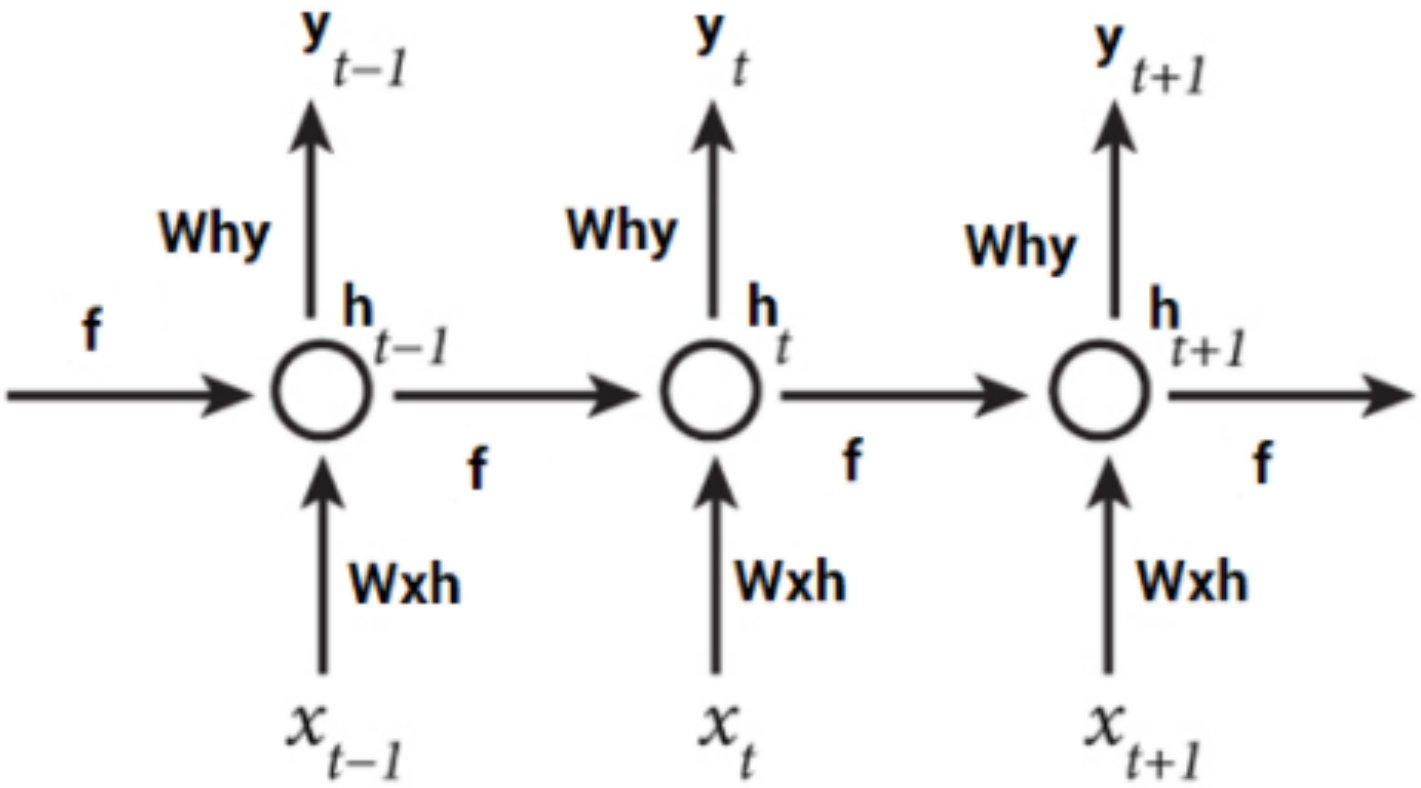
Applying softmax, to compute letter probabilities:

0.419748
0.194682
0.162429
0.223141

This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

Example: Prediction for Time Step 2



Given our vocabulary is {h, e, l, o}, what letter is predicted?

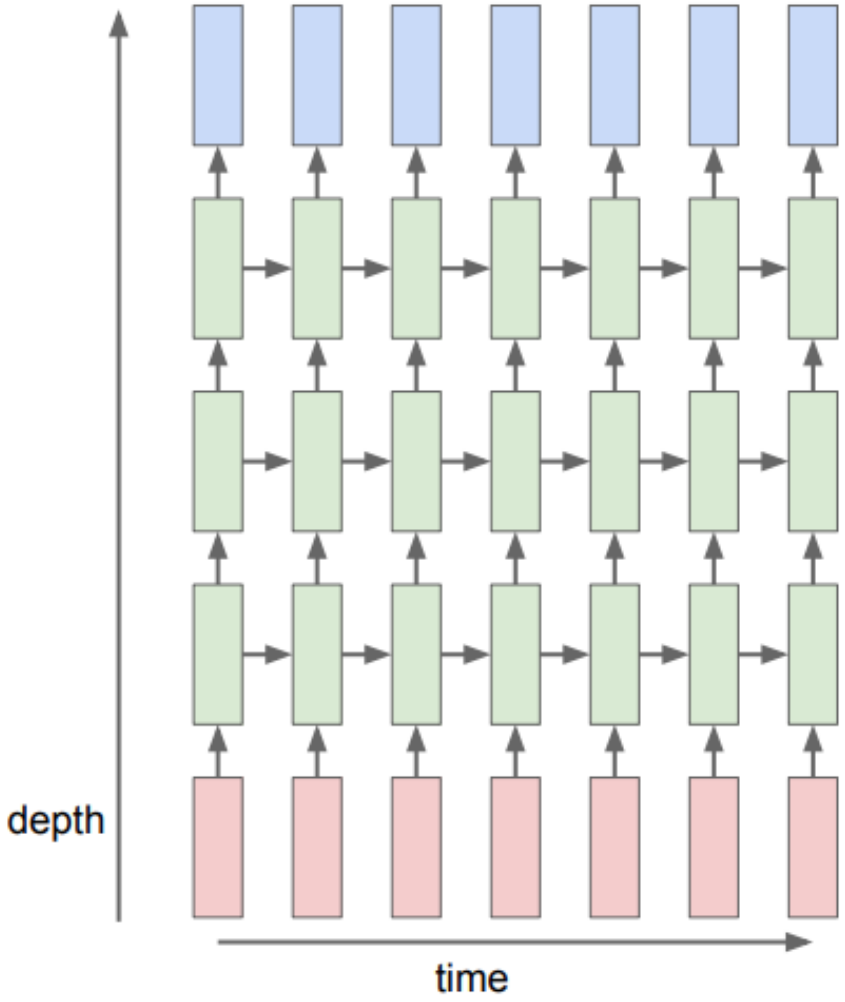
Applying softmax, to compute letter probabilities:

0.419748
0.194682
0.162429
0.223141

This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

RNN Variants: Different Number of Hidden Layers



Experimental evidence suggests deeper models can perform better:

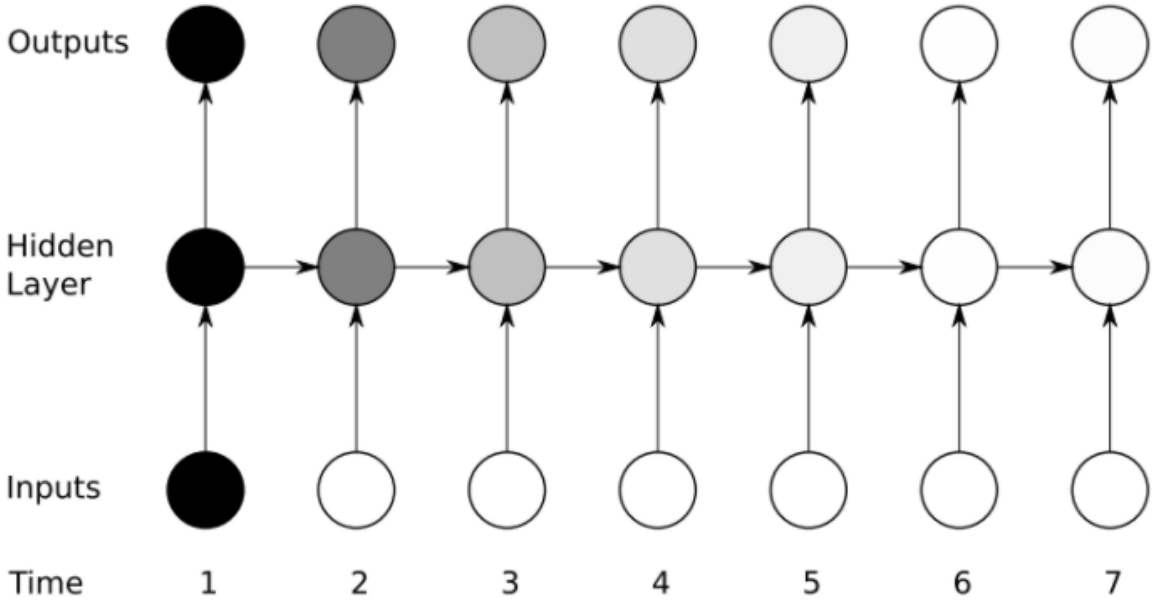
- Graves et al.; Speech Recognition with Deep Recurrent Neural Networks; 2013.
- Pascanu et al.; How to Construct Deep Recurrent Neural Networks; 2014.

This slide is taken from Danna Gurari

http://cs231n.stanford.edu/slides/2016/winter1516_lecture10.pdf

RNN: Vanishing Gradient Problem

- Problem: training to learn long-term dependencies
  - e.g., language: “In 2004, I started college” vs “I started college in 2004”



  - e.g., $\partial E/\partial W = \partial E/\partial y_3 * \partial y_3/\partial h_3 * \partial h_3/\partial y_2 * \partial y_2/\partial h_1$
- Vanishing gradient: a product of numbers less than 1 shrinks to zero
- Exploding gradient: a product of numbers greater than 1 explodes to infinity

This slide is taken from Danna Gurari

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/