Lecture Notes for

Neural Networks and Machine Learning

Fully Convolutional Learning II: Object Detection

Logistics and Agenda

- Logistics
  - Lab grading update
- Agenda
  - Lab Town Hall Review and Final Project (if needed)
  - Upsampling
  - Full Convolutional Architectures
    - Semantic Segmentation Basics (last time)
    - Object Detection (this time):
      - RCNN, YOLO
    - Instance Segmentation (next time, probably):
      - Mask-RCNN, YOLACT

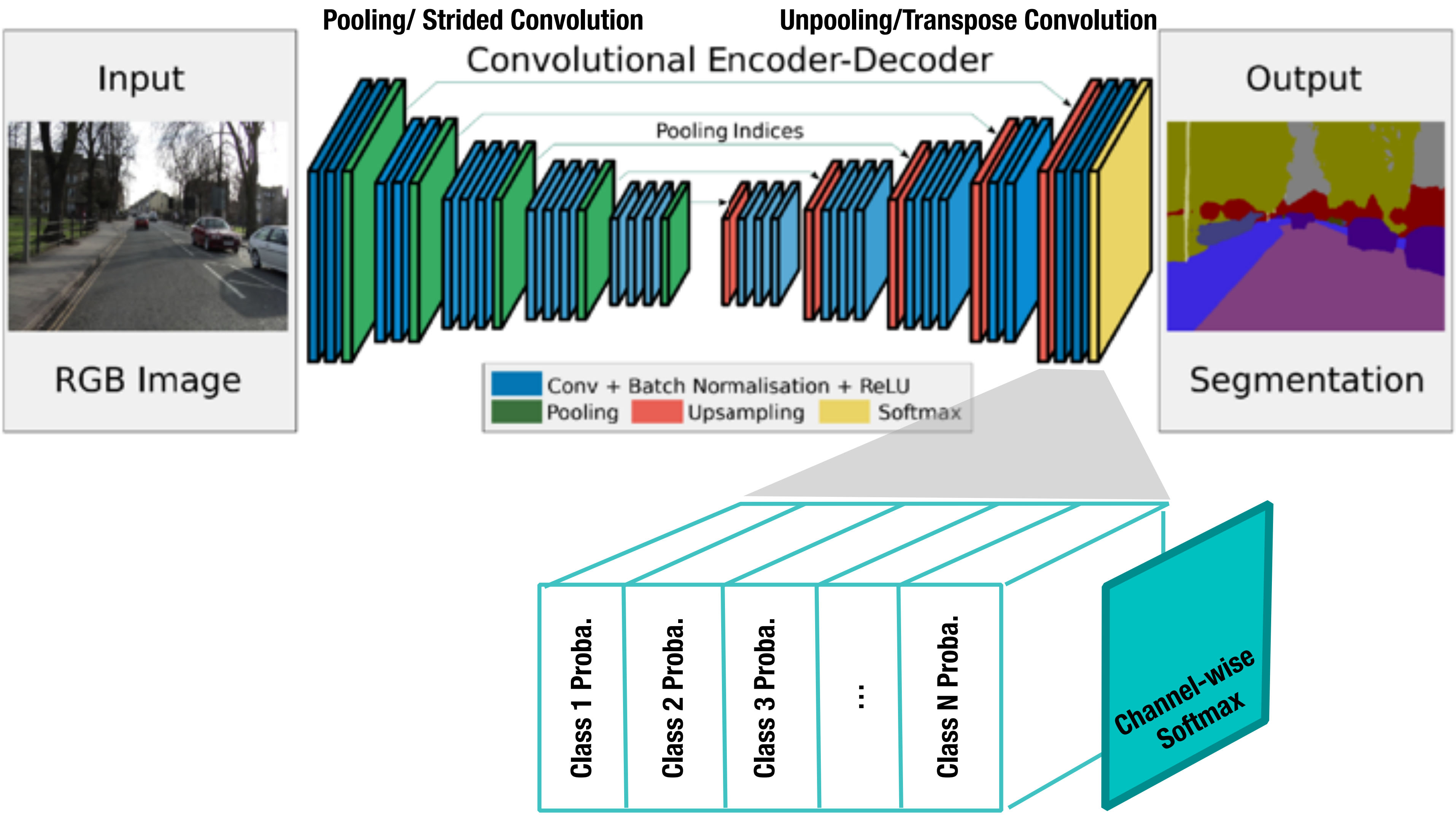
Town Hall Revisited

Final Project

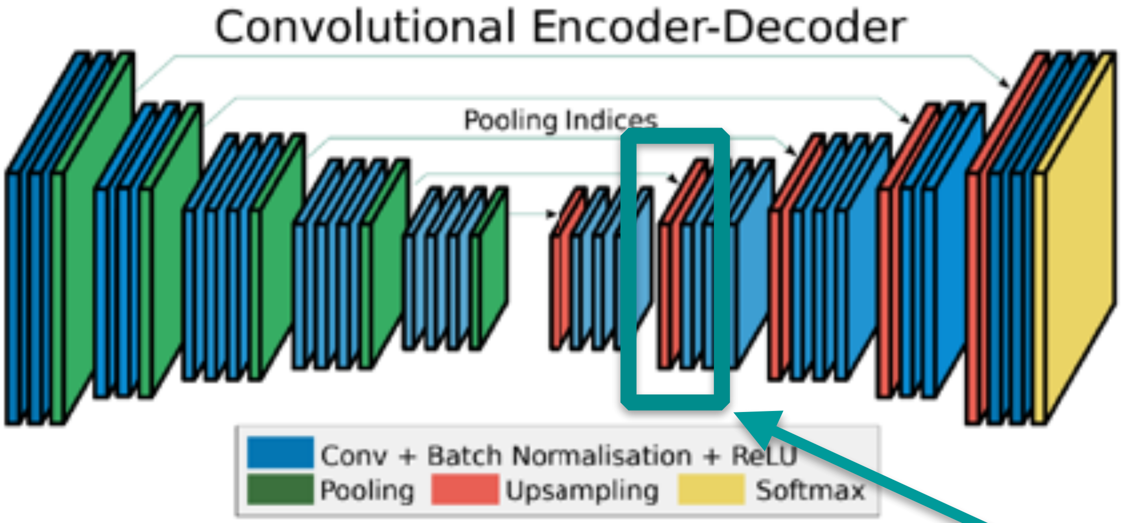
- I have biometric data for pilots…
- Perhaps something to use for the final project…

Basics: Upsampling Layers

Last Time


Pooling/ Strided Convolution
Unpooling/Transpose Convolution
Input
RGB Image
Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax
Output
Segmentation
Class 1 Proba.
Class 2 Proba.
Class 3 Proba.
...
Class N Proba.
Channel-wise Softmax

Decoder Network



Some researcher started calling this **deconvolution**.

If you use that term in this class, **you fail**.

This is upsampling and then convolution, but **now the interpolation filters are learned**!!

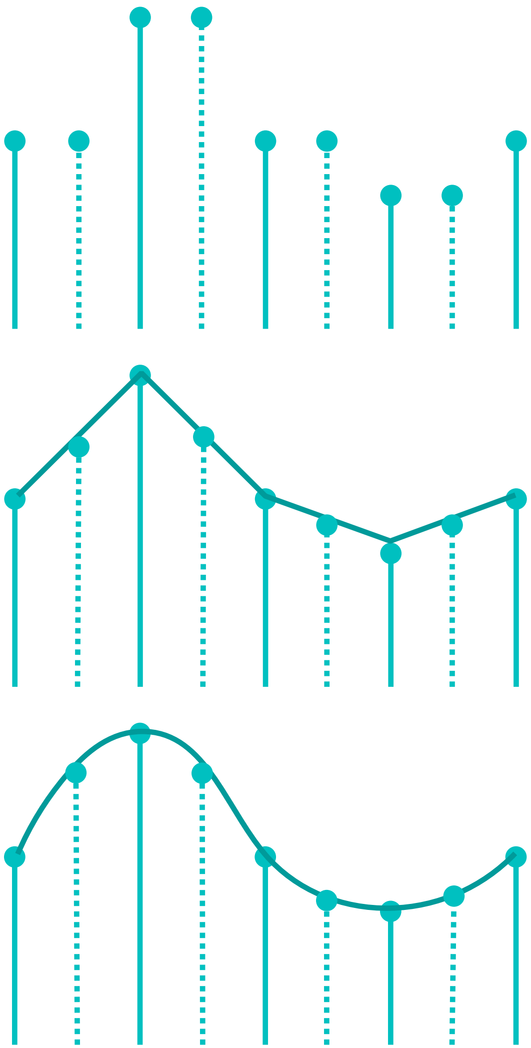
https://tjmachinelearning.com/lectures/1718/fcn/

Integer Upsampling via Interpolation

Nearest Neighbor

Linear

Cubic



All are equivalent to inserting zeros and applying convolutional filter

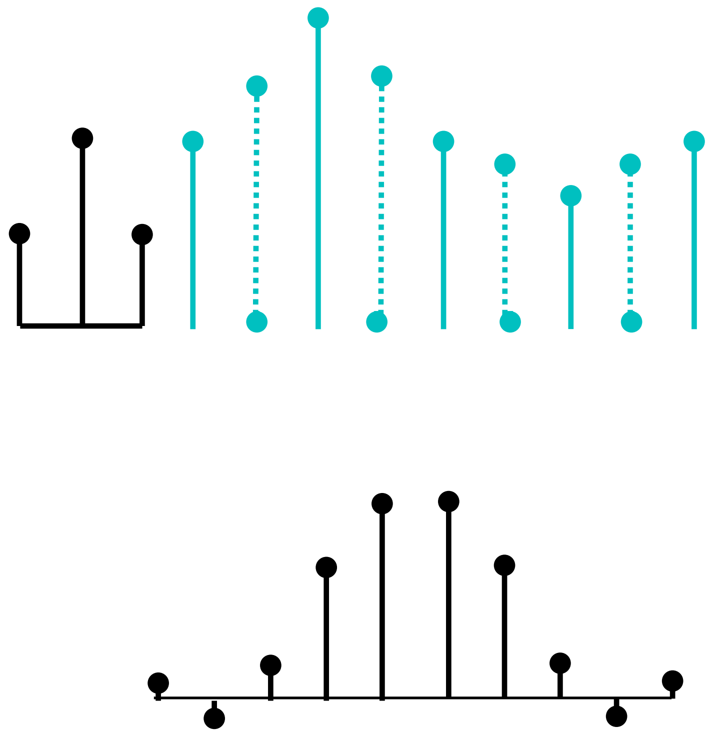


Image Upsampling, Integer Factor

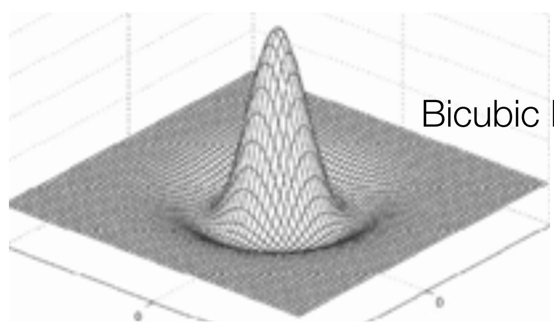
- Insert Zeros
- Convolve

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1		2		3		4	
5		6		7		8	
9		10		11		12	
13		14		15		16	

0.25	0.5	0.25
0.5	1	0.5
0.25	0.5	0.25

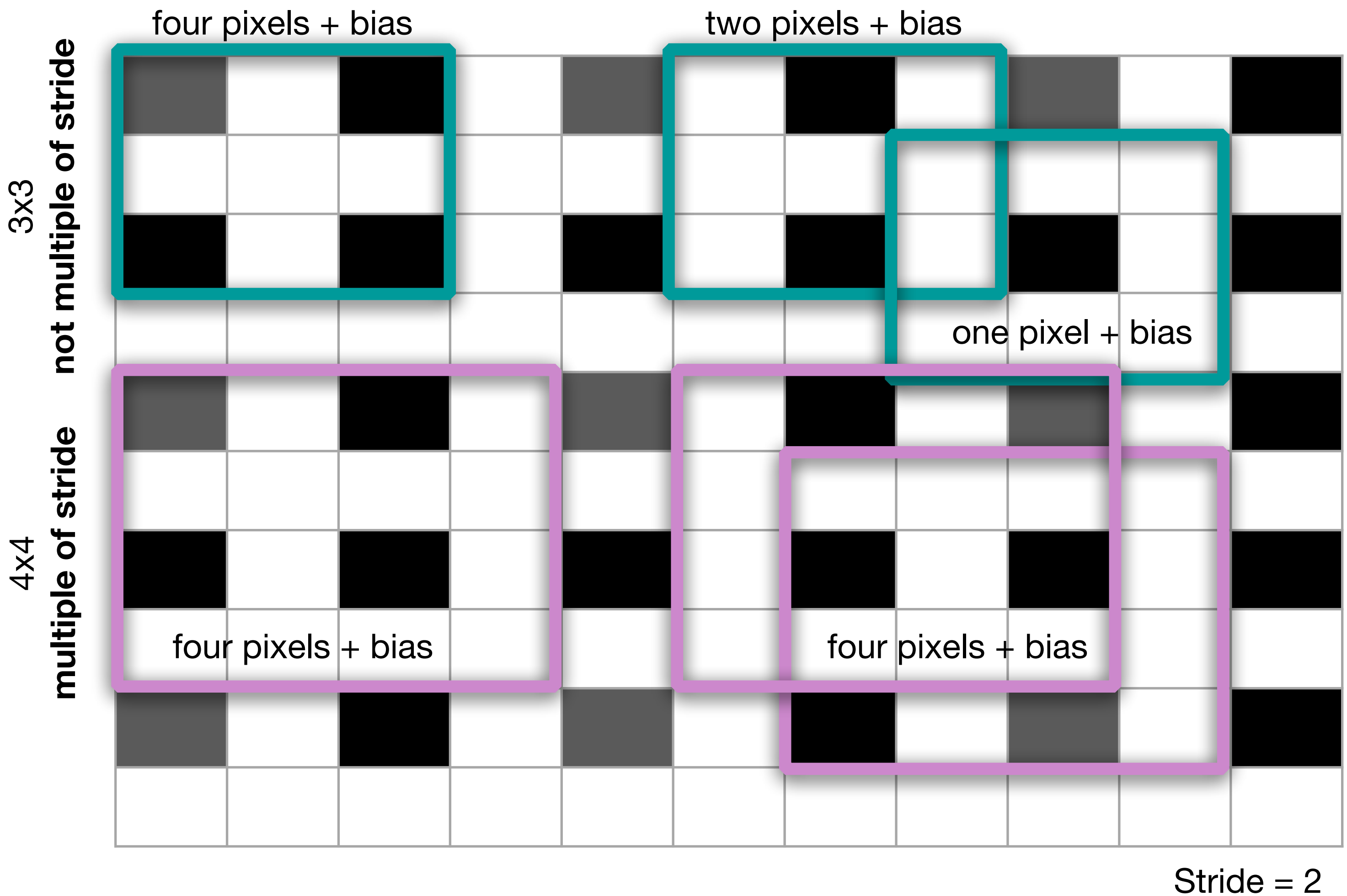
Bilinear Filtering



Bicubic Filter

Learned Upsampling after Zero Insertion

- Learning the interpolation filter has some caveats:



Bias needs to account for both when different numbers of pixels overlap with the kernel

Multiple of stride ensures that same number of active pixels overlap the kernel.

Image Upsampling, Integer Factor

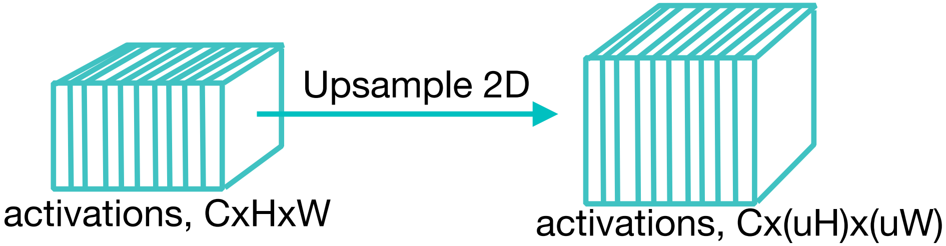
Nearest Neighbor

`UpSampling2D()`

Bilinear

`UpSampling2D(interpolation='bilinear')`

Bicubic



Many Types of Upsampling, with varying computational cost:
area, bicubic, gaussian, lanczos3, lanczos5, mitchellcubic

https://www.cs.toronto.edu/~guerzhoy/320/lec/upsampling.pdf

What about transpose convolution?

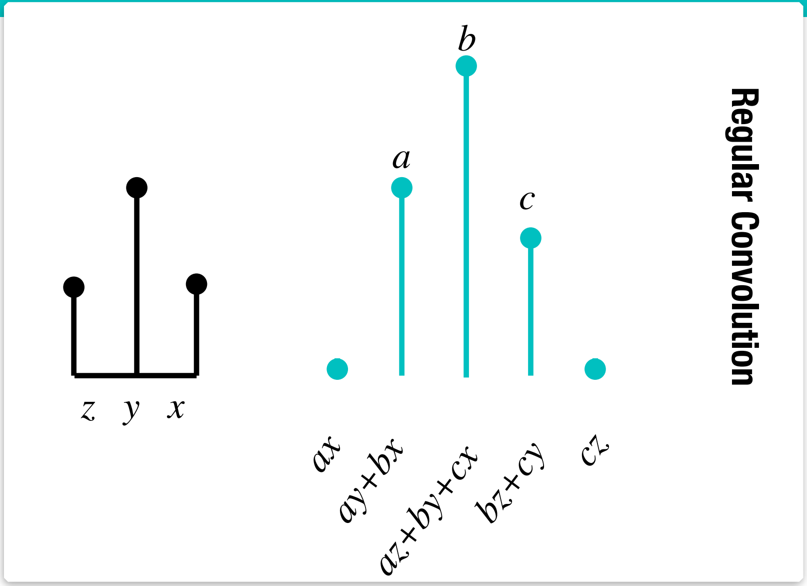
Convolution as Matrix Multiplication

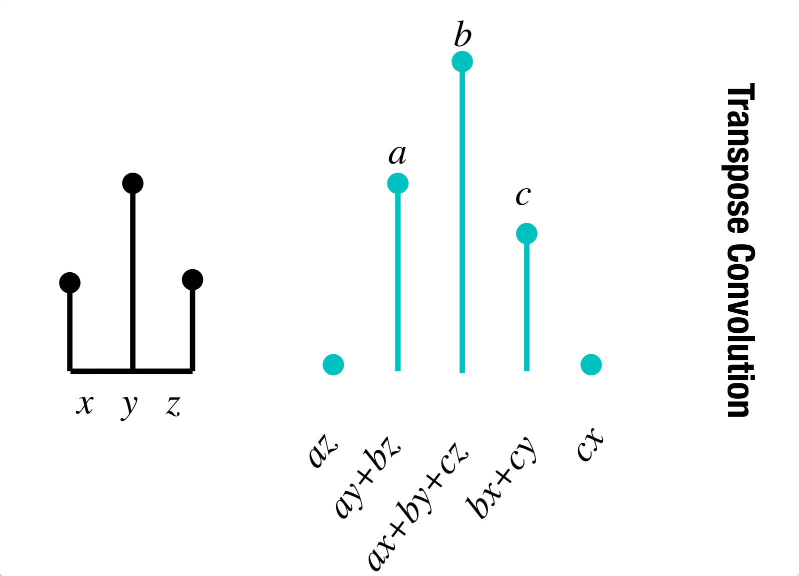
$$\begin{bmatrix} y & x & 0 & 0 & 0 \\ z & y & x & 0 & 0 \\ 0 & z & y & x & 0 \\ 0 & 0 & z & y & x \\ 0 & 0 & 0 & z & y \end{bmatrix} \times \begin{bmatrix} 0 \\ a \\ b \\ c \\ 0 \end{bmatrix} = \begin{bmatrix} ax \\ ay+bx \\ az+by+cx \\ bz+cy \\ cz \end{bmatrix}$$

Transpose

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

like convolving with “reversed coefficients”





Transpose Convolution: Strides

Strided Convolution as Matrix Multiplication

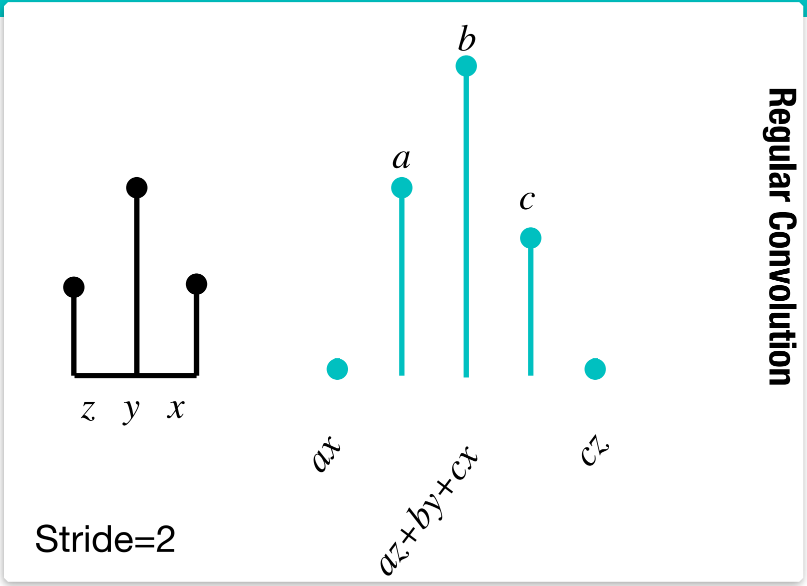
$$\begin{bmatrix} y & x & 0 & 0 & 0 \\ 0 & z & y & x & 0 \\ 0 & 0 & 0 & z & y \end{bmatrix} \times \begin{bmatrix} 0 \\ a \\ b \\ c \\ 0 \end{bmatrix} = \begin{bmatrix} ax \\ az+by+cx \\ cz \end{bmatrix}$$

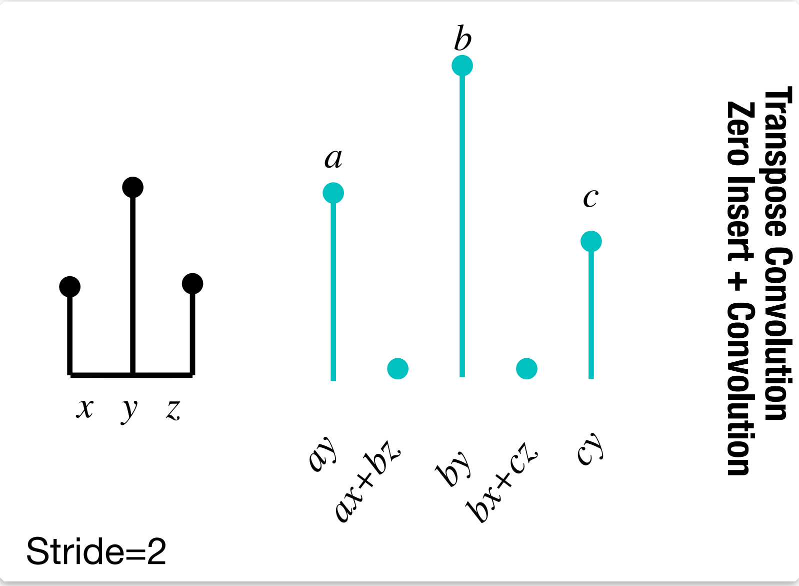
$\downarrow$

Transpose

$\downarrow$

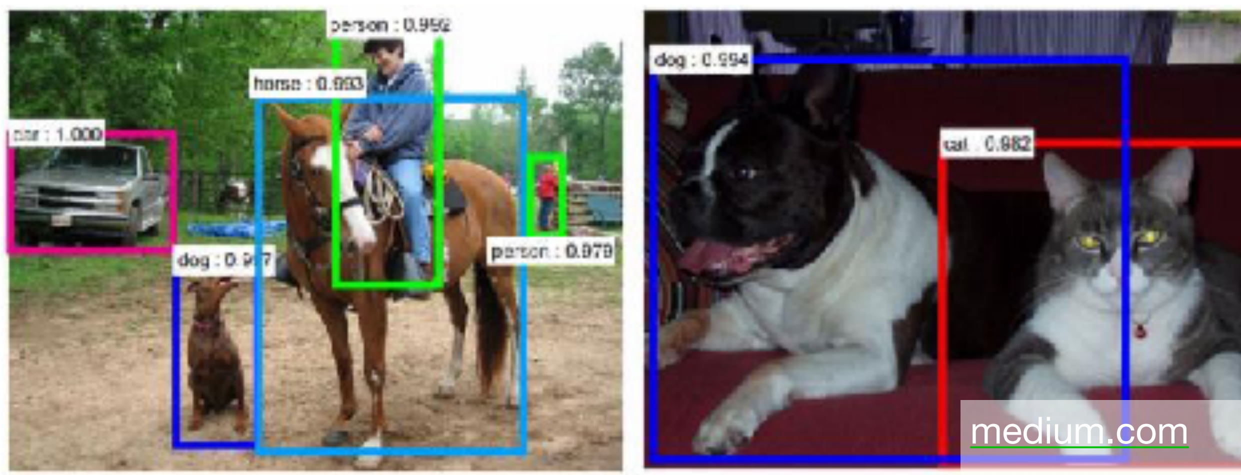
$$\begin{bmatrix} y & 0 & 0 \\ x & z & 0 \\ 0 & y & 0 \\ 0 & x & z \\ 0 & 0 & y \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} ay \\ ax+bz \\ by \\ bx+cz \\ cy \end{bmatrix}$$





This time… Object Detection Methods

- Semantic segmentation has good mIoU values (up to 90%) but this is exaggerated by background recognition, many classes are <40%
- How to adapt these techniques to get bounding boxes, not semantic segmentations?
  - Could this be easier? More stable?
  - More consistent labeling?
  - Suitable for “higher risk” tracking applications?

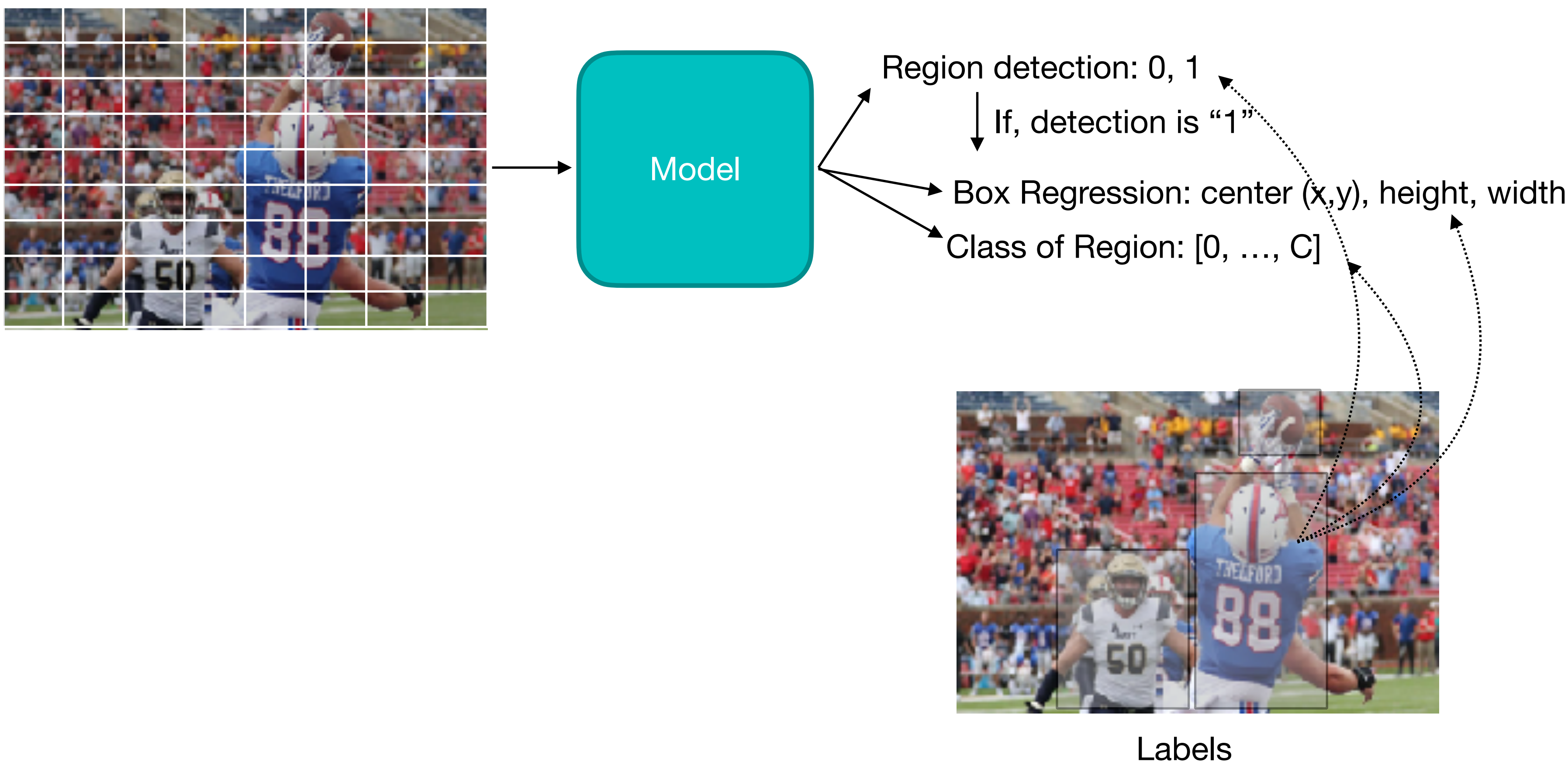


Object Detection with RCNN

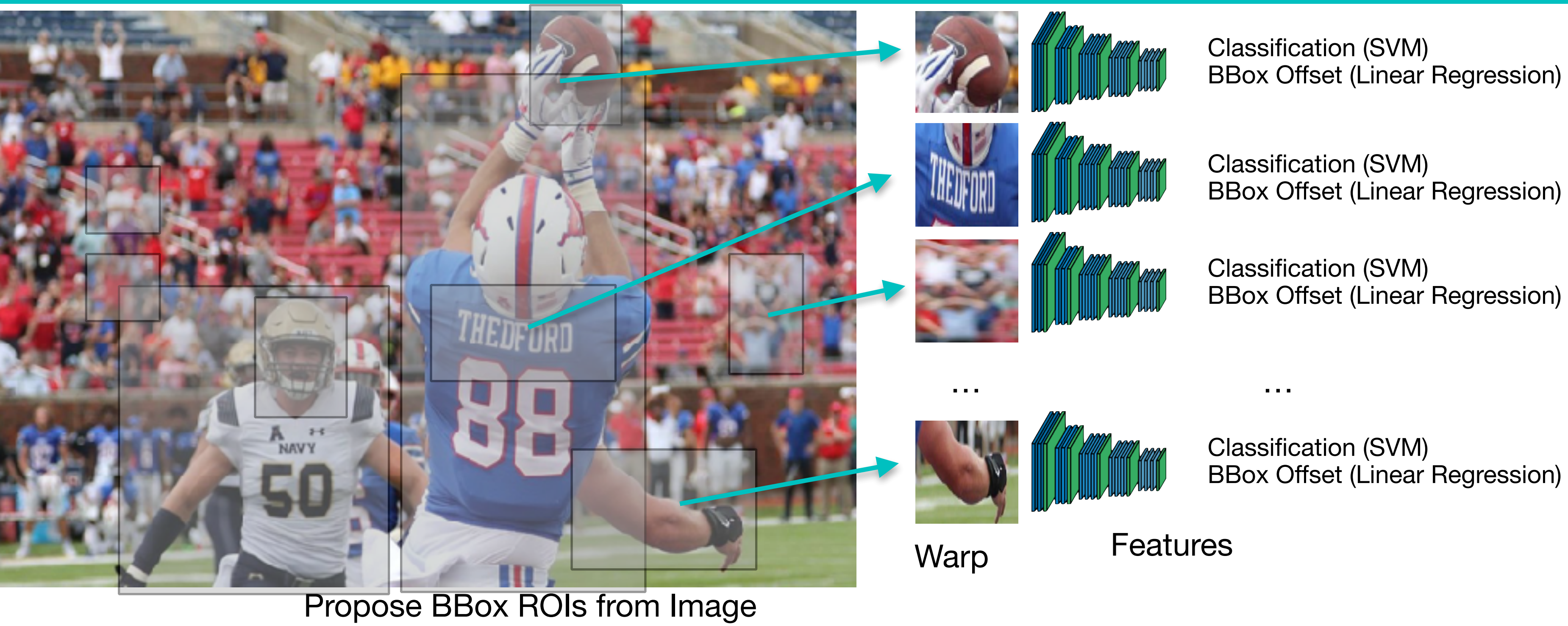
A history in naming one network five different times
with five different papers
each time changing one thing about the architecture

Research!

General Structure



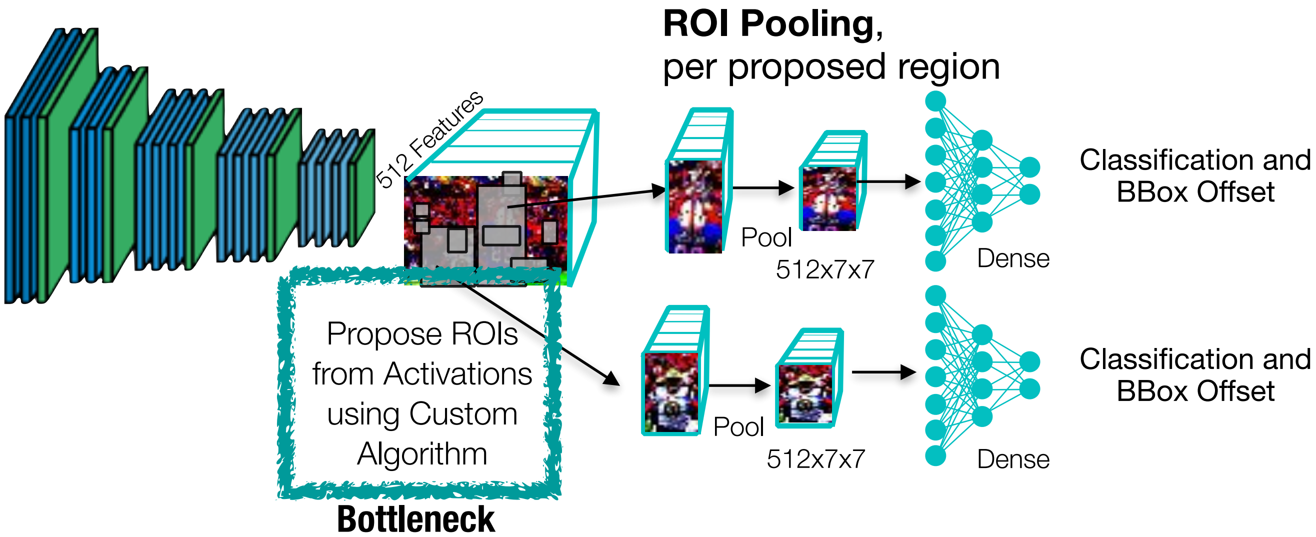
2014: R-CNN



- Too Slow to Be Useful
- SVM and BBox Regression Trained Separately
- Fine Tuned Existing ConvNet (for Warped Images)
- ~50 Seconds per Image when Deployed

Girshick et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, 2014

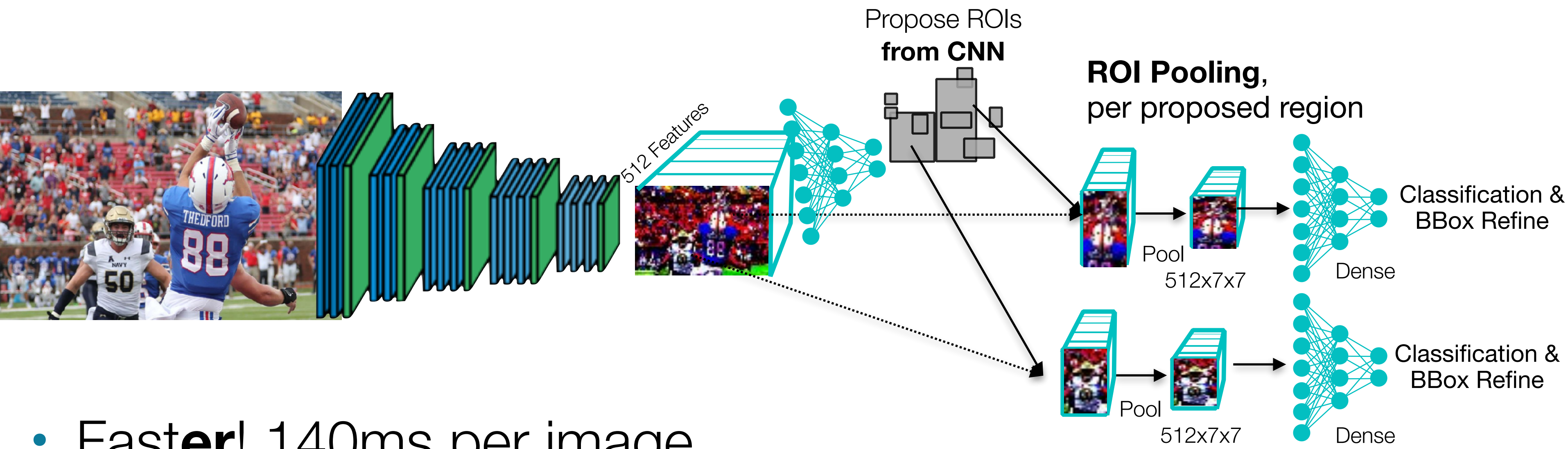
2015: Fast R-CNN

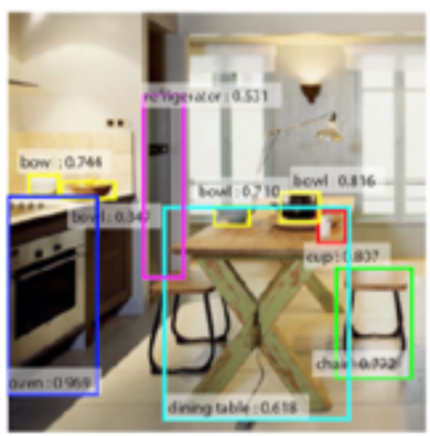
- Fast! 2.3 seconds per image (not ~50)
- But still not real time…

Girshick et al. Fast R-CNN, 2015, February

2015: Faster R-CNN

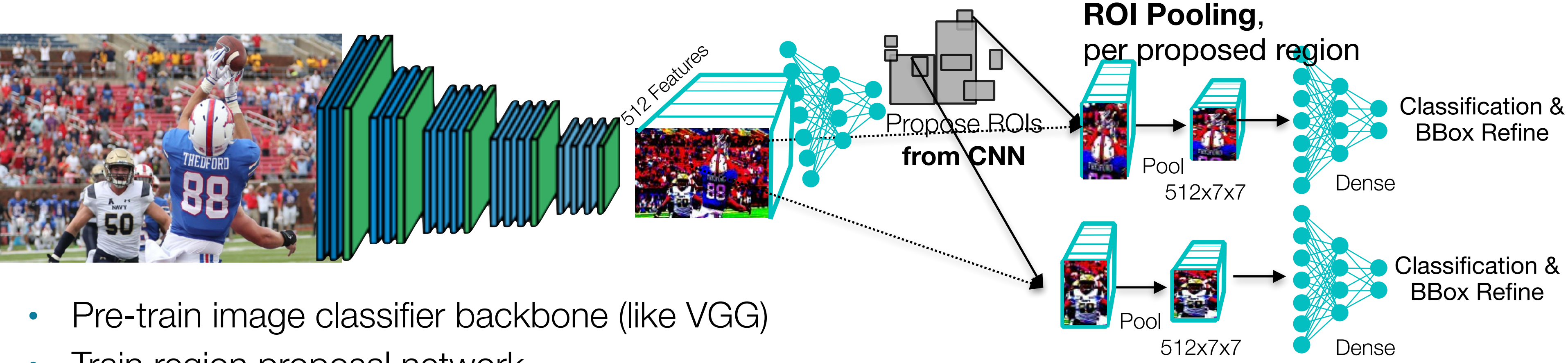


- Fast**er**! 140ms per image (7 FPS)
- Highly Accurate

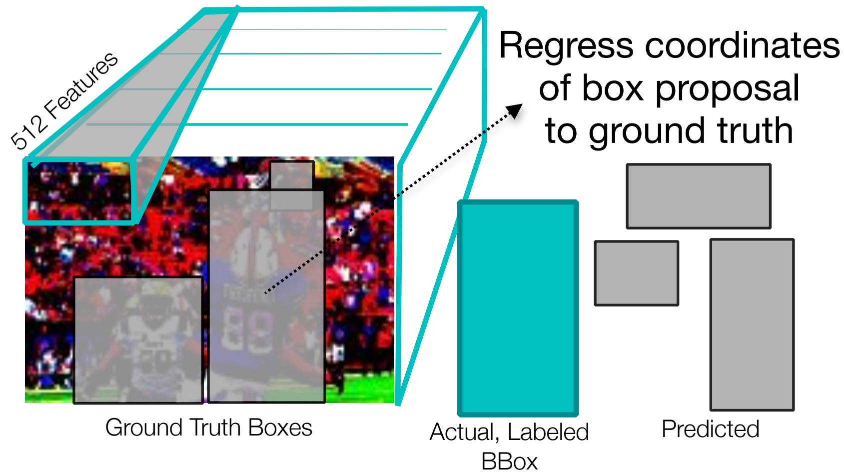


Ren et al. Faster R-CNN, 2015, November

2015: Faster R-CNN, Training



- Pre-train image classifier backbone (like VGG)
- Train region proposal network
  - Sliding window (1x1 conv) across CNN activations from backbone classifier
  - Regress multiple bounding boxes (usually k proposals)
  - Regress "object-ness" of each box
- Train Fast R-CNN on generated ROI proposals
- Fix weights of classifier pipeline, fine tune RPN
- Fix RPN and fine tune classifier
- Rinse, repeat fine tuning



$$l_{box} = \sum_i \hat{p}_i \left[(x-\hat{x}_i)^2 + (y-\hat{y}_i)^2 + (\log w - \log \hat{w}_i)^2 + (\log h - \log \hat{h}_i)^2 \right]$$

$$l_{class} = \sum_c CE(c, \hat{c})$$

$$l_{obj} = \sum_i CE(p_i, \hat{p}_i)$$

Ren et al. Faster R-CNN, 2015, November