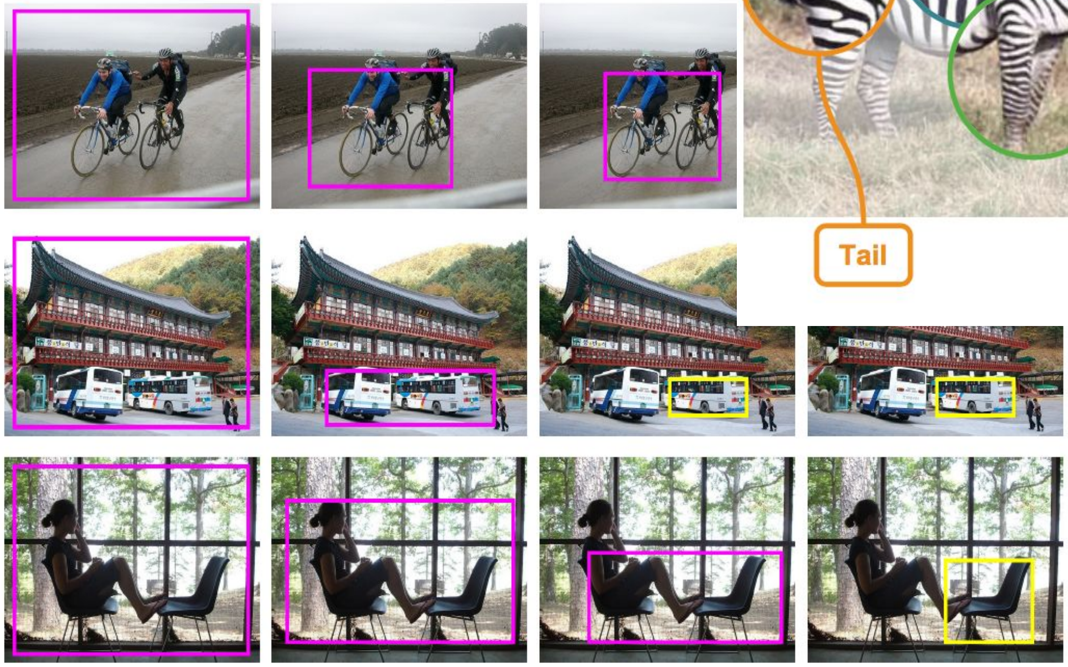
Evrişimsel ve Yinelemeli Sinir Ağları

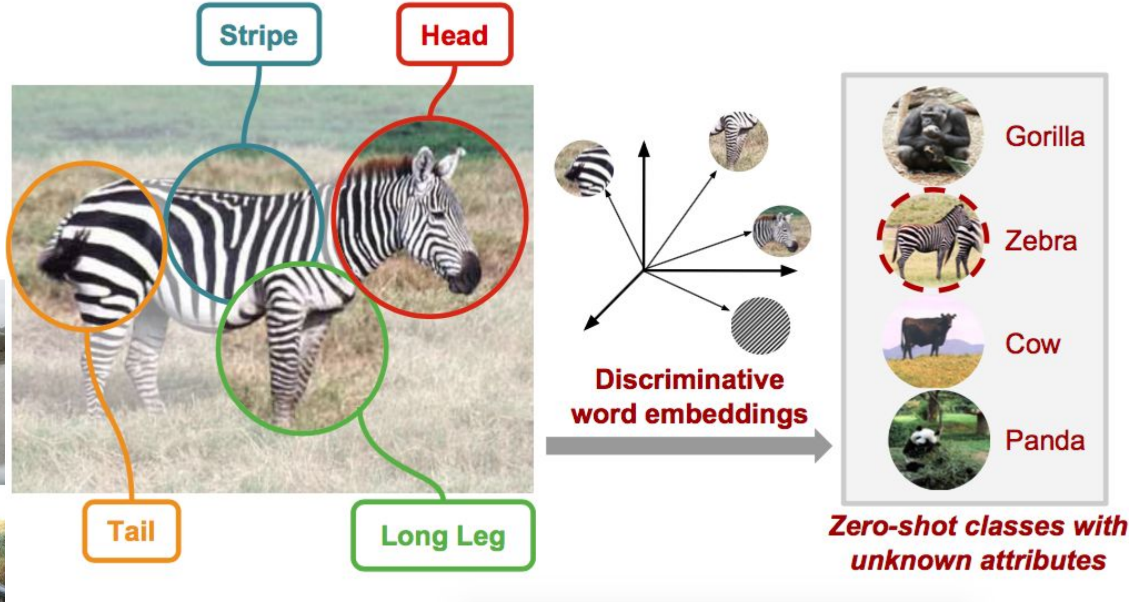
Bozkırda Yapay Öğrenme Yaz Okulu 2017

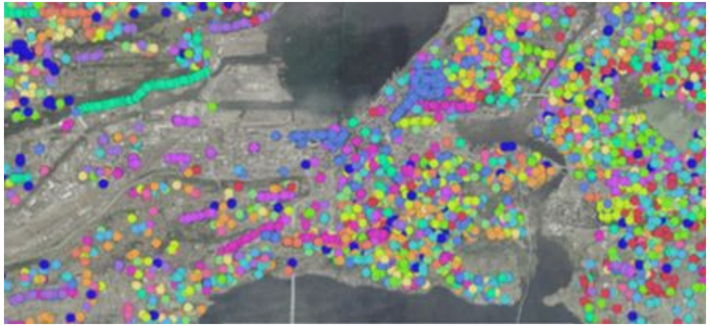
Gökberk Cinbiş

Araştırma hakkında.

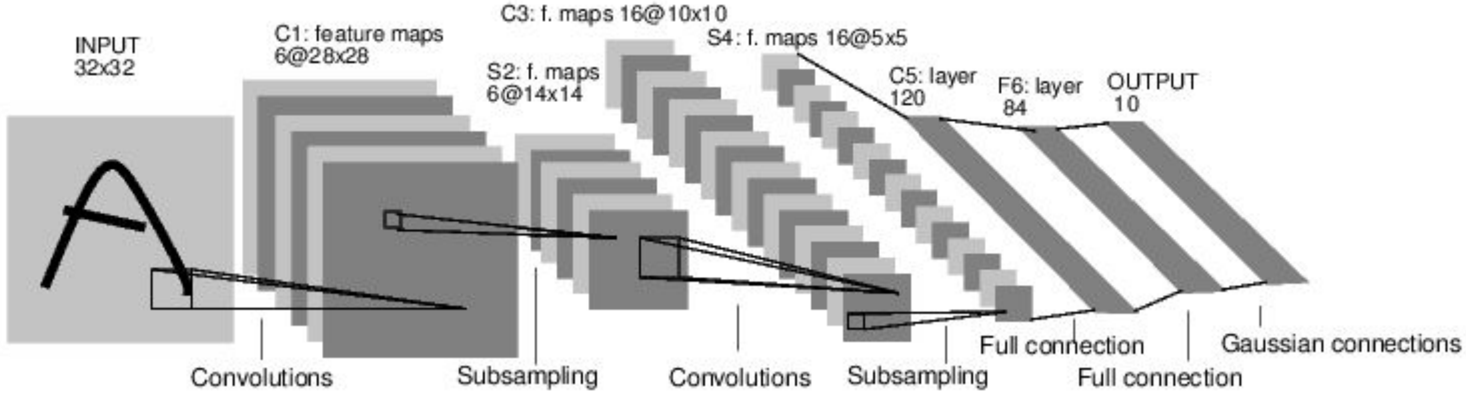


Initialization | Iteration 1 | Iteration 4 | Iteration 11





Evrişimsel Sinir Ağlarına Giriş



[LeNet-5, LeCun 1998]

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

A bit of history:

Hubel & Wiesel,

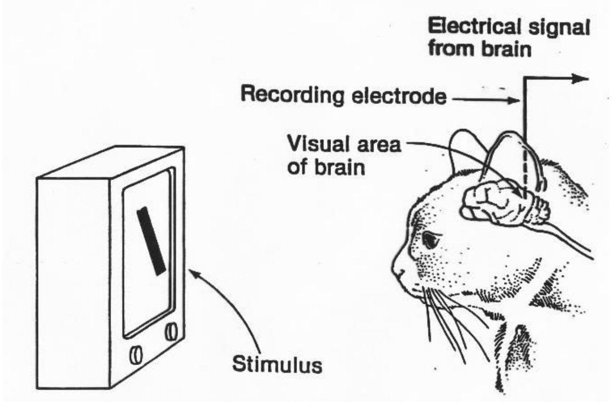
1959

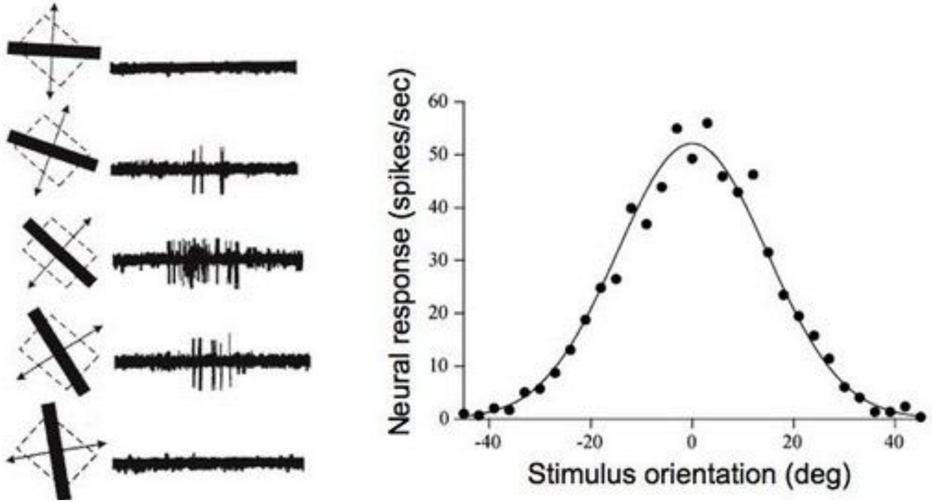
RECEPTIVE FIELDS OF SINGLE NEURONES IN
THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION
AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

1968...

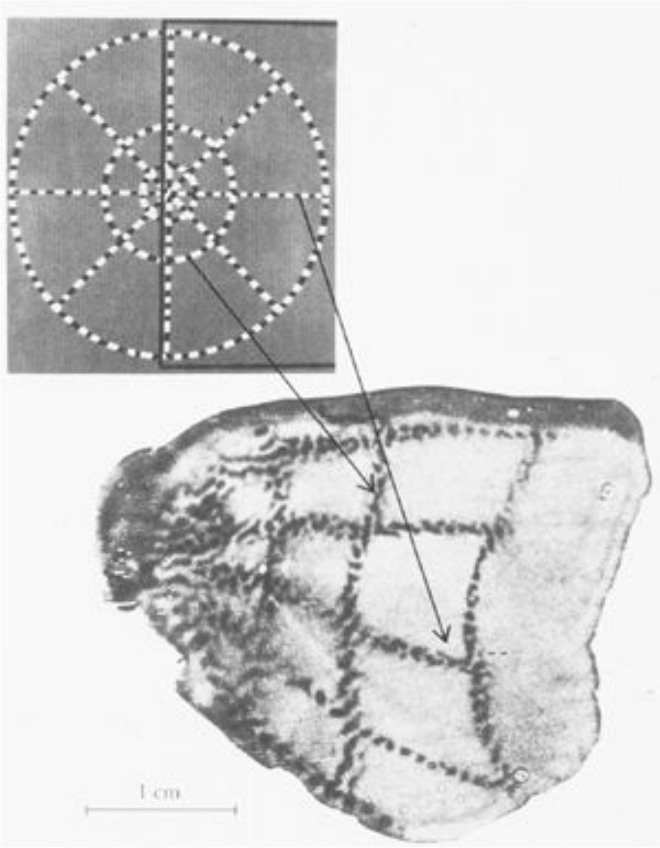




Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

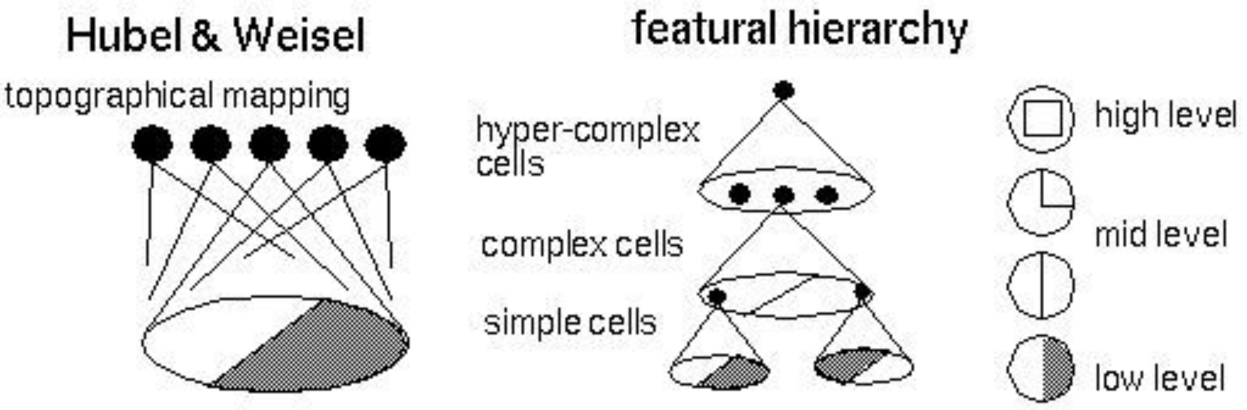
A bit of history

Topographical mapping in the cortex:
nearby cells in cortex represented
nearby regions in the visual field



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Hierarchical organization

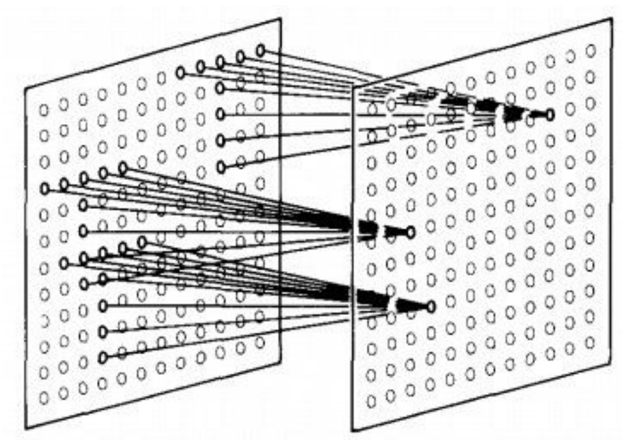


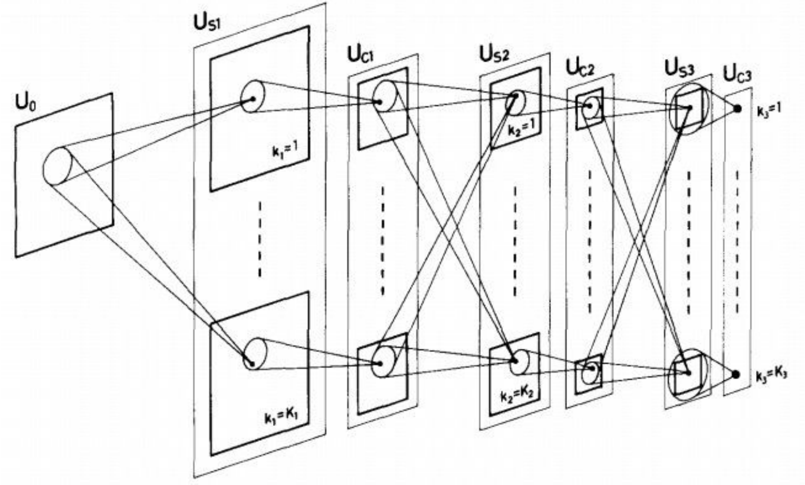
Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

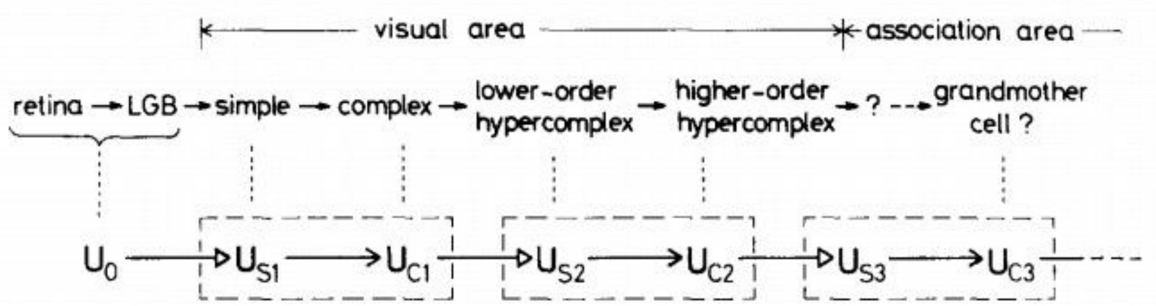
“sandwich” architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling

A bit of history:

Neurocognitron
[Fukushima 1980]





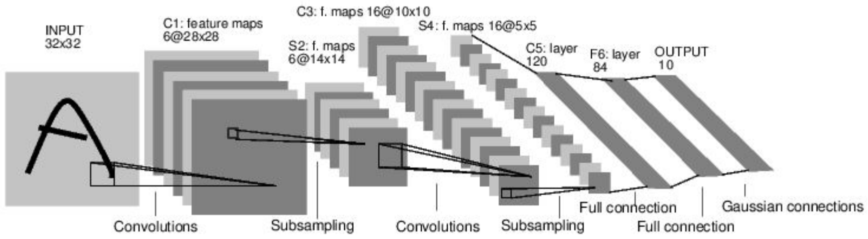


Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

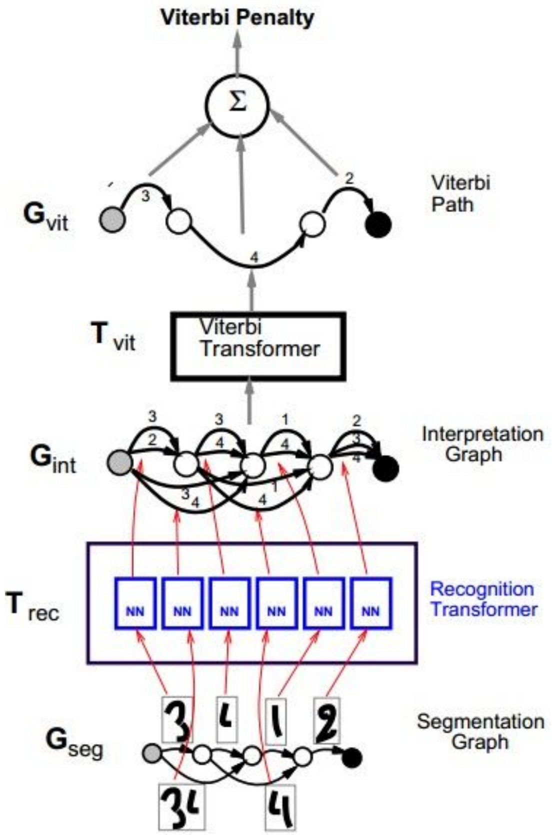
A bit of history:

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]



LeNet-5

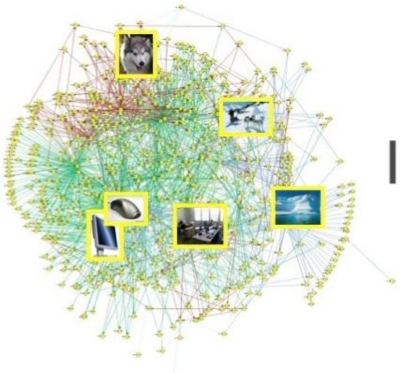


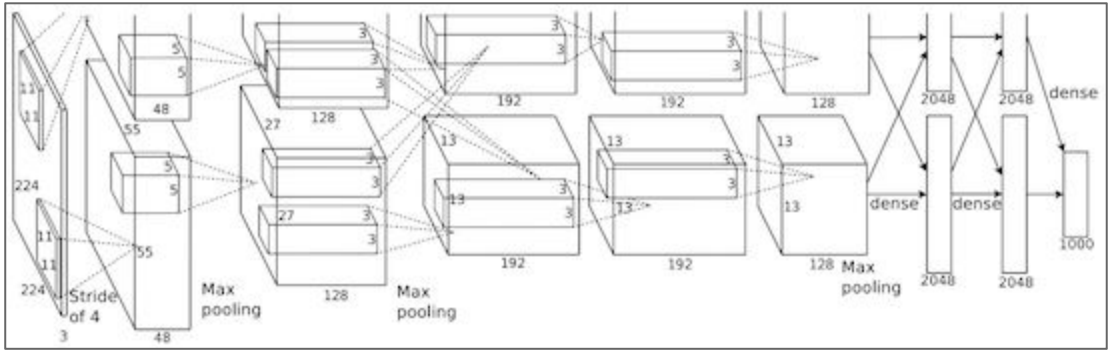
Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

A bit of history:

ImageNet Classification with Deep Convolutional Neural Networks

[Krizhevsky, Sutskever, Hinton, 2012]



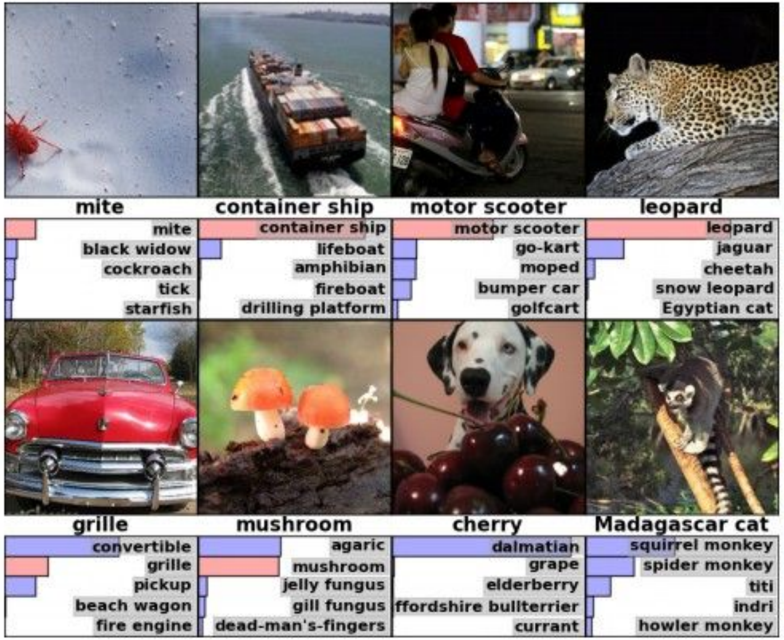


"AlexNet"

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Fast-forward to today: ConvNets are everywhere

Classification



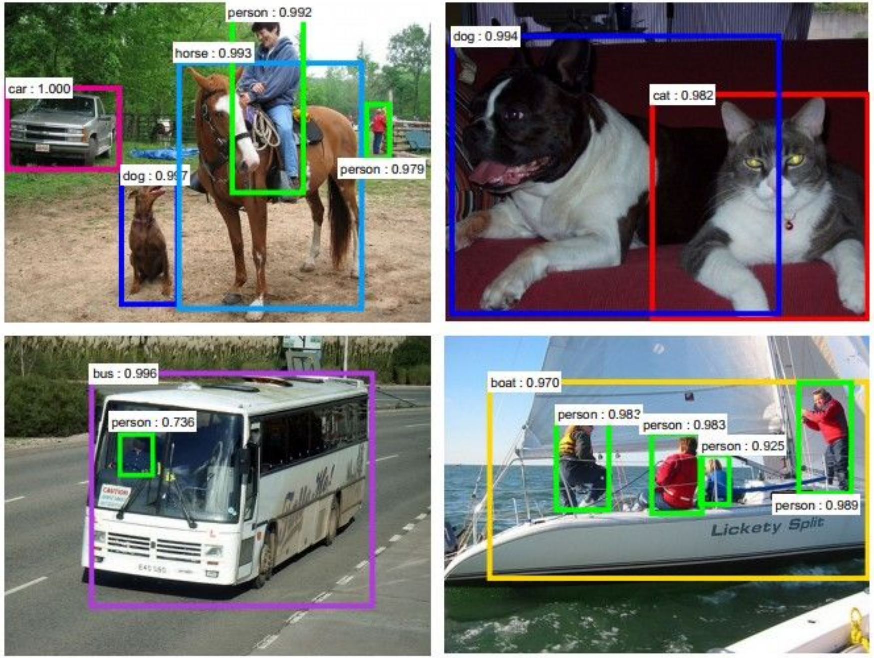
Retrieval

[Krizhevsky 2012]

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

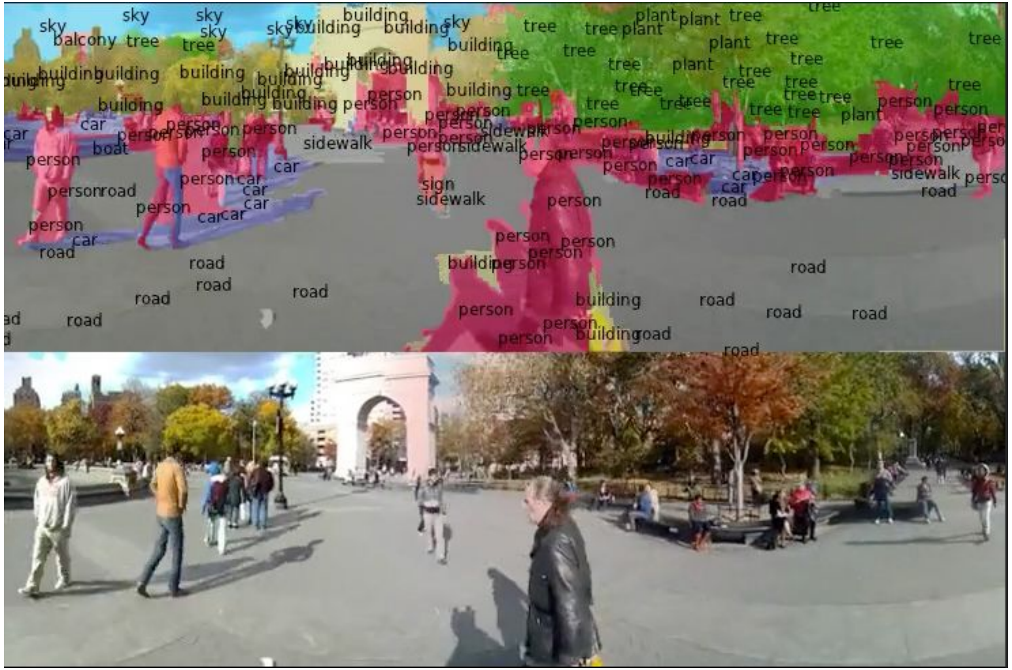
Fast-forward to today: ConvNets are everywhere

Detection



[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



[Farabet et al., 2012]

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Fast-forward to today: ConvNets are everywhere

self-driving cars

NVIDIA Tegra X1

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Fast-forward to today: ConvNets are everywhere

[Toshev, Szegedy 2014]

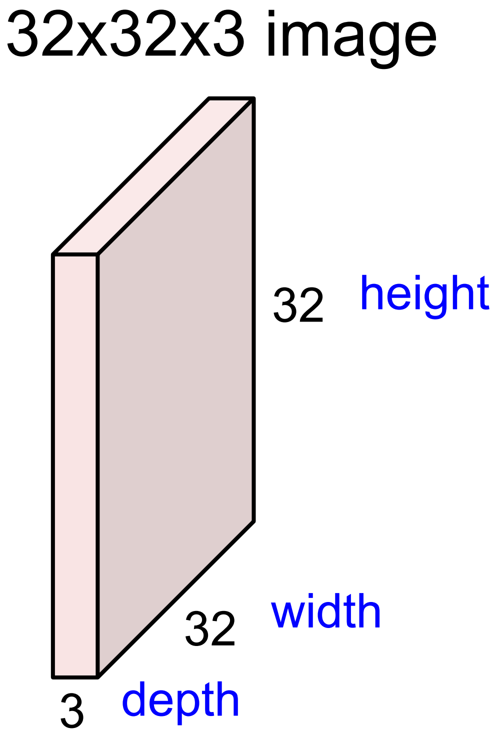
[Mnih 2013]

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

13

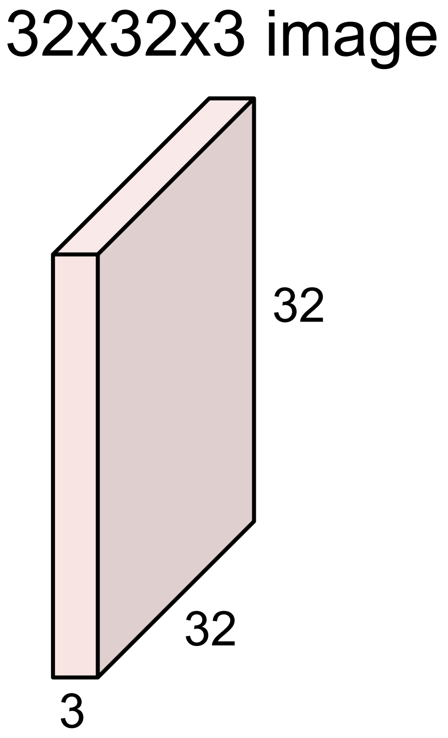
Evrişimsel Sinir Ağlarının Temelleri

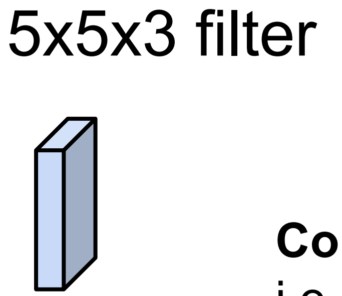
Convolution Layer



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Convolution Layer





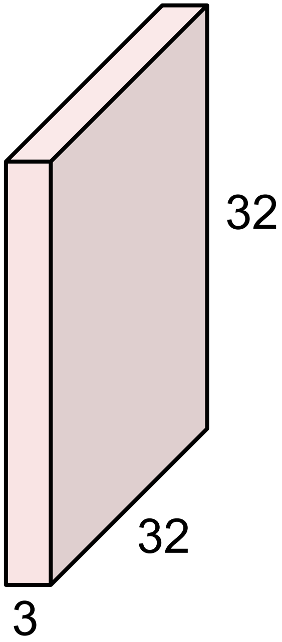
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

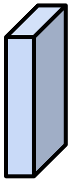
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



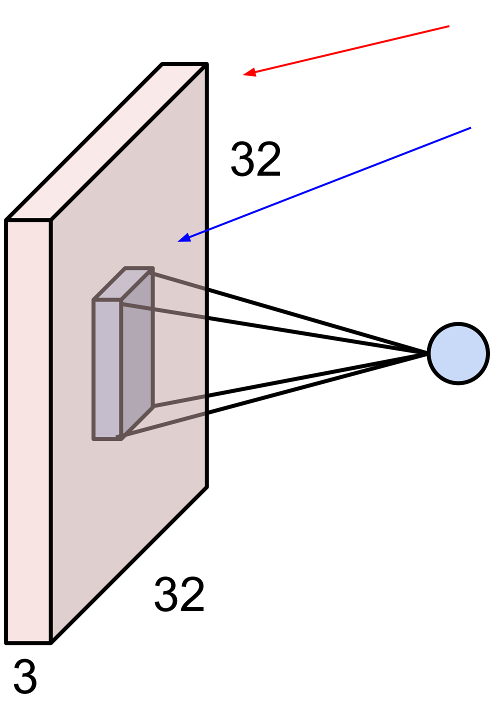
5x5x3 filter



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Convolution Layer

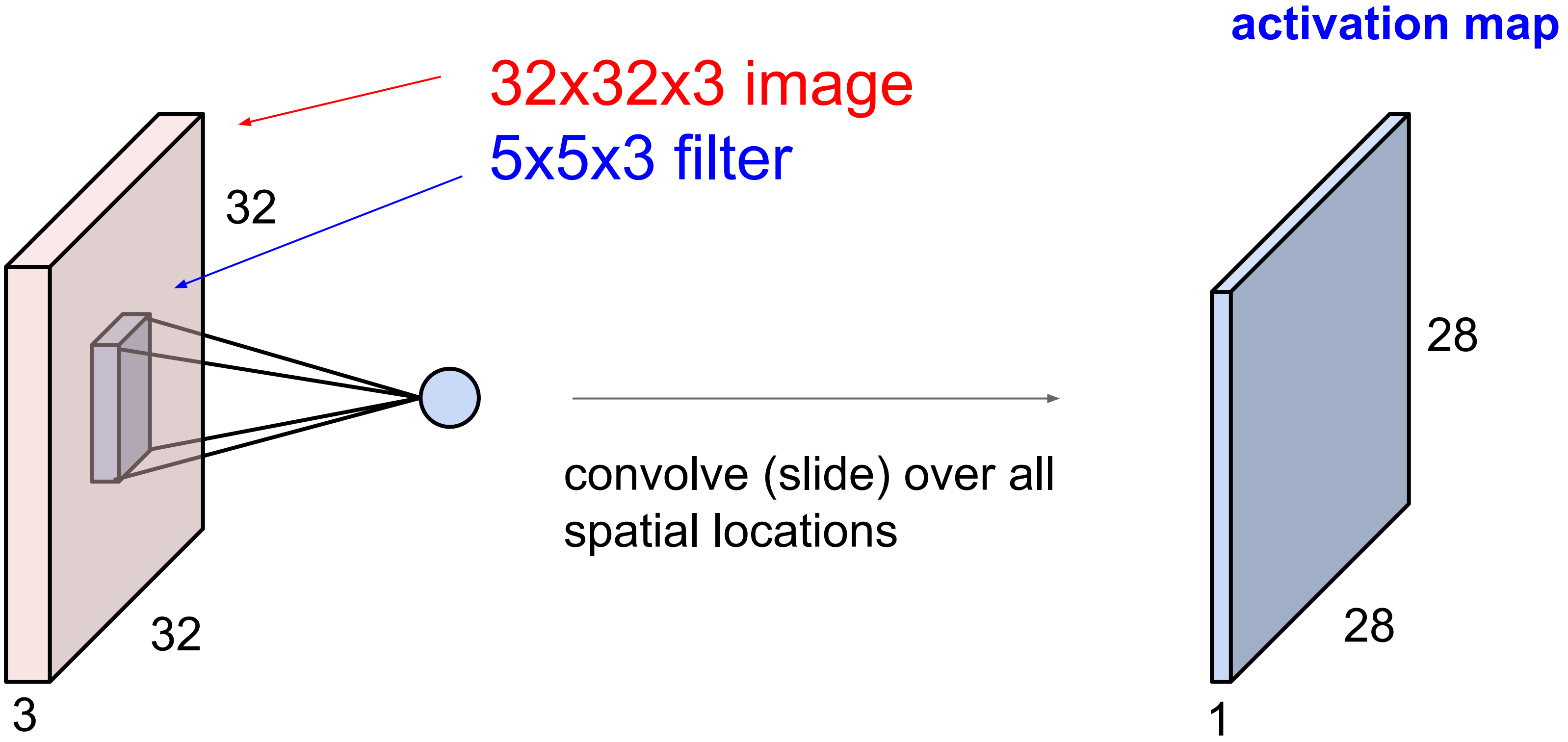


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

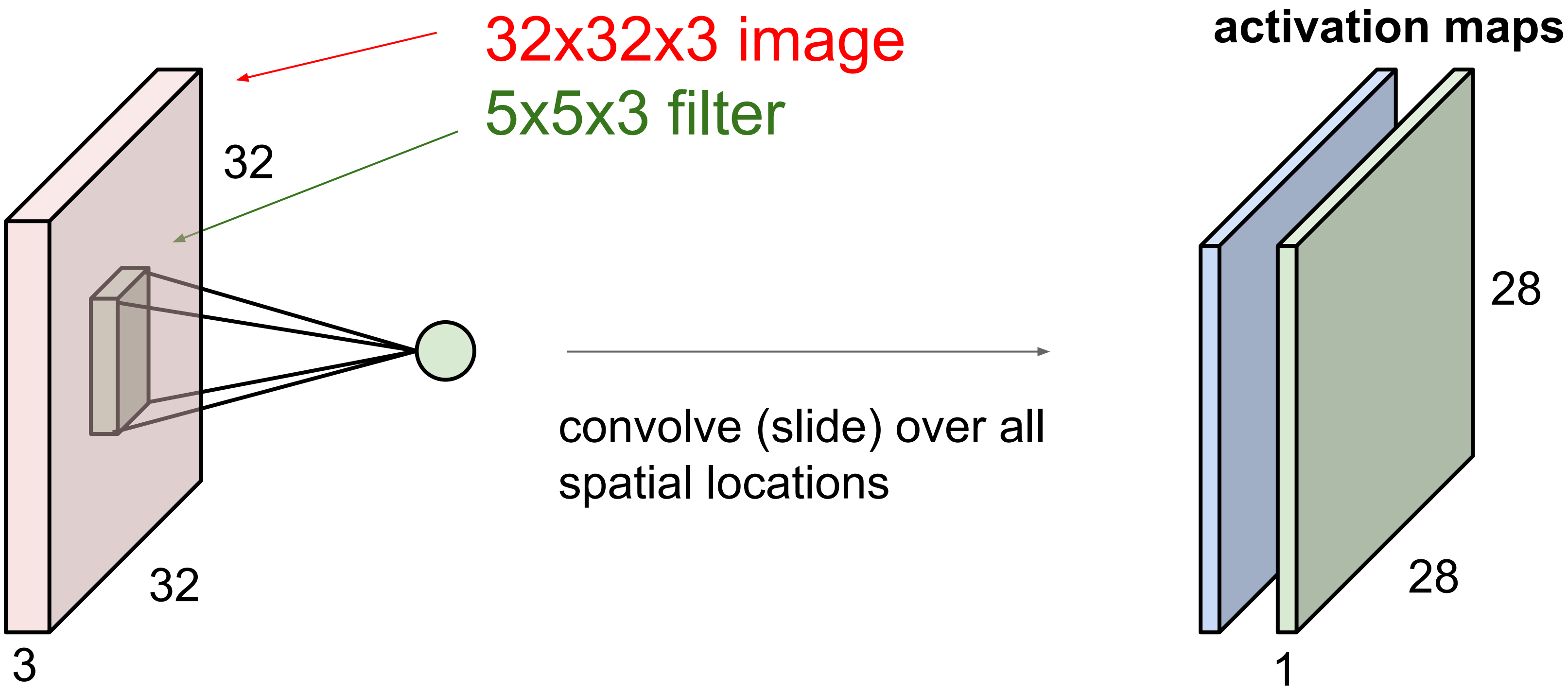
Convolution Layer



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

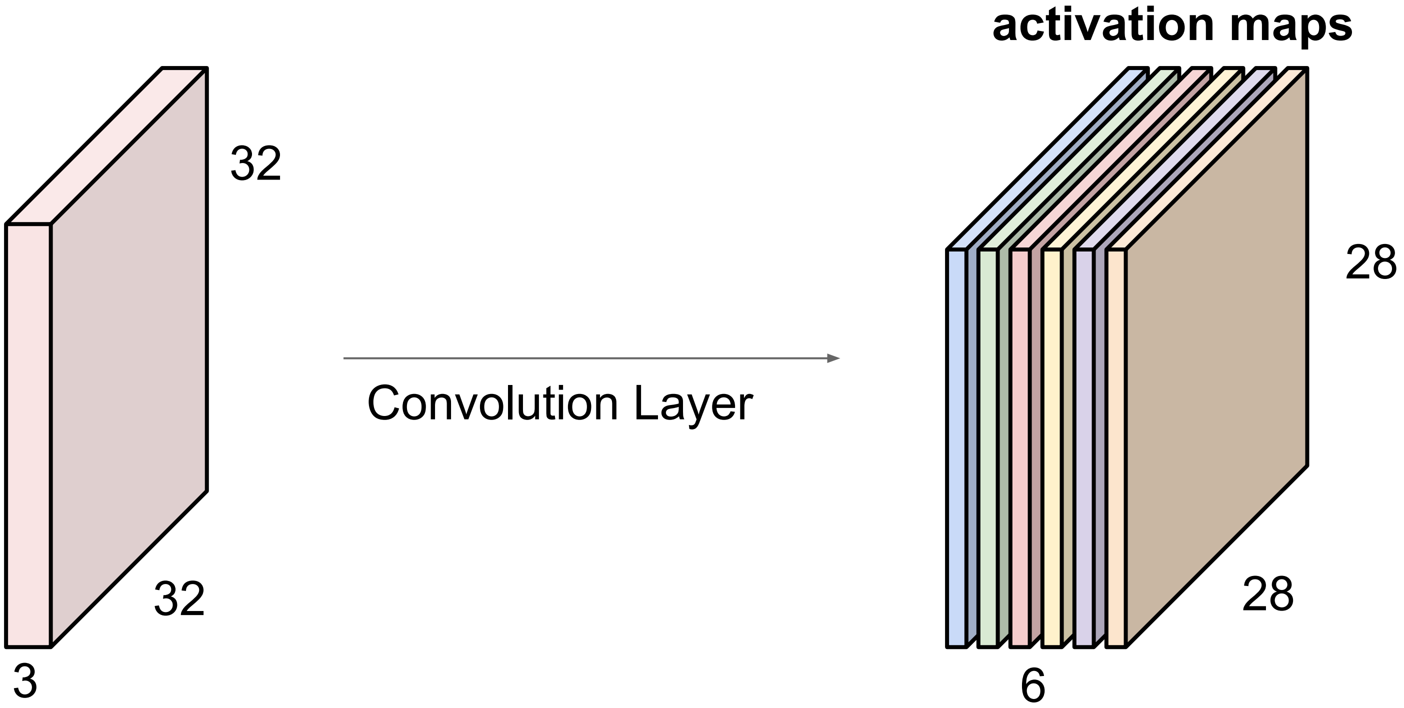
Convolution Layer

consider a second, green filter



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

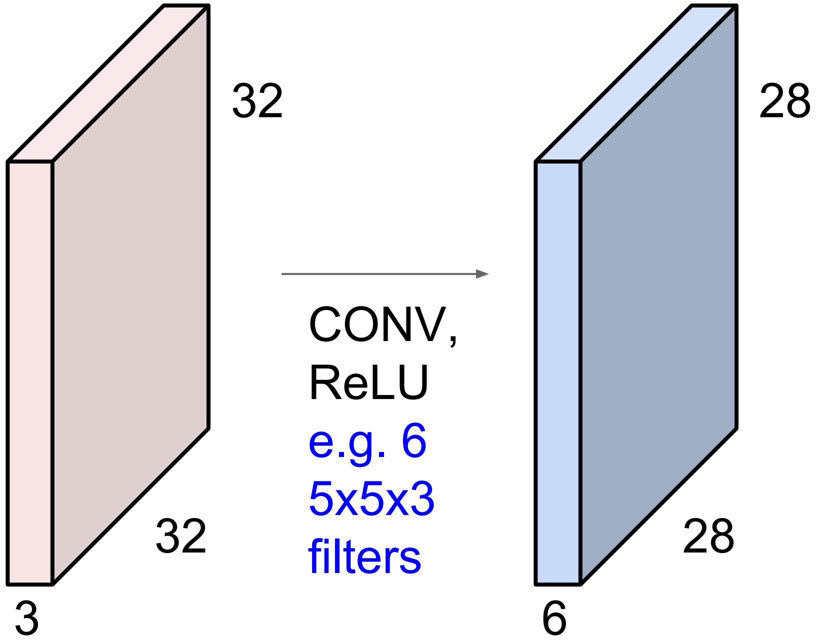
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

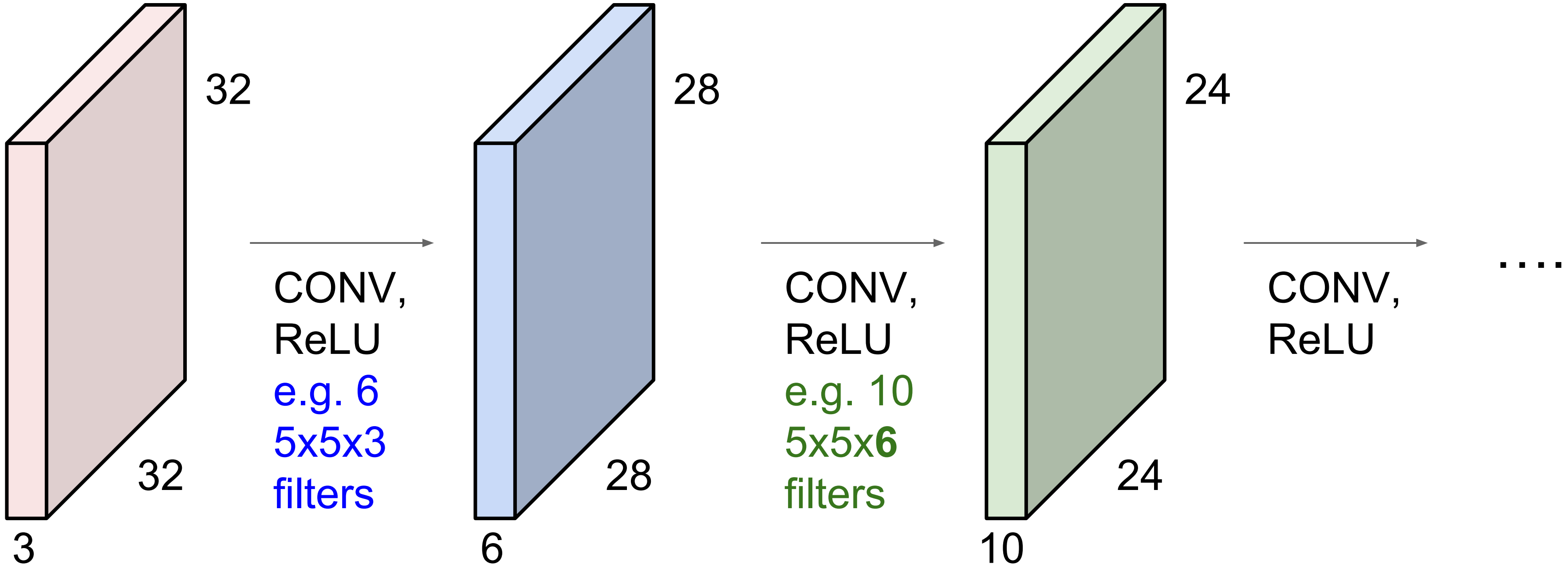
Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

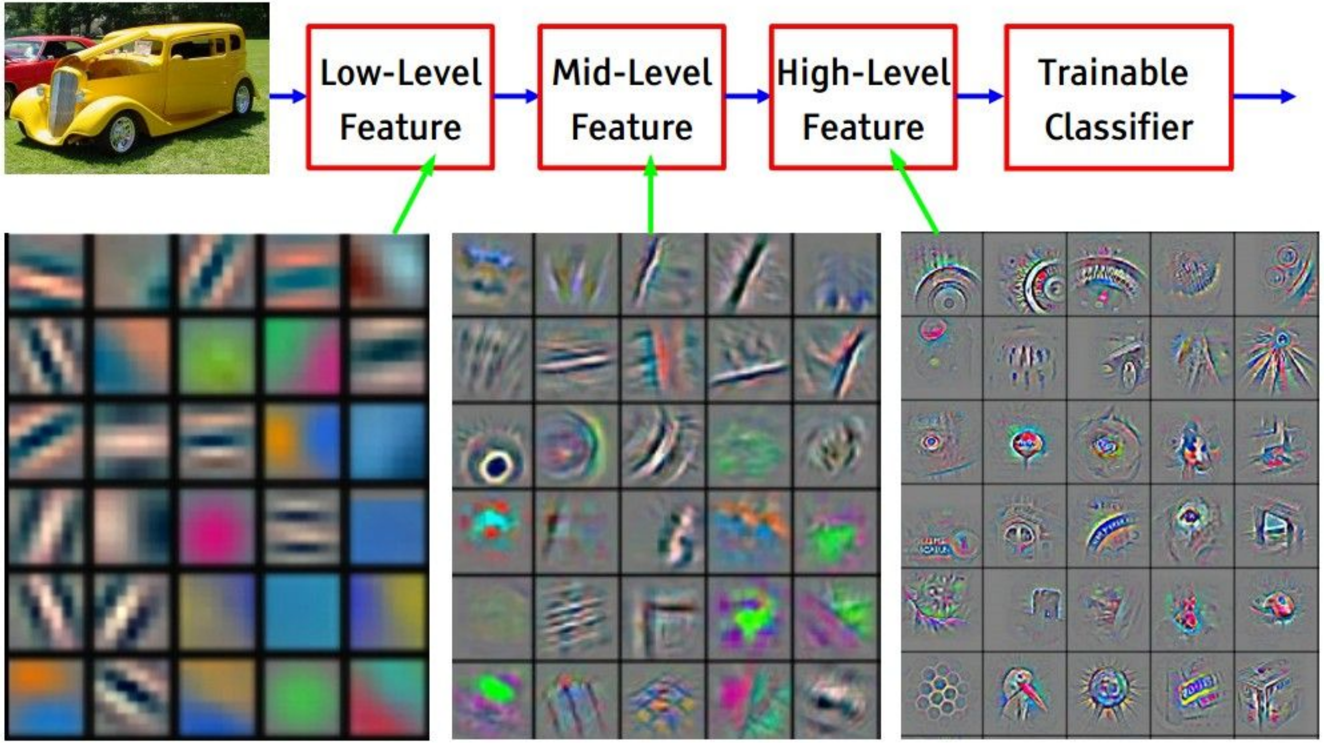
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Preview

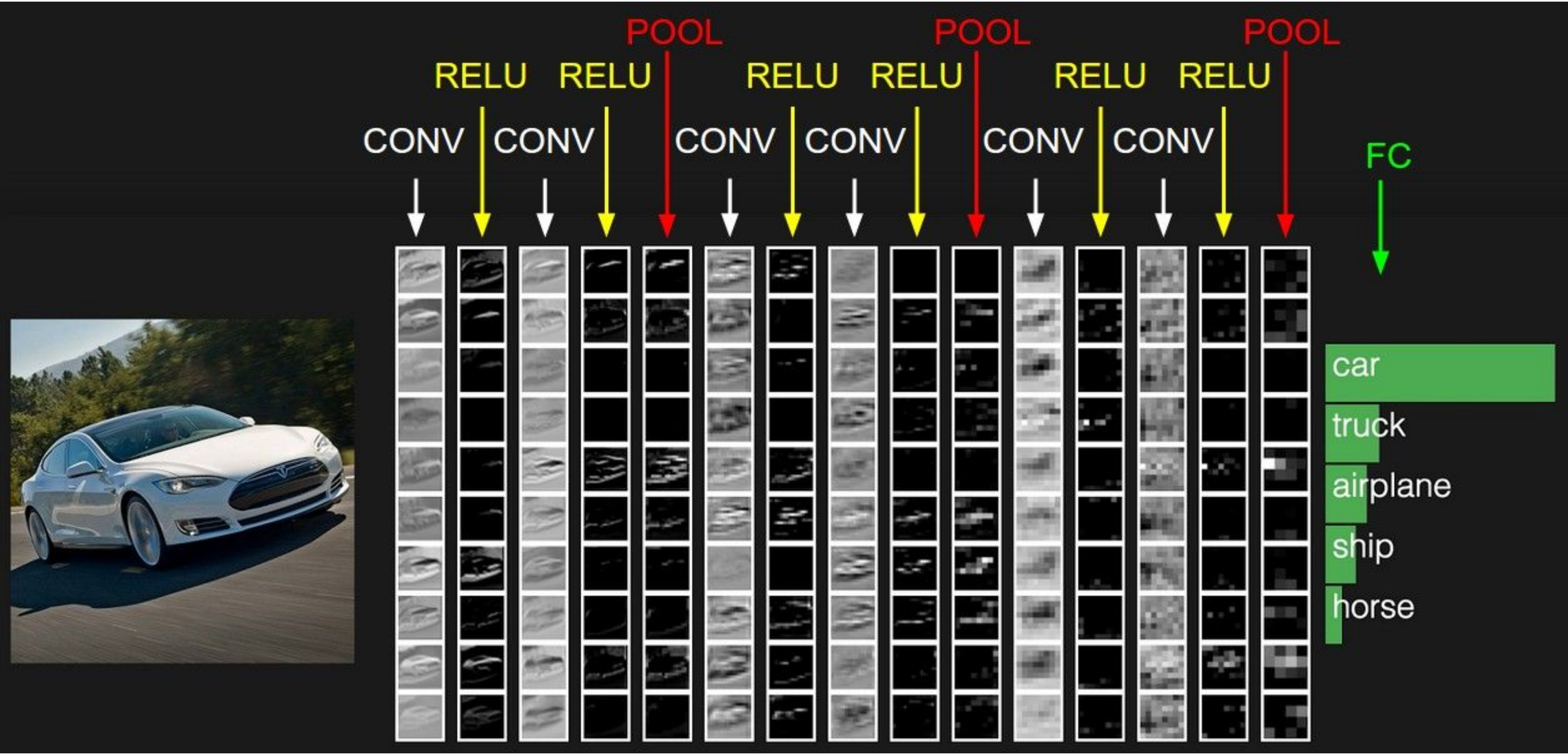
[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

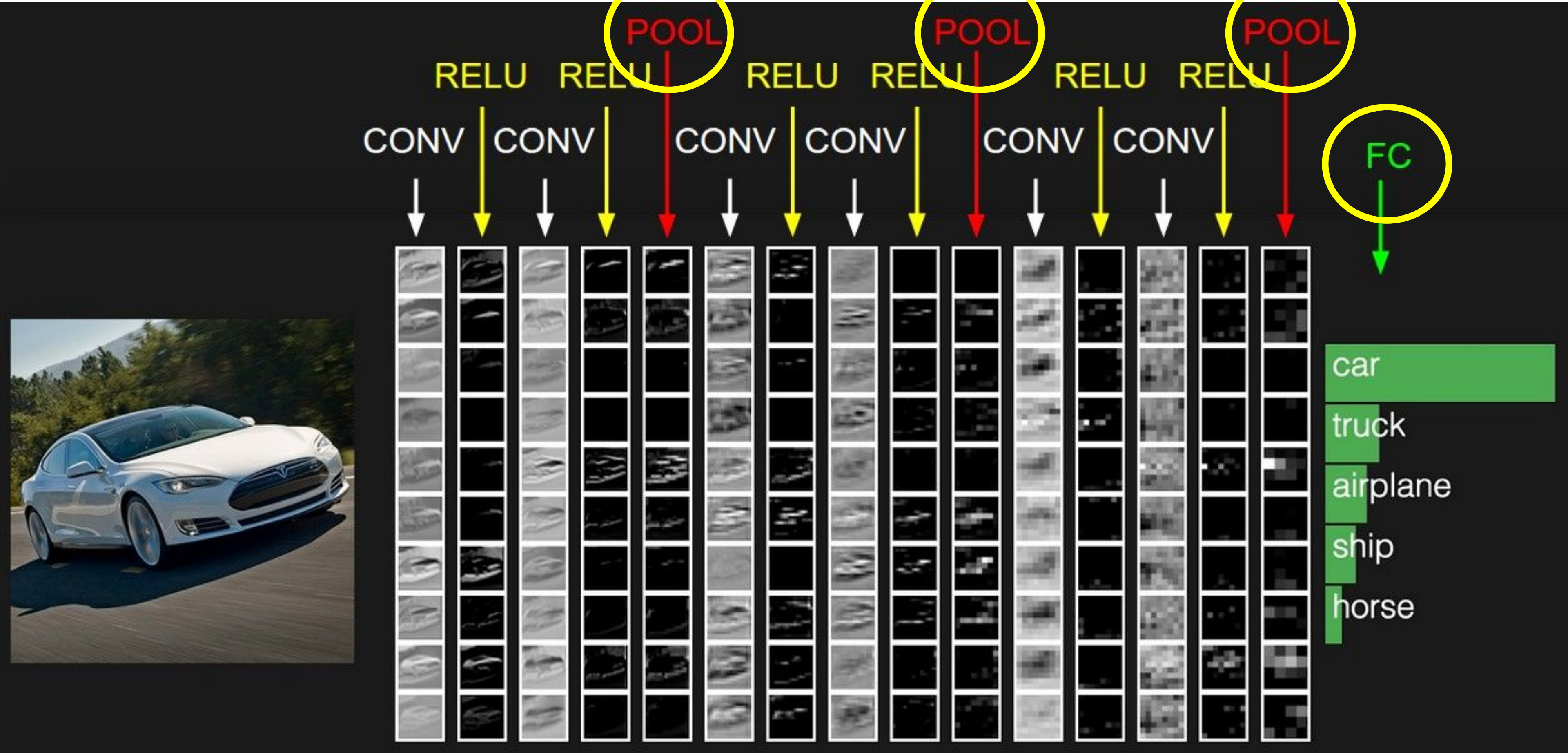
Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

preview:



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

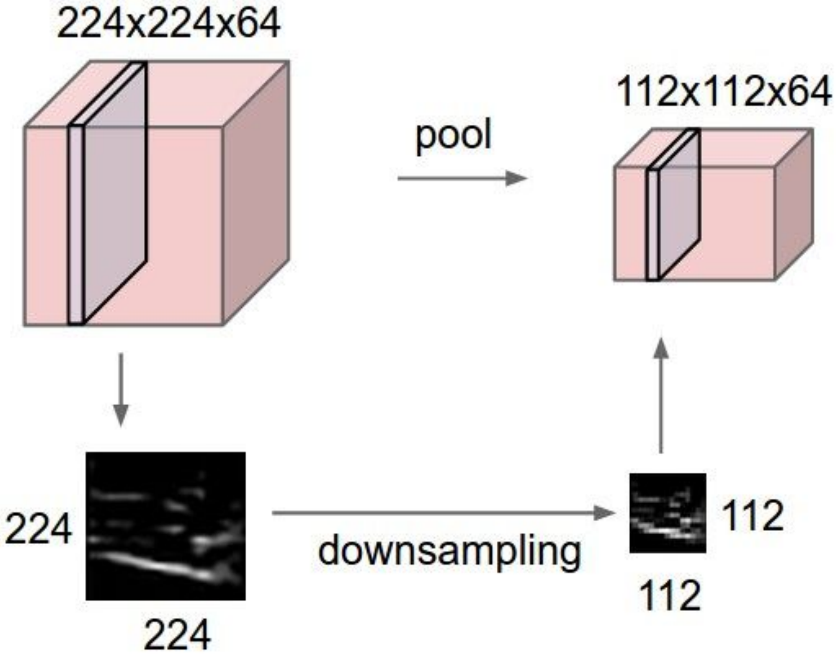
two more layers to go: **POOL**/**FC**



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

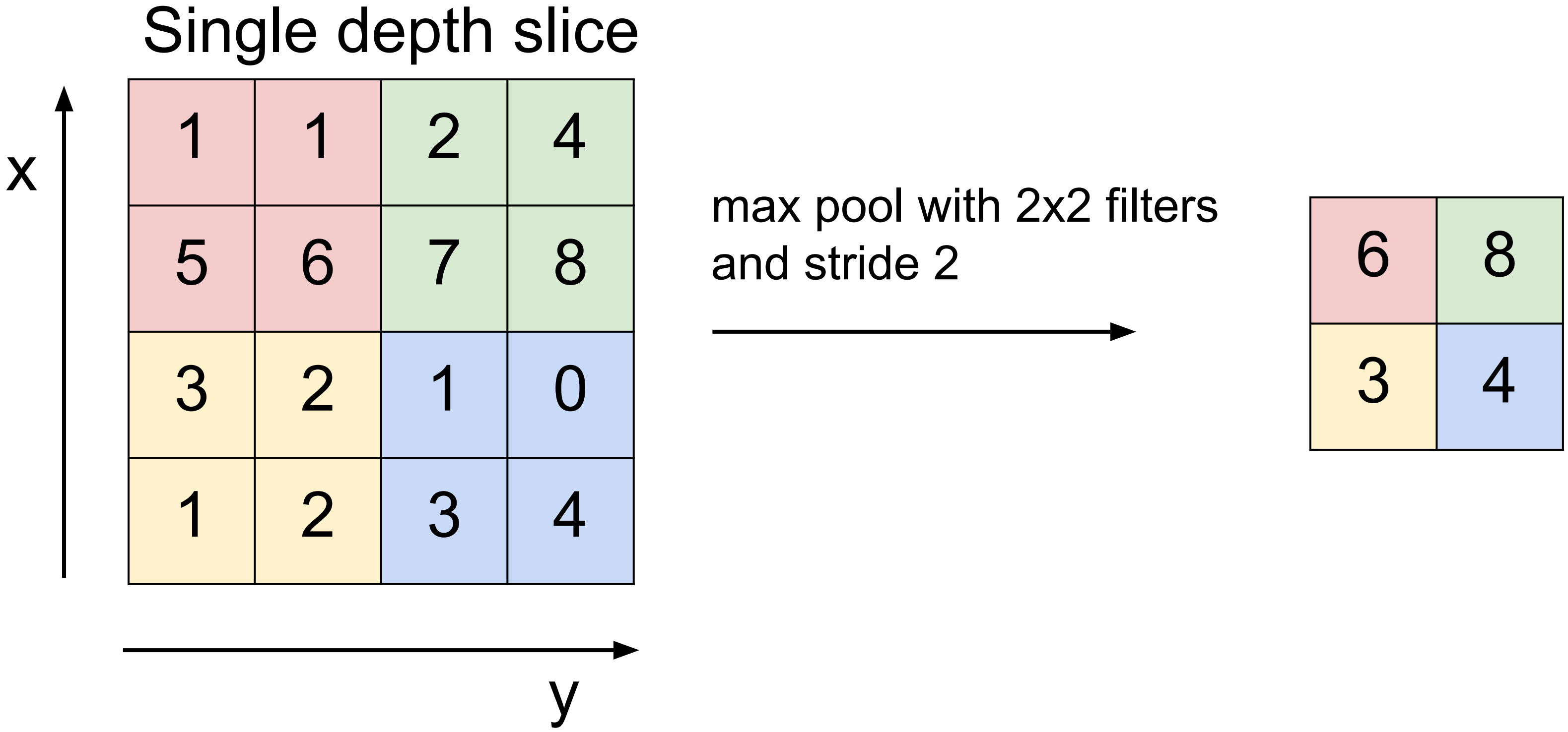
Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

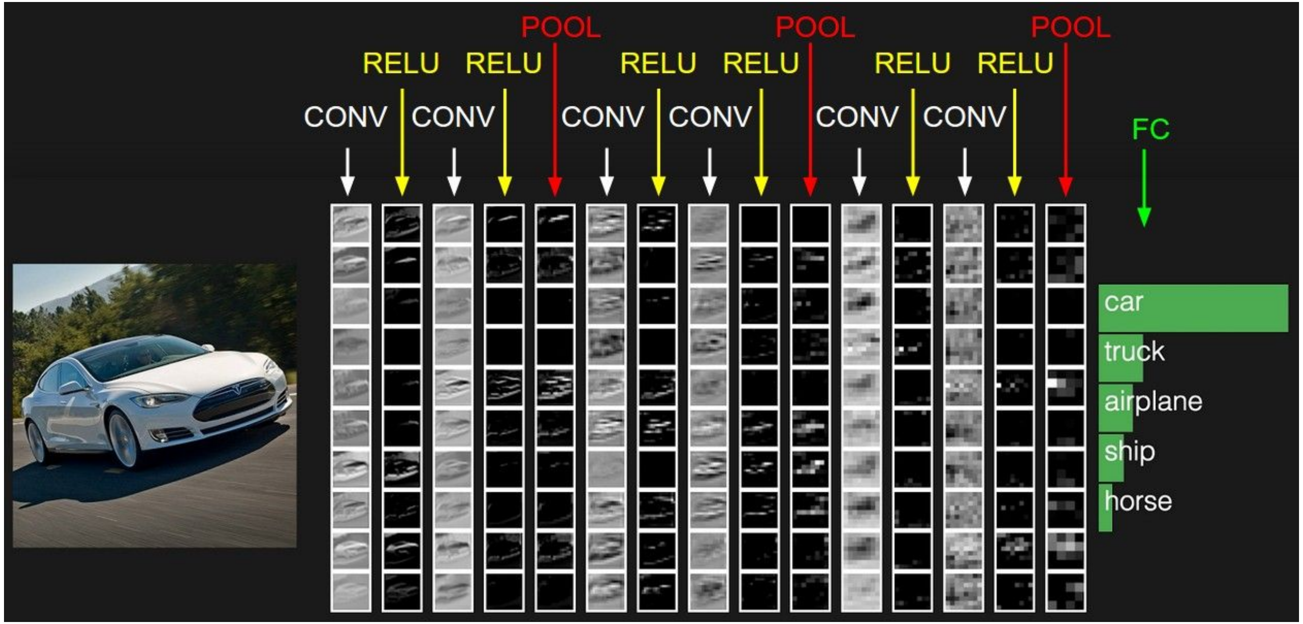
MAX POOLING



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

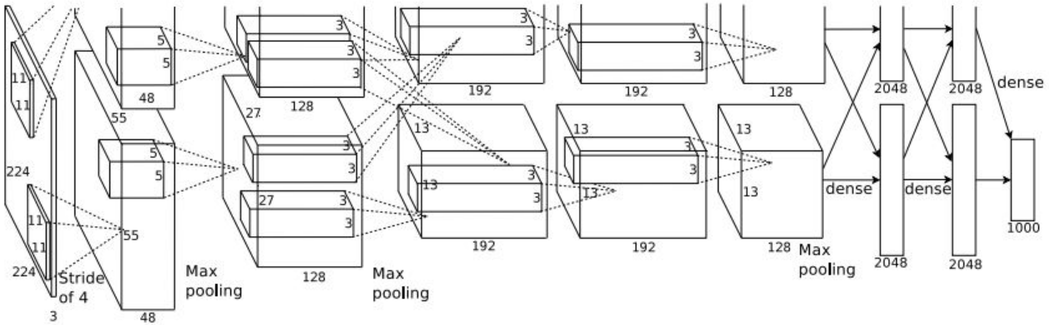


Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Evrişimsel Sinir Ağı Örnekleri

Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

- ReLU
- DropOut
- Data augmentation

Commonly used as feature representation

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

best model

16.4% (AlexNet), 11.2% (ZFNet) top 5 error in ILSVRC 2013
-> 7.3% top 5 error

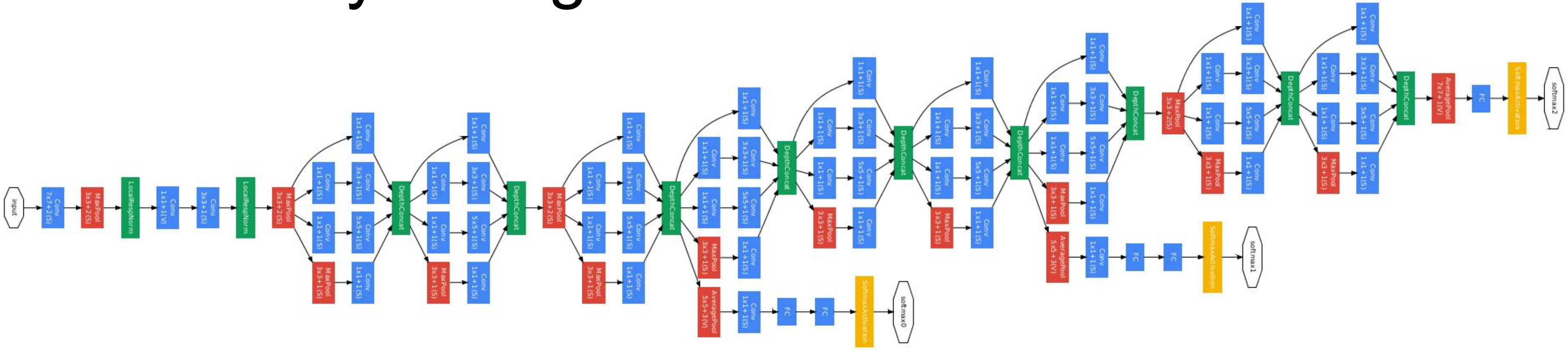
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

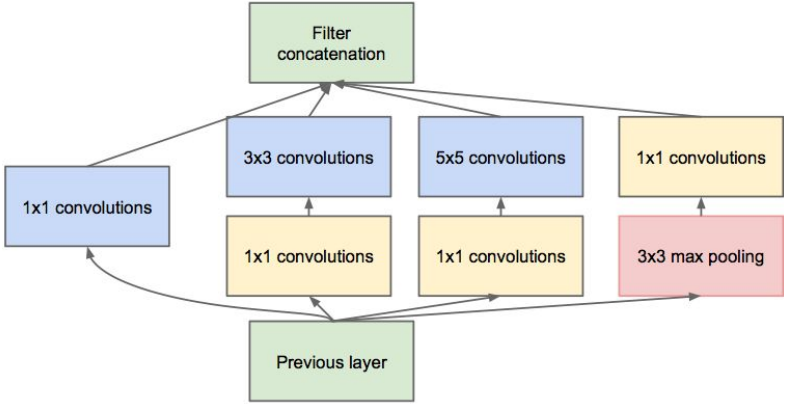
Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Case Study: GoogLeNet

[Szegedy et al., 2014]





Inception module

ILSVRC 2014 winner (6.7% top 5 error)

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Case Study: GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Fun features:

- Only 5 million params!
(Removes FC layers completely)

Compared to AlexNet:
- 12X less params
- 2x more compute
- 6.67% (vs. 16.4%)

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)

Microsoft Research

MSRA @ ILSVRC & COCO 2015 Competitions

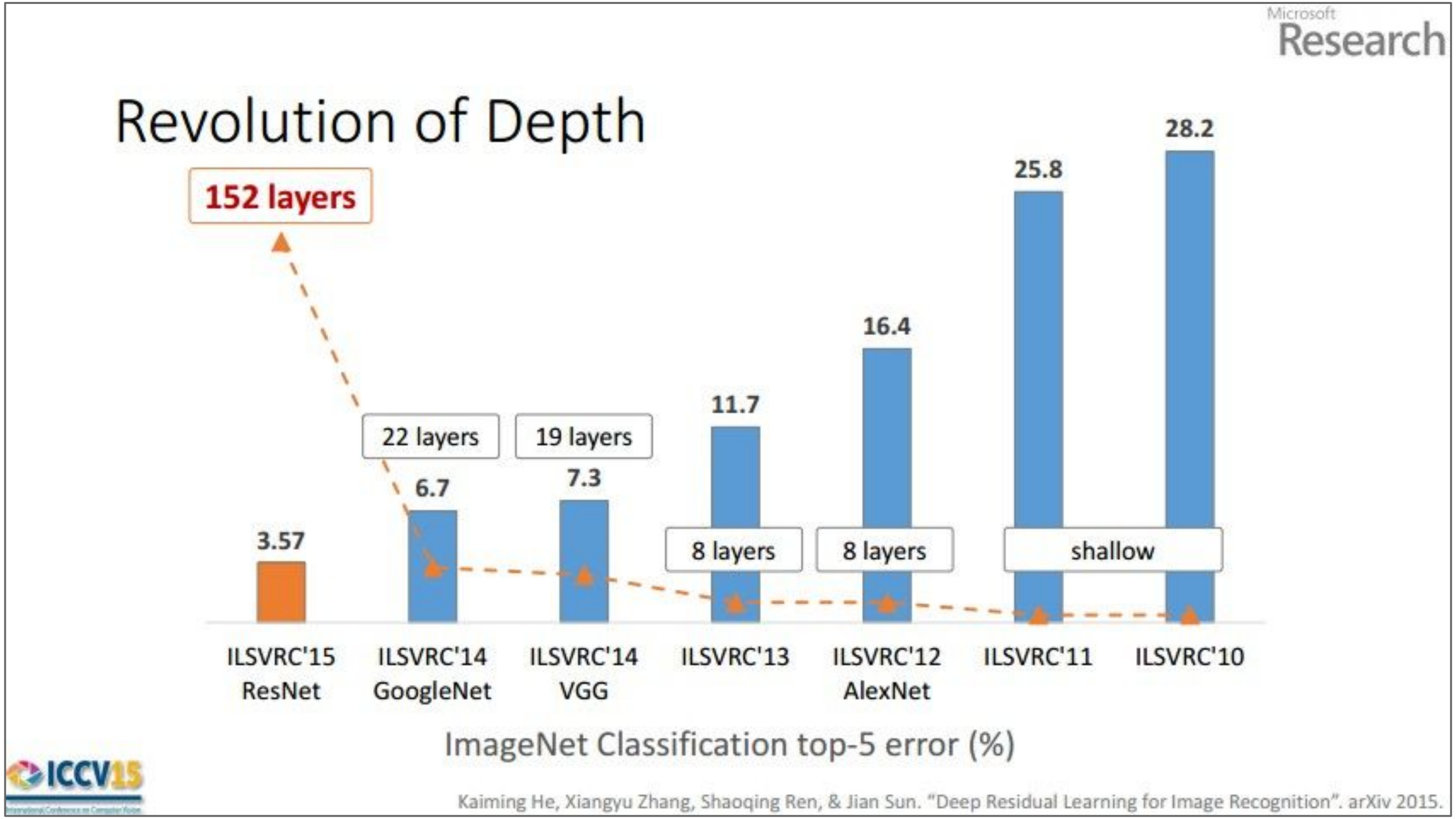
- **1st places in all five main tracks**
  - ImageNet Classification: *"Ultra-deep"* (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

*improvements are relative numbers

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Slide from Kaiming He's recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w

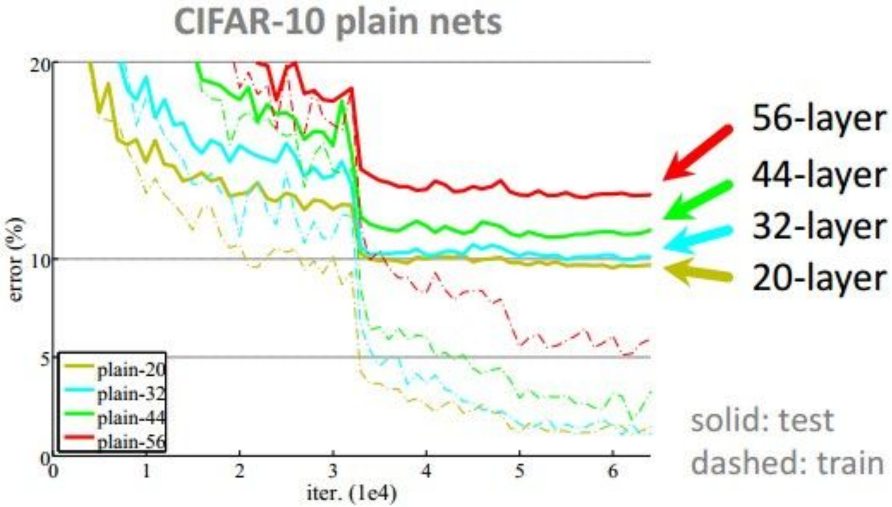
Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

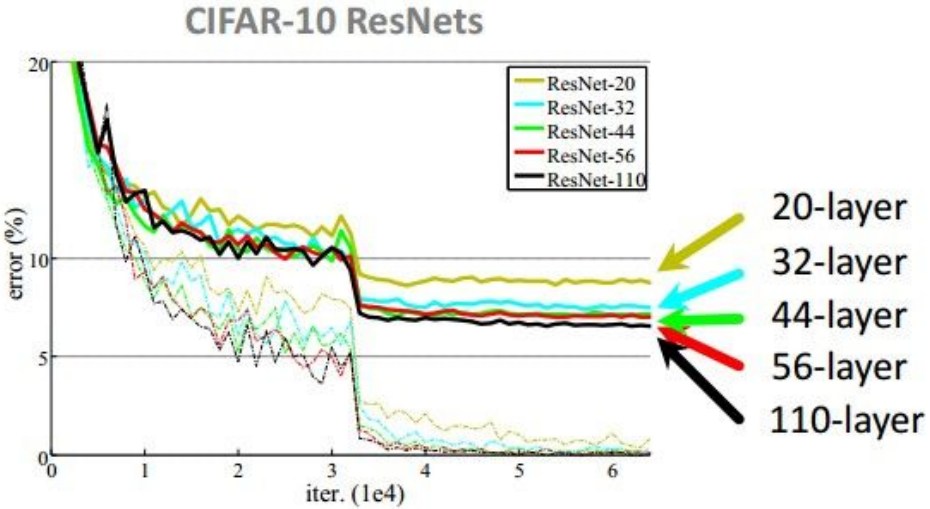


(slide from Kaiming He's recent presentation)

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

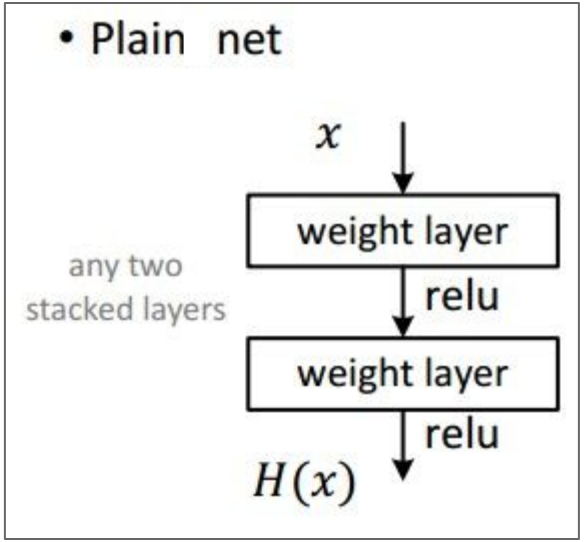
CIFAR-10 experiments

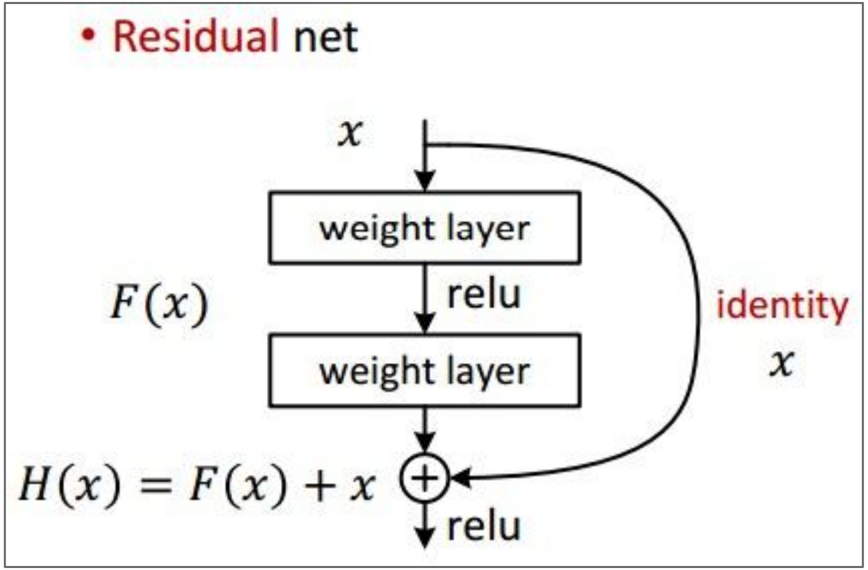




Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Case Study: ResNet *[He et al., 2015]*

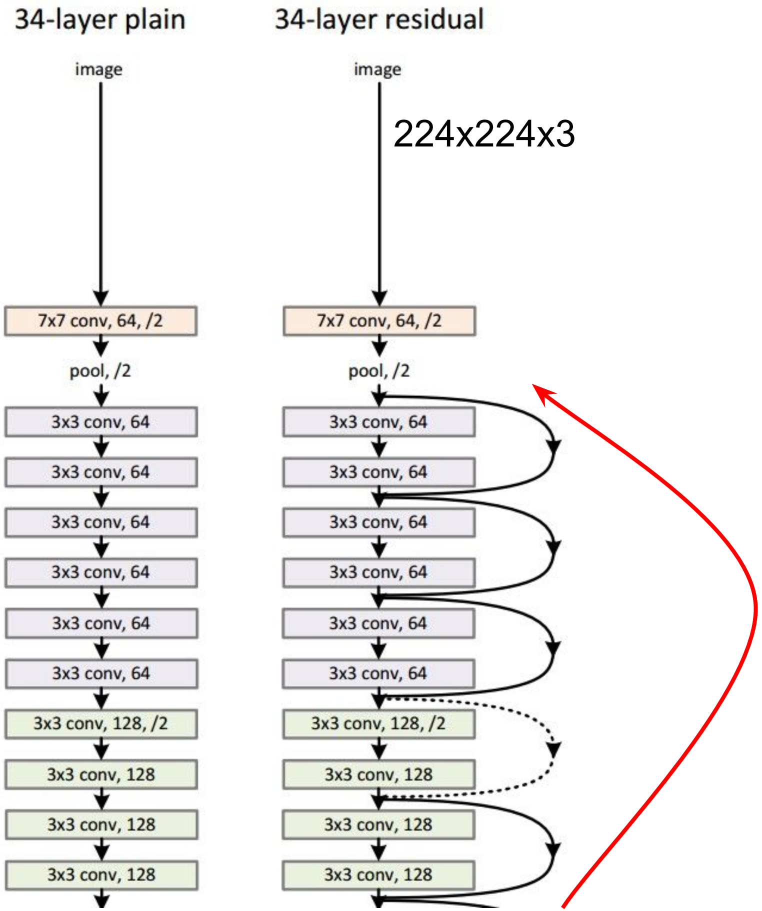




Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Case Study: ResNet

[He et al., 2015]

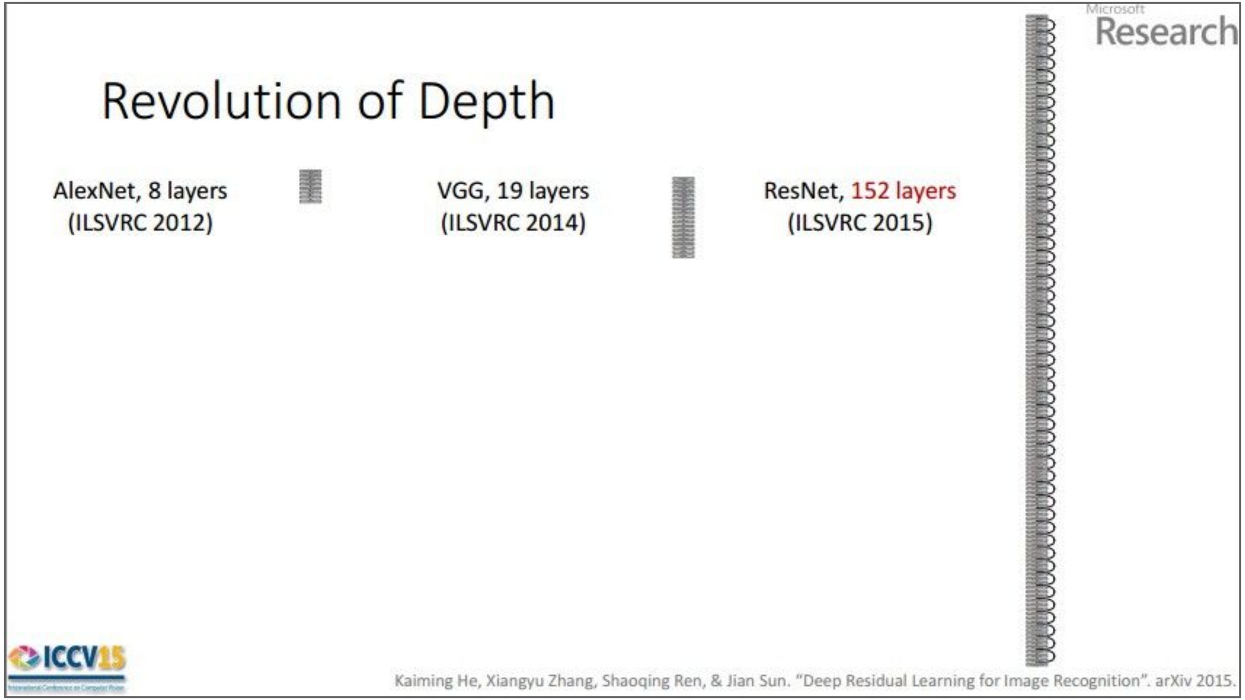


During back-prop, gradient flows through layers without vanishing

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Case Study: ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)



2-3 weeks of training on 8 GPU machine

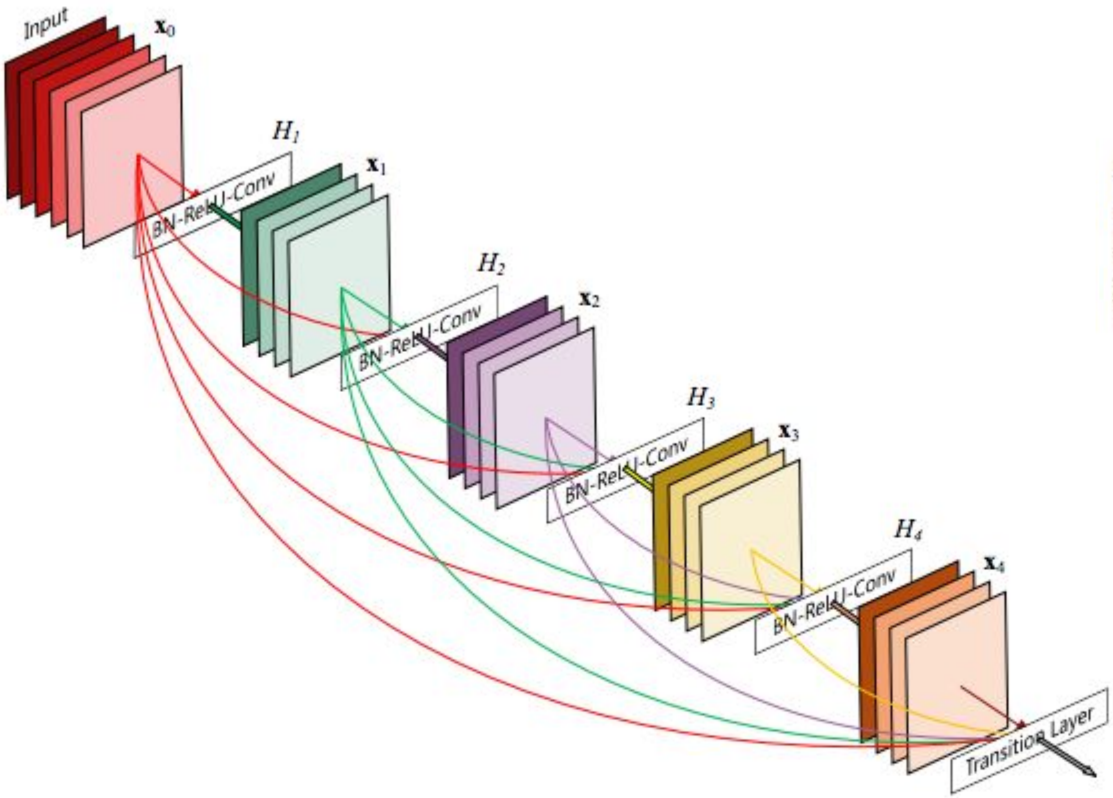
at runtime: faster than a VGGNet! (even though it has 8x more layers)

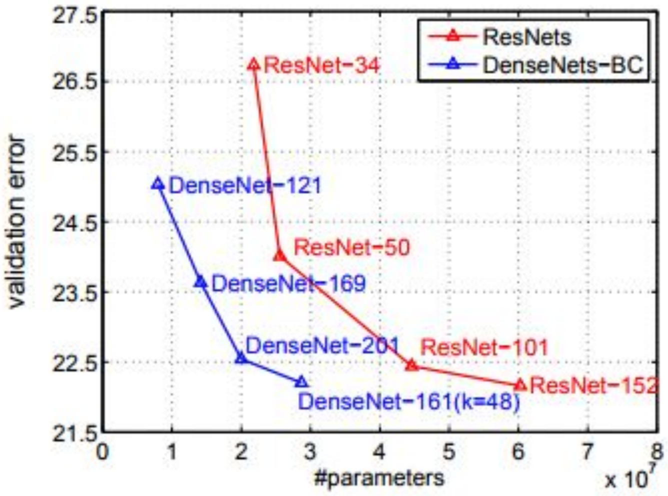
(slide from Kaiming He's recent presentation)

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Case Study: DenseNet

[Huang et al., 2017]





41

Summary

- ConvNets stack CONV,POOL,FC layers

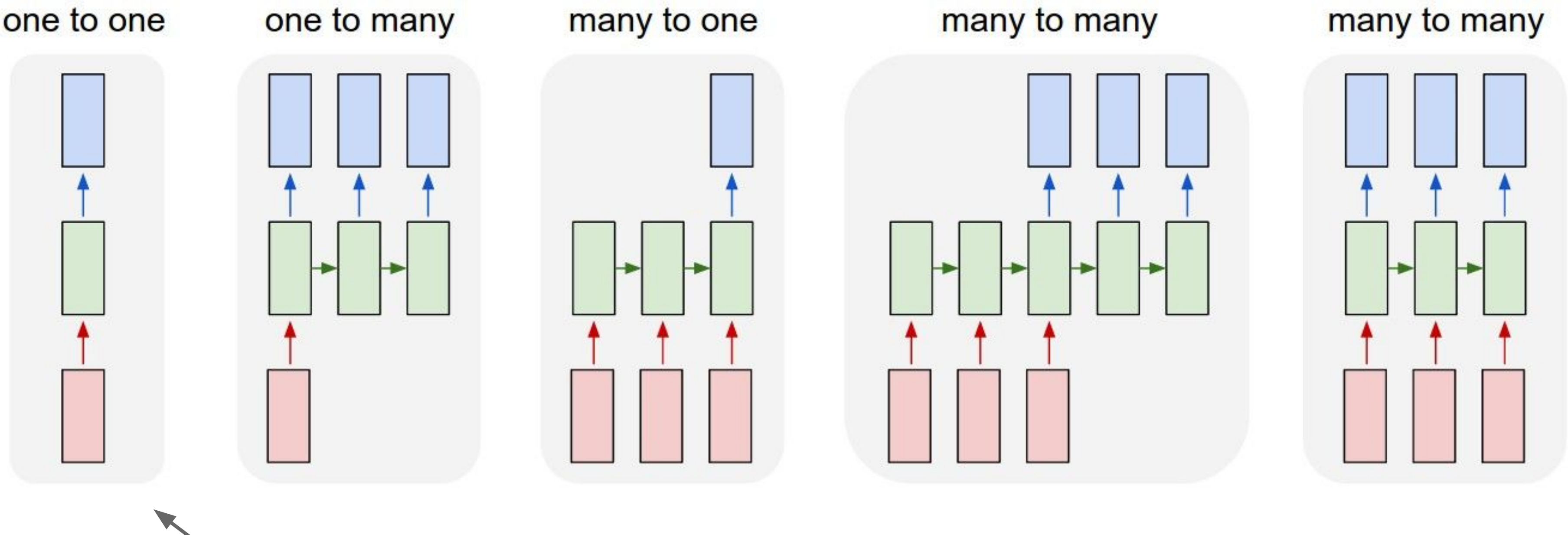
- Trend towards smaller filters and deeper architectures, with better connectivity

- Trend towards getting rid of POOL/FC layers (just CONV)

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Yinelemeli Sinir Ağları

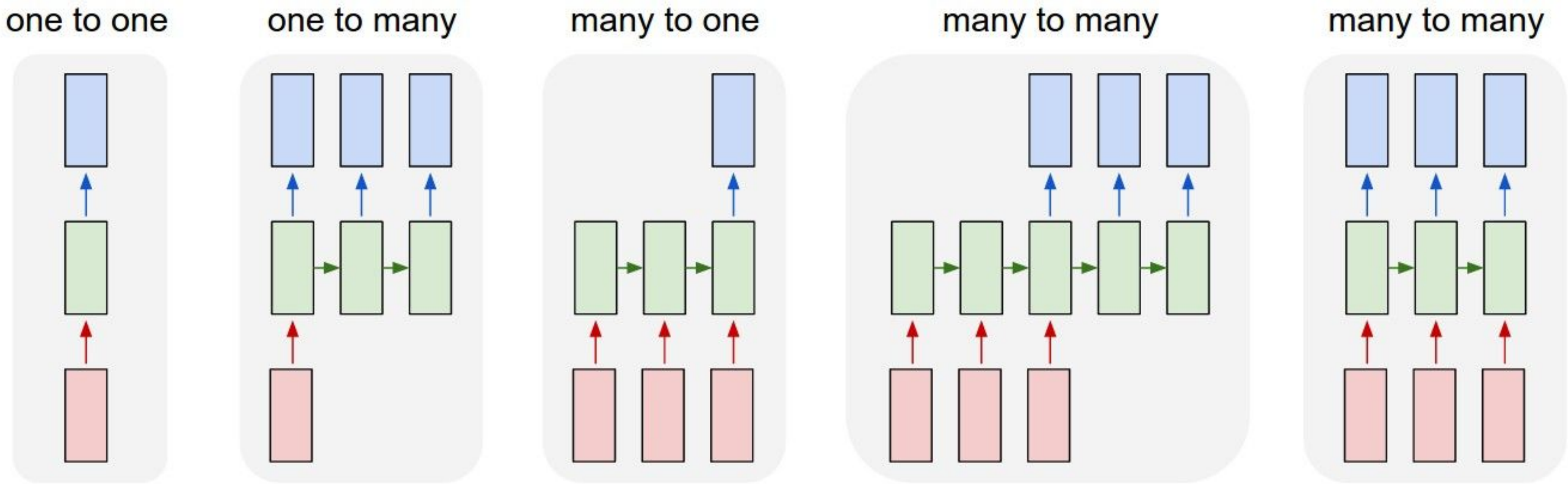
Recurrent Networks offer a lot of flexibility:



Vanilla Neural Networks

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

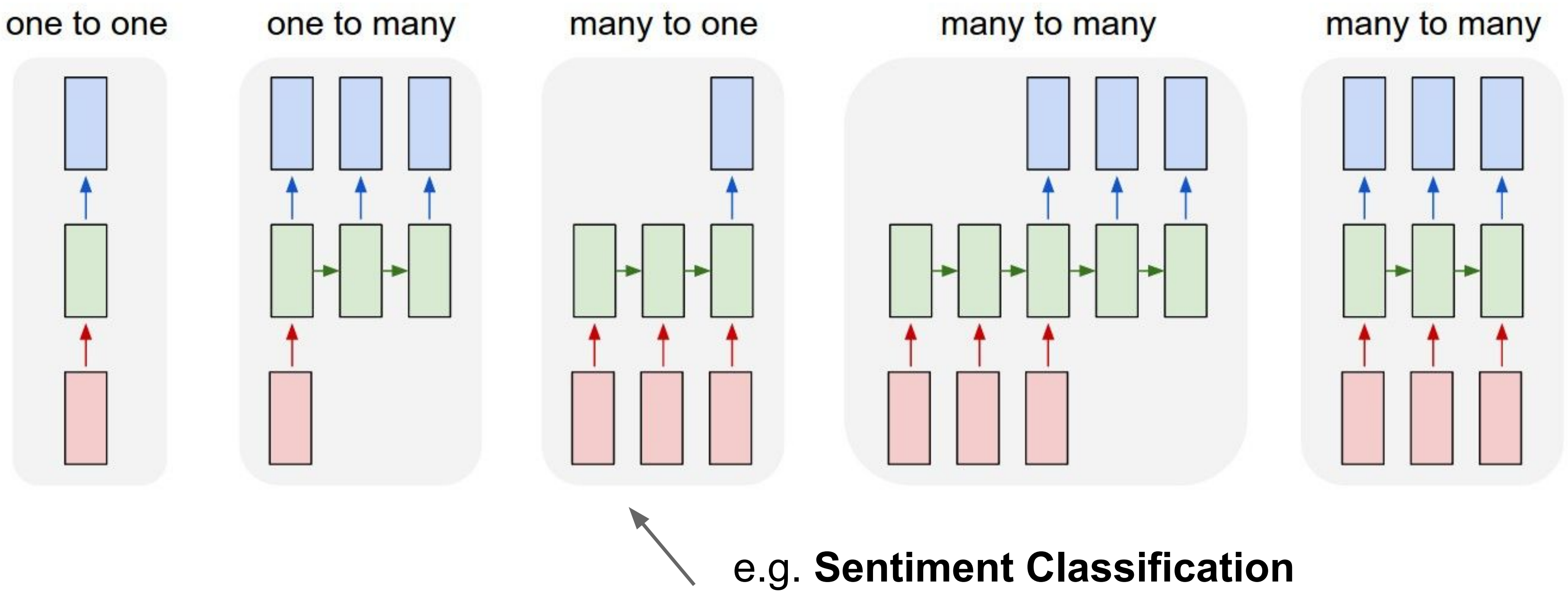
Recurrent Networks offer a lot of flexibility:



e.g. **Image Captioning**
image -> sequence of words

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Recurrent Networks offer a lot of flexibility:

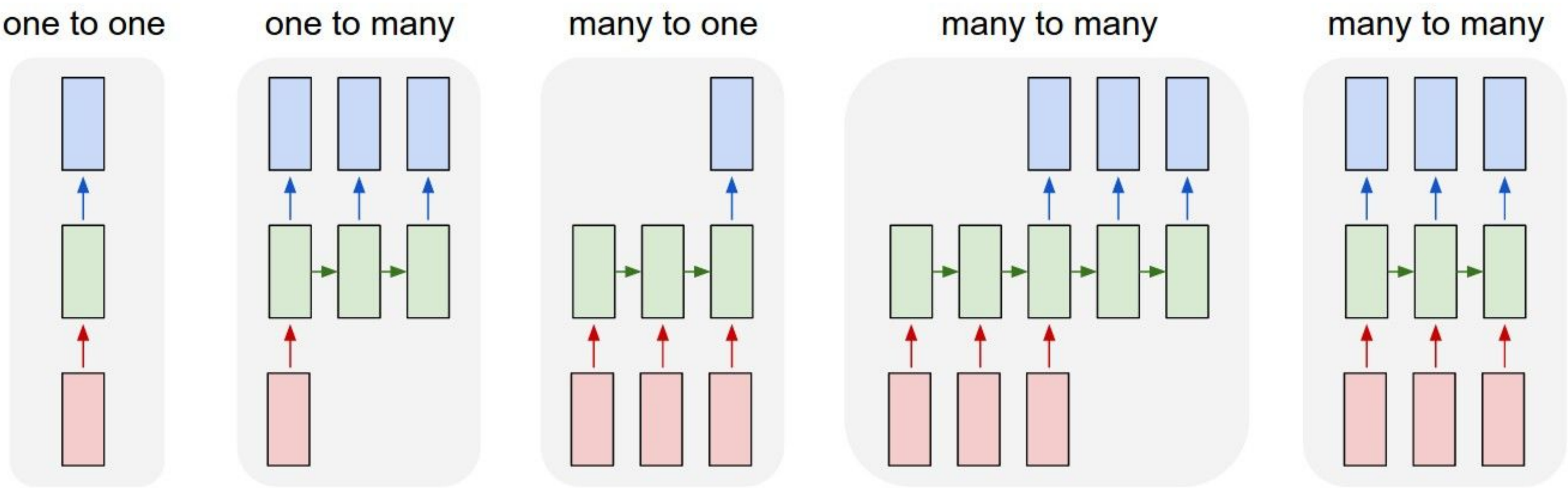


e.g. **Sentiment Classification**
sequence of words -> sentiment

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

46

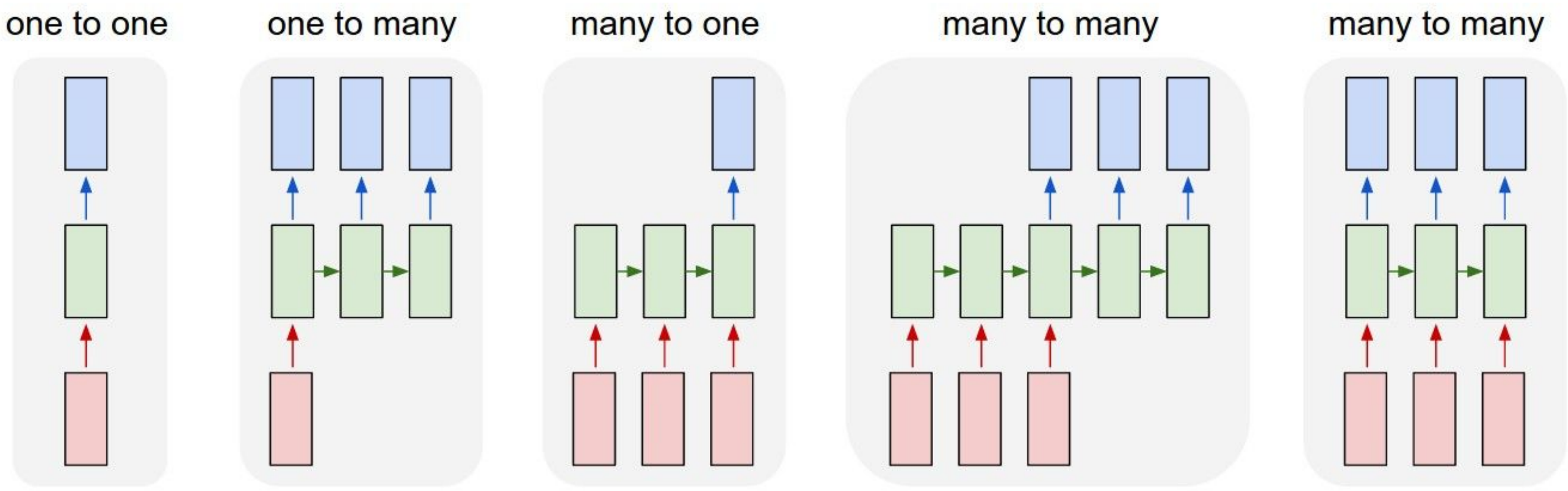
Recurrent Networks offer a lot of flexibility:



e.g. **Machine Translation**
seq of words -> seq of words

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Recurrent Networks offer a lot of flexibility:

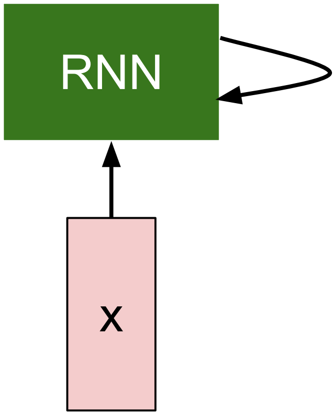


e.g. **Video classification on frame level**

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

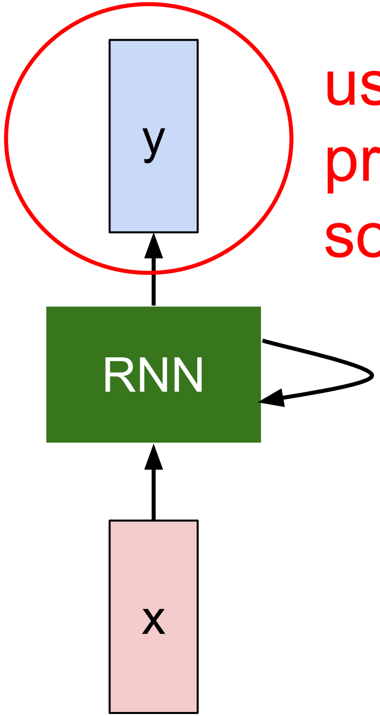
48

Recurrent Neural Network



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Recurrent Neural Network



usually want to predict a vector at some time steps

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Recurrent Neural Network

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

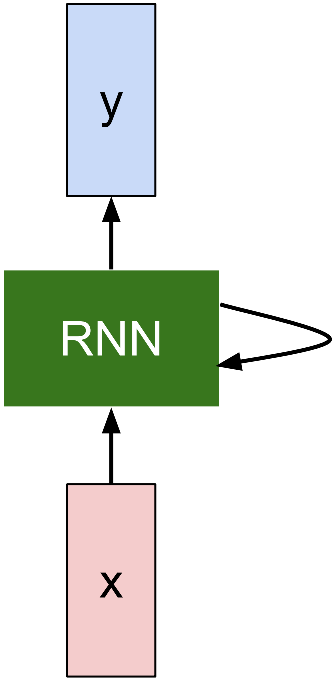
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W

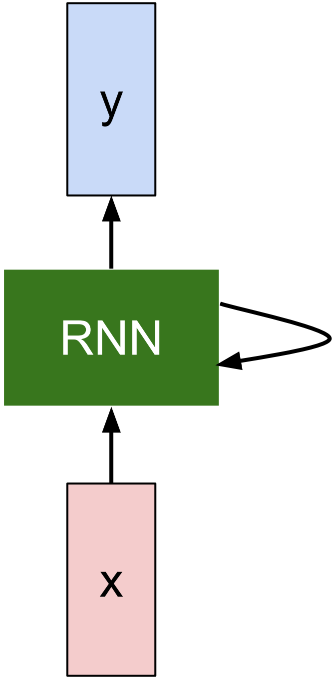


Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Recurrent Neural Network

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

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

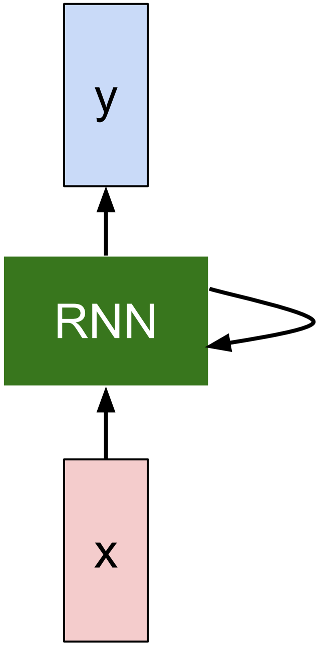


Notice: the same function and the same set of parameters are used at every time step.

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

(Vanilla) Recurrent Neural Network

The state consists of a single *“hidden”* vector **h**:



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

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

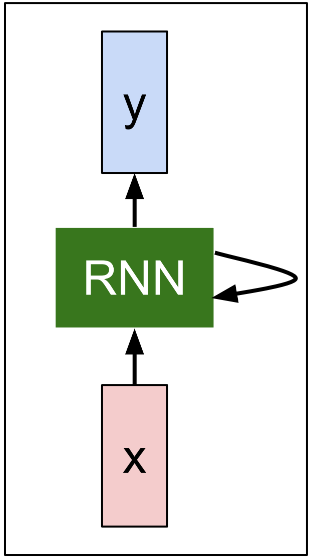
$$y_t = W_{hy}h_t$$

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

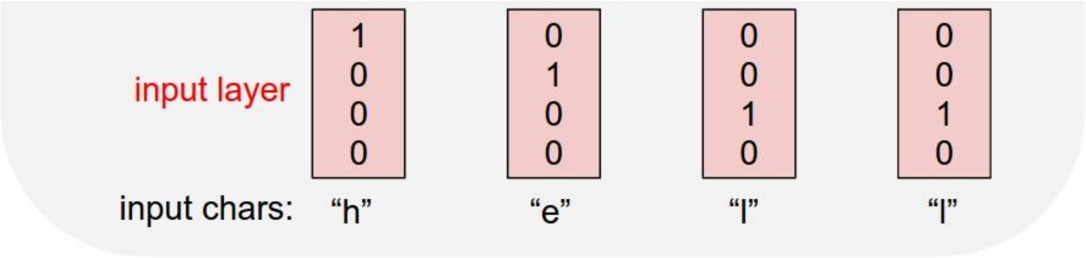


Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



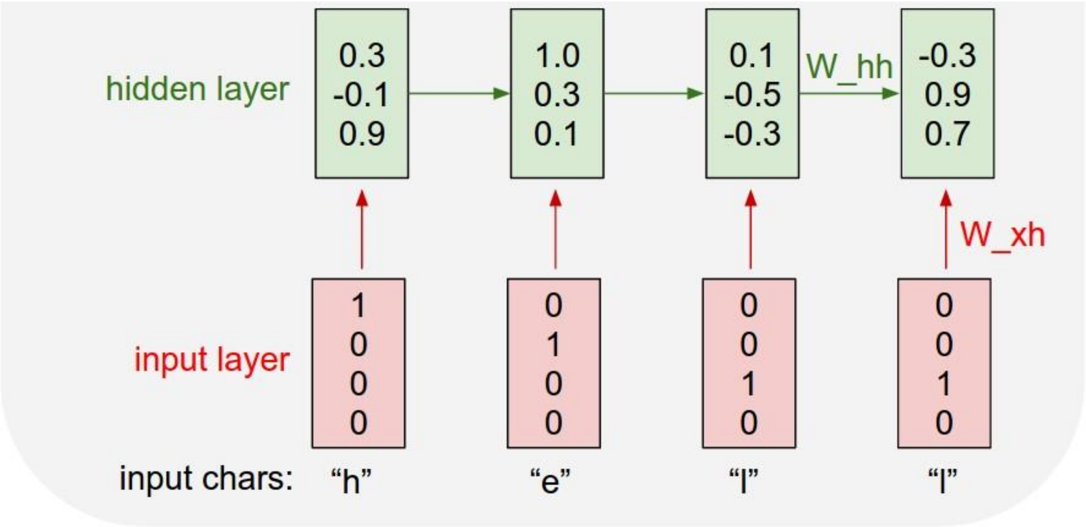
Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

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



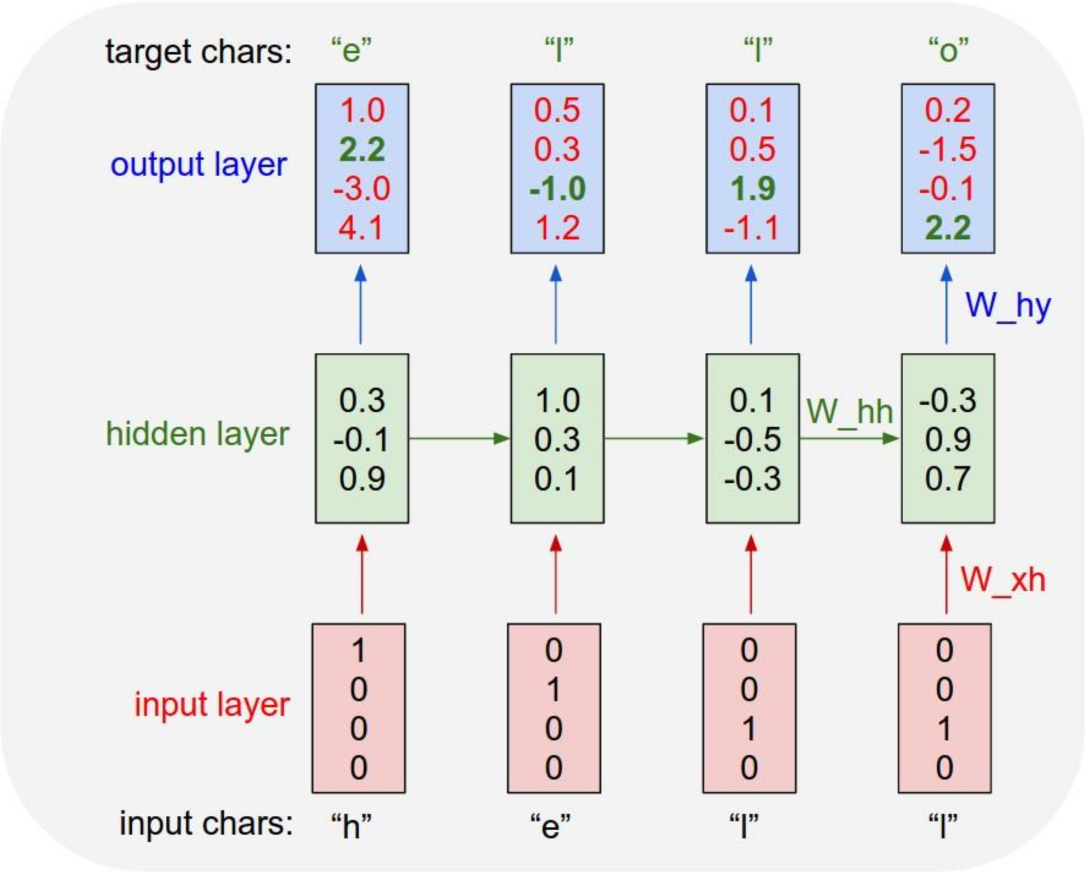
Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

56

Character-level language model example

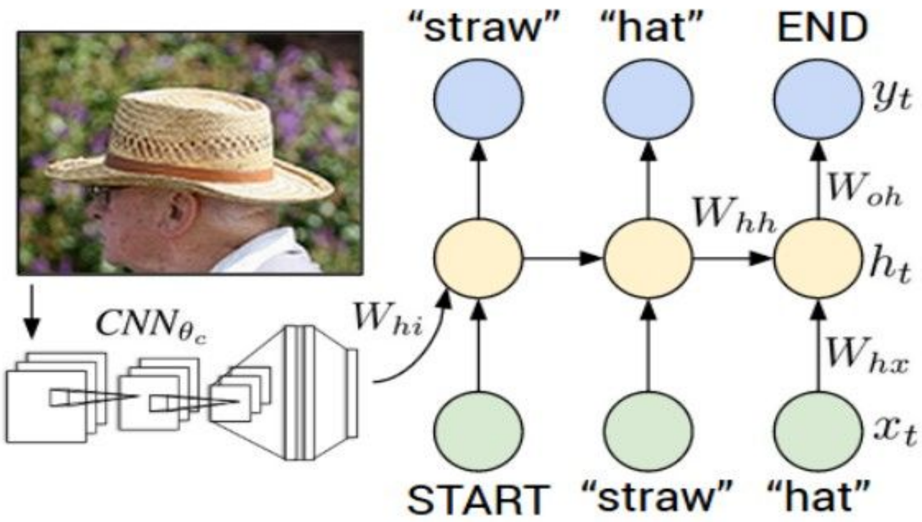
Vocabulary:
[h,e,l,o]

Example training sequence:
"hello"



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Image Captioning



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

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

test image

image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096
FC-1000
softmax

test image

image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax

test image

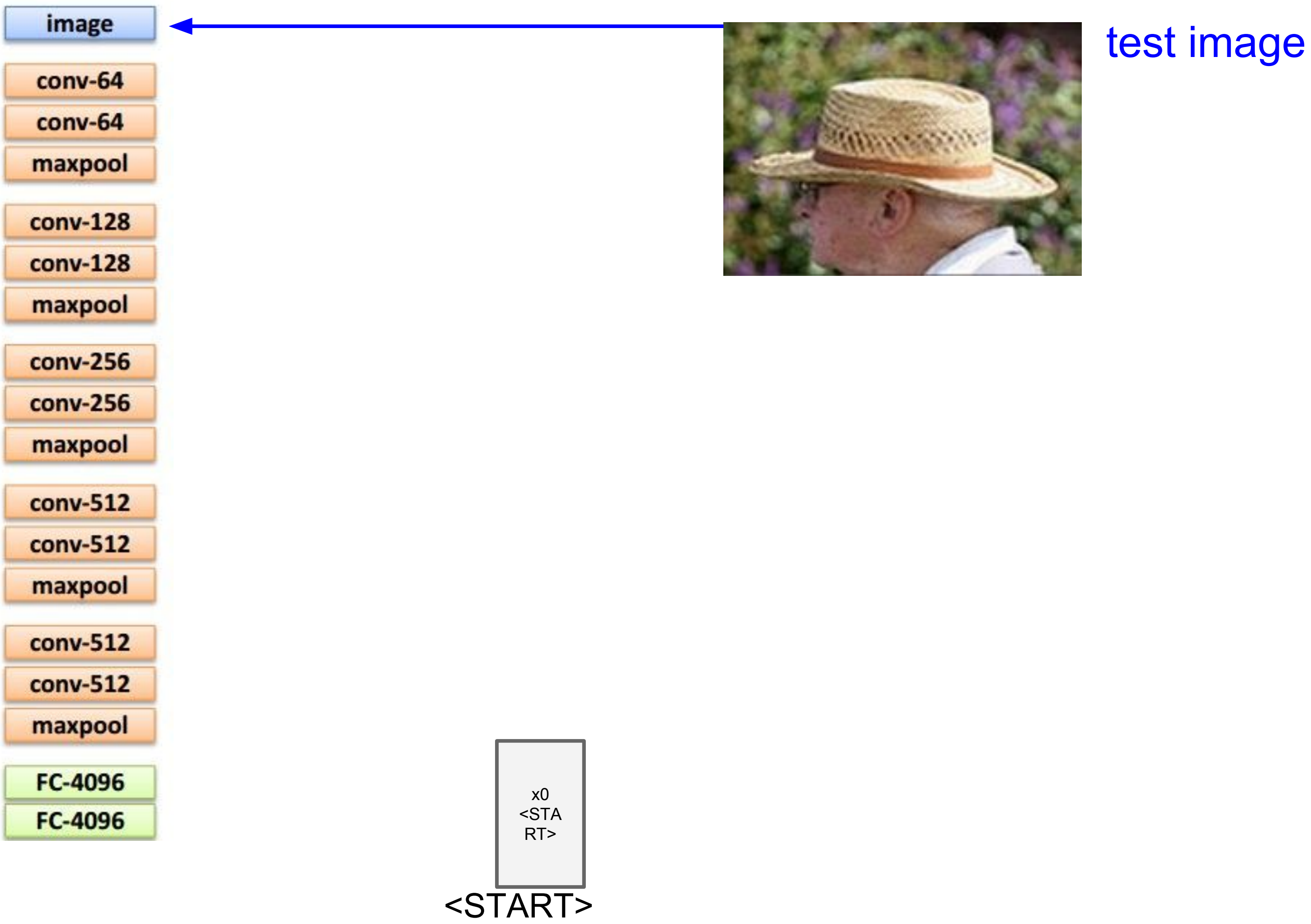
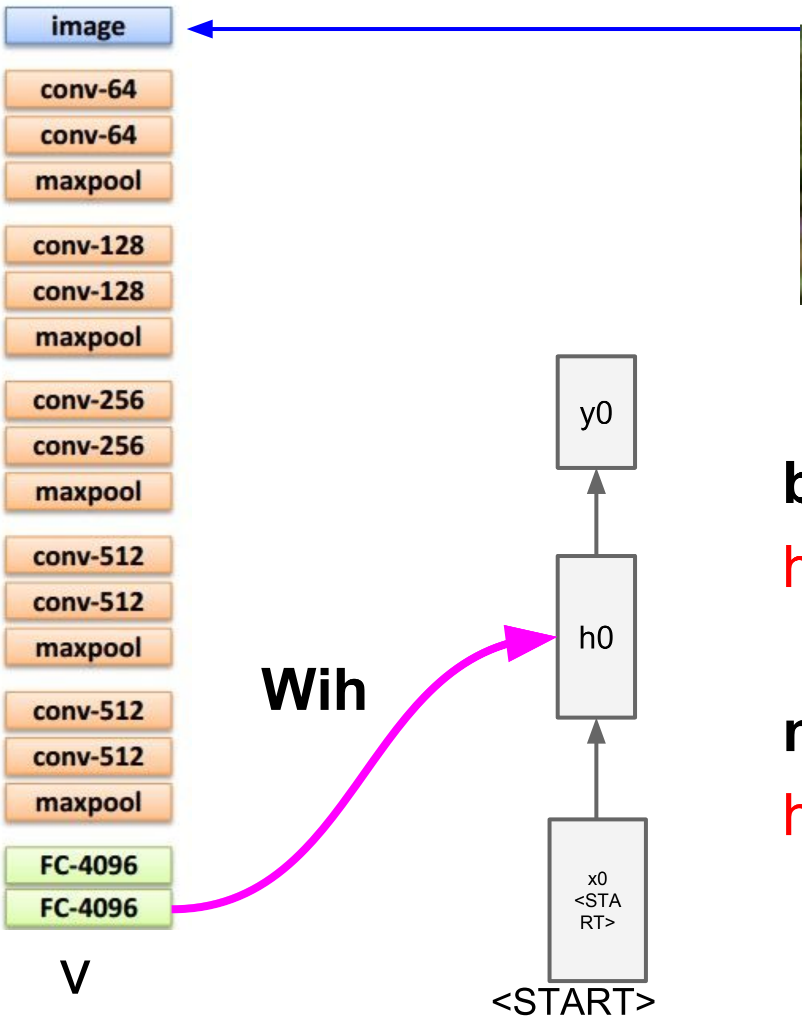


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
x0
<STA
RT>
<START>



test image

before:

h = tanh(Wxh * x + Whh * h)

now:

h = tanh(Wxh * x + Whh * h + **Wih * v**)

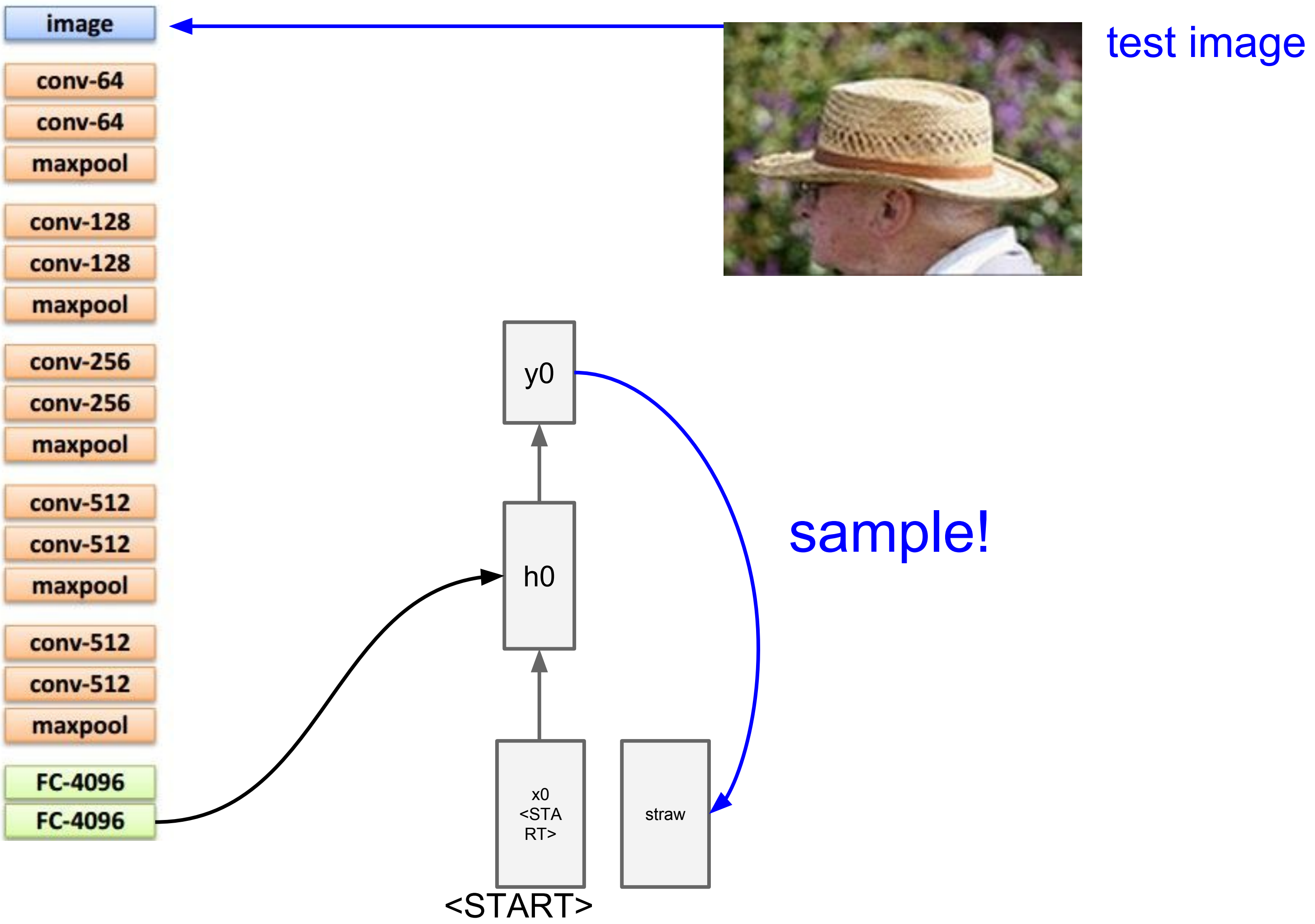


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
h0
sample!
x0
<STA
RT>
straw
<START>

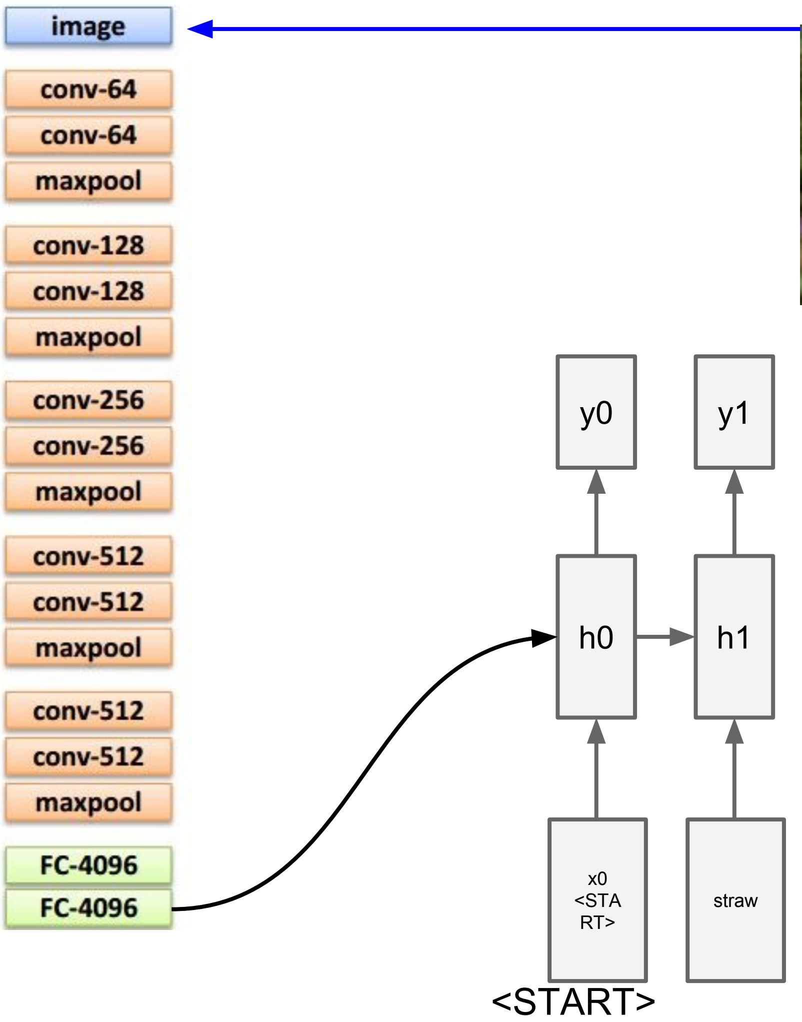


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
y0
y1
h0
h1
x0
<START>
straw
<START>

test image

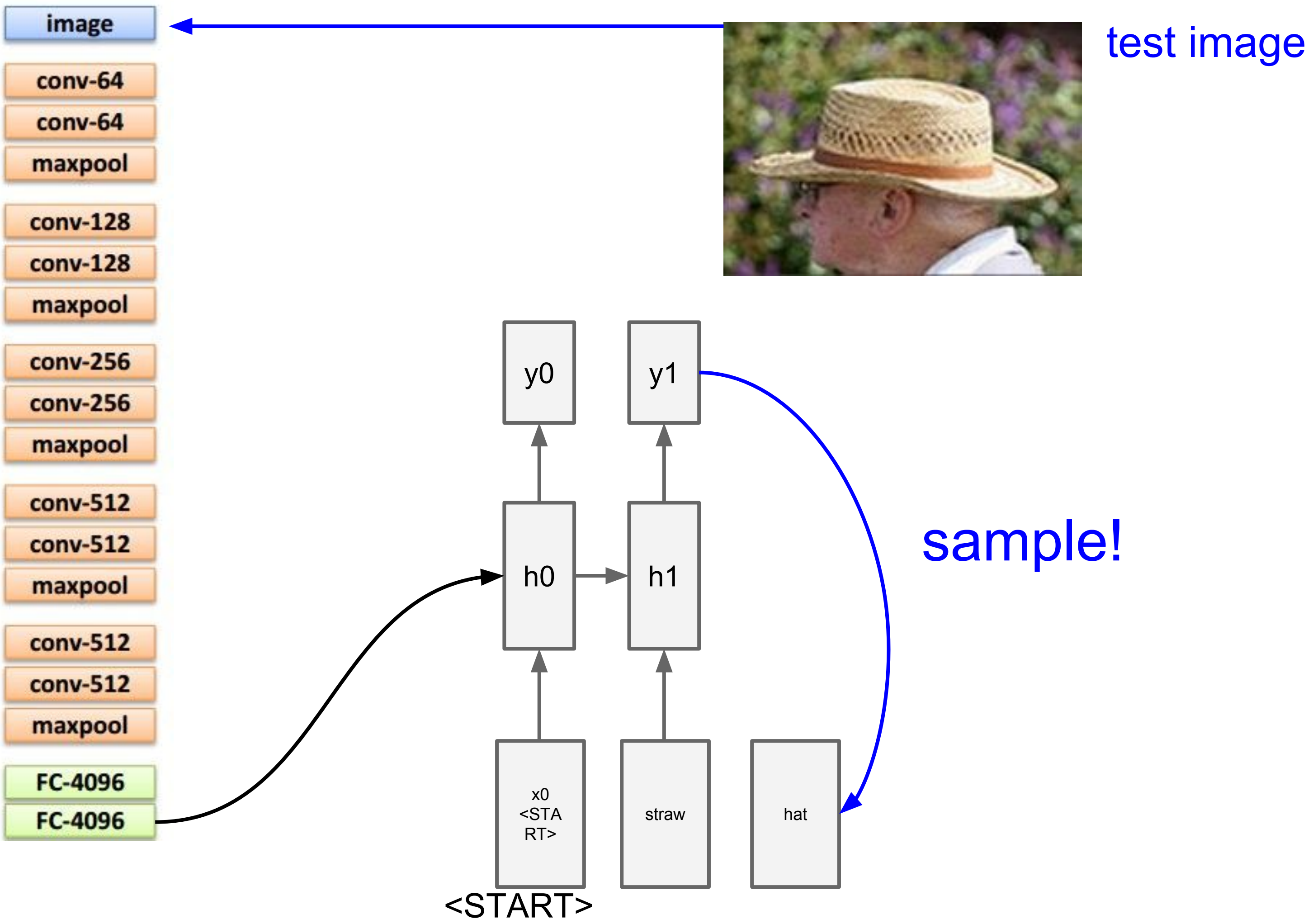


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
sample!
x0
<STA
RT>
straw
hat
<START>

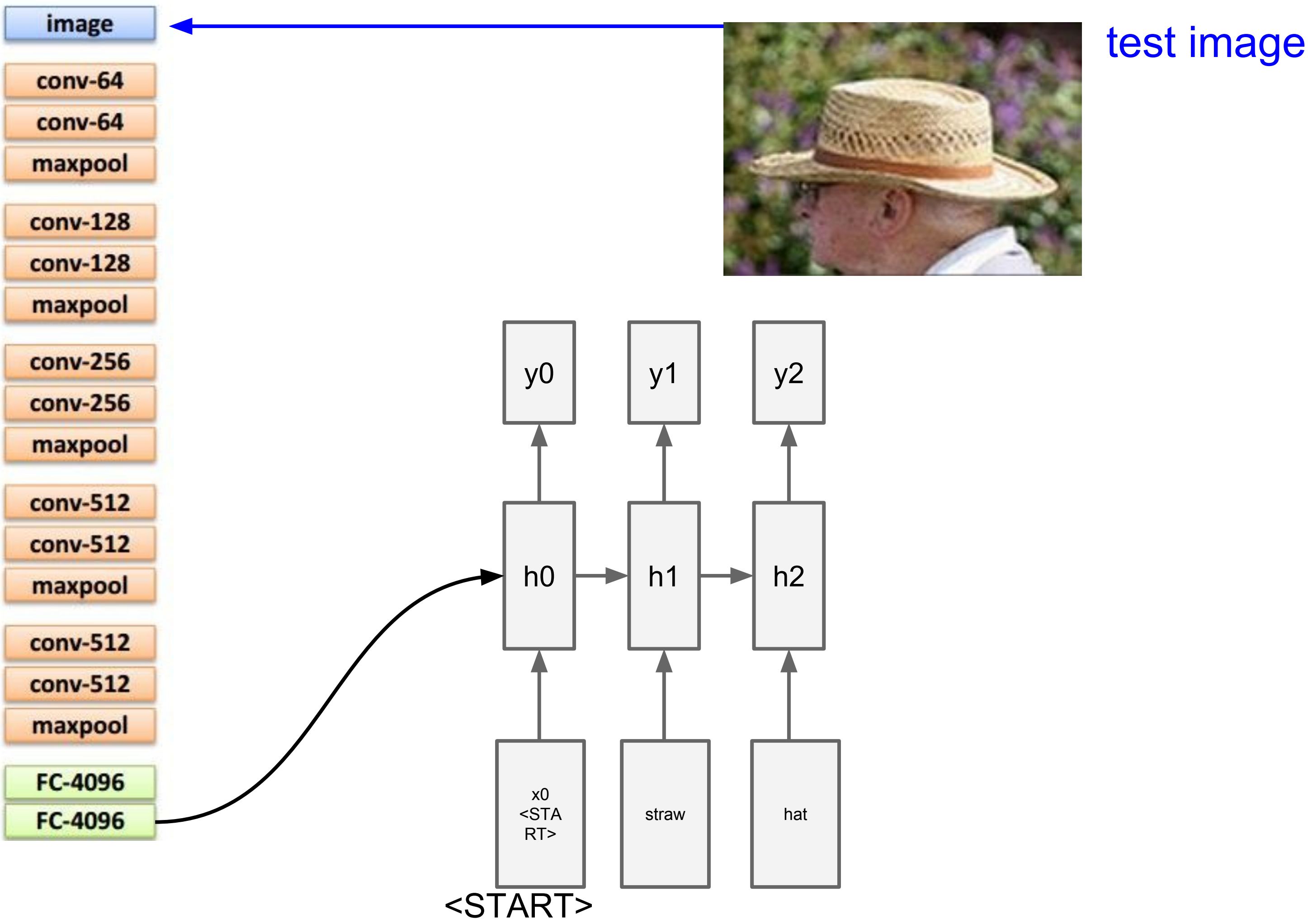


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0 <STA RT>
straw
hat
<START>

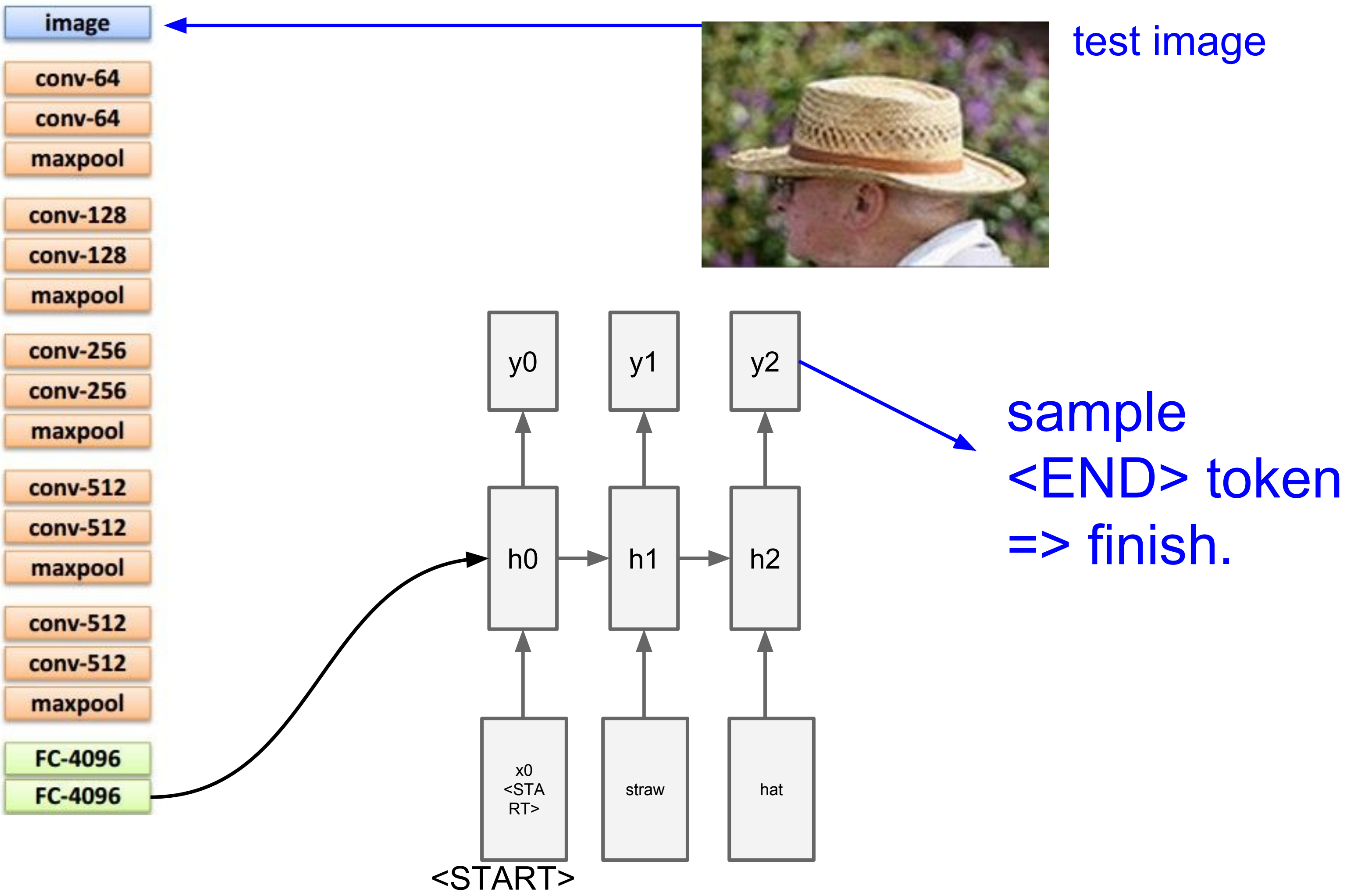


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0 <START>
straw
hat
<START>
sample <END> token => finish.

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

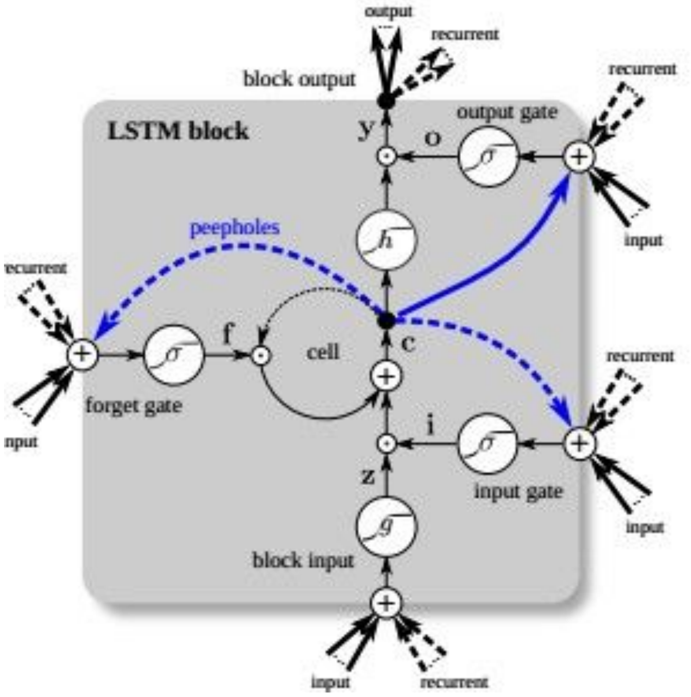
"a young boy is holding a baseball bat."

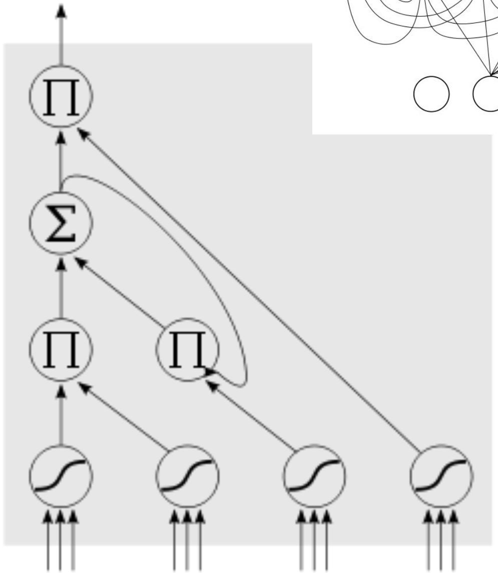
"a cat is sitting on a couch with a remote control."

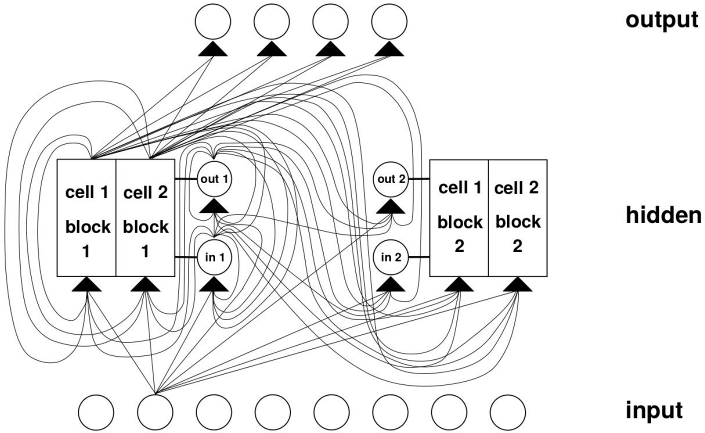
"a woman holding a teddy bear in front of a mirror."

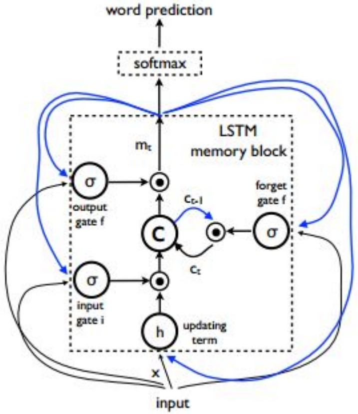
"a horse is standing in the middle of a road."

LSTM (Long-short term memory)



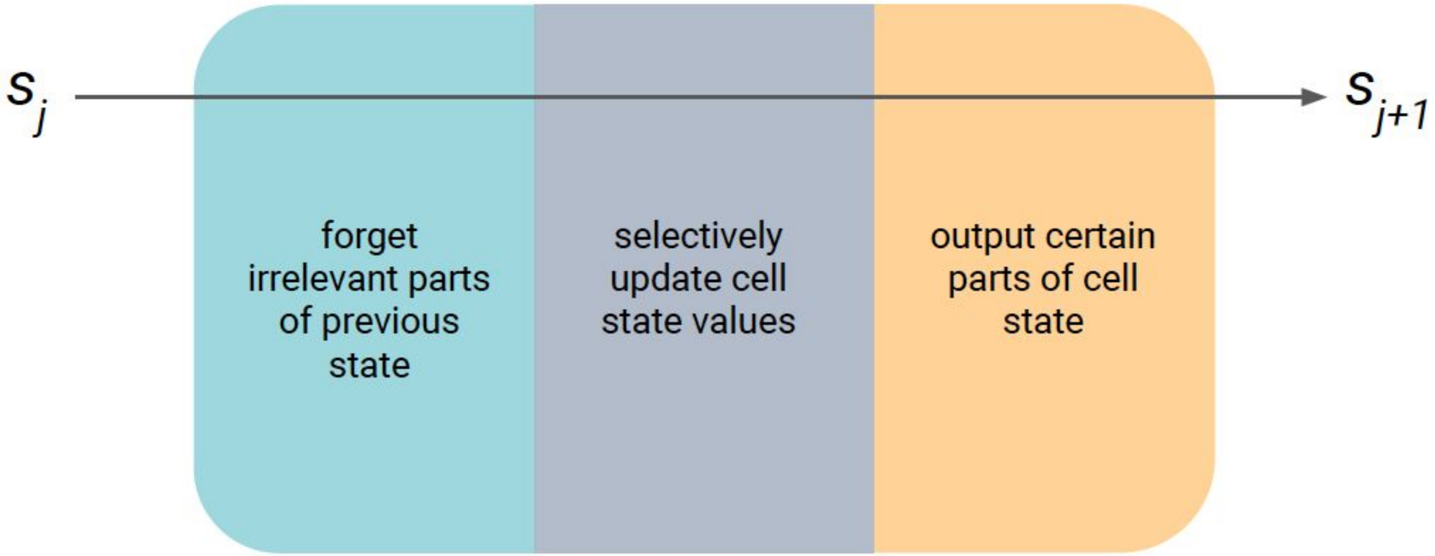






Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

LSTM - main idea

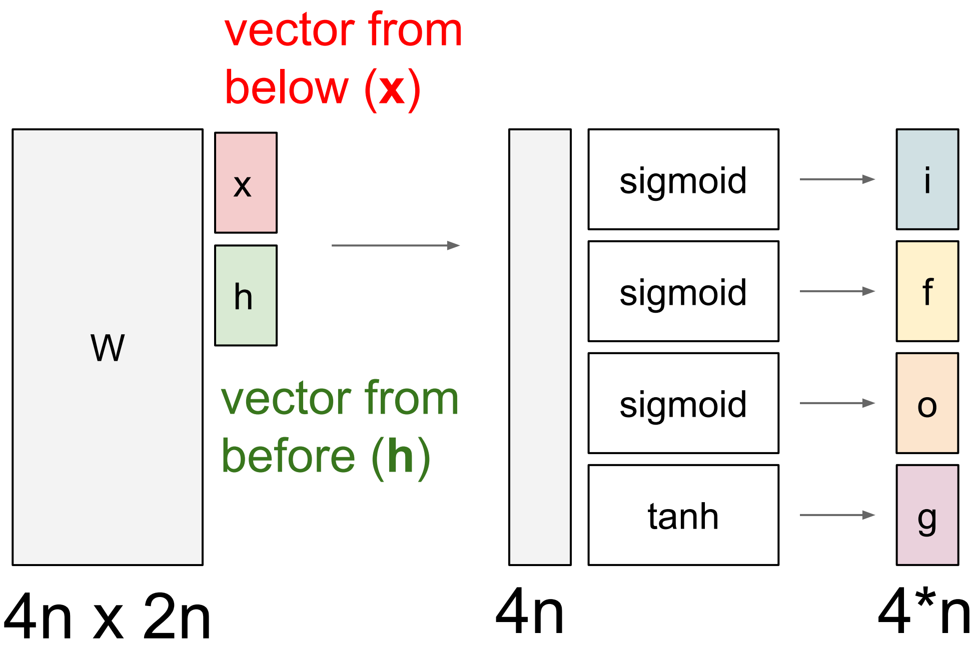


Slide adapted from MIT 6.S191 (IAP 2017), by Harini Suresh

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

- *c: cell state*
- *h: hidden state (cell output)*
- *i: input gate, weight of acquiring new information*
- *f*: forget gate, weight of remembering old information
- *g: transformed input ([-1,+1])*
- *o: output gate, decides values to be activated based on current memory*



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

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

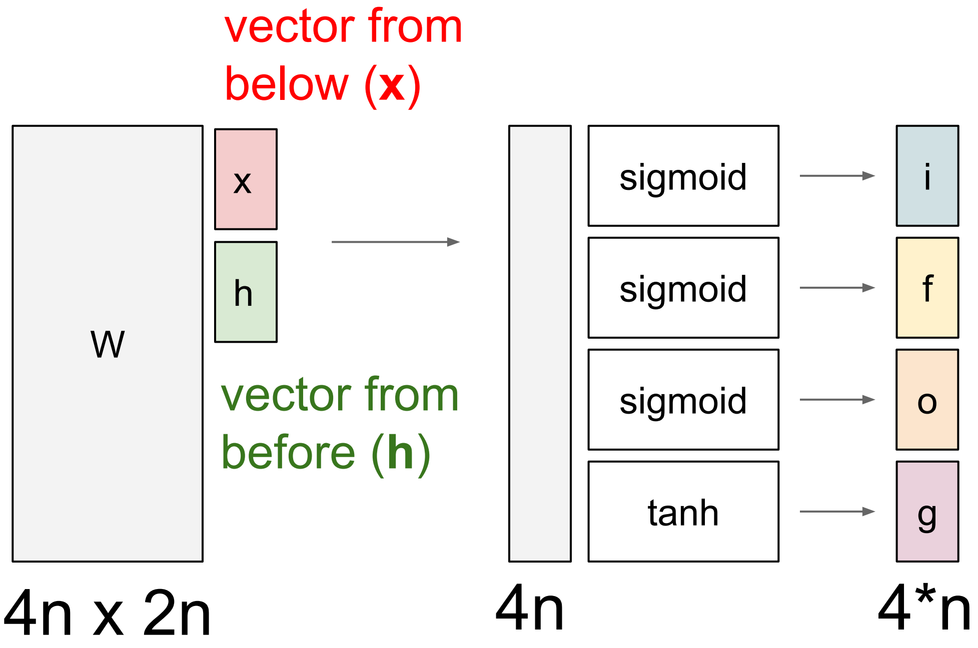
$$h_t^l = o \odot \tanh(c_t^l)$$

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

f decides *the degree of preservation for cell state*, by scaling it with a number in [0,1]



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

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

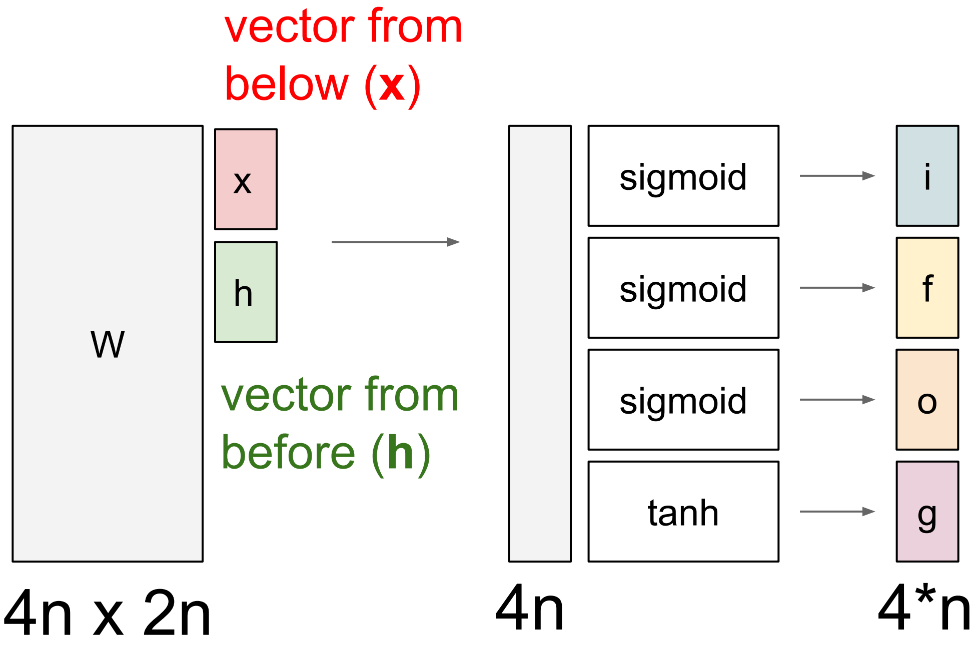
$$h_t^l = o \odot \tanh(c_t^l)$$

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

g is a transformation of input / hidden state



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

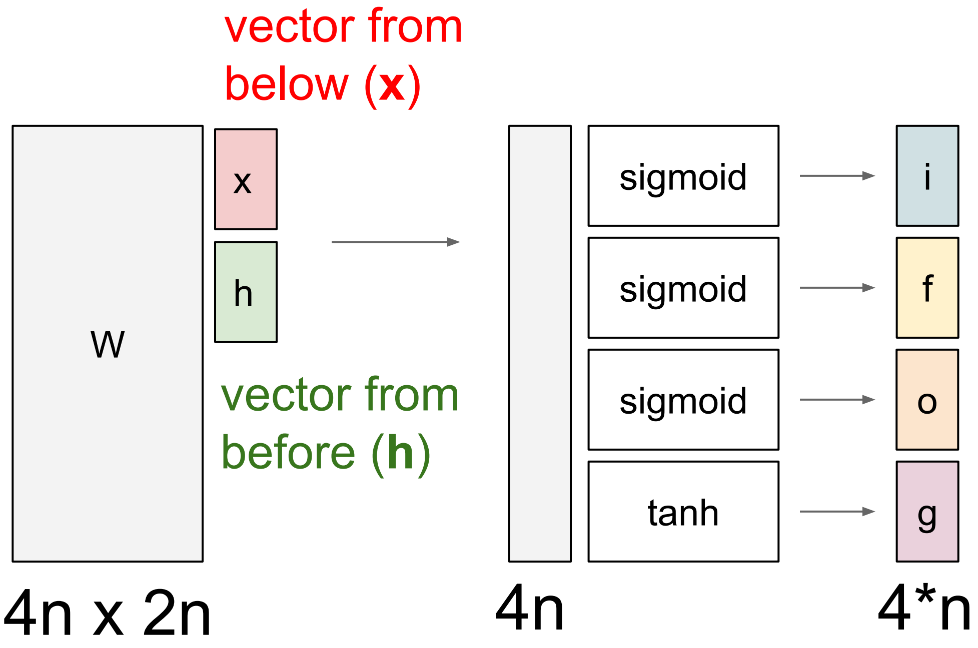
$$c_t^l = f \odot c_{t-1}^l + i \odot \boxed{g}$$

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

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



Add *g* into the *cell state*,
weighted by *i*
(weight of acquiring new information)

*Alternative interpretation: i*g decouples the "influence of g" and "g itself".*

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

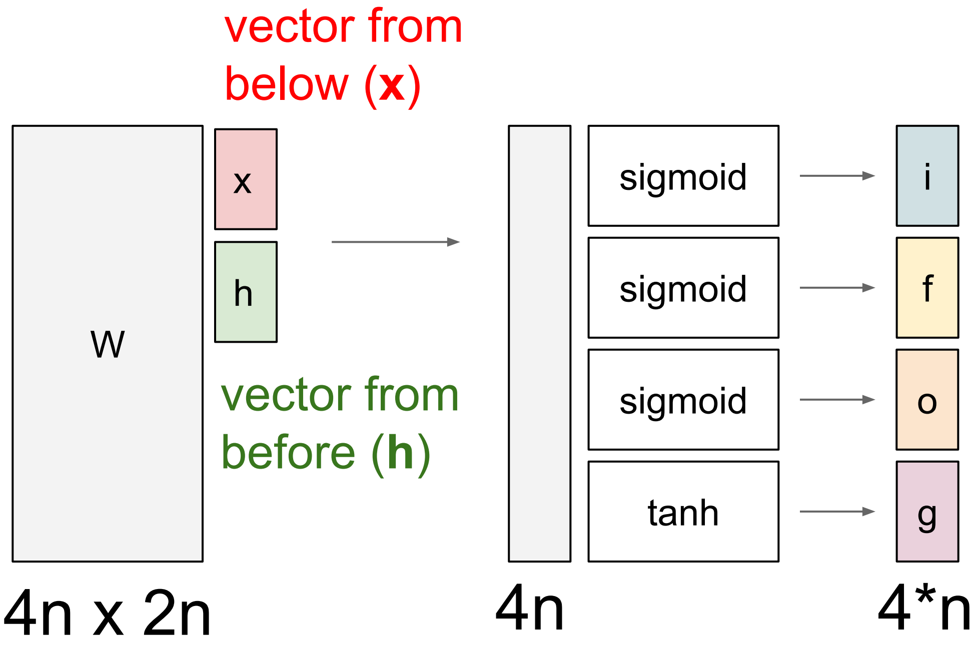
$$c_t^l = f \odot c_{t-1}^l \boxed{+ i \odot g}$$

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

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



New hidden state is a scaled version of tanh(cell state).

o: output gate, decides values to be activated based on current memory

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

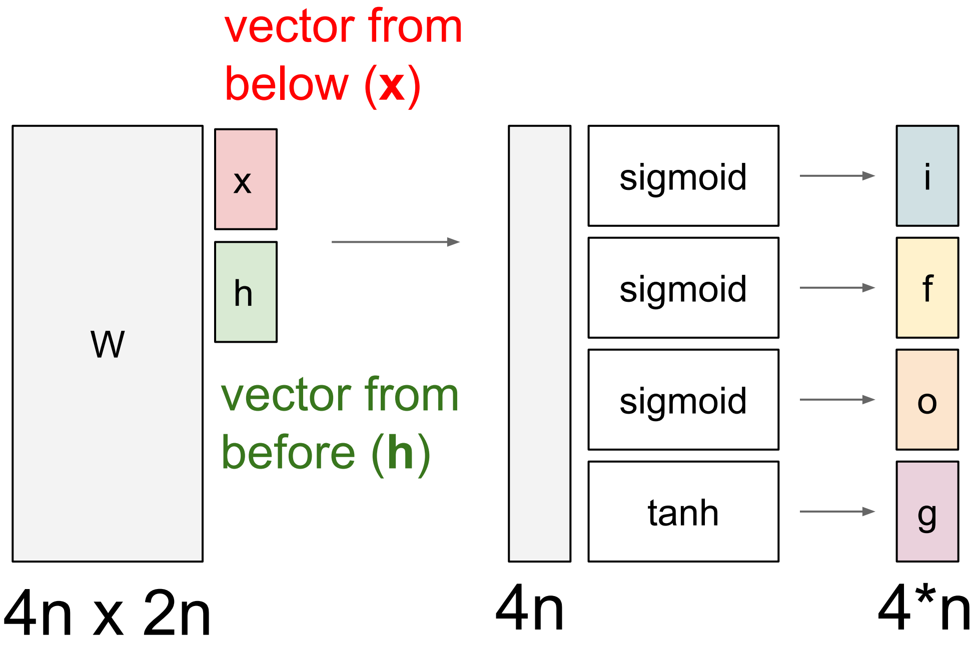
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

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

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



Q: Why tanh?

A: Not very crucial, sometimes not used

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

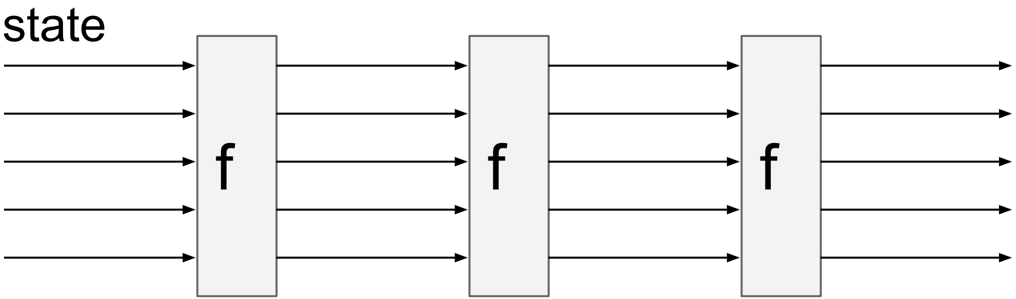
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

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

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

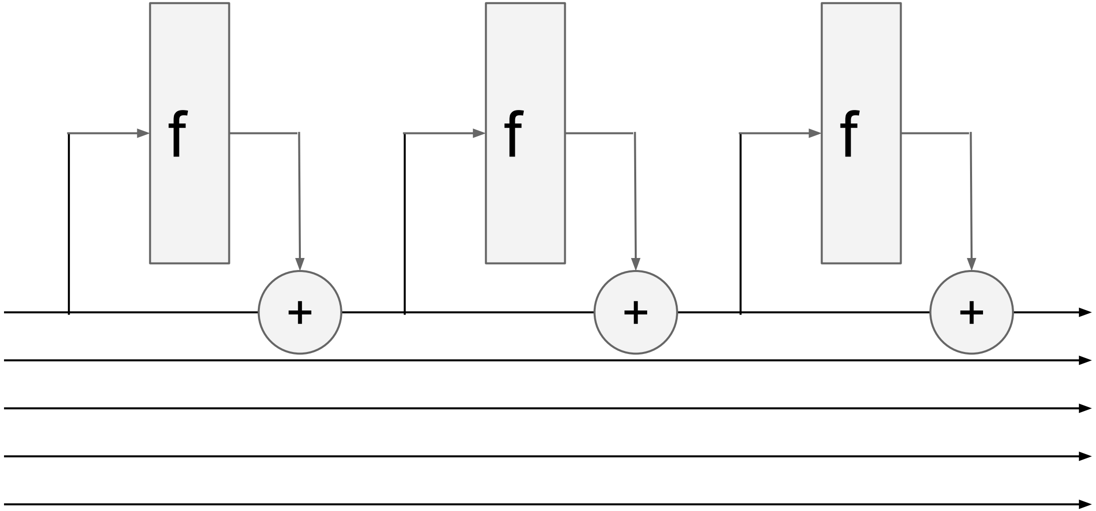
RNN

More prone to the vanishing gradient problem

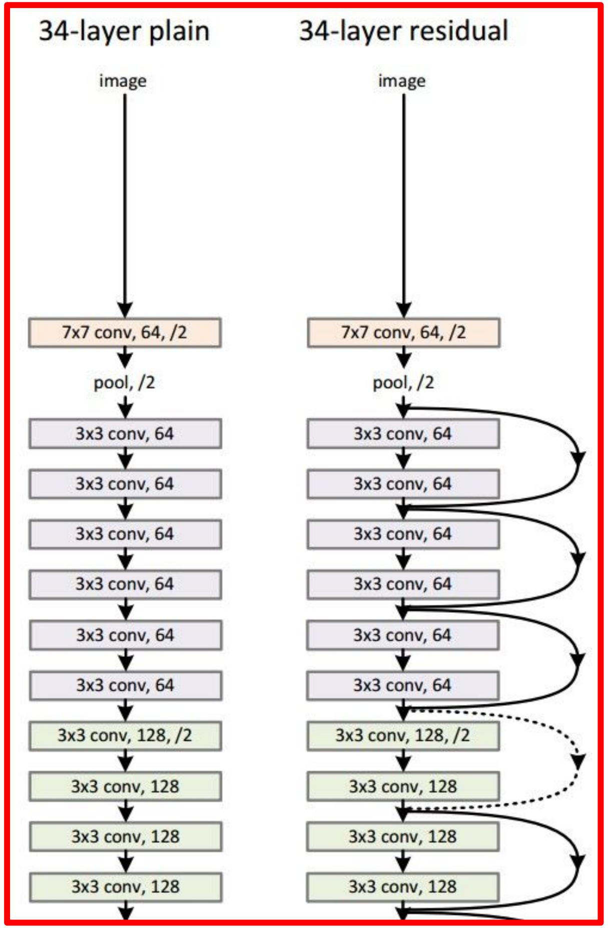


LSTM

(ignoring forget gates)

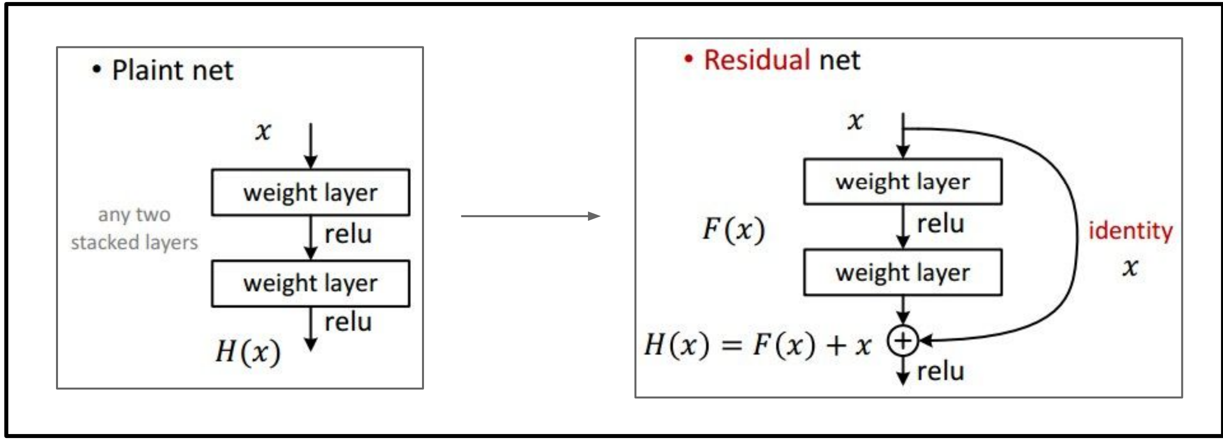


Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır



Recall:
“PlainNets” vs. ResNets

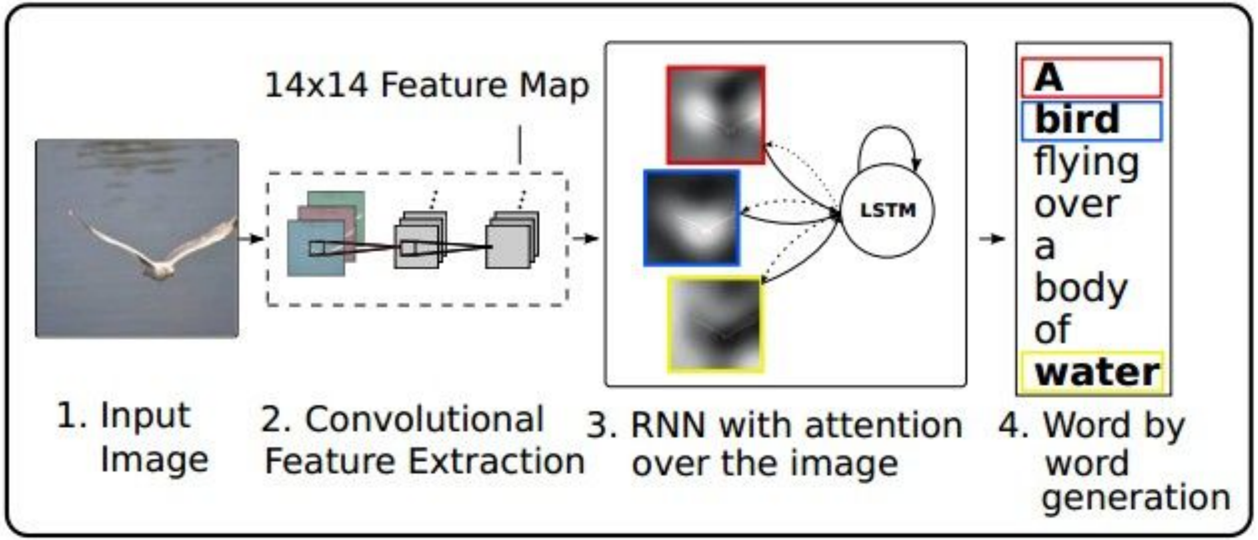
ResNet is to PlainNet what LSTM is to RNN, kind of.



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

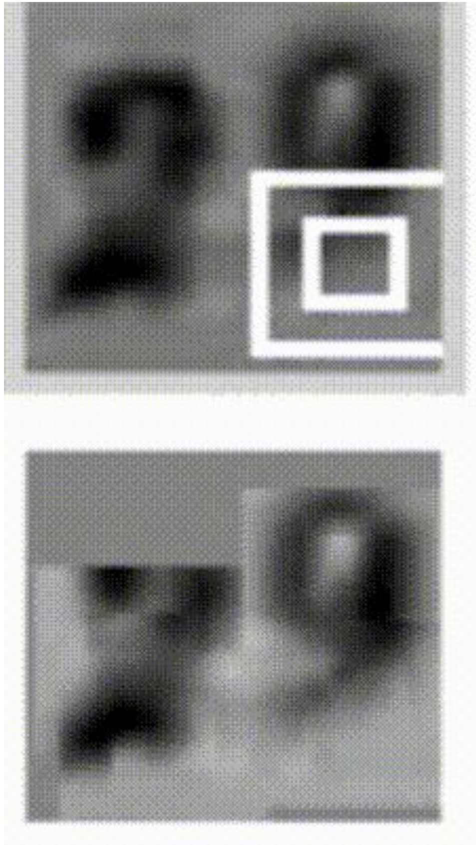
80

LSTM for spatial attention



Show Attend and Tell, Xu et al., 2015

Sequential Processing of fixed inputs

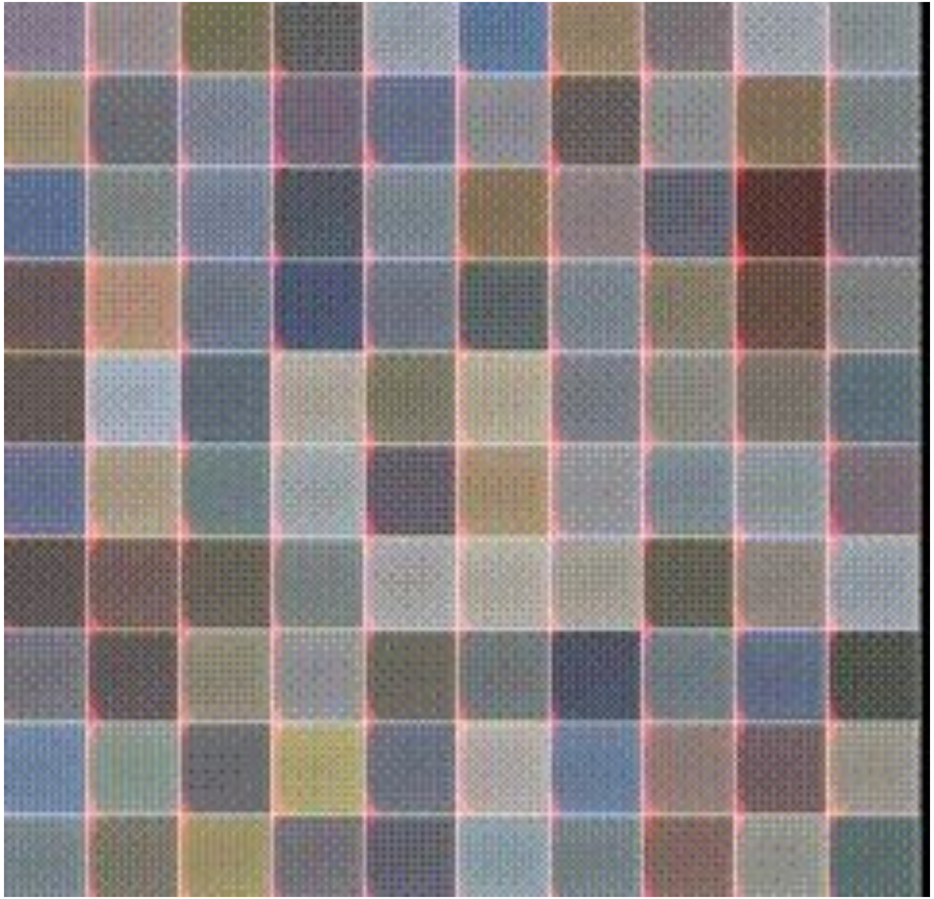


Multiple Object Recognition with Visual Attention, Ba et al.

Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

Sequential Processing of fixed outputs

DRAW: A Recurrent Neural Network For Image Generation, Gregor et al.



Fei-Fei Li, Andrej Karpathy & Justin Johnson'dan uyarlanmıştır

83

Summary

- RNNs allow a lot of flexibility in architecture design
- Common to use LSTM (or GRU): their additive interactions improve gradient flow
- Better/simpler architectures are a hot topic of current research
- Better understanding (both theoretical and empirical) is needed.

Teşekkürler!