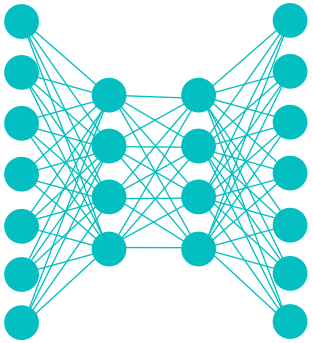
Lecture Notes for

# Neural Networks and Machine Learning

## CNN Visualization



# Logistics and Agenda

- Logistics
  - First Lab Due Soon
- Agenda
  - Visualizing Convolutional Architectures



# CNN Visualization

## Paper Gestalt

Carven von Bearnensquash
Department of Computer Science
University of Phoenix
bearnsquash@live.com

**Abstract**

*Peer reviews of conference paper submissions is an integral part of the research cycle, though it has unknown origins. For the computer vision community, this process has become significantly more difficult in recent years due to the volume of submissions. For example, the number of submissions to the CVPR conference has tripled in the last ten years. For this reason, the community has been forced to reach out to a less than ideal pool of reviewers, which unfortunately includes uninformed junior graduate students, disgruntled senior graduate students, and tenured faculty. In this work we take the simple intuition that the quality of a paper can be estimated by merely glancing through the general layout, and use this intuition to build a system that employs basic computer vision techniques to predict if the paper should be accepted or rejected. This system can then be used as a first cascade layer during the review process. Our results show that while rejecting 15% of "good papers", we can cut down the number of "bad papers" by more than 50%, saving valuable time of reviewers. Finally, we feed this very paper into our system and are happy to report that it received a posterior probability of 88.4% of being "good".*

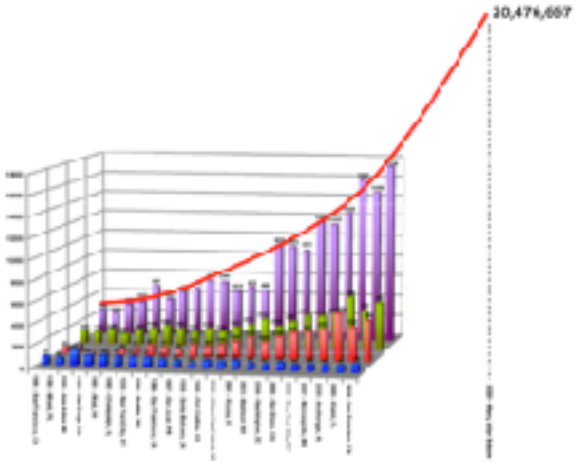



Figure 1. **Paper submission trends**. The number of submitted papers to CVPR, and other top tier computer vision conferences, is growing at an alarming rate. In this paper we propose an automated method of rejected sub par papers, thereby reducing the burden on reviewers.

and tenured faculty. Although many excellent research pa-

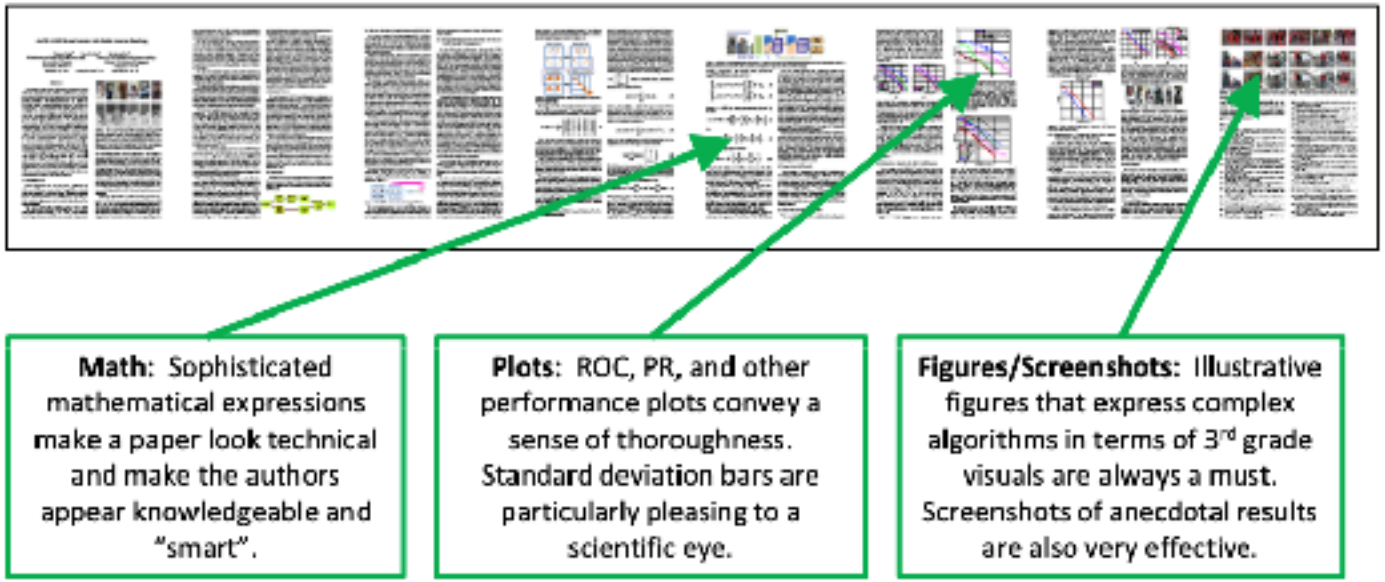



Figure 6. Characteristics of a "Good" paper.

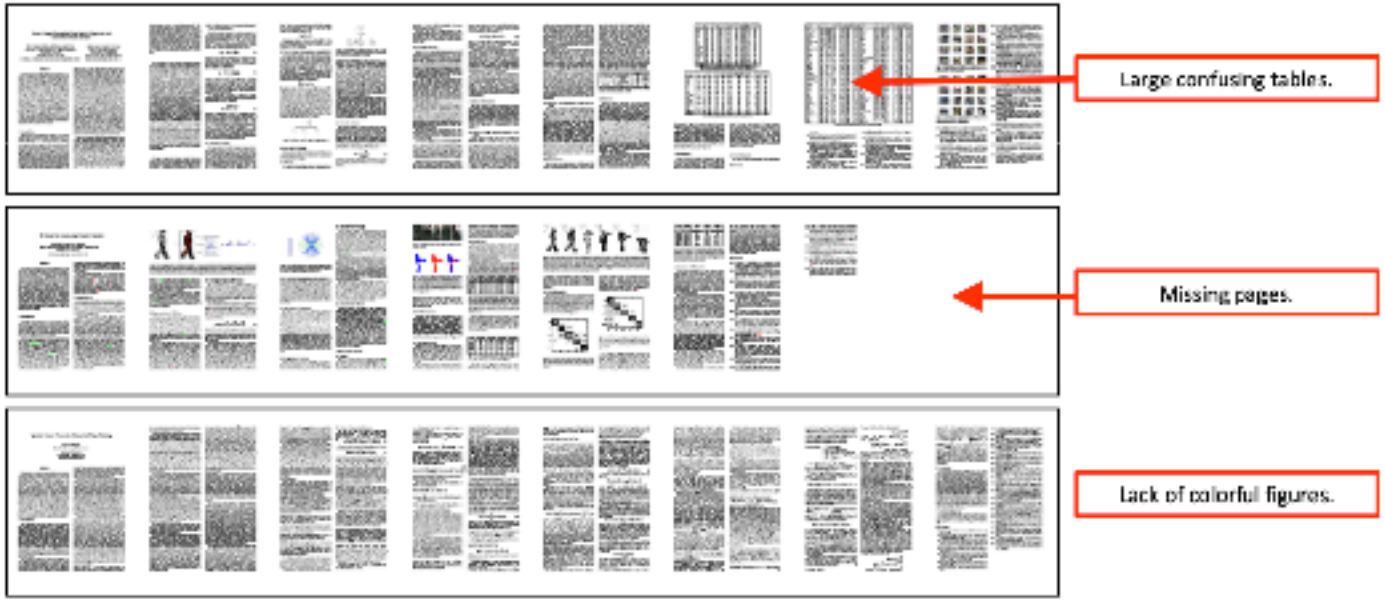



Figure 7. Characteristics of a "Bad" paper.

# Why Visualize Trained CNN Architectures?

- **Reliability**:
  "1998 Military Project"
  Tanks in Trees
  Maybe an Urban Legend, but Plausible

- **Curiosity**: Answering why something works and trying to discover it is *research*

- **Improvement**: You cannot change what you do not understand

https://neil.fraser.name/writing/tank/



# Attribution versus Influence

- **Attribution**:
  The degree to which a particular neuron can be attributed to the decision of the Network. No need to assign “influential regions.”

- **Influence**:
  The sensitivity of regions of interest to a particular input or input distribution. Influential regions must be grouped within the network.

- Difference is nuanced, but important in research community.

Leino, Klas, Shayak Sen, Anupam Datta, Matt Fredrikson, and Linyi Li. "Influence-directed explanations for deep convolutional networks." In *2018 IEEE International Test Conference (ITC)*, pp. 1-8. IEEE, 2018.

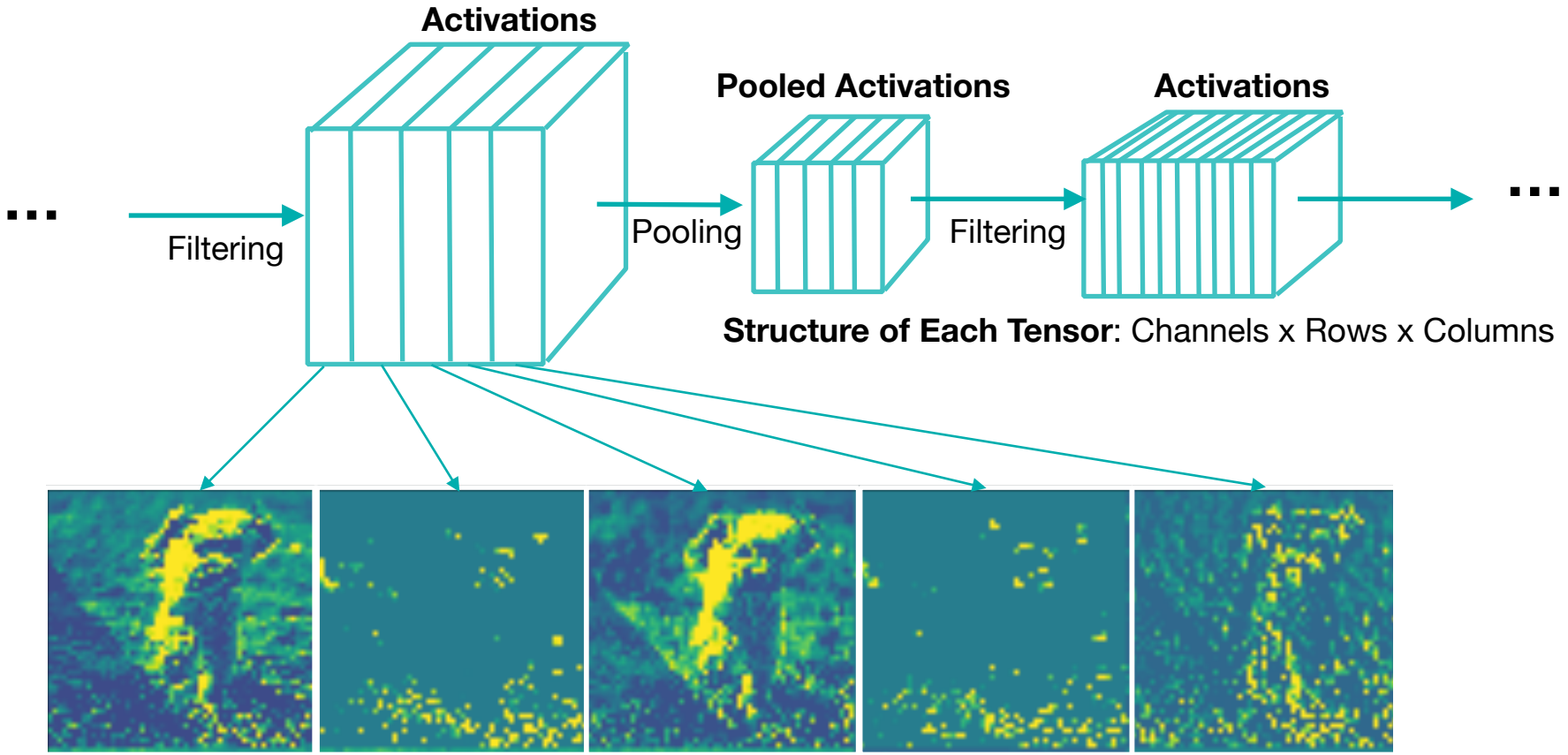


# Methods for Visualization

- No “Ideal” Technique, Depends on Question
- Visualize **Filter Activation Maps**
  - What parts of the images activate each filter?
- Visualize **Filters**
  - What is each filter most sensitive to?
- **Heatmaps** of Class Activation
  - What part of an input image most influences each final output?

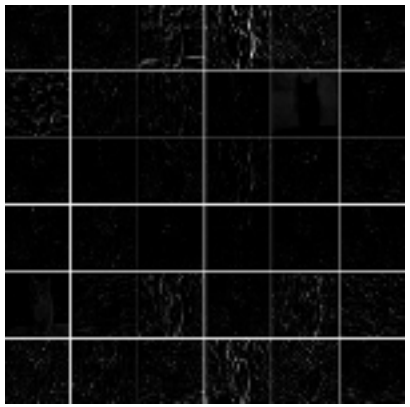

# Visualizing Intermediate Activations

- Look layer by layer
- **Assume**: each filter learns something independent



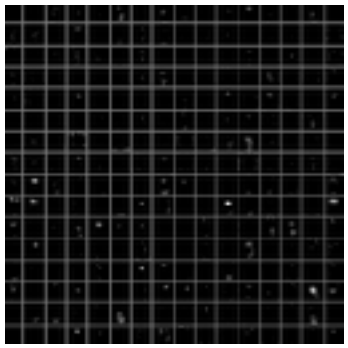
# Visualizing Intermediate Activations

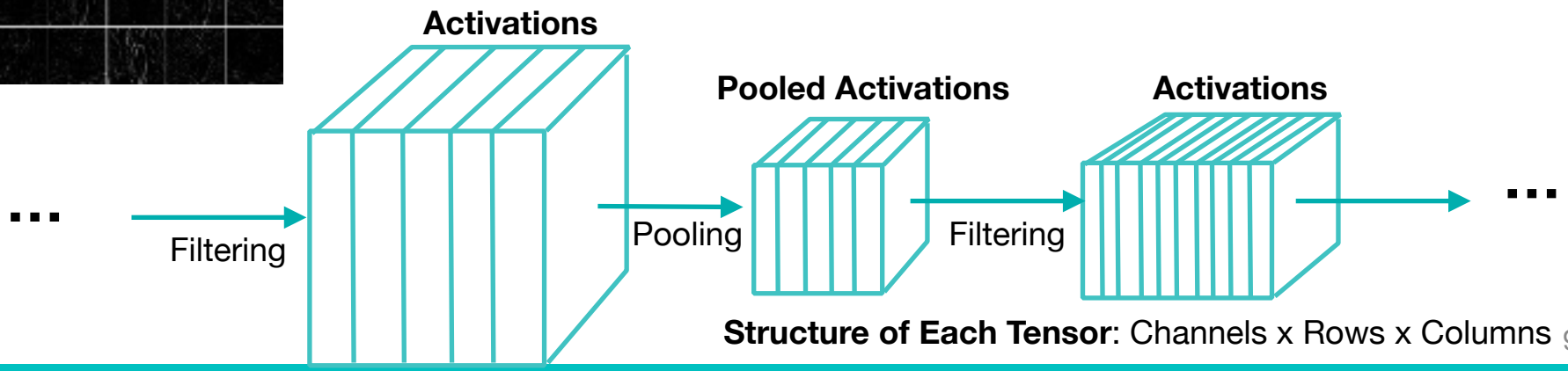
- **Recall**: general structure of most CNNs
  - Small kernels throughout (3x3)
  - Filtering followed by Pooling (spatial downsampling)
  - More filters in later layers



**Early Activations** are larger but not as numerous

**Later Activations** are smaller and more numerous



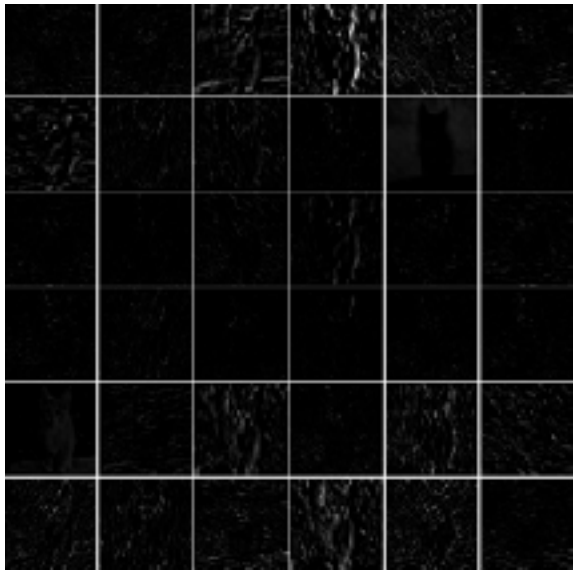



**Structure of Each Tensor**: Channels x Rows x Columns



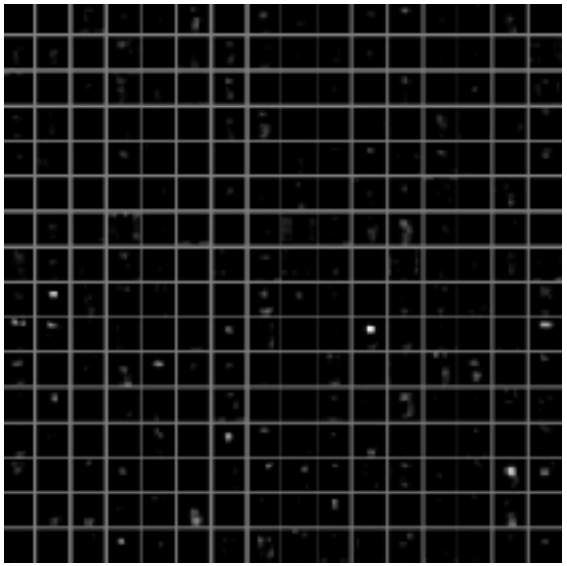
# Visualizing Intermediate Activations

- **Result:** Information Distillation Pipeline
  - Deeper layers have more abstract triggers
  - Deeper activations are increasingly sparse
  - Early layers are texture and edge detectors
  - Notion of "High Level Abstraction," has biological motivation



**Early Activations** are larger but not as numerous

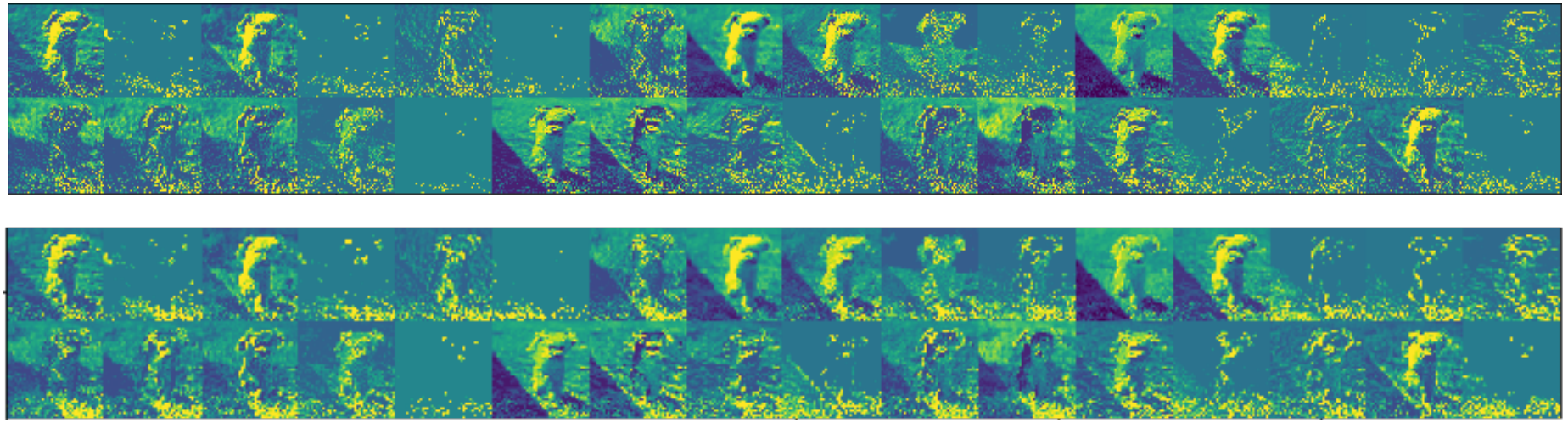
**Later Activations** are smaller and more numerous





# Visualizing ConvNets

## Part One: Filter Activations

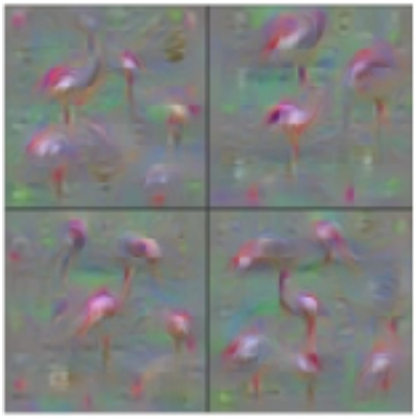



Follow Along: LectureVisualizingConvnets.ipynb
activation-demo

# Visualizing Filters: Class Neuron

- **Idea:** What Maximally Activates a Class Output?
  - Gradient Ascent in the Input Space

$$I \leftarrow I + \eta \nabla f_c(I)$$



Flamingo

where $c$ is a specific neuron in output layer

$f$ is the activation before softmax

$I$ is the input image, init to zeros (or random)

gradient update is for $I$

CNN weights stay unchanged

http://cs231n.github.io/understanding-cnn/



# Visualizing Filters: Maximal Activations

- **Idea:** What Maximally Activates a **Filter**?
  - **Again**: Gradient Ascent in the Input Space

$$I \leftarrow I + \eta \sum_{i,j} \nabla f_n(I)_{i,j}$$

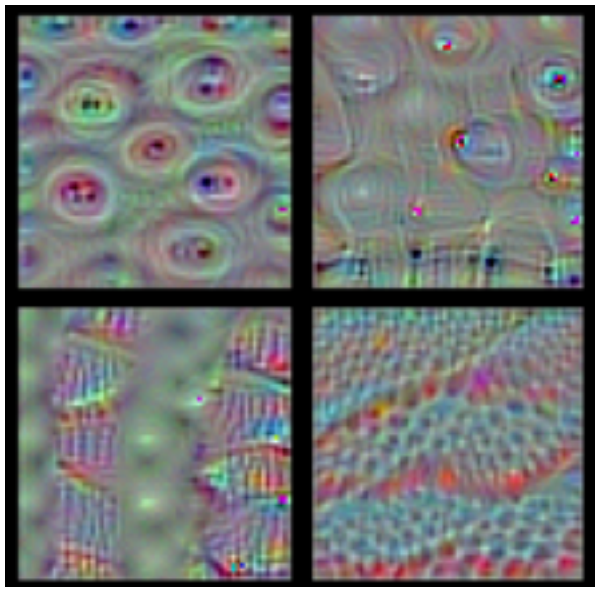
**"trick"** must use norm of gradient

where $n$ is a specific **filter** in a layer

$f$ is the activation of $n^{th}$ filter in layer

$I$ is some random image, or zeros

gradient update is for $I$





# Visualizing Filters: Deep Dream

- **Idea:** How to generate new concepts (**dream**)?
  - Reinforce what is learned? How?

noise process

$$I \leftarrow \epsilon_{iter}\left(I + \eta \sum_{i,j} \nabla f_A(I)_{i,j}\right)$$

use inverse L1 norm of gradient

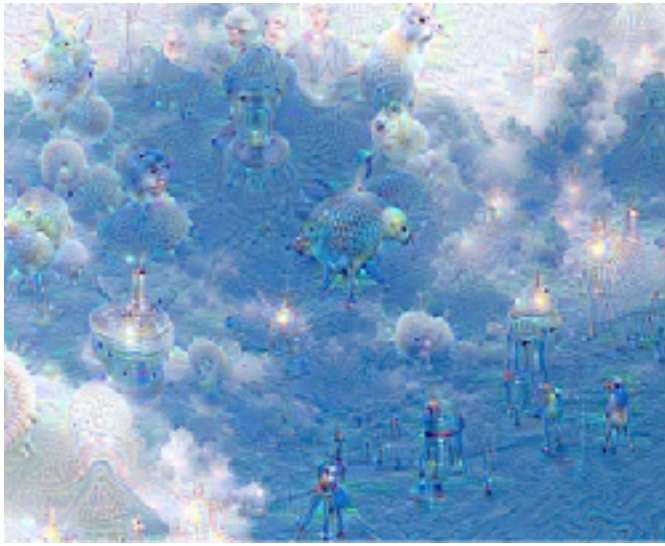
where $A$ is a specific **conv layer**

$f$ is all **activations** from $A^{th}$ layer

$I$ is init to **some input image**

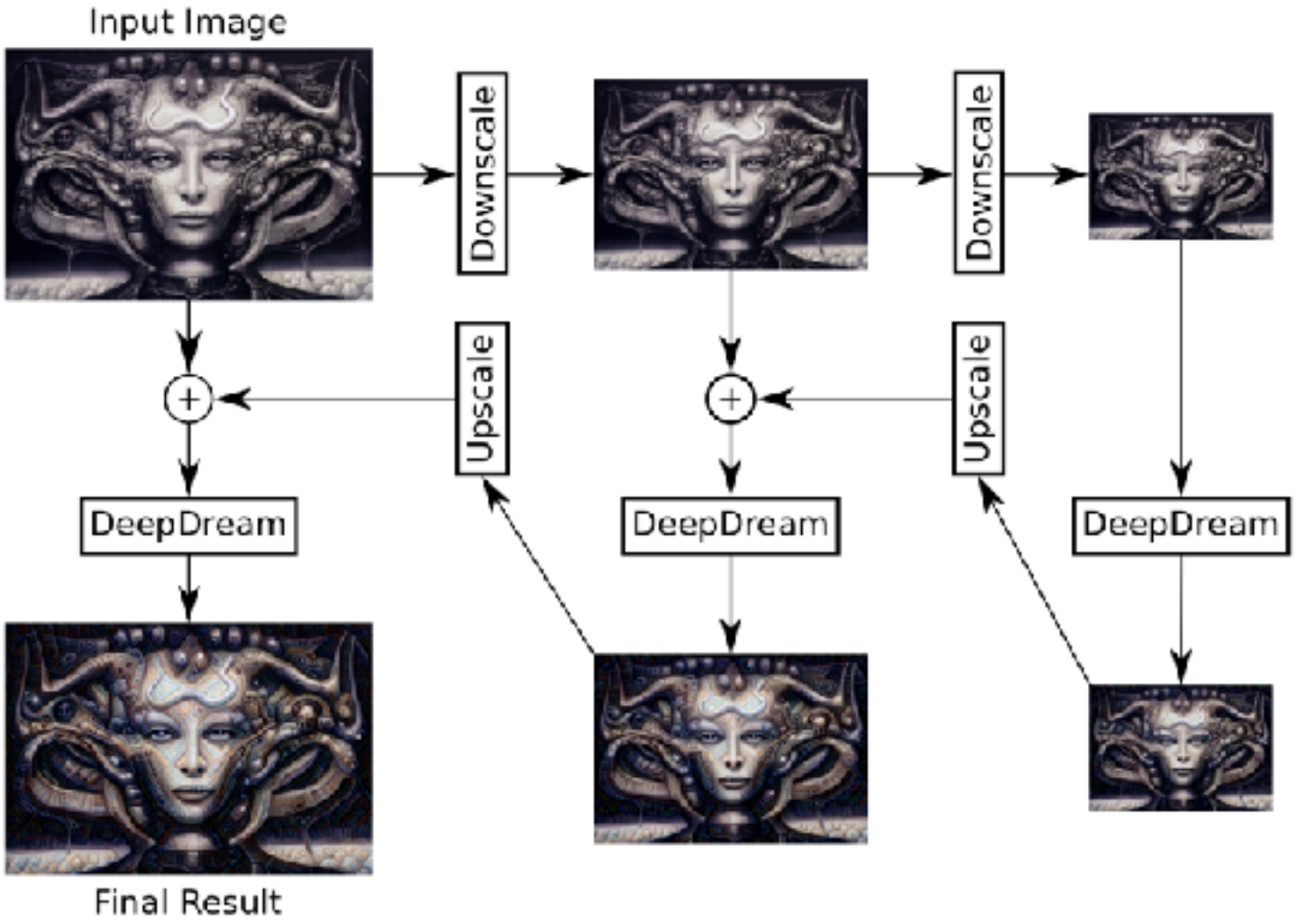
gradient update is for $I$

$\varepsilon$ add randomness through scaling





# Noise Injection via Scaling




https://hackernoon.com/dl06-deepdream-with-code-5f735052e21f



# DeepDream Practically

## Dreams and hallucinations

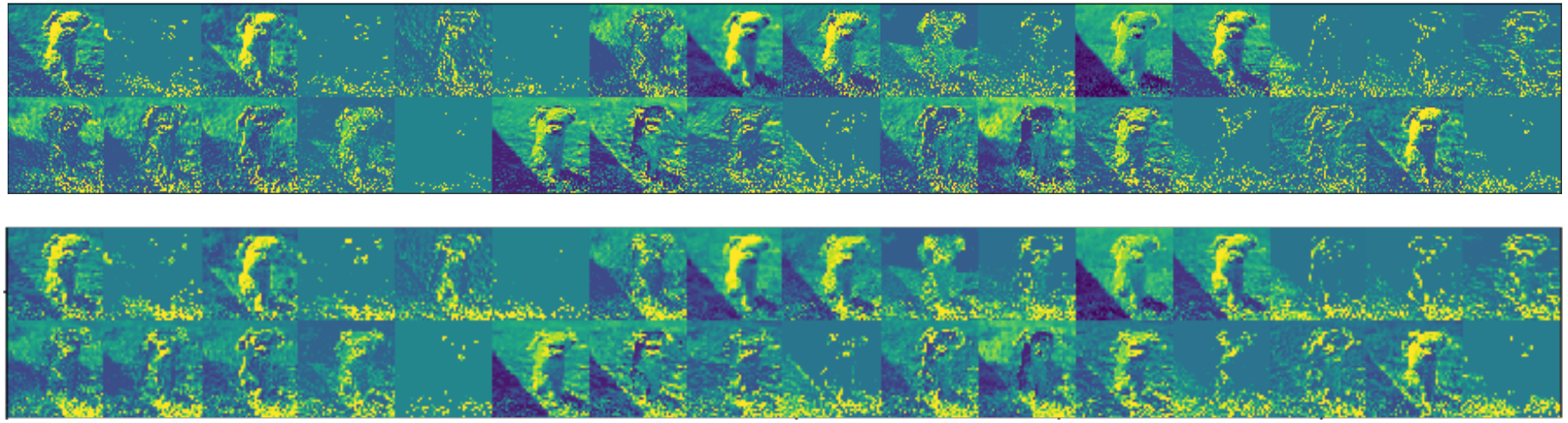
The process of investigating a neuron's response forces the network to *imagine* something which is not there. Terms such as *dreams* and *hallucinations* have often been used. Despite not being related at all with actual dreams and hallucinations, they've captured the media attention. Dreams are actually very important because they highlight what the network has learnt. Alexander Mordvintsev has shown how a network trained to recognise dumbbells only generates dumbbells which are hold by a human arm. That's a clear indication the network has failed to understand what a dumbbell really is, perhaps because it has always been shown pictures of people holding them.

https://www.alanzucconi.com/2015/07/06/live-your-deepdream-how-to-recreate-the-inceptionism-effect/



# Visualizing ConvNets

## Part Two: Image Gradients

Follow Along: LectureVisualizingConvnets.ipynb
activation-demo

# Class Activation Mapping (CAM)

- **Idea:** What areas of the image contributed most to the classification result?
- Also, for each class, what areas of the image exhibit features of that class?
- Use change in output, w.r.t. final conv layer

$$\alpha_k^c = \frac{1}{|I \times J|} \sum_{i,j} \frac{\partial f_c(I)}{\partial A_{i,j,k}^{(L)}}$$

final layer output in response to image $I$
$c$ is class of interest

final convolutional layer, $L$, activations for row, column, channel

gradient weight for channel $k$ and class $c$ in layer $L$
$k$ in $1 \ldots K$ activations in final layer

Ramprasath et al., **Grad-CAM:** Visual Explanations from Deep Networks via Gradient-based Localization



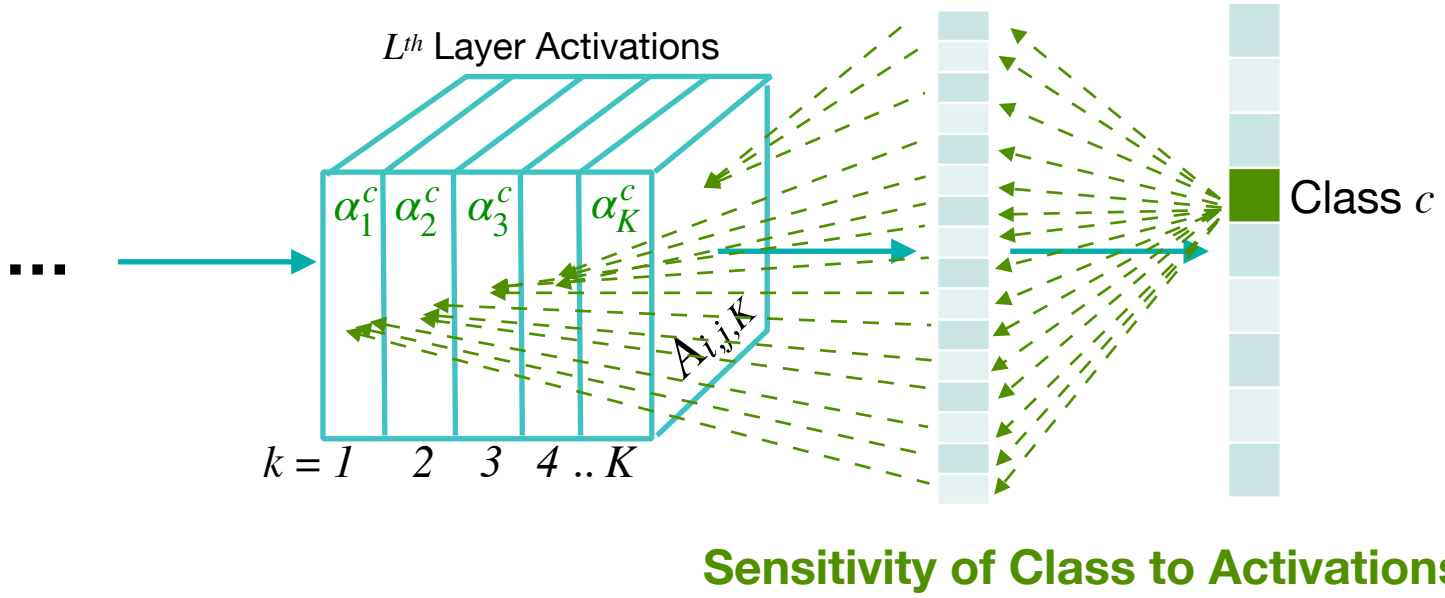
# Class Activation Mapping (CAM)

$$\alpha_k^c = \frac{1}{|I \times J|} \sum_{i,j} \frac{\partial f_c(I)}{\partial A_{i,j,k}^{(L)}}$$

final layer output in response to image $I$
$c$ is class of interest

final convolutional layer, $L$, activations for row, column, channel

gradient weight for channel $k$ and class $c$ in layer $L$
$k$ in $1 \ldots K$ activations in final layer




**Sensitivity of Class to Activations**

Ramprasath et al., **Grad-CAM:** Visual Explanations from Deep Networks via Gradient-based Localization



# Class Activation Mapping (CAM)

$$\alpha_k^c = \frac{1}{|I \times J|} \sum_{i,j} \frac{\partial f_c(I)}{\partial A_{i,j,k}^{(L)}}$$

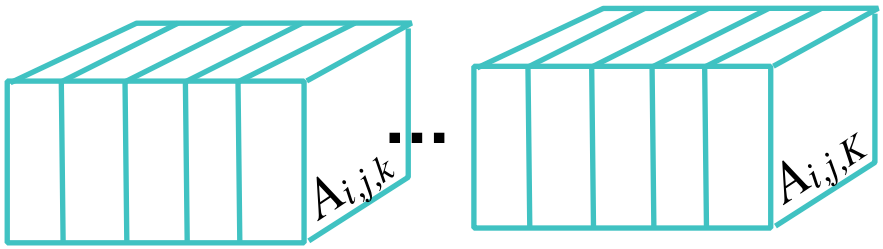
final layer output in response to image $I$
$c$ is class of interest

final convolutional layer, $L$, activations for row, column, channel

gradient weight for channel $k$ and class $c$ in layer $L$
$k$ in $1 \ldots K$ activations in final layer

**Heatmap**, $S$, is the **weighted sum** of final layer activations:

$$S_{i,j} = \frac{1}{S_{max}} \sum_k \alpha_k^c A_{i,j,k}^{(L)}$$

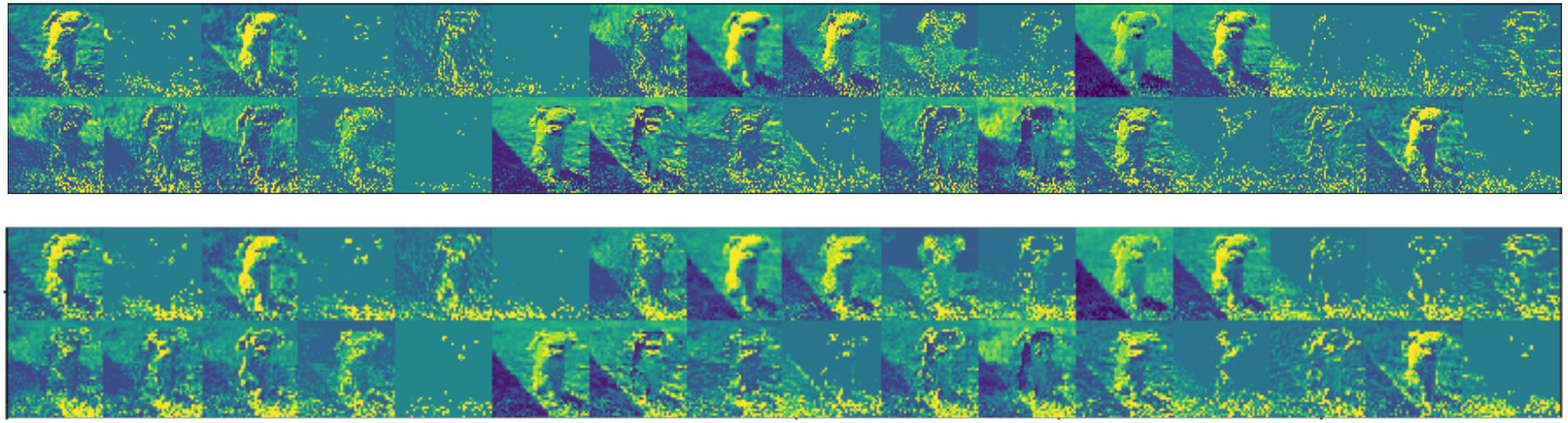



Ramprasath et al., **Grad-CAM:** Visual Explanations from Deep Networks via Gradient-based Localization



# Visualizing ConvNets

## Part Three: Grad-CAM

Follow Along: LectureVisualizingConvnets.ipynb
activation-demo



Lecture Notes for

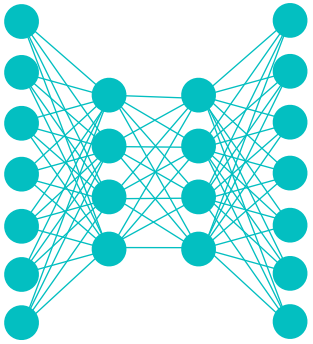
# Neural Networks and Machine Learning

CNN Visualization

**Next Time:**

Fully Convolutional Learning

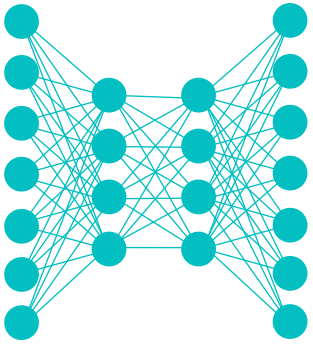
**Reading:** Chollet 5.4



Lecture Notes for

# Neural Networks and Machine Learning

## Fully Convolutional Learning



# Logistics and Agenda

- Logistics
  - Start on Lab One Now!
- Agenda
  - Segmentation
    - Semantic (this time)
    - Object (partially this time, maybe)
    - Instance (next time)



# Types of Fully Convolutional Problems

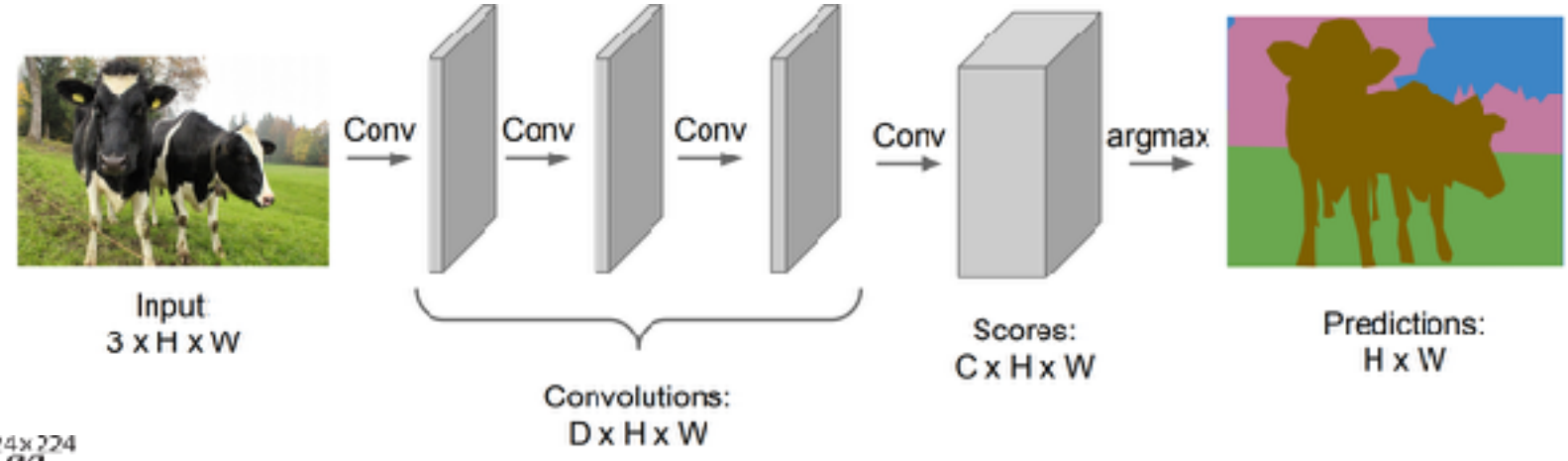
- Semantic Segmentation
- Object Detection
- Instance Segmentation

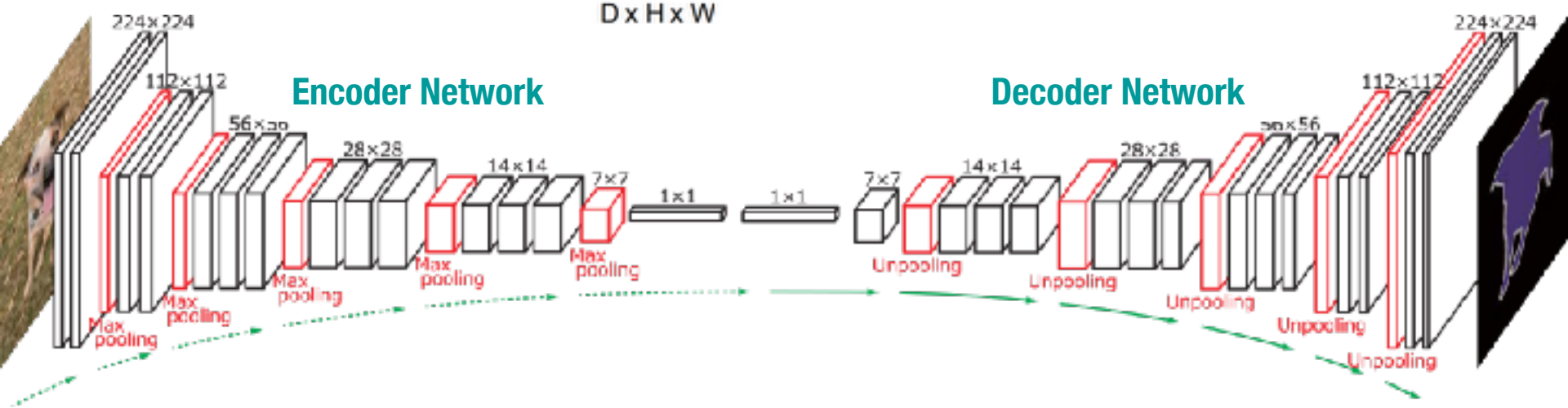


# Semantic Segmentation

# Semantic Segmentation

- Given a set of pixels, classify each pixel according to what instance it belongs

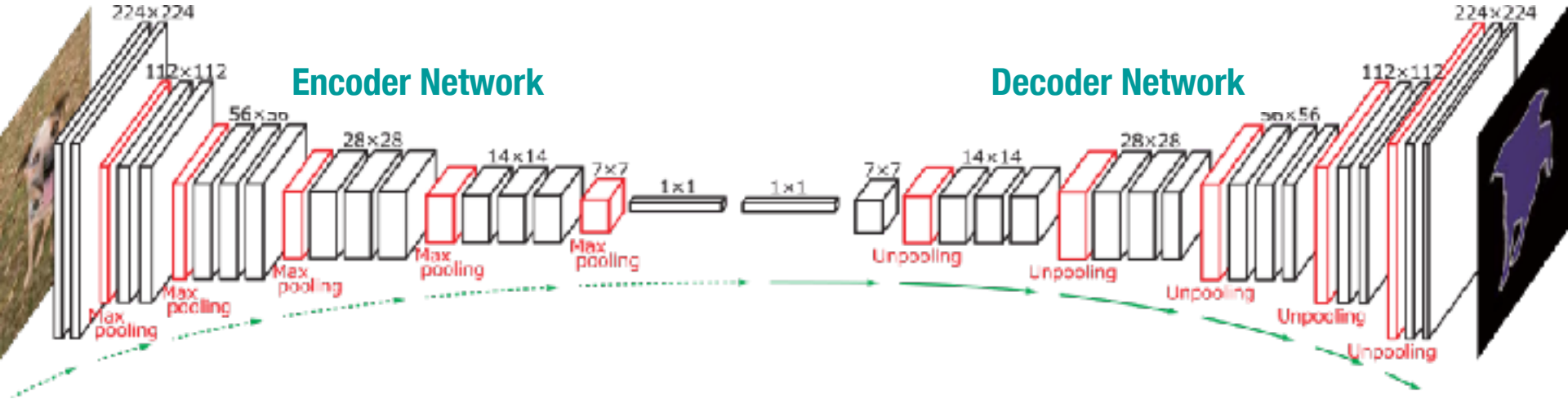



https://tjmachinelearning.com/lectures/1718/fcn/



# Training Procedure



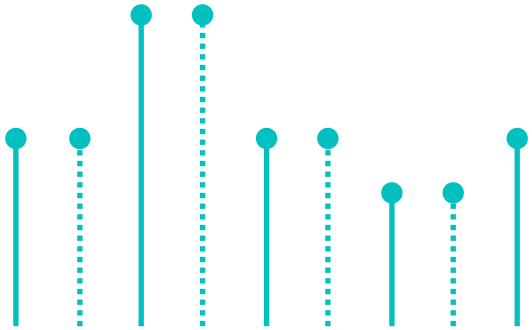

- Init Encoder with traditional CNN (like VGG)
- Freeze encoder and train decoder with segmented image maps
- Unfreeze encoder and fine tune
  - Repeat tuning as needed

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

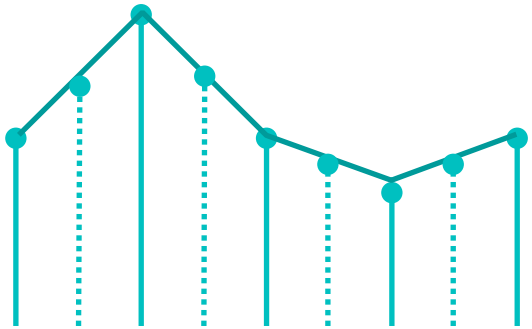


# **Aside**: Integer Upsampling via Interpolation

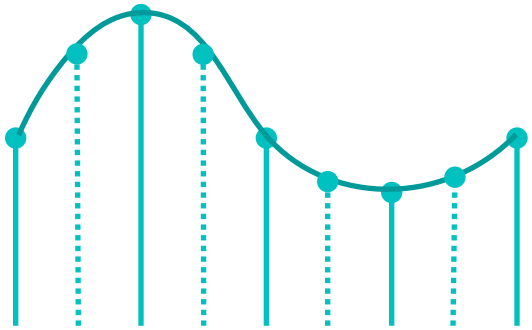
**Nearest Neighbor**



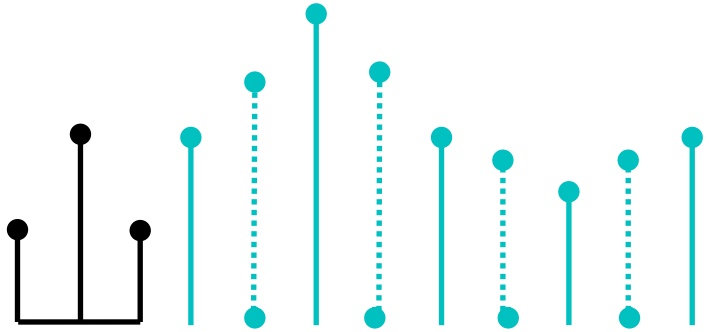
**Linear**

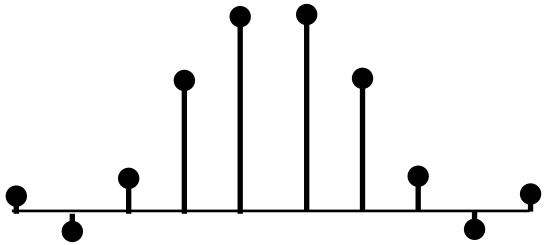


**Cubic**



All are equivalent to inserting zeros and applying convolutional filter







# **Aside**: Image Upsampling, Integer Factor

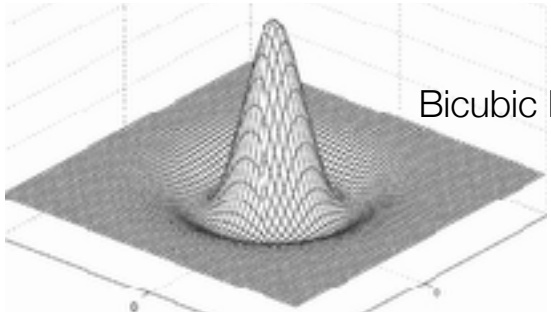
- Insert Zeros
- Convolve

| | | | |
|---|---|---|---|
| 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



# **Aside**: Image Upsampling, Integer Factor

**Nearest Neighbor**

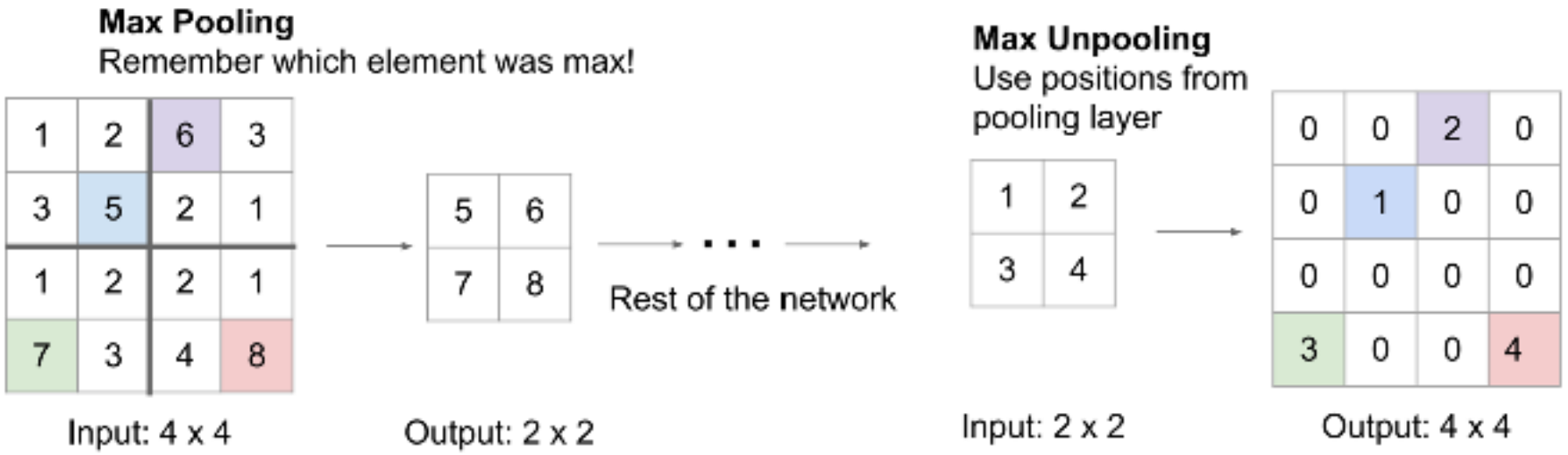
**Bilinear**

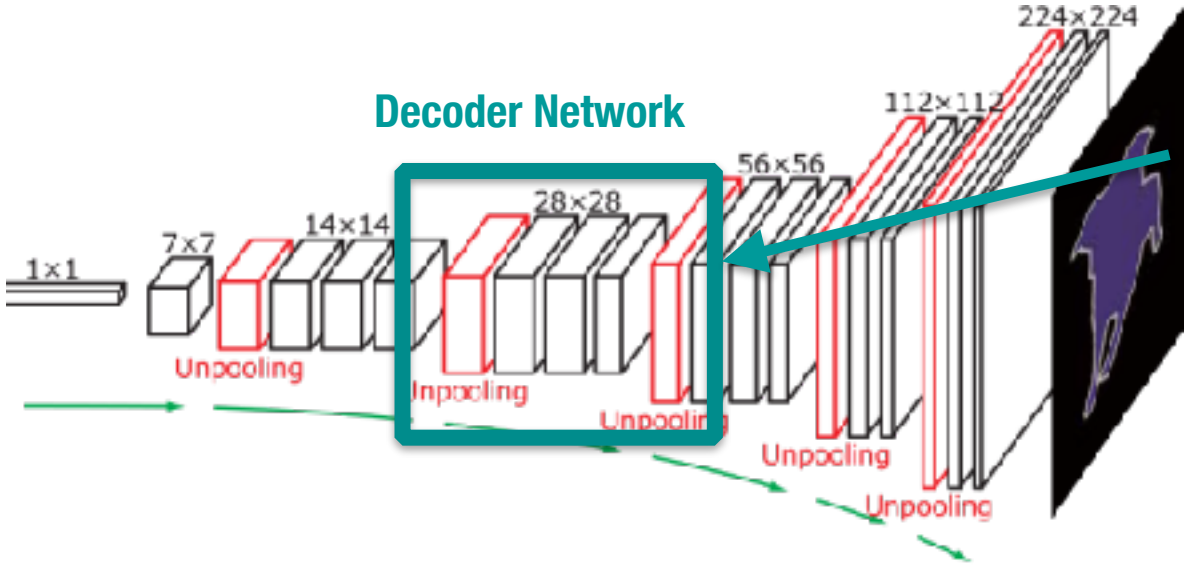
**Bicubic**

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



# Semantic Segmentation




Some knucklehead started calling this **deconvolution**. If you use that term in this class, **you fail**.

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

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



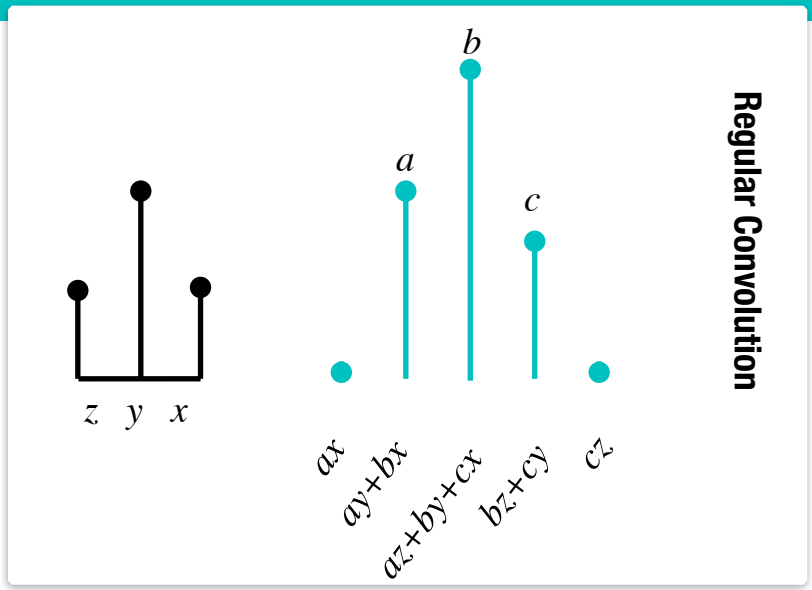
# What about transpose convolution?

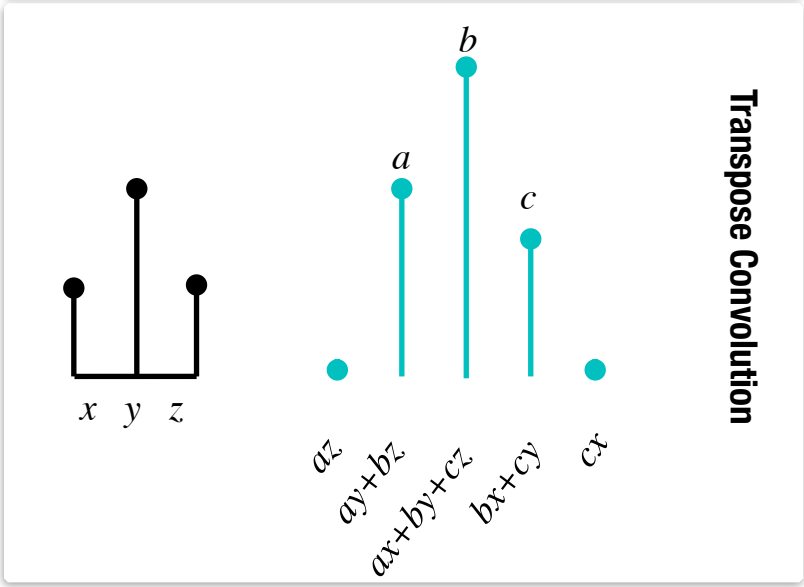
Convolution as Matrix Multiplication

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

Transpose

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





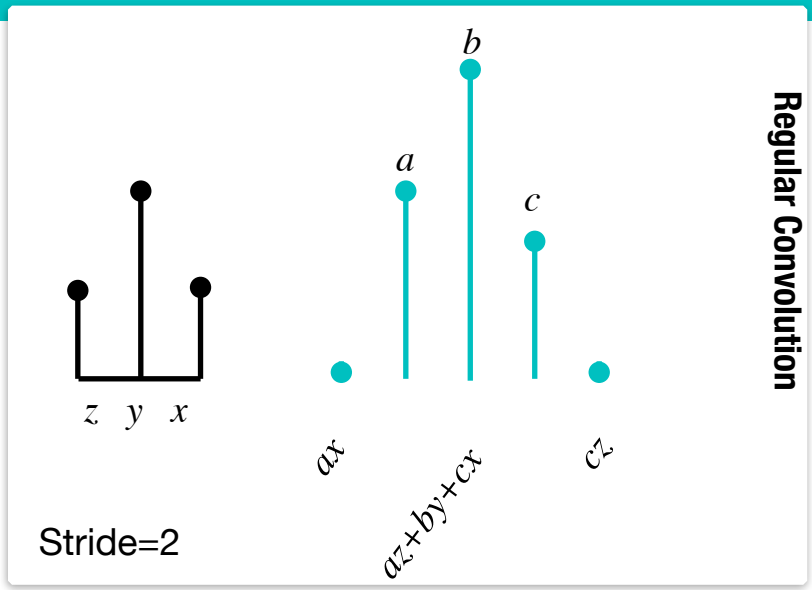

# Transpose Convolution: Strides

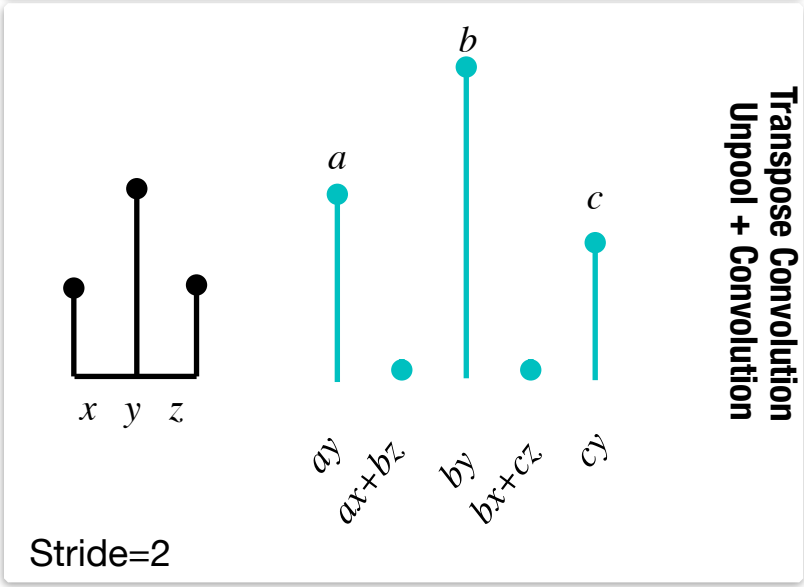
Strided Convolution as Matrix Multiplication

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

Transpose

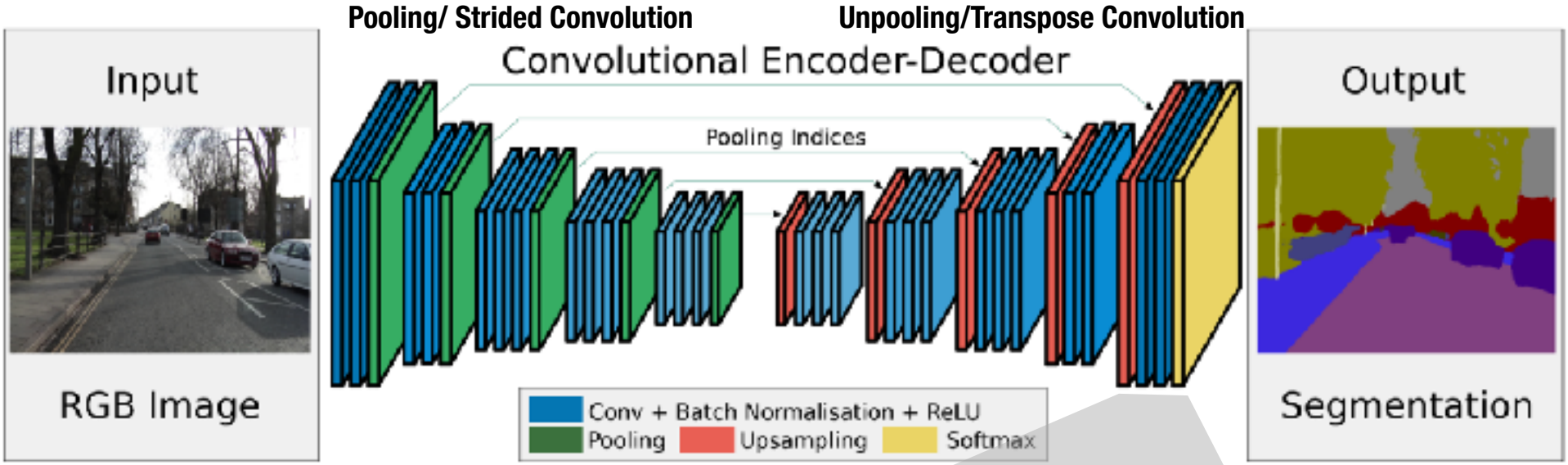
$$\begin{bmatrix} y & 0 & 0 \\ x & z & 0 \\ 0 & y & 0 \\ 0 & x & z \\ 0 & 0 & y \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} ay \\ ax+bz \\ by \\ bx+cz \\ cy \end{bmatrix}$$



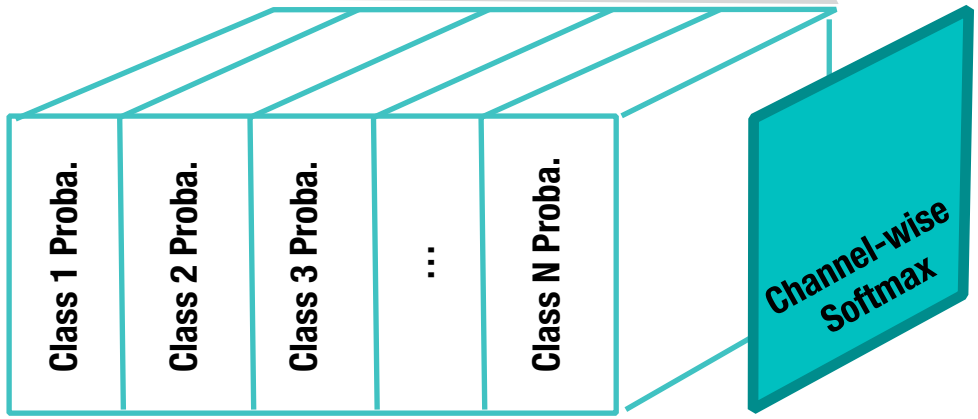






# Putting it all together



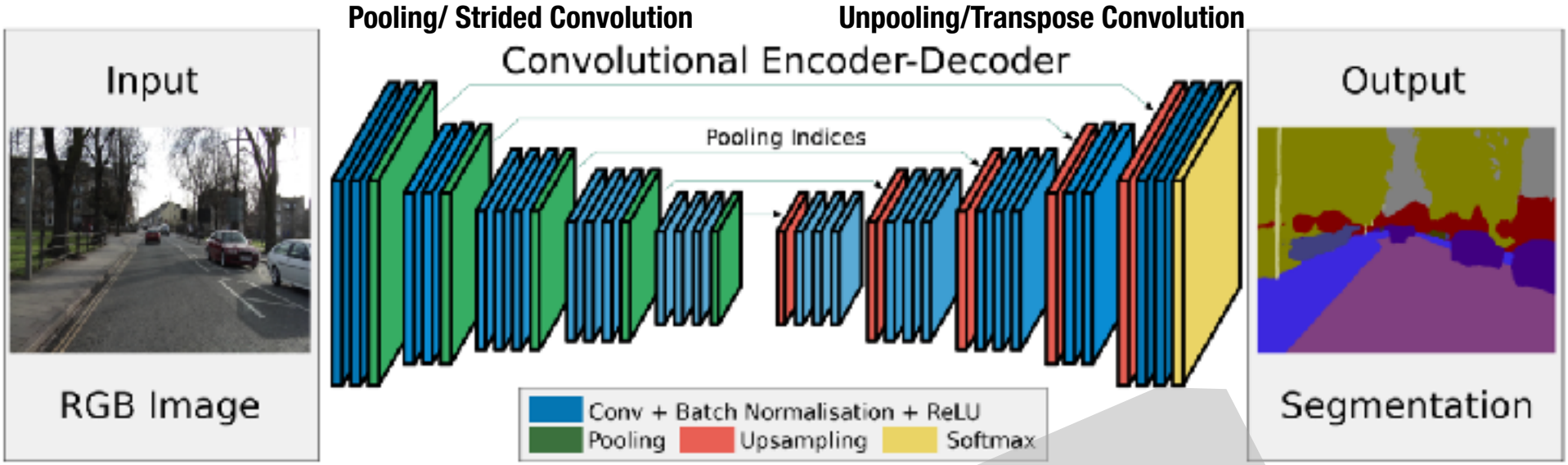

**Self Test:**
Does it change the architecture if the Image input size changes?

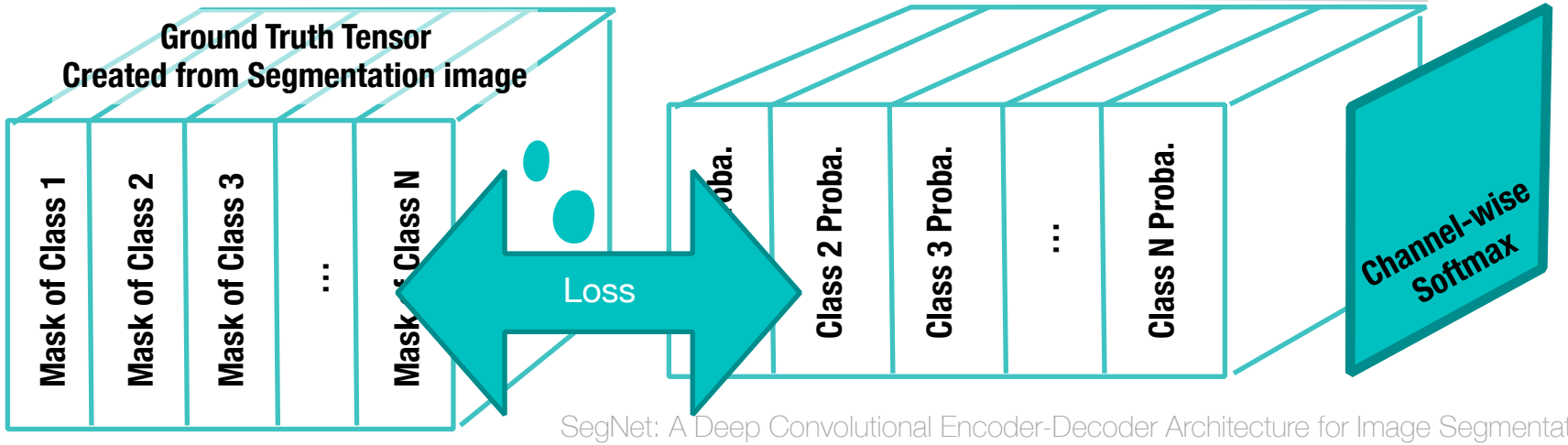



SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation



# Putting it all together



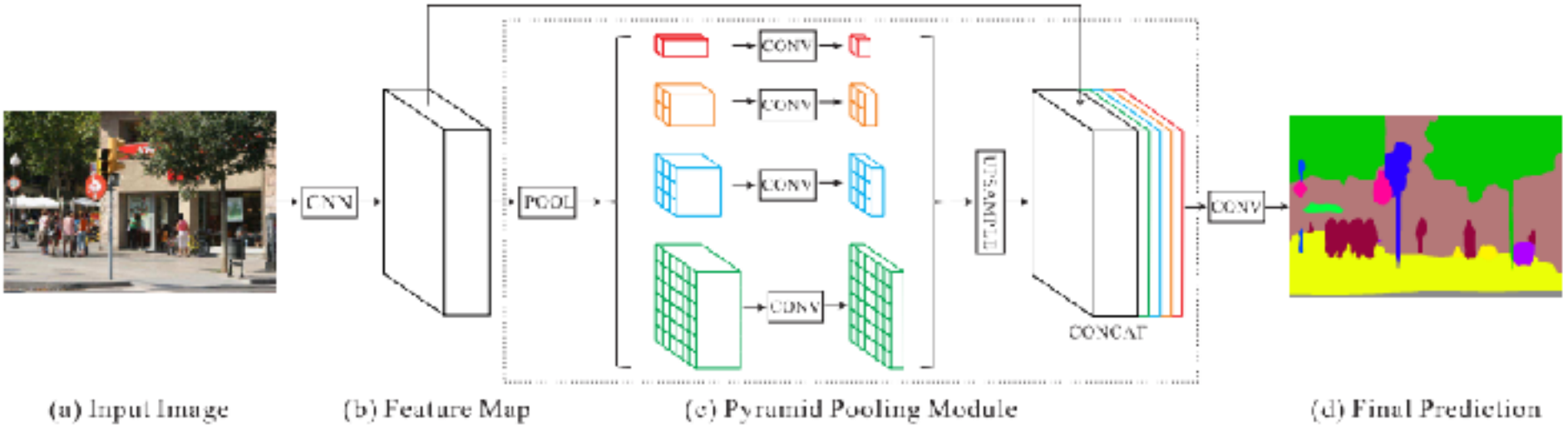




SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation

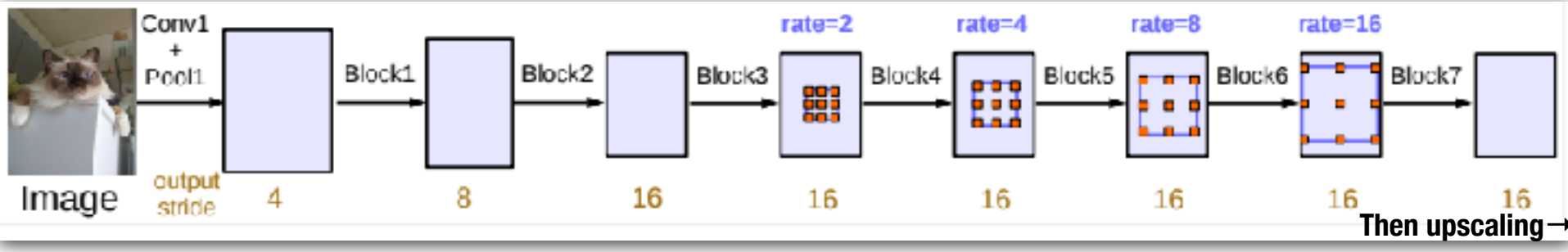


# Some Examples

**Pyramid Scene Parsing Network (PSPNet)**



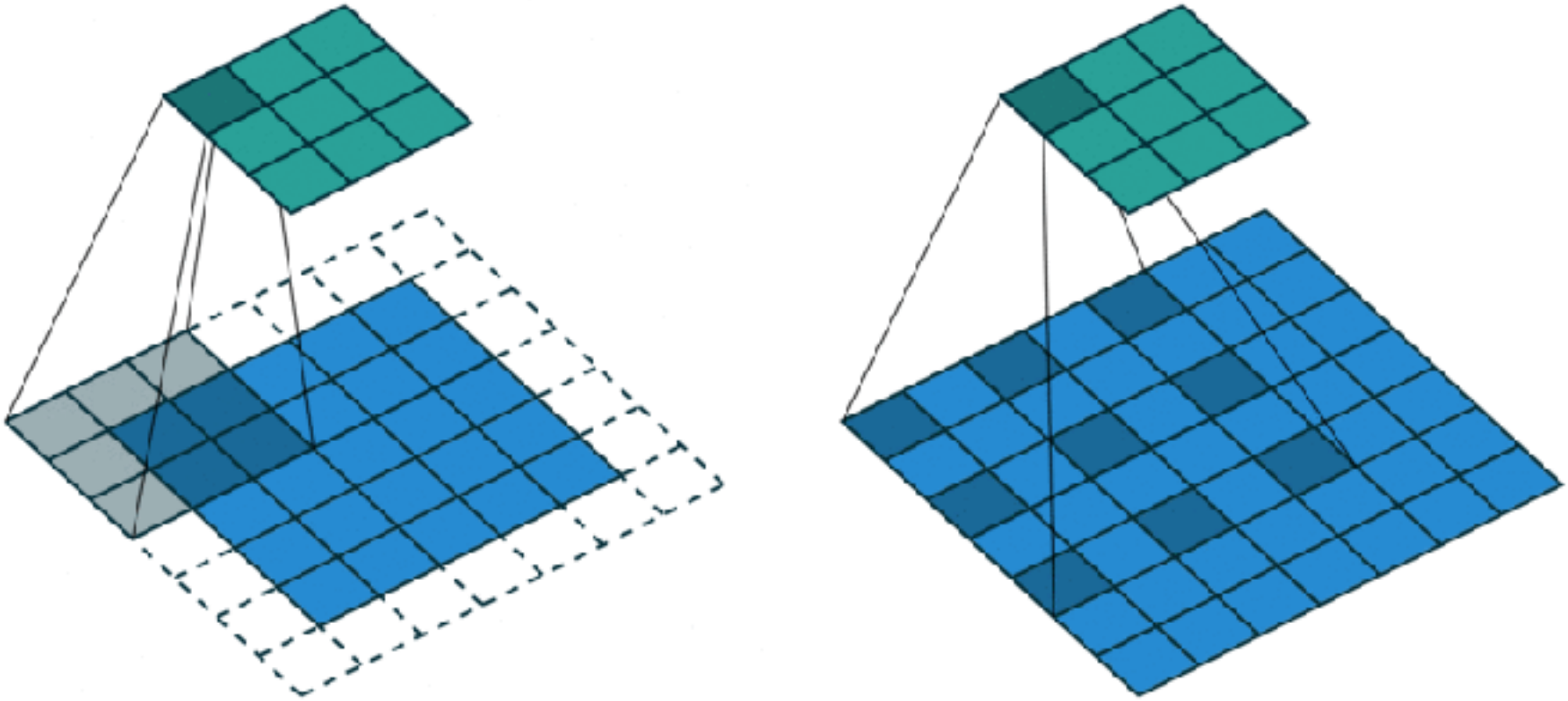

**DeepLabV3: Dilated Convolutions (Atrous Convolutions)**




https://towardsdatascience.com/semantic-segmentation-with-deep-learning-a-guide-and-code-e52fc8958823



# Dilated Convolution

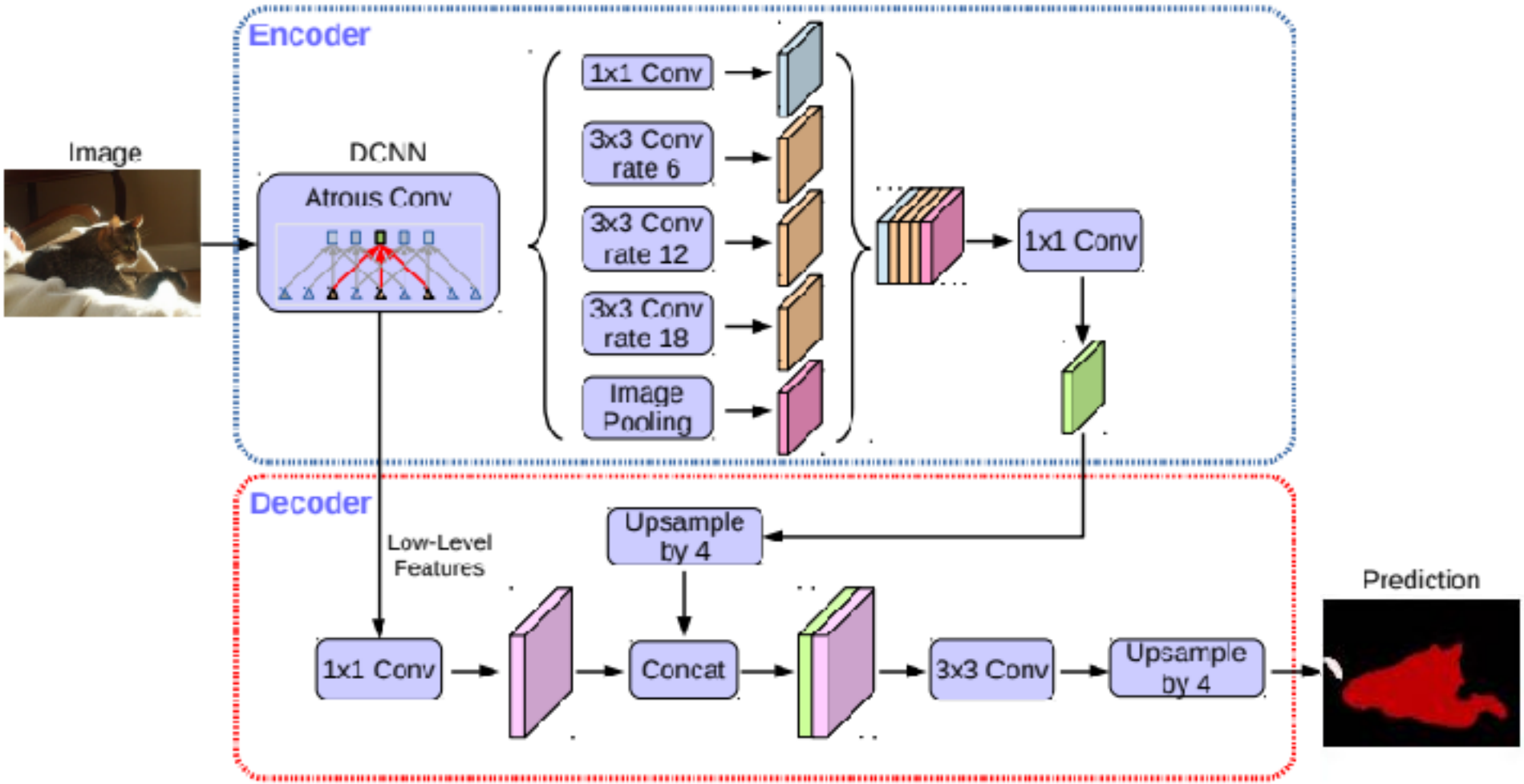


Outputs of convolution are the same size, except for edge effects!

https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5



# DeepLabV3+




https://towardsdatascience.com/semantic-segmentation-with-deep-learning-a-guide-and-code-e52fc8958823



Lecture Notes for

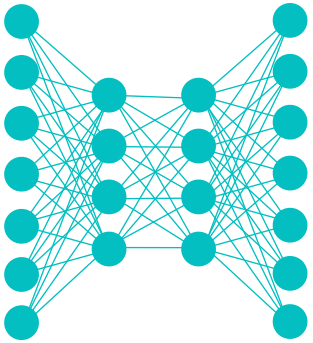
# Neural Networks and Machine Learning

FCN Learning

**Next Time:**

Fully Convolutional Objects

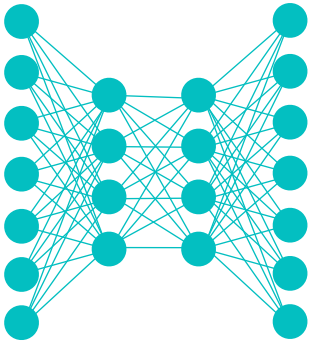
**Reading:** None



Lecture Notes for

# Neural Networks and Machine Learning

## Fully Convolutional Learning

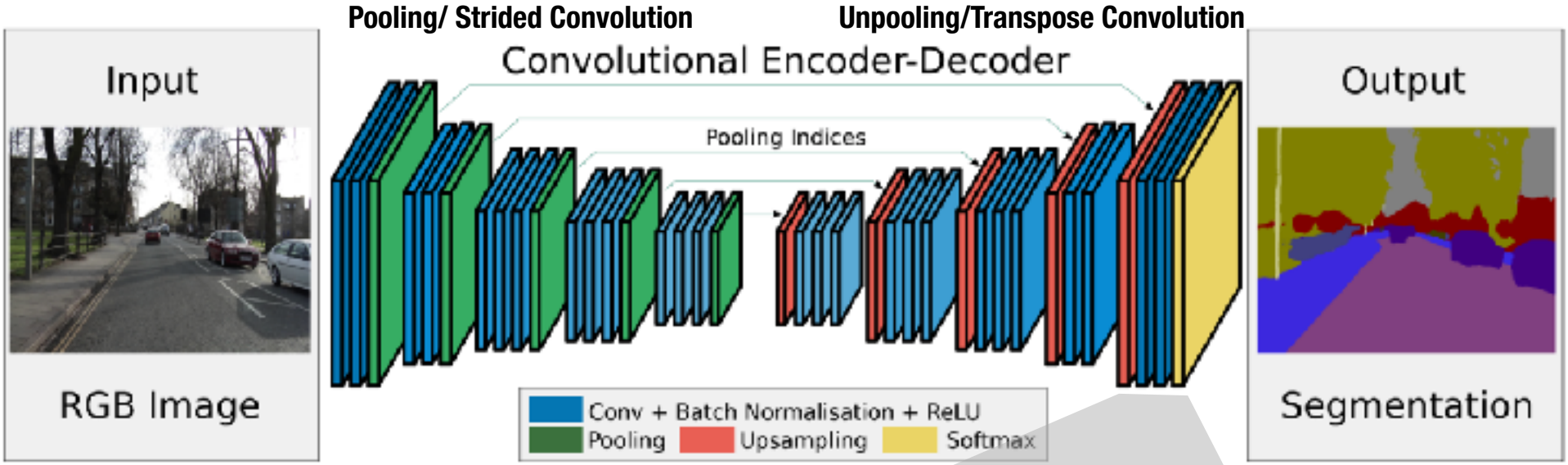


# Logistics and Agenda

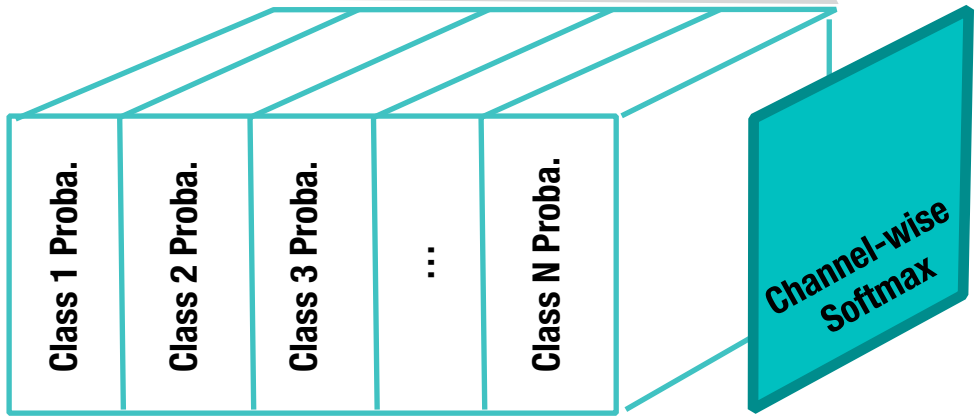
- Logistics
  - Lab One Due over the Weekend
- Agenda
  - Segmentation
    - Semantic (last time)
    - Object (this time)
    - Instance (this time, probably)



# Last Time




**Self Test:**
Does it change the architecture if the Image input size changes?




SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation

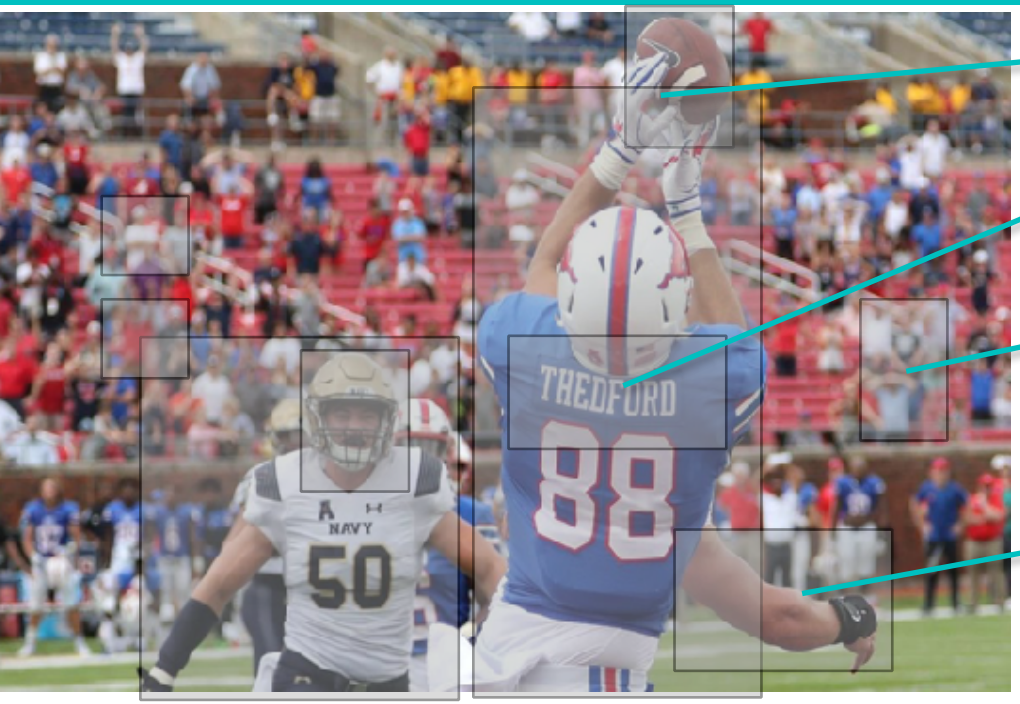


# Object Segmentation

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

**Research!**

# 2014: R-CNN



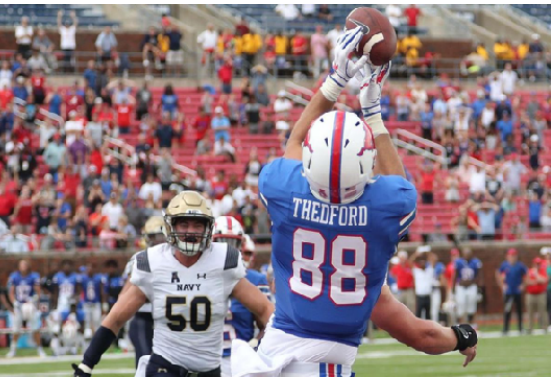


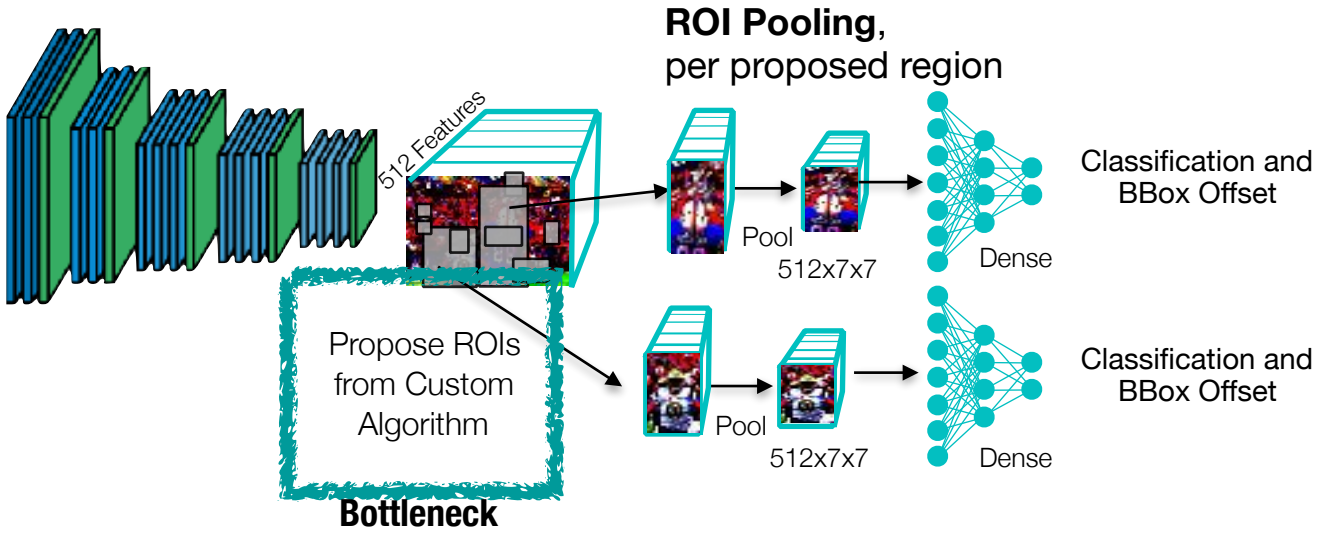
- Too Slow to Be Useful
- SVM and BBox Regression Trained Separately
- Fine Tuned Existing ConvNet (for Warped Images)
- ~50 Seconds per Image when Deployed

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



# 2015: Fast R-CNN



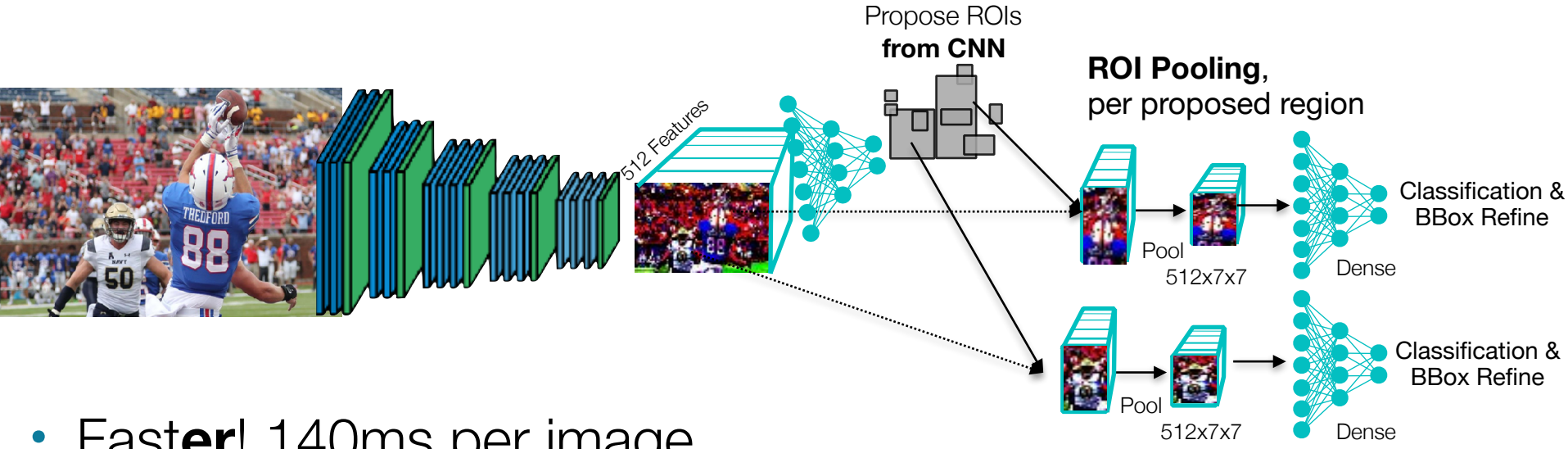




- Fast! 2.3 seconds per image (not ~50)
- But still not real time…

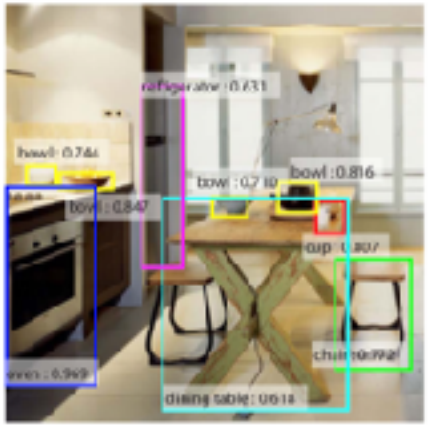
Girshick et al. Fast R-CNN, 2015, February



# 2015: Faster R-CNN

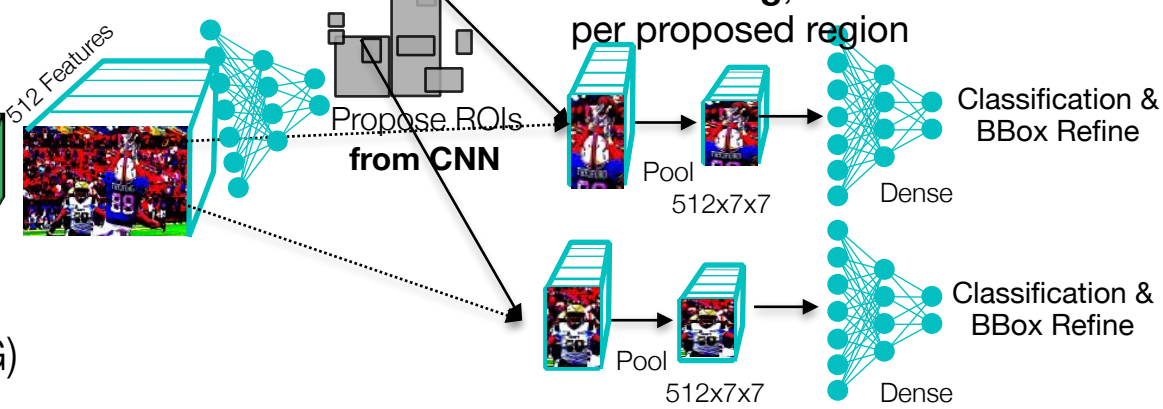



- Fast**er**! 140ms per image (7 FPS)
- Highly Accurate

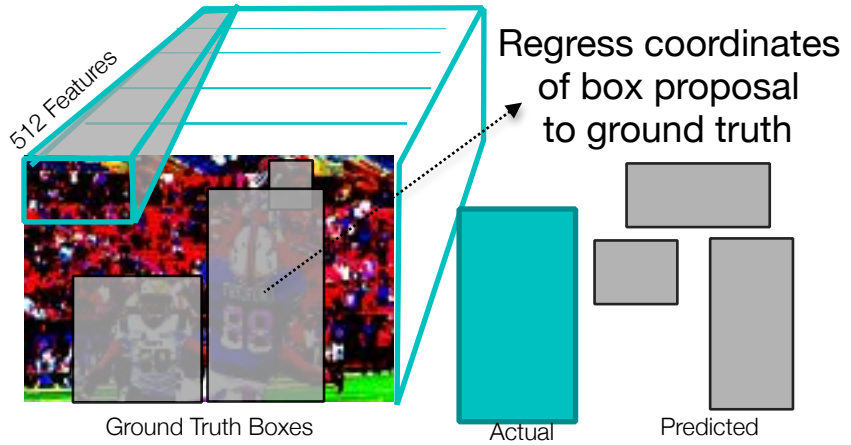


Ren et al. Faster R-CNN, 2015, November

# 2015: Faster R-CNN, Training




- Pre-train image classifier (like VGG)
- Train region proposal network
  - Sliding window across final CNN activations from pre-trained classifier
  - Regress multiple bounding boxes per proposal
- 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




$$l_{box} = \sum_i \mathbf{1}_i \left[ (x - \hat{x}_i)^2 + (y - \hat{y}_i)^2 + (\log w - \log \hat{w}_i)^2 + (\log h - \log \hat{h}_i)^2 \right]$$

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

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

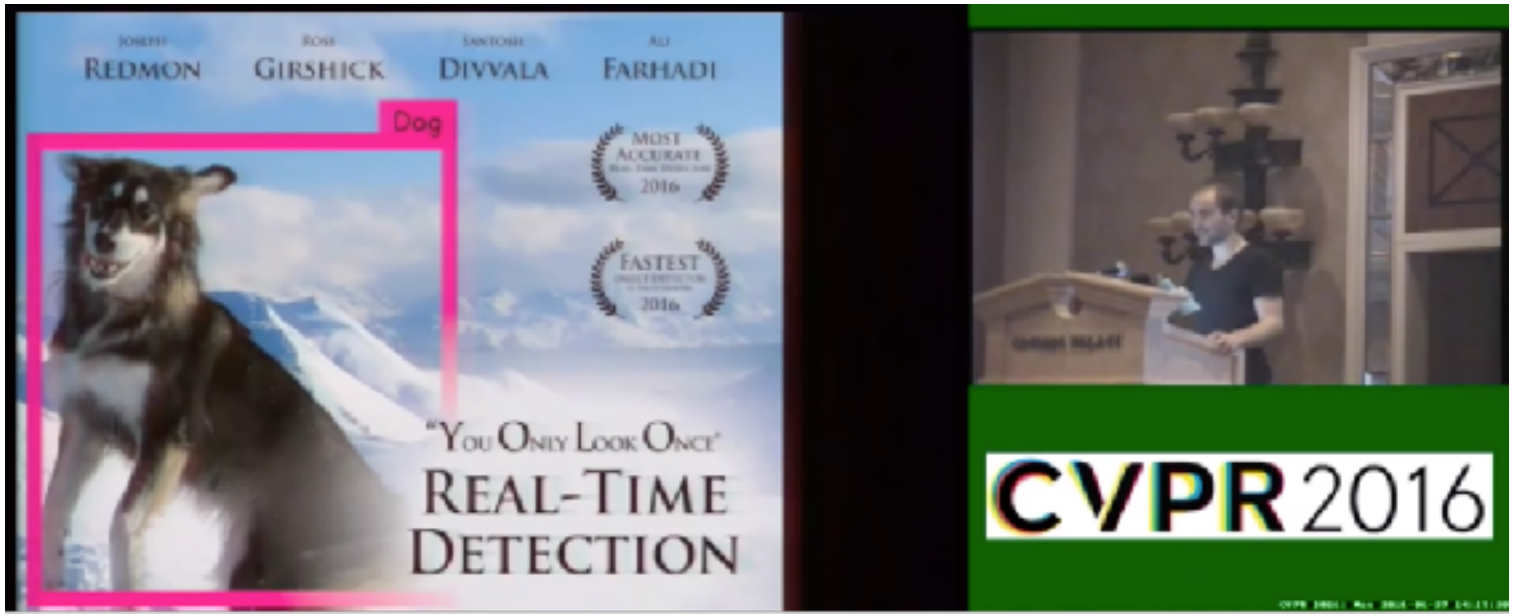


# In Parallel…

- Some of the same researchers started working on a similar, but radically different output approach

# Don’t want to listen to me explain it?

- Check out Joseph Redmon’s Talk at CVPR 2016:
  - https://www.youtube.com/watch?v=NM6lrxy0bxs
  - This is how you give a Technical presentation, plus he is from the school where I did graduate school… so he is clearly superior by grace of proximity…



# 2018: Everybody has a Gimmick

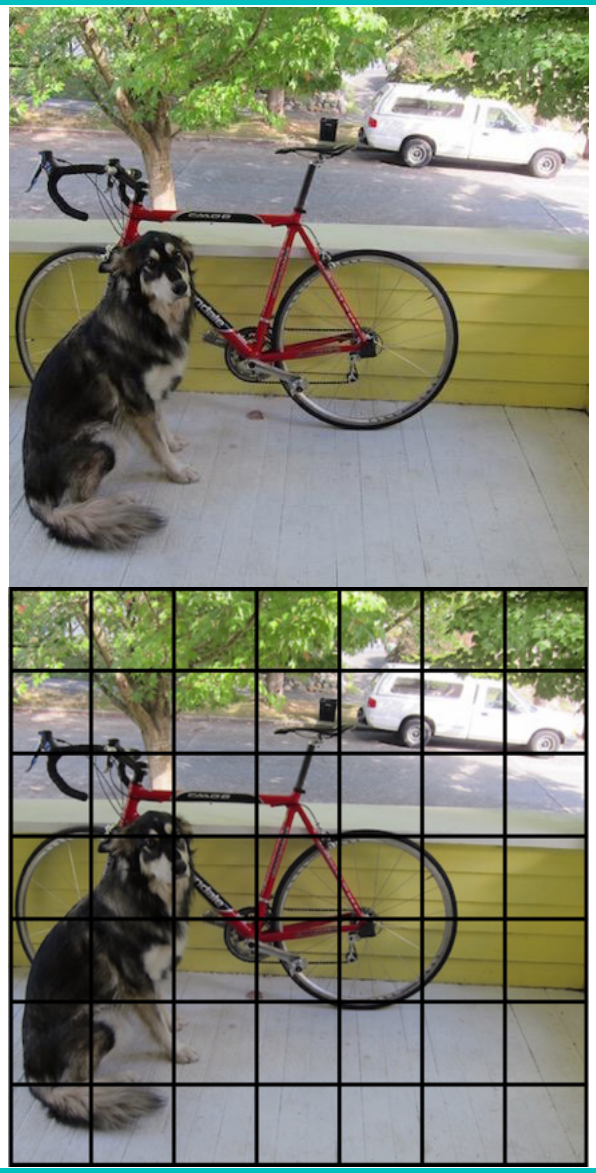

- YOLO ~40-60 FPS
- Slightly more Accurate than **Faster R-CNN**

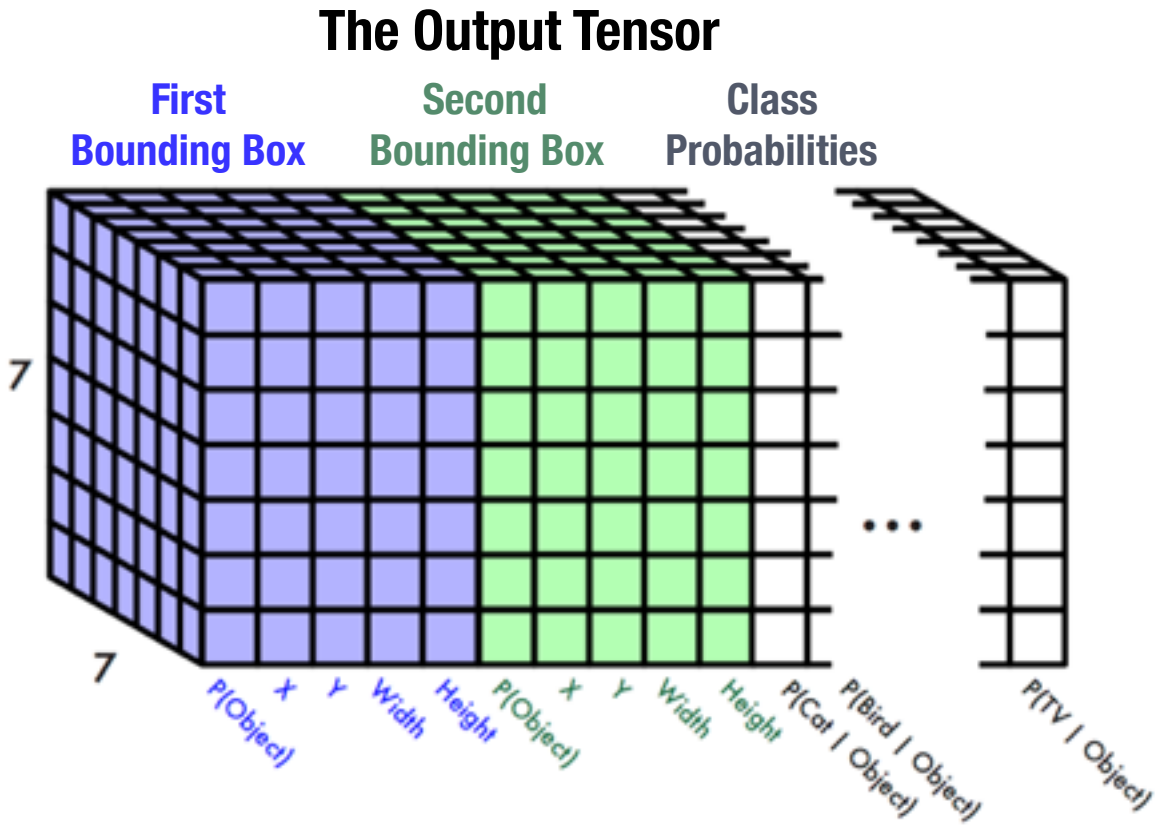
**YOLO9000:**
**Better, Faster, Stronger**

Joseph Redmon*†, Ali Farhadi*†
University of Washington*, Allen Institute for AI†
http://pjreddie.com/yolo9000/

Redmon and Farhadi, YOLO9000: Better, Faster, Stronger, 2016, December 25 — *Merry Christmas?*

# The YOLO Output Tensor

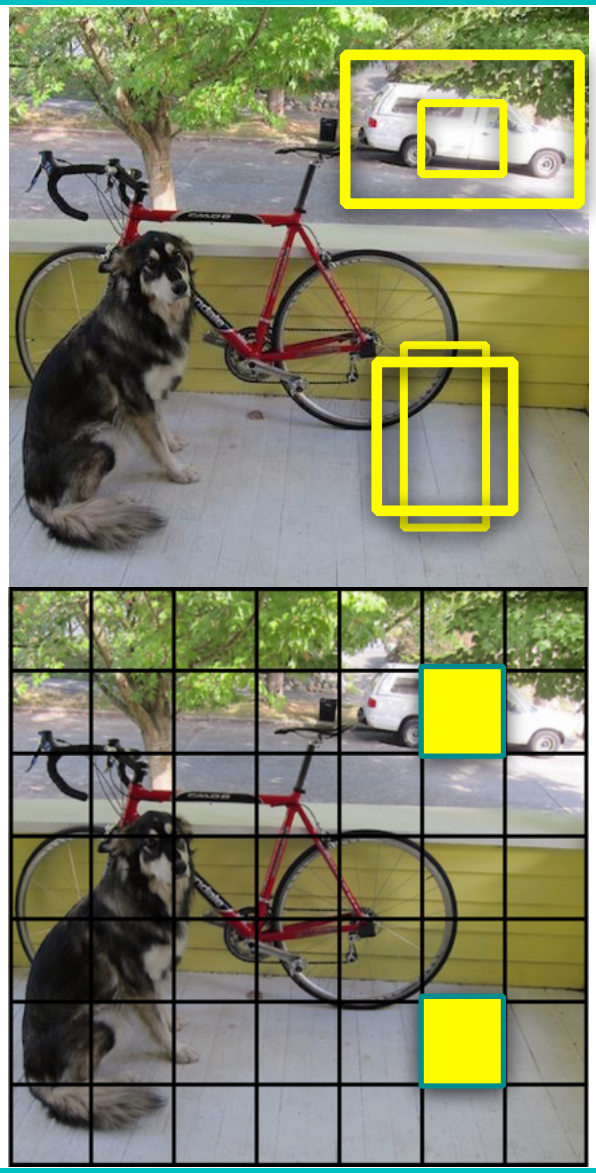


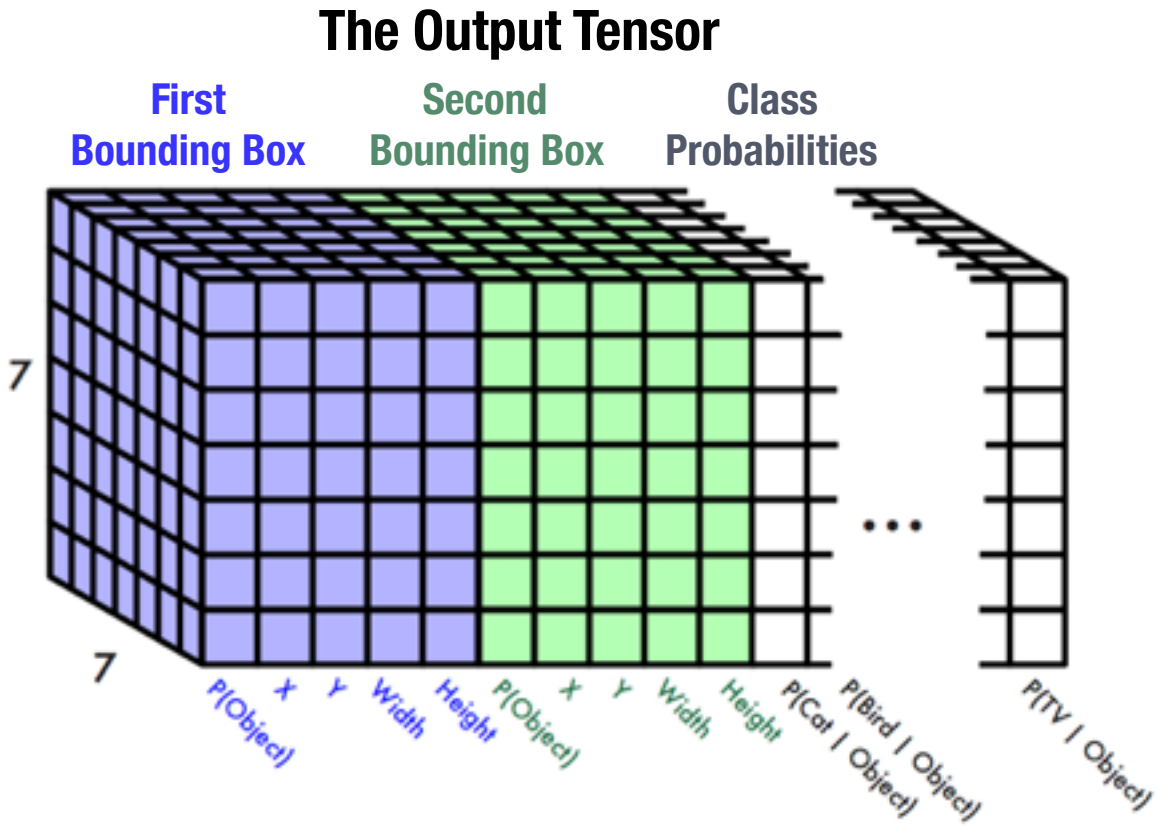



Redmon and Farhadi, YOLO9000: Better, Faster, Stronger, 2016, December 25 — *Merry Christmas?*



# The YOLO Output Tensor

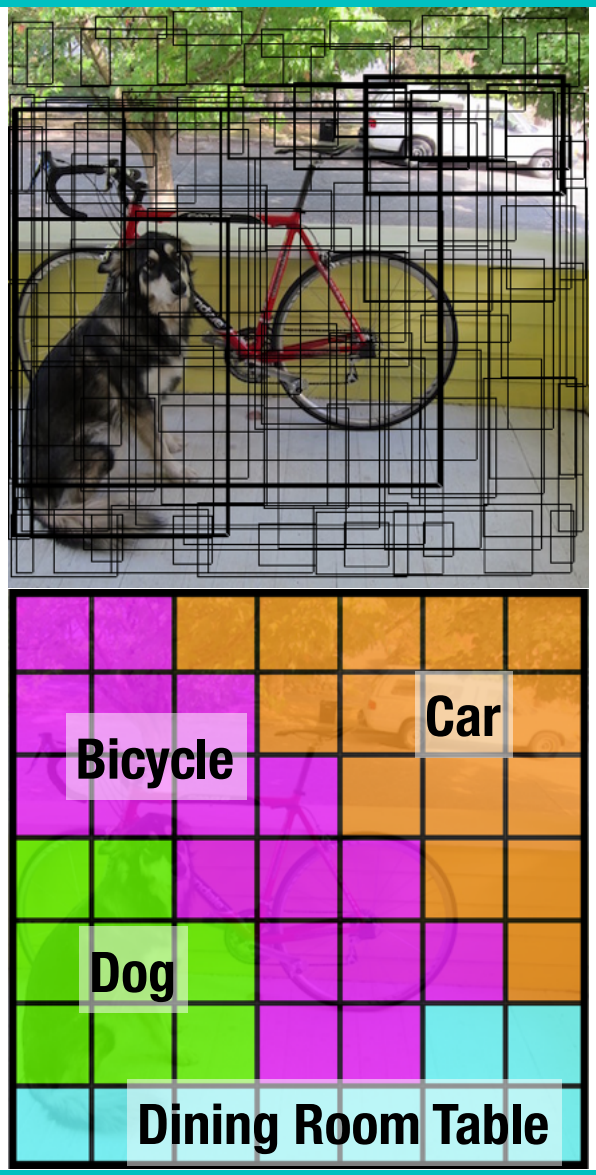


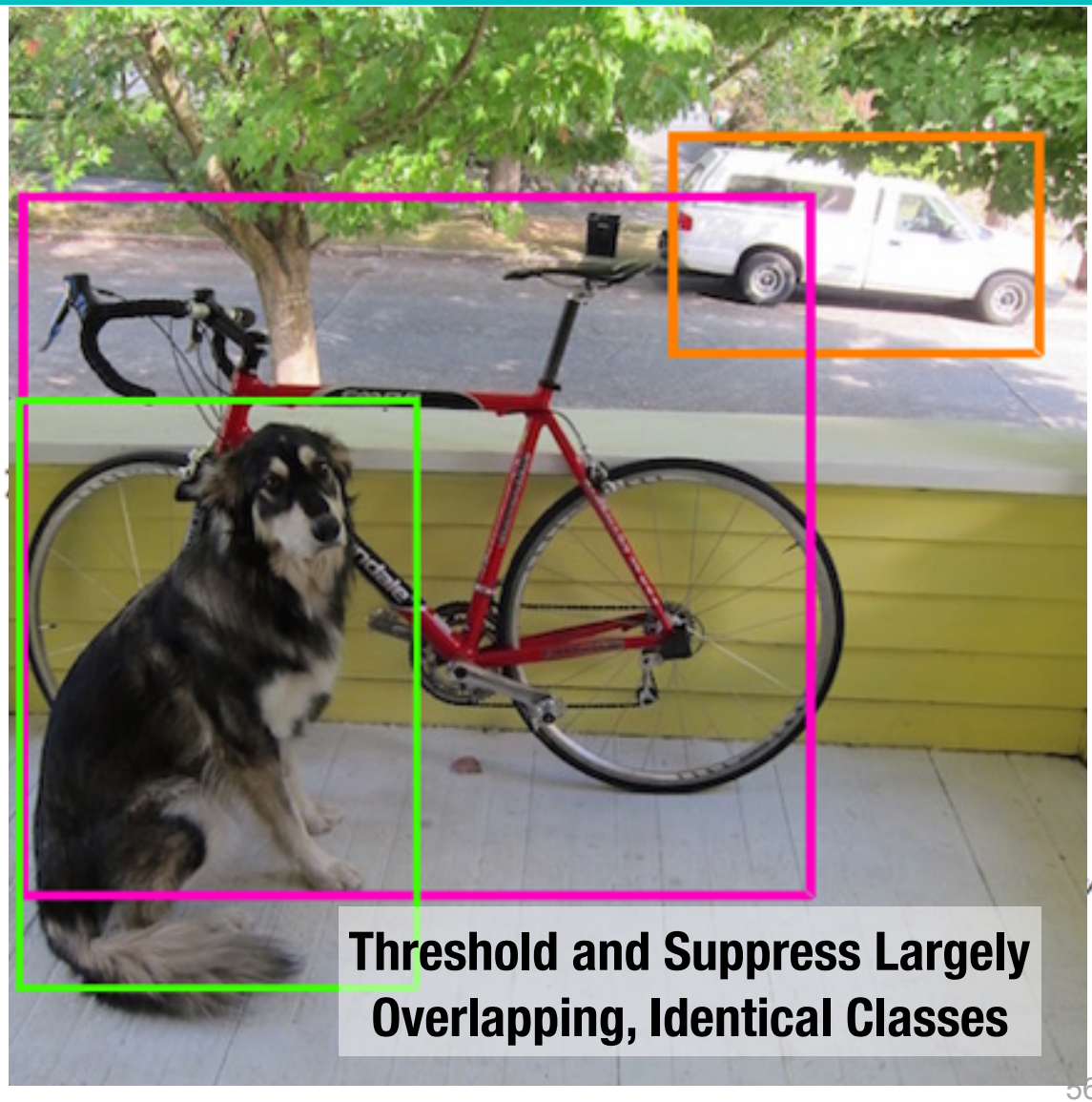



Redmon and Farhadi, YOLO9000: Better, Faster, Stronger, 2016, December 25 — *Merry Christmas?*

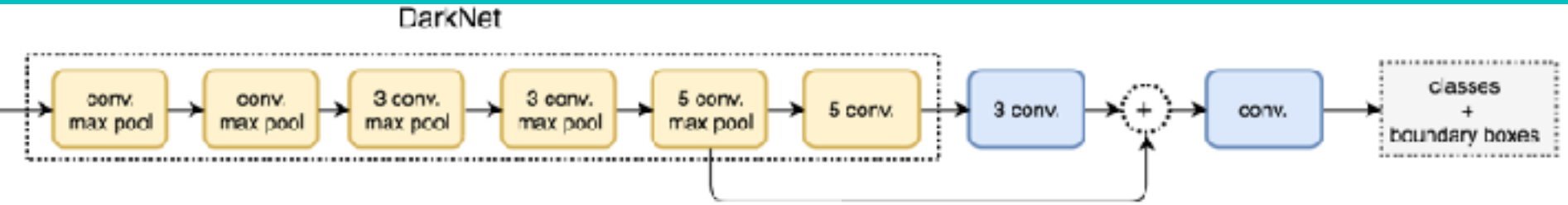


# The YOLO Output Tensor



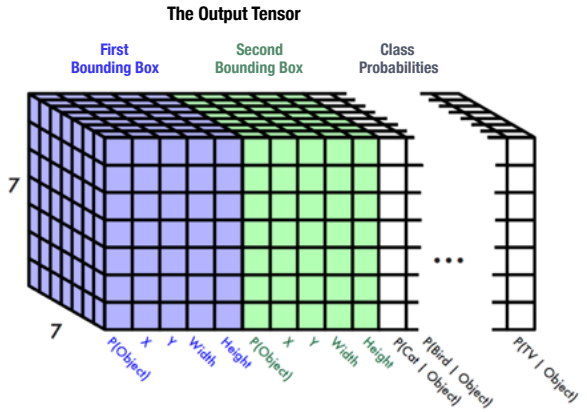



# The YOLO Architecture




**Trained from Traditional Image Dataset.**
**Architecture usually: DarkNet on ImageNet**


**Last layers:**
**Trained on images**
**with bounding boxes**

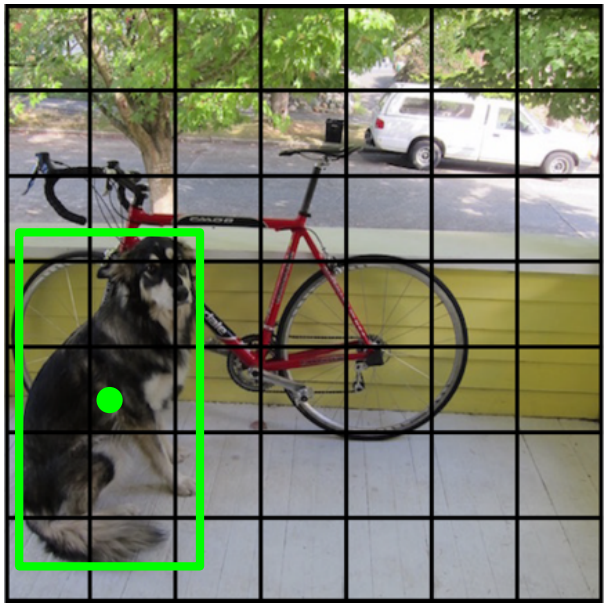



https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

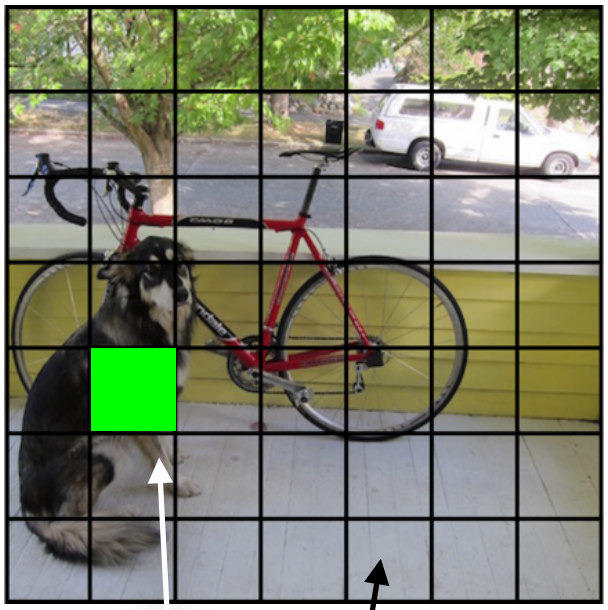


# Training the YOLO Architecture

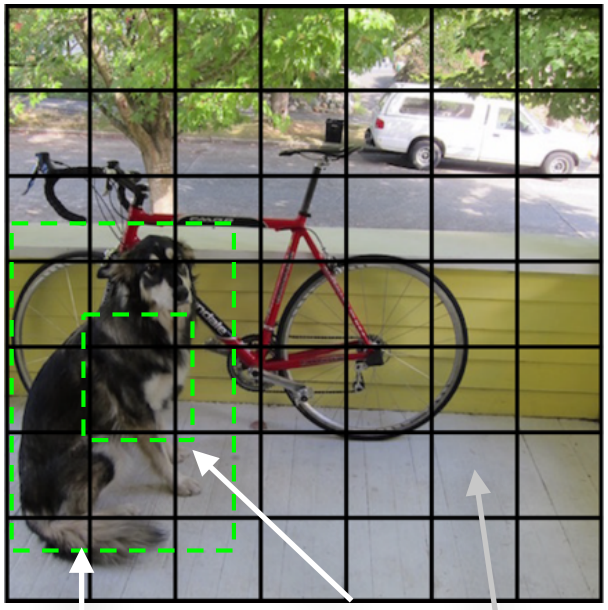
**Ground Truth**



**Class Loss Updated only for Anchor Box**



**Update Bounding Box Confidences**



Dog = 1
Cat = 0
Bike = 0
*etc.*

Don't adjust class confidence if there is no object in GT

Closest to GT
Increase Conf.

Farther From GT
Decrease Conf.

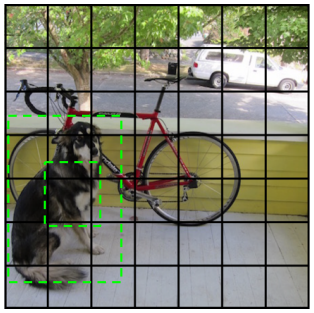
Any anchor box without object, decrease bounding box confidence decrease "objectness" score

Redmon and Farhadi, YOLO9000: Better, Faster, Stronger, 2016, December 25 — *Merry Christmas?*



# The YOLO Loss Function

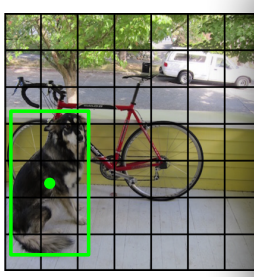
**Update Bounding Box**



$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_{ij})^2 + (y_i - \hat{y}_{ij})^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_{ij}} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_{ij}} \right)^2 \right]$$

Localization Loss



*SxS* cells, $i^{th}$ cell

*B* boxes per cell, j$^{th}$ box

$\mathbb{1}^{\text{obj}}$ indicator function, from GT

$\hat{C}$ is confidence per box

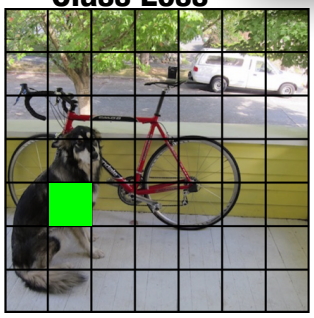
$\hat{p}(c)$ softmax output, per class

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left( C_i - \hat{C}_{ij} \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left( C_i - \hat{C}_{ij} \right)^2$$

Object Detection Loss

**Class Loss**



$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left( p_i(c) - \hat{p}_i(c) \right)^2$$

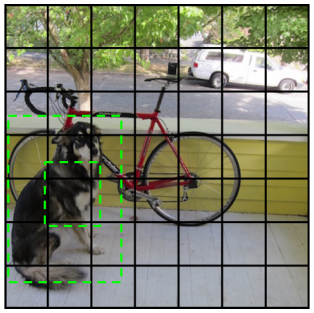
Classification Loss

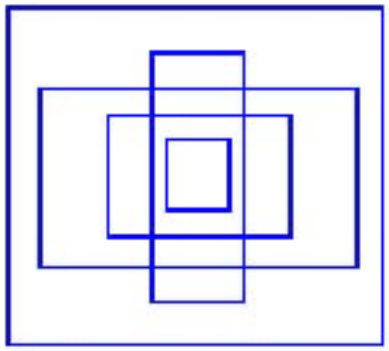
https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088



# Updated YOLO Localization

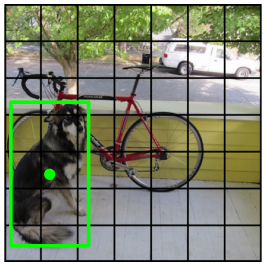
**Update Bounding Box**



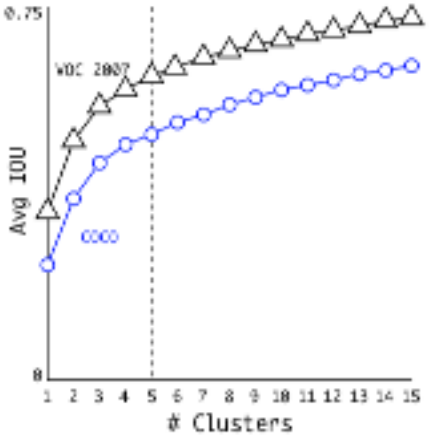


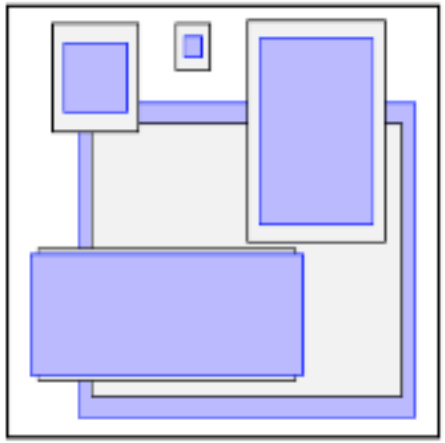
- Define 5 pre-defined box shapes
- Regress $x$, $y$ offset from cell center
- Bound $x$ and $y$ to the bounds of the cell
- And $w$ and $h$ scaling of predefined shape

**Localization Loss**

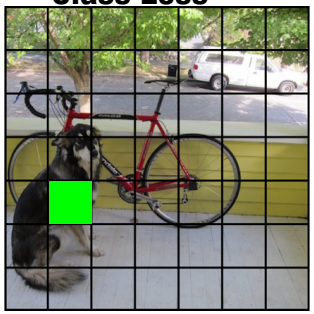


**"Good" Shape Priors Found via Clustering**



**Class Loss**



$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{obj}}\left(C_i-\hat{C}_{ij}\right)^2$$

$$+\lambda_{\text{noobj}}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{noobj}}\left(C_i-\hat{C}_{ij}\right)^2$$

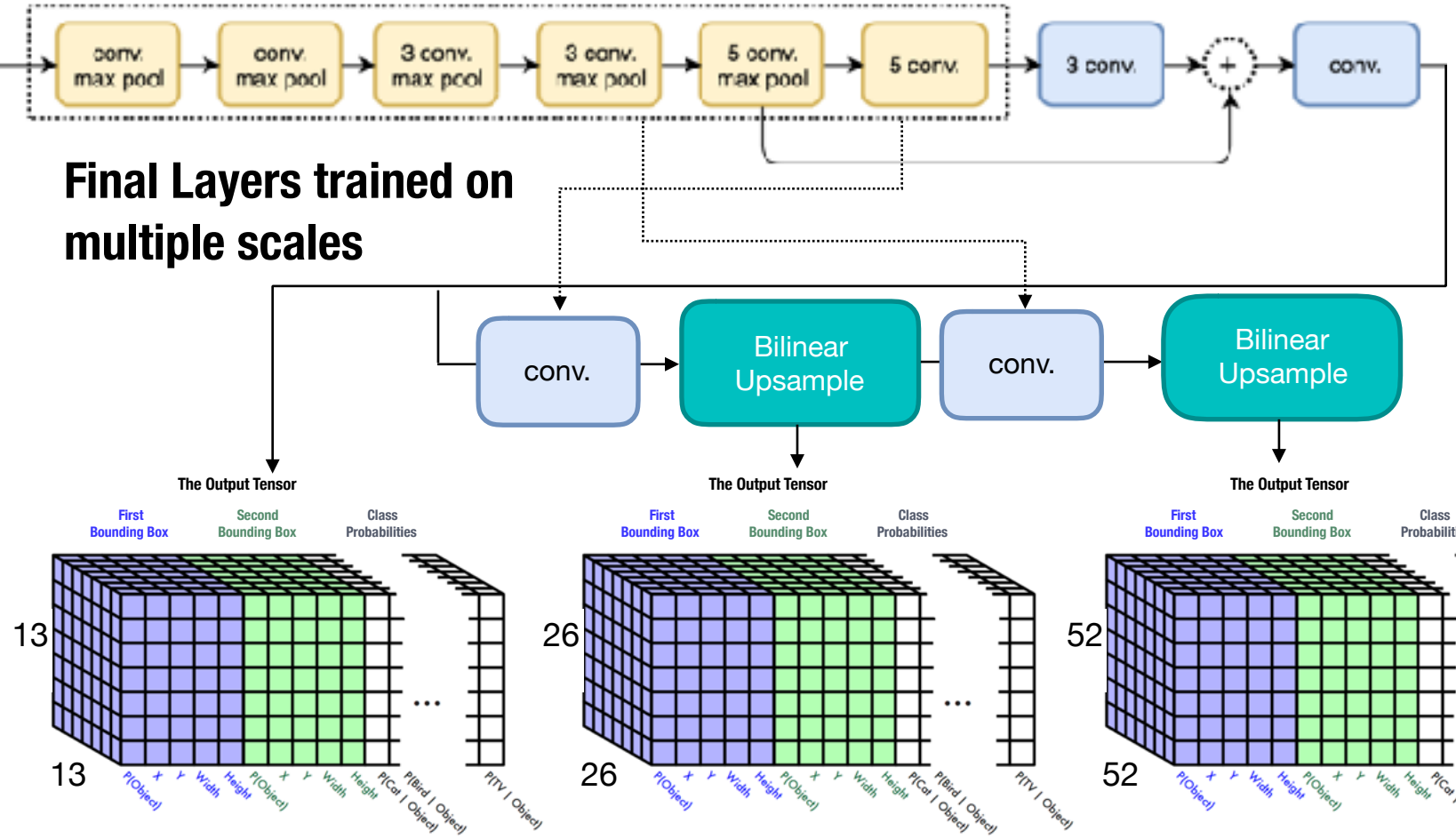
$$+\sum_{i=0}^{S^2}\mathbb{1}_{i}^{\text{obj}}\sum_{c\in\text{classes}}\left(p_i(c)-\hat{p}_i(c)\right)^2$$

**Classification Loss**

https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088



# The YOLOv3 Architecture




https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088



# SSD

- Should we talk about single shot detection?



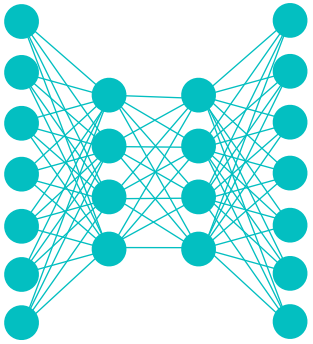
Lecture Notes for

# Neural Networks and Machine Learning

## FCN Learning

**Next Time:**
Instance Segmentation

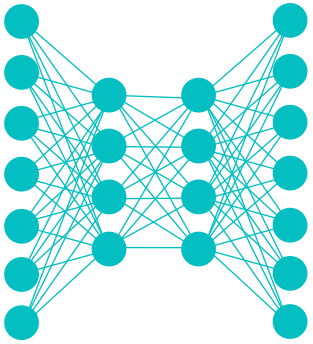
**Reading:** None



Lecture Notes for

# Neural Networks and Machine Learning

## Fully Convolutional Learning

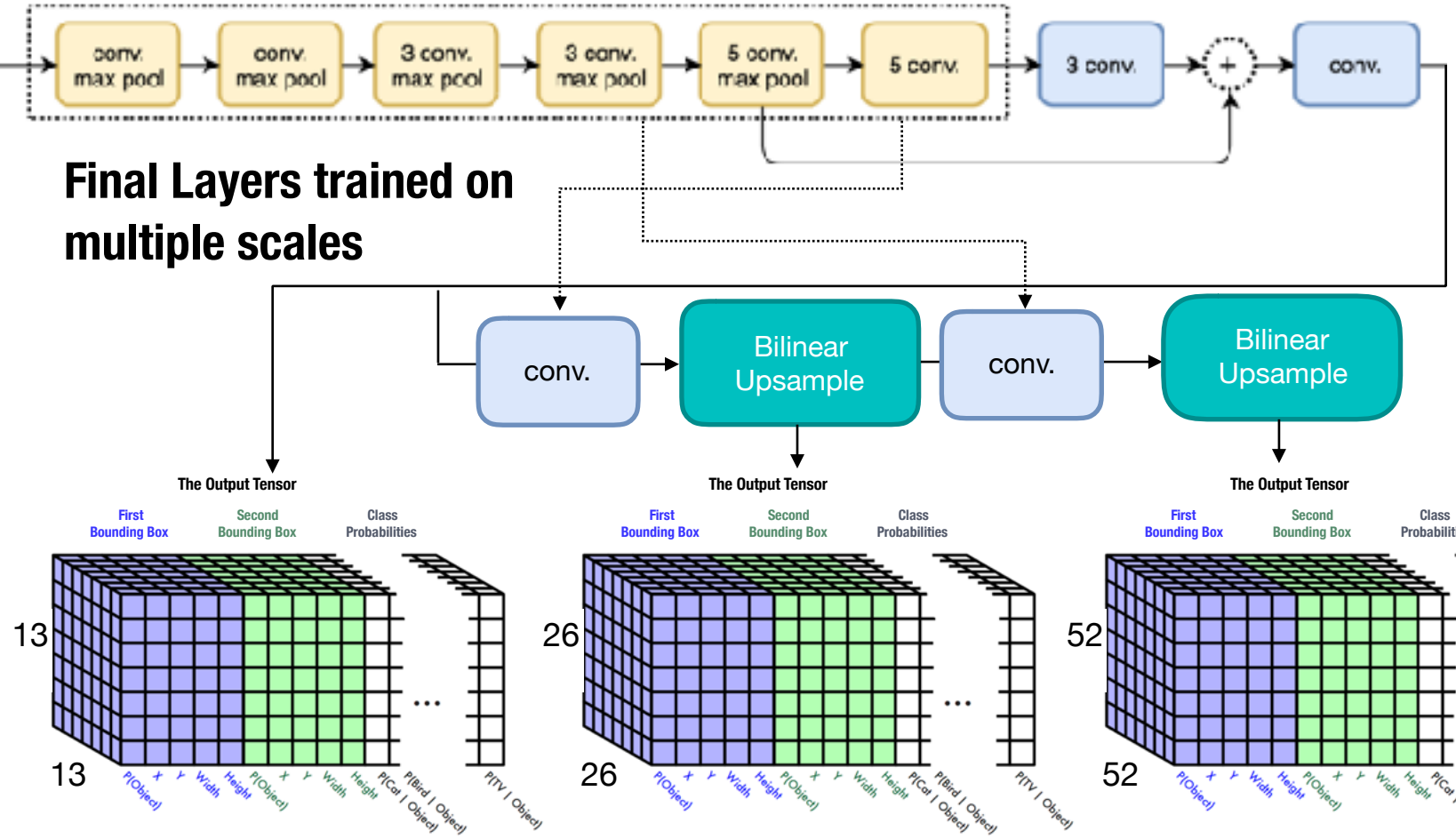


# Logistics and Agenda

- Logistics
  - Lab One was Due over the Weekend
- Agenda
  - Segmentation
    - Semantic (last last time)
    - Object (last time)
    - Instance (this time)
    - Paper Presentation
    - YOLO Demo



# Last Time: The YOLOv3 Architecture




https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088



# A closing thought from YOLOv3 Report


**Abstract**

*We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 $AP_{50}$ in 51 ms on a Titan X, compared to 57.5 $AP_{50}$ in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at* https://pjreddie.com/yolo/.


But maybe a better question is: "What are we going to do with these detectors now that we have them?" A lot of the people doing this research are at Google and Facebook. I guess at least we know the technology is in good hands and definitely won't be used to harvest your personal information and sell it to.... wait, you're saying that's exactly what it will be used for?? Oh.

Well the other people heavily funding vision research are the military and they've never done anything horrible like killing lots of people with new technology oh wait.....[1]

I have a lot of hope that most of the people using computer vision are just doing happy, good stuff with it, like counting the number of zebras in a national park [13], or tracking their cat as it wanders around their house [19]. But computer vision is already being put to questionable use and as researchers we have a responsibility to at least consider the harm our work might be doing and think of ways to mitigate it. We owe the world that much.

In closing, do not @ me. (Because I finally quit Twitter).


[1] The author is funded by the Office of Naval Research and Google.


Redmon and Farhadi, YOLOv3: An incremental Improvement



# A closing thought from YOLOv3 Report

## The Rebuttal I wish I could Write:

Reviewer #2 AKA Dan Grossman (lol blinding who does that) insists that I point out here that our graphs have not one but two non-zero origins. You're absolutely right Dan, that's because it looks way better than admitting to ourselves that we're all just here battling over 2-3% mAP. But here are the requested graphs. I threw in one with FPS too because we look just like super good when we plot on FPS.

Reviewer #4 AKA JudasAdventus on Reddit writes "Entertaining read but the arguments against the MSCOCO metrics seem a bit weak". Well, I always knew you would be the one to turn on me Judas. You know how when you work on a project and it only comes out alright so you have to figure out some way to justify how what you did actually was pretty cool? I was basically trying to do that and I lashed out at the COCO metrics a little bit. But now that I've staked out this hill I may as well die on it.

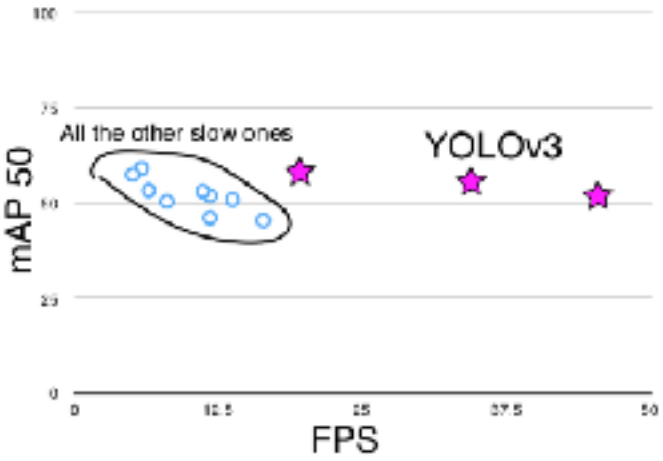



Figure 5. These two hypothetical detectors are perfect according to mAP over these two images. They are both perfect. Totally equal.

Now this is OBVIOUSLY an over-exaggeration of the problems with mAP but I guess my newly reteconned point is that there are such obvious discrepancies between what people in the "real world" would care about and our current metrics that I think if we're going to come up with new metrics we should focus on these discrepancies. Also, like, it's already mean average precision, what do we even call the COCO metric, average mean average precision?

Here's a proposal, what people actually care about is given an image and a detector, how well will the detector find and classify objects in the image. What about getting rid of the per-class AP and just doing a global average precision? Or doing an AP calcu-

Boxes are stupid anyway though, I'm probably a true believer in masks except I can't get YOLO to learn them.



# Paper Presentation

## Do we need pre-training?

**Rethinking ImageNet Pre-training**

Kaiming He Ross Girshick Piotr Dollár

Facebook AI Research (FAIR)

**Abstract**

*We report competitive results on object detection and instance segmentation on the COCO dataset using standard models trained* ***from random initialization****. The results are* ***no worse*** *than their ImageNet pre-training counterparts even when using the hyper-parameters of the baseline system (Mask R-CNN) that were optimized for fine-tuning pre-trained models, with the sole exception of increasing the number of training iterations so the randomly initialized models may converge. Training from random initialization is surprisingly robust; our results hold even when: (i) using only 10% of the training data, (ii) for deeper and wider models, and (iii) for multiple tasks and metrics. Experiments show that ImageNet pre-training speeds up convergence early in training, but does not necessarily provide regularization or improve final target task accuracy. To push the envelope we demonstrate 50.9 AP on COCO object detection without using any external data—a result on par with the top COCO 2017 competition results that used ImageNet pre-training. These observations challenge the conventional wisdom of ImageNet pre-training for dependent tasks and we expect these discoveries will encourage people to rethink the current de facto paradigm of 'pre-training and fine-tuning' in computer vision.*

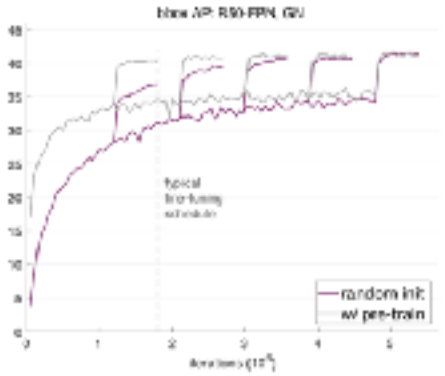
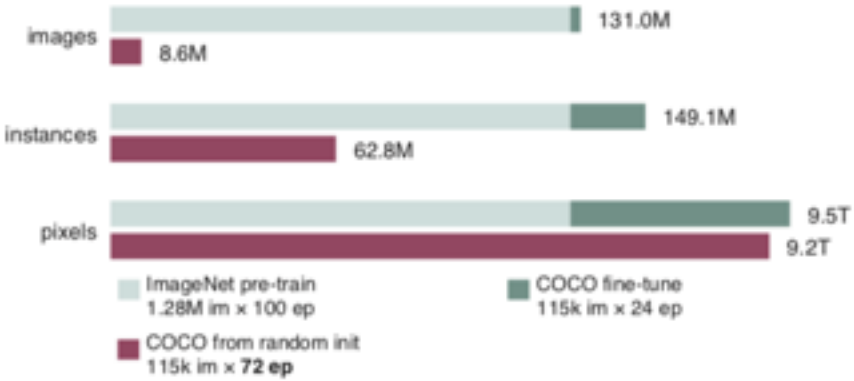
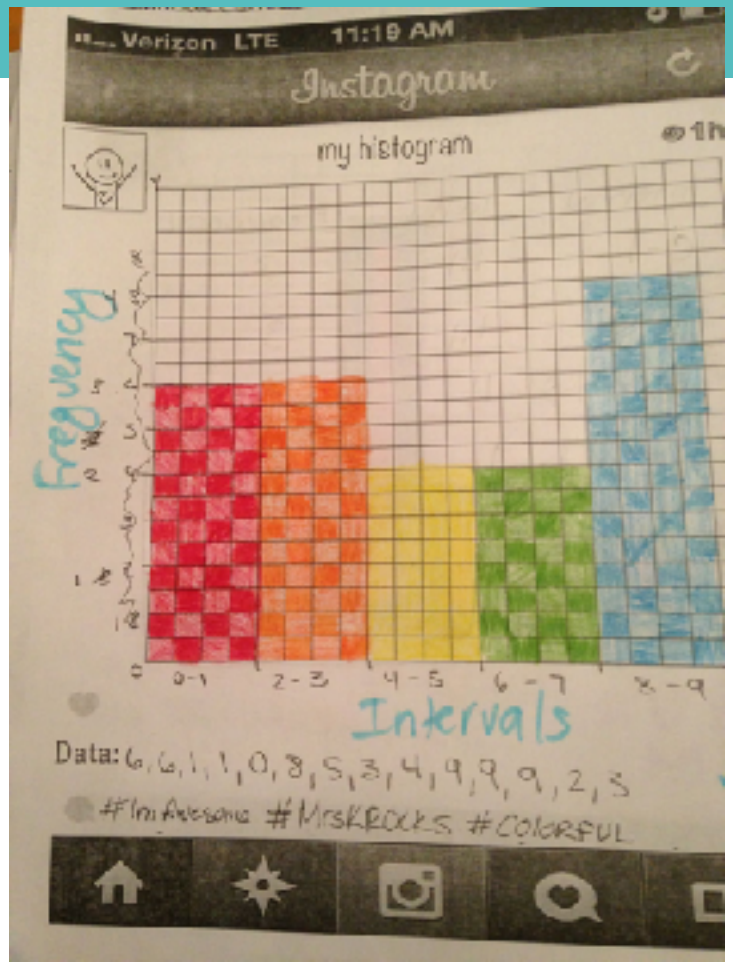


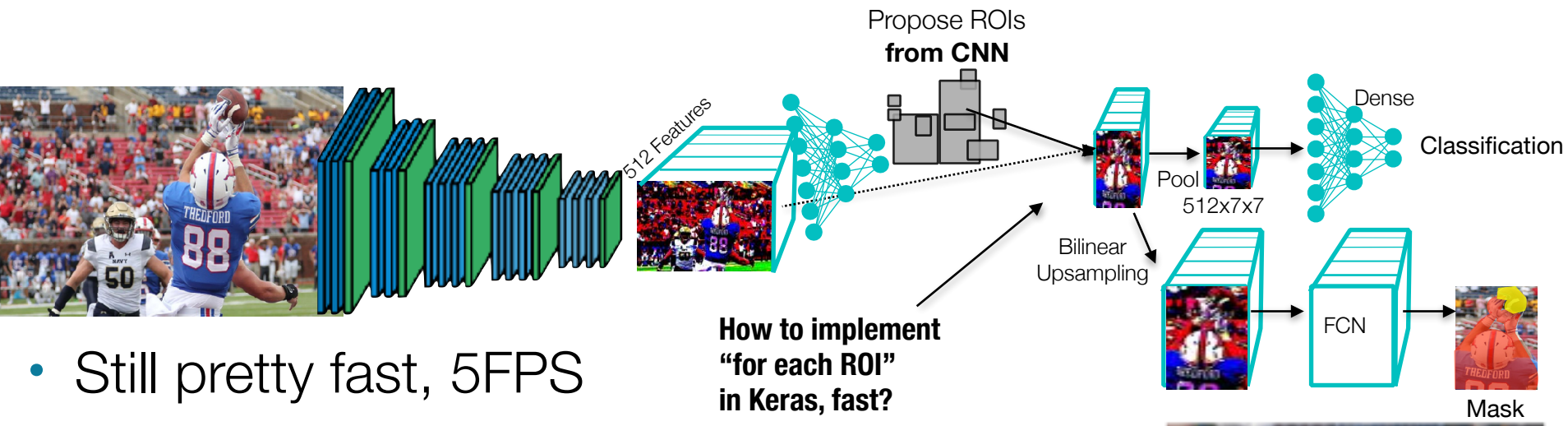

Figure 1. We train Mask R-CNN [13] with a ResNet-50 FPN [26] and GroupNorm [48] backbone on the COCO `train2017` set and evaluate bounding box AP on the `val2017` set, initializing the model by random weights or ImageNet pre-training. We employ different training schedules by varying the iterations at which the learning rate is reduced (where the accuracy leaps). The model trained from random initialization needs more iterations to converge, but converges to a solution that is *no worse* than the fine-tuning counterpart. Table 1 shows the resulting AP numbers.



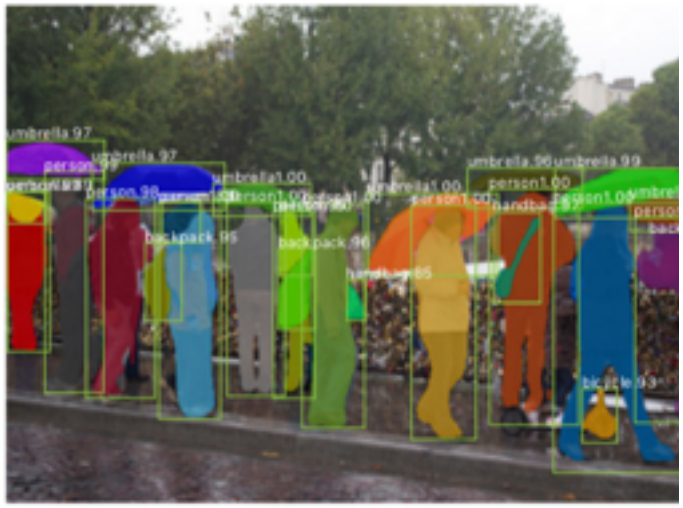
# Instance Segmentation

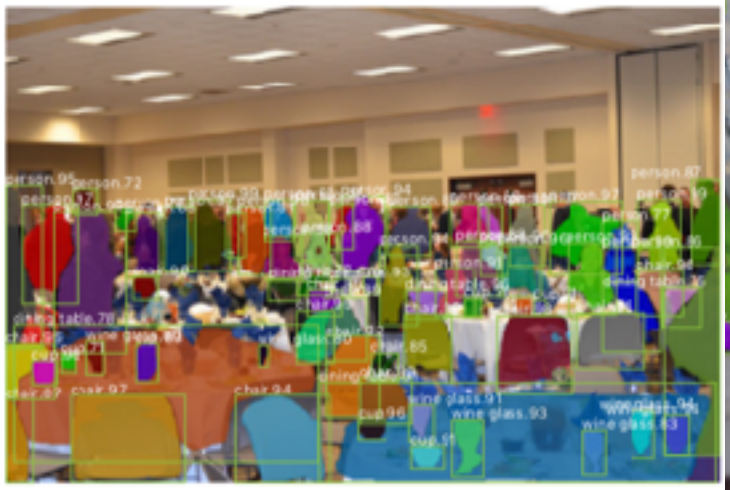



# 2018: Mask R-CNN



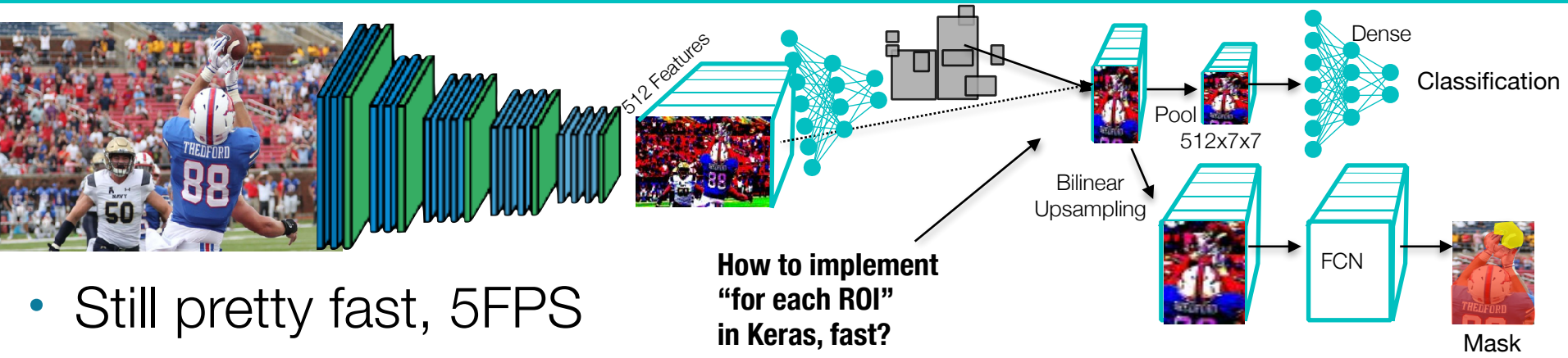

- Still pretty fast, 5FPS
- Excellent results





Ren et al. Mask R-CNN, 2018

# 2018: Mask R-CNN



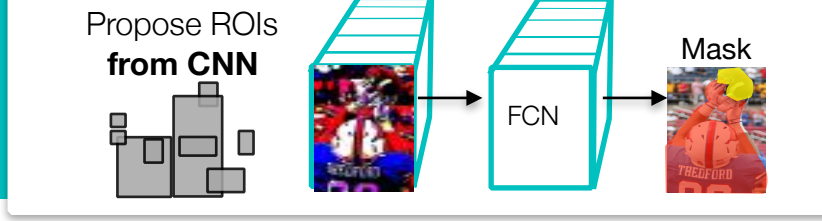

- Still pretty fast, 5FPS
- Excellent results

An Excellent, well documented Implementation here:
https://github.com/matterport/Mask_RCNN

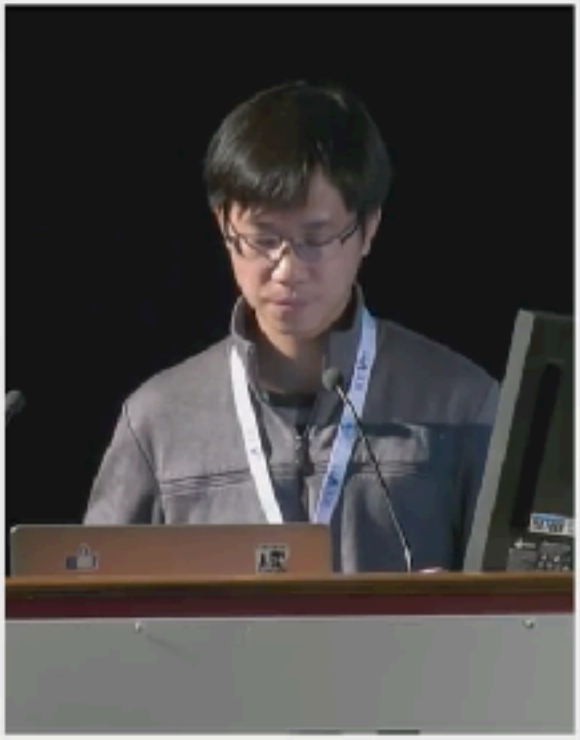
```
# Use shape of first image. Images in a batch must have the same size.
image_shape = parse_image_meta_graph(image_meta)['image_shape'][0]
# Equation 1 in the Feature Pyramid Networks paper. Account for
# the fact that our coordinates are normalized here.
# e.g. a 224x224 ROI (in pixels) maps to P4
image_area = tf.cast(image_shape[0] * image_shape[1], tf.float32)
roi_level = log2_graph(tf.sqrt(h * w) / (224.0 / tf.sqrt(image_area)))
roi_level = tf.minimum(5, tf.maximum(
```

Ren et al. Mask R-CNN, 2018

# 2018: Mask R-CNN

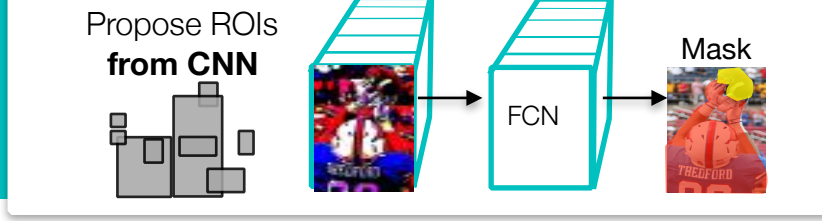



Can also provide **key point detection** from same FCN features (not real time, post processed)




ICCV17

# 2018: Mask R-CNN




- Real time, Mask R-CNN

# Lots of Use Cases for Mask-RCNN

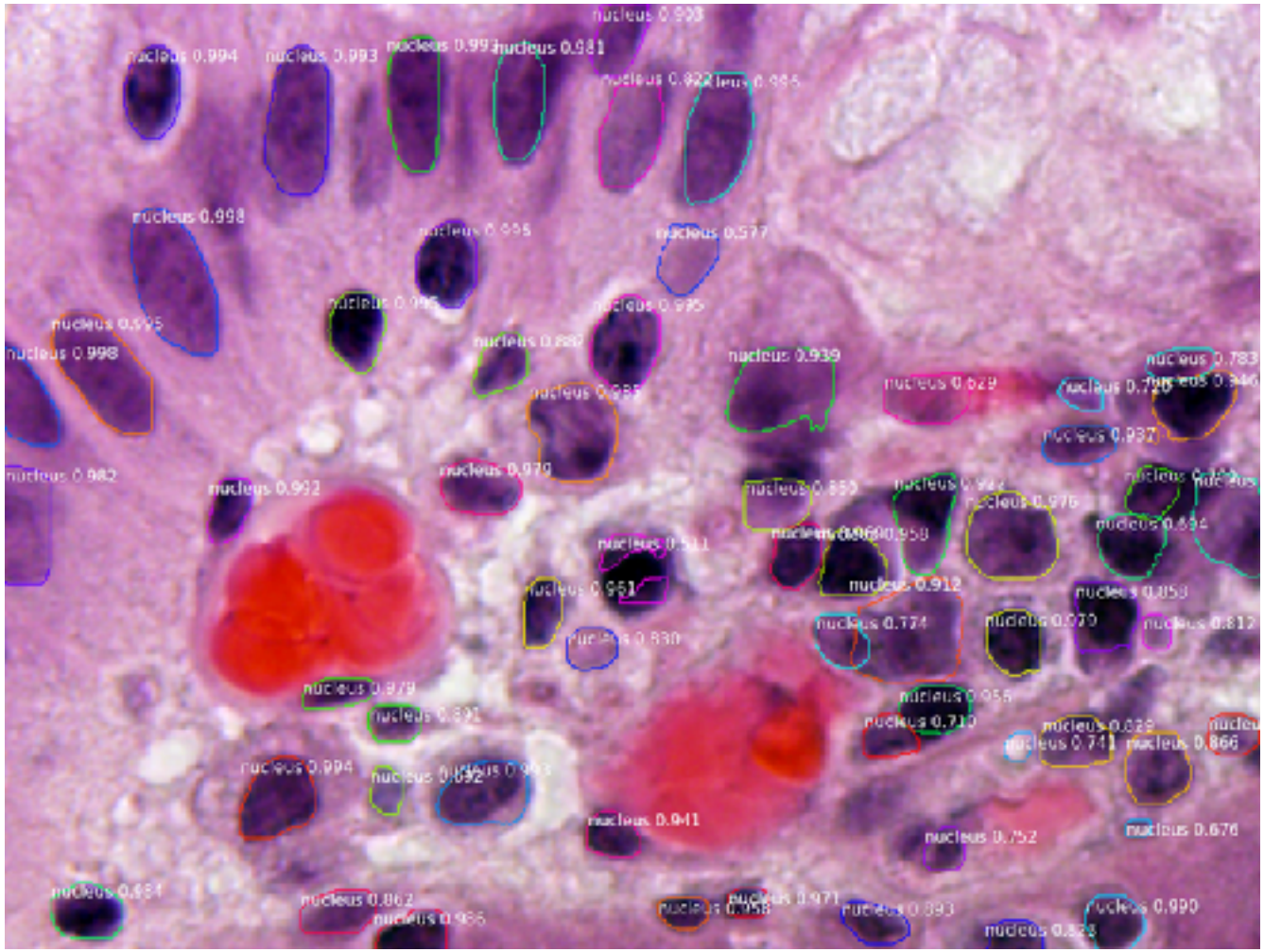
**3D Building Reconstruction (mask become 3D point cloud)**

https://github.com/matterport/Mask_RCNN



# Lots of Use Cases for Mask-RCNN

**Segmenting Nuclei**

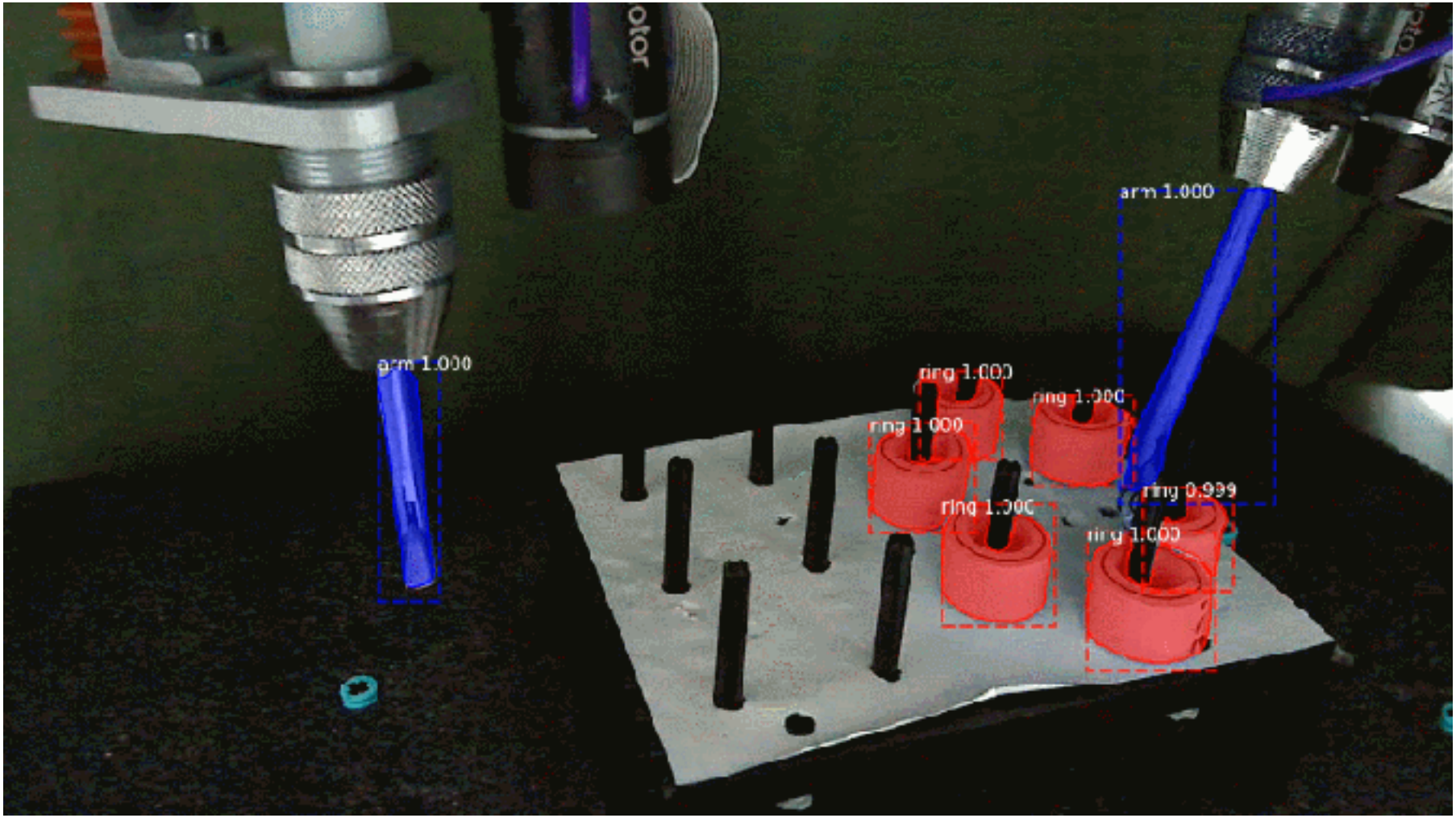


https://github.com/matterport/Mask_RCNN



# Lots of Use Cases for Mask-RCNN

**Robotic Movement (like surgery)**




https://github.com/matterport/Mask_RCNN



# In summary

- Semantic segmentation through FCN is active research area
  - DeepLabV3+ is excellent choice
- Object segmentation is good, ready for use in industry (Apple's ObjectDetector uses YOLO variant)
  - Already deployed in a number of Apps
  - At 60 FPS, supports tracking applications and AR
  - Can backoff to CPU only at about 5 FPS
- Instance Segmentation is ready for deployment in a number of areas, but is still 5 FPS, using a good deal of the hardware



# YOLOv3 in Keras

Excellent implementation with
… not very many comments
Look at “YOLOv3” and “TrainBottleneck”

`Self-Guided:` https://github.com/qqwweee/keras-yolo3

Another Great resource, with Excellent Documentation (PyTorch).
Step-by-Step Layer Construction and Loss Explanation
https://blog.paperspace.com/how-to-implement-a-yolo-v3-object-detector-from-scratch-in-pytorch-part-2/


Lecture Notes for

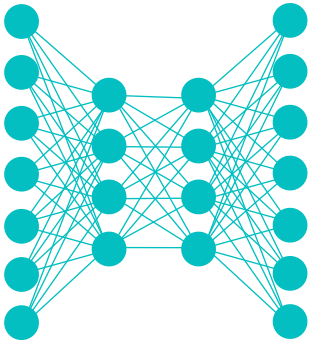
# Neural Networks and Machine Learning

## Fully Convolutional Learning

**Next Time:**

Image Style Transfer

**Reading:** Chollet 8.1– 8.3



# Backup slides



# Title Between Topics


# Example Slide



# Title

Subtitle

`Follow Along: Notebook Name`