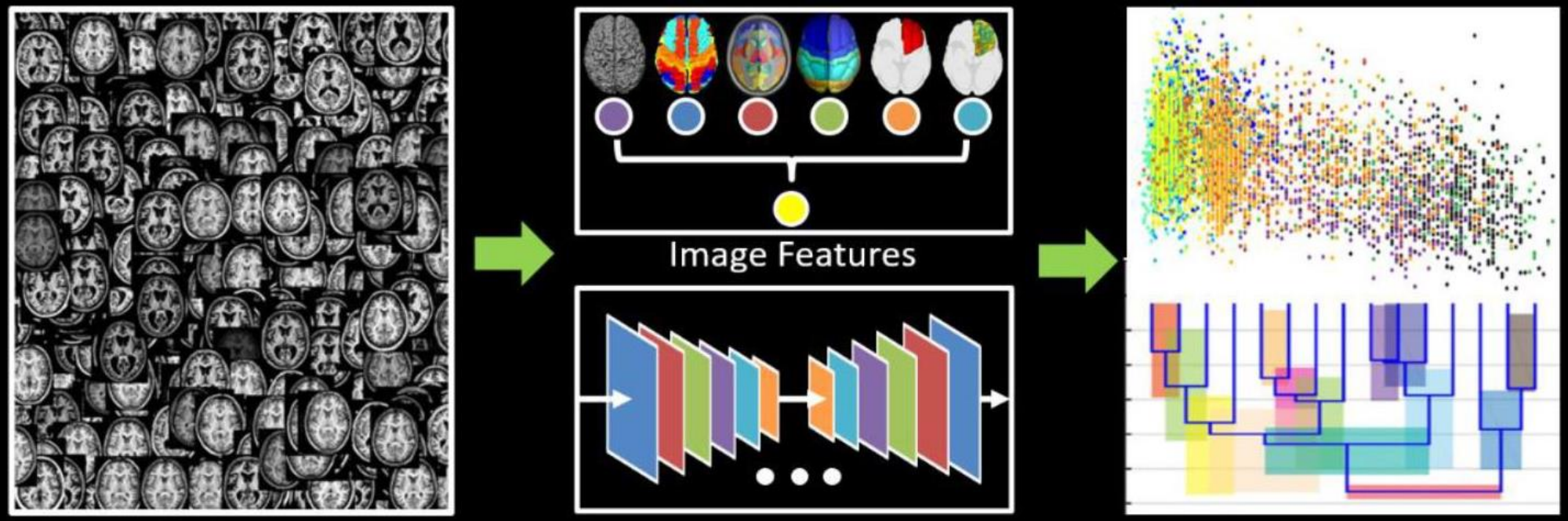


Deep Image Detection

[Spring 2020 CS-8395 Deep Learning in Medical Image Computing]

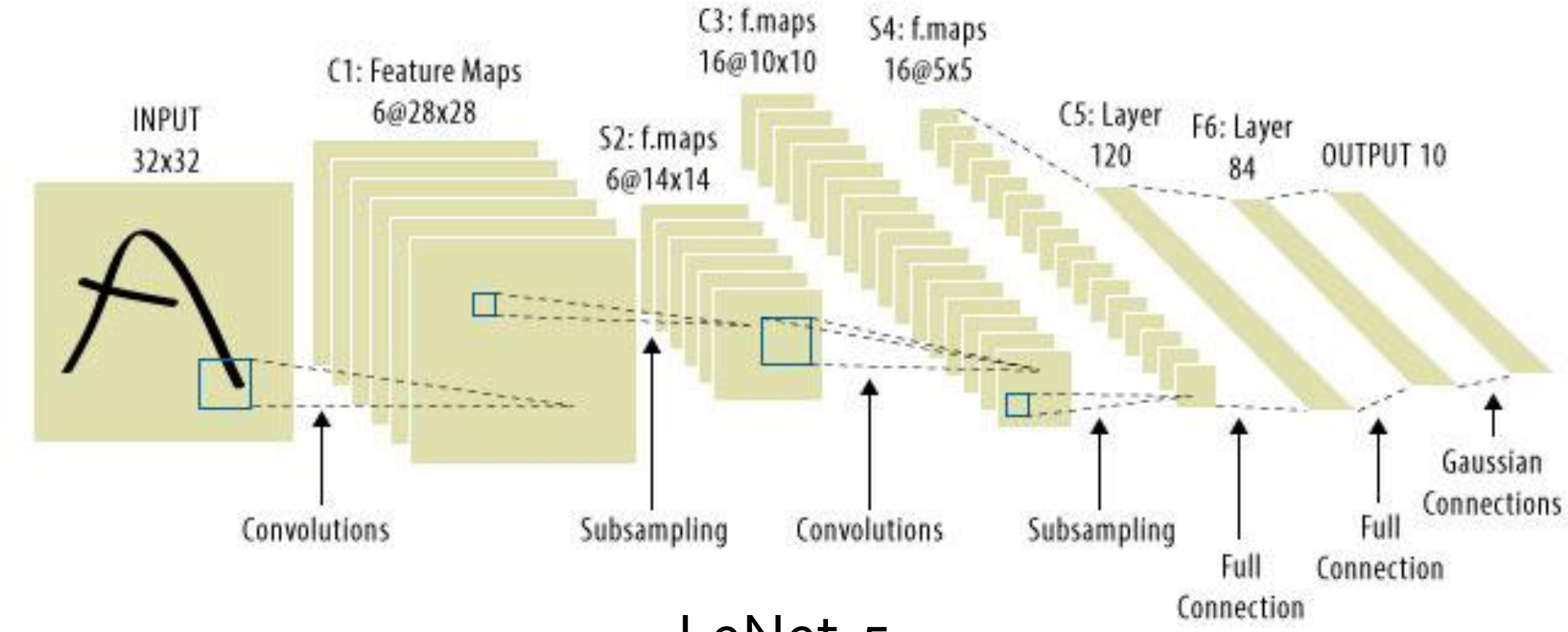
Instructor: Yuankai Huo, Ph.D.
Department of Electrical Engineering and Computer Science
Vanderbilt University

Topics

- Review
- Classification
- Localization
- Detection

Summary of Classification

"CONV-POOL-FC"



LeNet-5

DNN vs CNN

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential([
    Dense(32, activation='relu', input_shape=(input_size,)),
    Dense(64, activation='relu'),
    Dense(128, activation='relu'),
    Dense(num_classes, activation='softmax')
])

model.summary()
```

```
Using TensorFlow backend.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                4816928
_________________________________________________________________
dense_2 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_3 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
=================================================================
Total params: 4,828,650
Trainable params: 4,828,650
Non-trainable params: 0
```

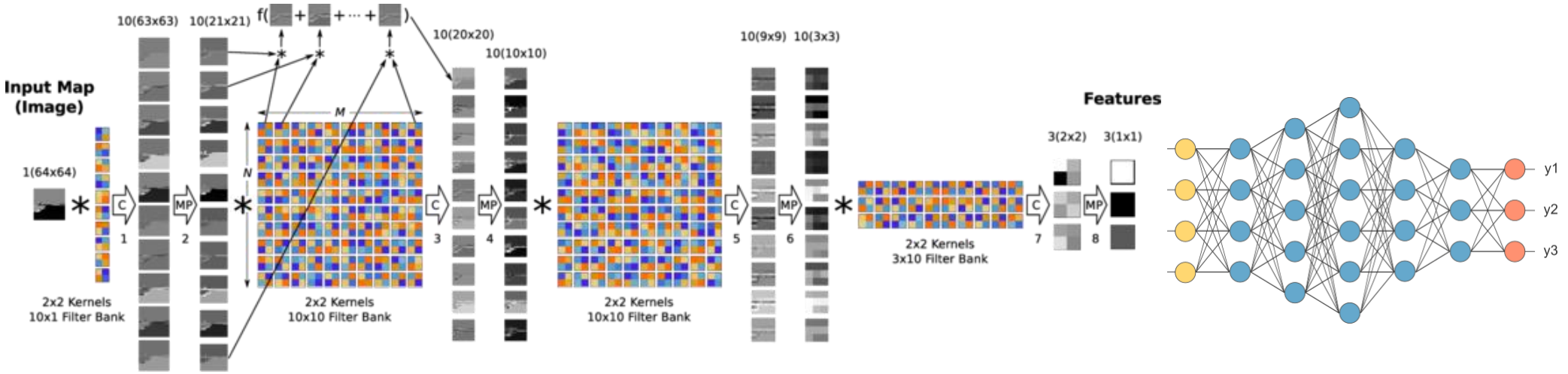
DNN

```
from keras.models import Sequential
from keras.layers import Conv2D


model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
    Conv2D(64, (3, 3), activation='relu'),
    Conv2D(128, (3, 3), activation='relu'),
    Dense(num_classes, activation='softmax')
])

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 220, 220, 64)      18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 218, 218, 128)     73856
_________________________________________________________________
dense_5 (Dense)              (None, 218, 218, 10)      1290
=================================================================
Total params: 94,538
Trainable params: 94,538
Non-trainable params: 0
```

CNN

Deep Convolutional Network Classification



Loss Function

Convolutional Layer

Fully Connected Layer (Dense Layer)

https://www.researchgate.net/figure/Max-Pooling-Convolutional-Neural-Network-MPCNN-with-8-layers-alternating-between_fig3_266656356

Three Examples

computed			targets			correct?
0.3	0.3	0.4	0	0	1	yes
0.3	0.4	0.3	0	1	0	yes
0.1	0.2	0.7	1	0	0	no

computed			targets			correct?
0.1	0.2	0.7	0	0	1	yes
0.1	0.7	0.2	0	1	0	yes
0.3	0.4	0.3	1	0	0	no

Classification Accuracy: CA = (#correct / #all)

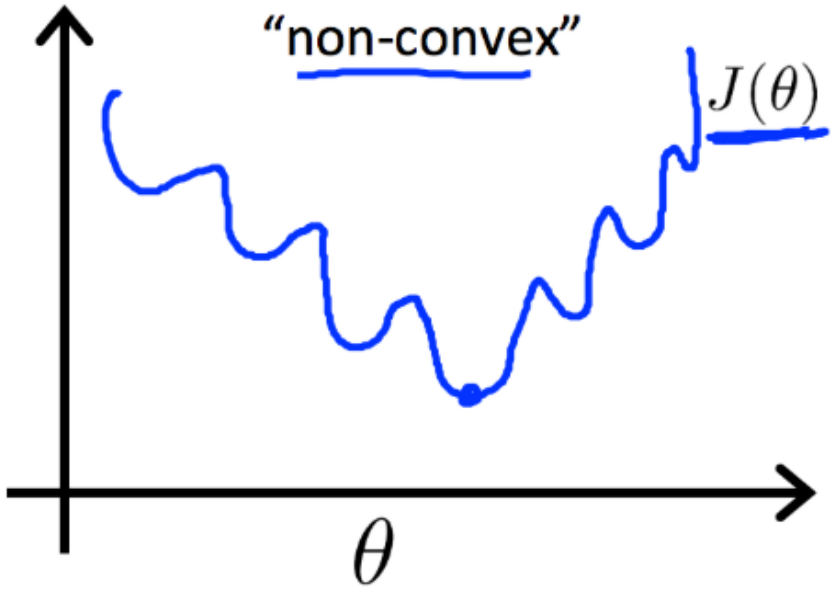
Mean square Error (MSE)

MSE $\quad \mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$

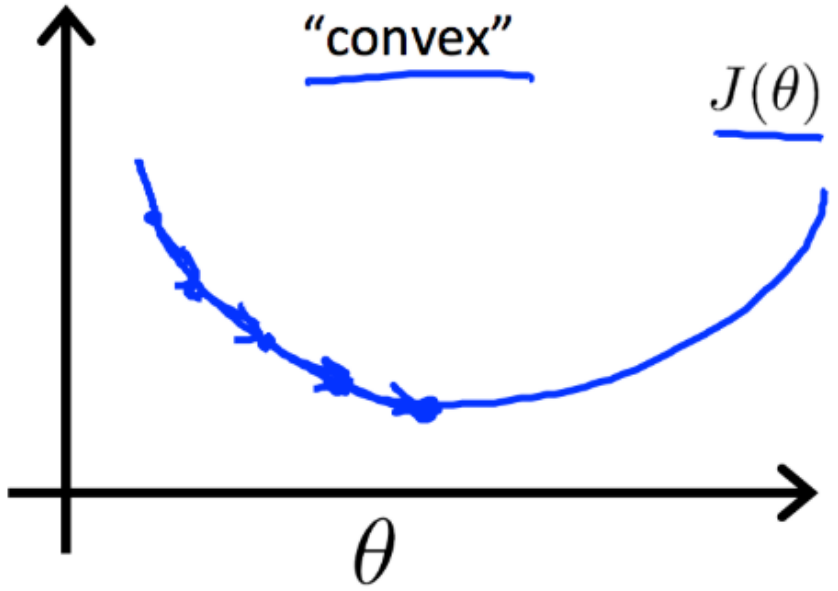
Cross-entropy $\quad H(p,q) = -\sum_x p(x)\,\log q(x)$

MSE vs Cross-entropy

MSE



Cross-entropy

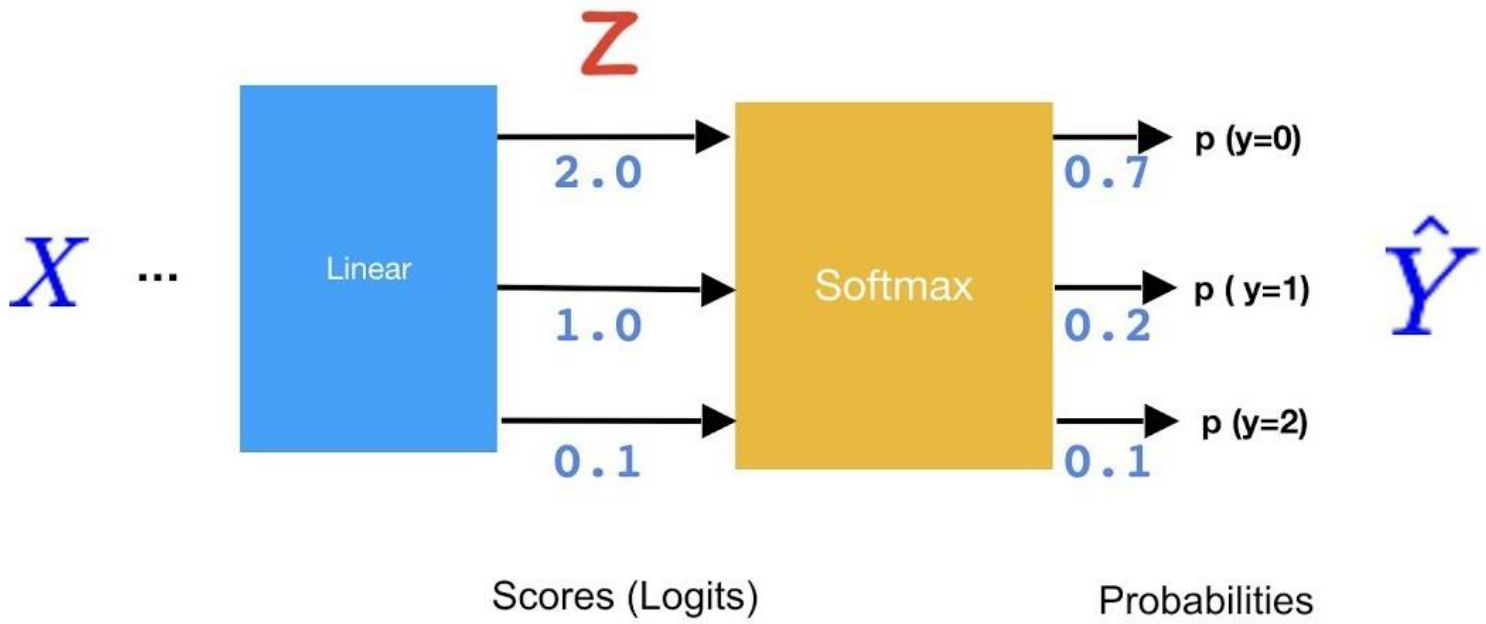


Andrew Ng

Softmax

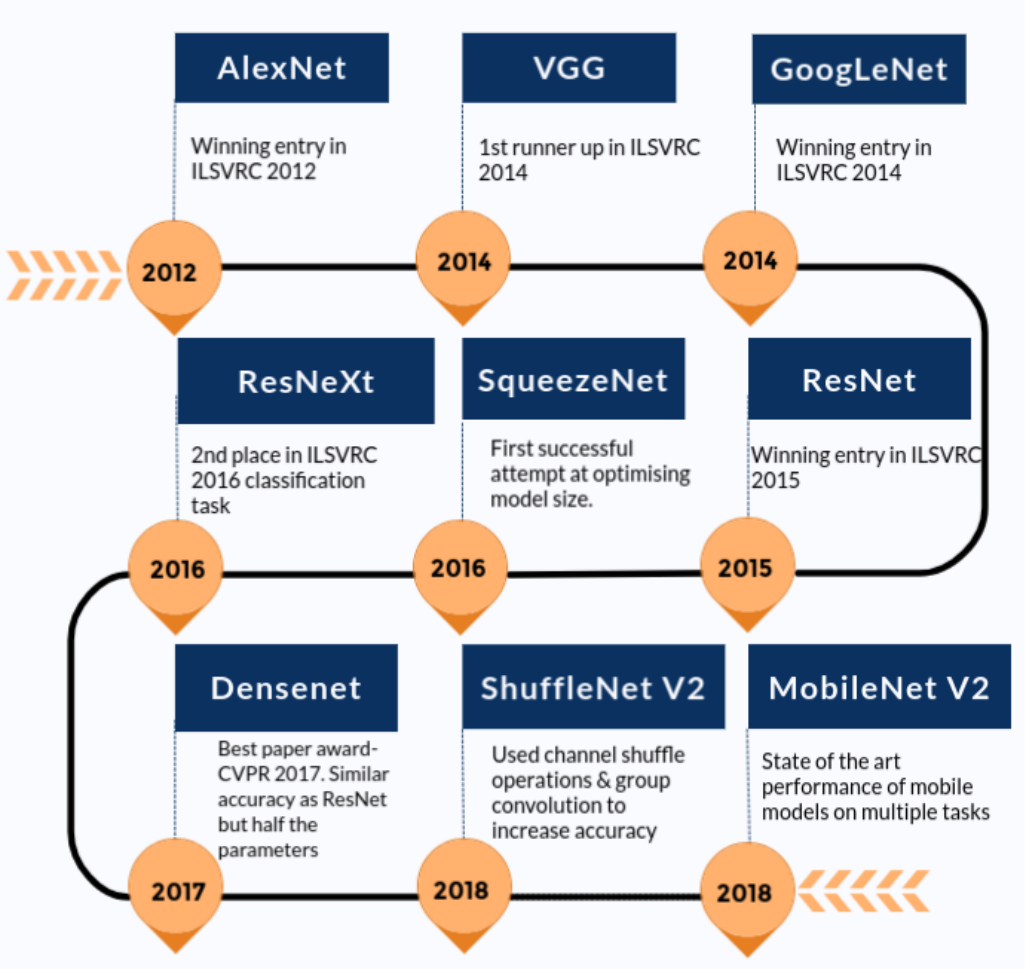
Meet Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



https://www.youtube.com/watch?v=lvNdl7yg4Pg

State-of-the-art



AlexNet

VGG

GoogleNet

ResNet

DenseNet

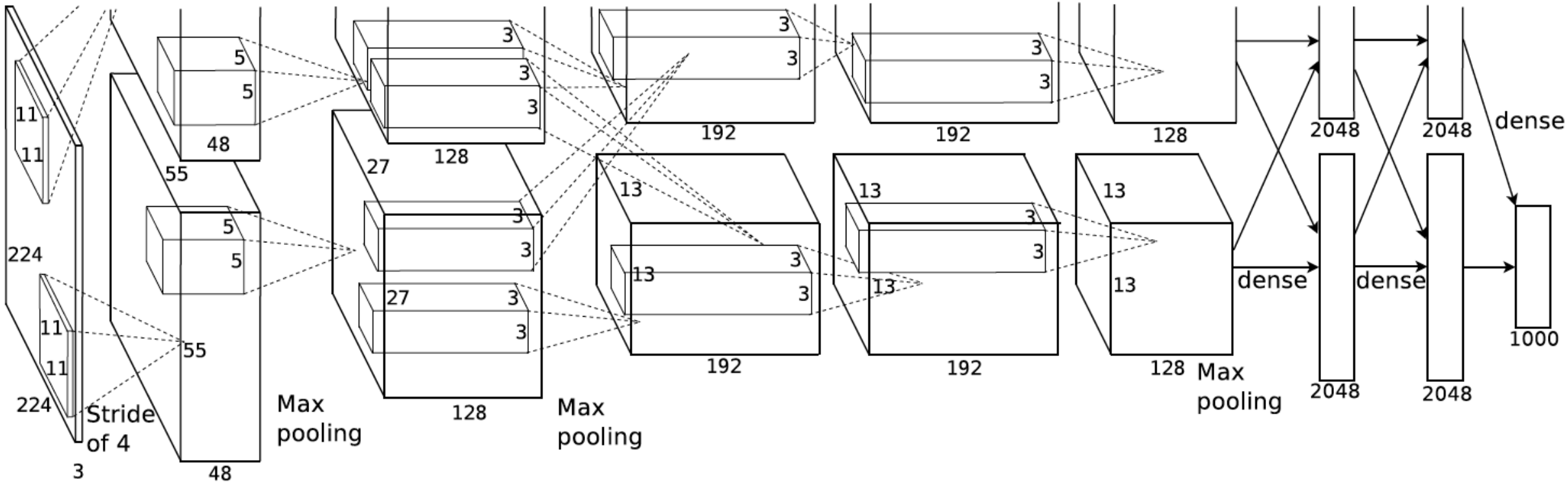
https://www.learnopencv.com/pytorch-for-beginners-image-classification-using-pre-trained-models/

AlexNet

[PDF] ImageNet Classification with Deep Convolutional ... - NIPS Proceedings

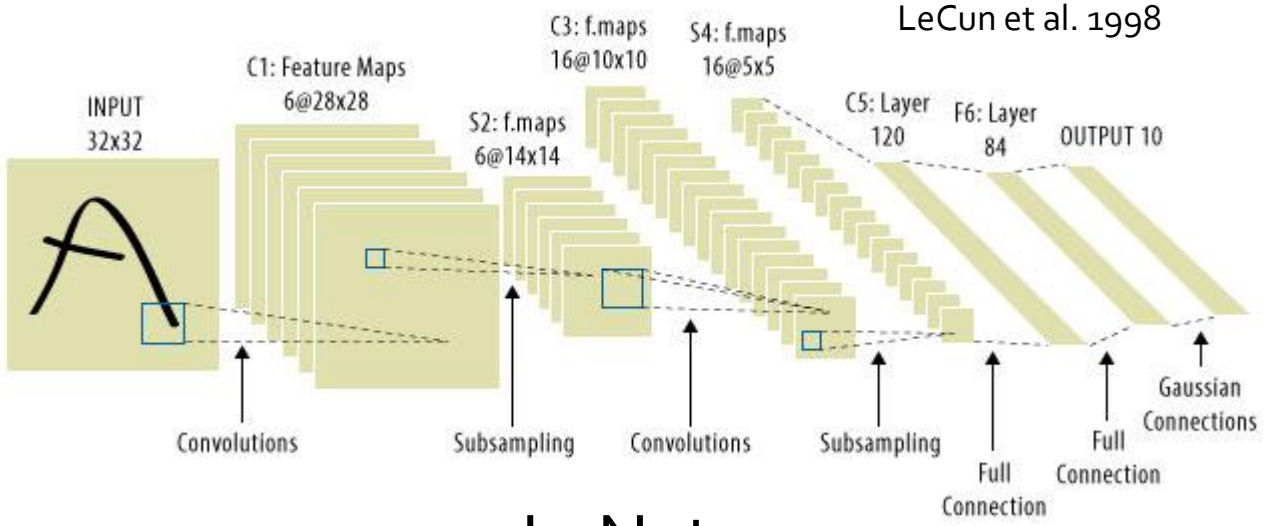
https://papers.nips.cc/.../4824-imagenet-classification-with-deep-convolutional-neural-...

by A Krizhevsky - 2012 - Cited by 27351 - Related articles



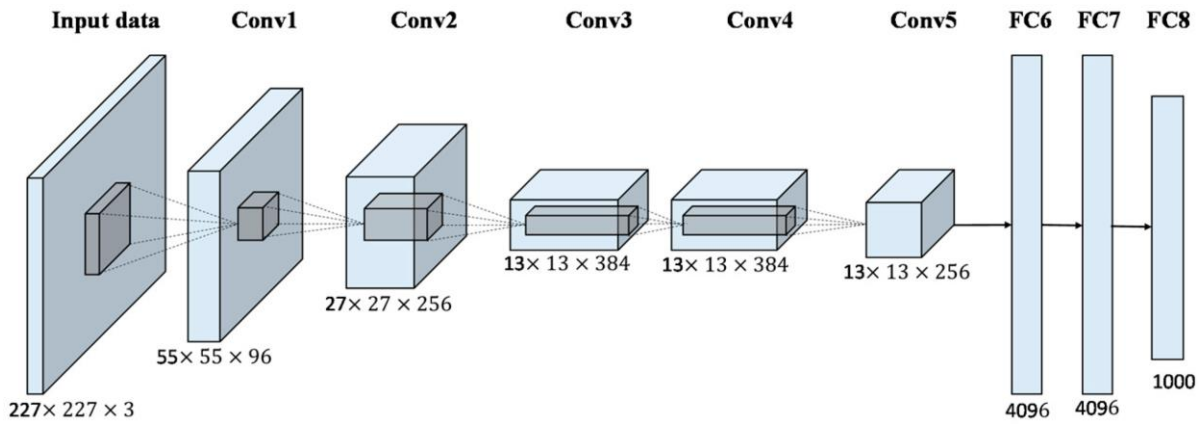
AlexNet vs LeNet-5

LeCun et al. 1998



LeNet-5

Krizhevsky et al. 2012

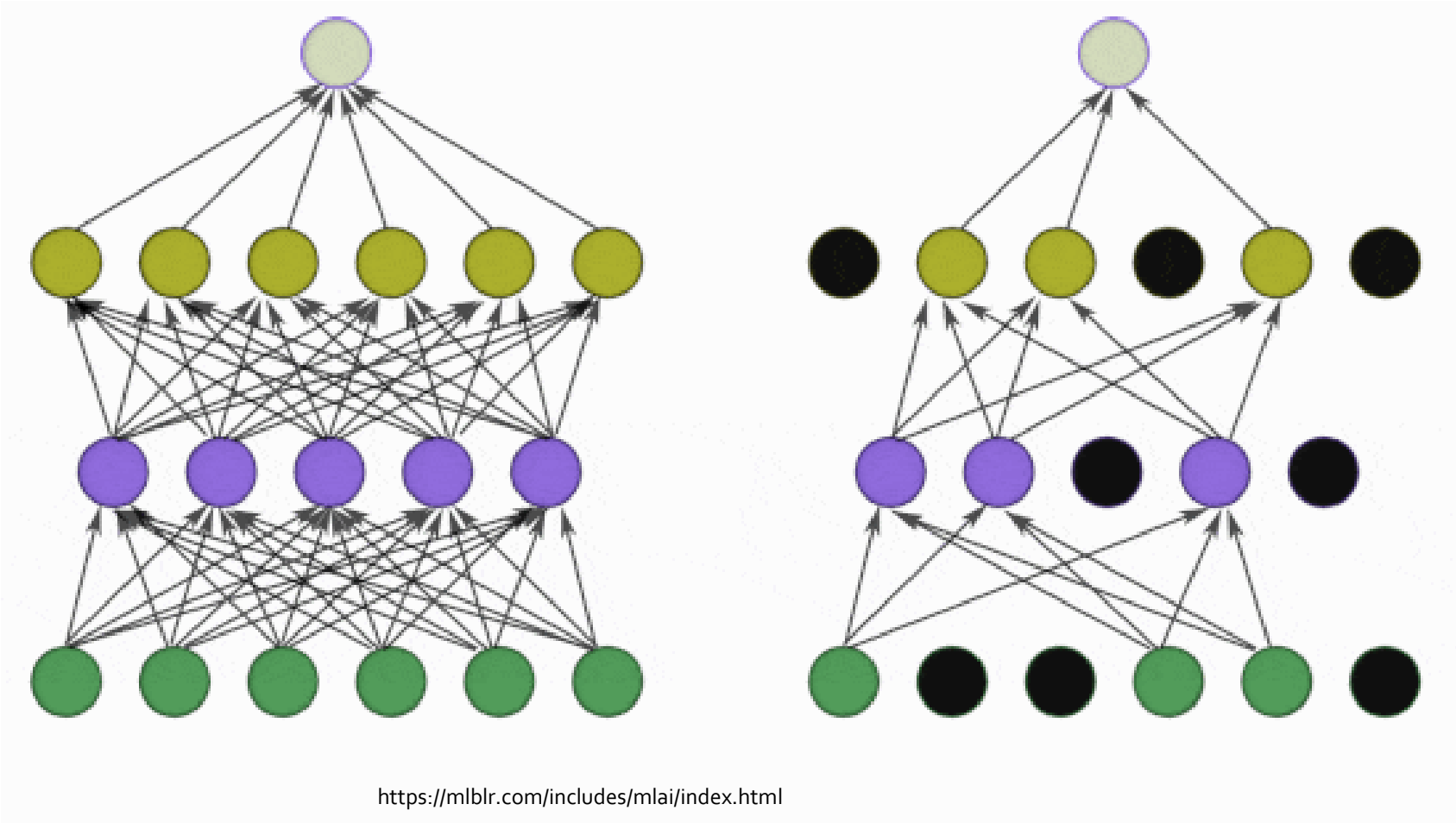


AlexNet

The network had a very similar architecture as LeNet by Yann LeCun et al but was deeper, with more filters per layer, and with stacked convolutional layers. It consisted 11x11, 5x5,3x3, convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum. It attached ReLU activations after every convolutional and fully-connected layer. AlexNet was trained for 6 days simultaneously on two Nvidia Geforce GTX 580 GPUs which is the reason for why their network is split into two pipelines.

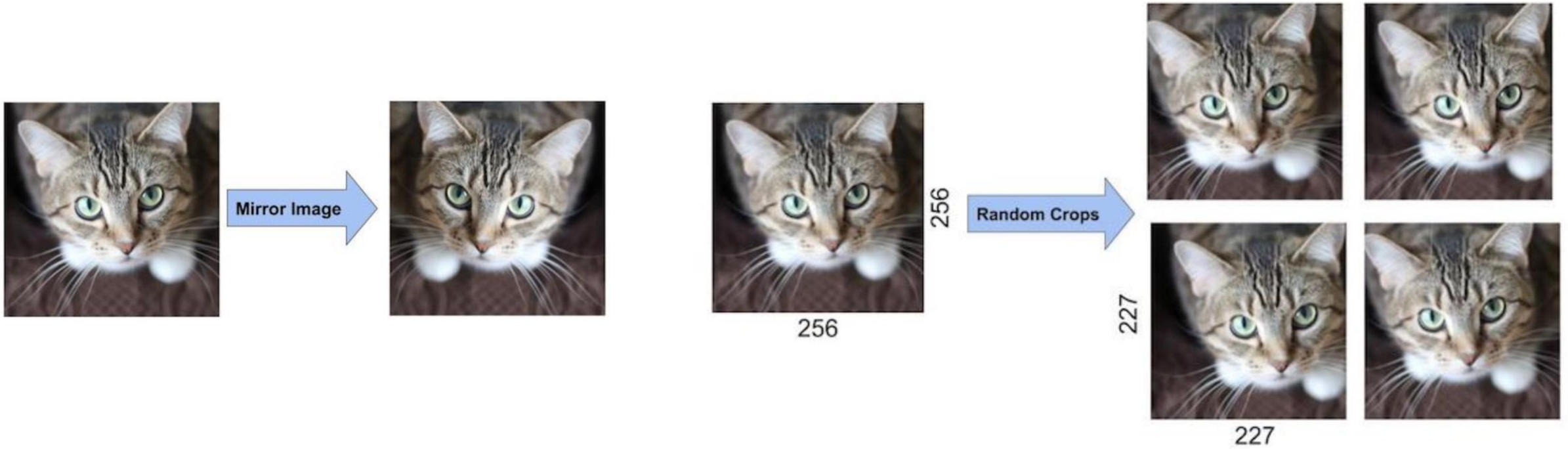
https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Dropout



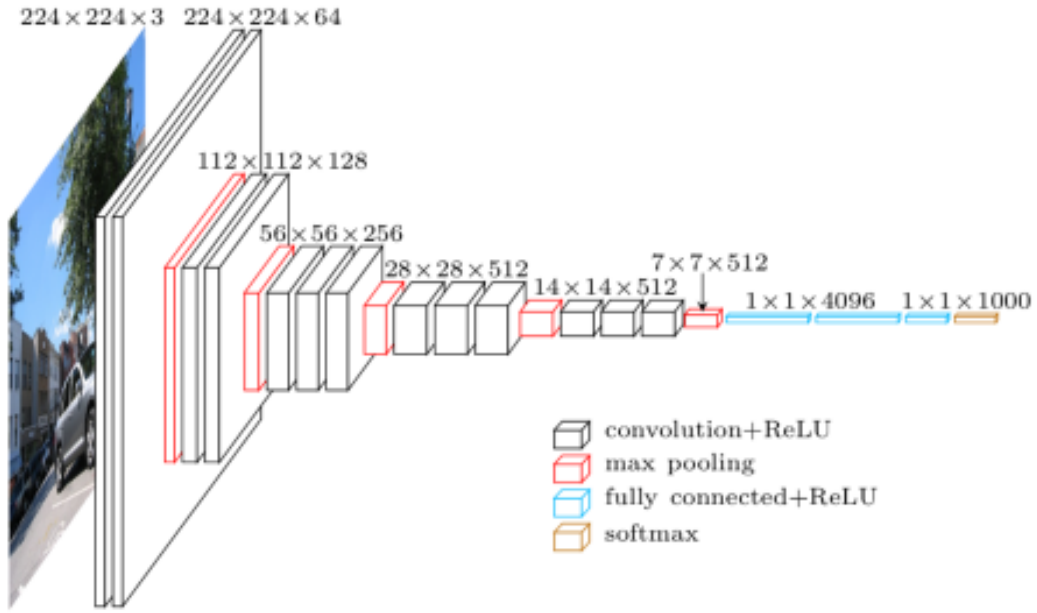
https://mlblr.com/includes/mlai/index.html

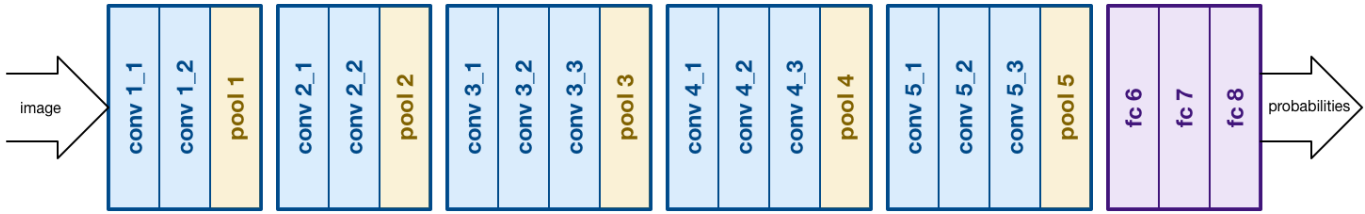
Data Augmentation



https://www.learnopencv.com/understanding-alexnet/

VGGNet (2014)

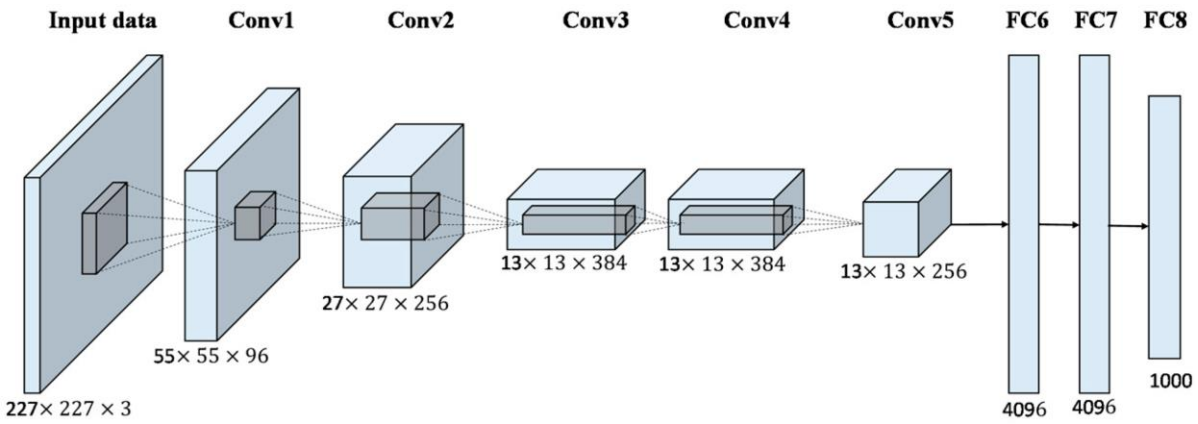




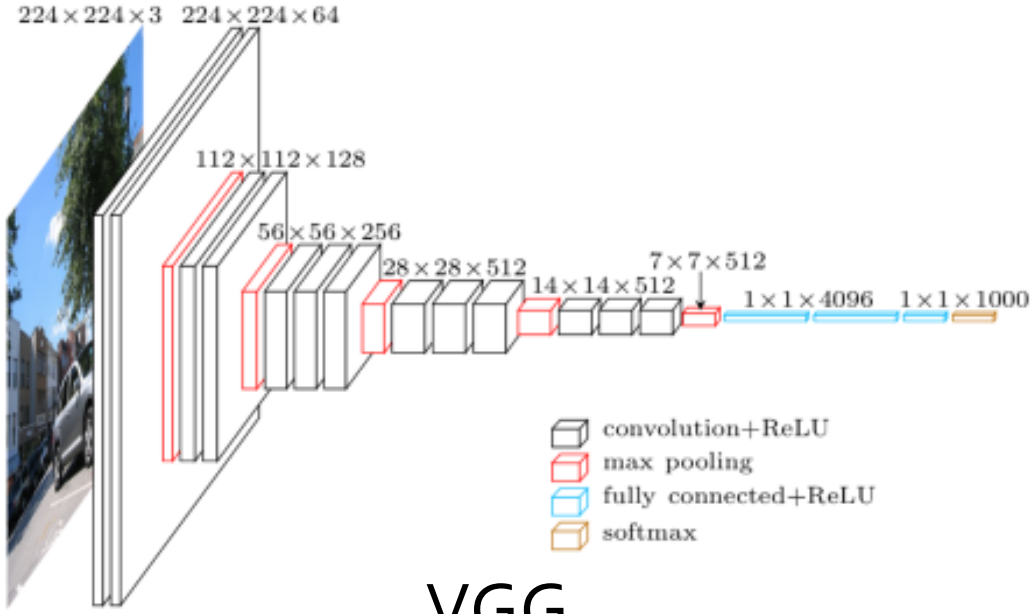
https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/
https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

AlexNet vs VGG

Krizhevsky et al. 2012



AlexNet

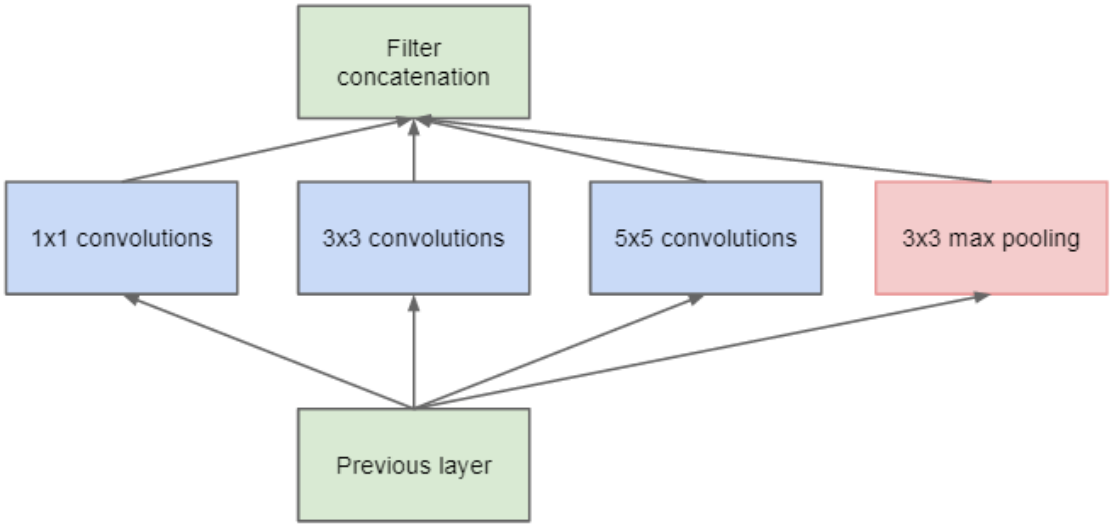


VGG

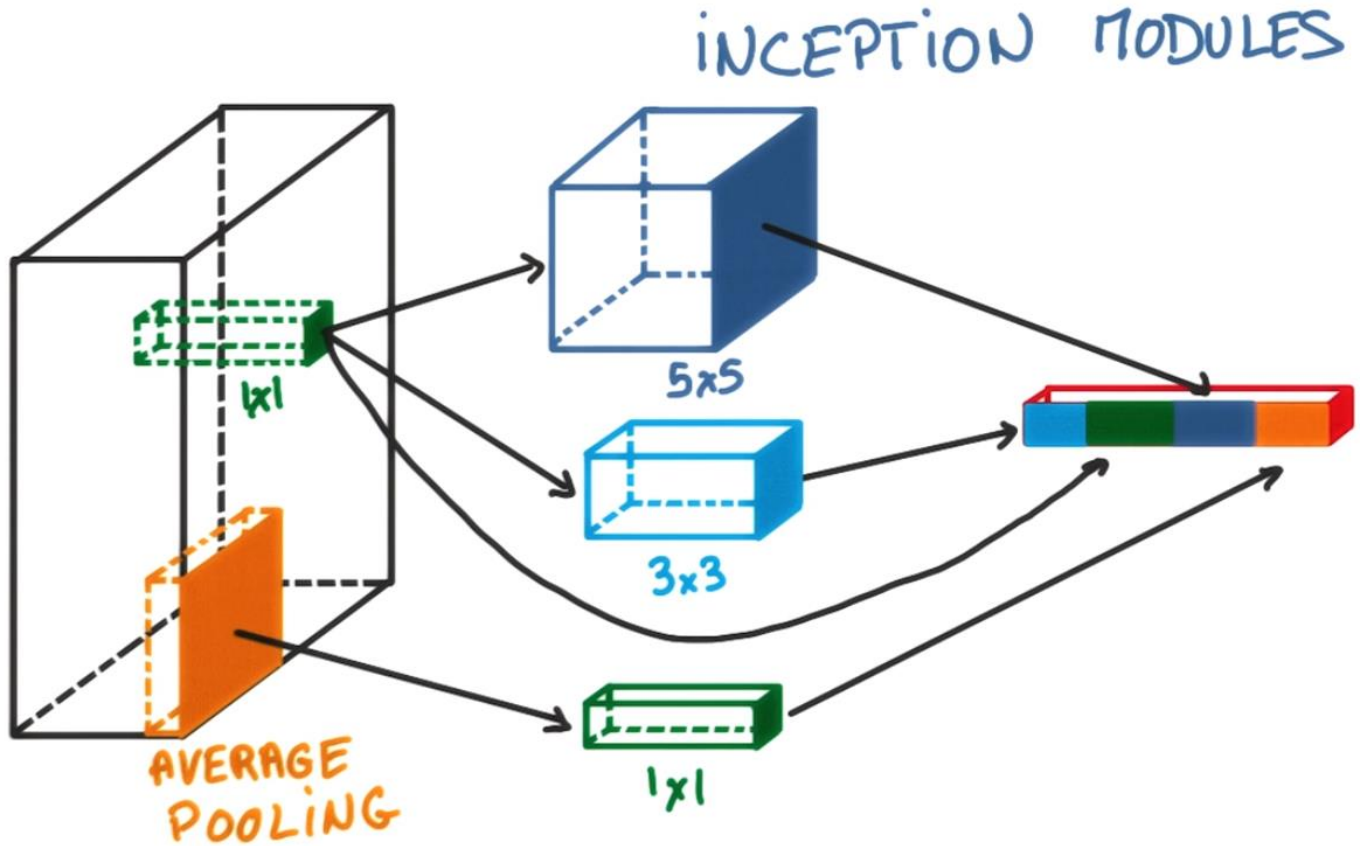
The idea behind having fixed size kernels is that all the variable size convolutional kernels used in Alexnet (11x11, 5x5, 3x3) can be replicated by making use of multiple 3x3 kernels as building blocks. The replication is in terms of the receptive field covered by the kernels. This reduces the number of trainable variables by 44.9% (62.8%). Reduced number of trainable variables means faster learning and more robust to over-fitting. max poolings should be placed after each 2 convolutions and the number of filters should be doubled after each max-pooling.

https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

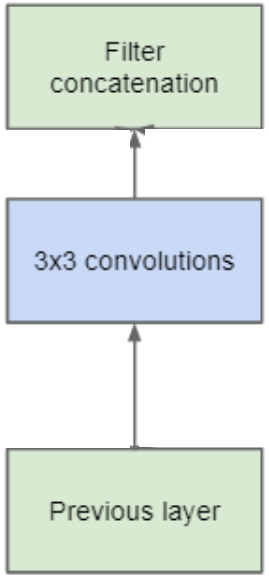
Inception



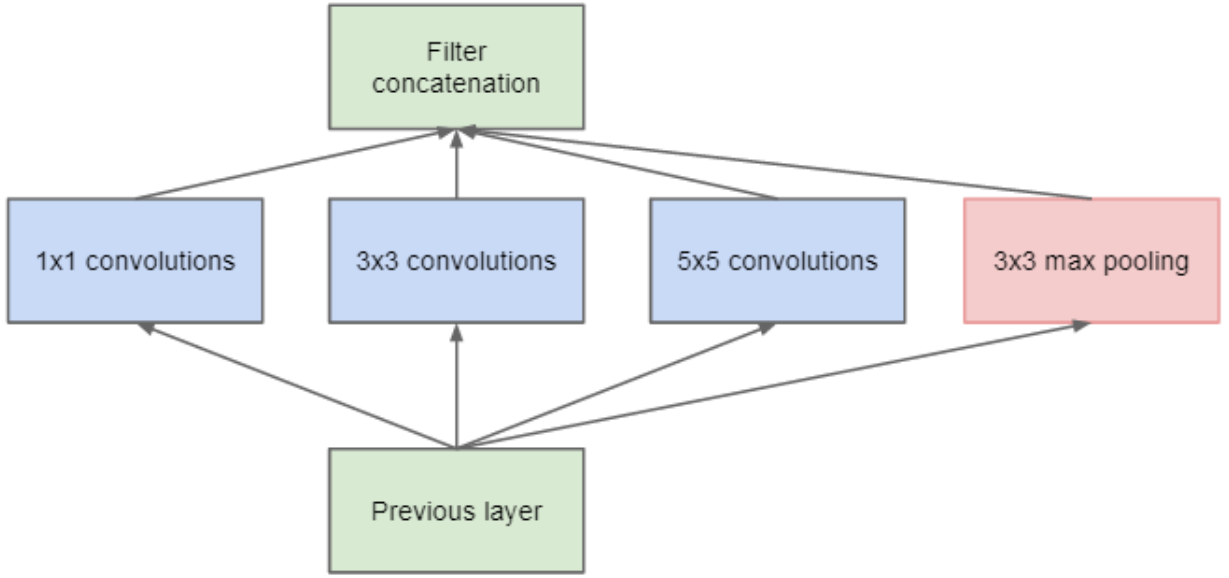
(a) Inception module, naïve version



Difference from VGG



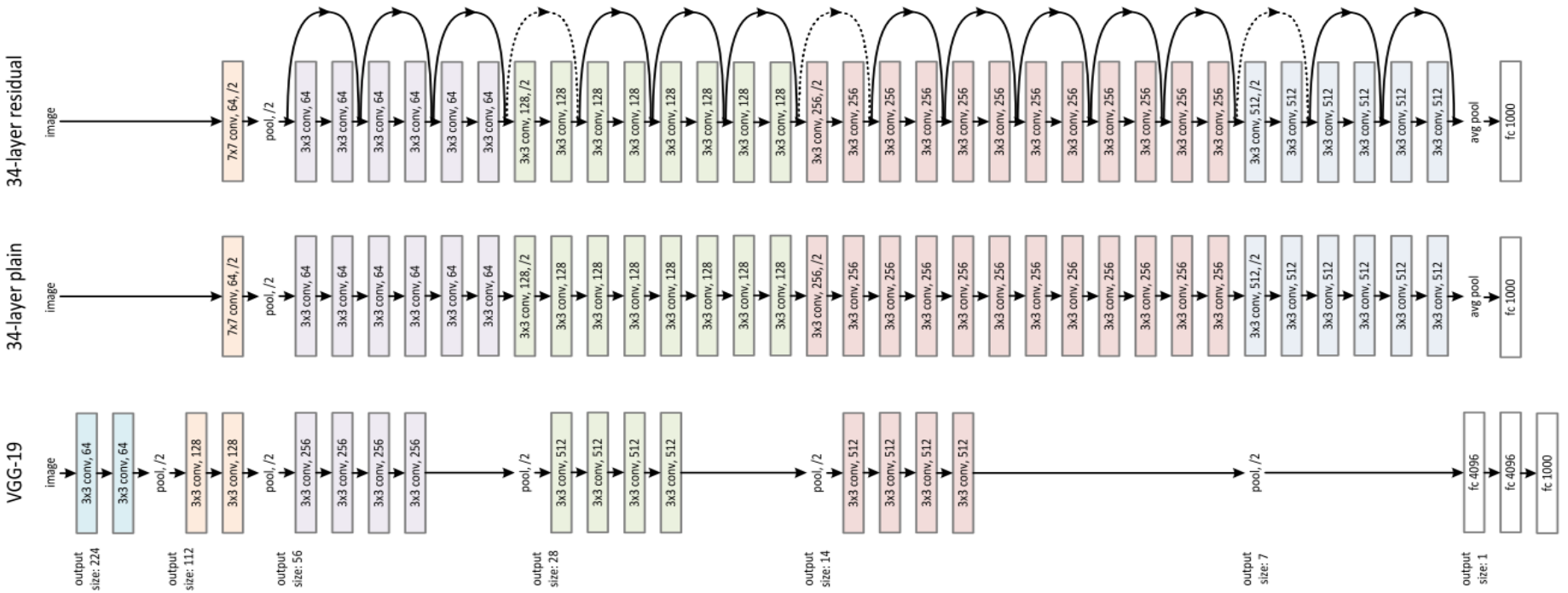
VGG



Google

https://becominghuman.ai/understanding-and-coding-inception-module-in-keras-eb56e9056b4b

ResNet(2015) vs VGG

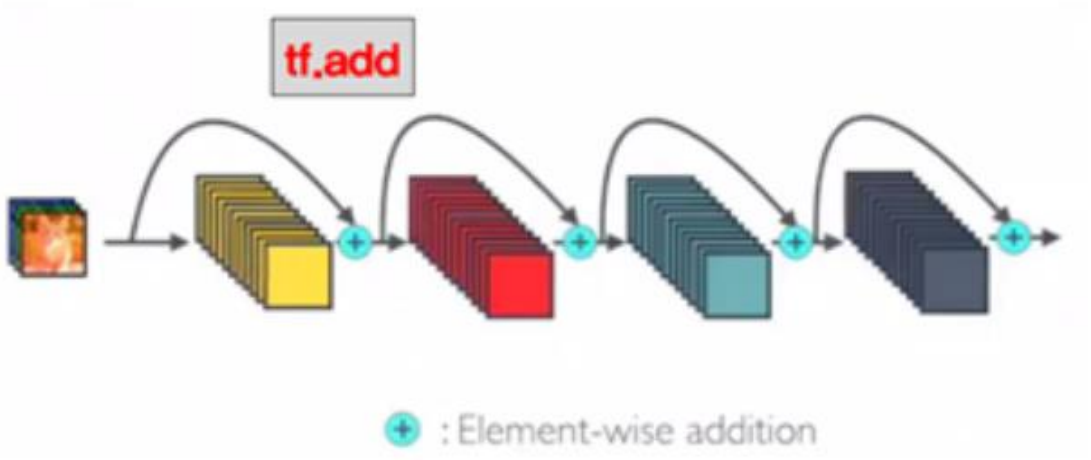


Ay the ILSVRC 2015, the so-called Residual Neural Network (ResNet) by Kaiming He et al introduced anovel architecture with “skip connections” and features heavy batch normalization. Such skip connections are also known as gated units or gated recurrent units and have a strong similarity to recent successful elements applied in RNNs. Thanks to this technique they were able to train a NN with 152 layers while still having lower complexity than VGGNet. It achieves a top-5 error rate of 3.57% which beats human-level performance on this dataset.

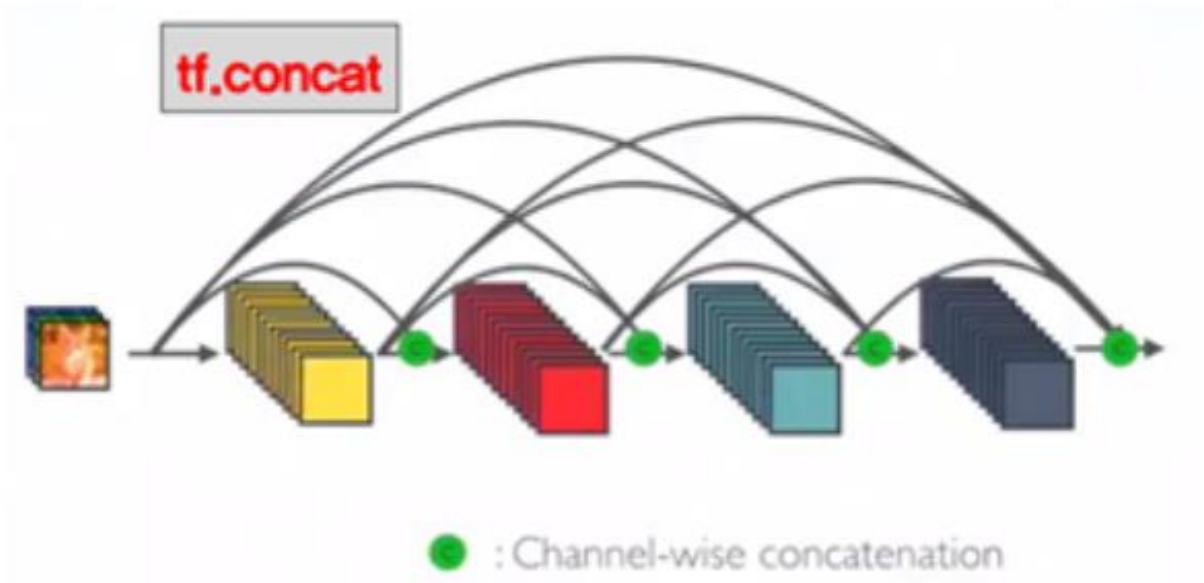
https://medium.com/@pierre_guillou/understand-how-works-resnet-without-talking-about-residual-64698f157e0c

https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

More Connections



ResNet Connectivity

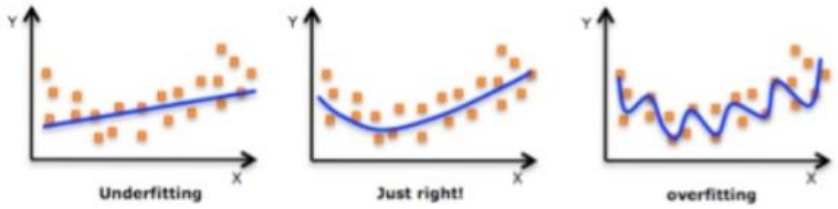


Dense Connectivity

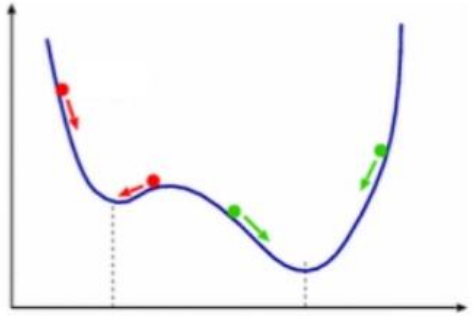
https://jayhey.github.io/deep%20learning/2017/10/13/DenseNet_1/

Challenges

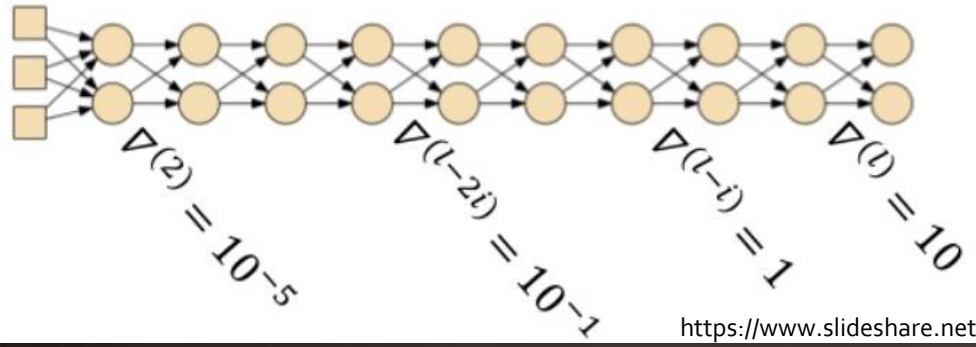
- Big networks => Too huge separating ability => **Overfitting**



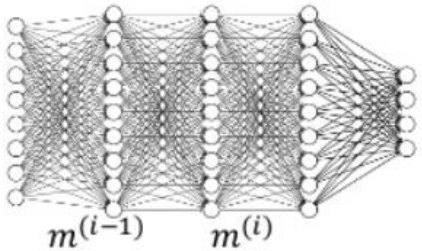
- Complex error's surface => **Local minimum**



- **Vanishing gradient problem** during training



- Curse of dimensionality => **memory & computations**



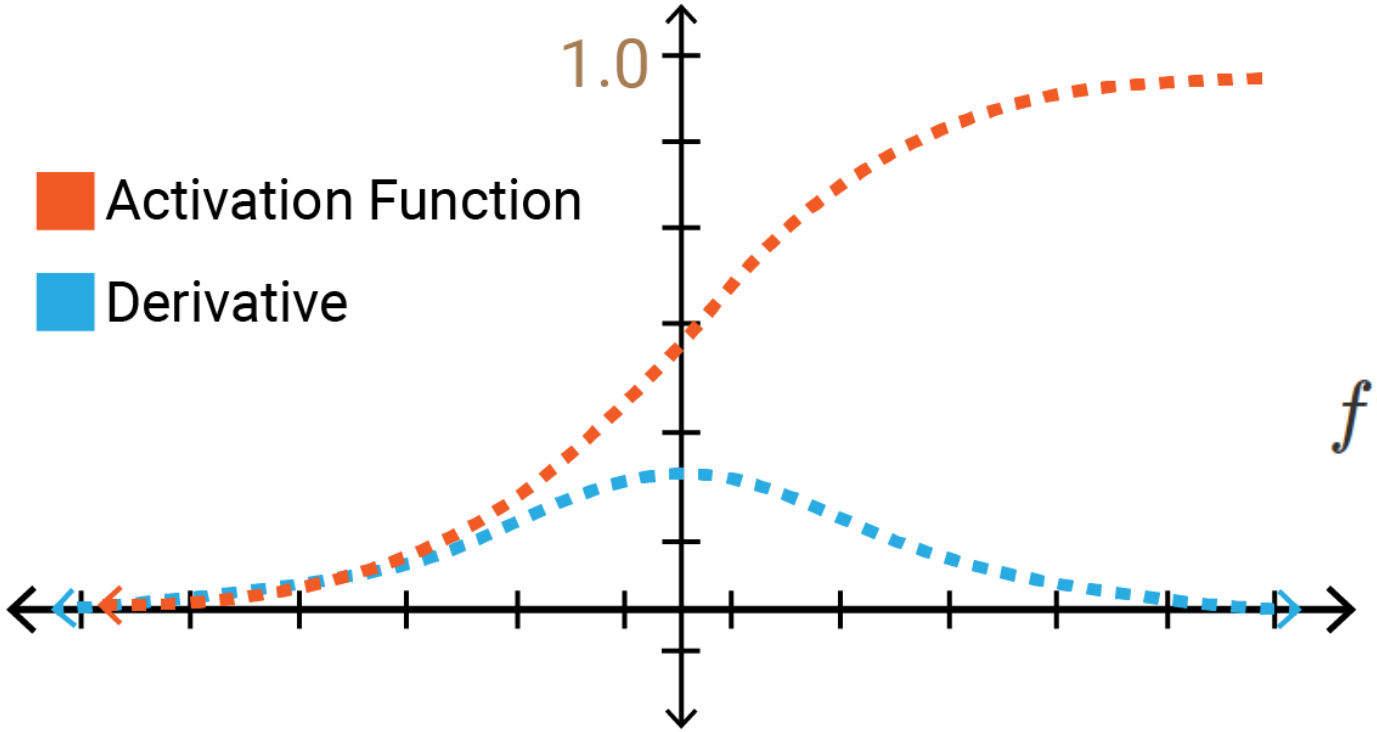
$$\dim(W^{(i)}) = m^{(i-1)} * m^{(i)}$$

https://www.slideshare.net/baiev1/neural-networks-overview

Overfitting
Local minimum
Gradient Vanishing
Computation & Memory

Gradient Vanishing/Exploding problems

Sigmoid (Logistic)

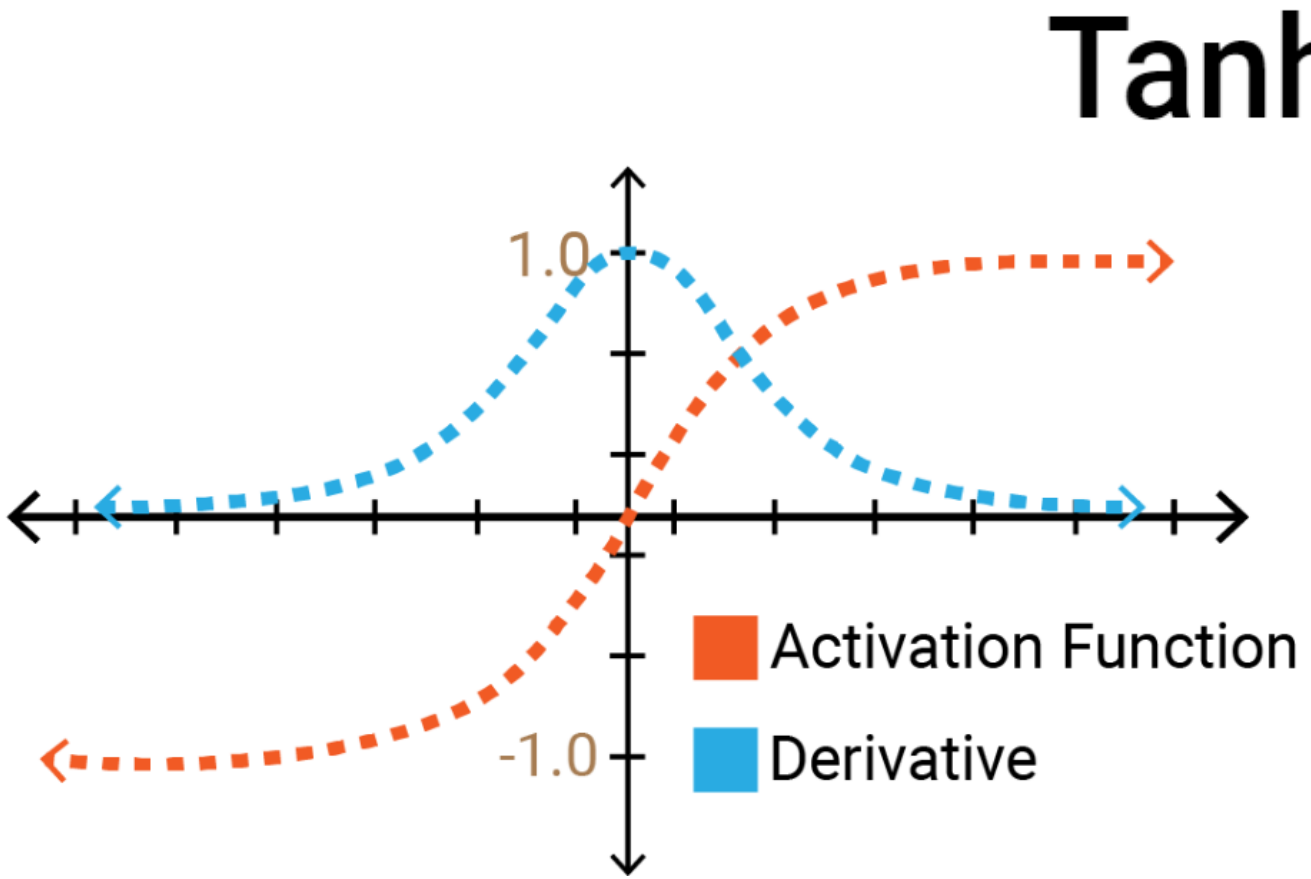


$$f(x) = \frac{1}{1 + e^{-x}}$$

$$f'(x) = \frac{1}{1 + e^{-x}} \left(1 + \frac{1}{1 + e^{-x}}\right)$$

https://codeodysseys.com/posts/activation-functions/

Better, but still

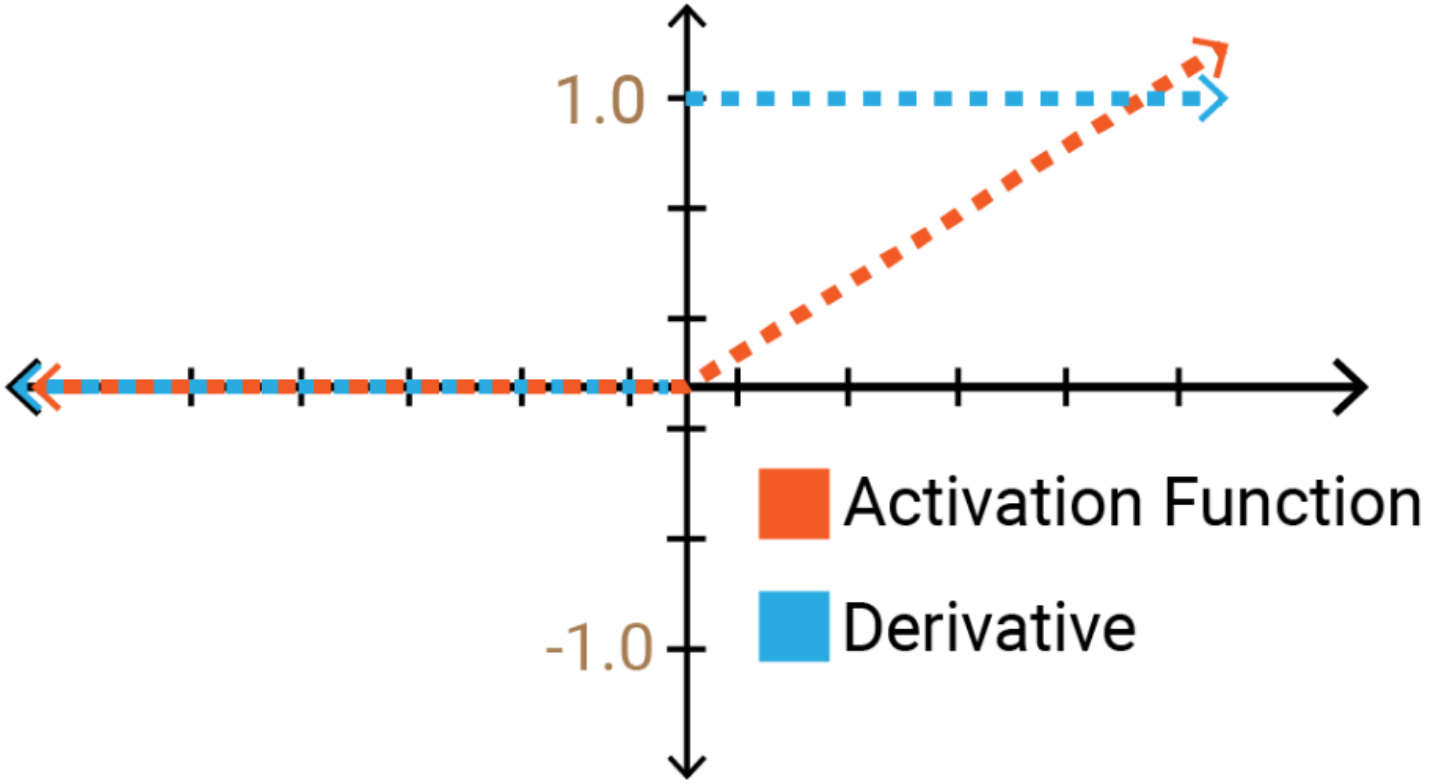


$$f(x) = \frac{2}{1 + e^{-2x}} - 1 = \tanh(x)$$

$$f'(x) = 1 - \tanh^2(x)$$

https://codeodysseys.com/posts/activation-functions/

ReLU



$$f(x) = max(0, x)$$

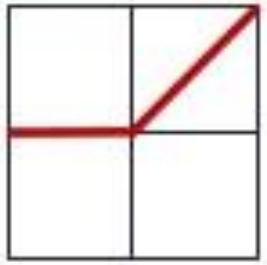
$$f'(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

•It's sparsely activated. Since ReLU is zero for all negative inputs, it's likely for any given unit to not activate at all. This is often desirable

Improve ReLU

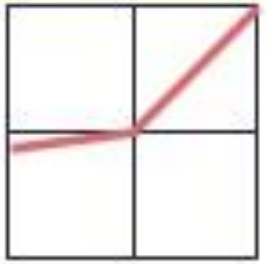
ReLU

$$x' = \max(0, x)$$



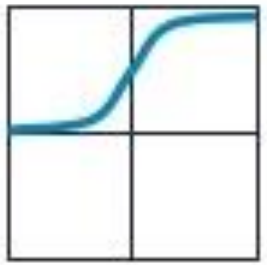
Leaky ReLU

$$x' = \max(0.1x, x)$$



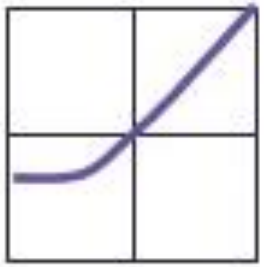
Sigmoid

$$x' = 1/(1 + e^{-x})$$



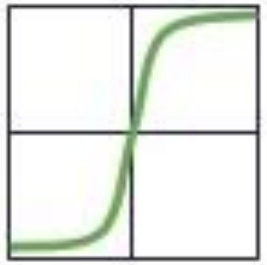
ELU

$$x' = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$$



TanH

$$x' = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



When in doubt, **ReLU**

Worth trying Leaky ReLU, ELU

Avoid Sigmoid

https://www.slideshare.net/embeddedvision/demystifying-deep-neural-networks-a-presentation-from-bdti

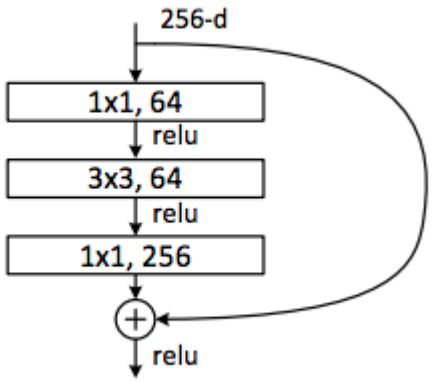
Assignment o

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
```

VGG

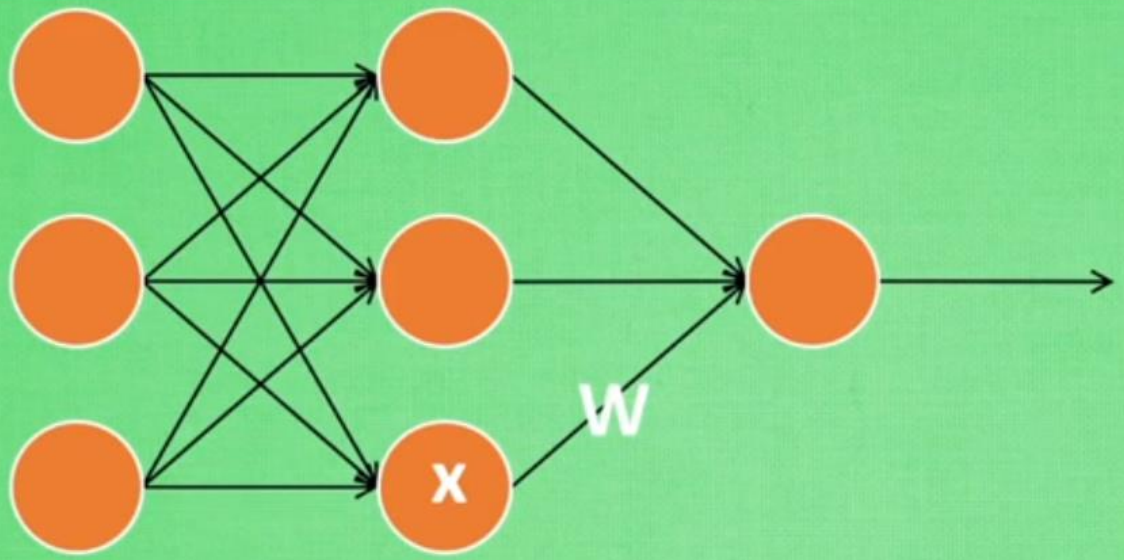
```
def forward(self, x, training=True):
    x = F.relu(self.conv1_1(x))
    x = F.relu(self.conv1_2(x))
    x = self.pool(x)
    x = F.relu(self.conv2_1(x))
    x = F.relu(self.conv2_2(x))
    x = self.pool(x)
    x = F.relu(self.conv3_1(x))
    x = F.relu(self.conv3_2(x))
    x = F.relu(self.conv3_3(x))
    x = self.pool(x)
    x = F.relu(self.conv4_1(x))
    x = F.relu(self.conv4_2(x))
    x = F.relu(self.conv4_3(x))
    x = self.pool(x)
    x = F.relu(self.conv5_1(x))
    x = F.relu(self.conv5_2(x))
    x = F.relu(self.conv5_3(x))
    x = self.pool(x)
    x = x.view(-1, 7 * 7 * 512)
    x = F.relu(self.fc6(x))
    x = F.dropout(x, 0.5, training=training)
    x = F.relu(self.fc7(x))
    x = F.dropout(x, 0.5, training=training)
    x = self.fc8(x)
    return x
```

ResNet




```python
class Bottleneck(nn.Module):
    expansion = 4

    def __init__(self, in_planes, planes, stride=1):
        super(Bottleneck, self).__init__()
        self.conv1 = nn.Conv2d(in_planes, planes, kernel_size=1, bias=False)
        self.bn1 = nn.BatchNorm2d(planes)
        self.conv2 = nn.Conv2d(planes, planes, kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(planes)
        self.conv3 = nn.Conv2d(planes, self.expansion*planes, kernel_size=1, bias=False)
        self.bn3 = nn.BatchNorm2d(self.expansion*planes)

        self.shortcut = nn.Sequential()
        if stride != 1 or in_planes != self.expansion*planes:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_planes, self.expansion*planes, kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(self.expansion*planes)
            )

    def forward(self, x):
        out = F.relu(self.bn1(self.conv1(x)))
        out = F.relu(self.bn2(self.conv2(out)))
        out = self.bn3(self.conv3(out))
        out += self.shortcut(x)
        out = F.relu(out)
        return out
```

Batch Normalization

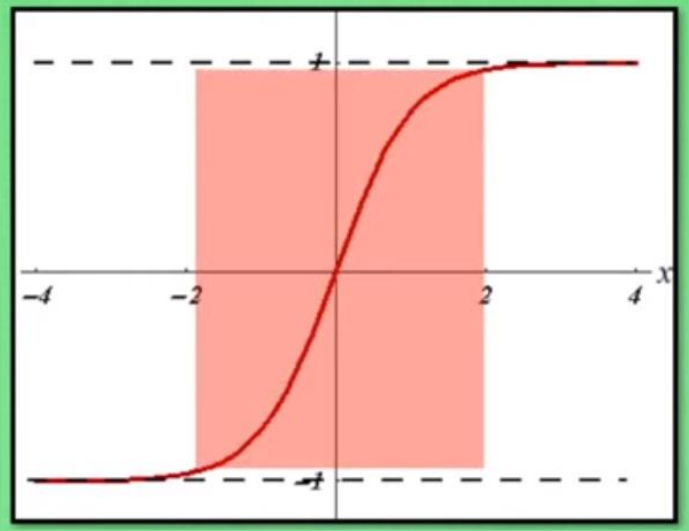


Wx1 = 0.1*1
=0.1

tanh(Wx1)
~=0.1

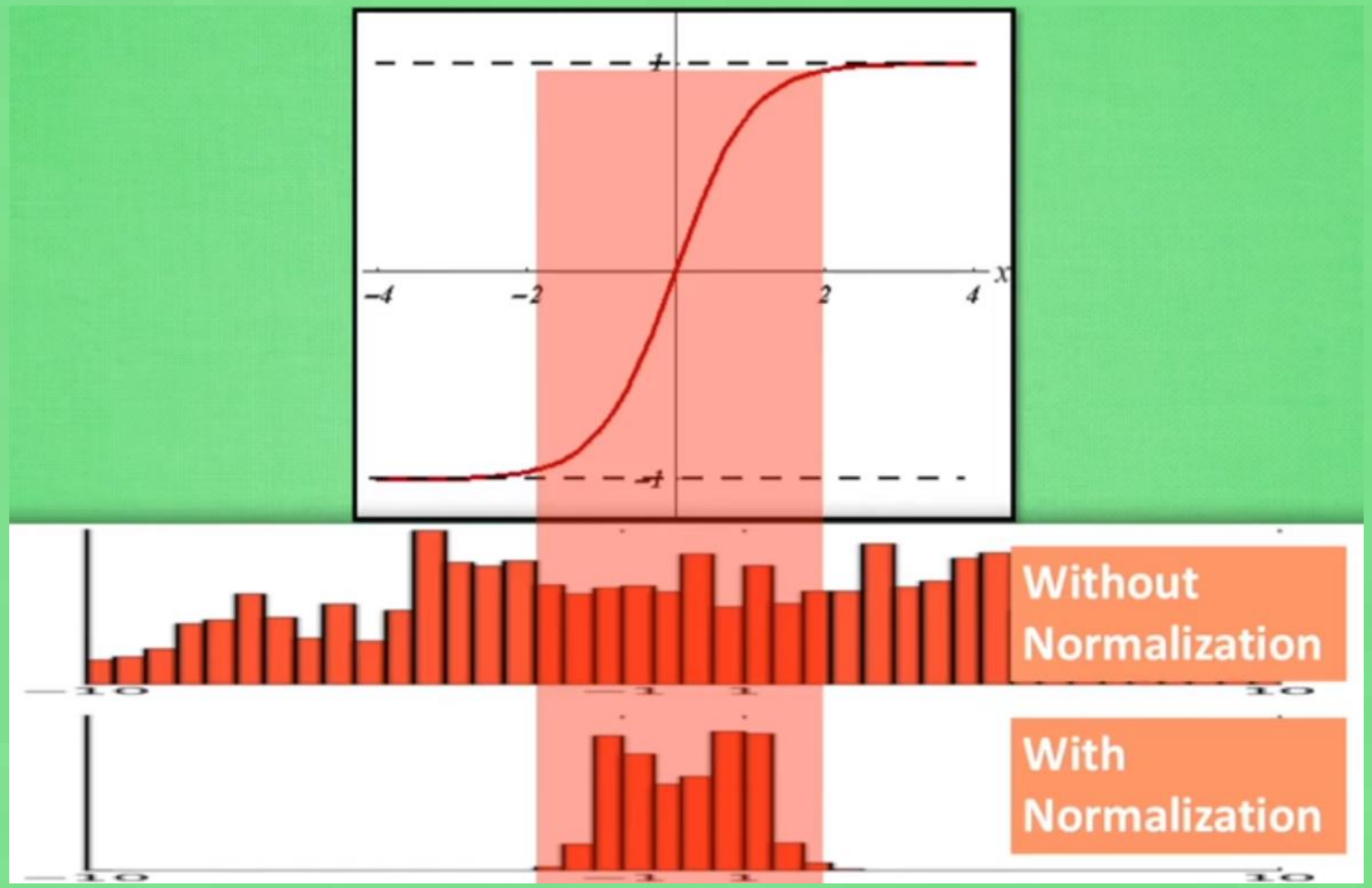
Wx2 = 0.1*20
=2

tanh(Wx2)
~= 0.96



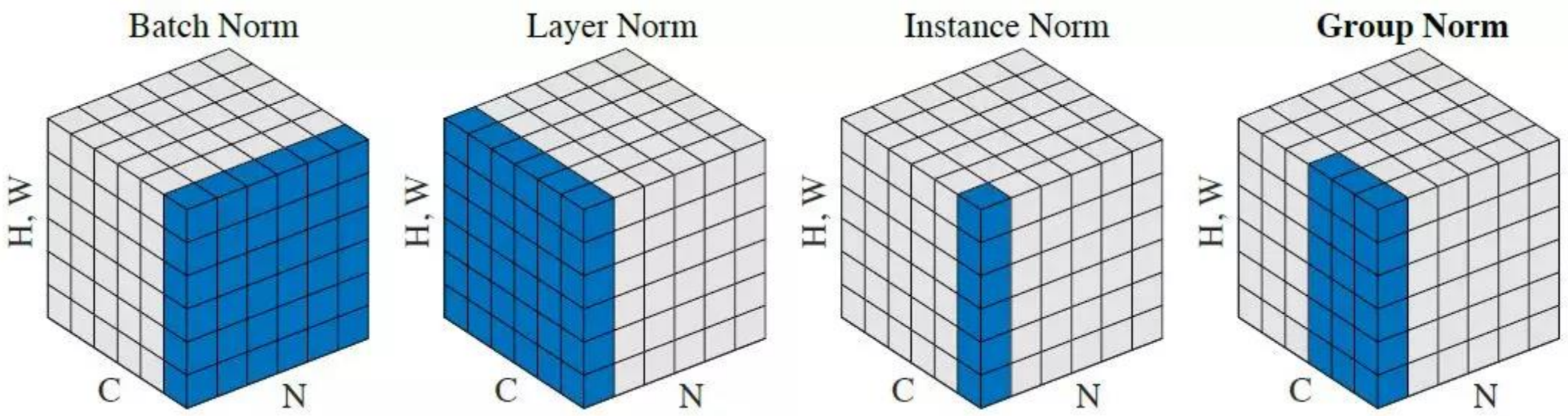
https://morvanzhou.github.io/tutorials/machine-learning/ML-intro/3-08-batch-normalization/

Batch Normalization

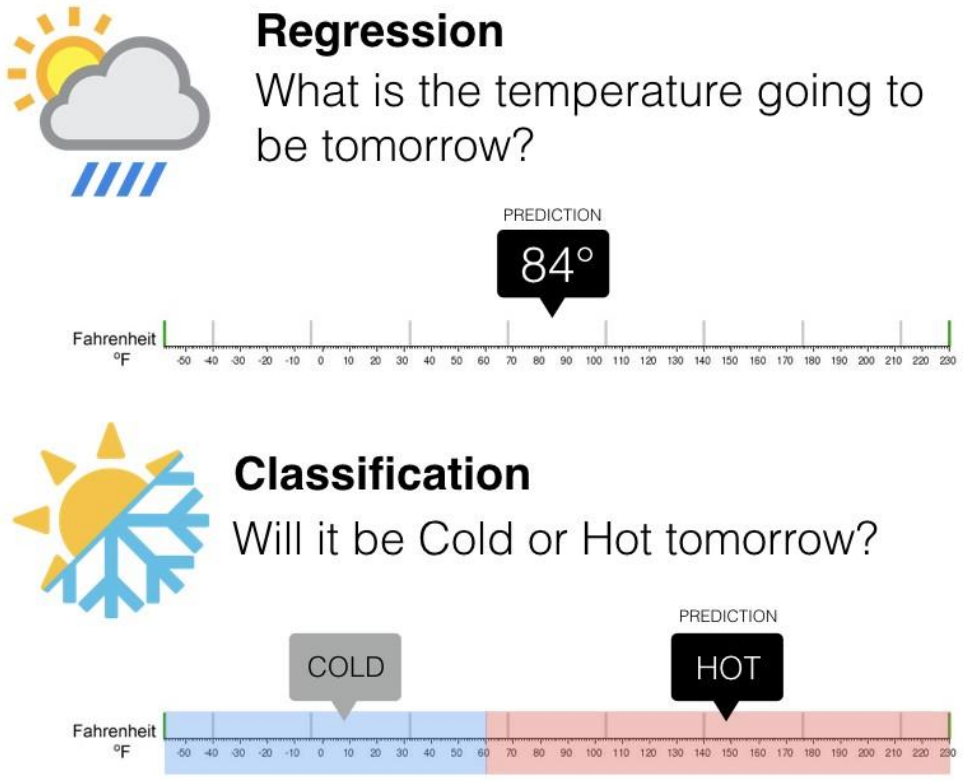


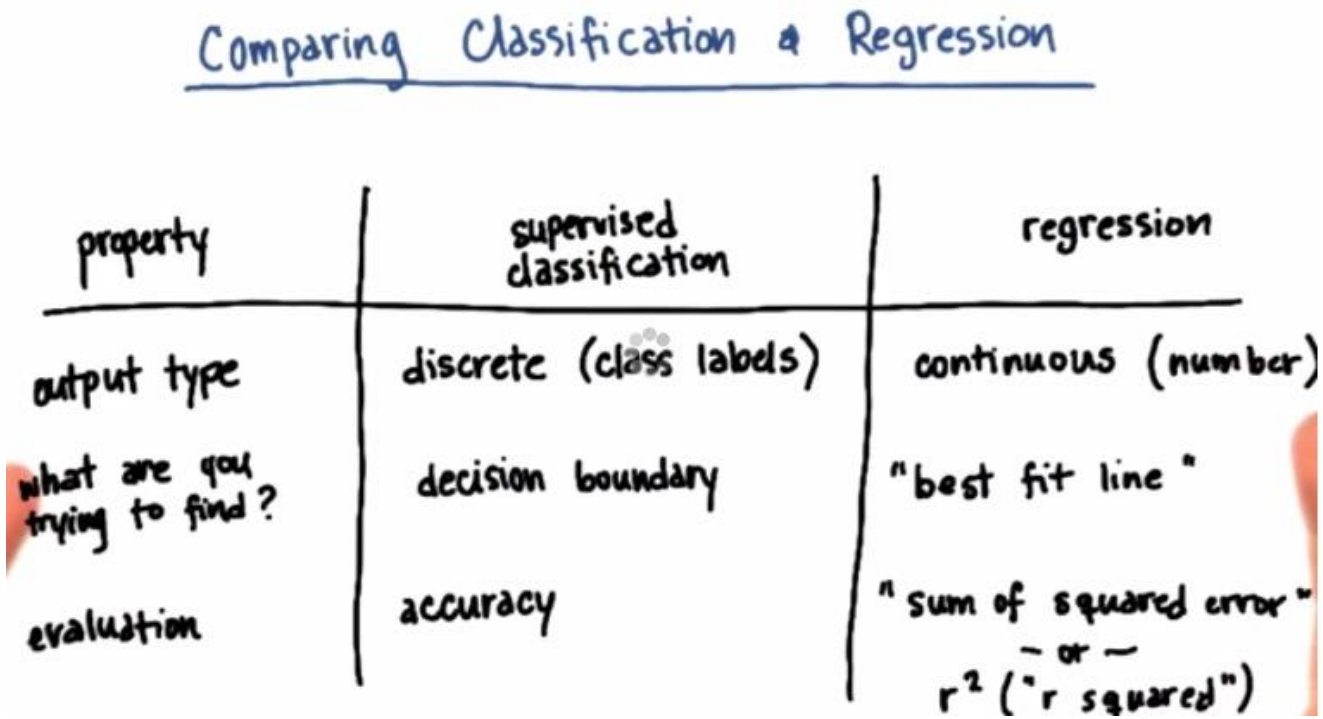
https://morvanzhou.github.io/tutorials/machine-learning/ML-intro/3-08-batch-normalization/

Batch Norm and Layer Norm



Classification & Regression



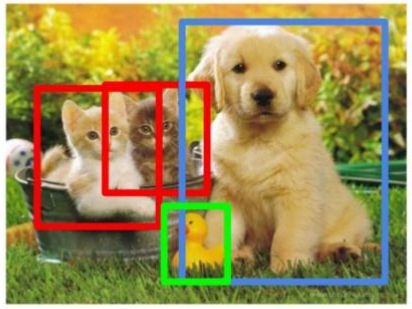
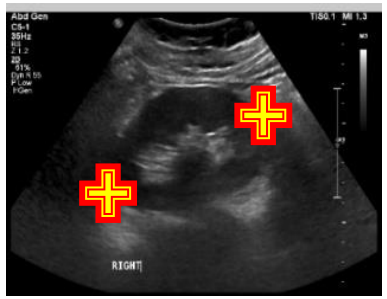
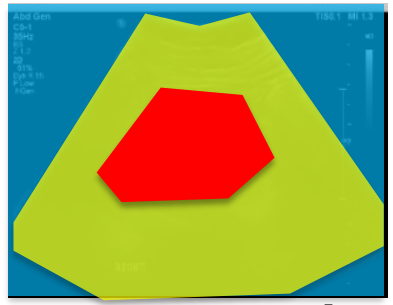


classification is a **special case** of **regression**

https://medium.com/@ali_88273/regression-vs-classification-87c224350d69

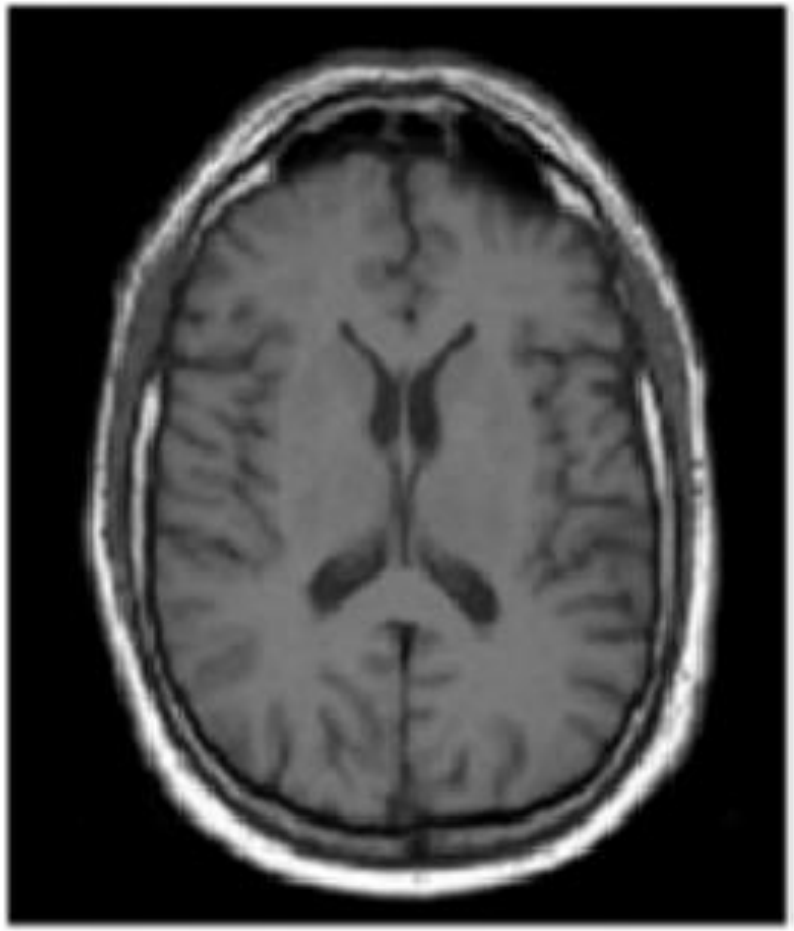
http://napitupulu-jon.appspot.com/posts/regression-ud.html

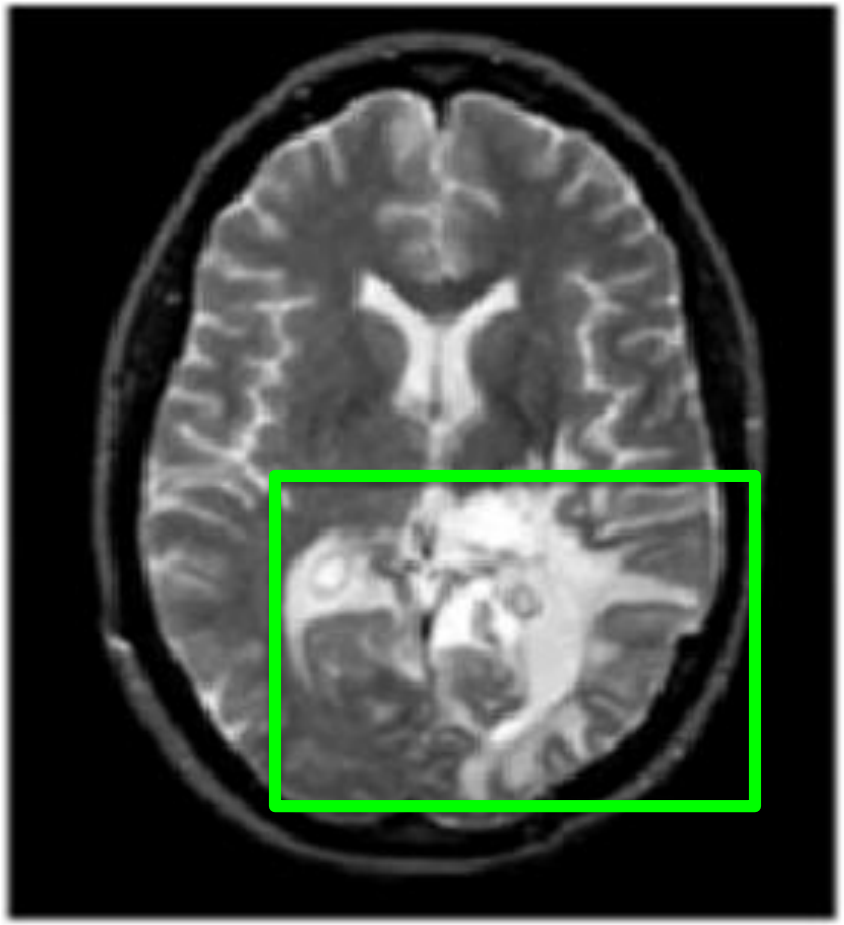
Compare with Computer Vision

	Classification	Classification + Localization	Object Detection	Instance Segmentation	Synthesis
Computer Vision	 CAT	 CAT	 CAT, DOG, DUCK	 CAT, DOG, DUCK	
Medical Image Computing	 Input	 Classification	 Detection	 Segmentation	 Synthesis

https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Applications on Medical Image

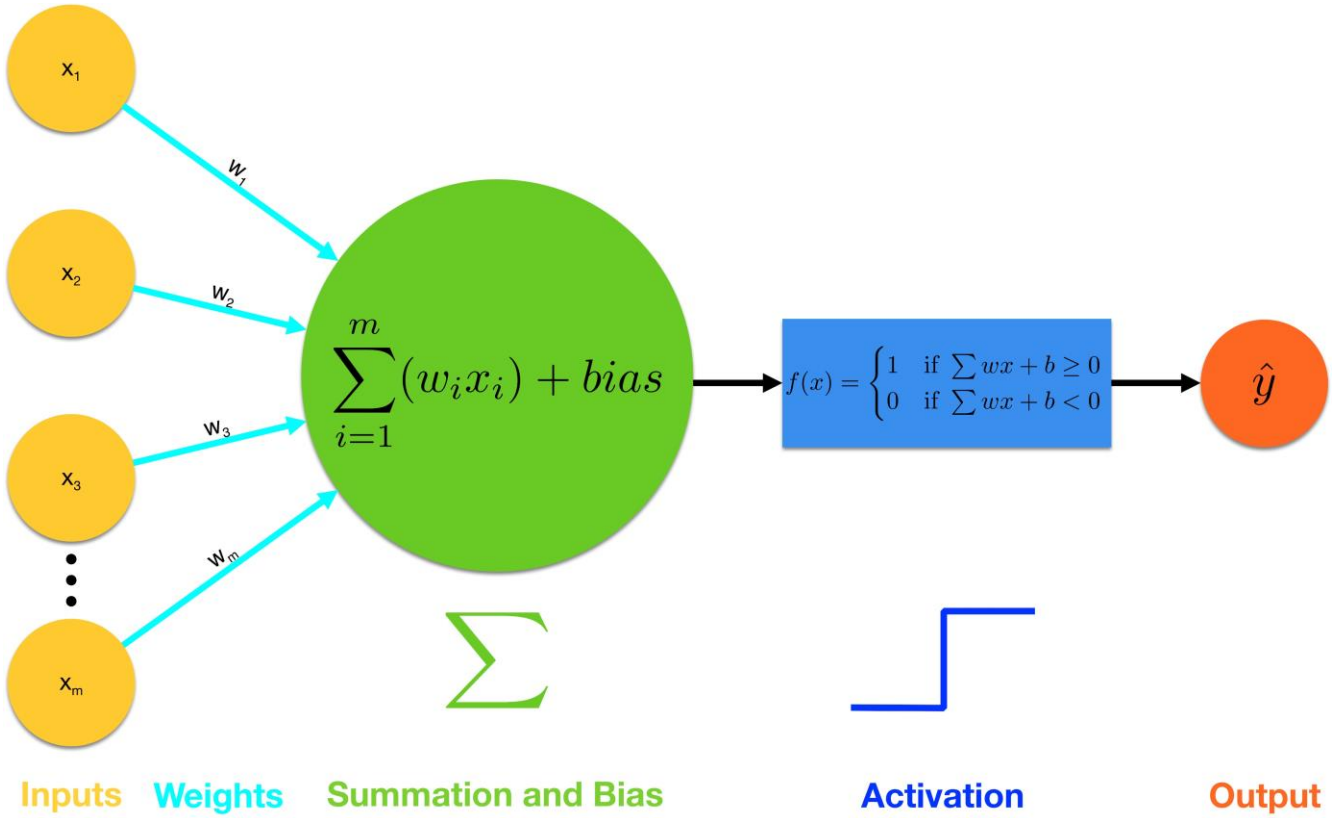




•Chinnu et al. 2015

classification is a **special case** of **regression**

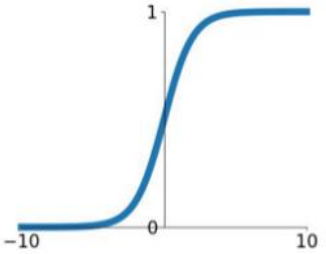
Single Layer Regression



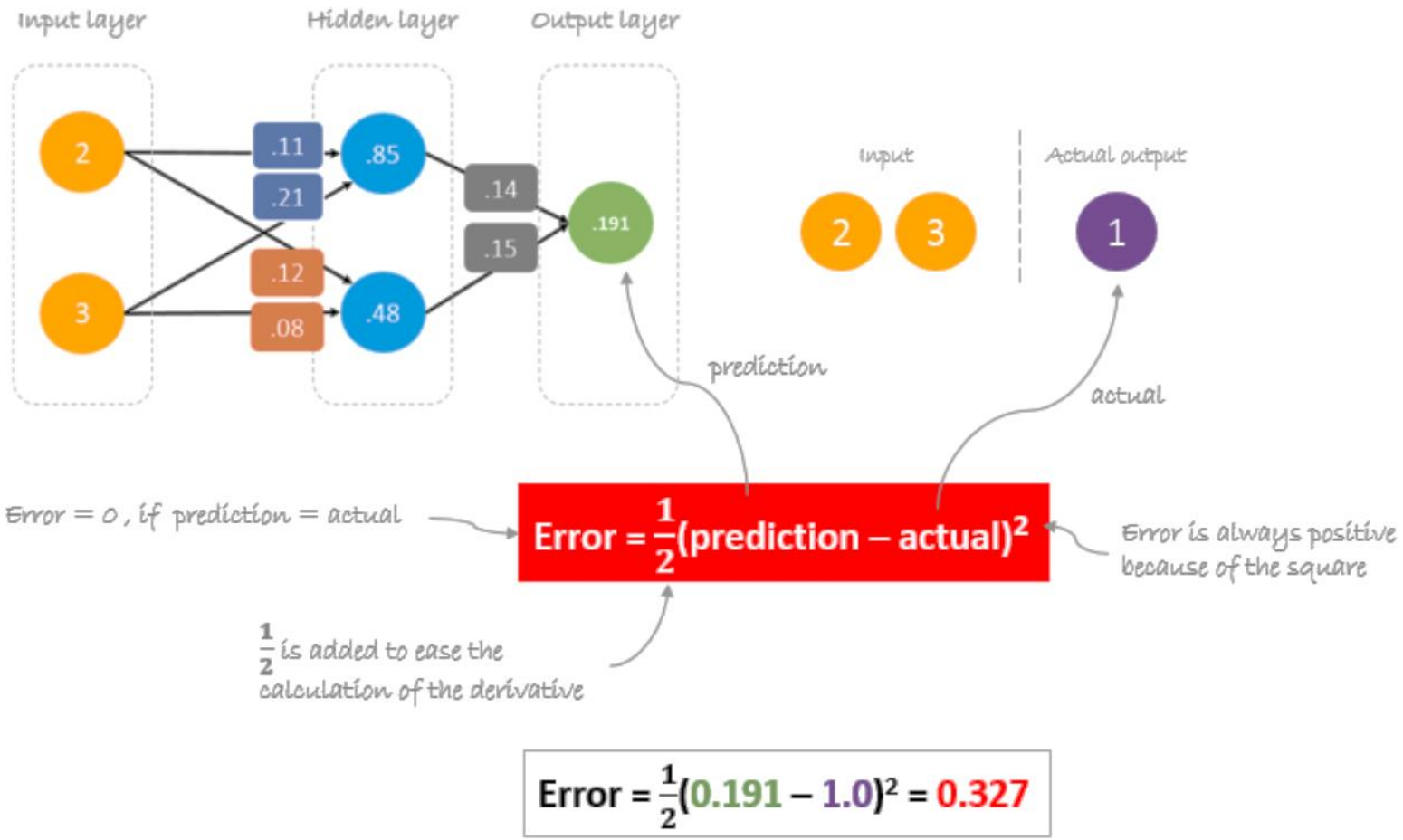
$$\hat{y} = \sigma\left(w^T x + b\right)$$

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

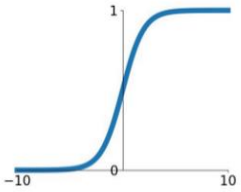


Multiple Layers Regression



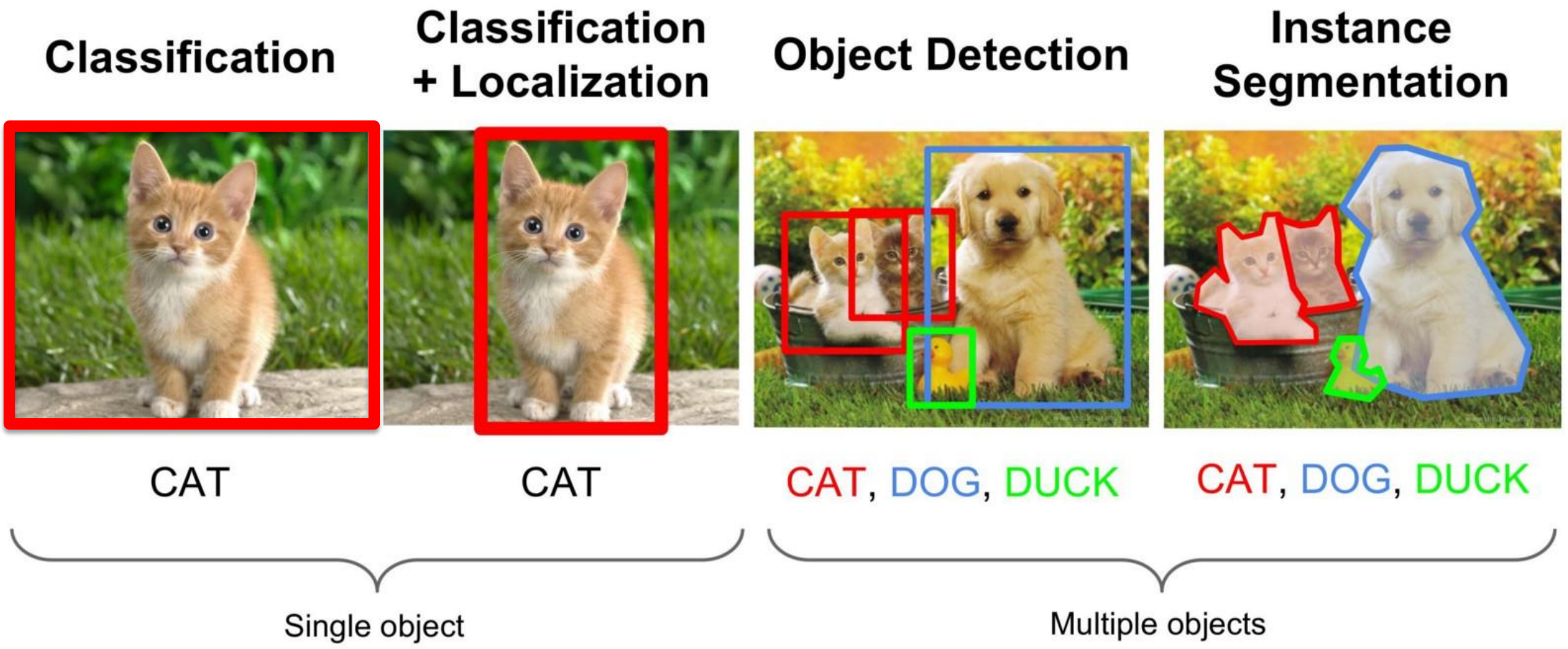
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



http://hmkcode.github.io/ai/backpropagation-step-by-step/

Localization & Detection



https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Localization

Classification + Localization: Task

Classification: C classes
Input: Image
Output: Class label
Evaluation metric: Accuracy

→ CAT

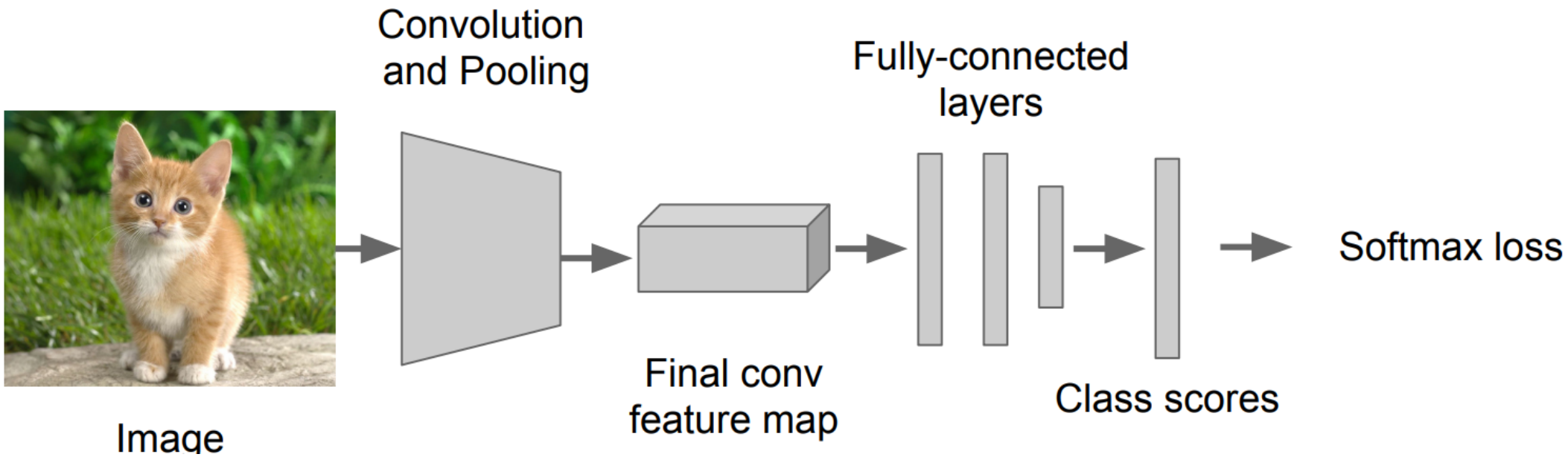
Localization: **Regression + Classification**
Input: Image
Output: Box in the image (x, y, w, h)
Evaluation metric: Intersection over Union

→ (x, y, w, h)

Classification + Localization: Do both

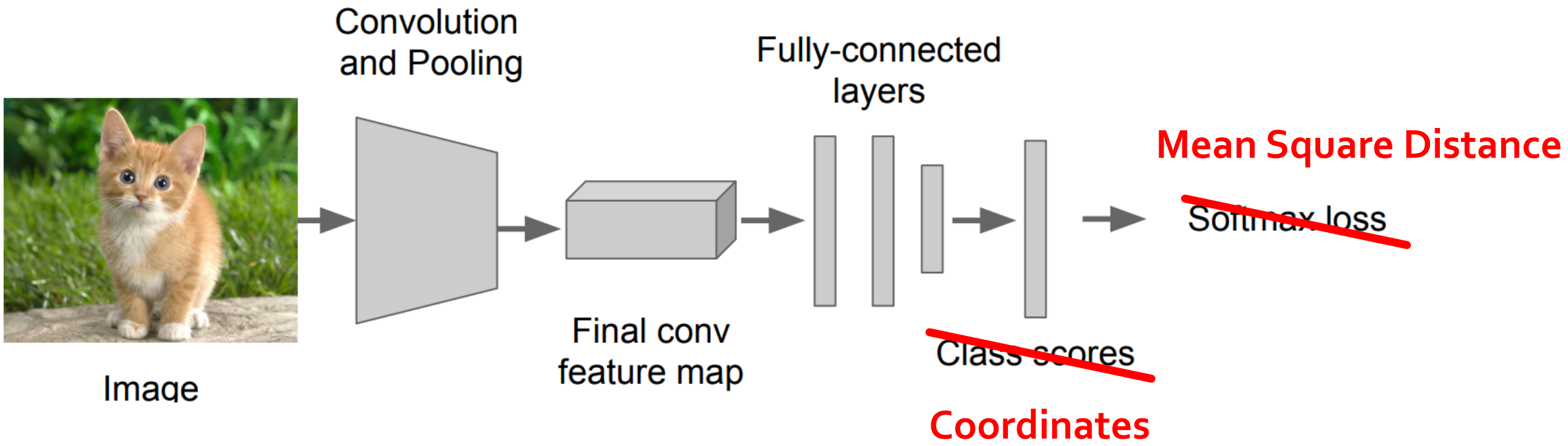
http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Simple Recipe for Classification



http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

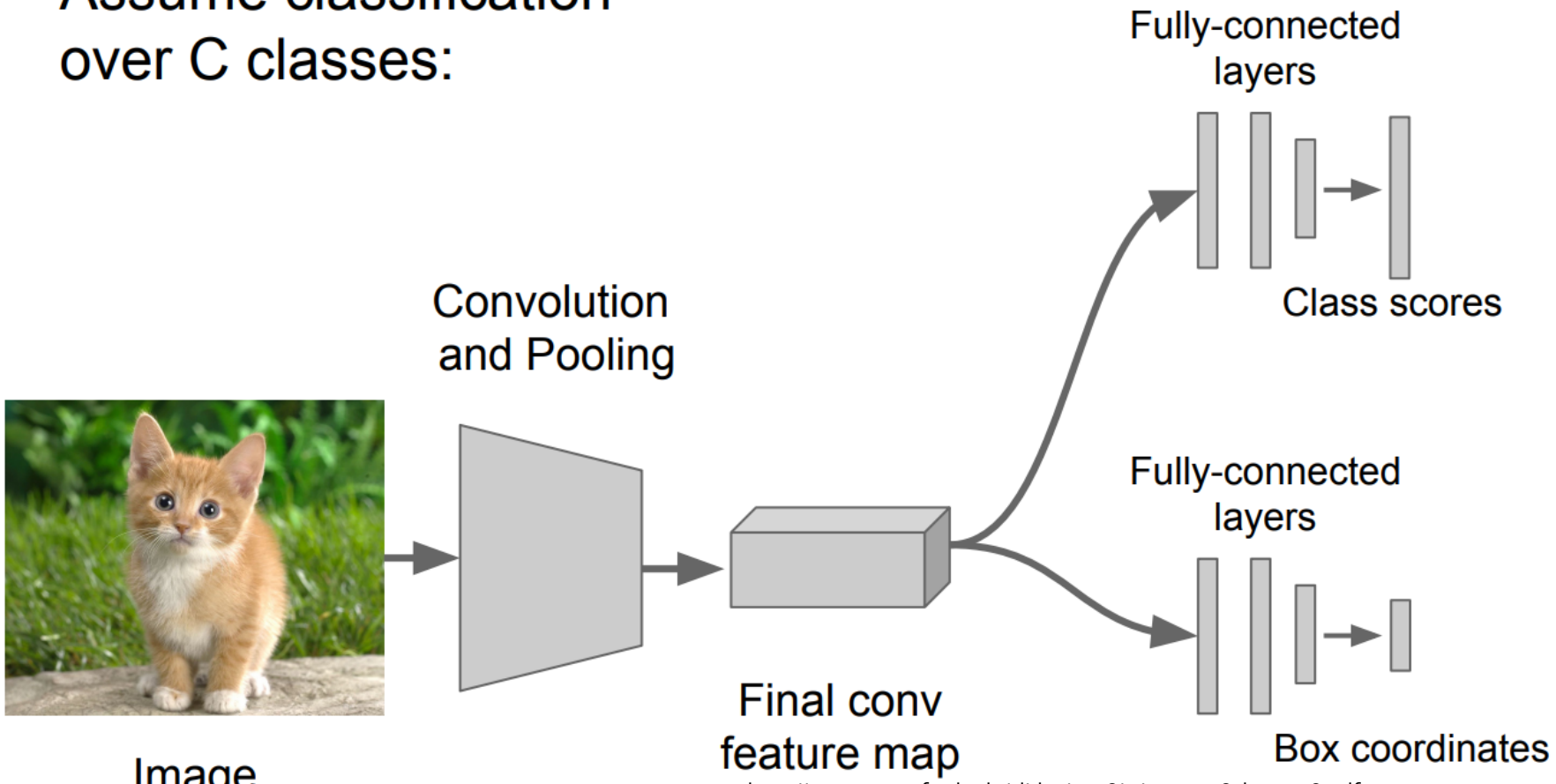
Simple Recipe for Localization



http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Simple Recipe for Classification + Localization

Assume classification over C classes:



Classification head:
C numbers
(one per class)

Class agnostic:
4 numbers
(one box)

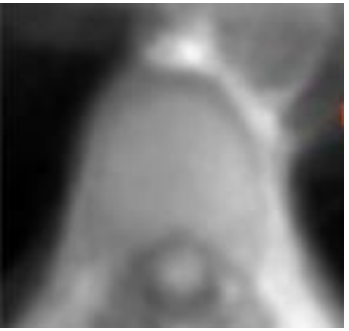
http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Single Landmark Detection

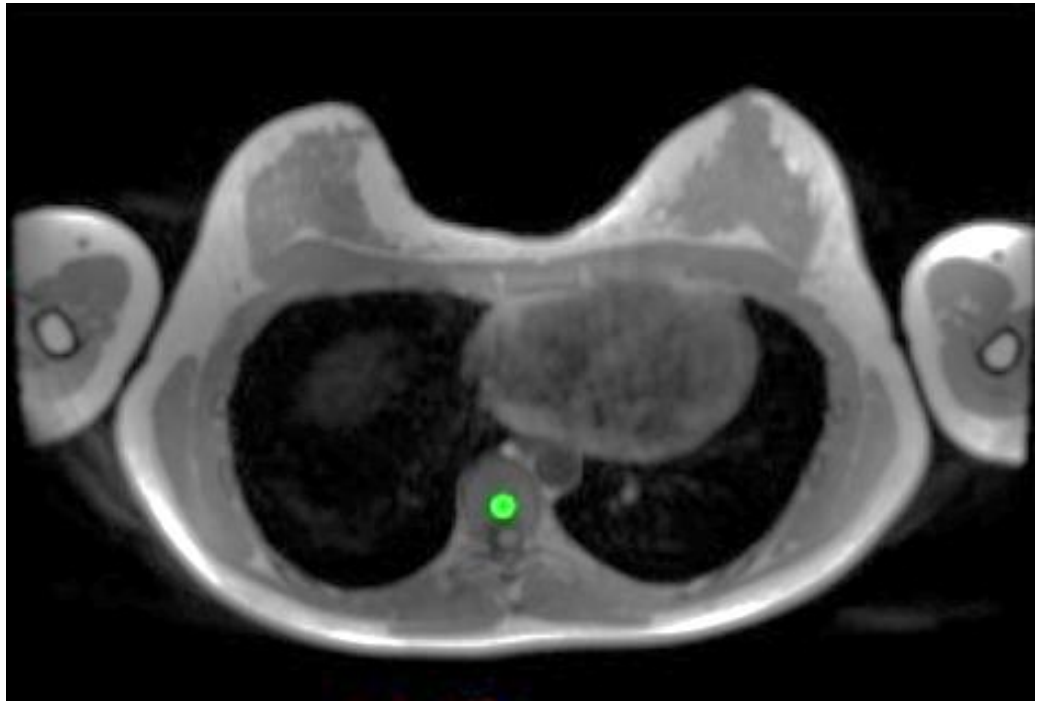
Center Point (x, y)

Bounding Box (x1, y1,x2,y2)

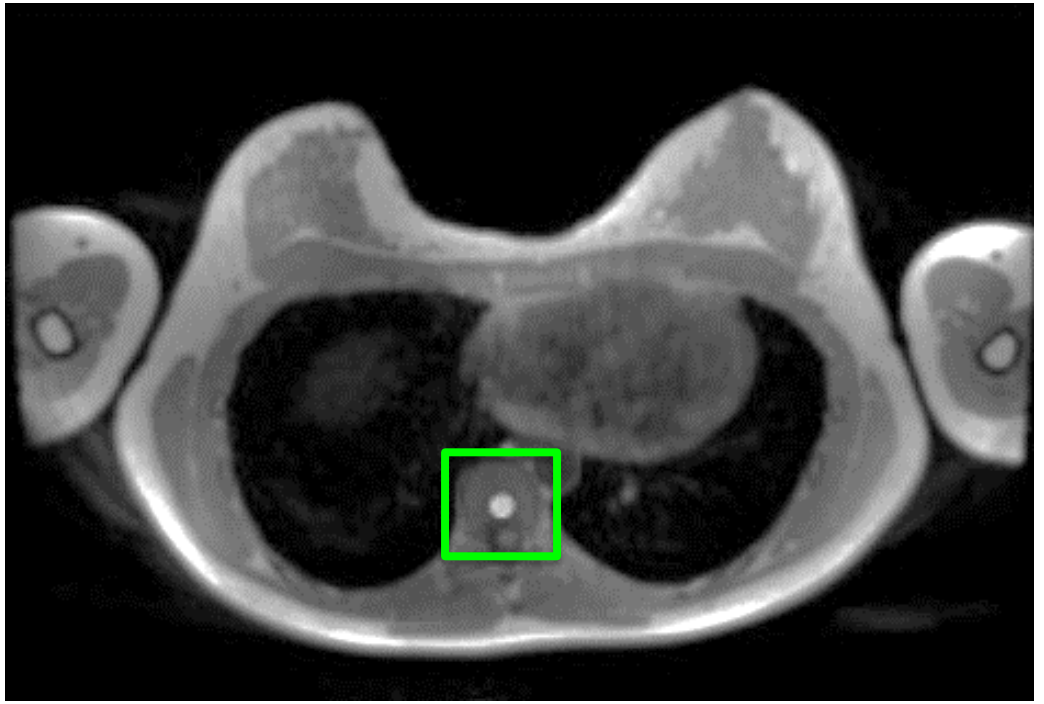
Single Landmark Detection



Vertebra centre template



Center Point (x, y)



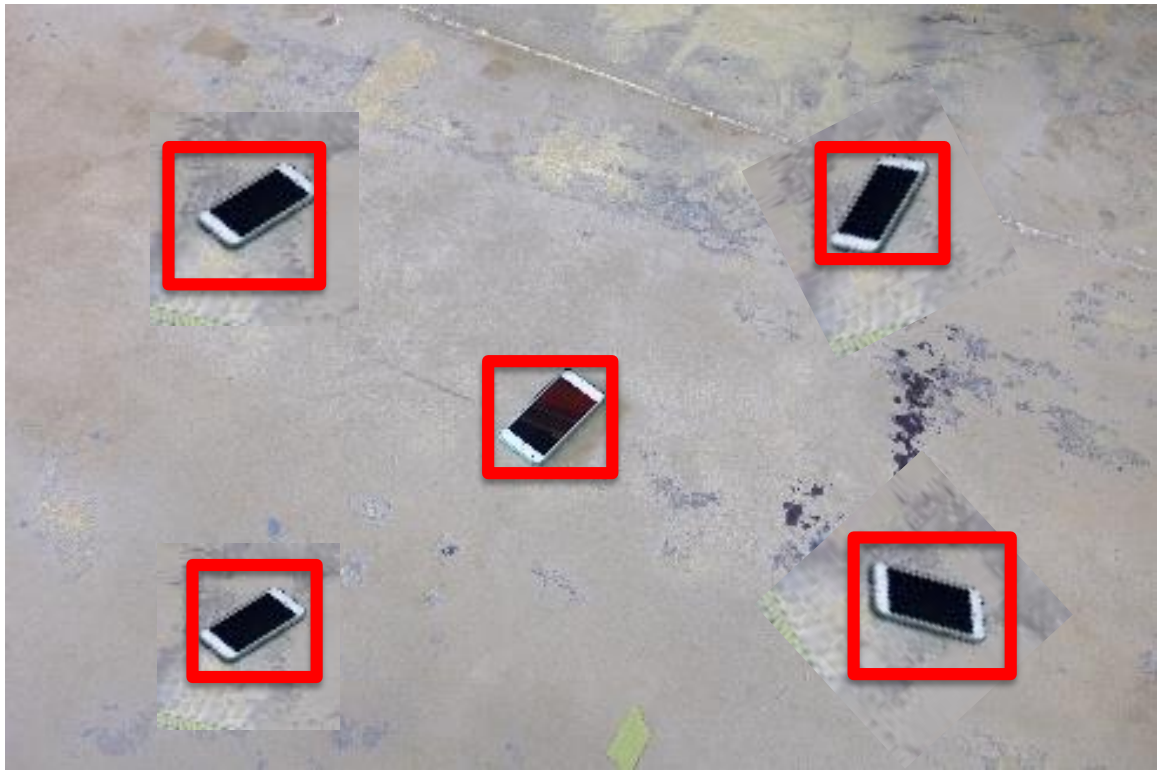
Bounding Box (x1, y1,x2,y2)

https://www.slideshare.net/habibbaluwala/landmark-detection-and-statistical-shape-modelling-miccai-2014-cbm-workshop

Now, what to do?

Multi Landmark Detection

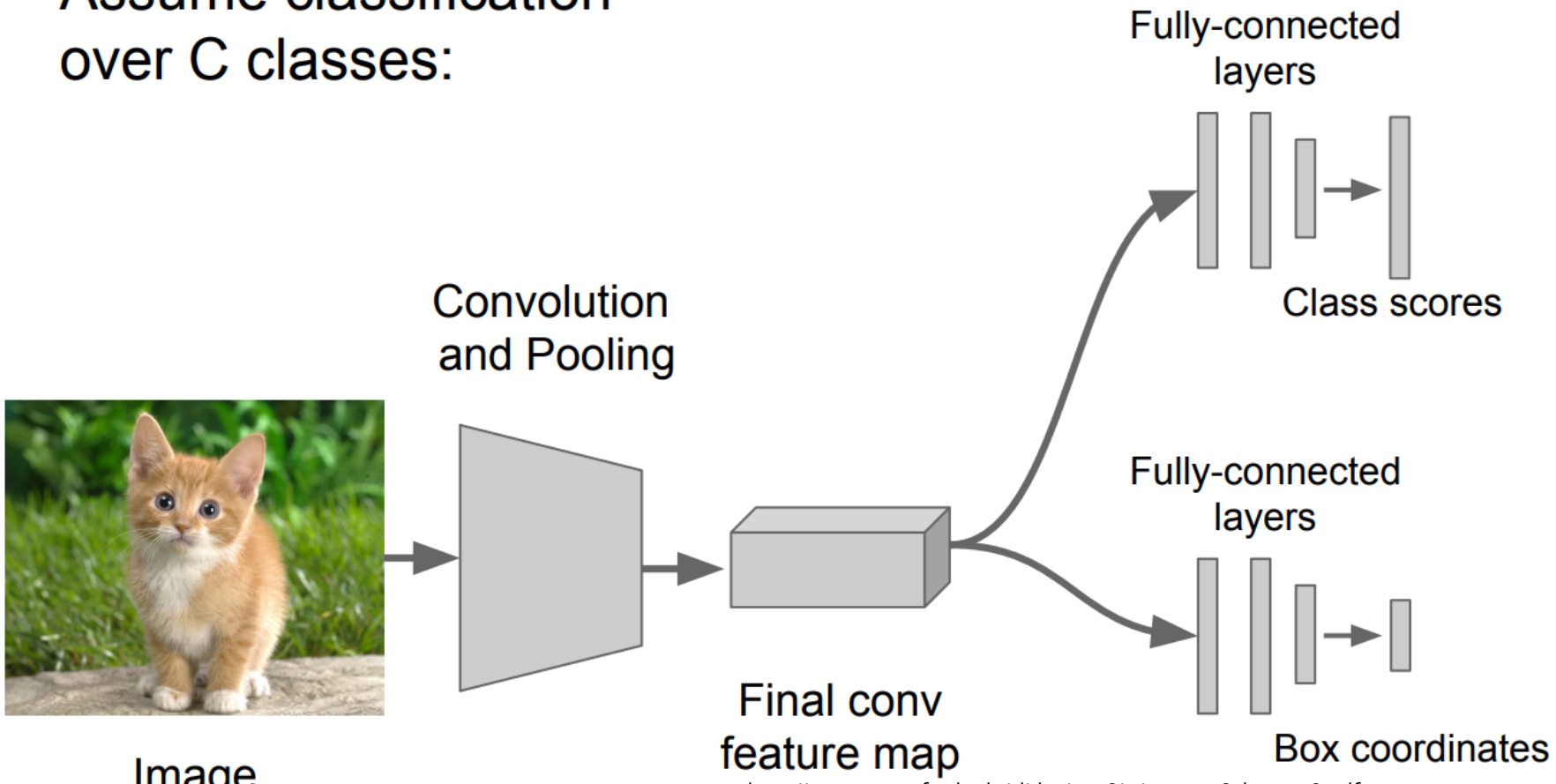
Center Points (x1, y1, x2, y2 …)



Bounding Boxs (x11, y11,x12,y12 x21,y21,x22,y22 …)

More Than 1 Object

Assume classification over C classes:

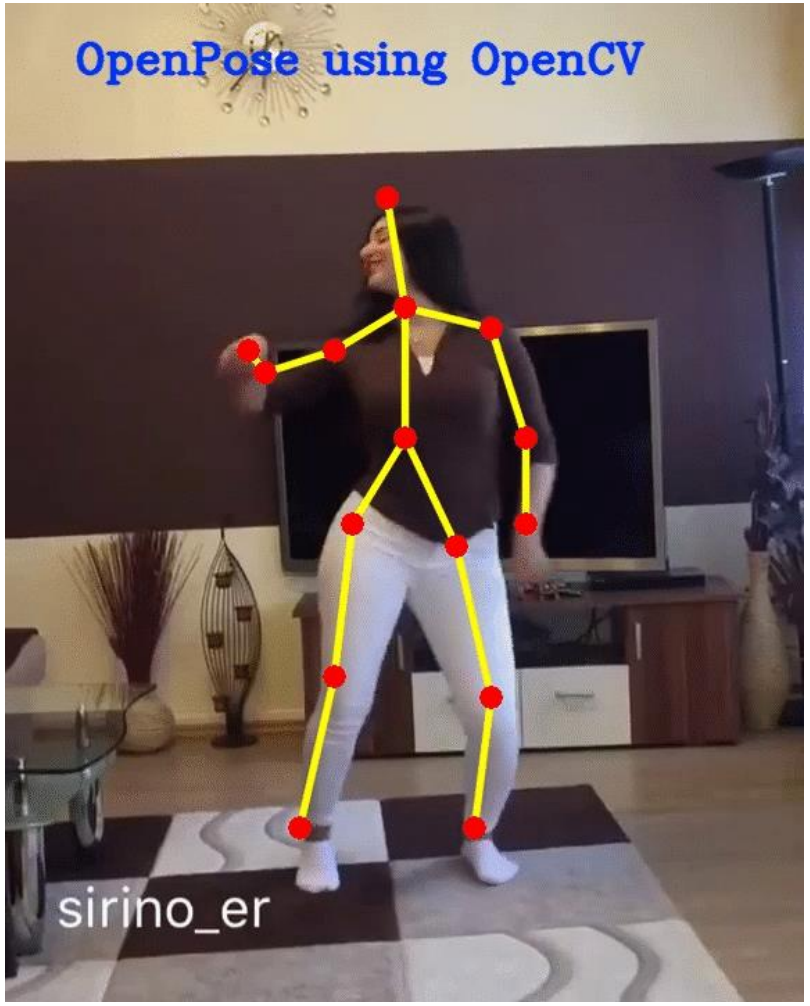


Classification head:
C numbers
(one per class)

Class specific:
C x 4 numbers
(one box per class)

http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

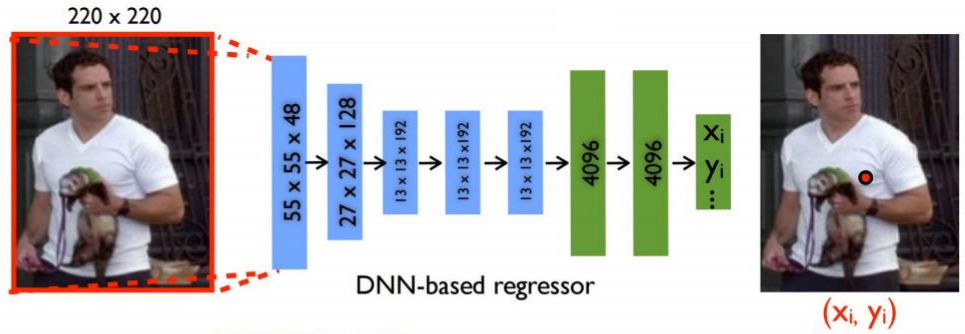
Pose Estimation (Multiple-Points)



Represent a person by K joints

Regress (x, y) for each joint from last fully-connected layer of AlexNet

(Details: Normalized coordinates, iterative refinement)

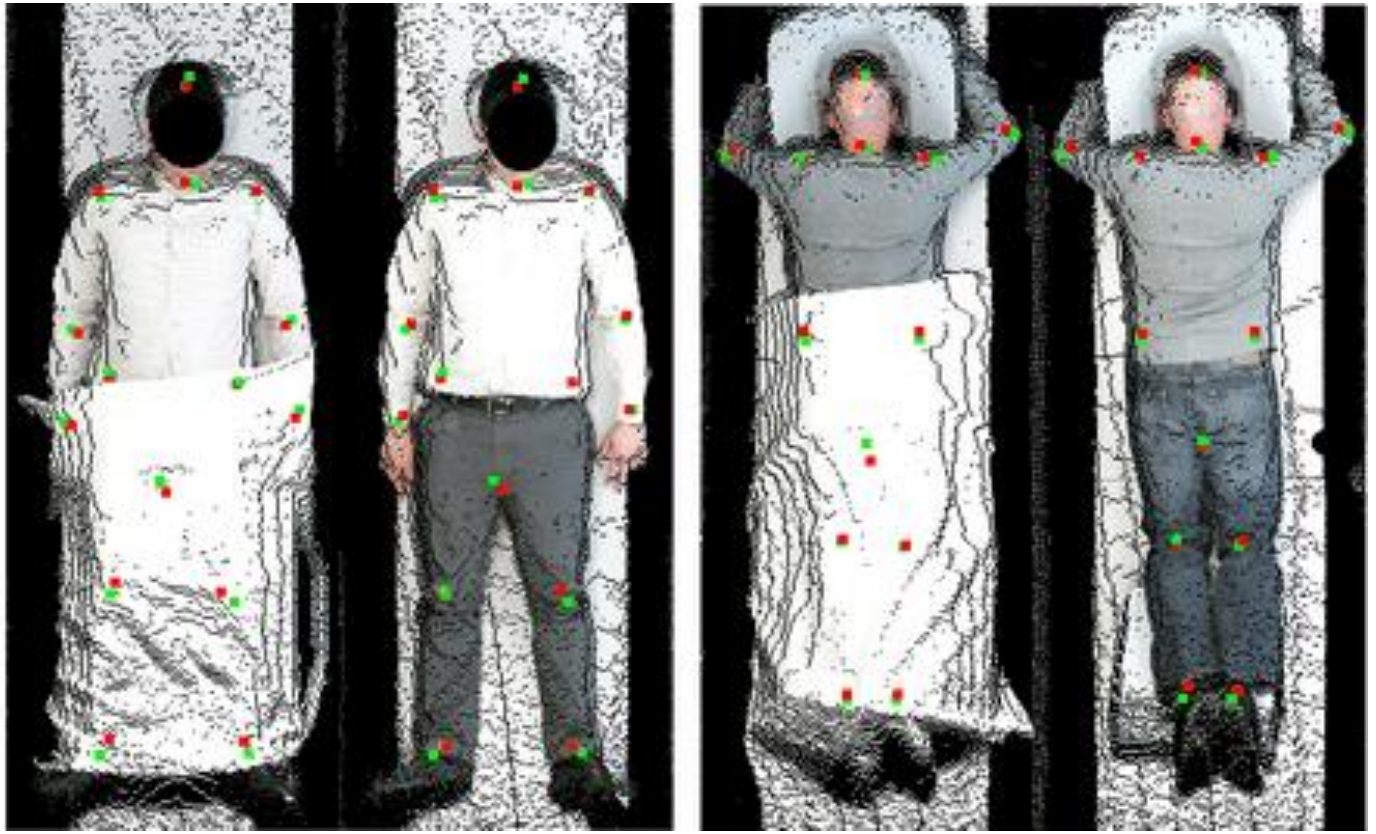


v and Szegedy, "DeepPose: Human Pose
ıtion via Deep Neural Networks", CVPR 2014

https://www.learnopencv.com/deep-learning-based-human-pose-estimation-using-opencv-cpp-python/
http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Pose Estimation (Multiple-Points)

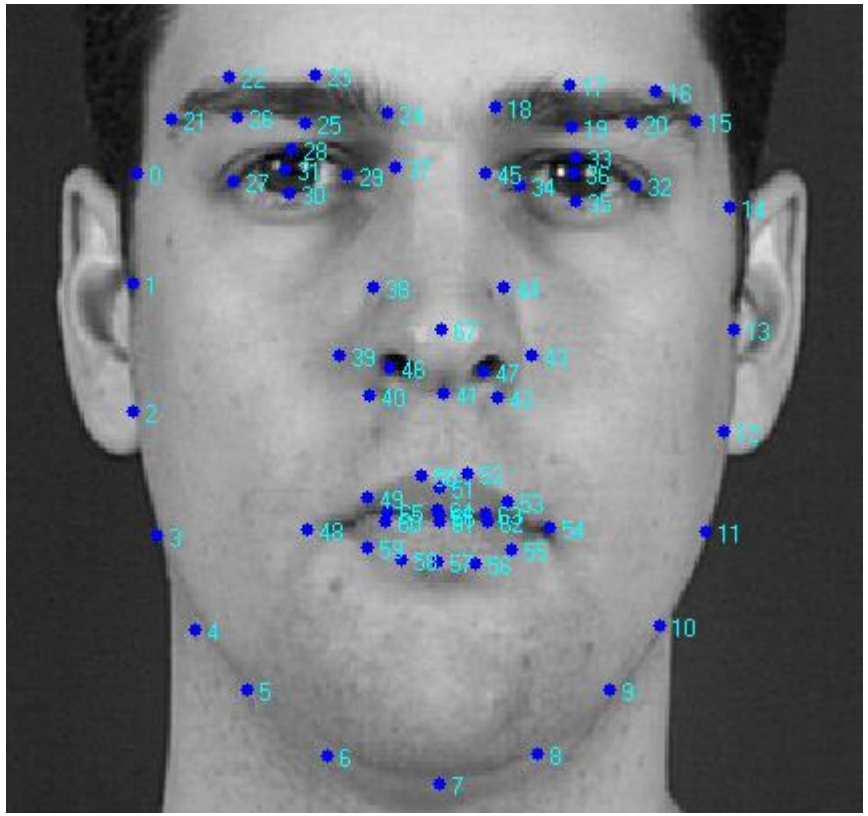
Singh et al. (2014)



Singh et al. (2017)

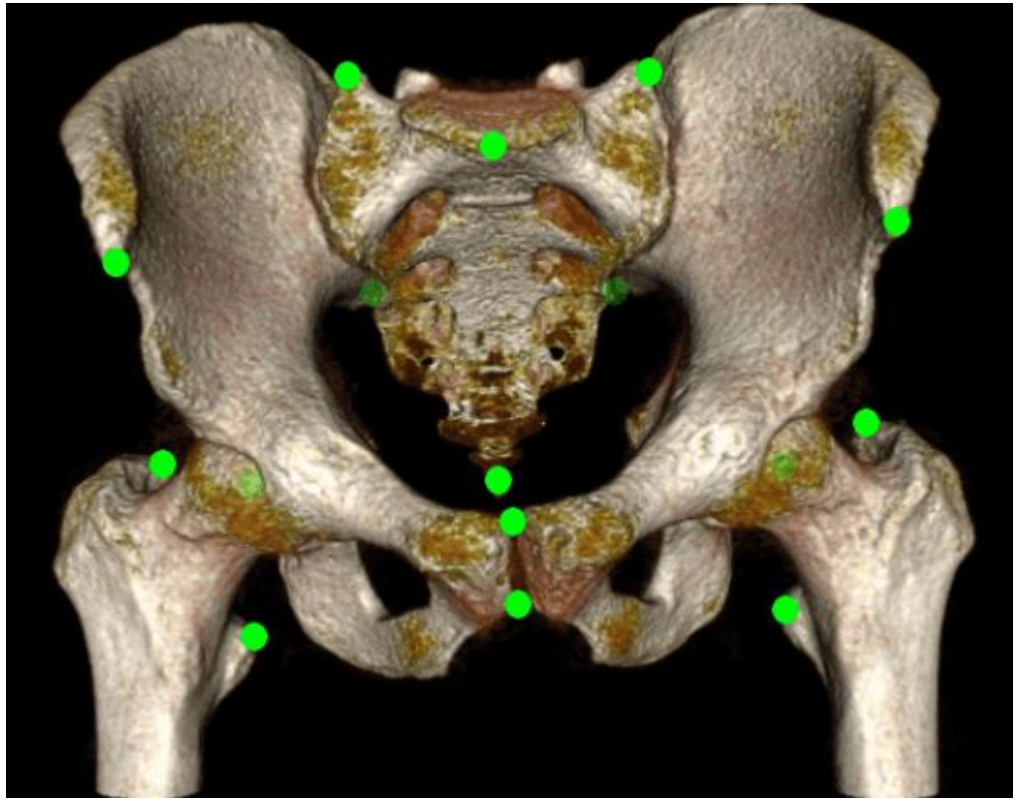
Landmark Detection (Multiple-Points)

https://gfycat.com/gifs/detail/SpitefulAffectionateAllosaurus

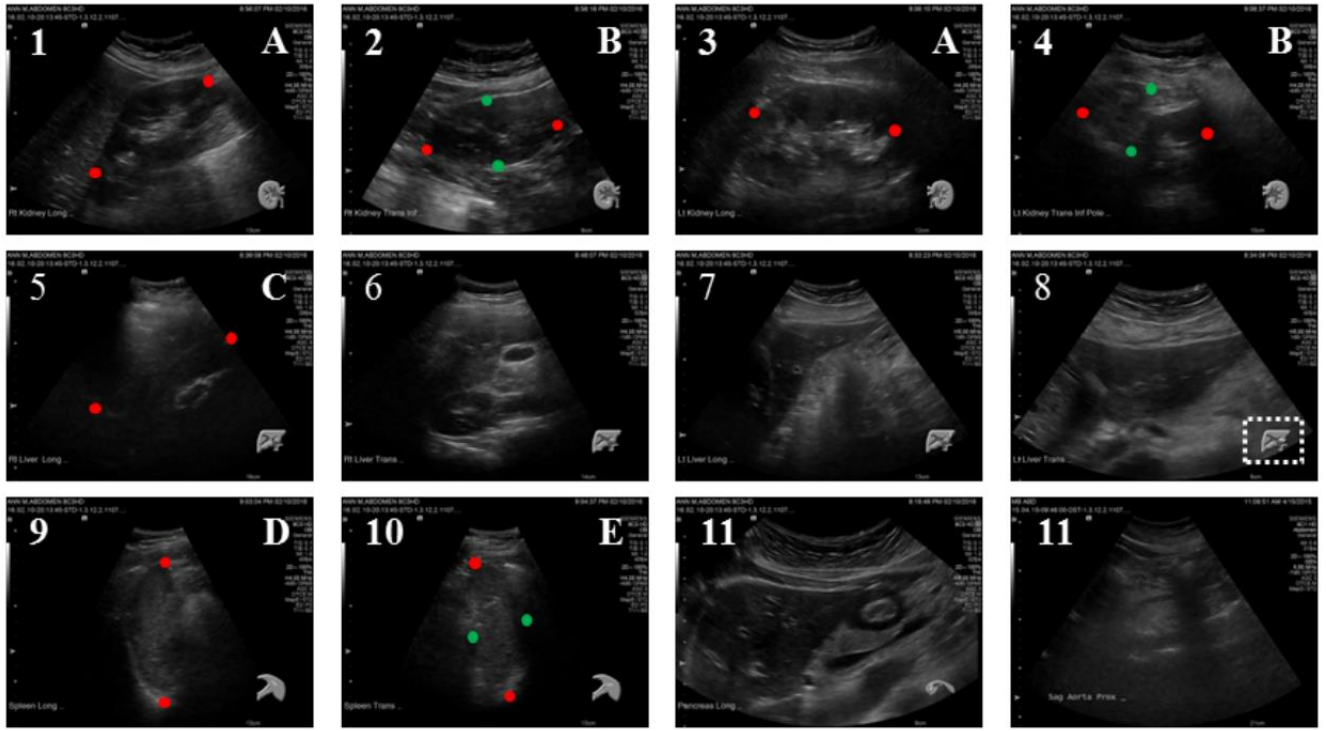


https://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/xm2vts/xm2vts_markup.html

Landmark Detection (Multiple-Points)



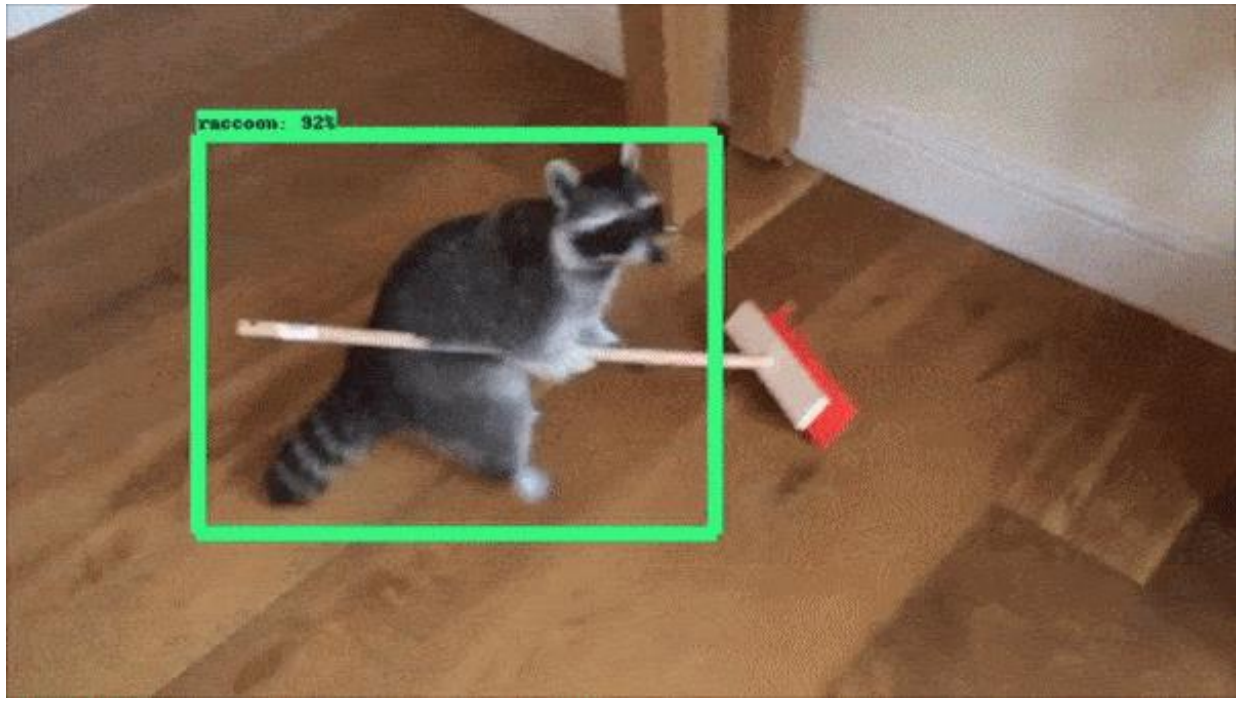
https://medtech4health.se/aida/projekt/

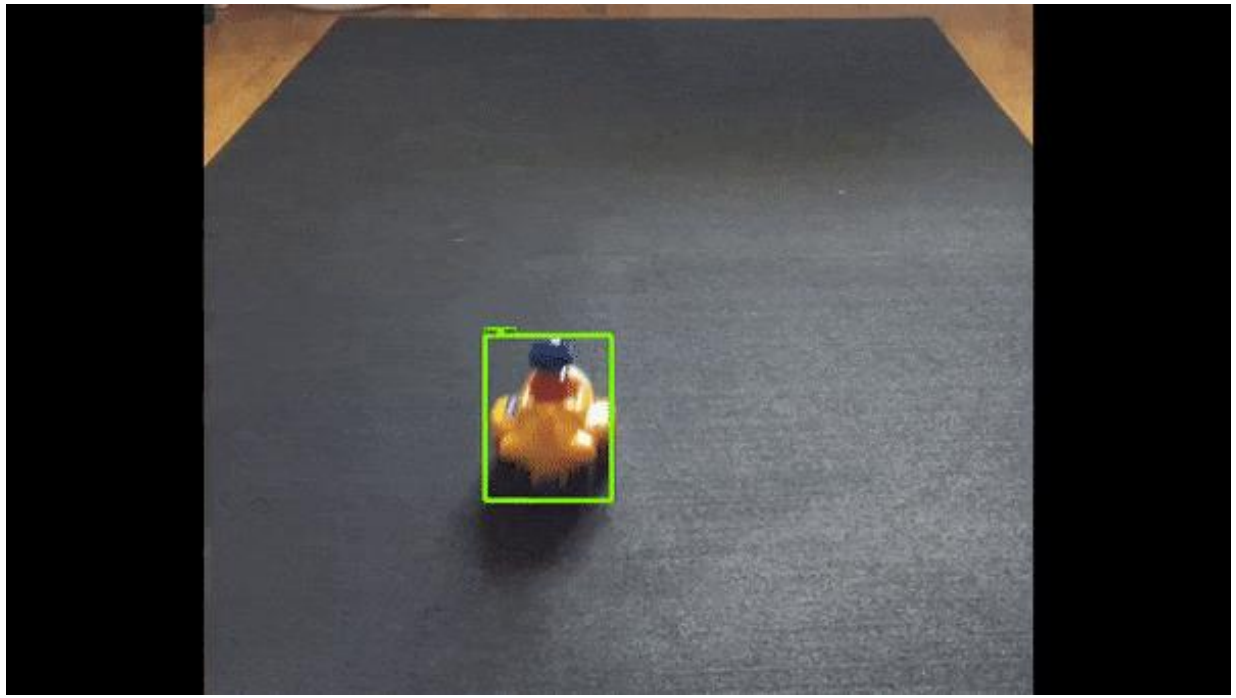


Landmark Detection

A. Kidney Long **B**. Kidney Trans **C**. Liver Long **D**. Spleen Long **E**. Spleen Trans

Single Object Detection (Bounding Box)



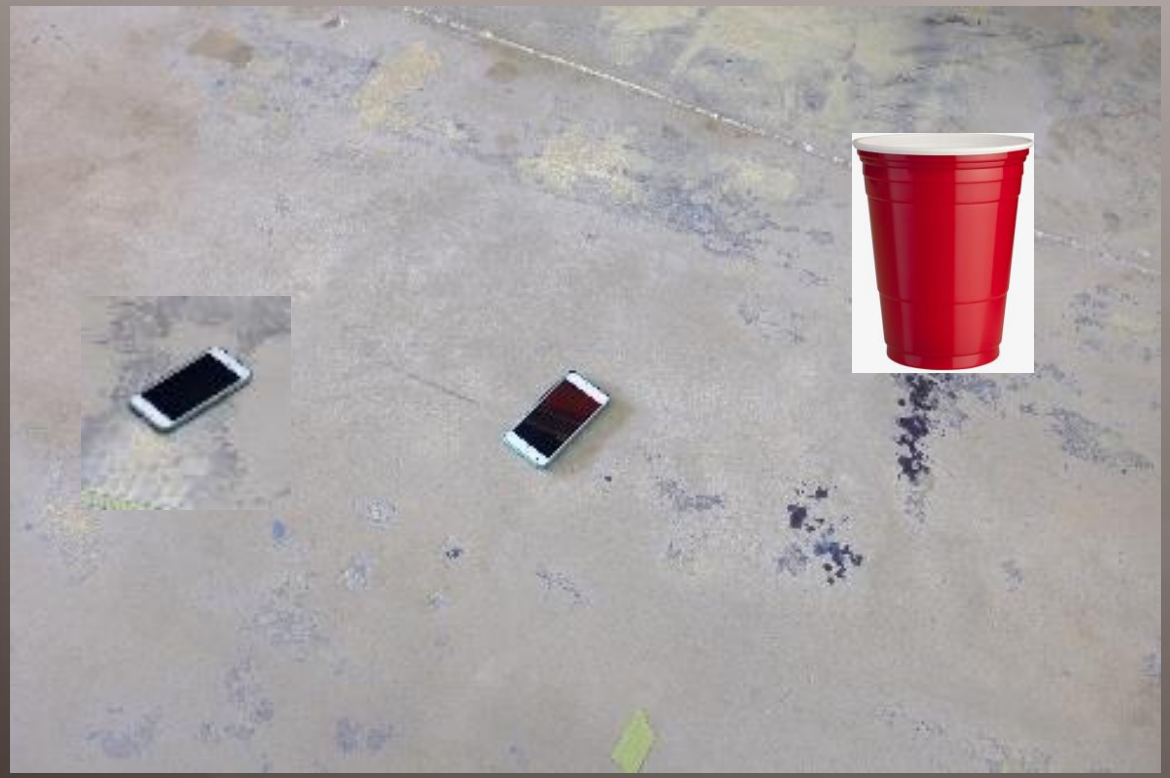


https://towardsdatascience.com/how-to-train-your-own-object-detector-with-tensorflows-object-detector-api-bec72ecfe1d9

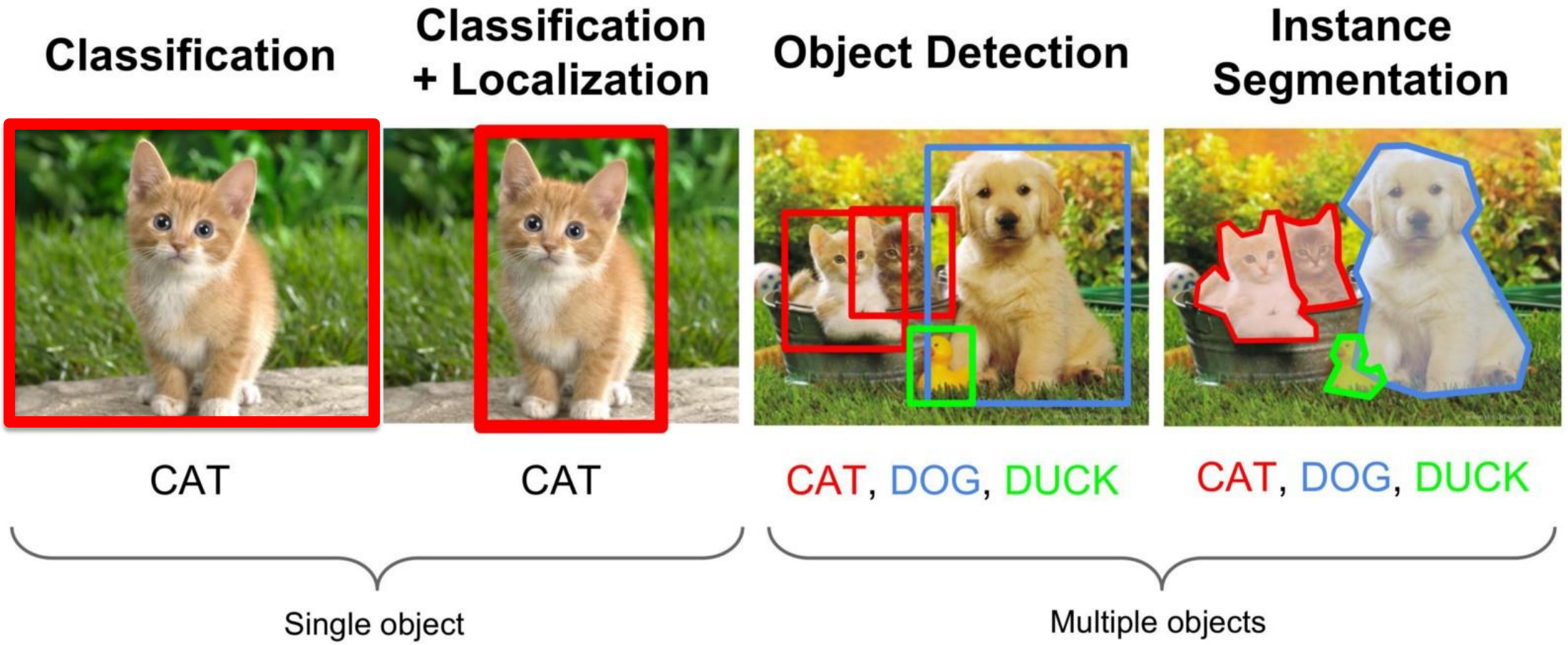
https://towardsdatascience.com/building-a-toy-detector-with-tensorflow-object-detection-api-63c0fdf2ac95

Now, what to do?

Now, what to do?



Localization & Detection



https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Detection as Regressioon

DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h)

= 16 numbers

Detection as Regression

DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h)

= 16 numbers

DOG, (x, y, w, h)
CAT, (x, y, w, h)

= 8 numbers

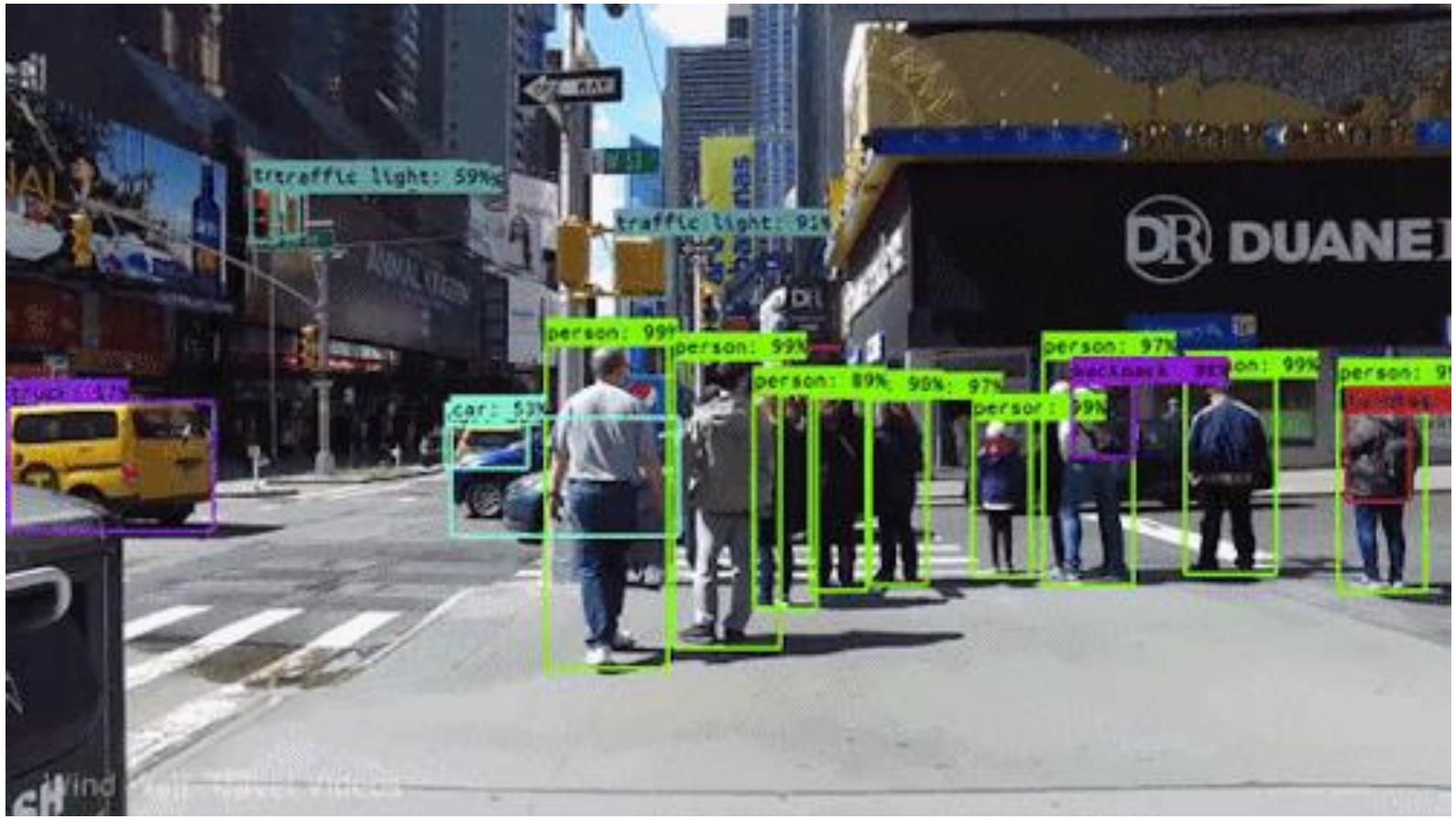
http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

Detection as Regression

CAT, (x, y, w, h)
CAT, (x, y, w, h)
....
CAT (x, y, w, h)

= many numbers

Detection



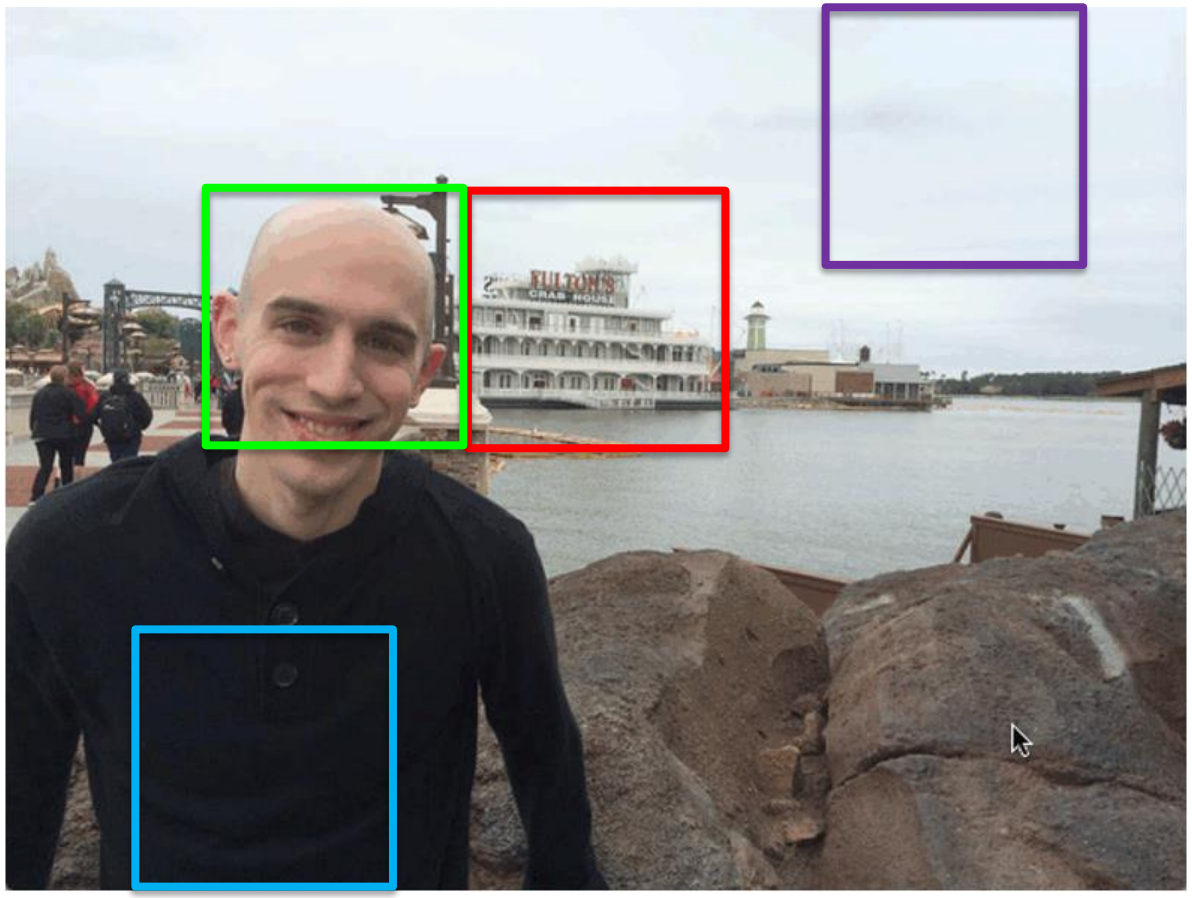
https://mc.ai/deep-learning-for-object-detection-from-the-start-to-the-state-of-the-art-2-2/

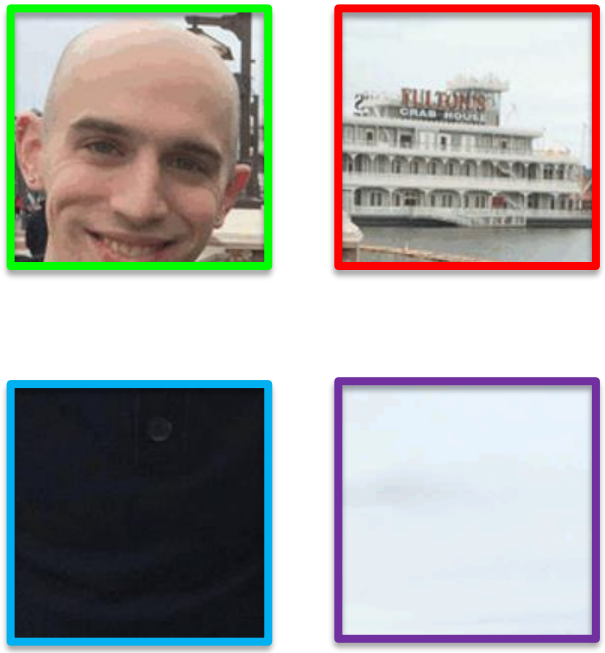
everything can be solved by classification

Idea 1: Sliding Window

Detect Face

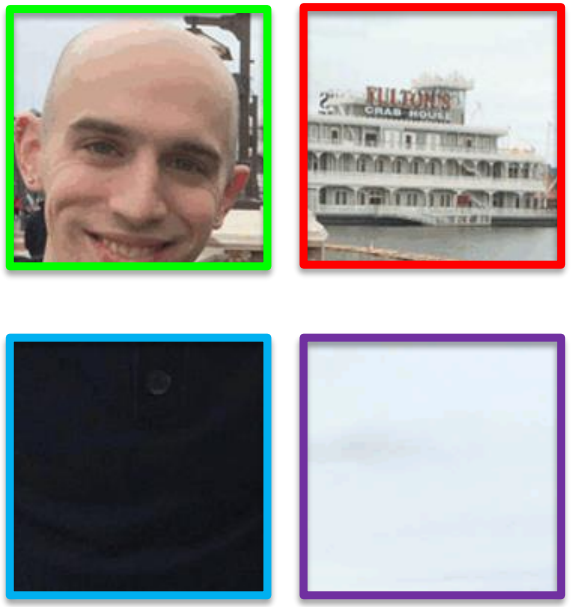
Get Patches

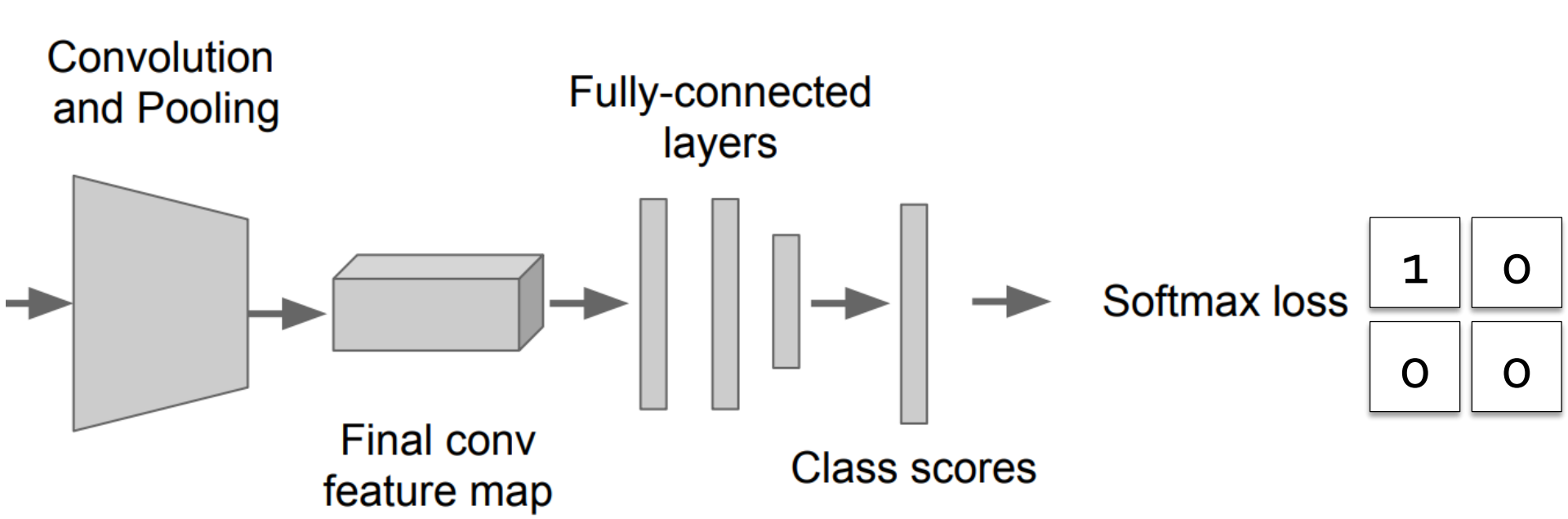




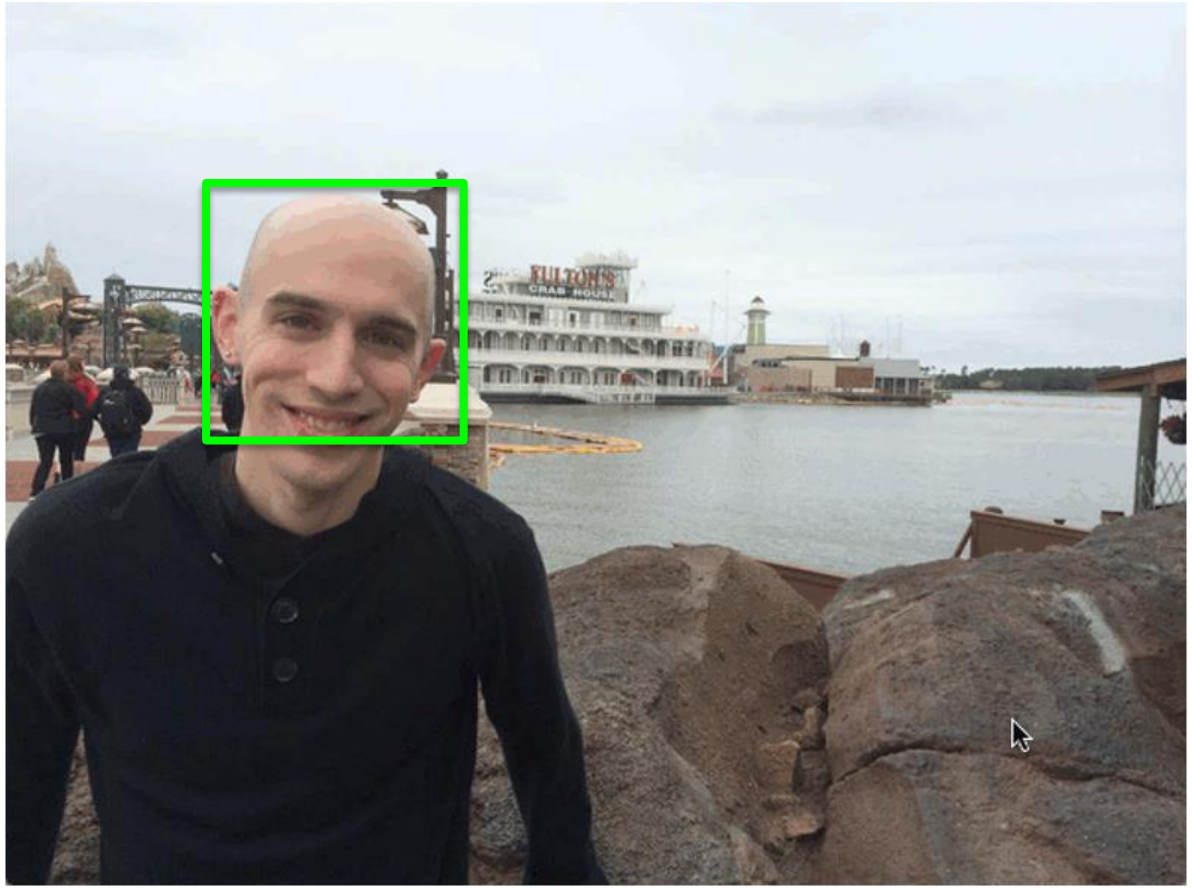
Classification

"CONV-POOL-FC"



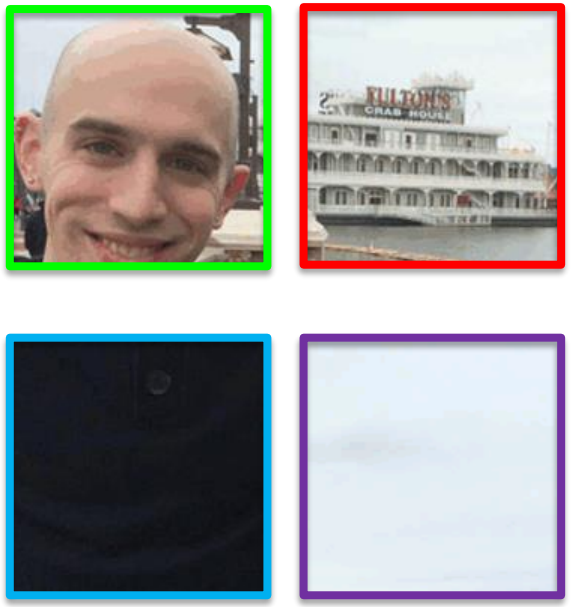


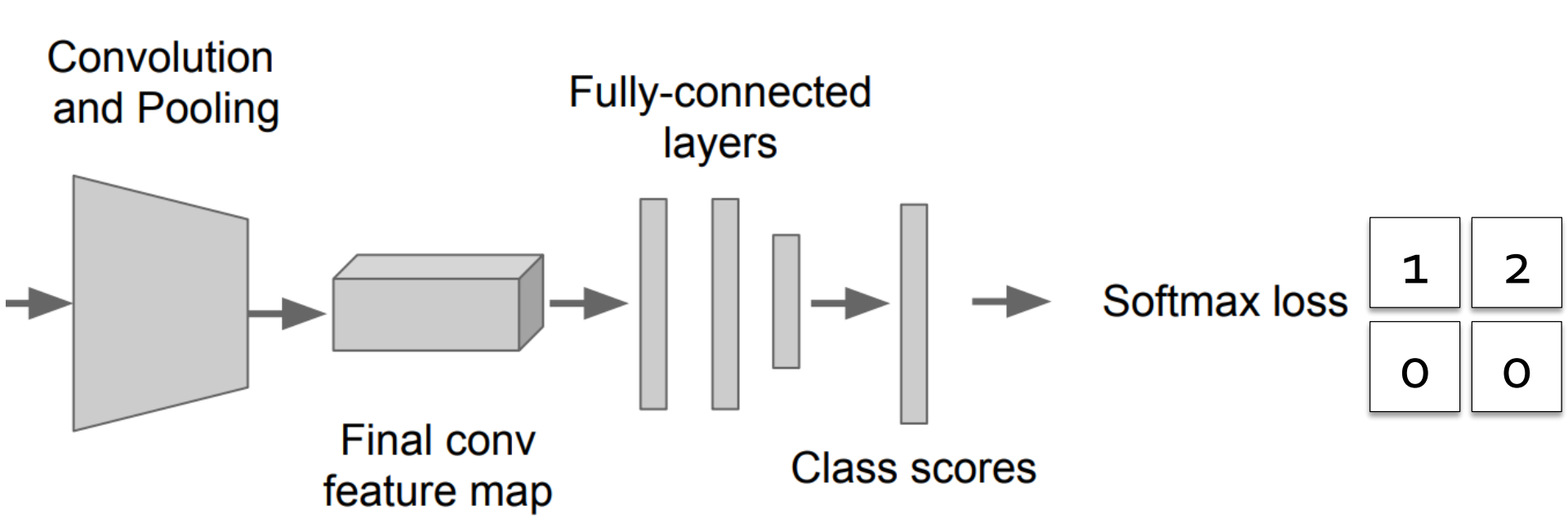
Face



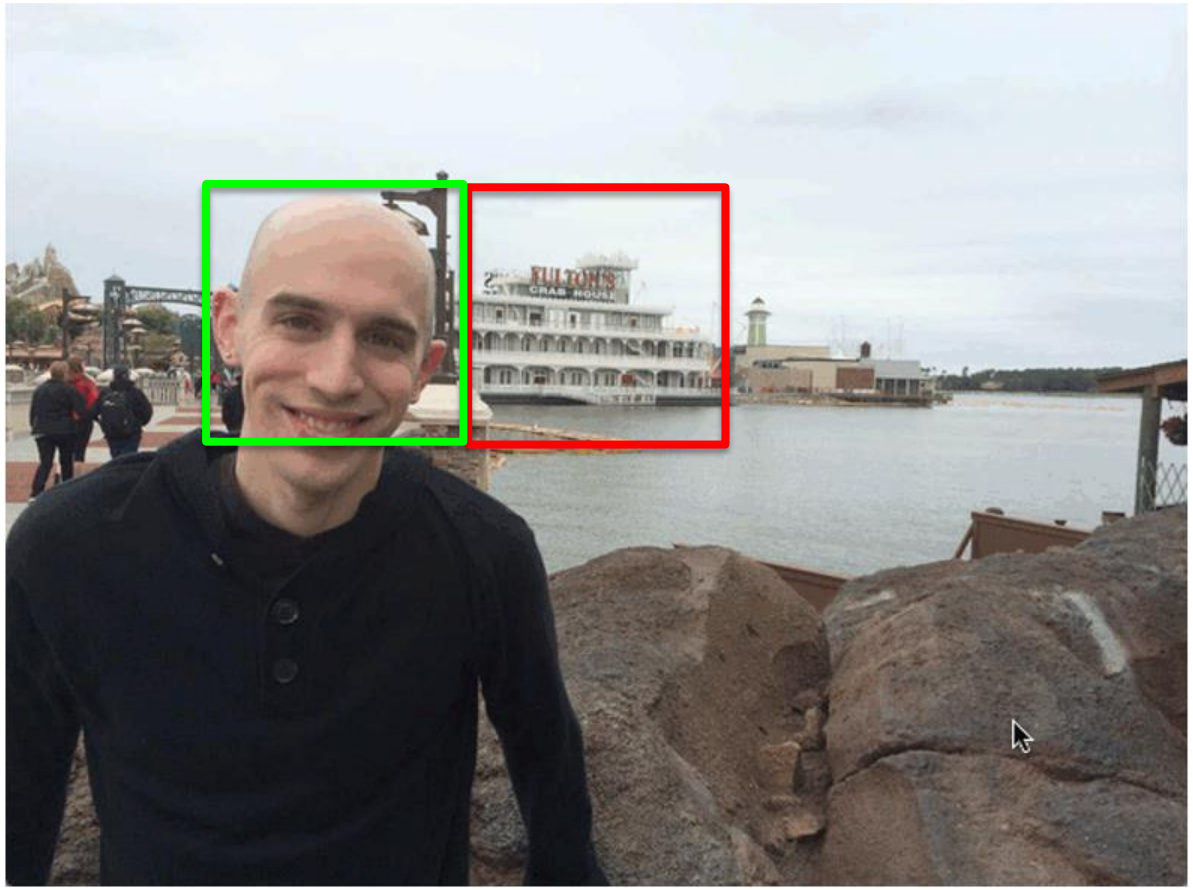
Classification

"CONV-POOL-FC"





Face + Boat

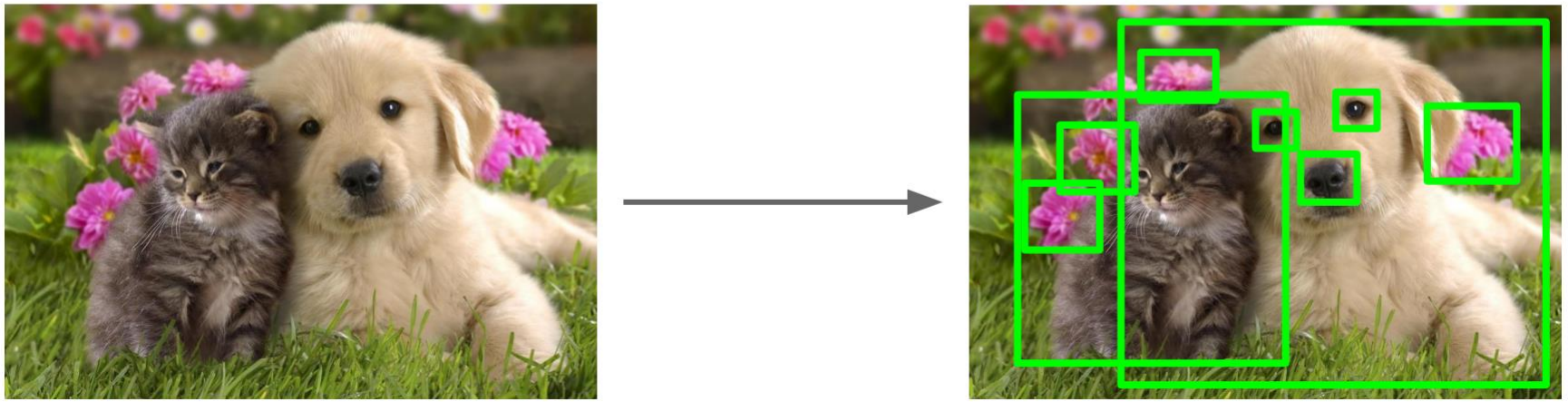


Problem?

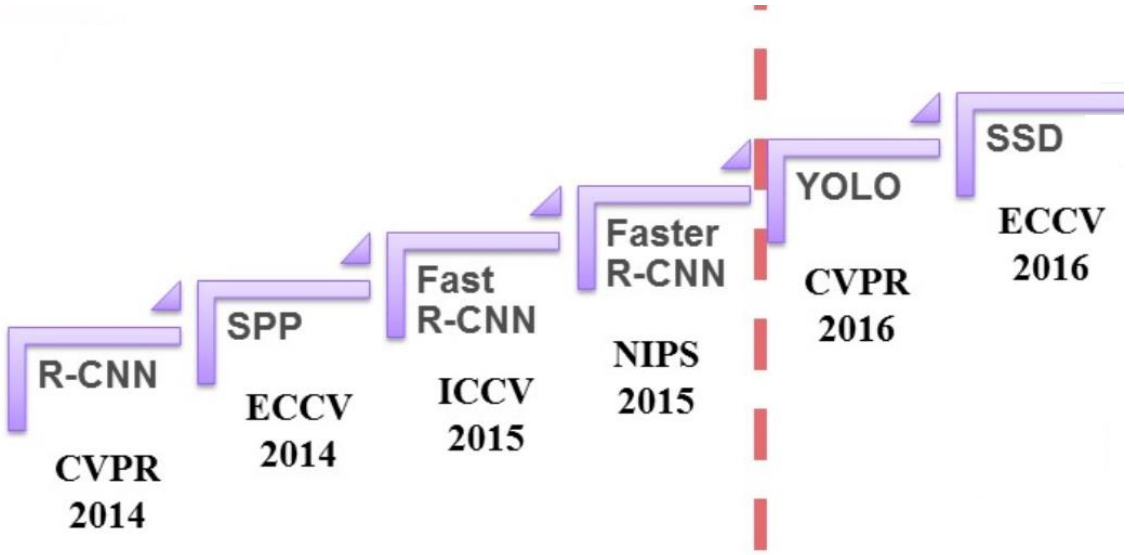
Idea 2. Region Proposal

Region Proposals

- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



R-CNN Family



R-CNN

Rich feature hierarchies for accurate object detection and semantic ...
https://arxiv.org › cs
by R Girshick - 2013 - Cited by 5968 - Related articles
Nov 11, 2013 - We also compare **R-CNN** to OverFeat, a recently proposed sliding-window detector based on a similar CNN ... From: **Ross Girshick** [view email]

SPP

Spatial Pyramid Pooling in Deep Convolutional Networks for Visual ...
https://arxiv.org › cs
by K He - 2014 - Cited by 1228 - Related articles
Jun 18, 2014 - The new network structure, called **SPP-net**, can generate a fixed-length representation regardless of image size/scale. Pyramid pooling is also ...

Fast R-CNN

Fast R-CNN
https://arxiv.org › cs
by R Girshick - 2015 - Cited by 3173 - Related articles
Apr 30, 2015 - Fast **R-CNN** trains the very deep VGG16 network 9x faster than **R-CNN**, is 213x faster at test-time, and ... From: **Ross Girshick** [view email]

Faster R-CNN

Faster R-CNN: Towards Real-Time Object Detection with Region ...
https://arxiv.org › cs
by S Ren - 2015 - Cited by 4313 - Related articles
Jun 4, 2015 - Advances like SPPnet and **Fast R-CNN** have reduced the running time of these detection

Mask R-CNN

Mask R-CNN
https://arxiv.org › cs
by K He - 2017 - Cited by 633 - Related articles
Mar 20, 2017 - The method, called **Mask** R-**CNN**, extends Faster R-**CNN** by adding a branch for predicting an object **mask** in parallel with the existing branch ...

YOLO

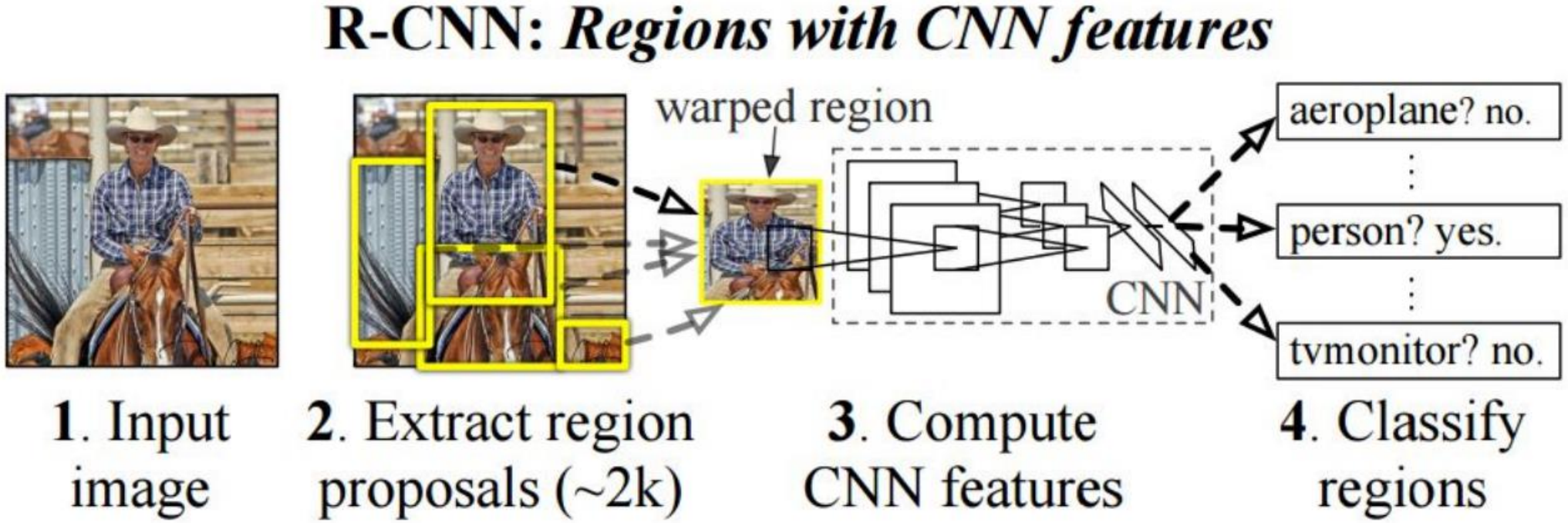
You only look once: Unified, real-time object detection
J Redmon, S Divvala, R Girshick… - Proceedings of the IEEE …, 2016 - cv-foundation.org
We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class …
☆ 99 Cited by 2060 Related articles All 21 versions »

SSD

Ssd: Single shot multibox detector
W Liu, D Anguelov, D Erhan, C Szegedy… - European conference on …, 2016 - Springer
… 2 The Single Shot Detector (**SSD**). This section describes our proposed **SSD** framework for **detection** (Sect … Open image in new window. To understand the performance of our two **SSD** models in more details, we used the **detection** analysis tool from [19] …
☆ 99 Cited by 1707 Related articles All 20 versions

https://blog.csdn.net/a8039974/article/details/78841746

R-CNN



Stage 1: Region Proposal

Stage 2: Classification

https://arxiv.org/abs/1311.2524

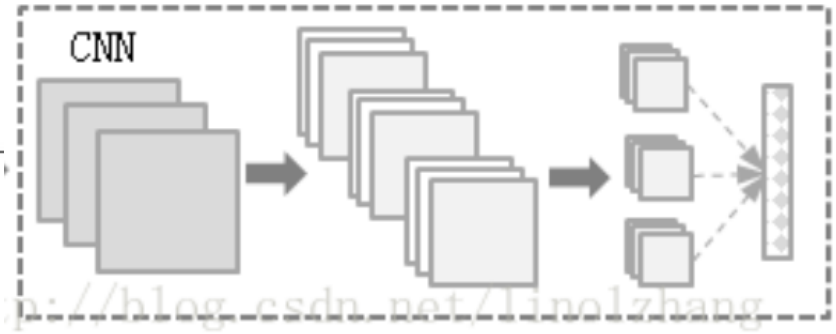
Region Proposal: Selective Search

http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

R-CNN

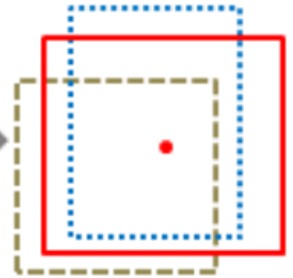
Crop & Warp

AlexNet

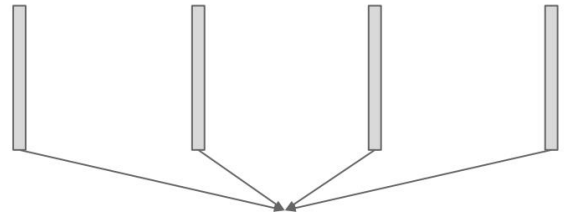


SVM

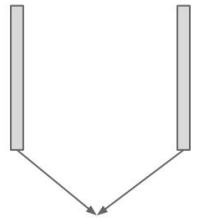
Bbox Regression



linear regression



Negative samples for Bicycle



Positive samples for Bicycle

http://slazebni.cs.illinois.edu/spring17/lec07_detection.pdf

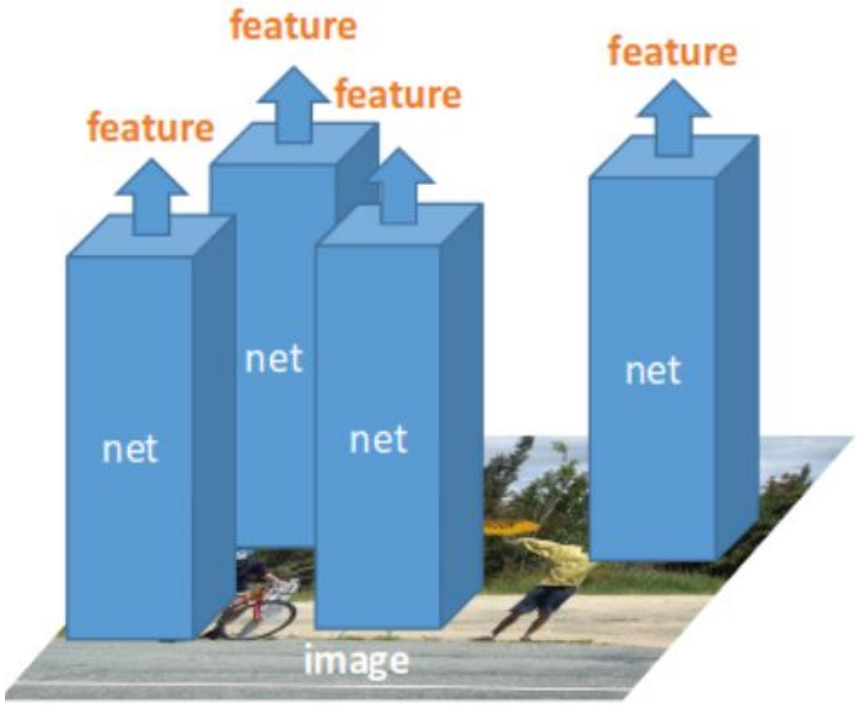
https://blog.csdn.net/sum_nap/article/details/80388110

Limitation of R-CNN

slow

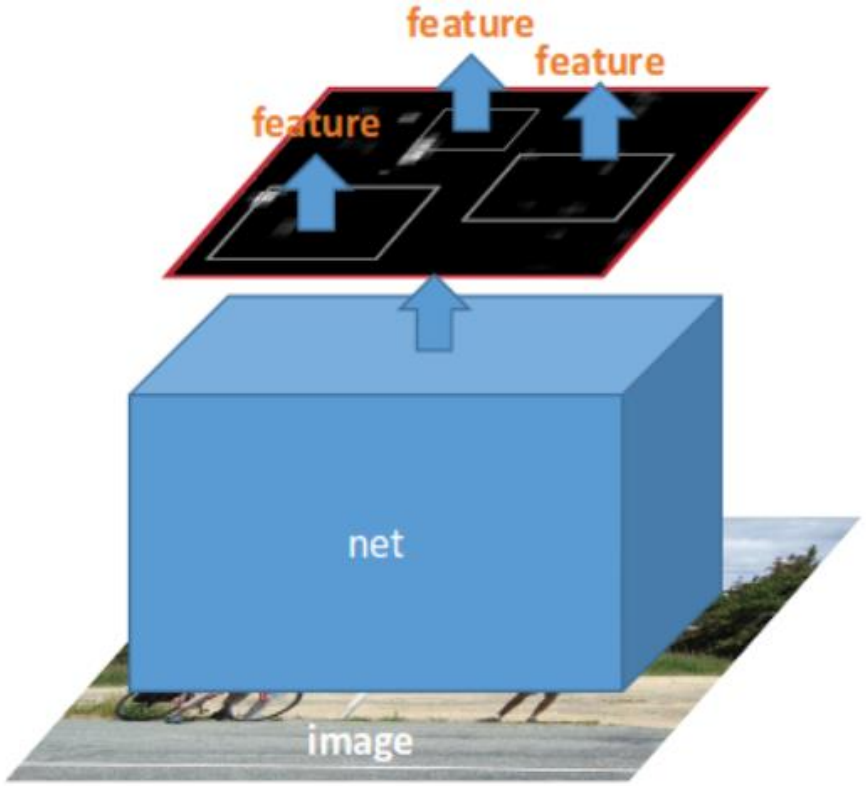
- **Each Region Goes Through Entire Network**
- **Three Steps (not end-to-end)**

SPP-net



R-CNN

2000 nets on image regions

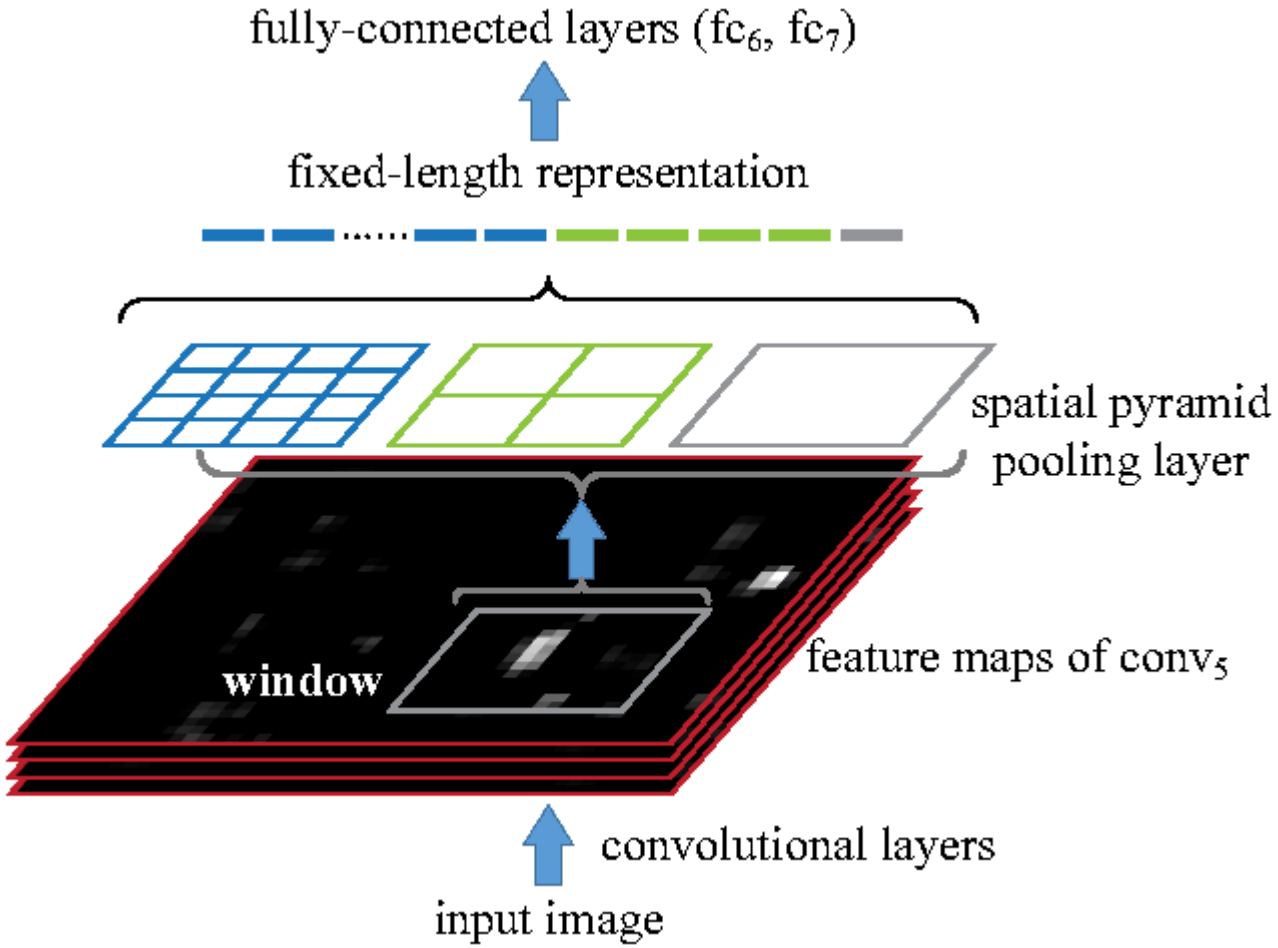


SPP-net

1 net on full image

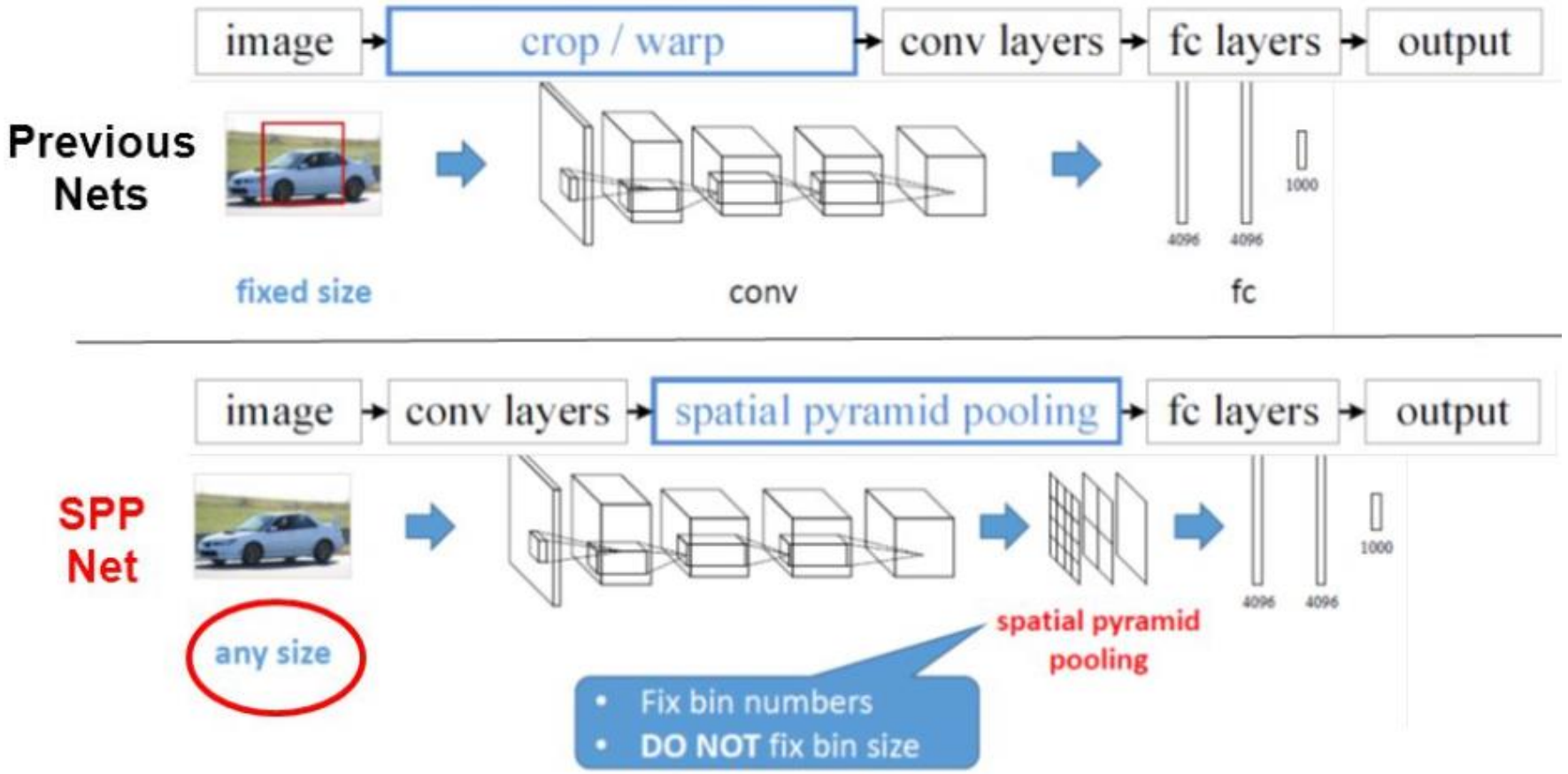
https://arxiv.org/abs/1406.4729

Spatial Pyramid Pooling



https://arxiv.org/abs/1406.4729

R-CNN vs. SPP



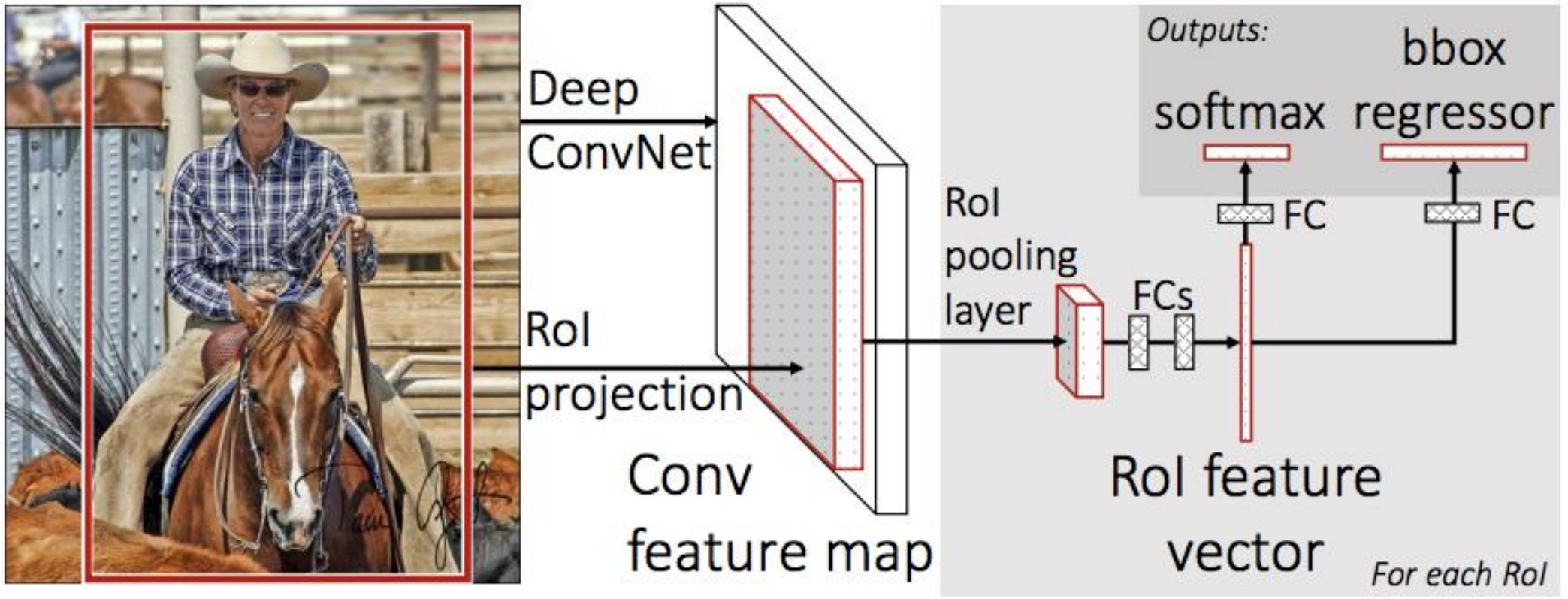
https://slideplayer.com/slide/5277459/

Limitation of SPP

still slow

- ~~Each Region Goes Through Entire Network~~
- Three Steps (not end-to-end)
- Cannot Do Backpropagation

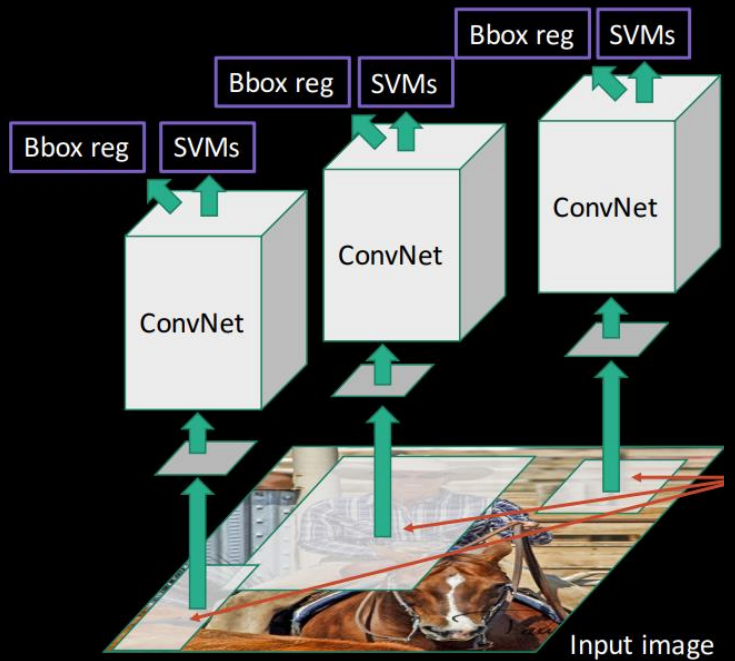
Fast R-CNN



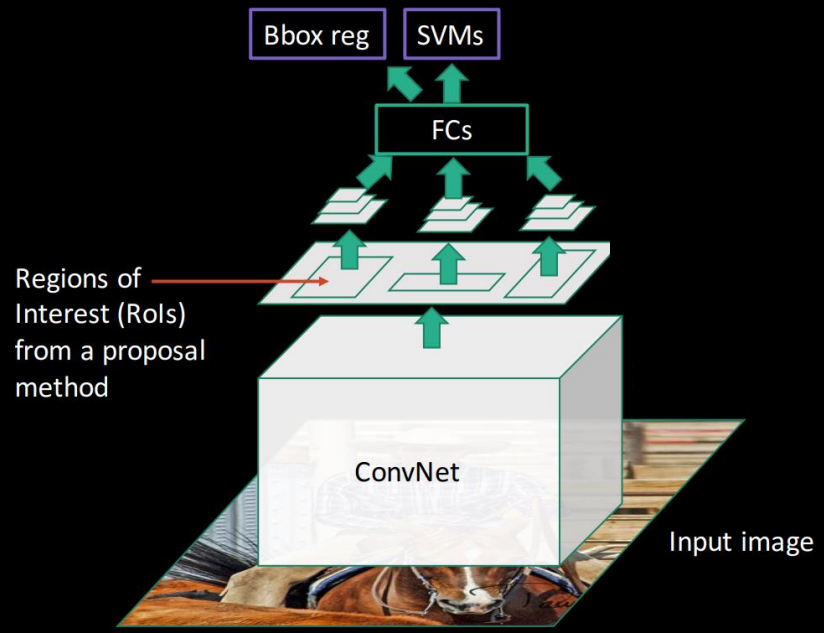
https://arxiv.org/abs/1504.08083

RCNN → Fast RCNN

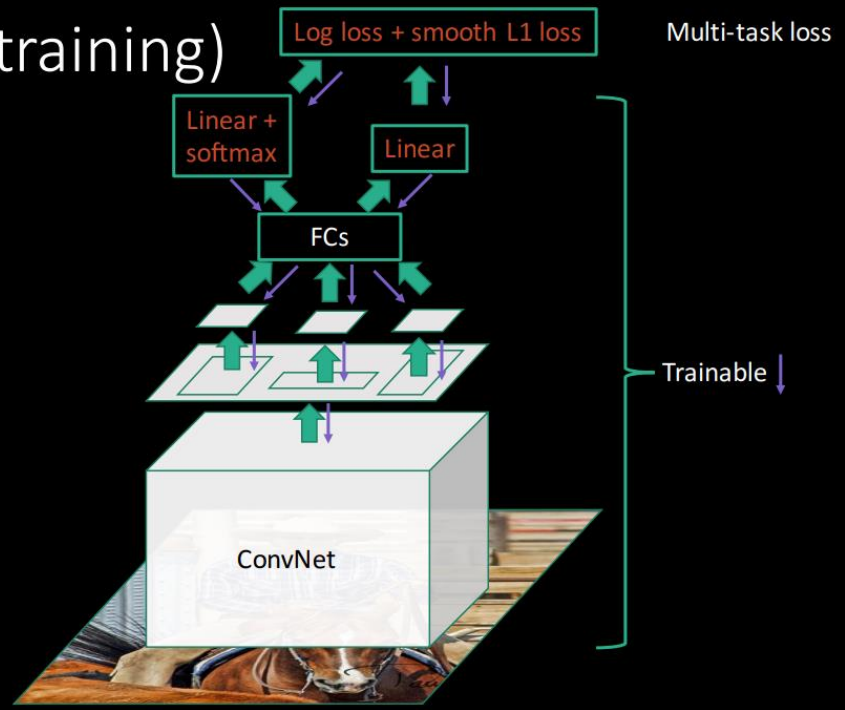
Slow R-CNN



SPP-net

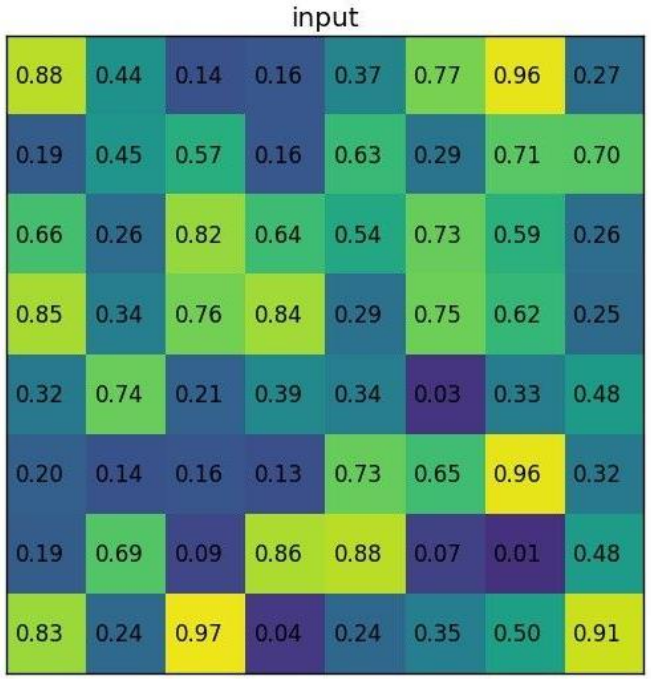


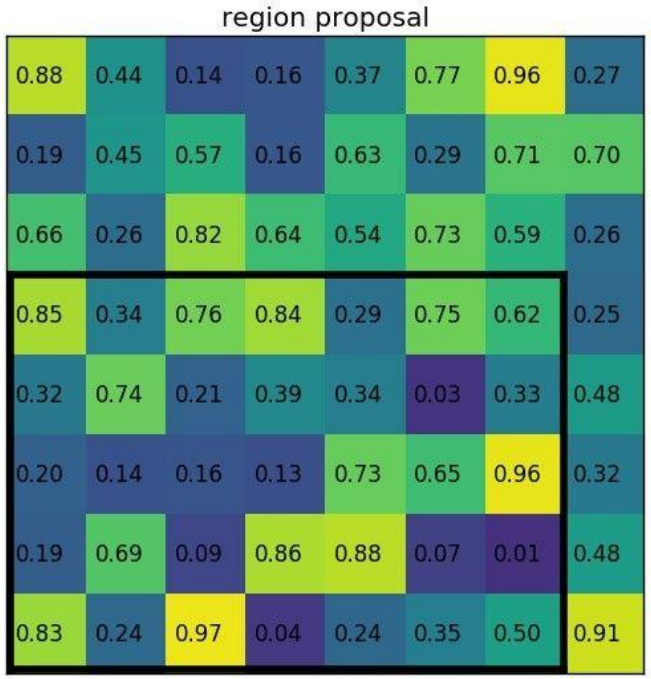
Fast R-CNN (training)

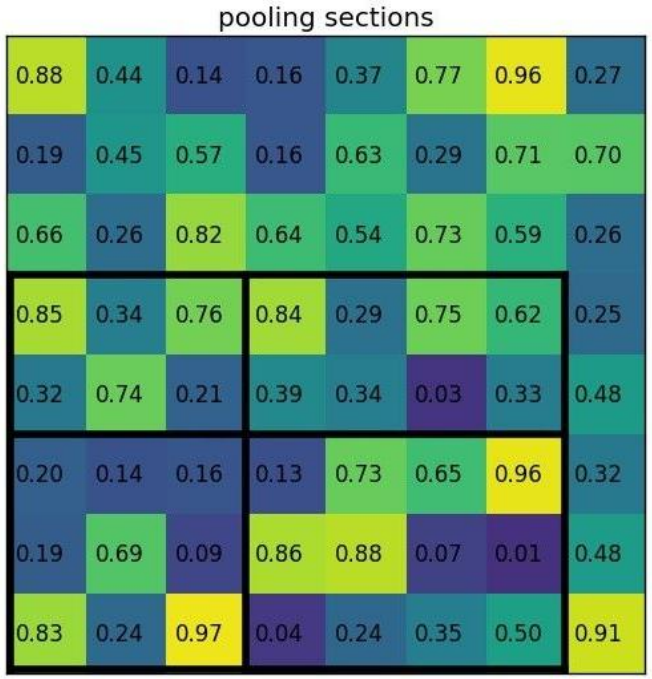


http://www.robots.ox.ac.uk/~tvg/publications/talks/fast-rcnn-slides.pdf

ROI Pooling





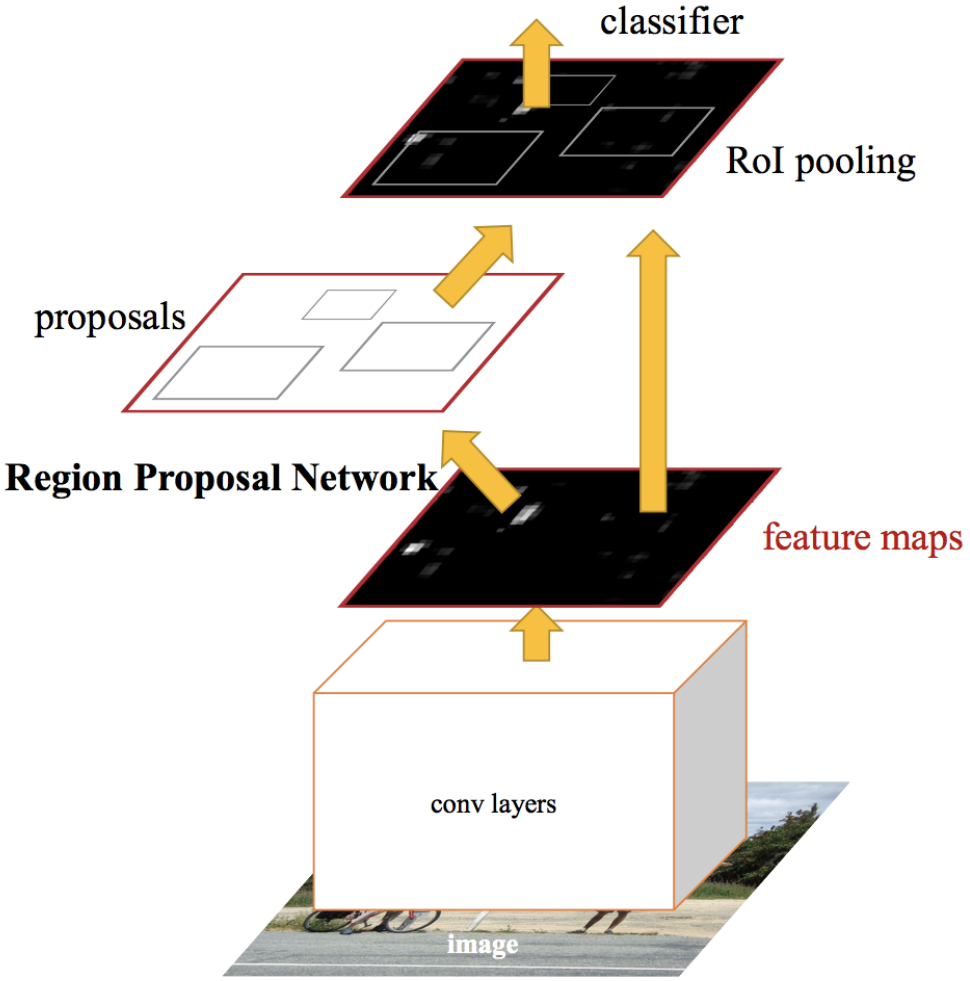


https://deepsense.ai/region-of-interest-pooling-explained/

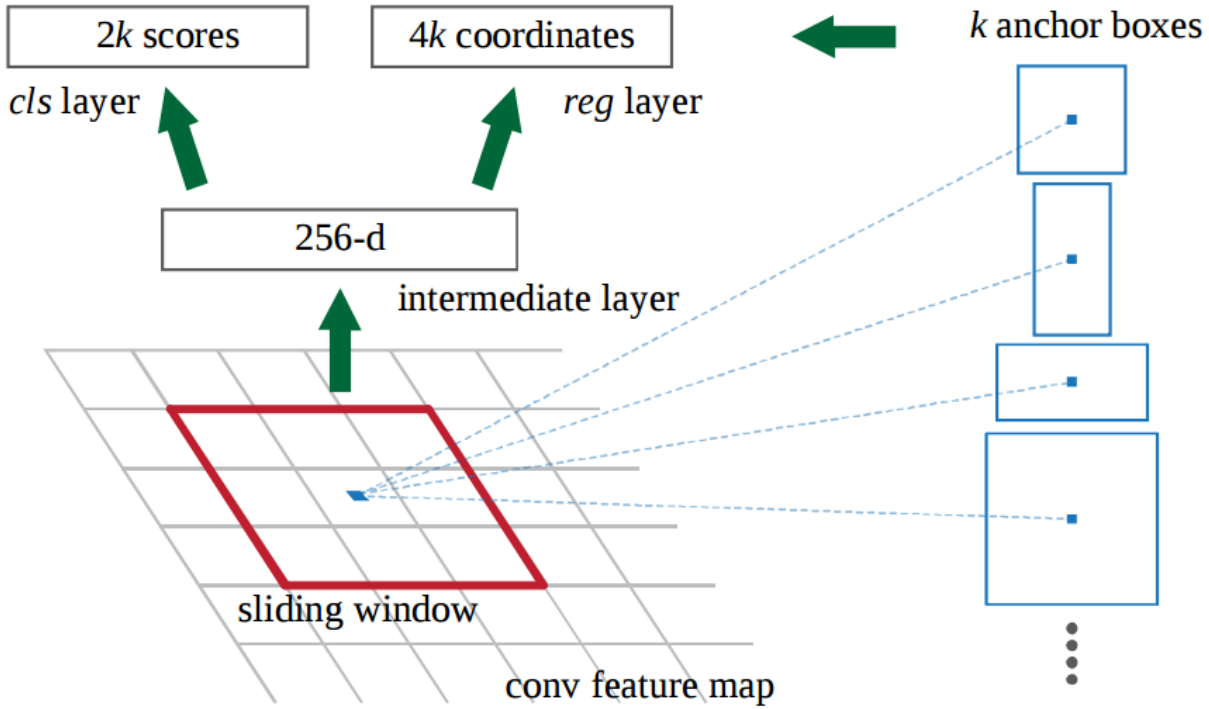
Limitation of Fast-RCNN

- ~~Each Region Goes Through Entire Network~~
- ~~Three Steps (not end-to-end)~~
- ~~Cannot Do Backpropagation~~
- Region Proposal is not End-to-End

Faster R-CNN

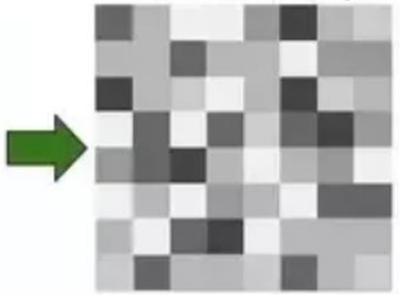


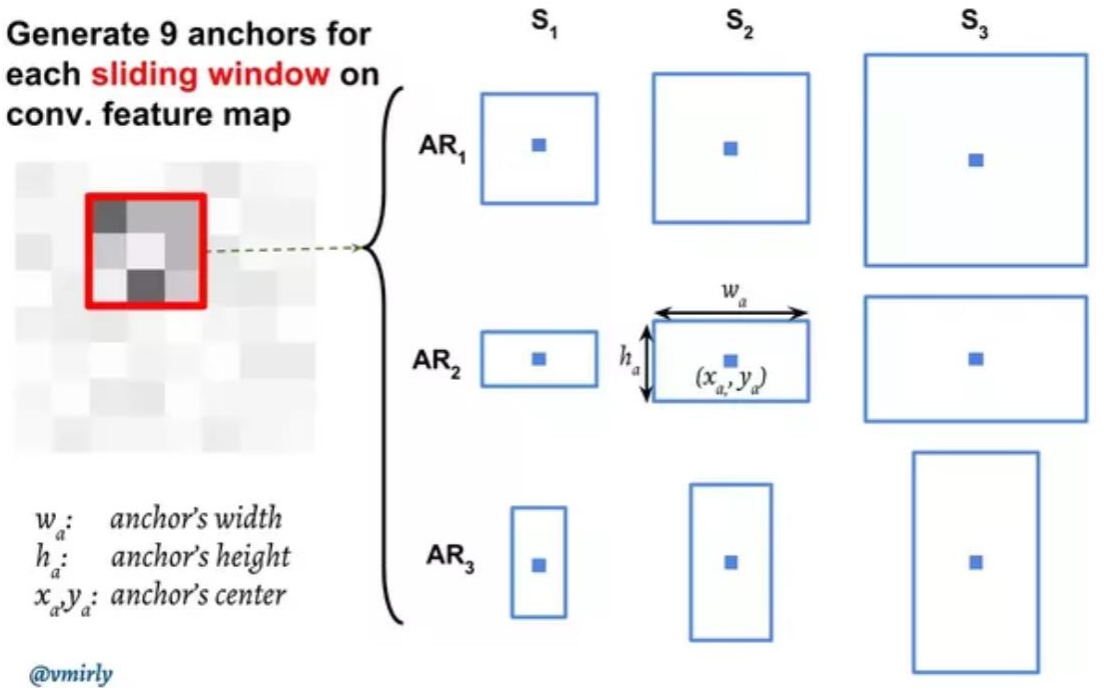
region proposal network (RPN)



https://arxiv.org/abs/1506.01497

Region Proposal Network



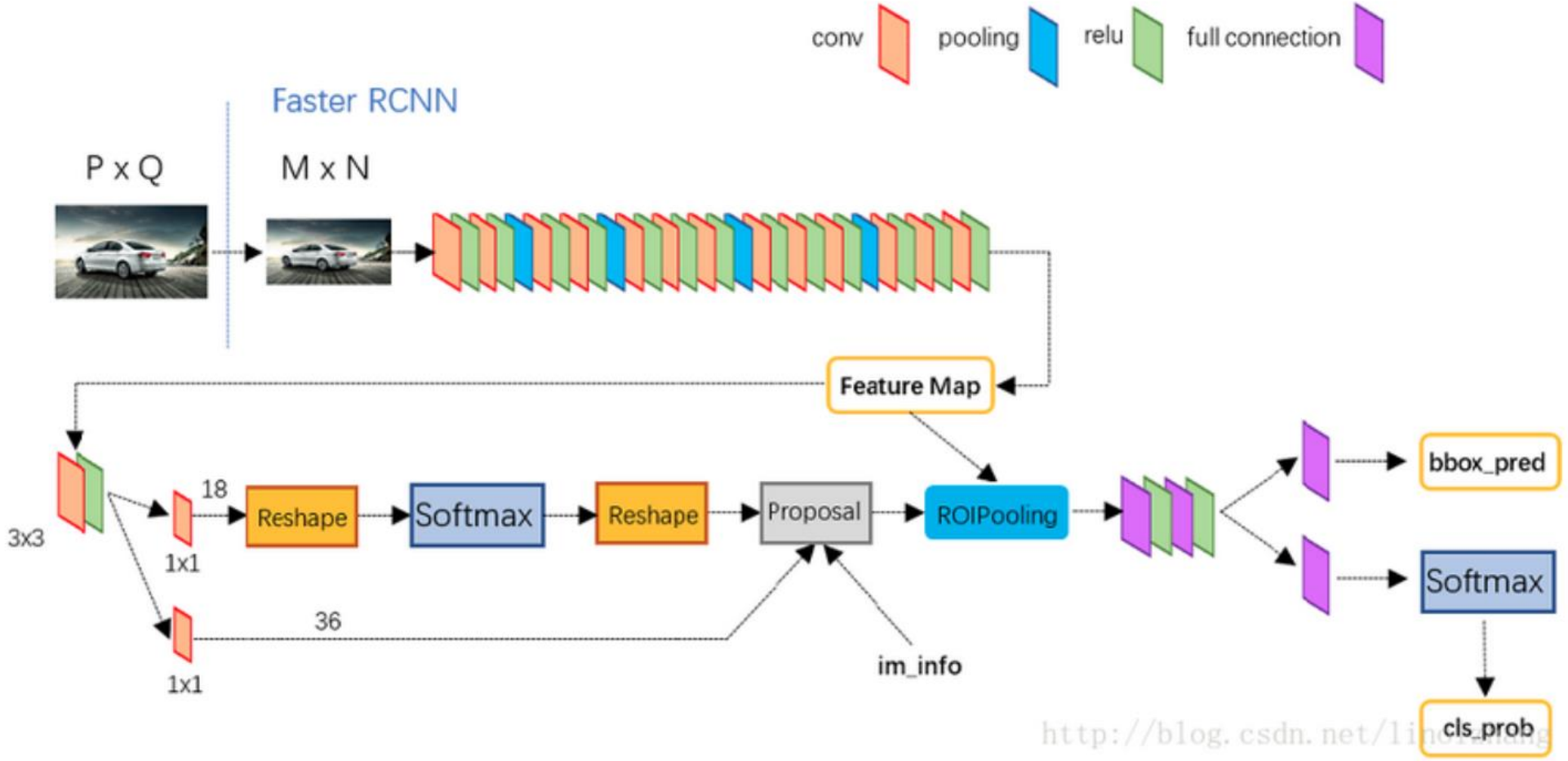


$$p^* = \begin{cases} 1 & if \quad IoU > 0.7 \\ -1 & if \quad IoU < 0.3 \\ 0 & otherwise \end{cases}$$

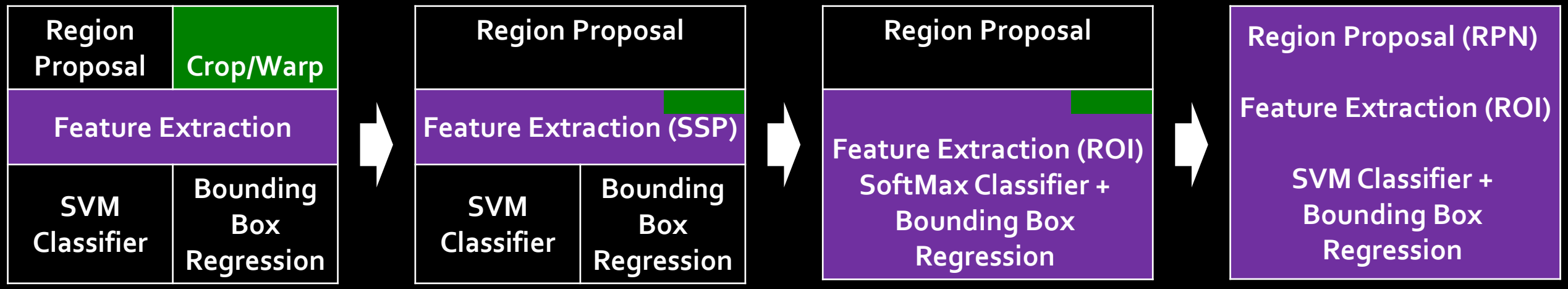
$$IoU = \frac{Anchor \cap GTBox}{Anchor \cup GTBox}$$

https://www.quora.com/How-does-the-region-proposal-network-RPN-in-Faster-R-CNN-work

End-to-End

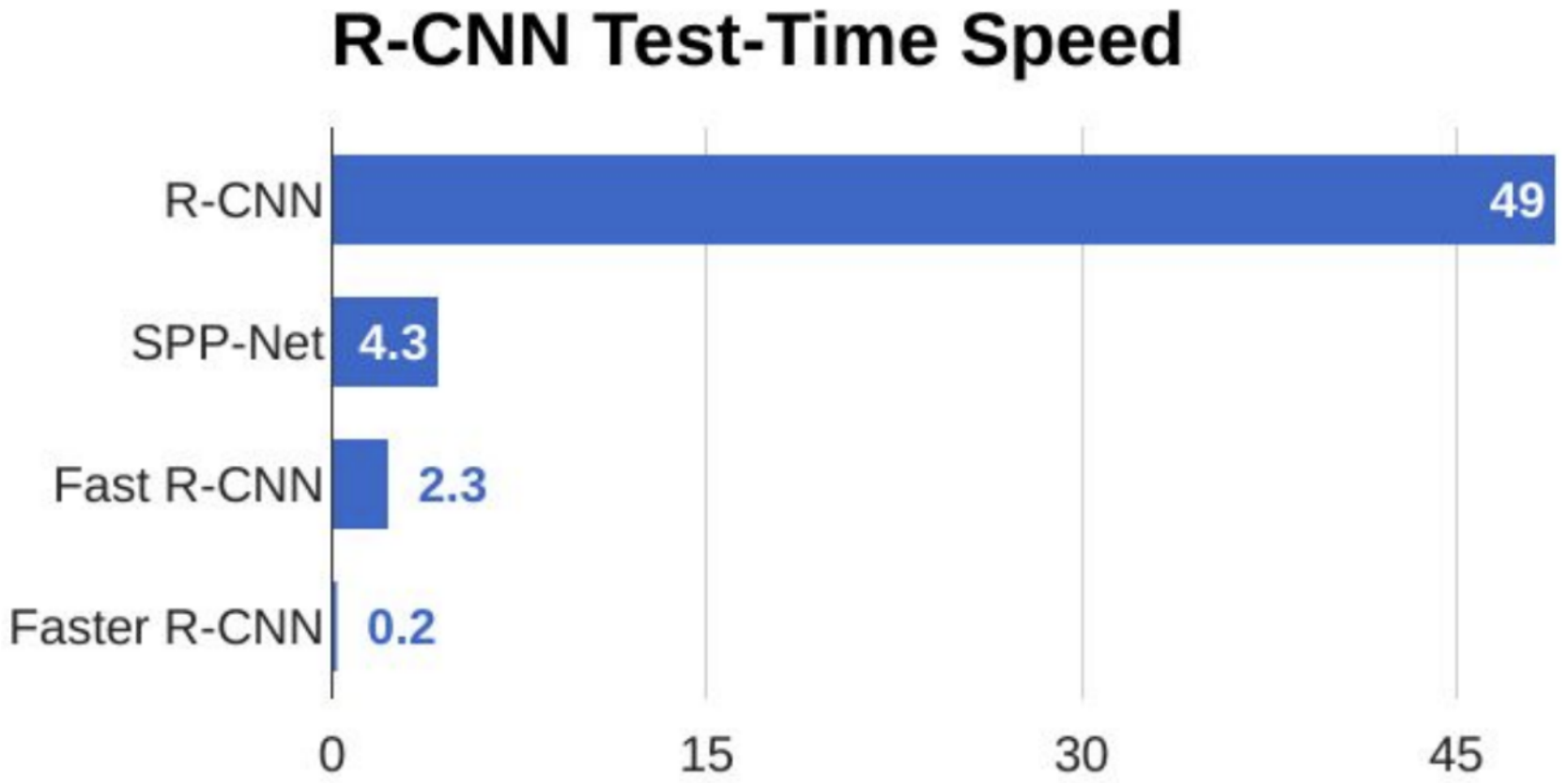

conv
pooling
relu
full connection
Faster RCNN
P x Q
M x N
Feature Map
3x3
18
1x1
Reshape
Softmax
Reshape
Proposal
ROIPooling
bbox_pred
Softmax
cls_prob
36
1x1
im_info
http://blog.csdn.net/lin

Summerize



R-CNN SPP-Net Fast R-CNN Faster R-CNN

Comparison


R-CNN Test-Time Speed
R-CNN
SPP-Net
Fast R-CNN
Faster R-CNN
49
4.3
2.3
0.2
0
15
30
45

Yolo: You Only Look Once

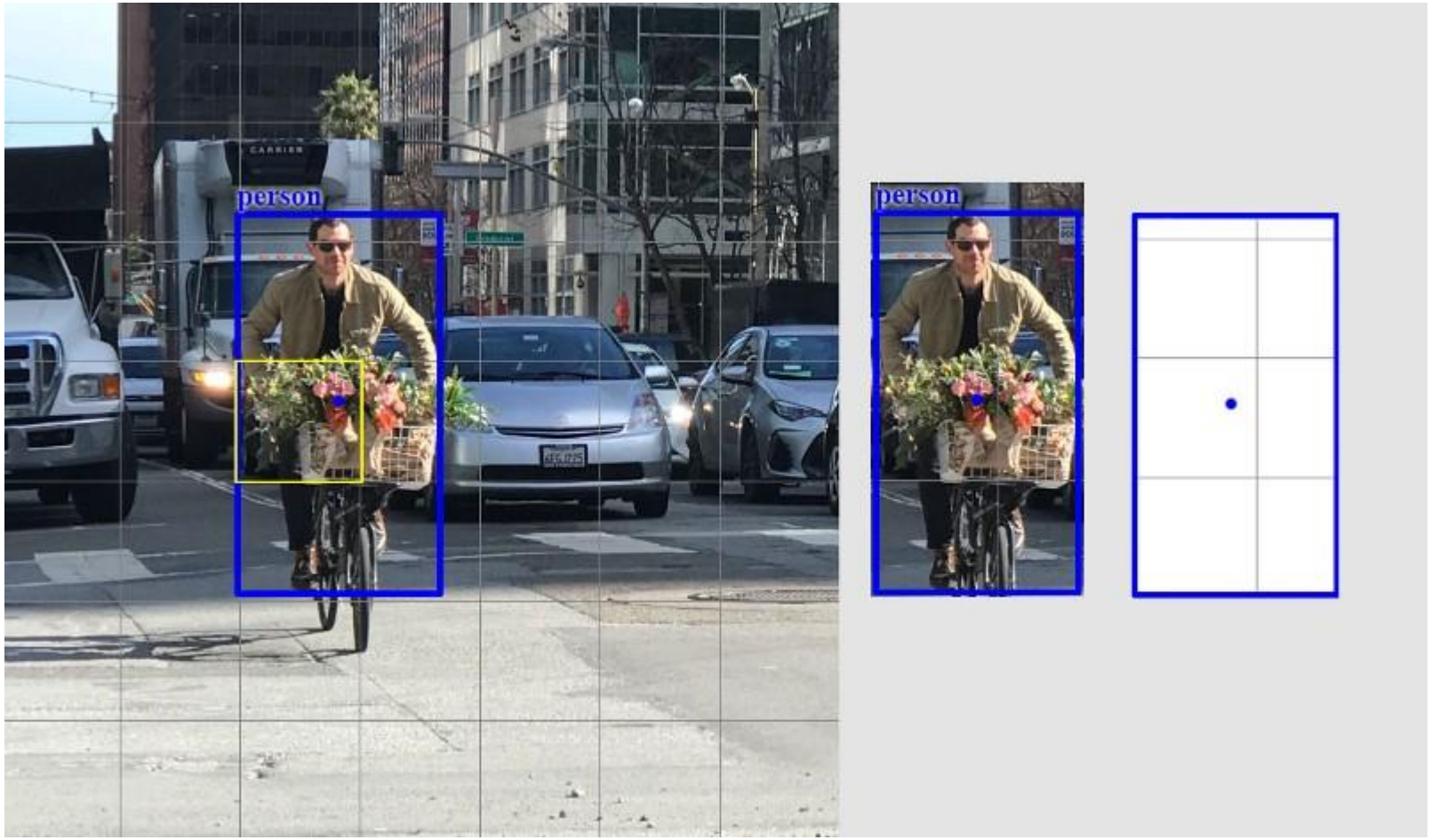
https://imgur.com/gallery/pvCnUGc

- Consider detection a regression problem
- Use a single ConvNet
- Runs once on entire image. Very Fast!

YOLO

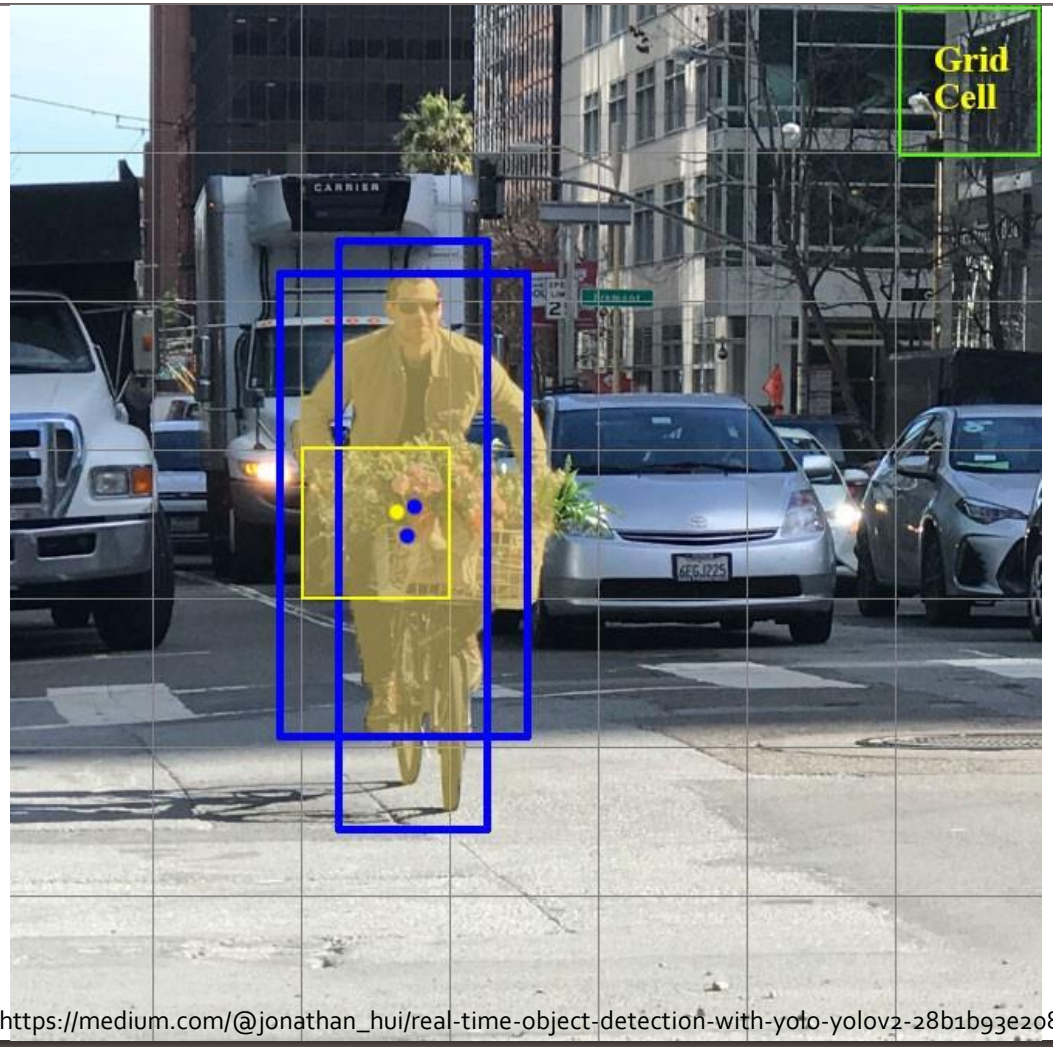
https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLO



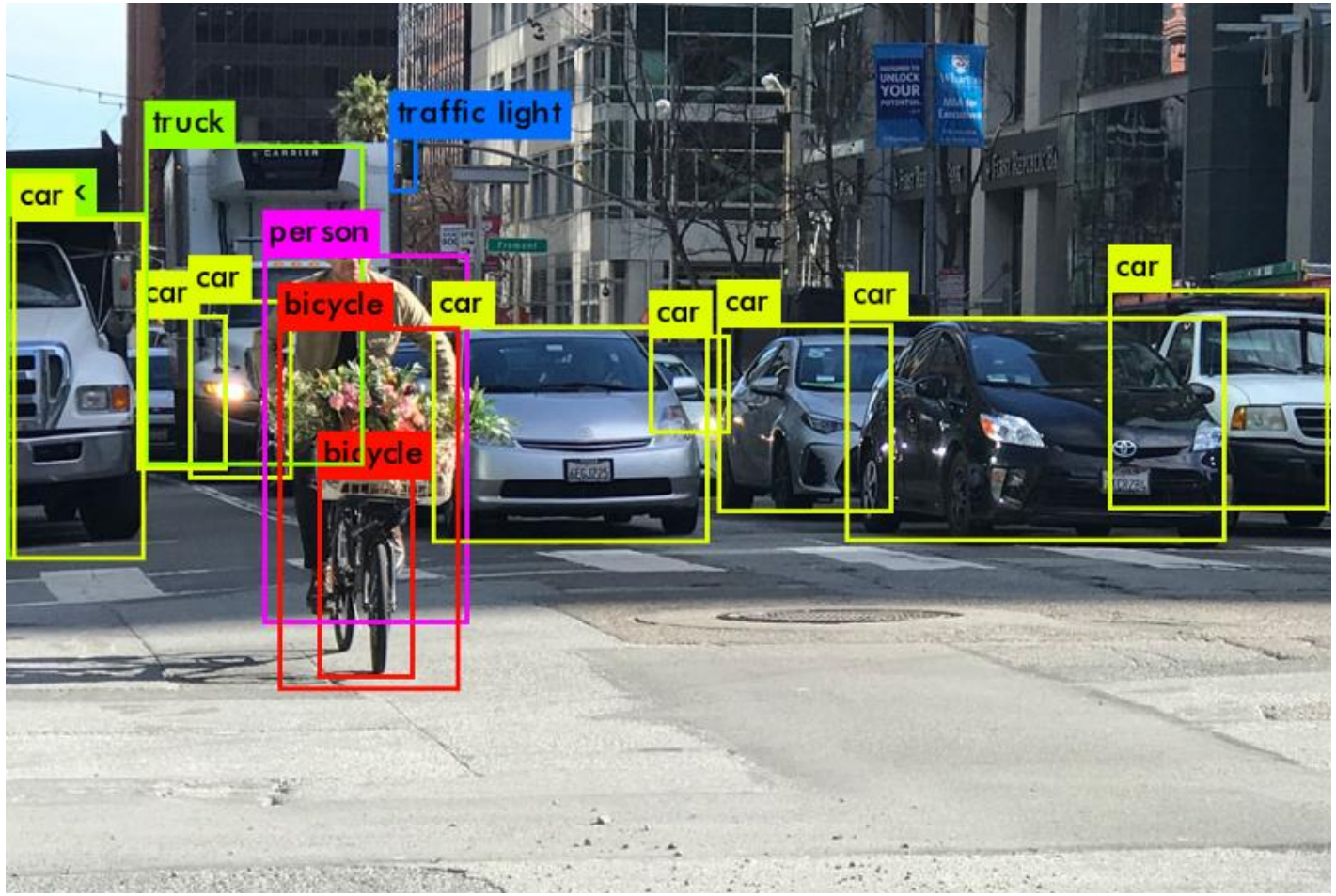
https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLO



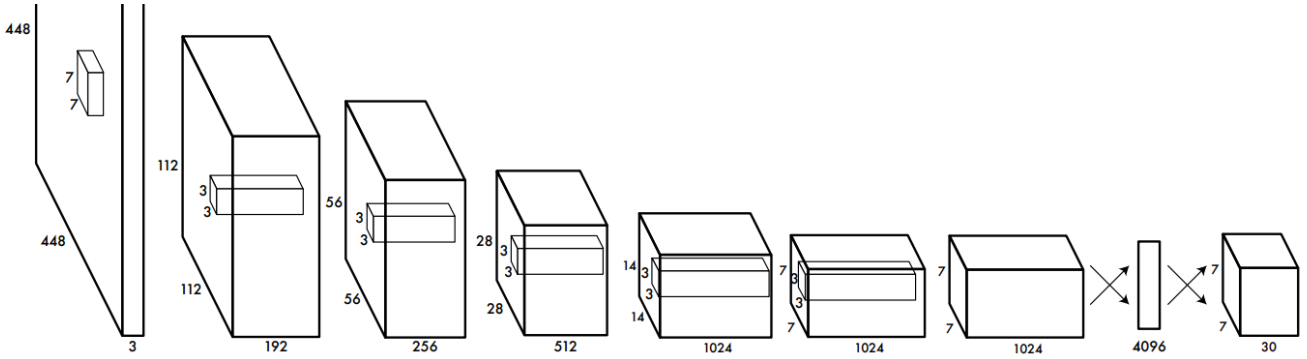
https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

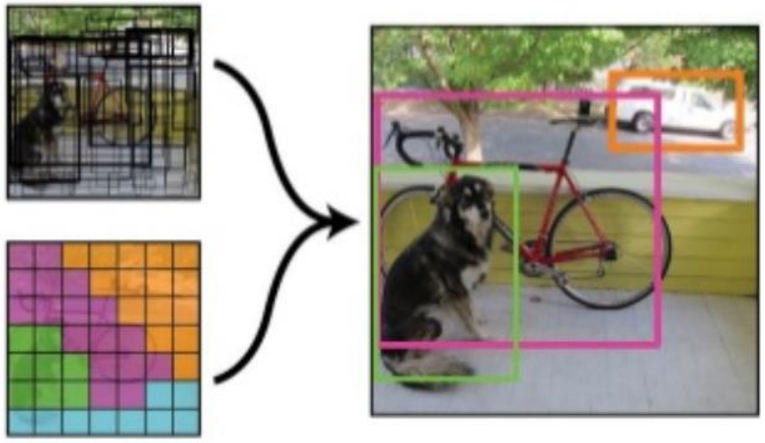
YOLO



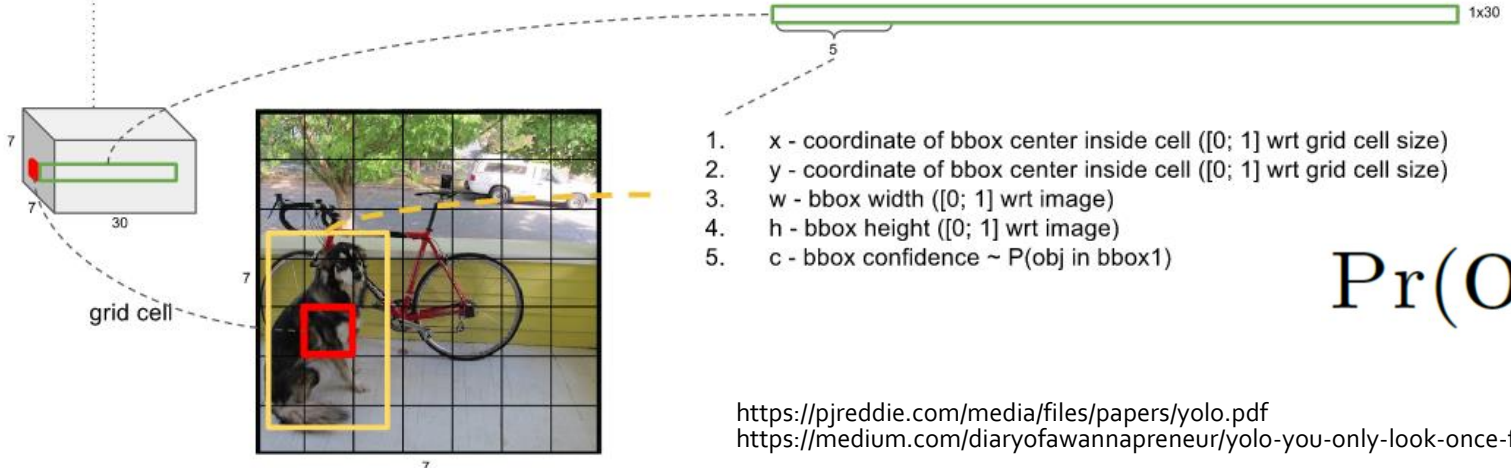
https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLO





Tensor values interpretation



1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

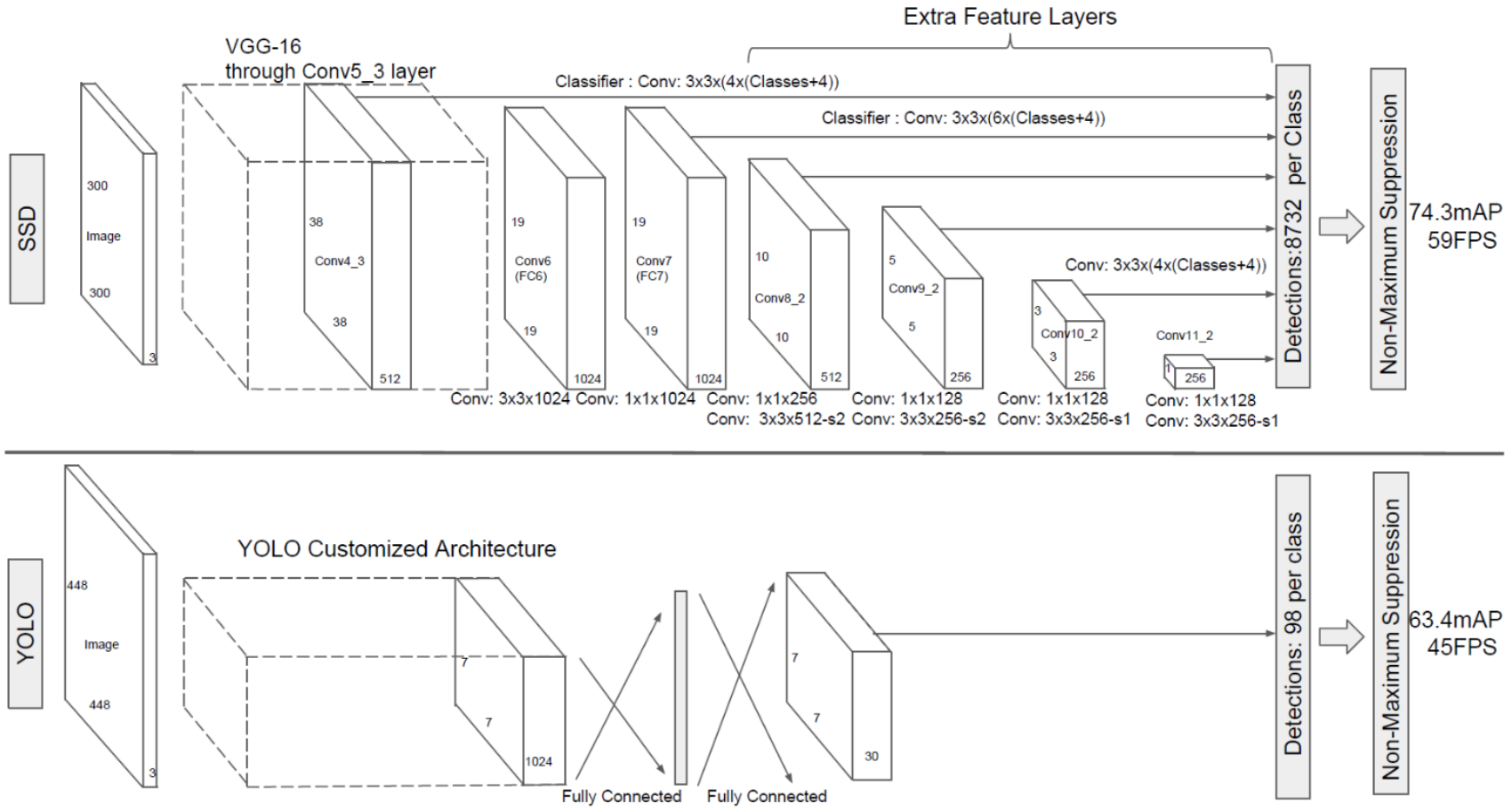
$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

https://pjreddie.com/media/files/papers/yolo.pdf
https://medium.com/diaryofawannapreneur/yolo-you-only-look-once-for-object-detection-explained-6f80ea7aaa1e

YOLO Limiation

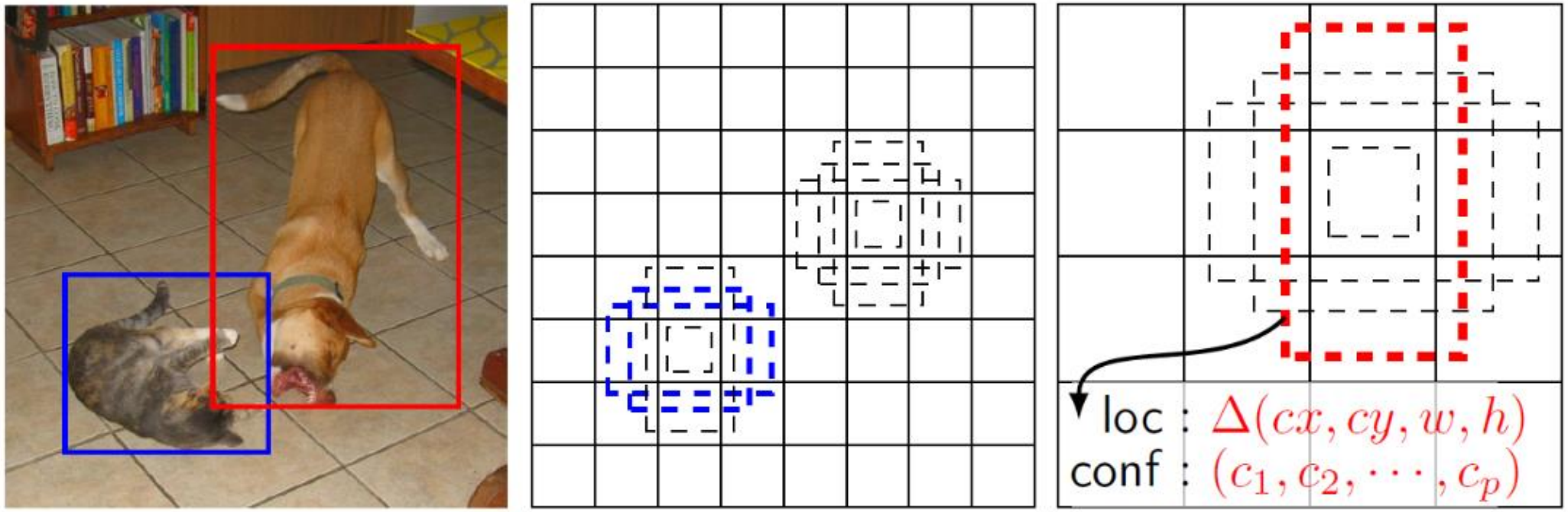
https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

SSD: Single Shot Detector



https://arxiv.org/abs/1512.02325

Feature Maps



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

(C+4)*k*m*n

https://arxiv.org/abs/1512.02325

Comparison

Single **S**hot **D**etector (SSD) achieves a good balance between speed and accuracy.

https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

Anchor Free

CornerNet
CenterNet
ExtremNet
…