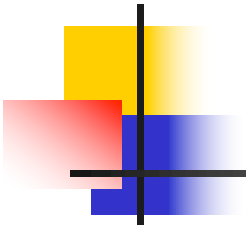
Object Recognition and Localization

Jayanta Mukhopadhyay
Dept. of Computer Science and Engg.

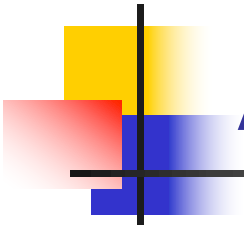
Courtesy: K. Sairam



References

Most content are adapted from:

- Deep Learning by Ian Goodfellow, Yoshua Bengio & Aaron Courville, An MIT Press Book
- CS231n: Convolutional Neural Networks for Visual Recognition by Fei Fei Li
- CS131 Computer Vision: Foundations and Applications by Juan Carlos Niebles
- Recent Research Papers



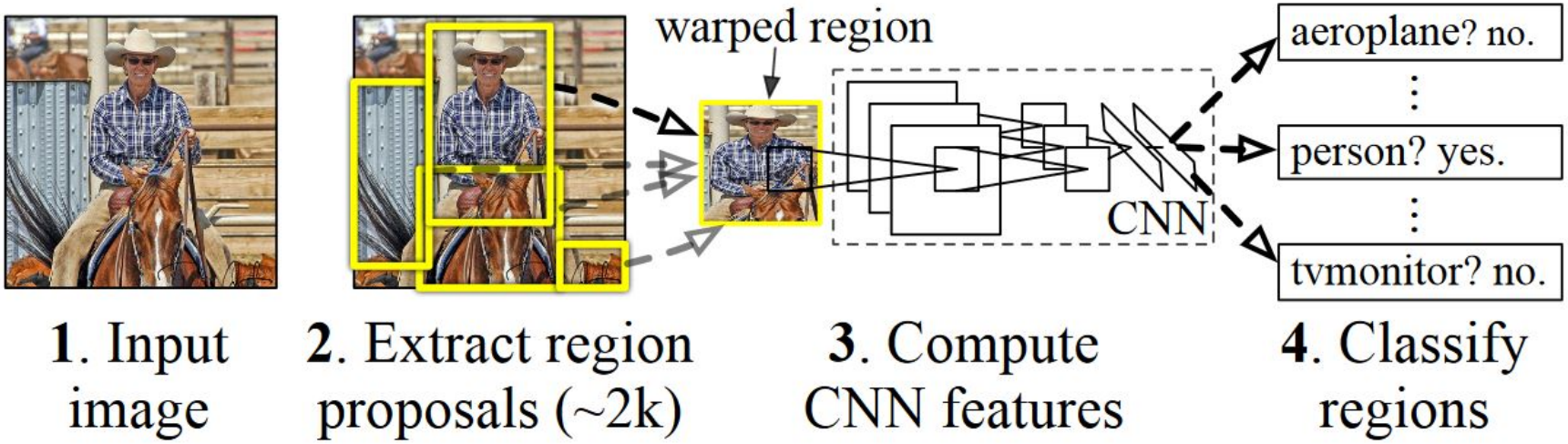
Applications:

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

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

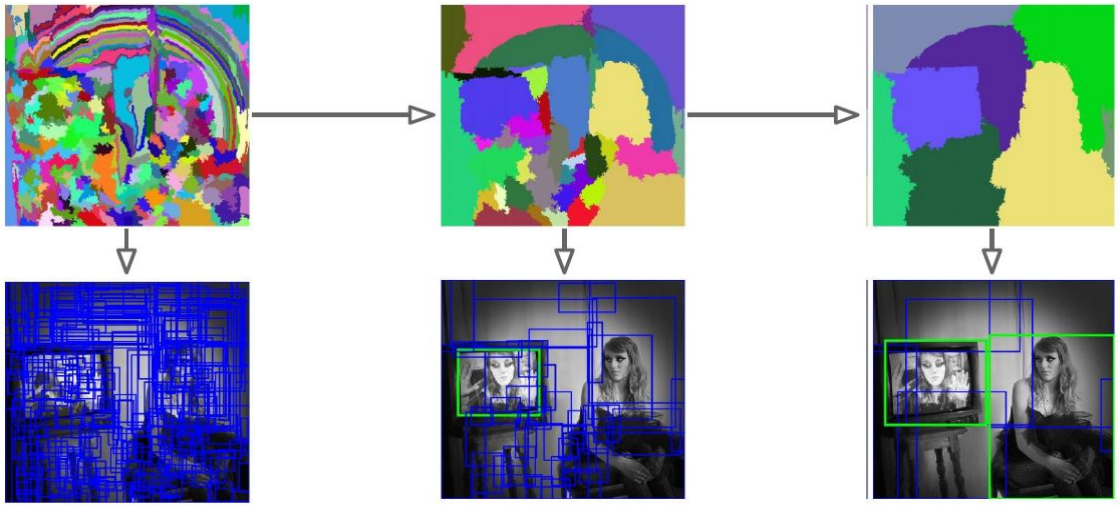


*Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

RCNN – Region Proposal

Selective Search:

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



Other Proposal techniques:
EdgeBoxes, Constrained parametric min-cuts (CPMC), Multiscale combinatorial grouping (MCG), Objectness in windows

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

RCNN Problems

- Ad hoc training objectives
- Fine-tune network with softmax classifier (log loss)
- 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 Approach Inference Time:**
  - PropTime + NumProp*ConvTime + NumProp*fcTime

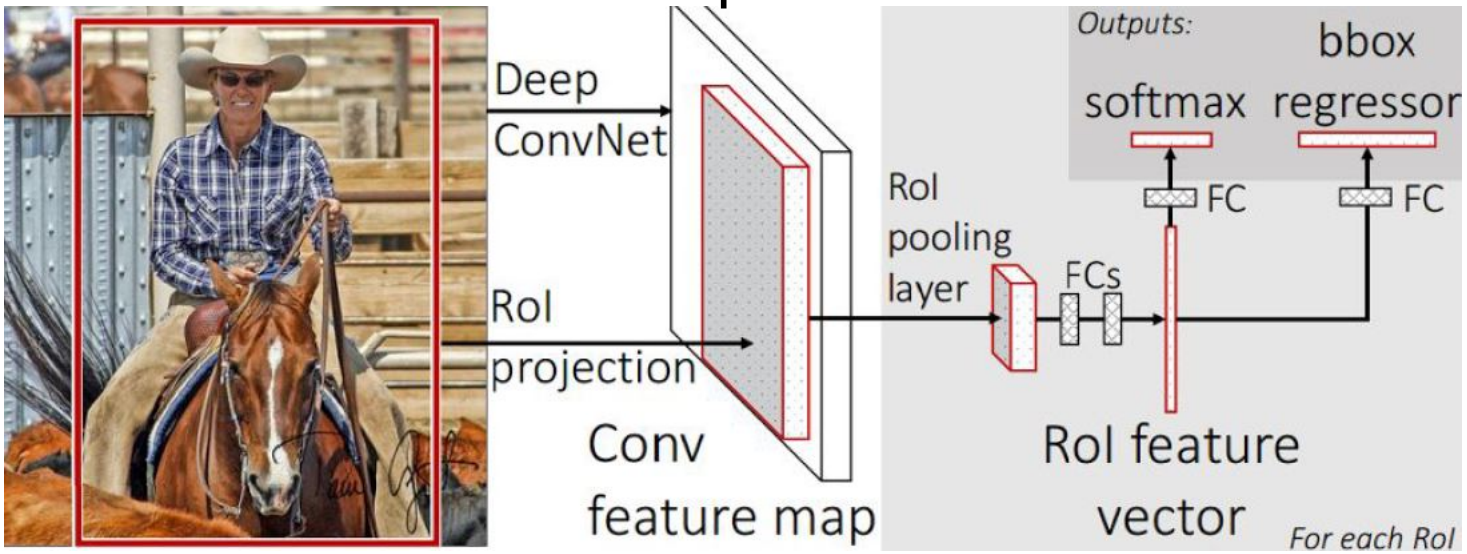
PropTime: Time taken for generating all proposals

NumProp: Number of proposals generated

fcTime : Time taken to identify the object in the image

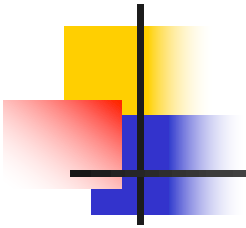
Fast RCNN

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 Approach 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

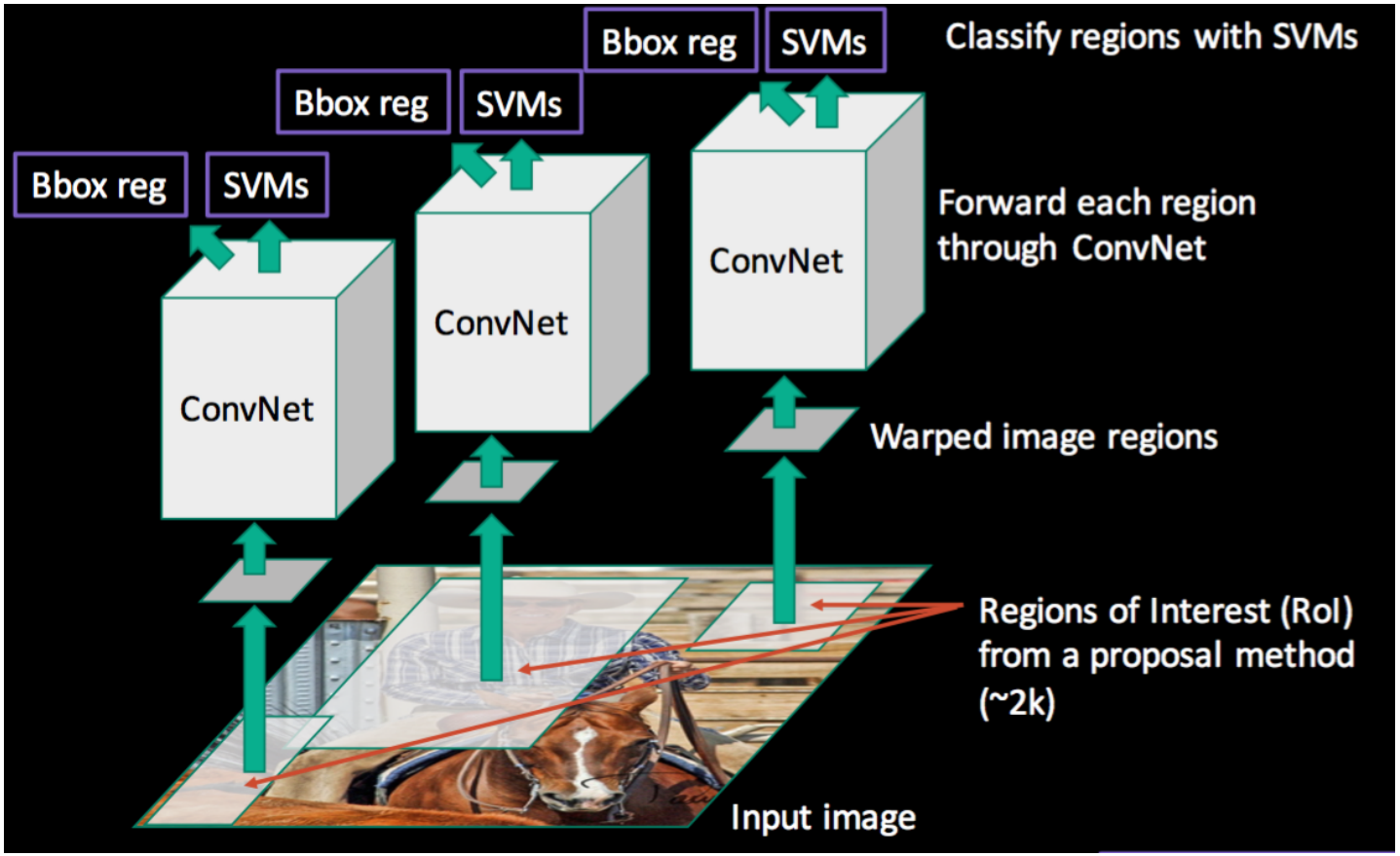
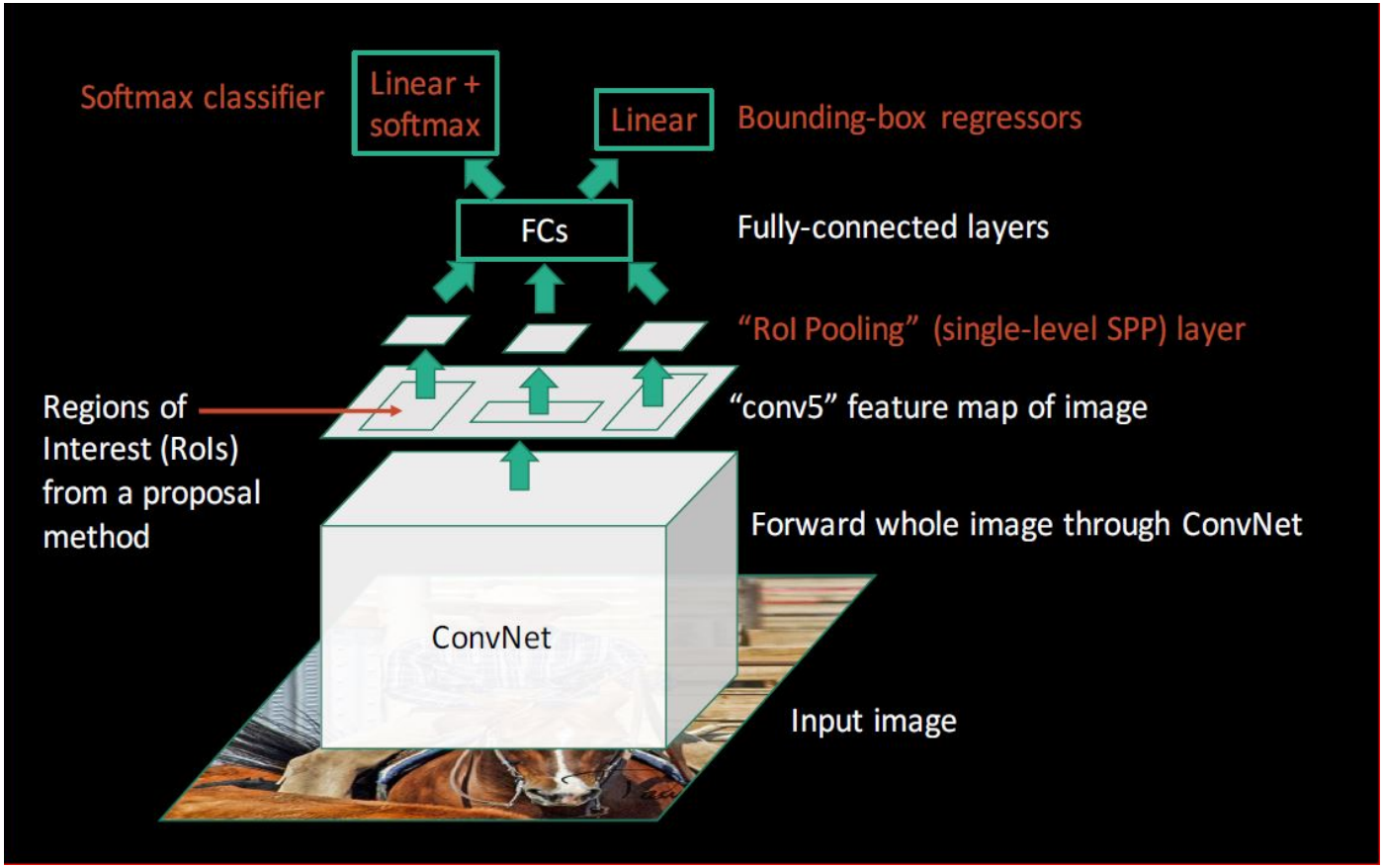


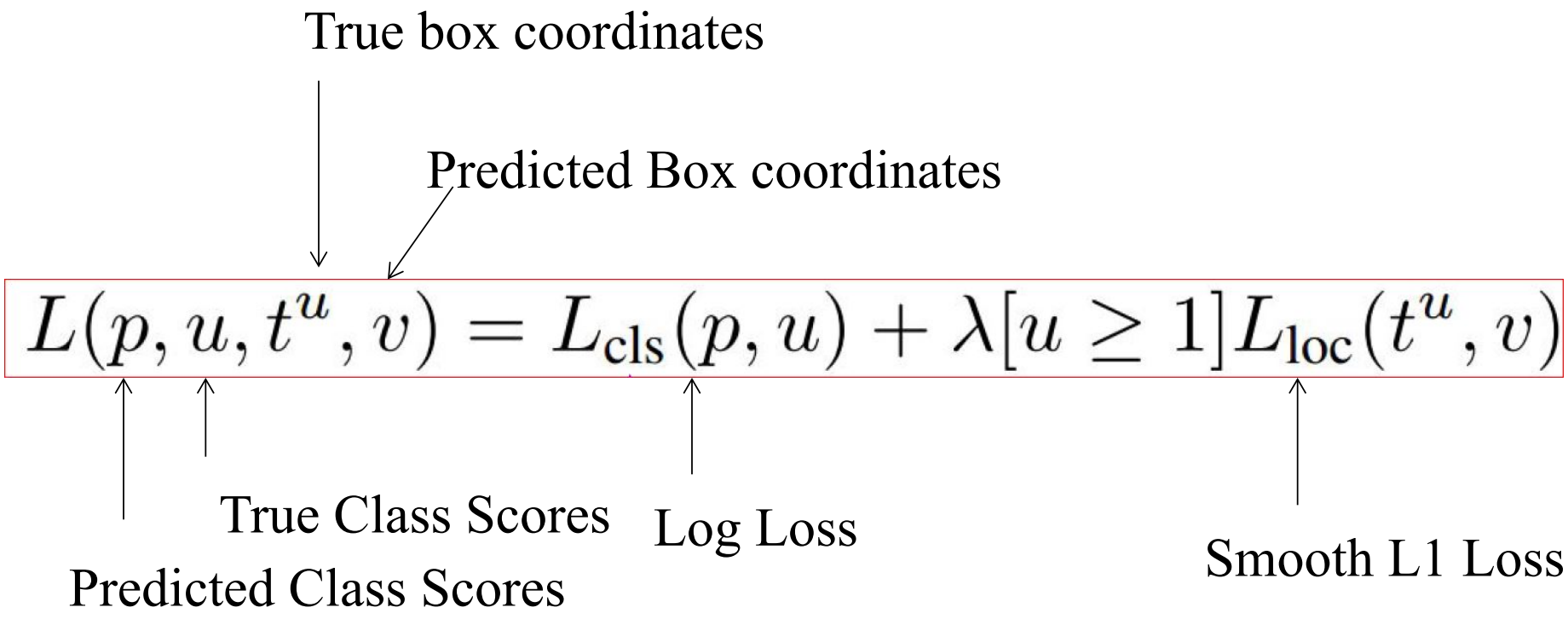
Image: Ross Girshick, 2015.

Fast RCNN



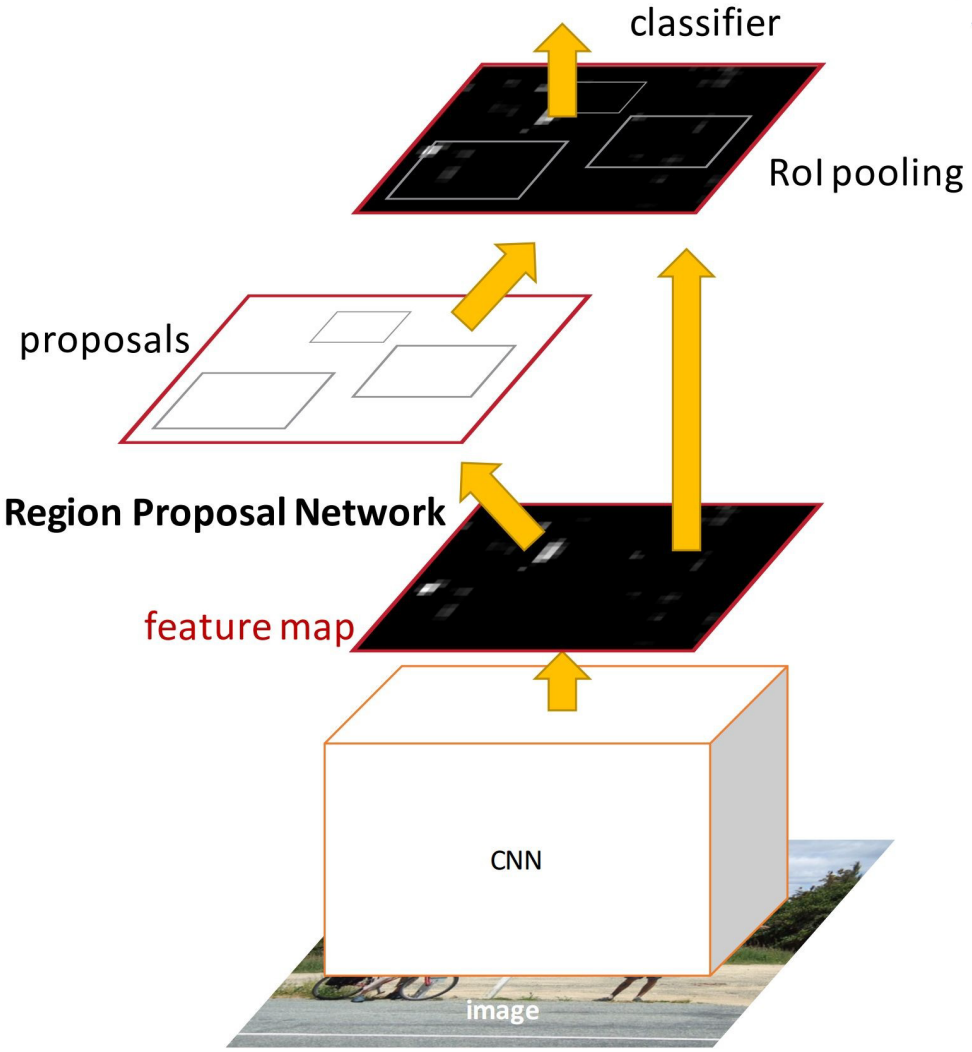
* Image: Ross Girshick, 2015.

Fast RCNN Loss



Faster RCNN

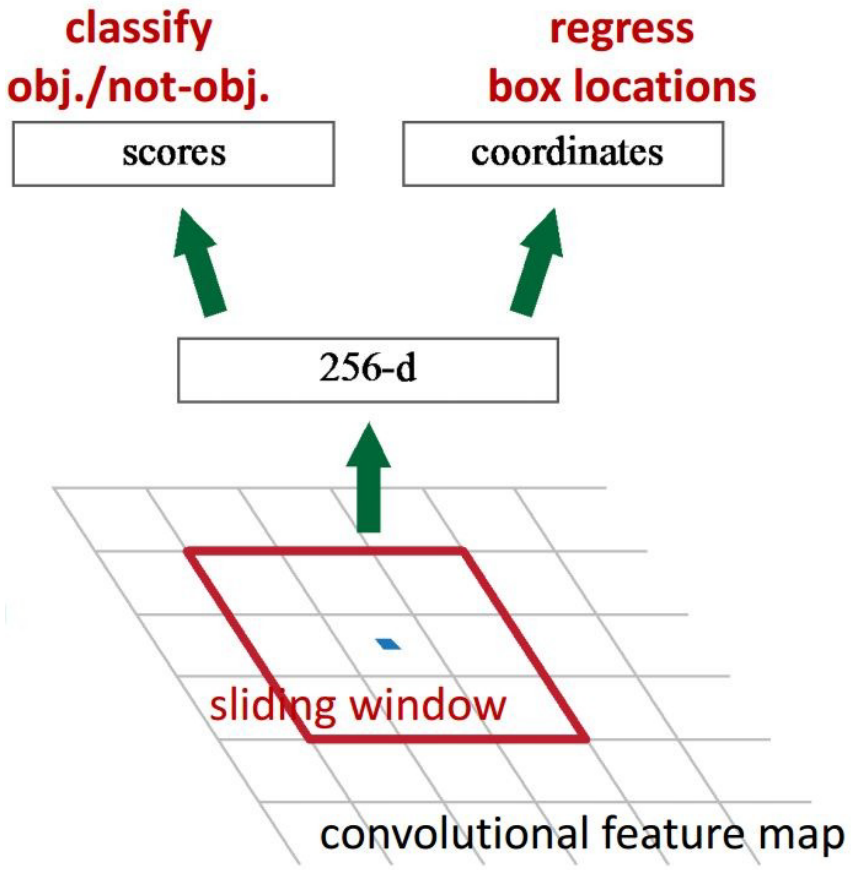
- Insert a **Region Proposal Network (RPN)** after the last convolutional layer
- RPN trained to produce region proposals directly; no need for external region proposals!
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN



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

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
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window

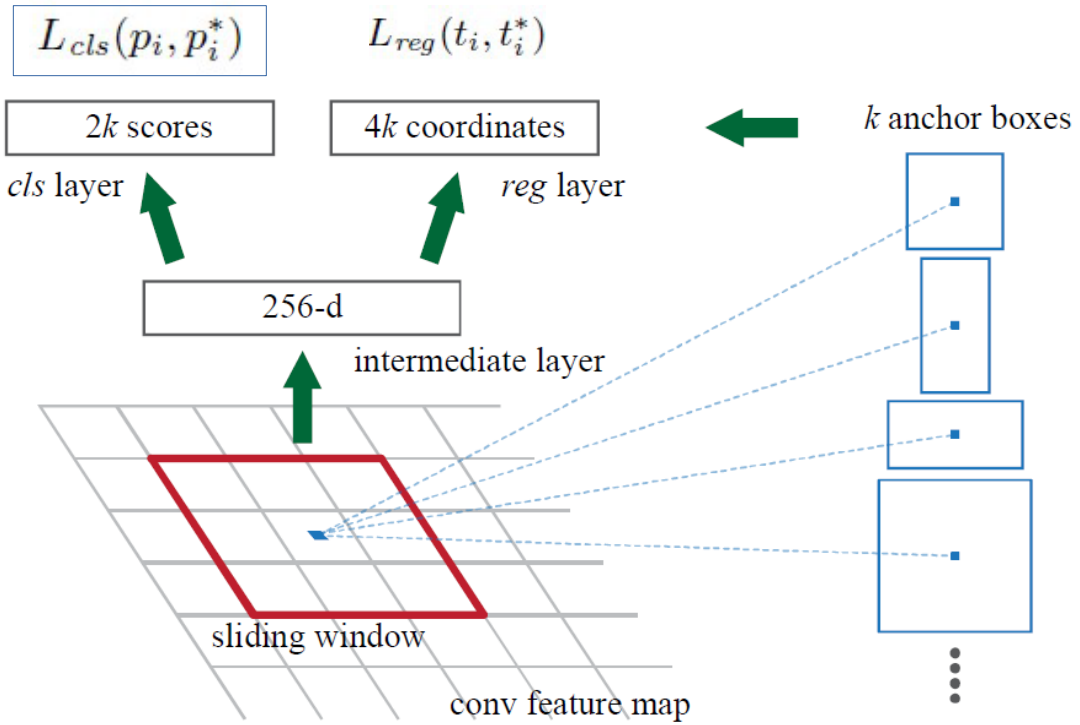


Faster RCNN Approach Time: 1*ConvTime + NumProp*fcTime

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



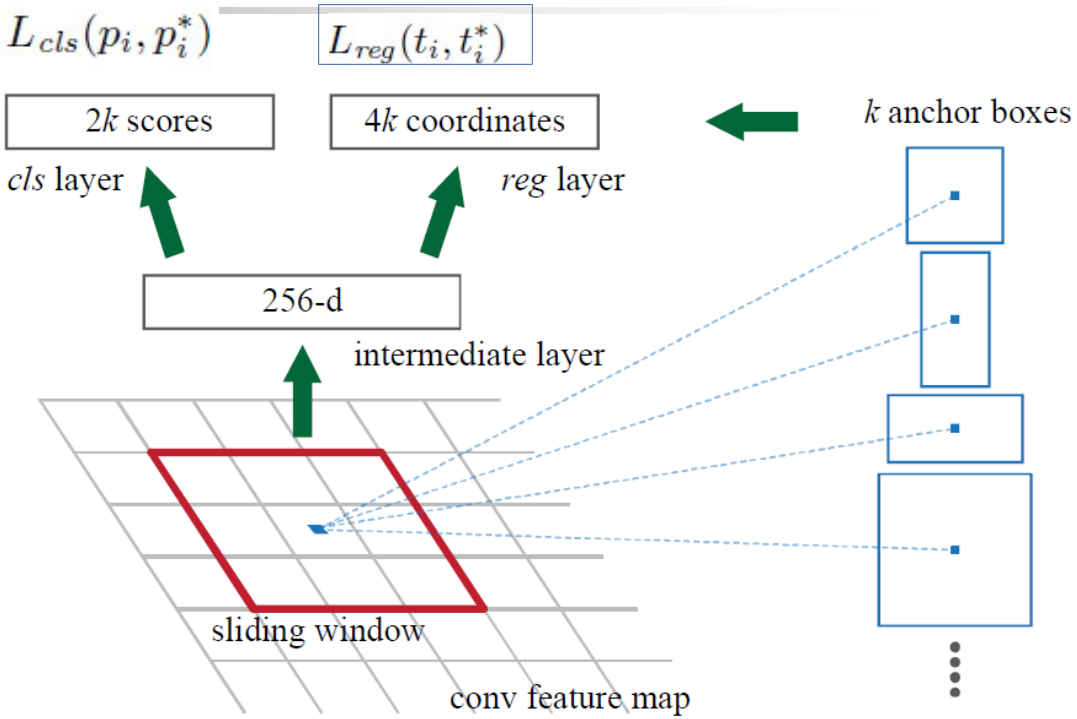
$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

RPN Loss

$$L_{loc}(t, t^*) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i, t_i^*), \qquad (2)$$

in which

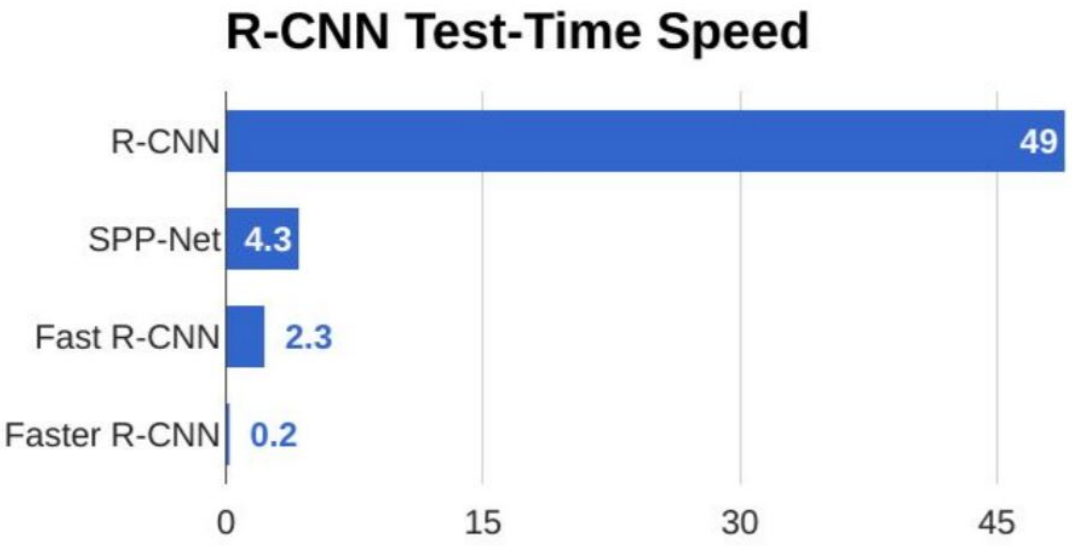
$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \qquad (3)$$

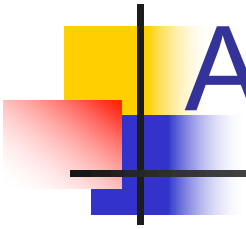


$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \quad t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \quad t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

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

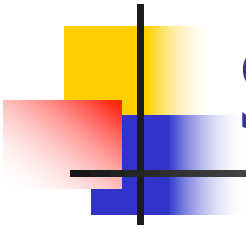
Comparison for CNN based Object Detection





Applications:

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



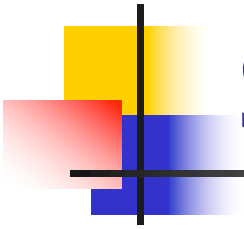
Segmentation

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

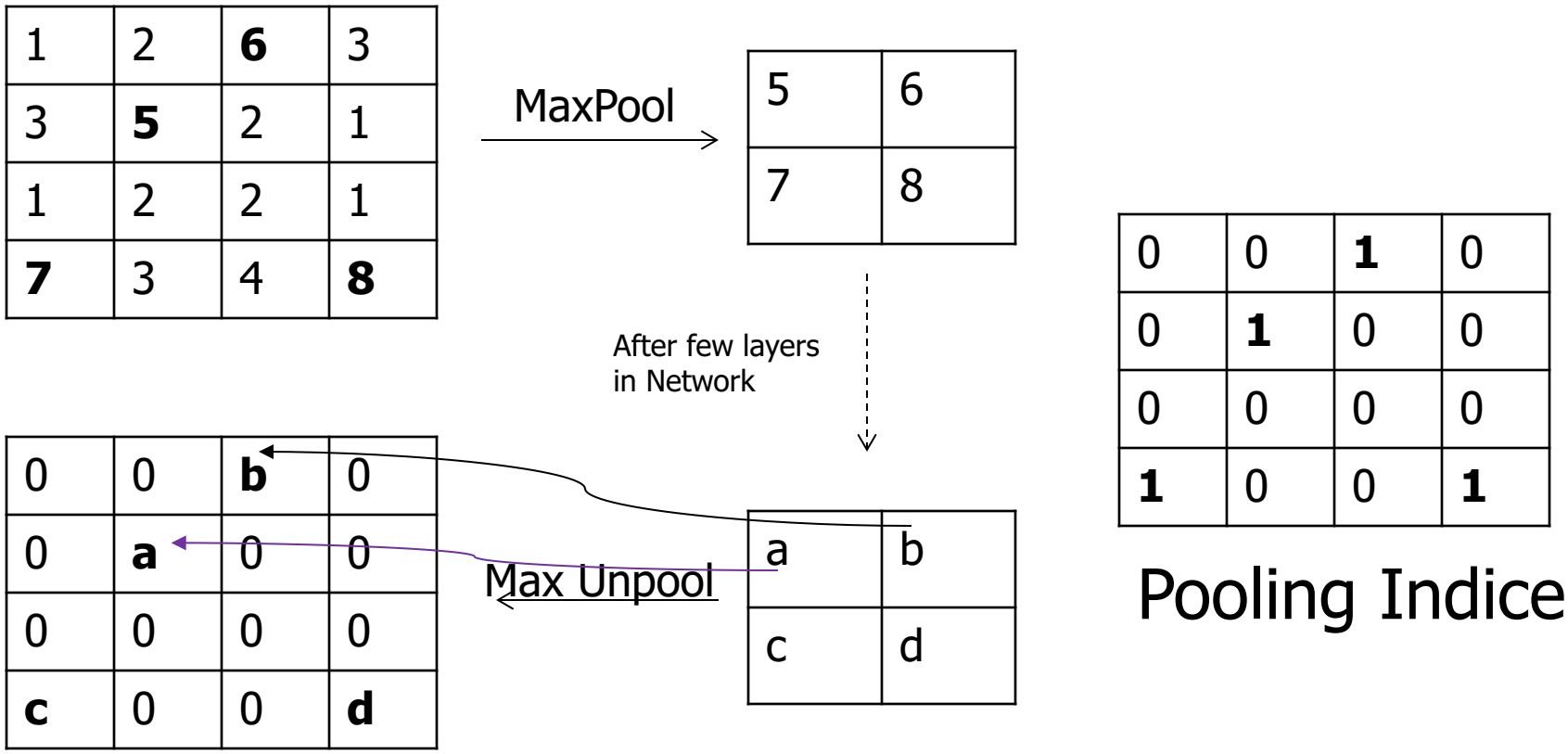


Semantic Segmentation

Building Blocks of CNNs:

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

Up- Sampling: Max Unpooling

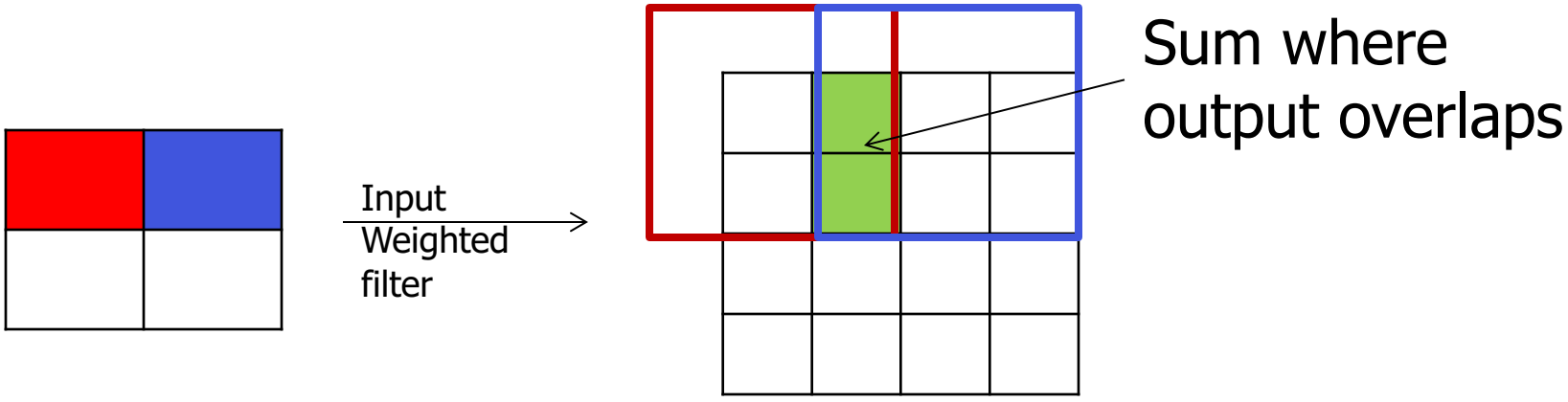


Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

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 at overlaps in the output

3 x 3 **transpose** convolution, stride 2 pad 1



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

Convolution as matrix Multiplication

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

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

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

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

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

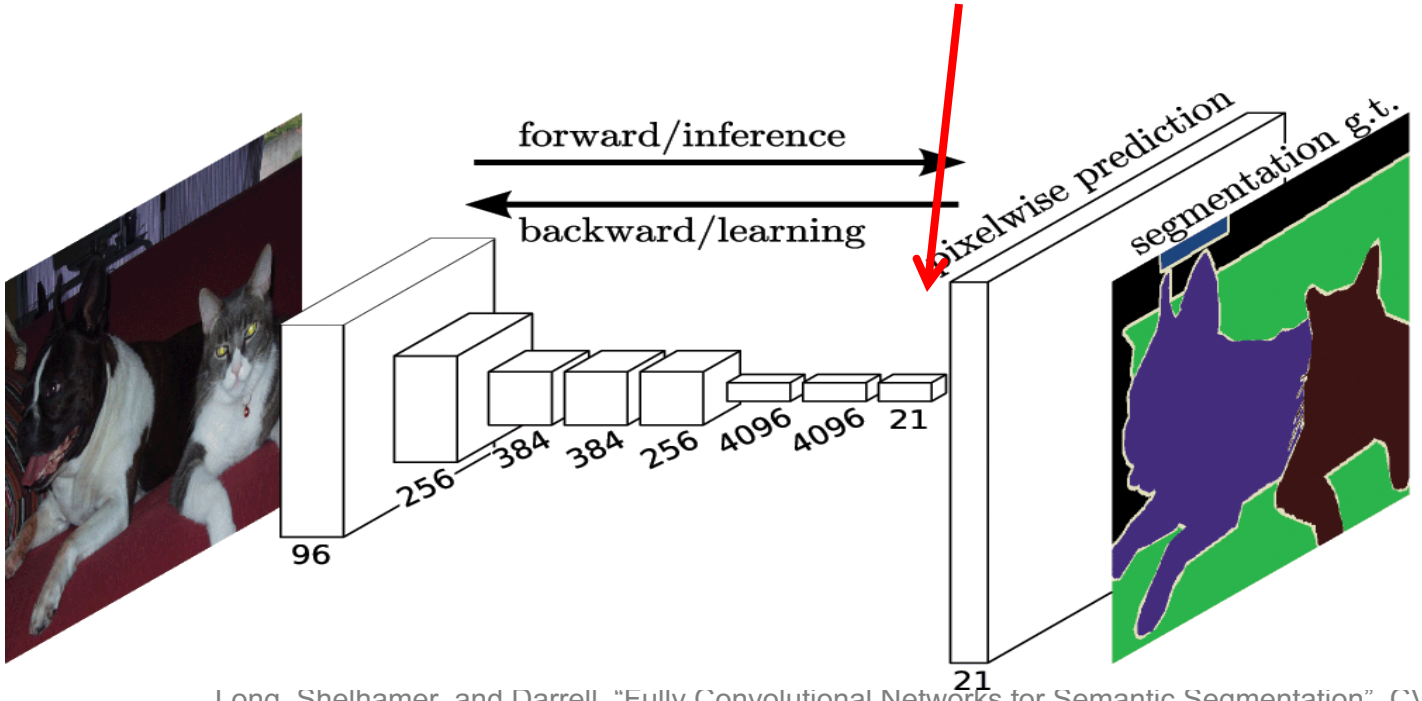
$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Transpose Convolution

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

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)

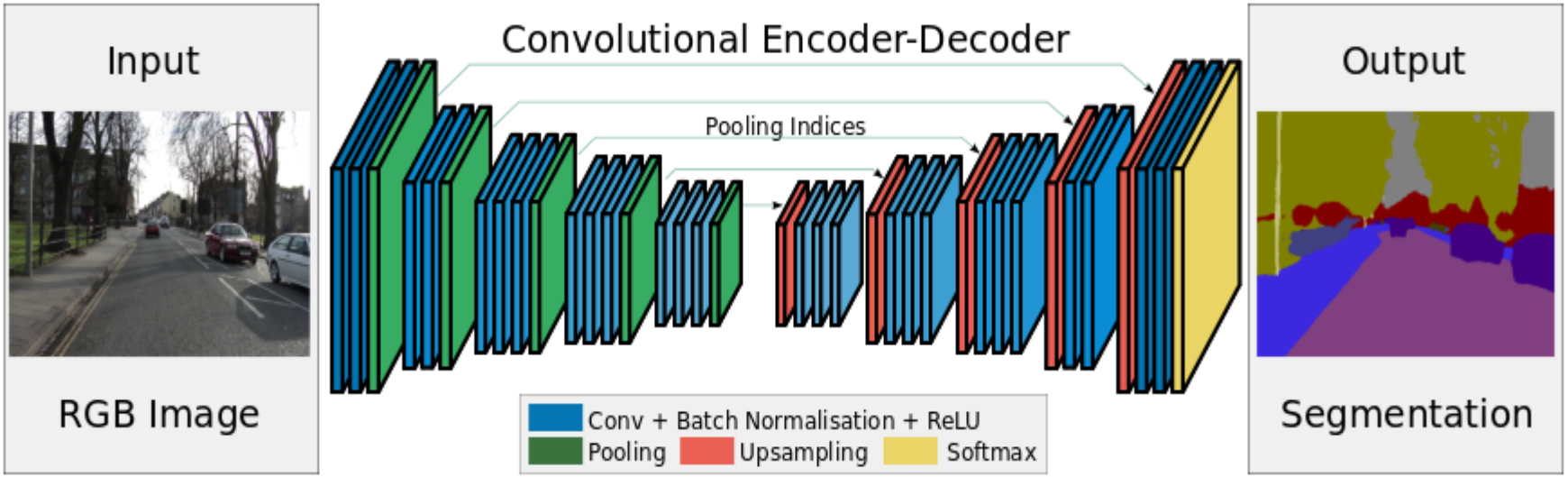
Encoder

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

Decoder

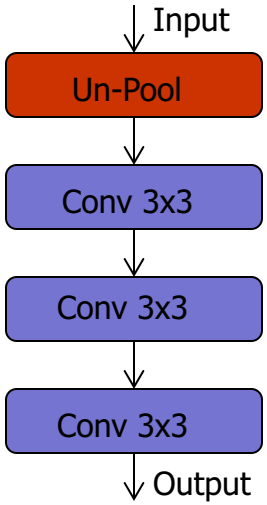
- Takes a high-dimensional feature vector and generates a semantic segmentation mask
- Decode features aggregated by encoder at multiple levels.
- Semantically project the discriminative features (lower resolution) learnt by the encoder onto the pixel space (higher resolution) to get a dense classification.

Semantic Segmentation (Encoder-Decoder)

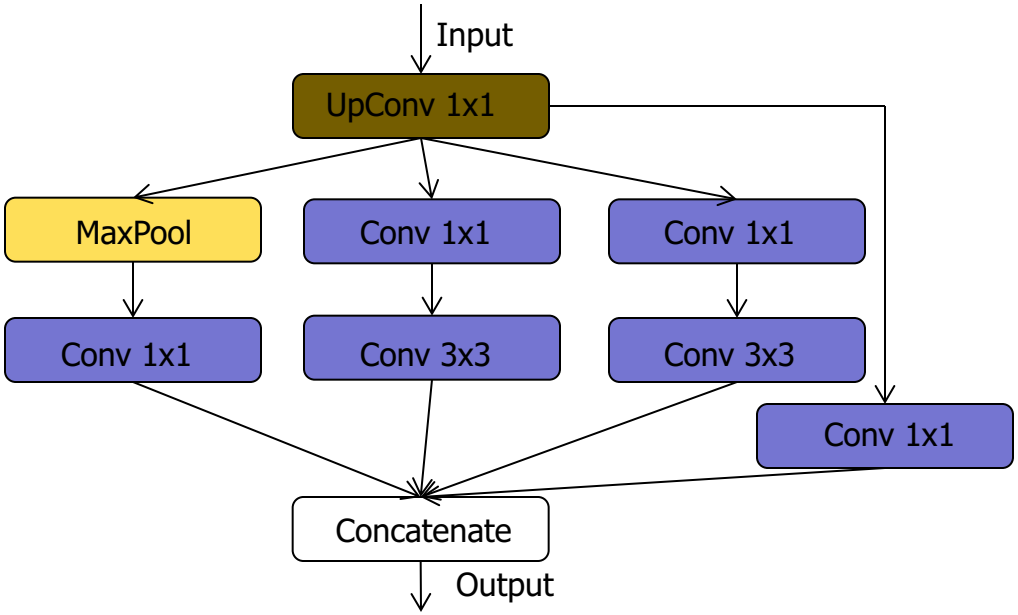


SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, Vijay Badrinarayanan, et al. PAMI, 2017

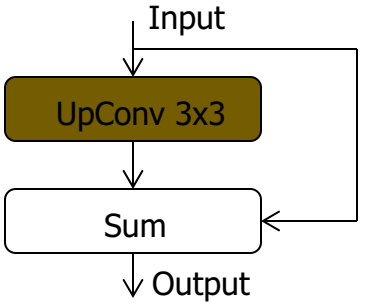
Semantic Segmentation (**Decoder**)



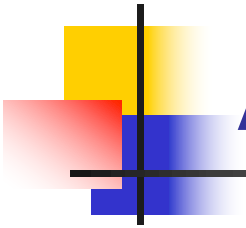
VGG



GoogleNet

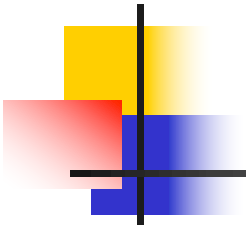


ResNet



Applications

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



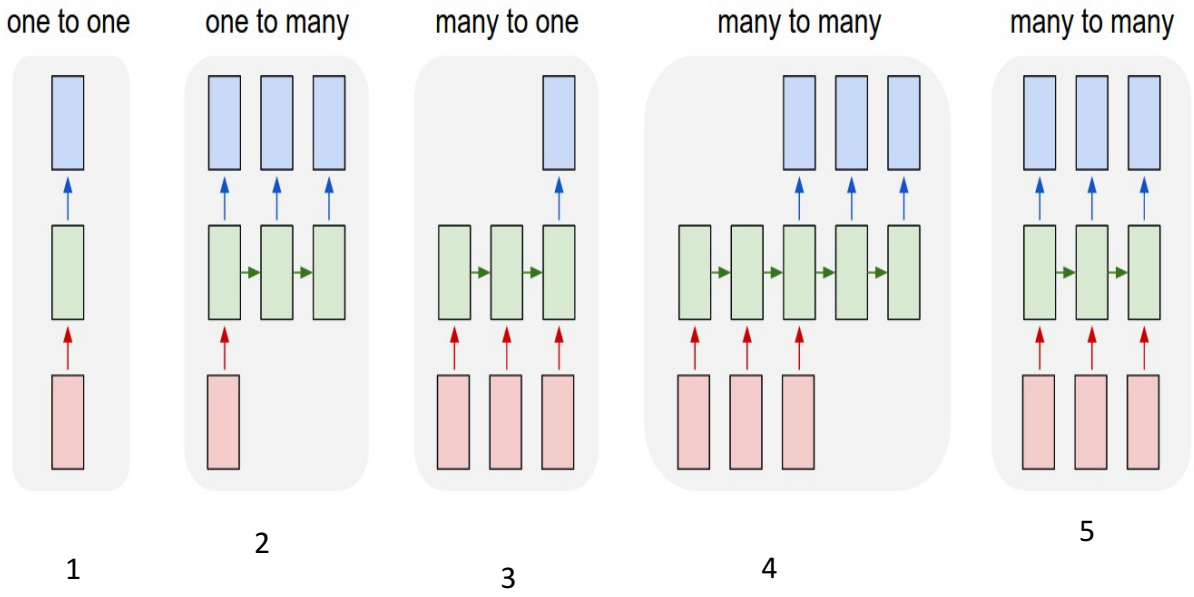
Recurrent Neural Networks

- The applications of standard Neural Networks (and also Convolutional Networks) are limited due to:
  - They only accepted a fixed-size vector as input (e.g., an image) and produce a fixed-size vector as output (e.g., probabilities of different classes).
  - These models use a fixed amount of computational steps (e.g. the number of layers in the model).
- Recurrent Neural Networks are unique as they allow us to operate over sequences of vectors.
  - Sequences in the input, the output, or in the most general case both

Recurrent Neural Networks

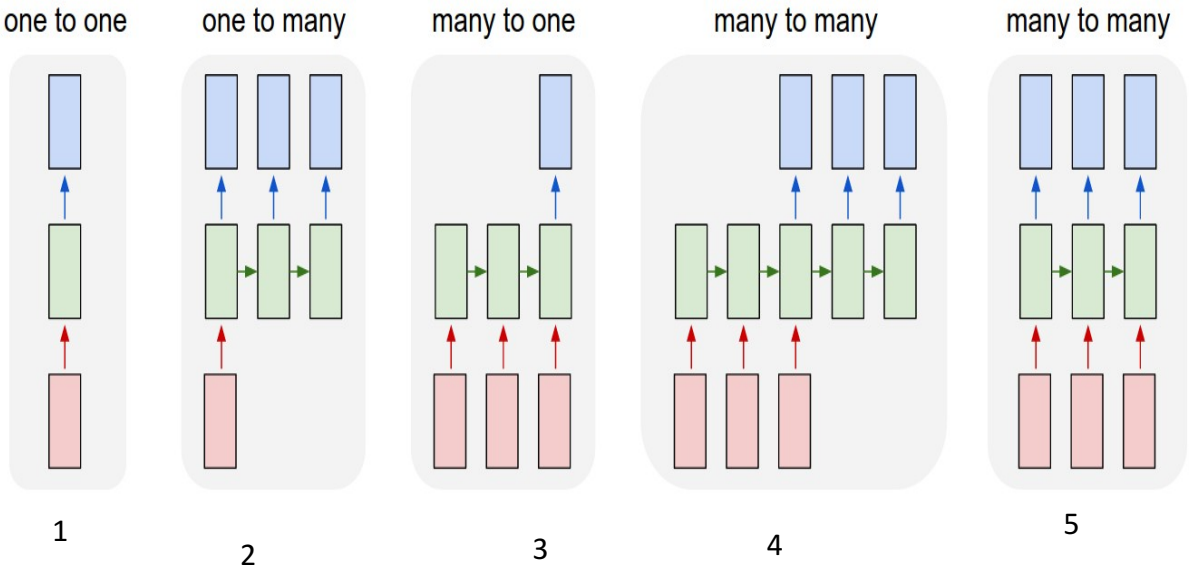
- Intuition of Recurrent Neural Networks
  - Human thoughts have persistence; humans don't start their thinking from scratch every second.
  - As you read this sentence, you understand each word based on your understanding of previous words.
- One of the appeals of RNNs is the idea that they are able to connect previous information to the present task
  - 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.

Examples of Recurrent Neural Networks



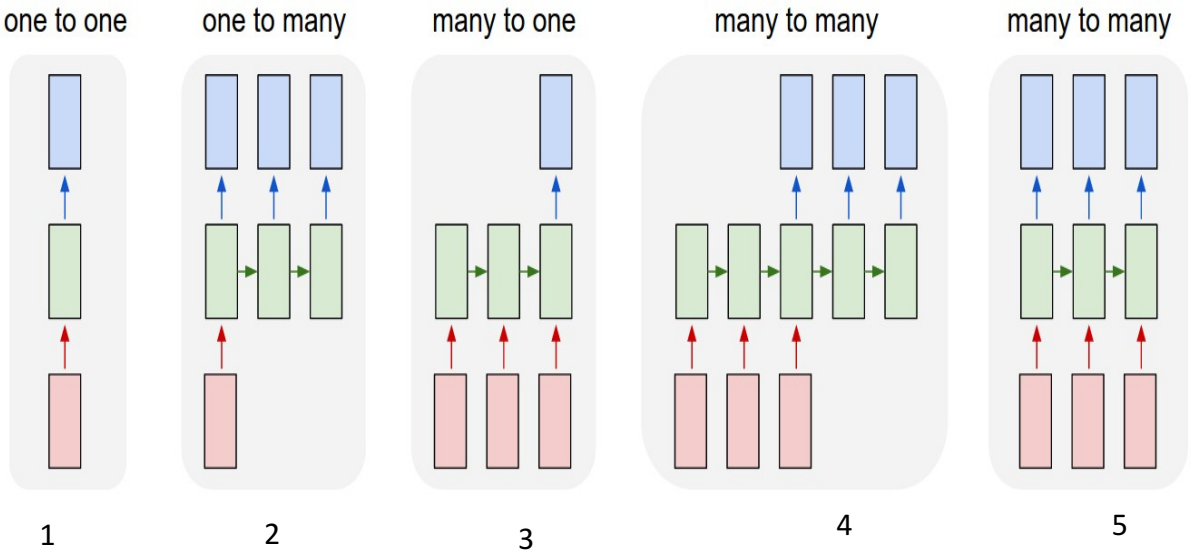
- Each rectangle is a vector and arrows represent functions (e.g. matrix multiply).
- Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state

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).

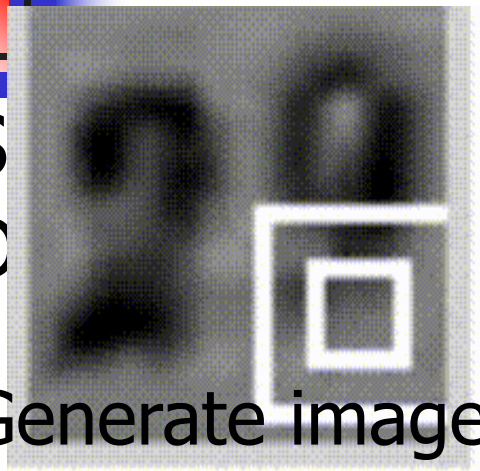
Examples of Recurrent Neural Networks



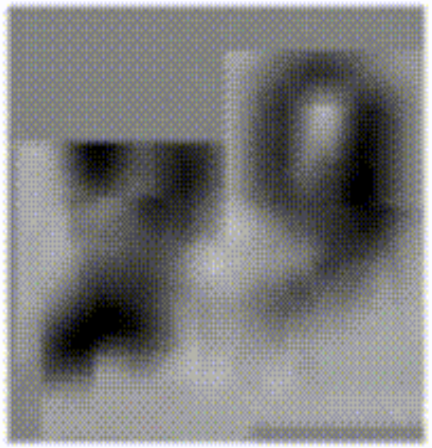
3. Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).
4. Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French).
5. Synced sequence input and output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

Recurrent Neural Networks

S ocessin
D

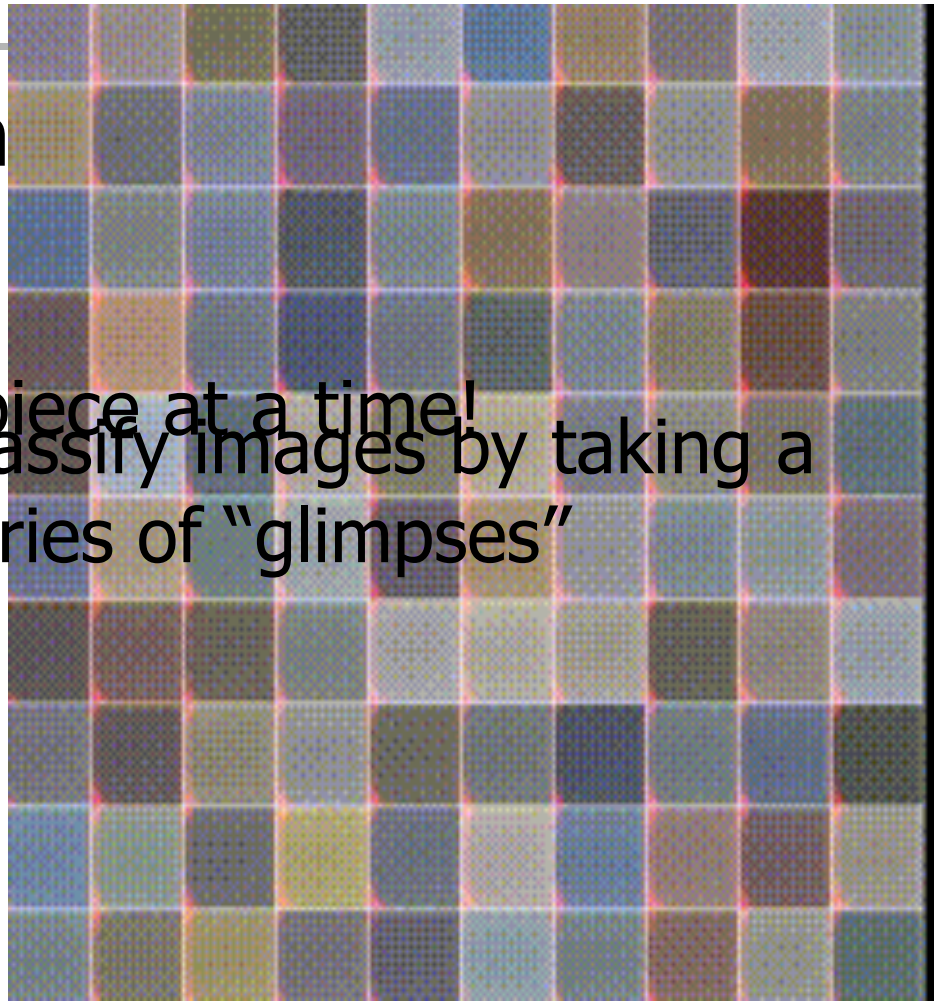


Generate images one piece at a time!



*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.

Recurrent Neural Networks

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

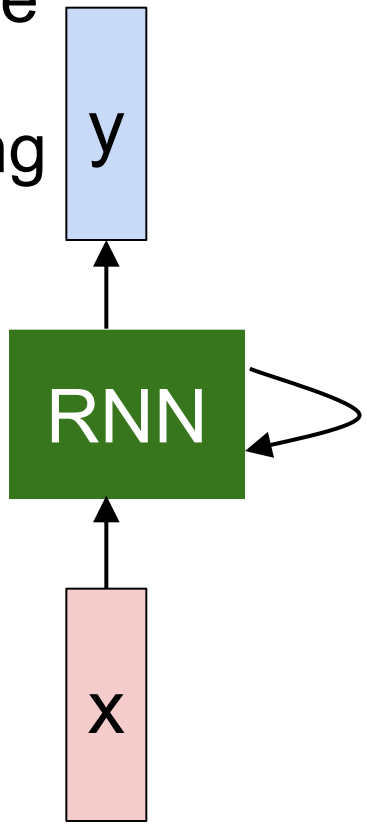
$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

input vector at some time step

some function with parameters W



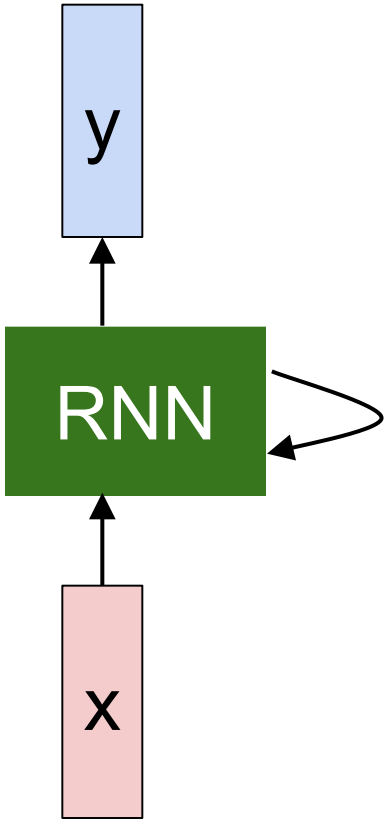
The same function and the same set of parameters are used at every time step.

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 Networks

Vocabulary: [h,e,l,o]

Example training sequence:
“hello”

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

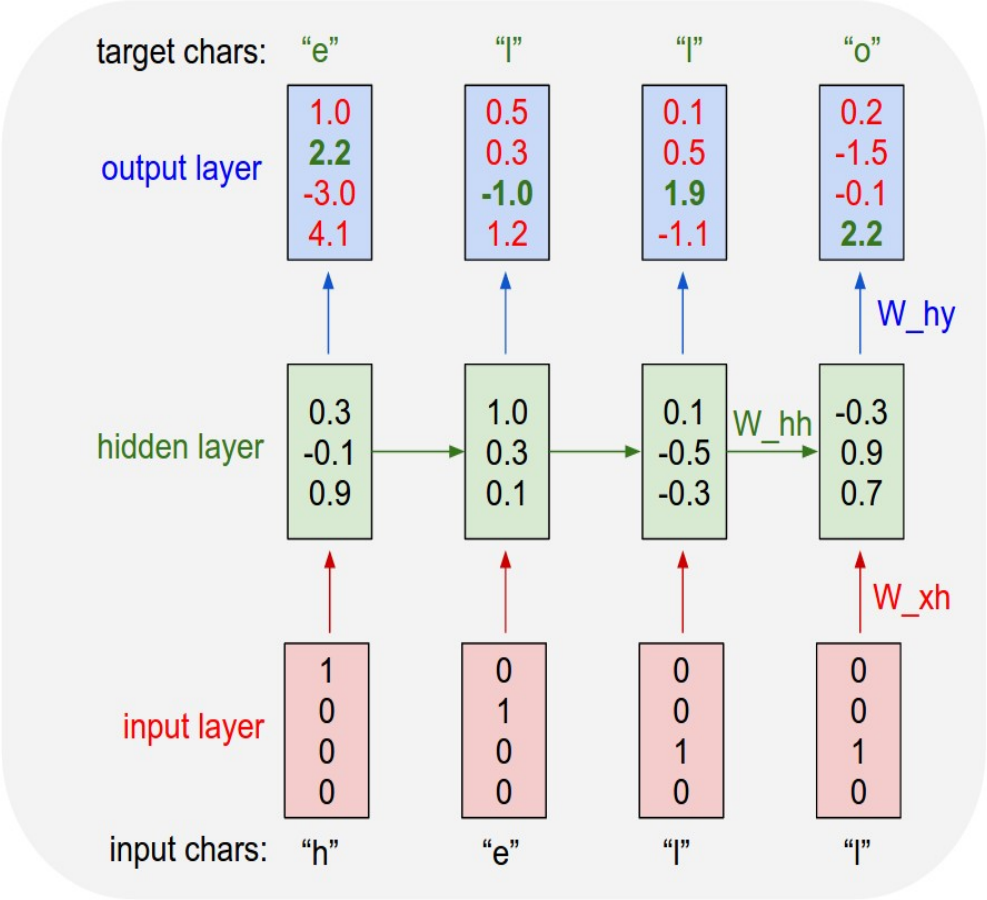
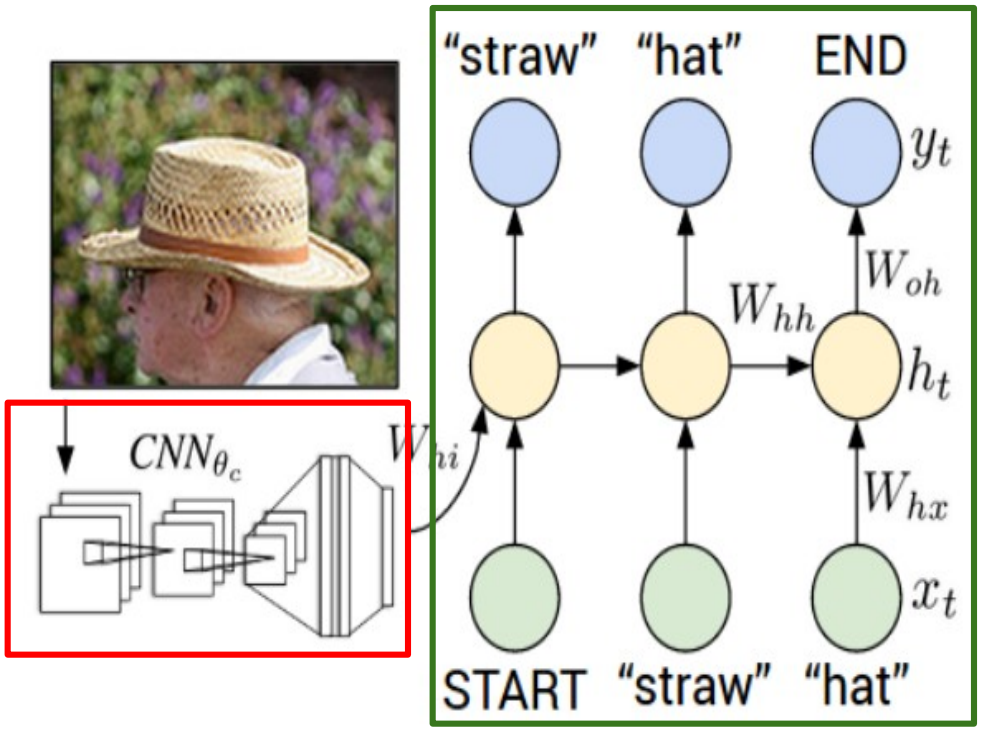


Image Captioning



* 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

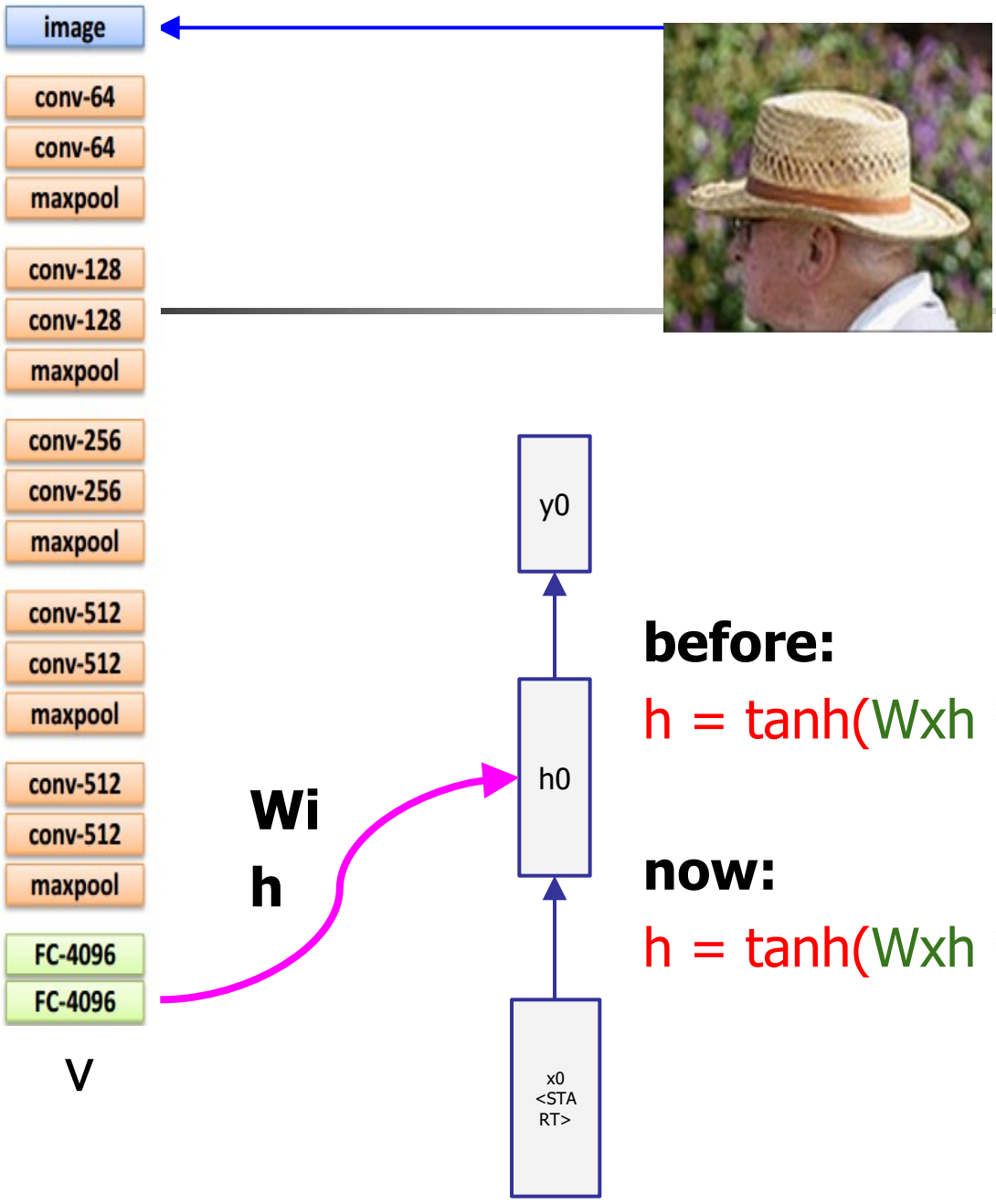


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
V
Wi
h
y0
h0
x0
<STA
RT>
<START>
before:
h = tanh(Wxh * x + Whh * h)
now:
h = tanh(Wxh * x + Whh * h + Wih * v)

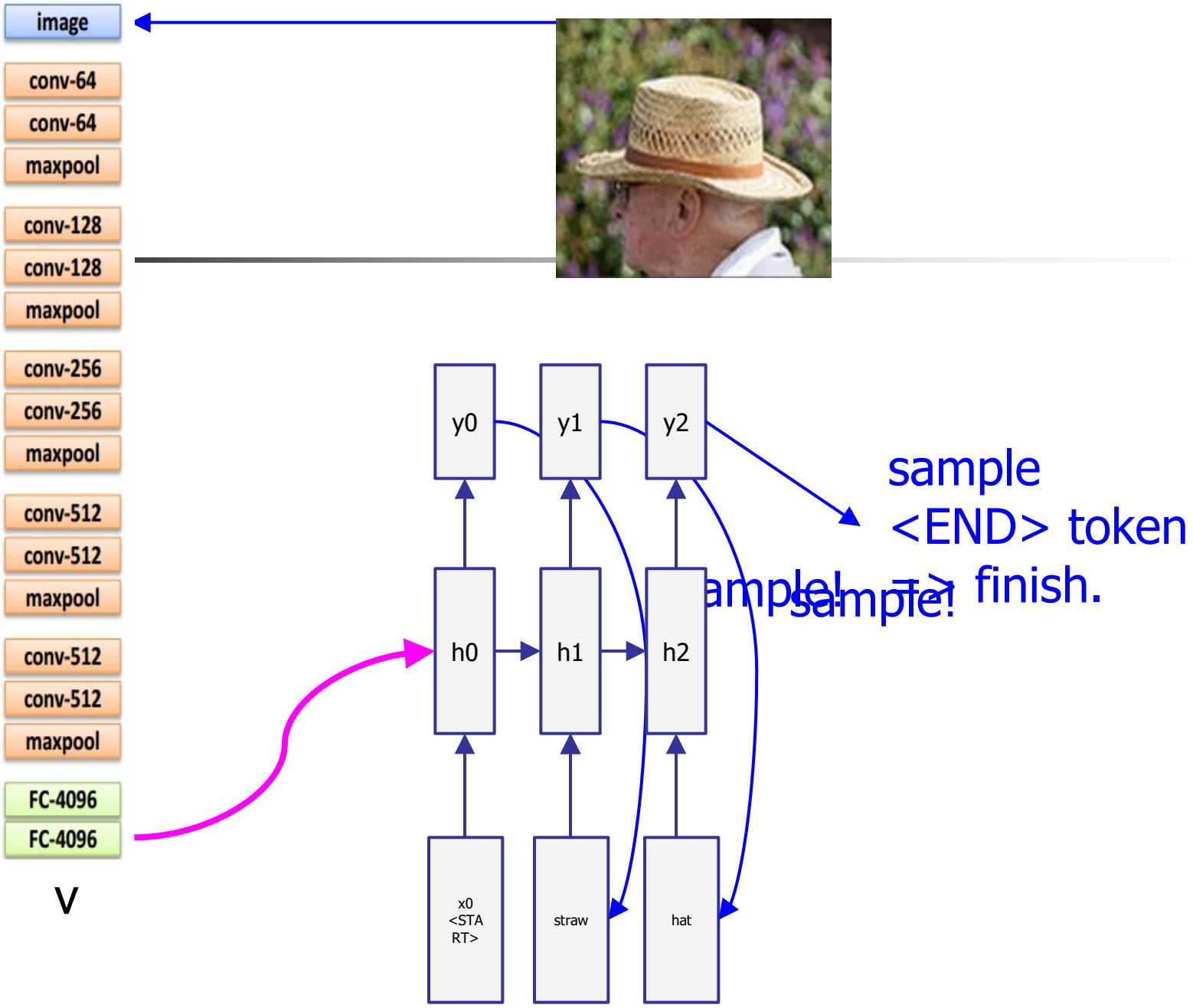


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
v
y0
y1
y2
h0
h1
h2
x0
<STA
RT>
straw
hat
<START>
sample
<END> token
=> finish.
sample!

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."


Thank you!