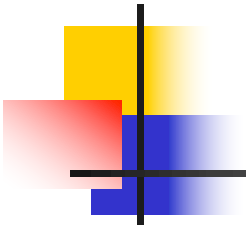
Deep neural architecture and applications (Part III) Week 12: Lectures 58 - 60

Jayanta Mukhopadhyay
Dept. of Computer Science and Engg.

Courtesy: K. Sairam

Ex.1

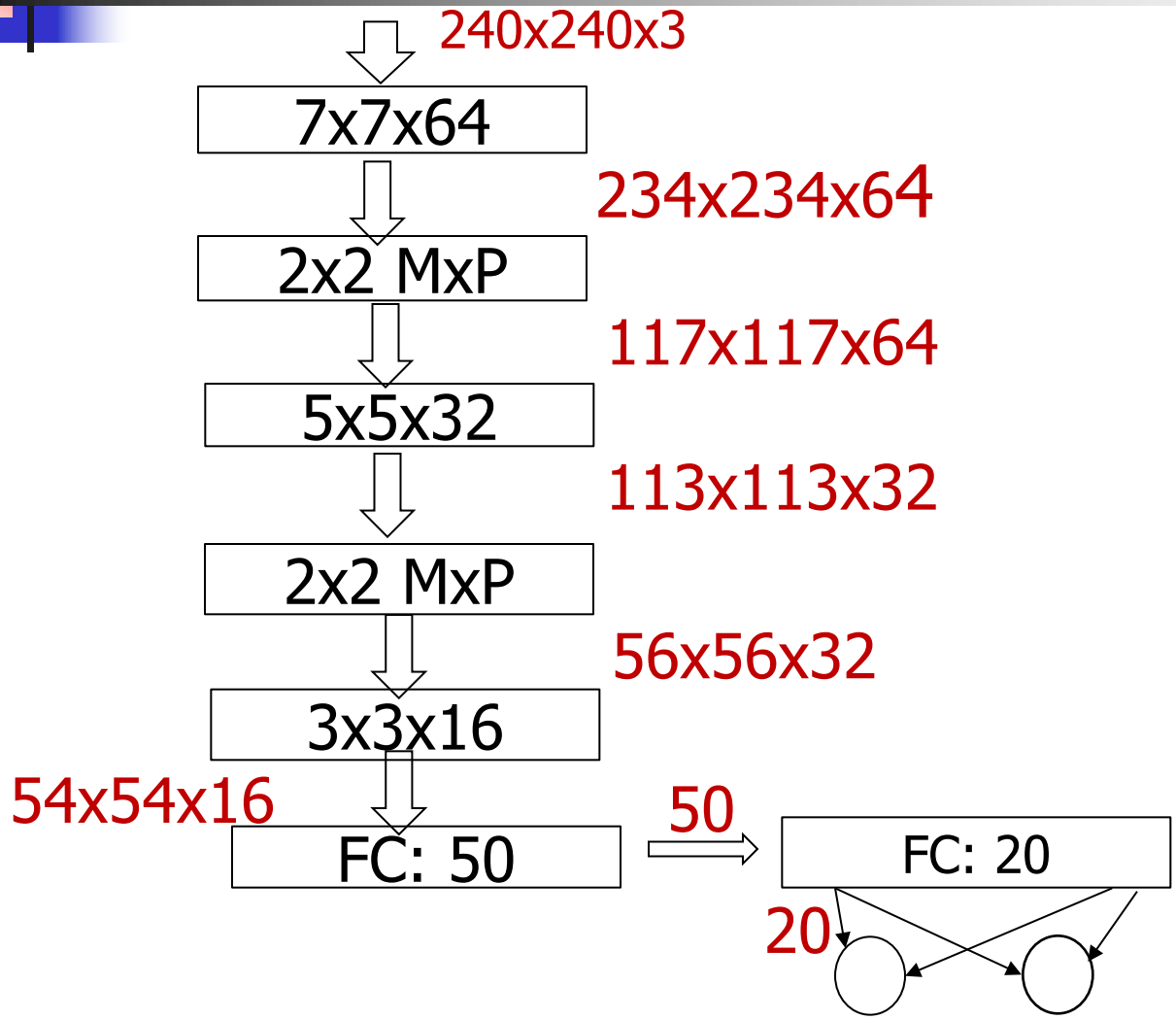
- Consider a deep CNN architecture which takes an image of size **240 x 240 x3 as input** for the purpose of classification into **two classes**. The architecture has
  - **three convolution layers** with filter sizes of **7x7, 5x5, and 3x3**, respectively (the ordering of layers starts from the input). The number of **channels in these layers are 64, 32 and 16**, respectively.
  - **two max pool layers of filter size 2 x 2 with stride 2** in both the directions between the first and second, and second and third convolution layers.
  - **two fully connected (FC) layers** having the **number of neurons as 50 and 20**, respectively. The activation function of each neuron used in this network is the ReLU (Rectified linear unit) function.



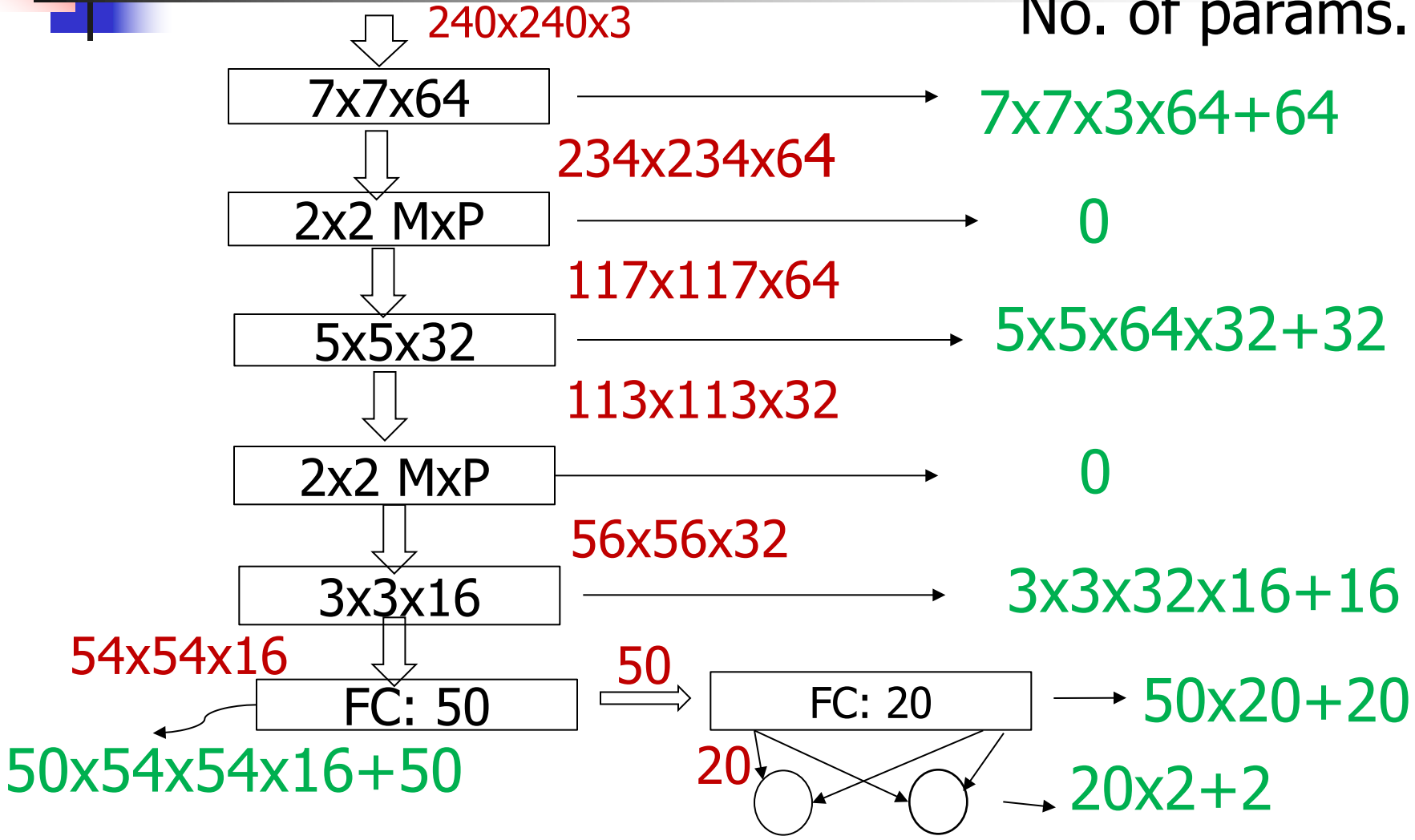
Ex. 1 (contd.)

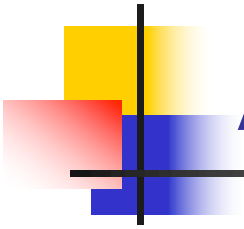
- Answer the following questions.
- (a) Provide a schematic diagram of the architecture of the network.
- (b) Provide the sizes of outputs from each convolution layers.
- (c) Compute the number of parameters that you require to learn in each layer.

Ans. 1


240x240x3
7x7x64
234x234x64
2x2 MxP
117x117x64
5x5x32
113x113x32
2x2 MxP
56x56x32
3x3x16
54x54x16
FC: 50
50
FC: 20
20

Ans. 1

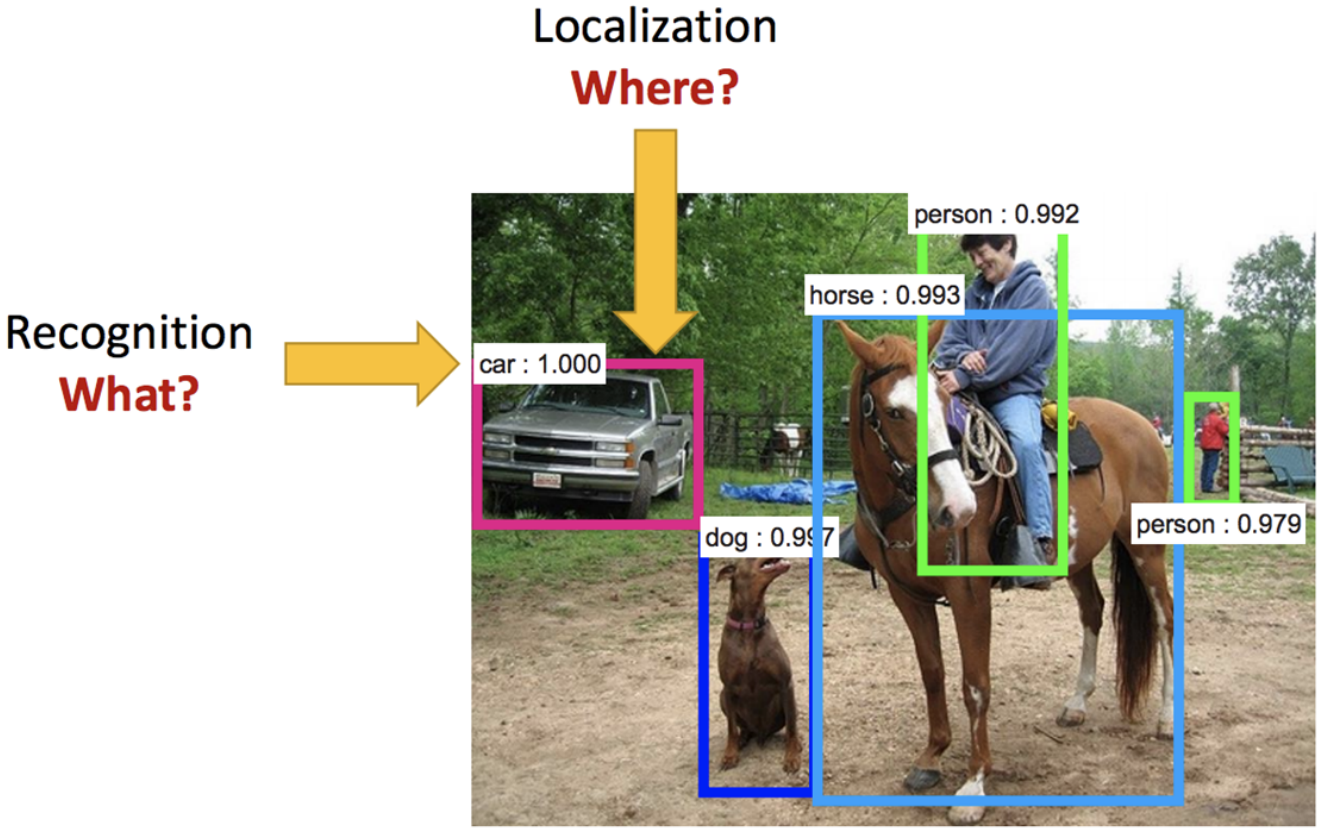

No. of params.
240x240x3
7x7x64
7x7x3x64+64
234x234x64
2x2 MxP
0
117x117x64
5x5x32
5x5x64x32+32
113x113x32
2x2 MxP
0
56x56x32
3x3x16
3x3x32x16+16
54x54x16
FC: 50
50
FC: 20
50x20+20
50x54x54x16+50
20
20x2+2



Applications:

1. Image Classification
2. Object Recognition and Localization
3. Image Segmentation
4. Image Captioning

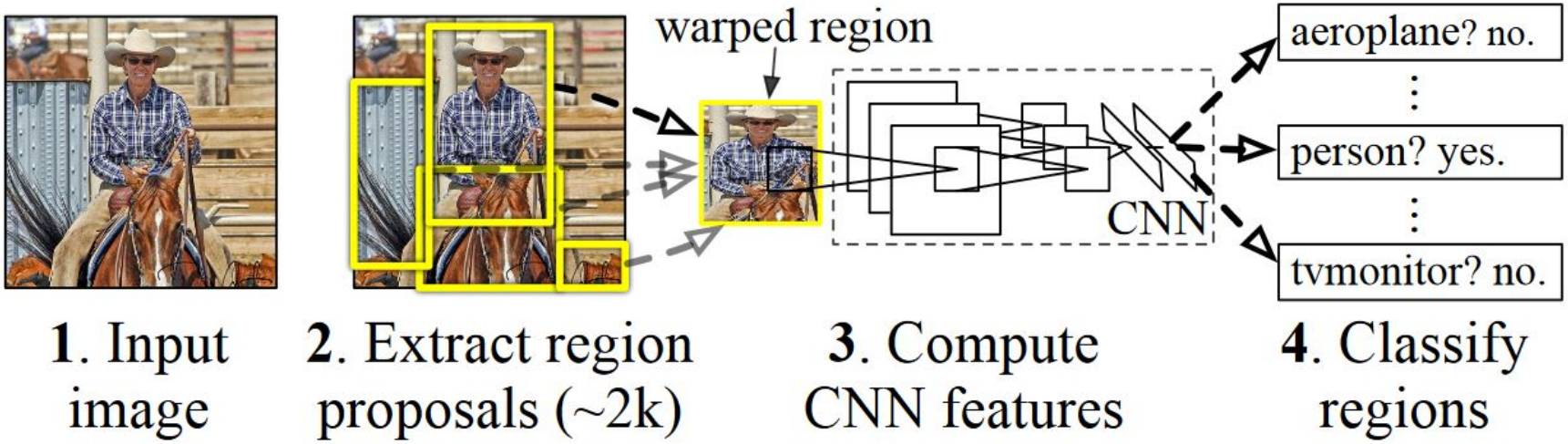
Object Recognition and Localization



Object Recognition and Localization

	Region Proposal	Feature Extraction	Classification
Pre-CNN	Exhaustive	Hand Crafted	Linear
RCNN	Region Proposal	CNN	Linear SVM
Fast RCNN	Region Proposal	Deep	
Faster RCNN	Deep		

RCNN with CNN feature extractor

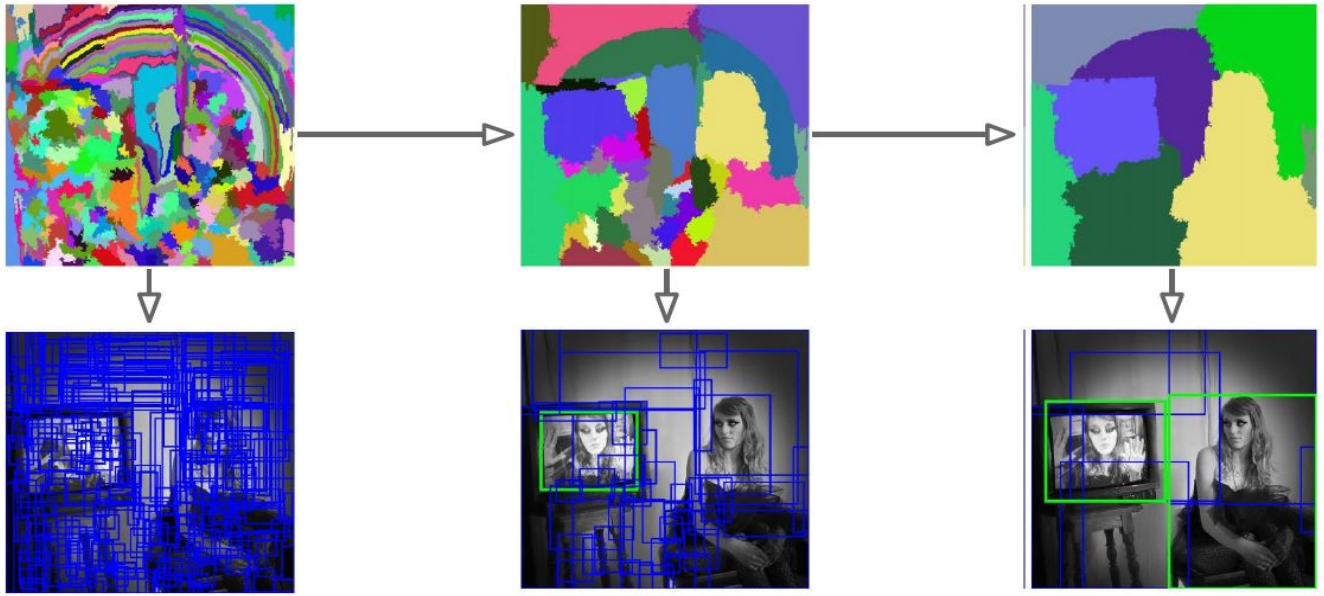


*Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

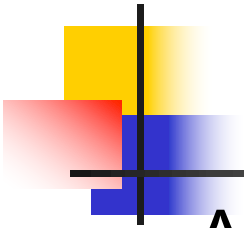
RCNN – Region Proposal

Selective Search:

- Hierarchical grouping based on color, texture, size.
- Bottom-up Segmentation, merging regions at multiple scales.



Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013



RCNN Problems

- Ad hoc training objectives.
- Fine-tune network with softmax classifier (log loss) from pre-trained network in imageNet.
- Train post-hoc linear SVMs (hinge loss).
- Train post-hoc bounding-box regressions (least squares).
- Training is slow, takes a lot of disk space.
- Inference (detection) is slow. Need to run full forward pass of CNN for each region proposal.

RCNN Inference Time

- PropTime + NumProp*ConvTime + NumProp*fcTime

PropTime:

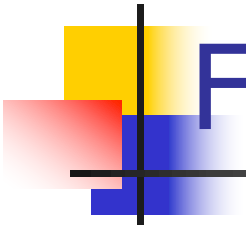
Time taken for generating all proposals.

NumProp:

Number of proposals generated.

fcTime :

Time taken to identify the object in the image.



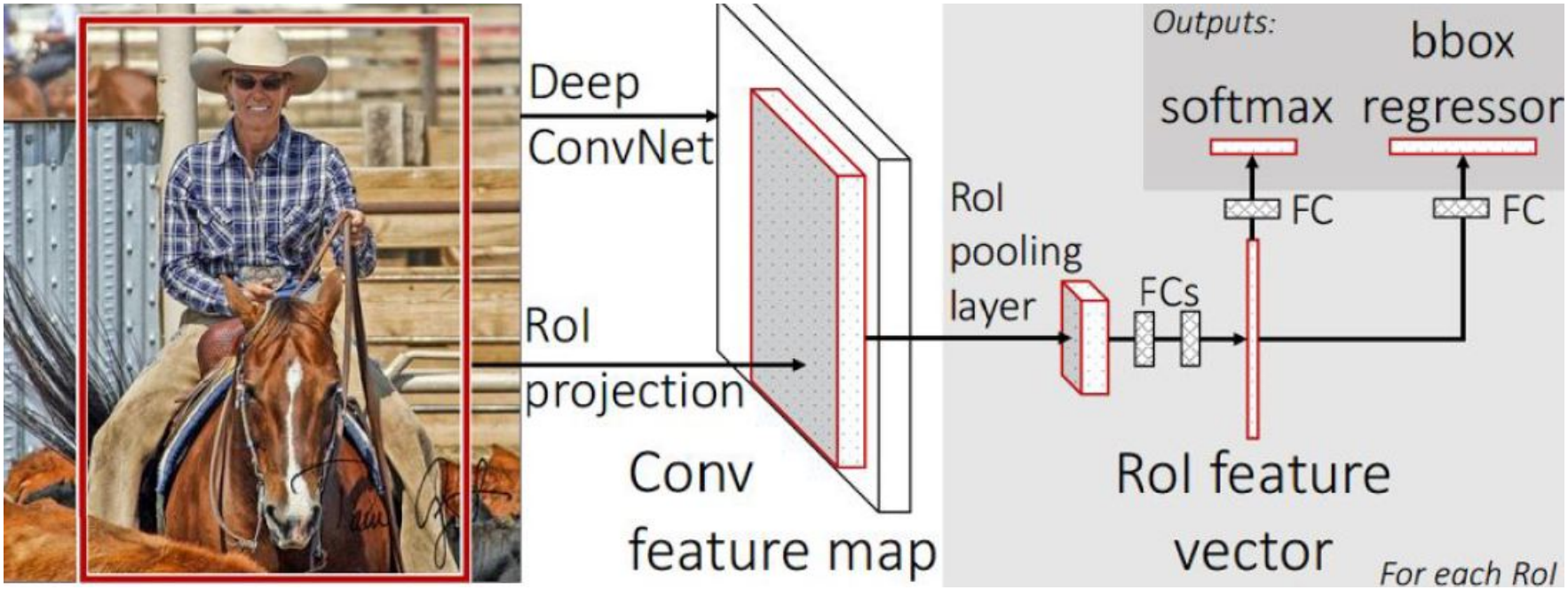
Fast RCNN

1. A Fast R-CNN network takes as input an entire image and a set of object proposals.
2. Computes CNN feature map for the whole image.
3. Processes features maps in ROIs.

Fast RCNN Time: PropTime + 1*ConvTime + NumProp*fcTime

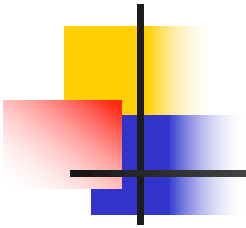
*Girshick, “Fast R-CNN”, ICCV 2015.

Fast RCNN



Fast RCNN Time: PropTime + 1*ConvTime + NumProp*fcTime

*Girshick, "Fast R-CNN", ICCV 2015.



Fast RCNN

- Each feature vector to a sequence of fully connected (fc) layers.
- Two sibling output layers:
  - A layer with softmax probability estimates over K object classes plus a catch-all “background” class
  - another layer producing four real-valued numbers for each of the K object classes.

*Girshick, “Fast R-CNN”, ICCV 2015.

RCNN

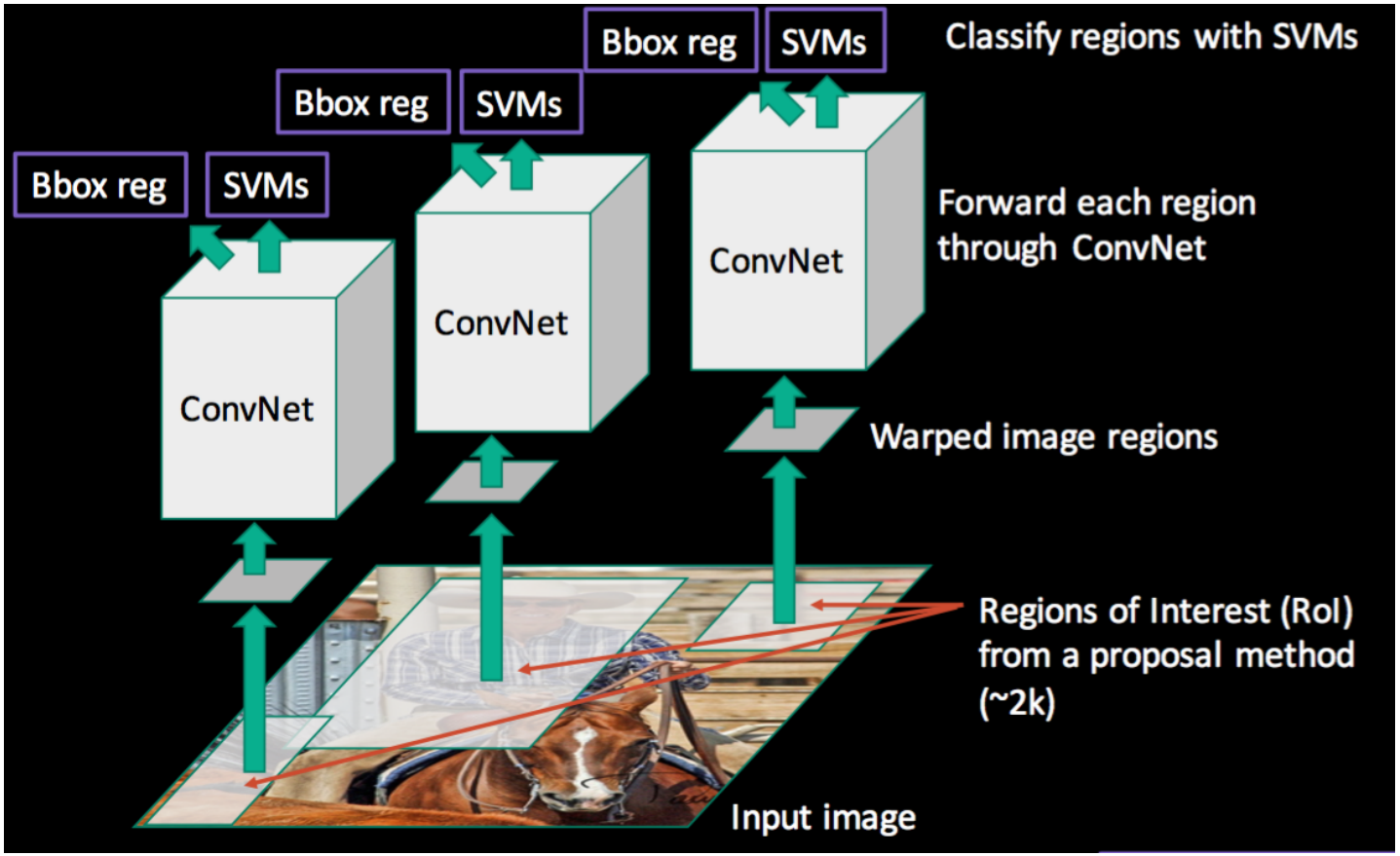
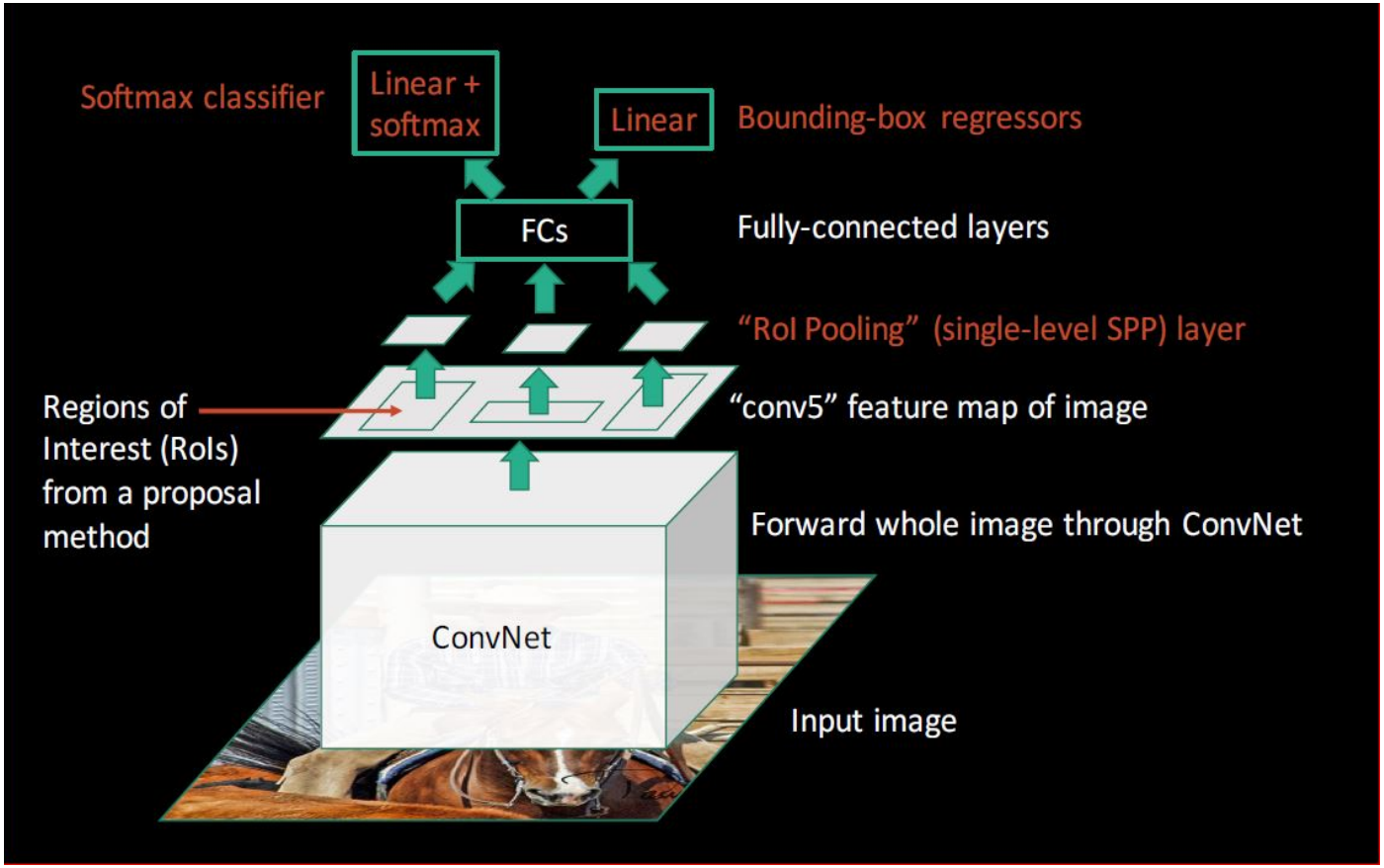


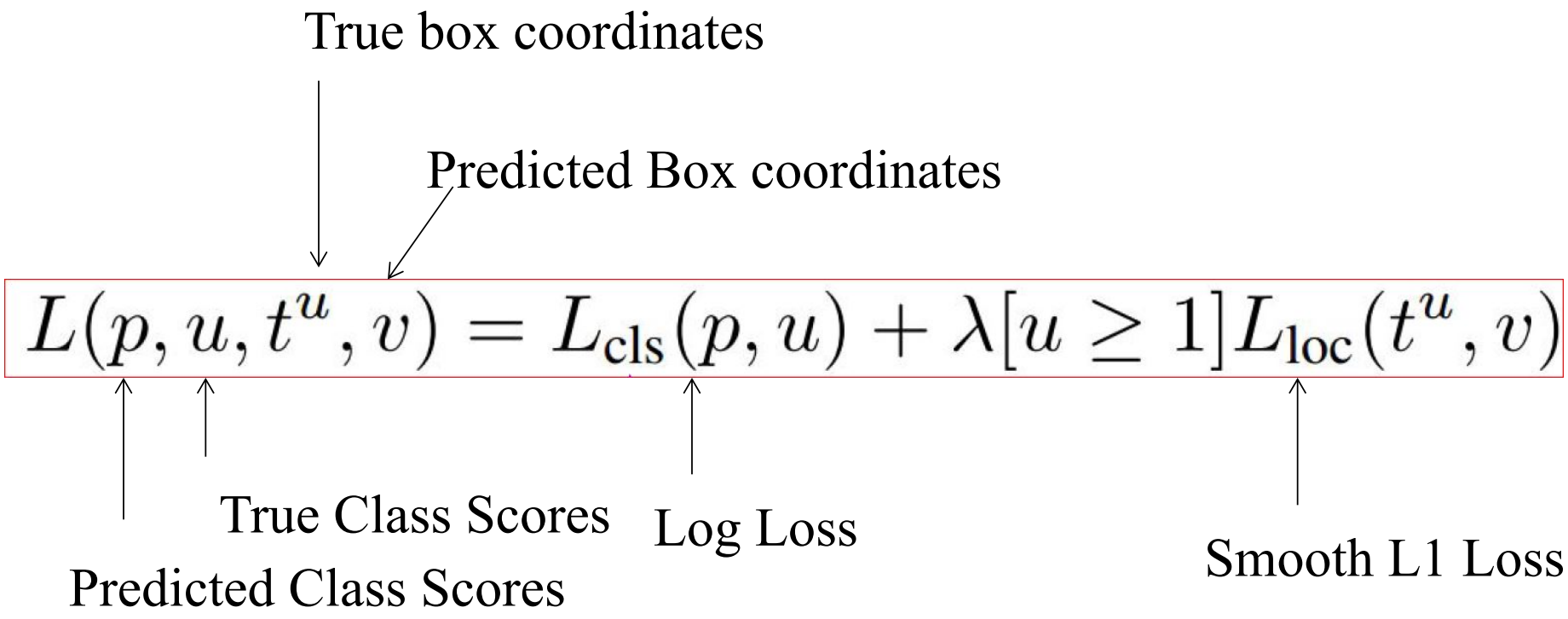
Image: Ross Girshick, 2015.

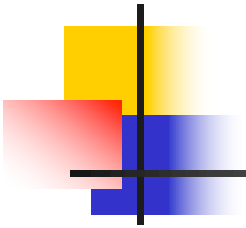
Fast RCNN



* Image: Ross Girshick, 2015.

Fast RCNN Loss



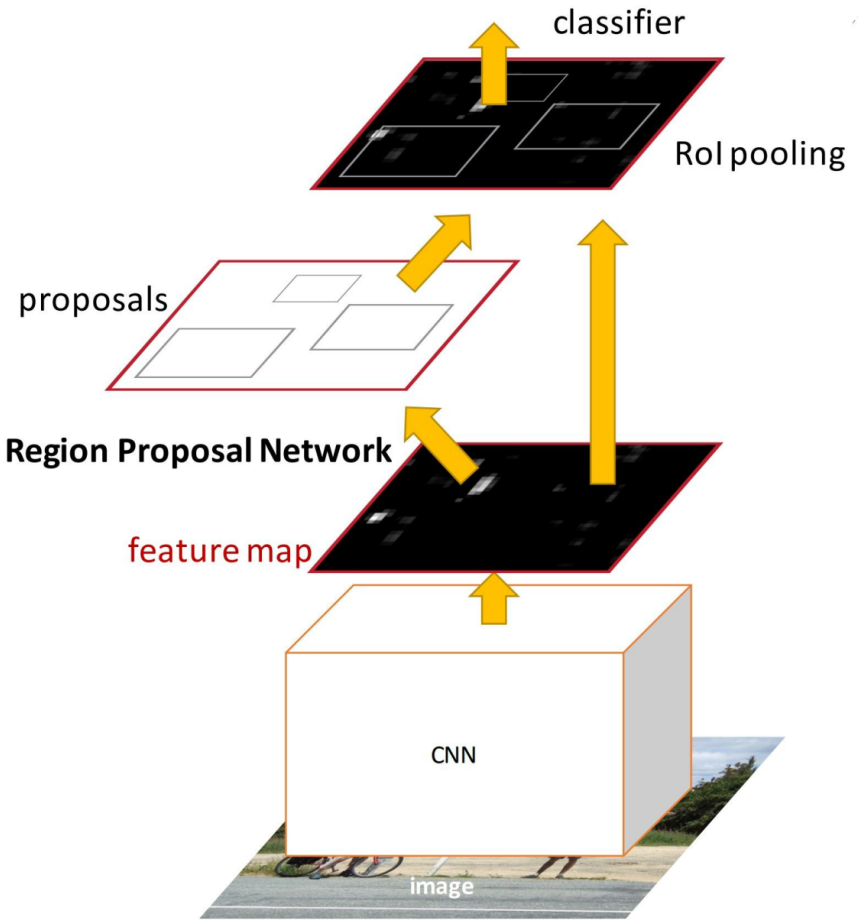


Faster RCNN

- Insert a **Region Proposal Network (RPN)** after the last convolutional layer.
- RPN trained to produce region proposals directly; no need for external region proposals!
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN.

*Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Faster RCNN



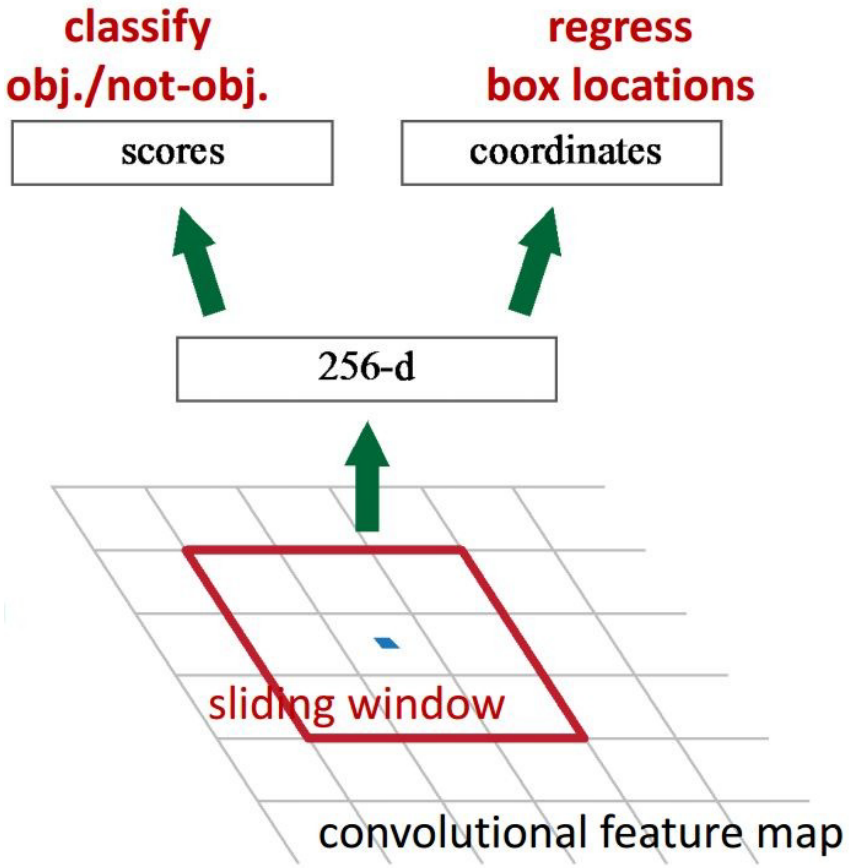
*Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Region Proposal Network

- Slide a small window on the feature map
- Build a small network for:
  - classifying object or not-object, and
  - regressing bbox locations
- Initial localization:
  - Position of the sliding window w.r.t image.
- Finer localization:
  - Box regression performs finer localization w.r.t this sliding window.

Faster RCNN Approach Time: 1*ConvTime + NumProp*fcTime

Region Proposal Network

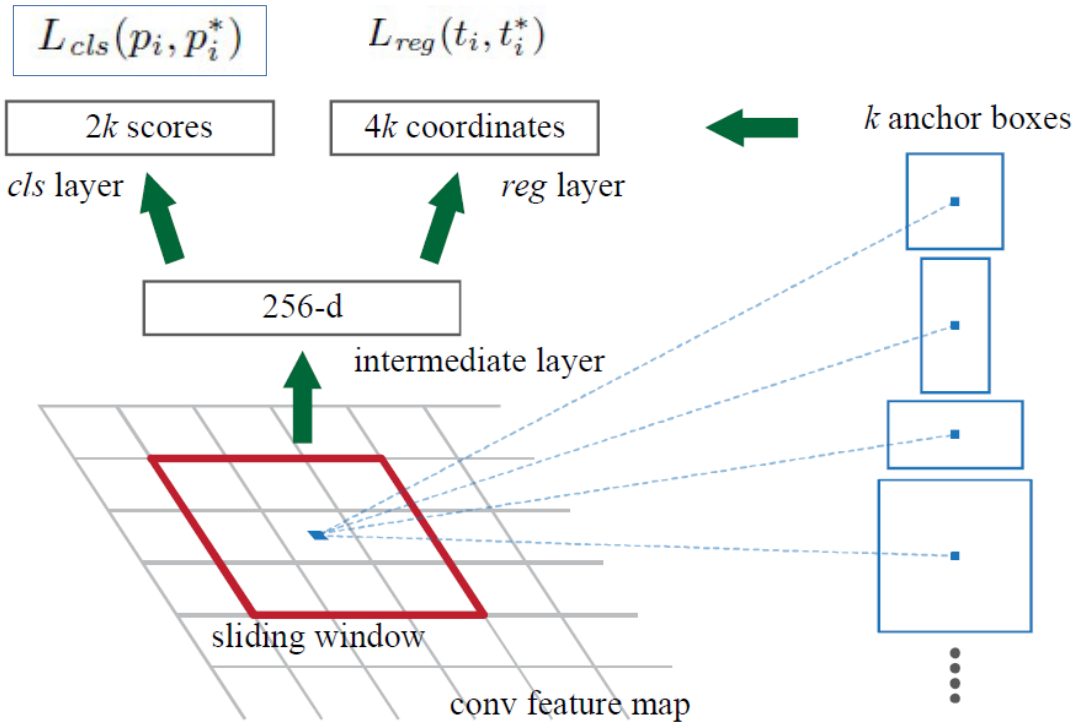


Faster RCNN Approach Time: 1*ConvTime + NumProp*fcTime

RPN Loss

2 class Softmax cross entropy loss
Either object or not object

- pi* = 1 if IoU > 0.7 (it is object)
- pi* = 0 if IoU < 0.3 (it is not object)
- otherwise, do not contribute to loss



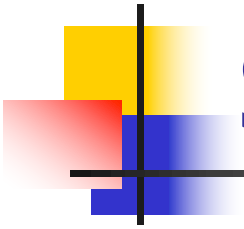
$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Semantic Segmentation

- Label each pixel in the image with a category label.
- Don’t differentiate instances, only care about pixels.

Instance Level Semantic Segmentation

- Even differentiate instances

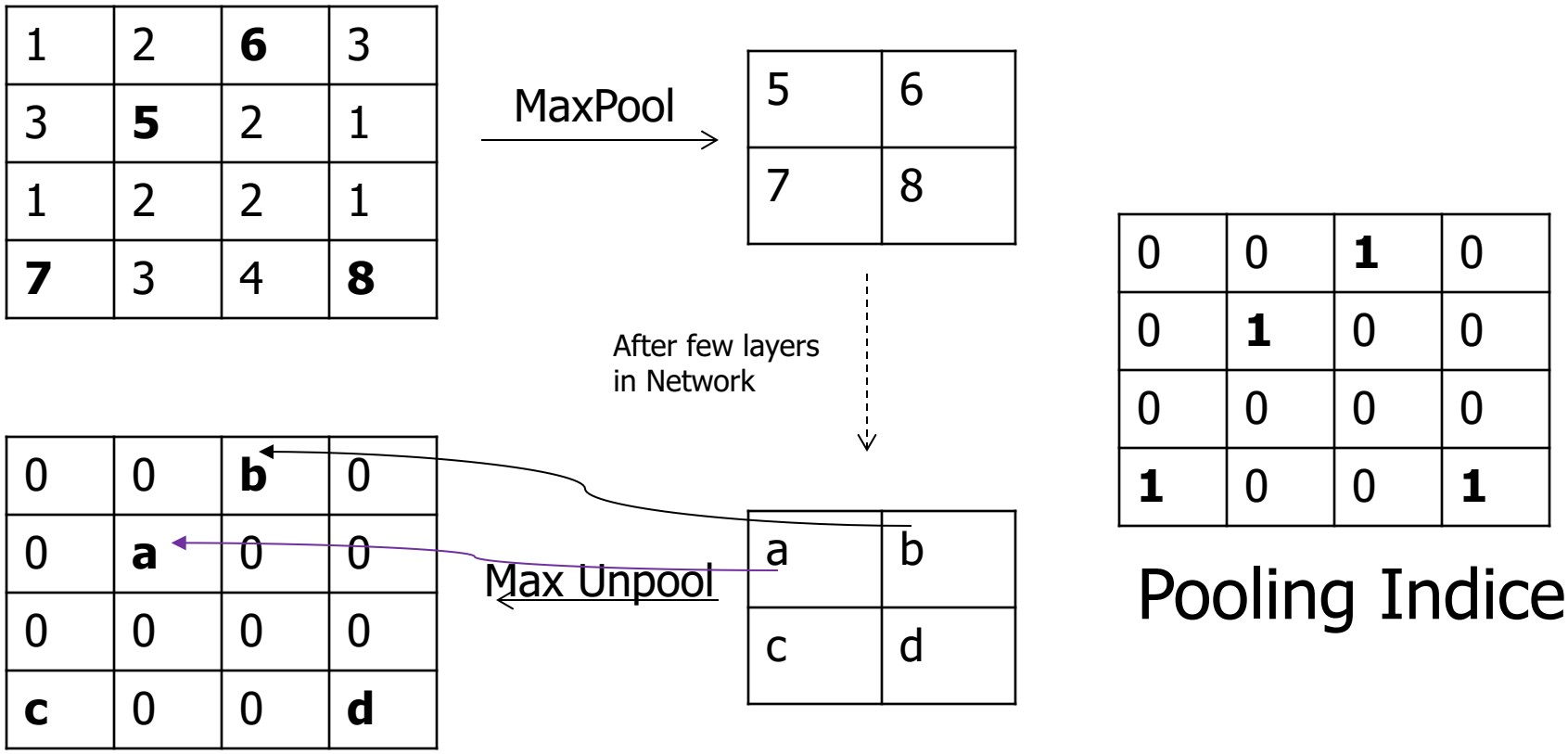


Semantic Segmentation

Building Blocks of CNNs:

- Convolution
- Down-Sampling
  - MaxPool, AvgPool, Strided Convolution (S>1)
- Up-Sampling
  - UnPooling, Upconvolution

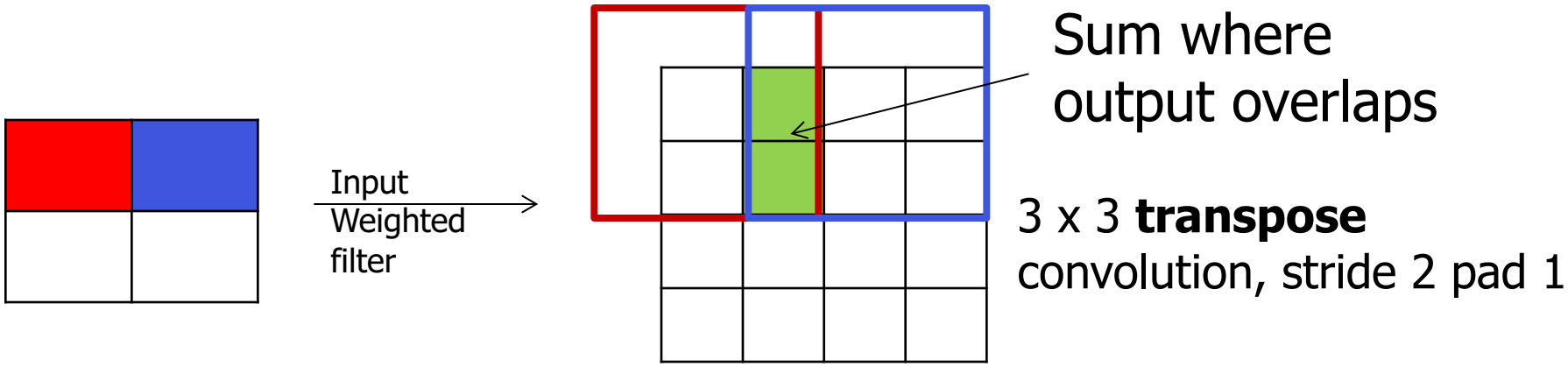
Up- Sampling: Max Unpooling



Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Up- Sampling: Upconvolution

- Also called as
  - Transpose Convolution, Fractionally strided convolution, Backward strided convolution, Deconvolution.
- Output contains copies of the filter weighted by the input, summing at where it overlaps in the output



Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Convolution as matrix Multiplication

1D Conv, filter size = 3, stride = 1, padding = 1

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + z \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Convolution

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

Transpose Convolution

Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Convolution as matrix Multiplication

1D Conv, filter size = 3, stride = 2, padding = 1

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Convolution

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Transpose Convolution

Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Ex 2

- Given the following pooling indices Max unpool by 2x2 of the input block X as given below.

Pooling indices:

0	1	0	1
0	0	0	0
0	0	1	0
0	1	0	0

X

2	6
30	45

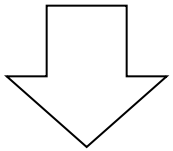
Ans. 2

Pooling indices

0	1	0	1
0	0	0	0
0	0	1	0
0	1	0	0

2	6
30	45

Input X



Max-unpool

0	2	0	6
0	0	0	0
0	0	45	0
0	30	0	0

Ex 3

- Up-convolve the following input x using the filter h.

5	6
7	8

x

0	0.2	0
0.2	0.2	0.2
0	0.2	0

Origin of the filter

h

Ans. 3

5	6
7	8

x

0	0.2	0
0.2	0.2	0.2
0	0.2	0

h

0	**1+0**	**0+1.2**	**0**
1+0	1+1.2+ 1.4+0	1+1.2+ 0+1.6	1.2+0
0+1.4	1+0+1. 4+1.6	1+1.2+ 1.4+1.6	0+1.6
0	1.4+0	0+1.6	0

Ans. 3

5	6
7	8

x

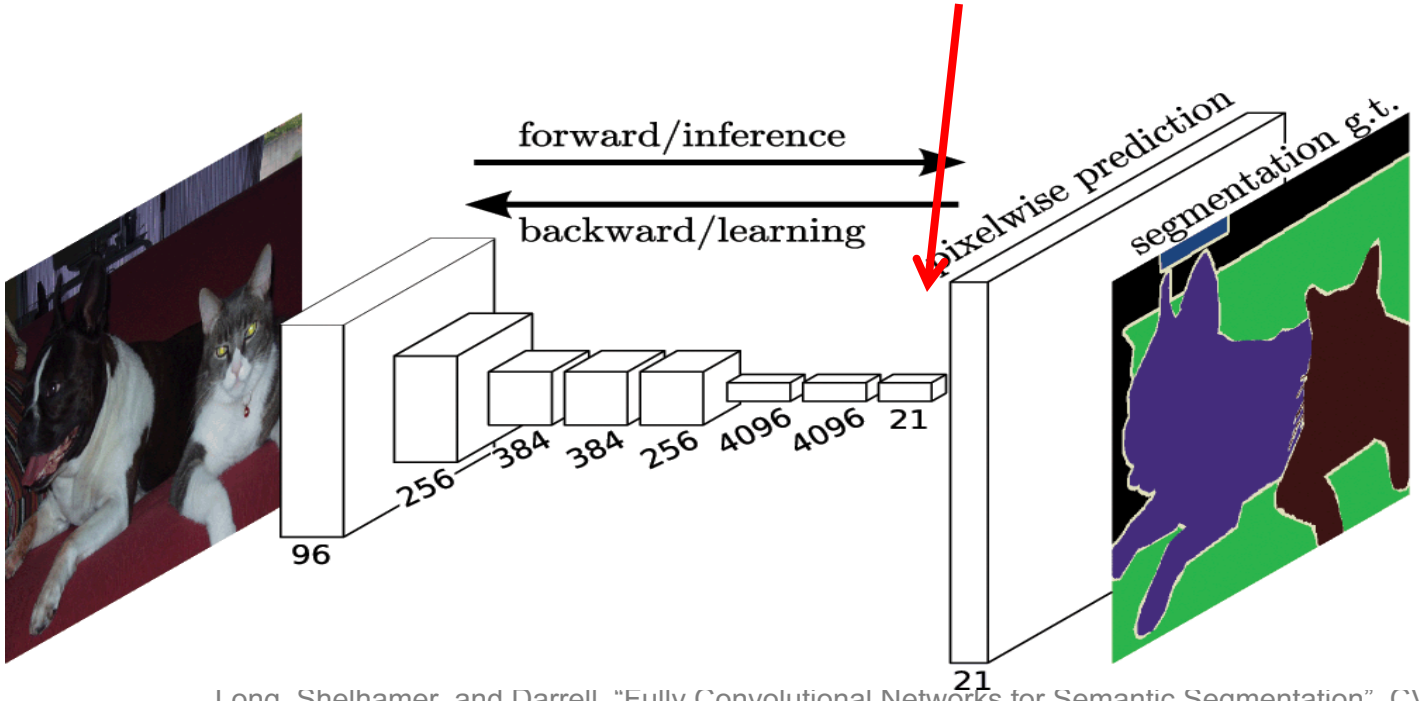
0	0.2	0
0.2	0.2	0.2
0	0.2	0

h

0	1	1.2	0
1	3.6	3.8	1.2
1.4	7.6	5.2	1.6
0	1.4	1.6	0

Semantic Segmentation

- The upsampling of learned low resolution semantic feature maps is done using upconvolutions which are initialized with billinear interpolation filters.



Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Semantic Segmentation (Encoder-Decoder)

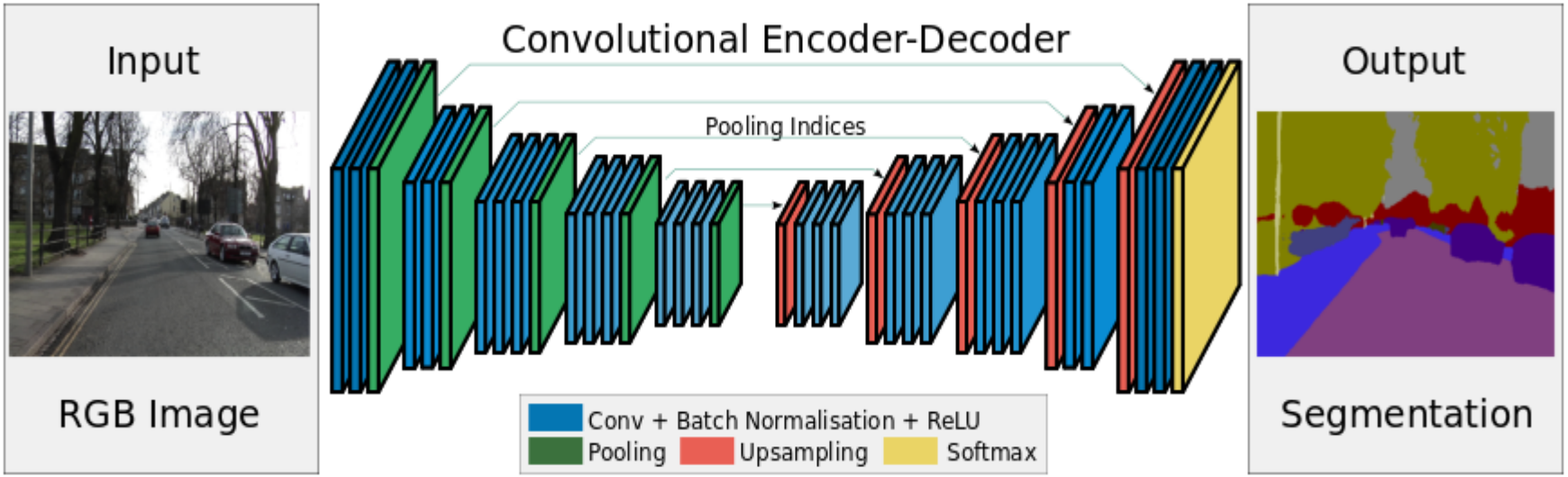
Encoder

- Takes an input image and generates a high-dimensional feature vector
- Aggregate features at multiple levels

Decoder

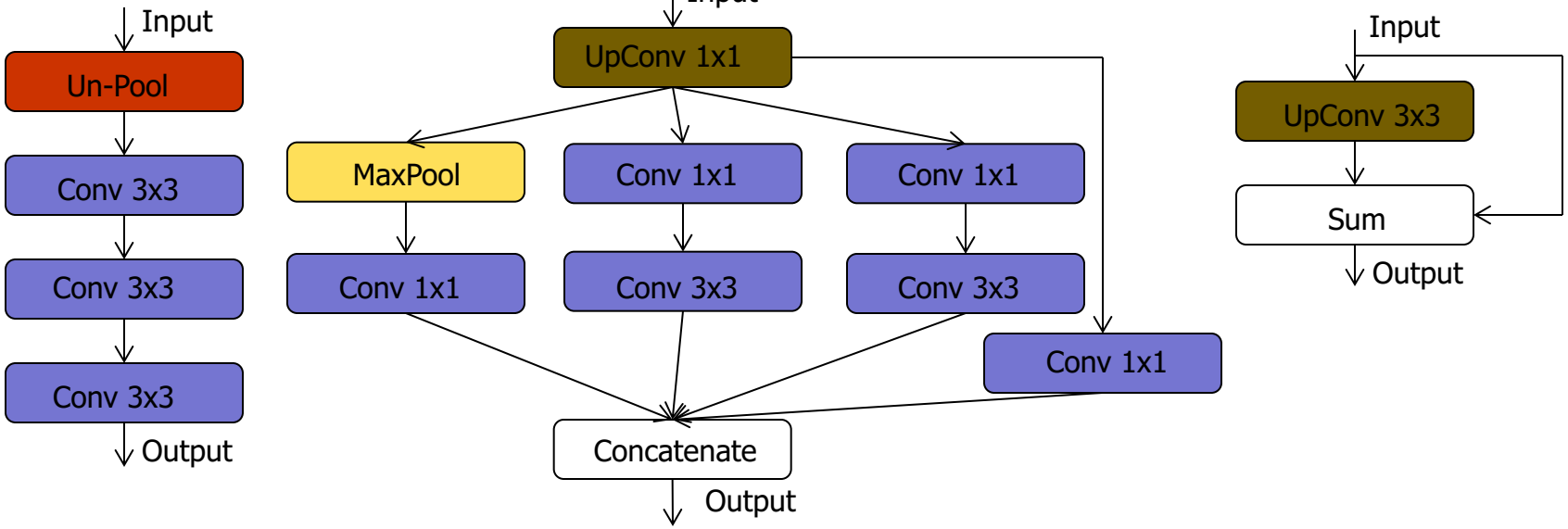
- Takes a high-dimensional feature vector and generates a semantic segmentation mask
- Decode features aggregated by encoder at multiple levels.
- Semantically project the discriminative features (lower resolution) learnt by the encoder onto the pixel space (higher resolution) to get a dense classification.

Semantic Segmentation (Encoder-Decoder)



SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, Vijay Badrinarayanan, et al. PAMI, 2017

Semantic Segmentation (**Decoder**): Various options



VGG GoogleNet ResNet

U-Net architecture

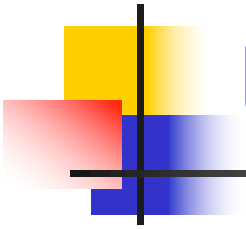
- Computation in two successive stages.
  - Contracting path and expansive path.
  - Contraction by downsampling
  - Expansion by upsampling.
- Contractive path: repeated application of two 3x3 convolutions each followed by
  - a rectified linear unit (ReLU), and
  - a 2x2 max pooling operation with stride 2.
  - At each downsampling double the number of feature channels.

U-Net: Convolutional Networks for Biomedical Image Segmentationm, Olaf Ronneberger, Philipp Fischer, Thomas Brox, https://arxiv.org/abs/1505.04597

U-Net architecture

- The expansive path: repeated upsampling of the feature map followed
  - by a 2x2 convolution (“up-convolution") that halves the number of feature channels,
  - a concatenation with the correspondingly cropped feature map from the contracting path, and
  - two 3x3 convolutions, each followed by a ReLU.
- The cropping is necessary due to the loss of border pixels in every convolution.

U-Net: Convolutional Networks for Biomedical Image Segmentationm, Olaf Ronneberger, Philipp Fischer, Thomas Brox, https://arxiv.org/abs/1505.04597

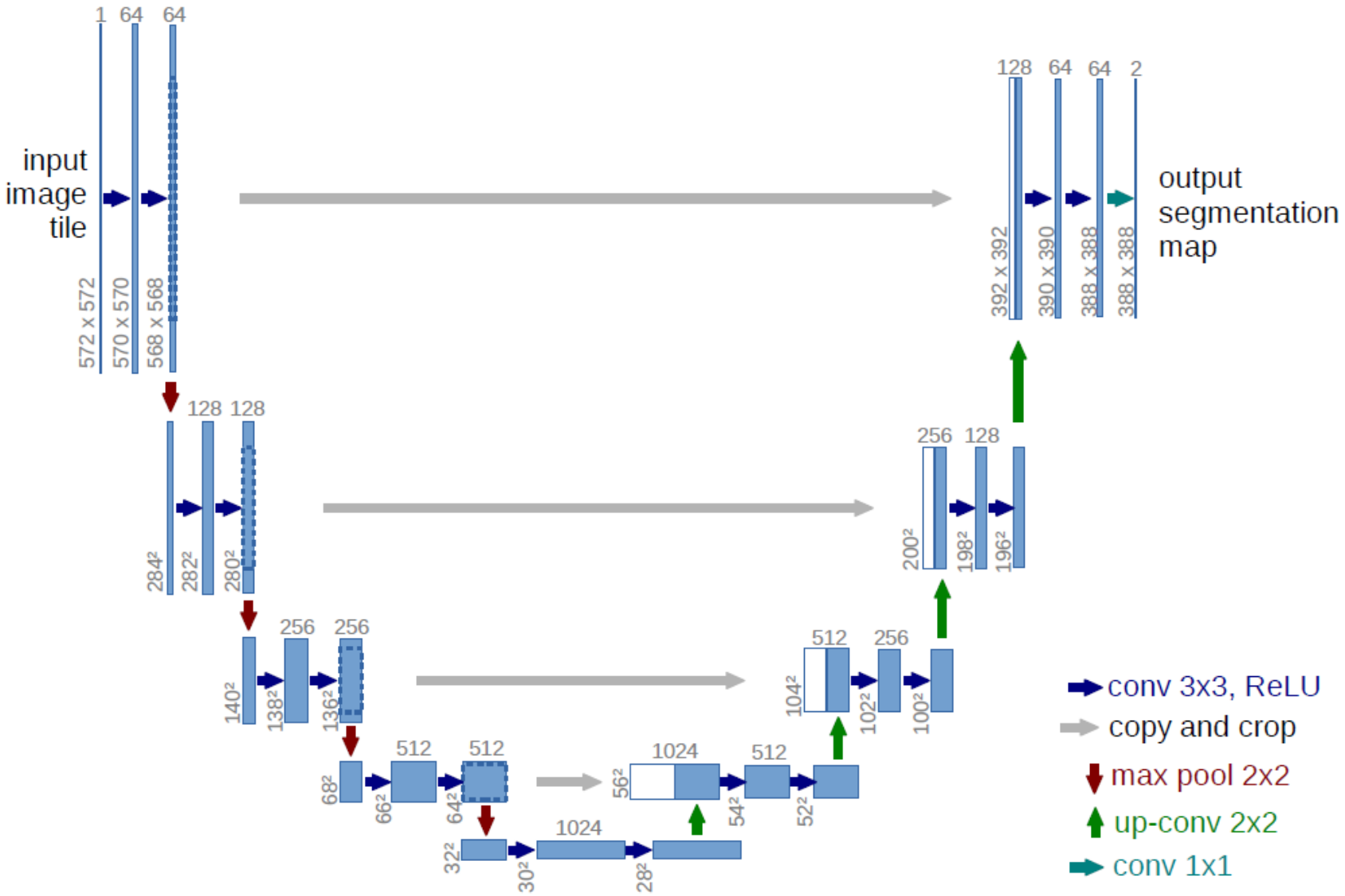


U-Net architecture

- At the final layer a 1x1 convolution used to map each 64-component feature vector to the desired number of classes.
- In total the network has 23 convolutional layers.
- Data augmentation needed to make the network robust to transformation and noise contamination.
  - Simulated deformation on training images.

U-Net: Convolutional Networks for Biomedical Image Segmentationm, Olaf Ronneberger, Philipp Fischer, Thomas Brox, https://arxiv.org/abs/1505.04597

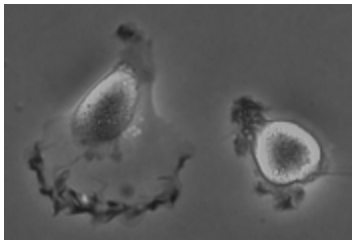
U-Net architecture

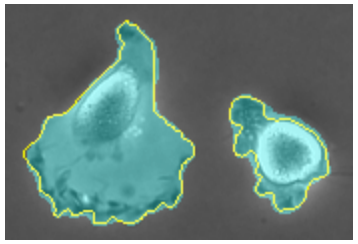


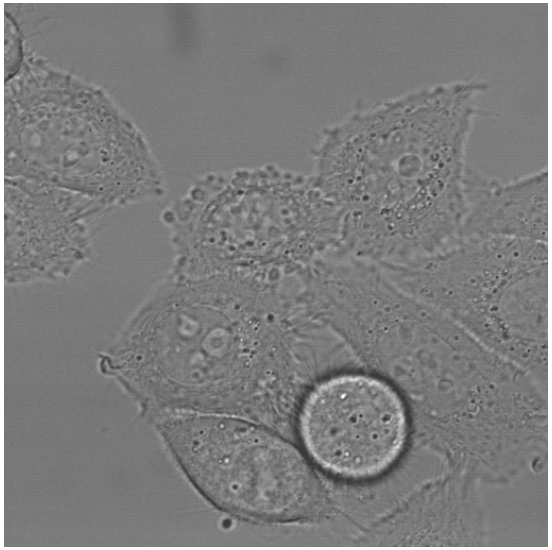
U-Net: Convolutional Networks for Biomedical Image Segmentationm, Olaf Ronneberger, Philipp Fischer, Thomas Brox, https://arxiv.org/abs/1505.04597

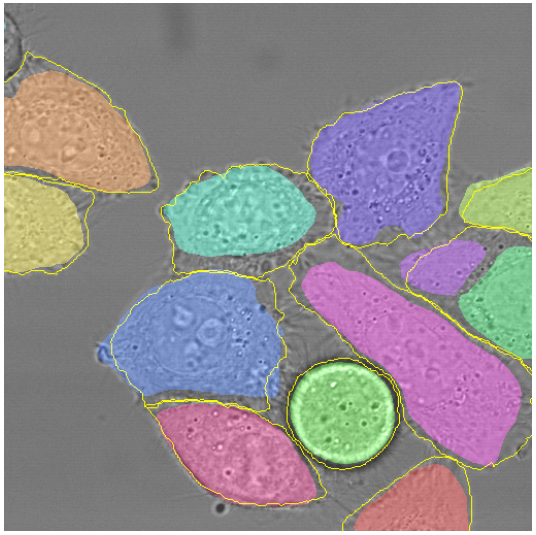
Results

Ground truth (Manual)
Yellow border
Color parts: Segmented results



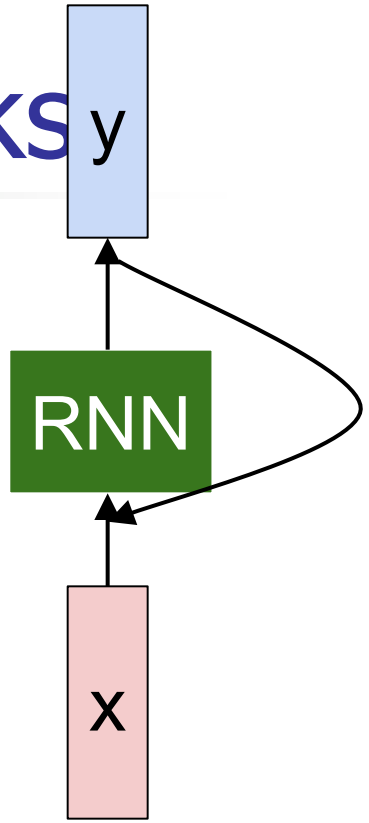




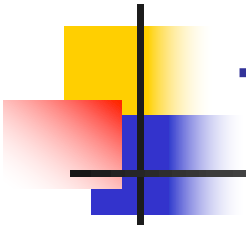


Recurrent Neural Networks

- Conventional NN (including CNN):
  - Feed forward, No feedback.
  - Fixed size input and output.
  - Fixed number of layers
- Recurrent Neural Networks
  - Feedback from output of a neuron.
  - Sequences in the input, the output, or in the most general case both.
  - Equivalent to unfolded feedforward network of infinite number of layers.



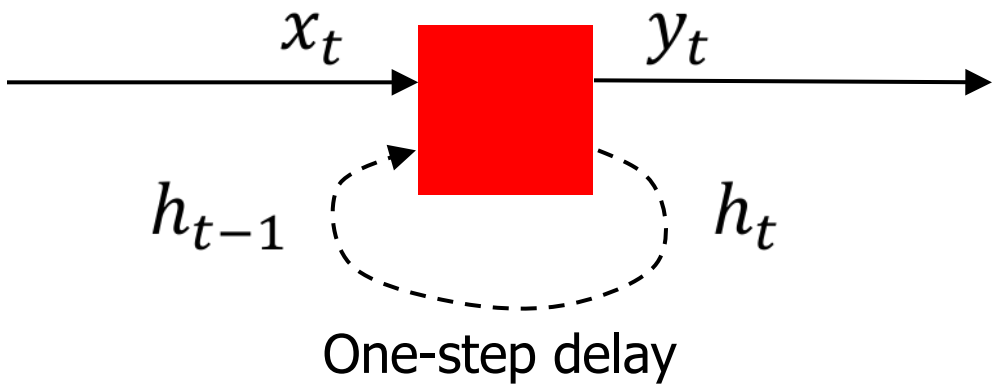
Recurrent Neural Networks: A few Applications



- Semantic labelling of a sequence
  - Classification with the input as a sequence.
- Prediction in a sequence
  - using previous video frames to inform the understanding of the present frame.
  - a language model tries to predict the next word based on the previous ones.
- Sequence translation / generation
  - Input a sequence and output a sequence.

Recurrent Neural Networks

Recurrent networks introduce cycles and a notion of time.

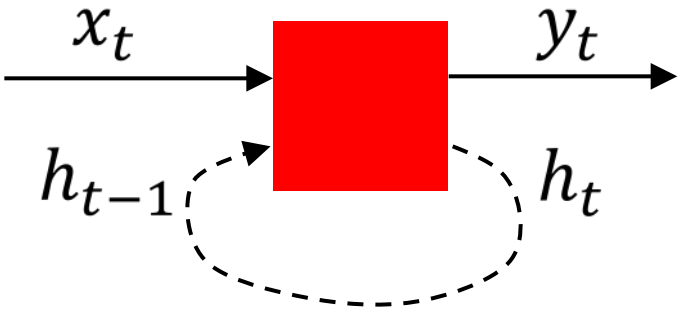


- They are designed to process sequences of data $x_1, \ldots, x_n$ and can produce sequences of outputs $y_1, \ldots, y_m$.

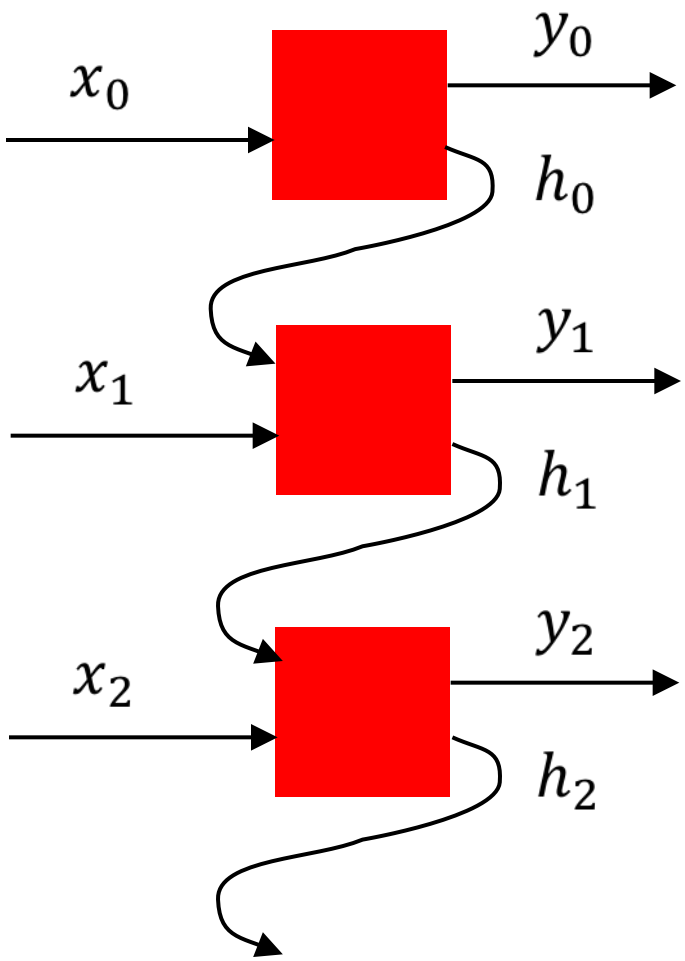
Courtesy: John Canny, UC Berkley.

Unrolling of RNN

Number of stages in unrolling depends on the input sequence length.

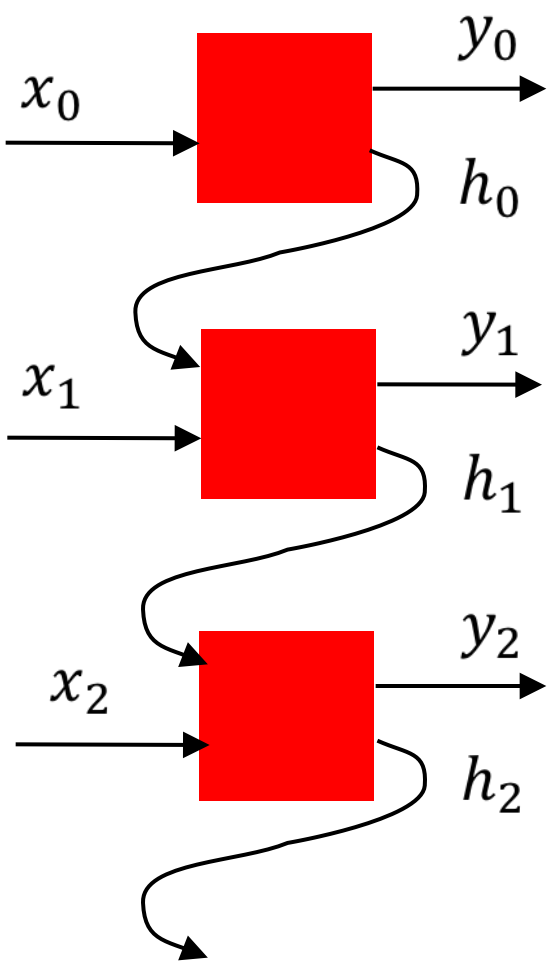


One-step delay

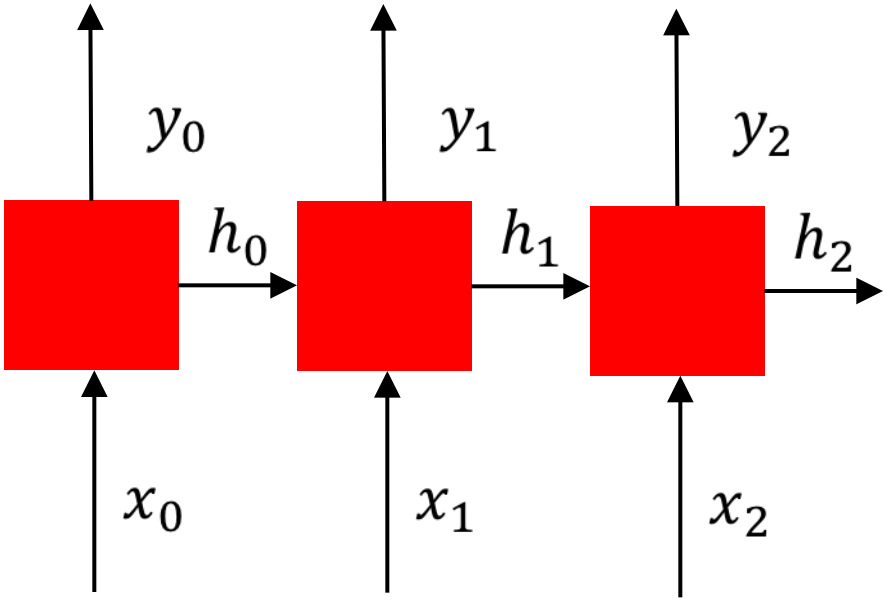


Courtesy: John Canny, UC Berkley.

Unrolling of RNN



Usually drawn as:



Learning algorithm: Back propagation through time (BPTT)

Courtesy: John Canny, UC Berkley.

Recurrent Neural Networks

- usually want to predict a vector at some time steps
- Process a sequence of vectors **x** by applying a recurrence formula at every time step.

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

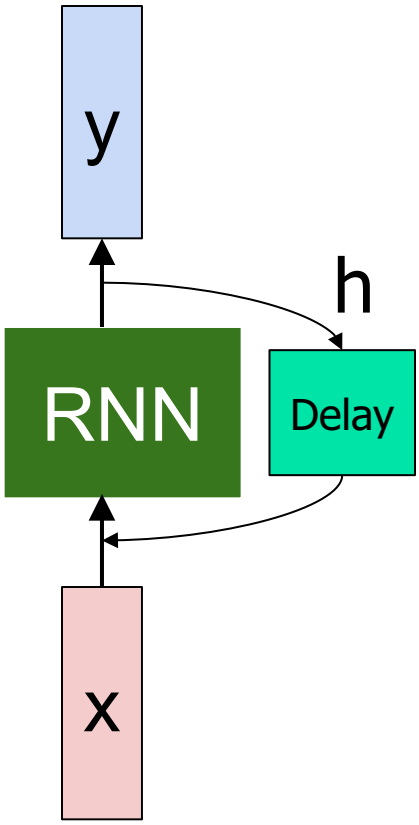
new state

old state

input vector at some time step

some function with parameters W

$$y_t = W_{hy} h_t$$



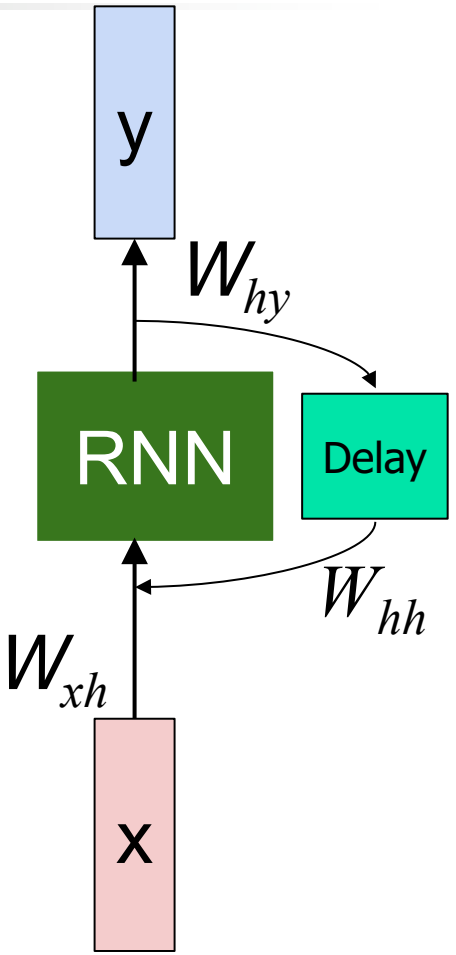
The same function and the same set of parameters are used at every time step.

Recurrent Neural Networks

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

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

$$y_t = W_{hy} h_t$$

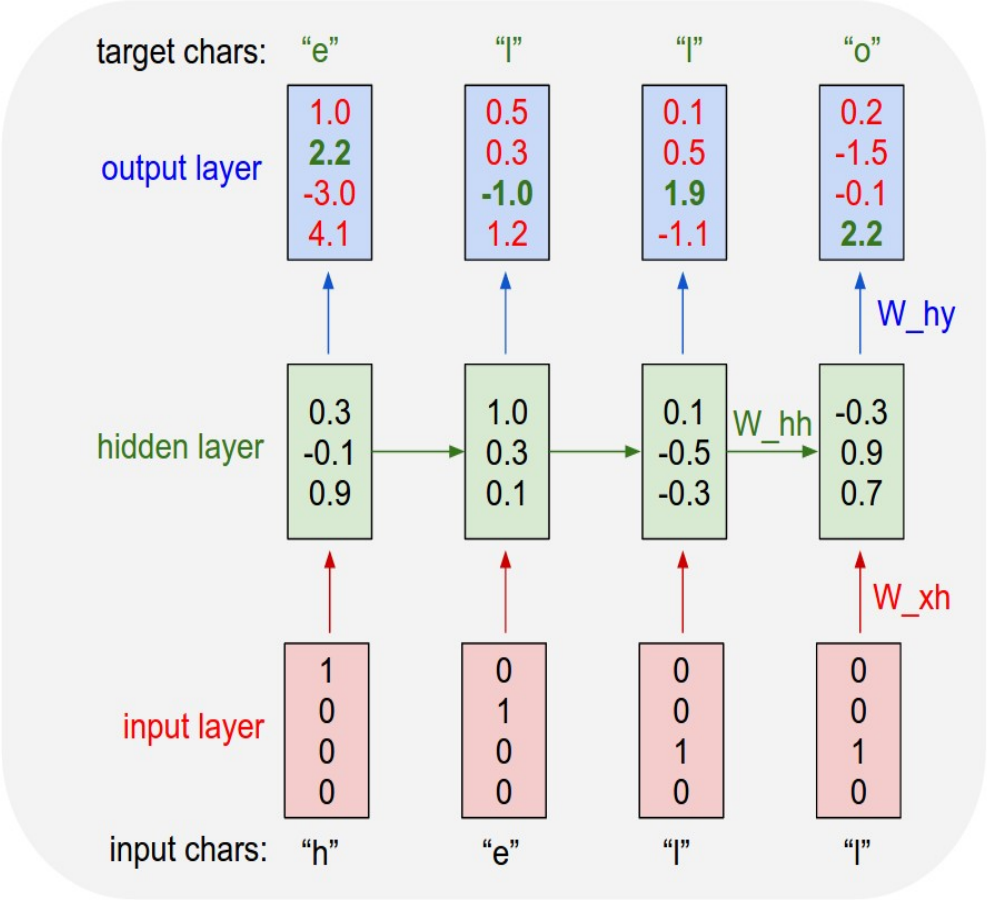


Recurrent Neural Network: An example

Vocabulary: [h,e,l,o]

Example training sequence:
“hello”

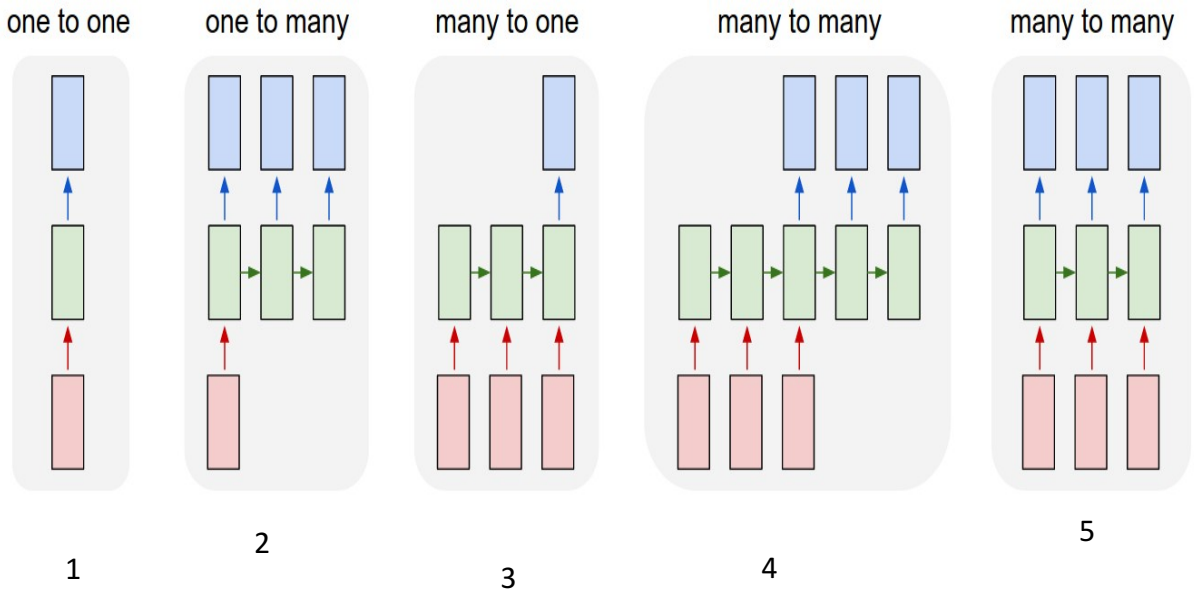
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$



Predicting next character of a word.

Courtesy: Andrej Karpathy

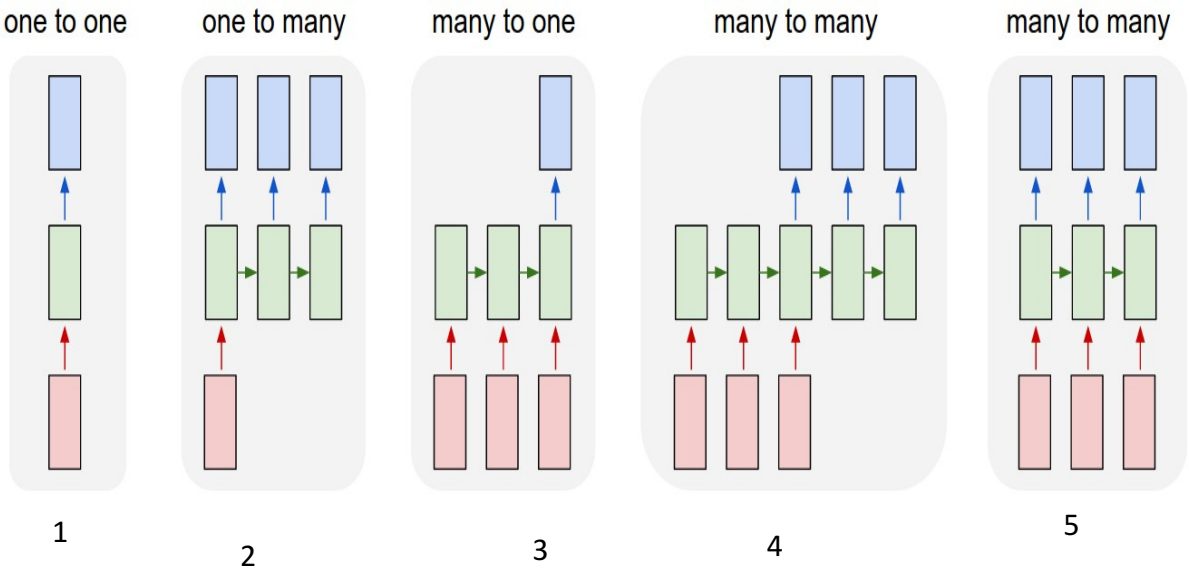
Examples of Recurrent Neural Networks



- Each rectangle is a vector and arrows represent functions (e.g. matrix multiply).
- Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state

Courtesy: Andrej Karpathy

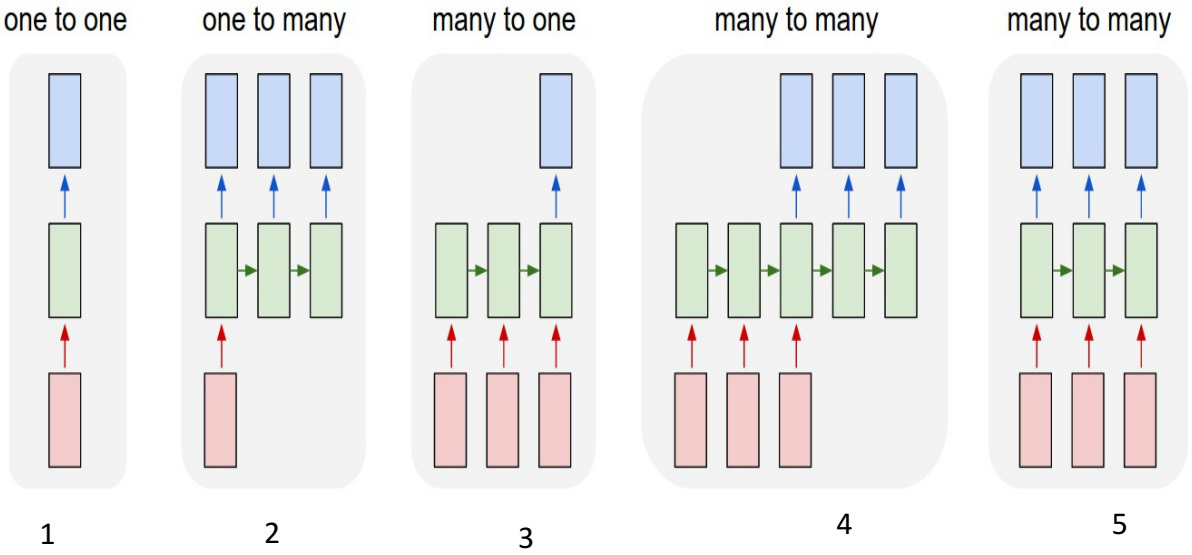
Examples of Recurrent Neural Networks



1. Standard mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification).
2. Sequence output (e.g. image captioning takes an image and outputs a sentence of words).

Courtesy: Andrej Karpathy

Examples of Recurrent Neural Networks



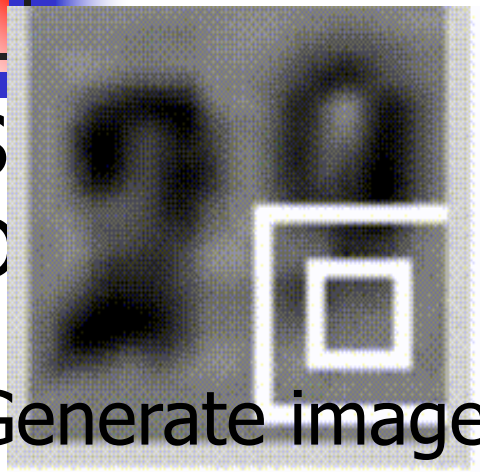
3. Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).
4. Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French).
5. Synced sequence input and output (e.g. video classification where we wish to label each frame of the video).

Courtesy: Andrej Karpathy

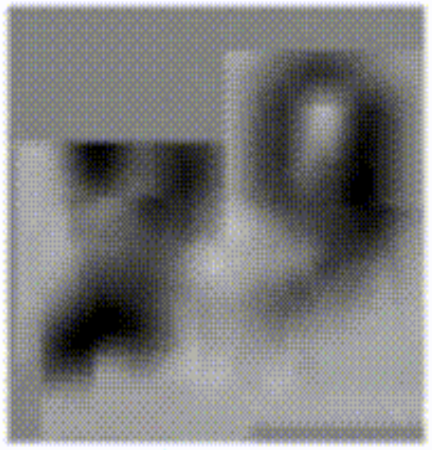
Recurrent Neural Networks

S ocessin

D

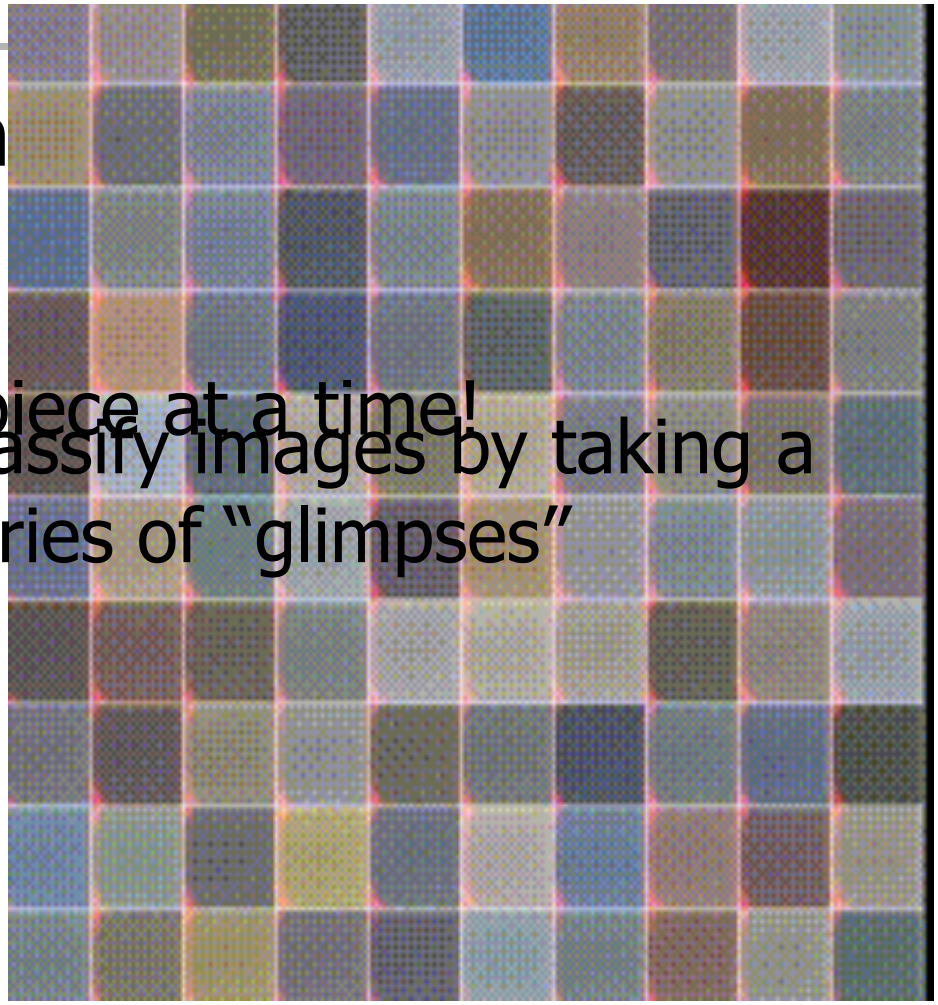


Generate images one piece at a time!



*Multiple Object Recognition with Visual Attention, Ba et al.

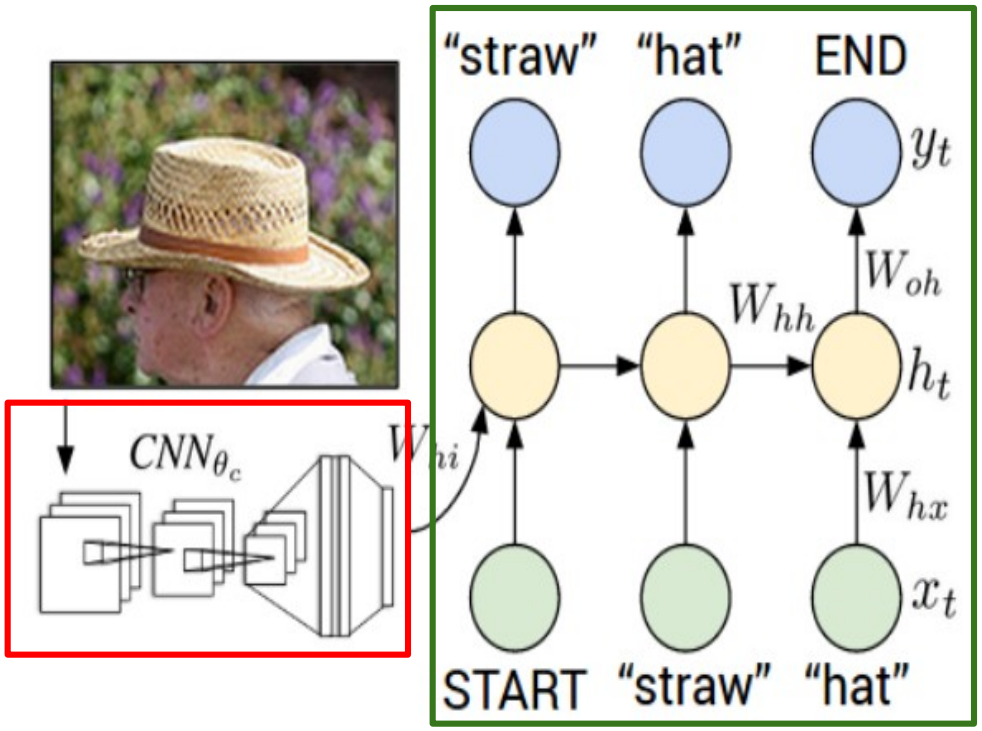
Classify images by taking a series of "glimpses"



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

Courtesy: Andrej Karpathy

Image Captioning



* Show and Tell: A Neural Image Caption Generator, Vinyals et al.

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

X

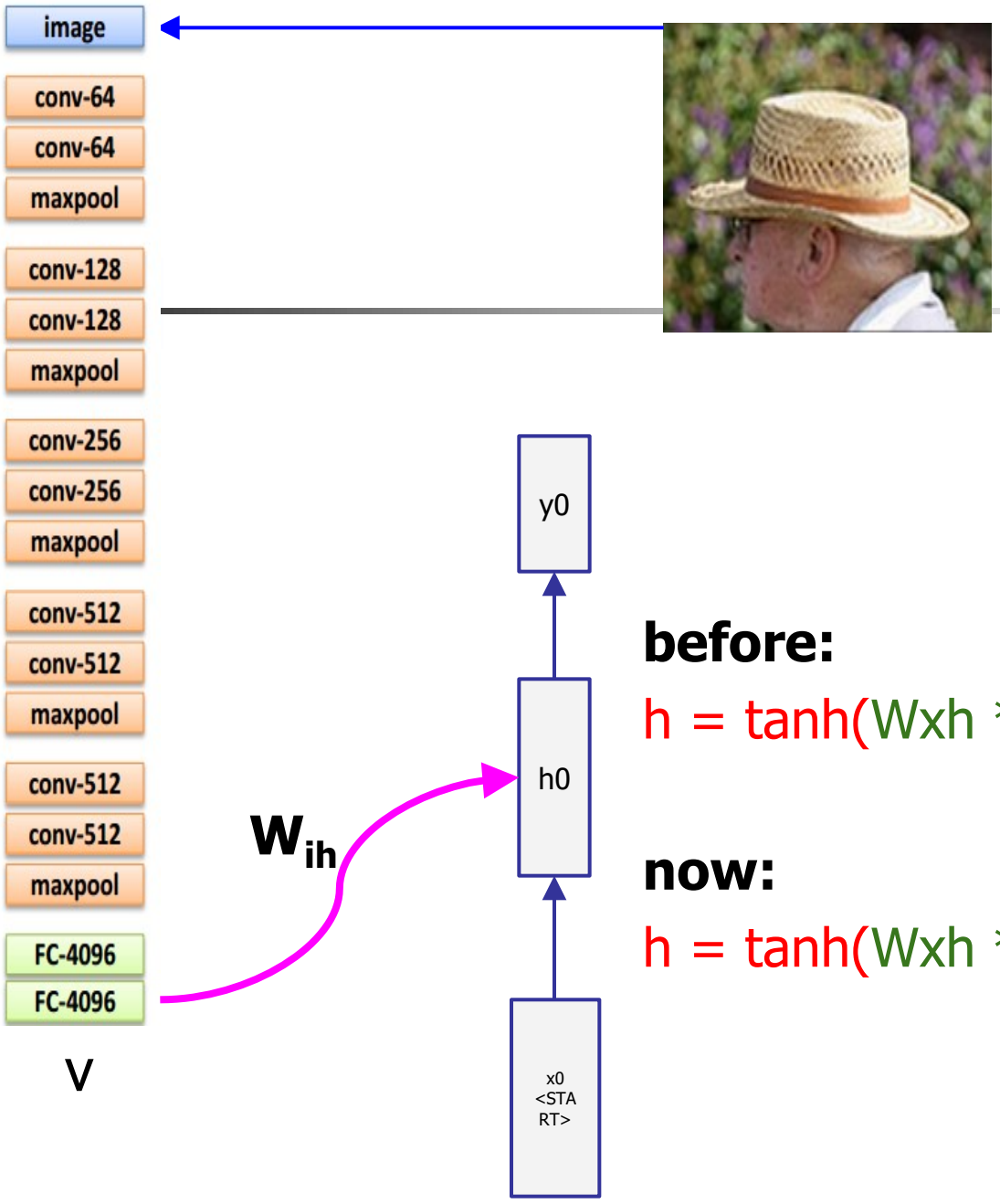


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
V
W_{ih}
y0
h0
x0
<STA
RT>
<START>
before:
h = tanh(Wxh * x + Whh * h)
now:
h = tanh(Wxh * x + Whh * h + Wih * v)

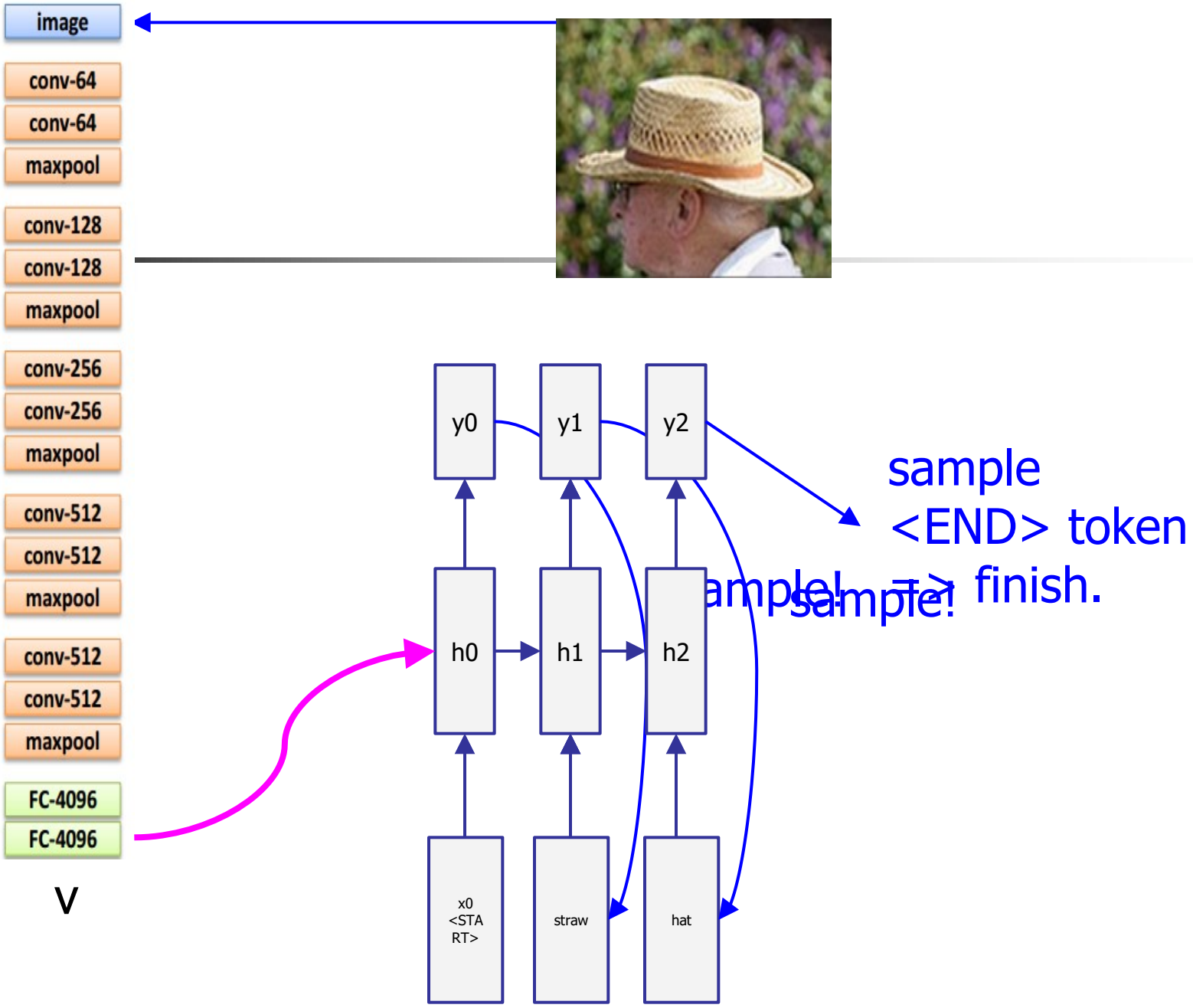


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
v
y0
y1
y2
h0
h1
h2
x0
<STA
RT>
straw
hat
<START>
sample
<END> token
=> finish.
sample!

Image Sentence Datasets

a man riding a bike on a dirt path through a forest.
bicyclist raises his fist as he rides on desert dirt trail.
this dirt bike rider is smiling and raising his fist in triumph.
a man riding a bicycle while pumping his fist in the air.
a mountain biker pumps his fist in celebration.

Microsoft COCO

[Tsung-Yi Lin et al. 2014]

currently:
~120K images
~5 sentences each

"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."

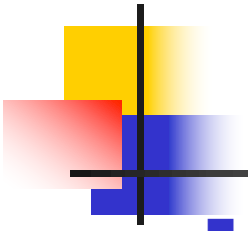
"a young boy is holding a baseball bat."

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

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

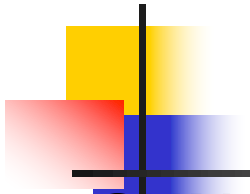
"a horse is standing in the middle of a road."

Summary



- Deep architecture works in the same principle of artificial neural network.
  - A large number of hidden layers.
  - A large number of weights.
- Convolution Neural Network (CNN)
  - Learns filter weights.
    - Sharing of weights.
  - Two types of layers
    - Convolutional and Pooling Layers
  - Two stages
    - Feature extraction
    - Classification

Summary



- CNN Variations
  - RESNET
    - Processes residual errors
  - RCNN
    - Region proposal network
    - Object localization
  - Fully connected CNN
    - Image segmentation
- Recurrent Neural Network (RNN)
  - Feedback loop.
  - Processing a sequence