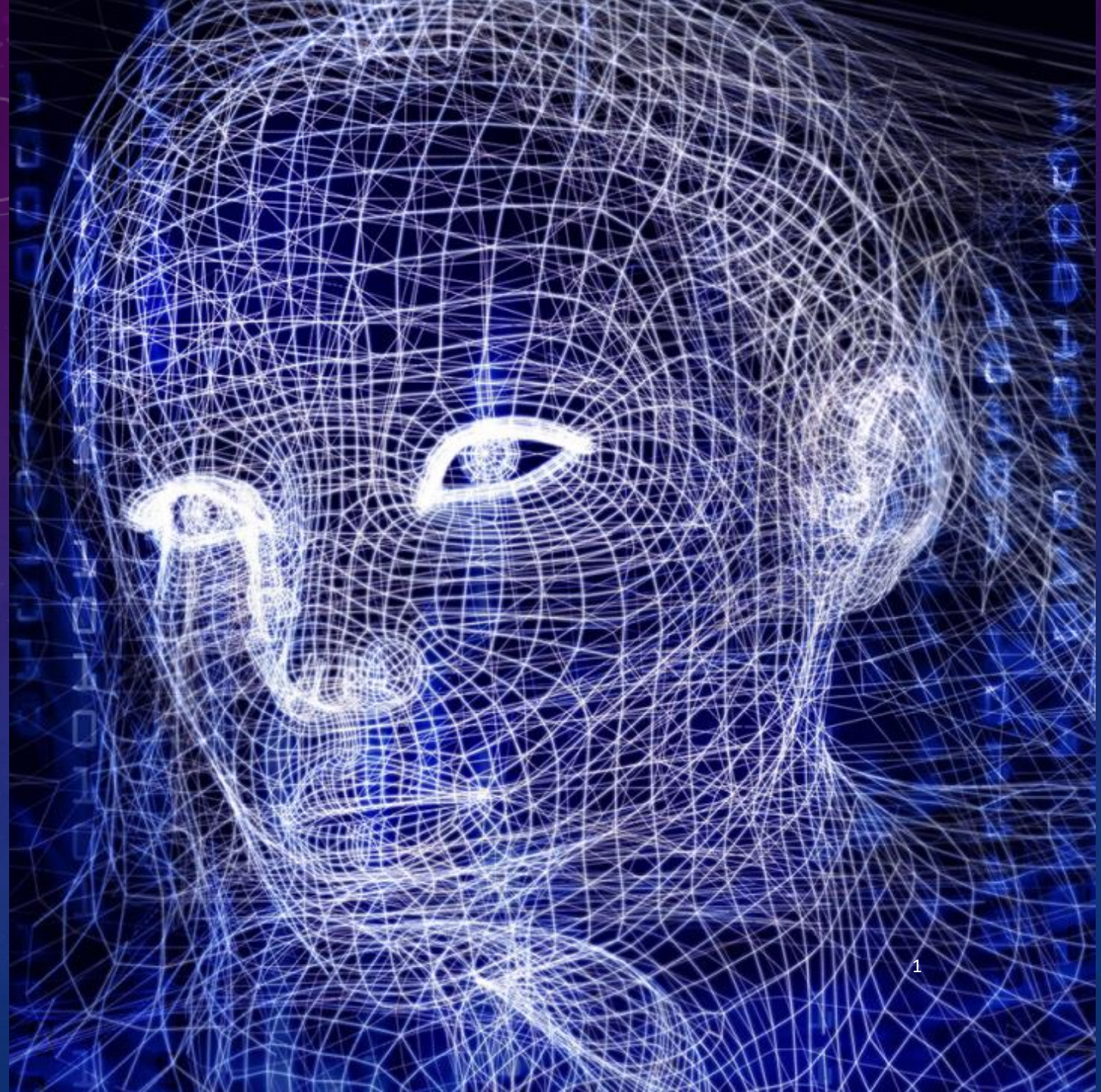
LECTURE SERIES FOR DIGITAL SURVEILLANCE SYSTEMS AND APPLICATION

INTRODUCTION TO DEEP LEARNING

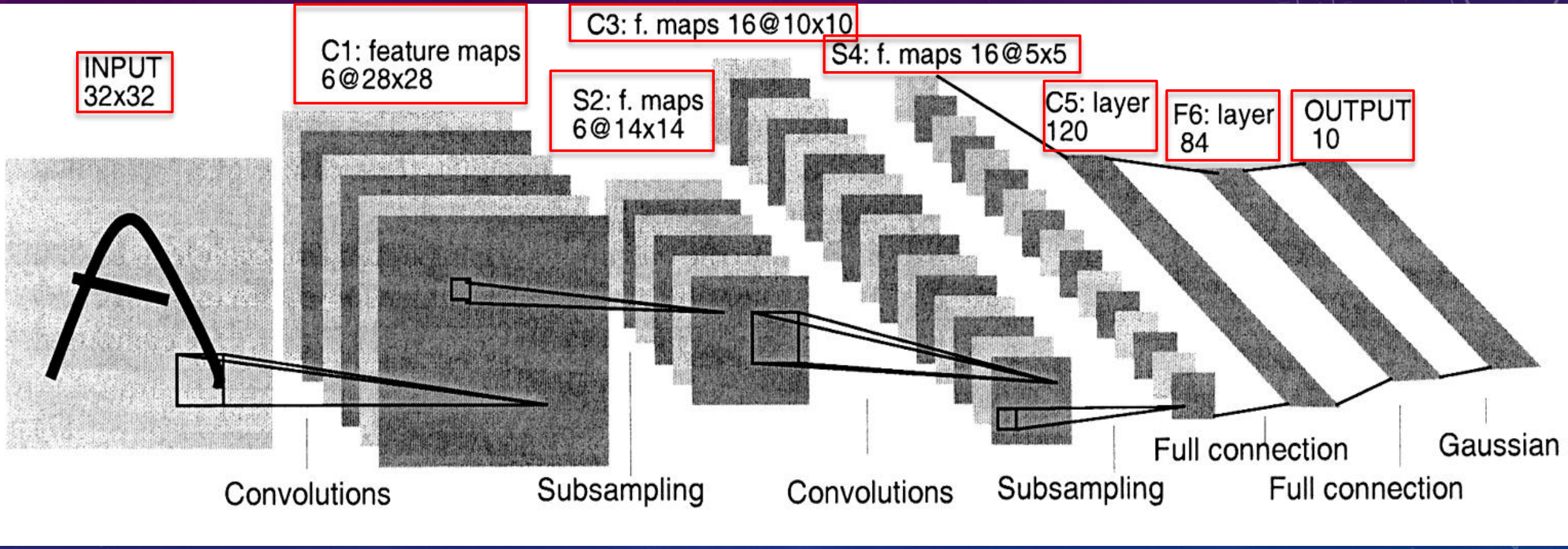


徐繼聖

Gee-Sern Jison Hsu

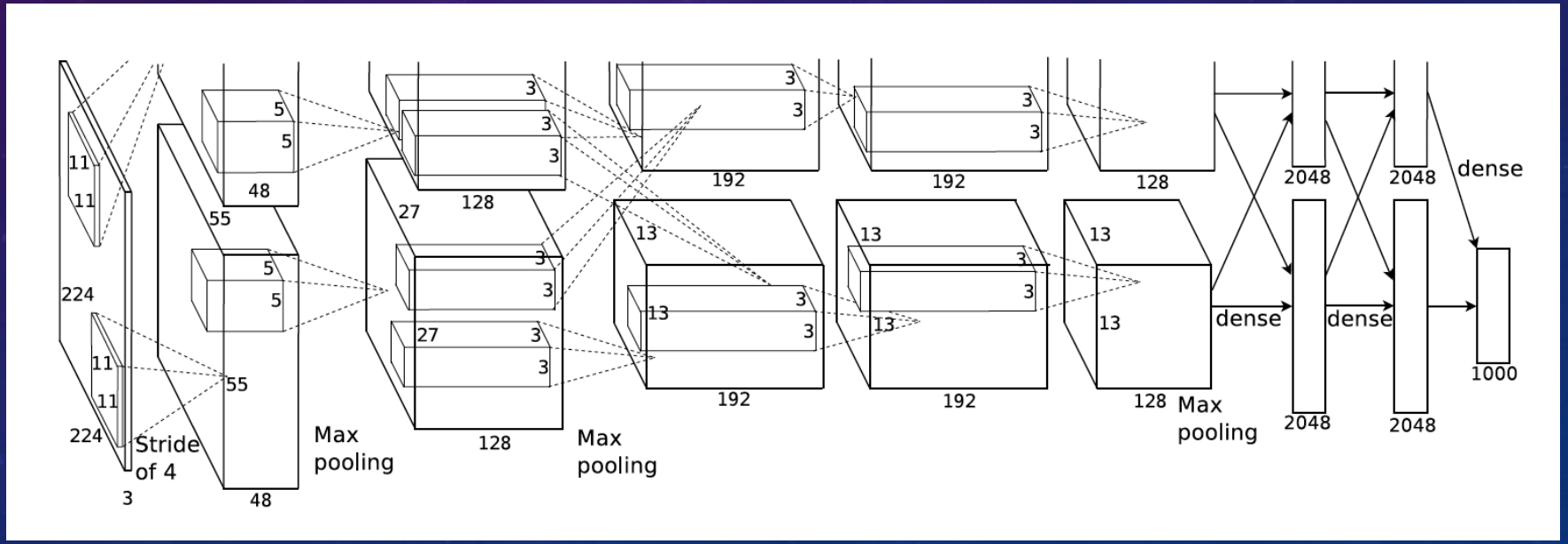
National Taiwan University of Science and Technology

The architecture of LeNet5

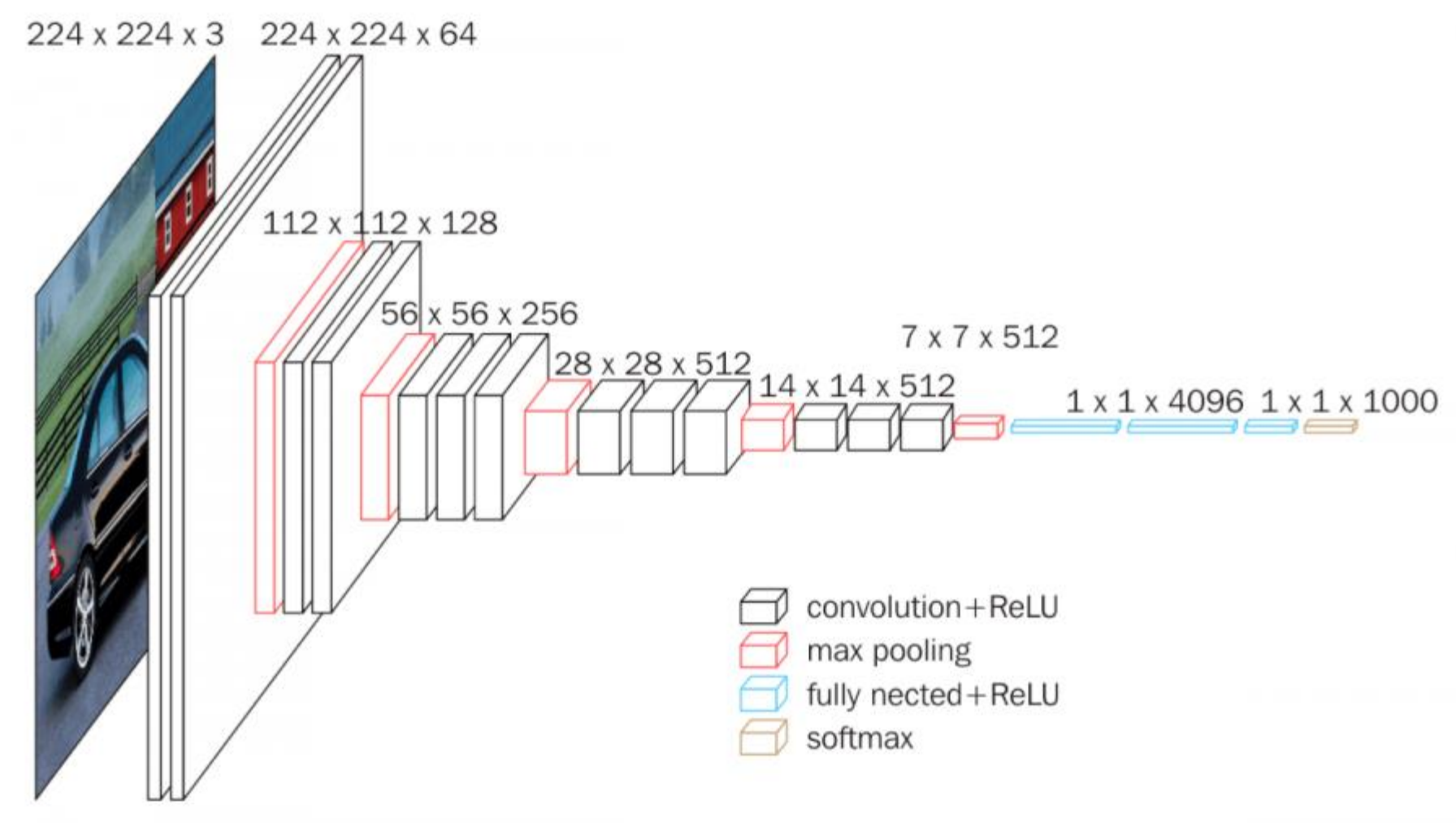


AlexNet

- Use Relu instead of Tanh to add non-linearity. It accelerates the speed by 6 times at the same accuracy.
- Use dropout instead of regularisation to deal with overfitting. However the training time is doubled with the dropout rate of 0.5.
- Overlap pooling to reduce the size of network. It reduces the top-1 and top-5 error rates by 0.4% and 0.3%, repectively.

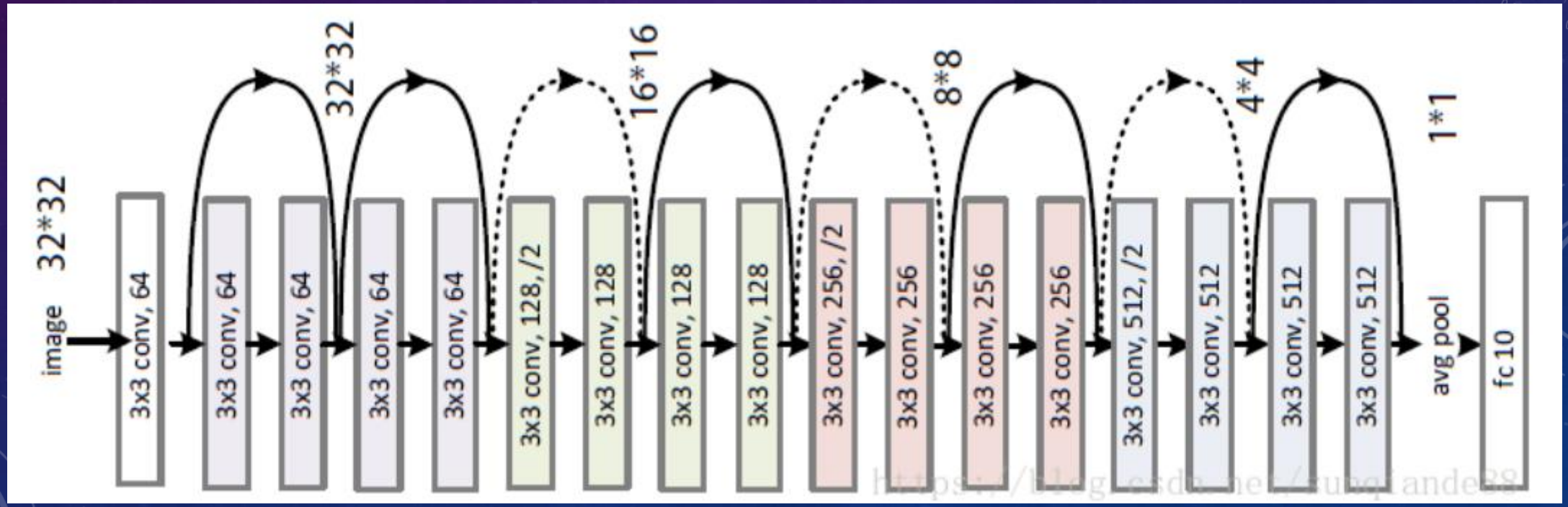


ILSVRC-2014 VGG Model



ResNet18

- Since AlexNet, the state-of-the-art CNN architecture is going deeper and deeper.
- However, increasing network depth does not work by simply stacking layers together. Deep networks are hard to train because of the notorious vanishing gradient problem
- The core idea of ResNet is introducing a so-called “identity shortcut connection” that skips one or more layers, as shown in the following figure:



CNN Architectures – Stanford University School of Engineering

LeNet (02:47 - 03:15)

AlexNet (03:16 – 05:26)

(05:48 – 06:39)

(06:56 – 07:33)

(08:14 – 11:54)

(12:08 – 12:53)

(12:56 – 13:48)

(14:02 – 15:29)

VGGNet (15:30 – 17:10)

(18:15 – 20:58)

(25:05 – 27:36)

(28:28 – 28:37)

GoogLeNet (28:37 – 33:17)

(36:50 – 40:11)

(41:20 – 43:55)

(47:07 – 47:23)

ResNet (47:24 – 53:08)

(53:56 – 55:01)

(58:16 – 1:02:33)

CNN Architectures – Stanford University School of Engineering

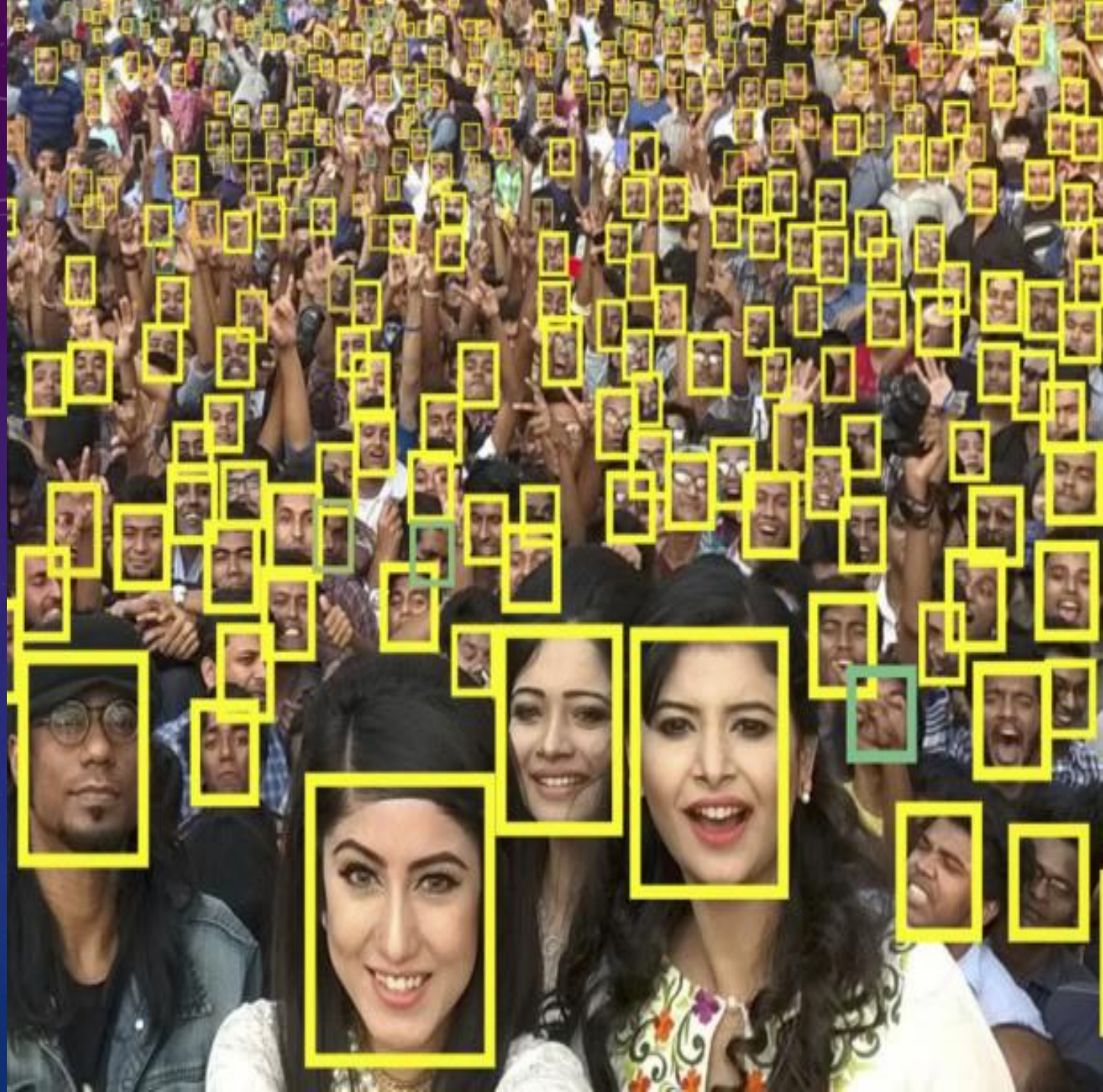
https://www.youtube.com/watch?v=DAOcjicFr1Y&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=9 [1:17:39]

See Coding Manual, Chapter 3, Page 74.

DSS

CH 4

OBJECT DETECTION

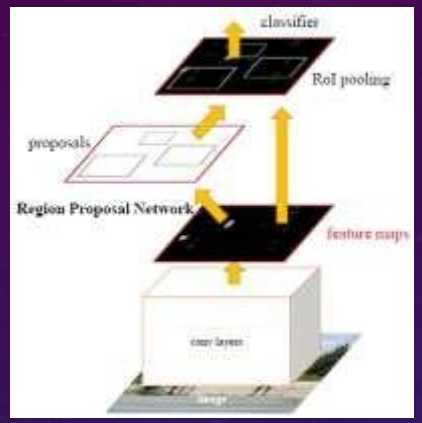


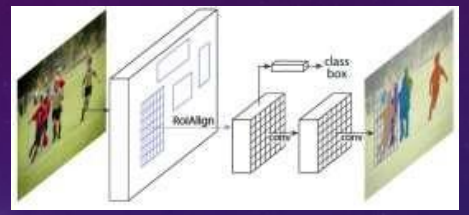
徐繼聖

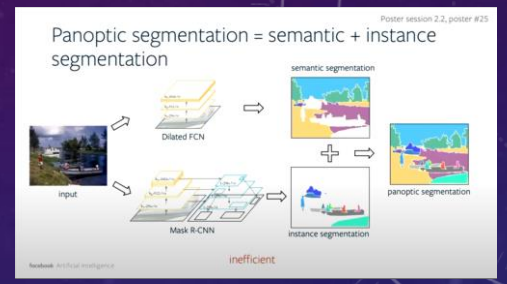
Gee-Sern Jison Hsu

National Taiwan University of Science and Technology

The History of Object Detection in Deep Learning







RCNN

Faster RCNN

Mask RCNN

Panoptic Feature Pyramid Network

2015.4.30

2015.6.8

2015.12.08

2015.12.25

2017.1.23

2018.4.8

2018.4.8

2013.11.11

2015.5.14

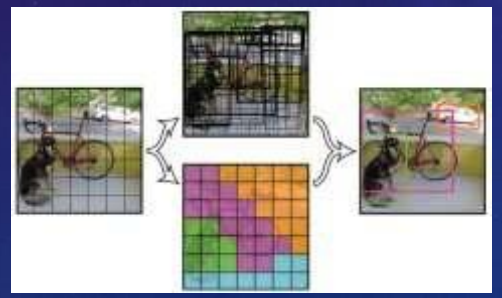
2017.3.20

2019.1.8

Fast RCNN

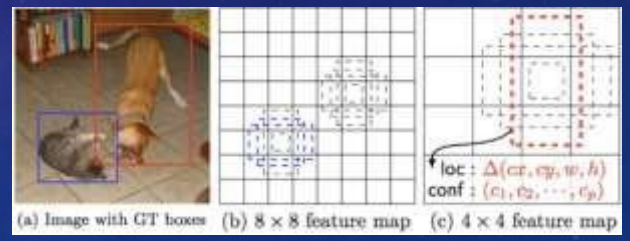
Yolo

Yolo v2



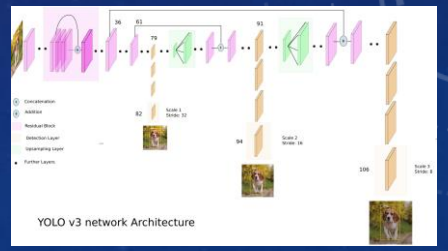
SSD

DSSD



Yolo V3

Yolo V4



2012.12
AlexNet

2014.9
VggNet &
InceptionNet

2015.12.10
ResNet

Contents

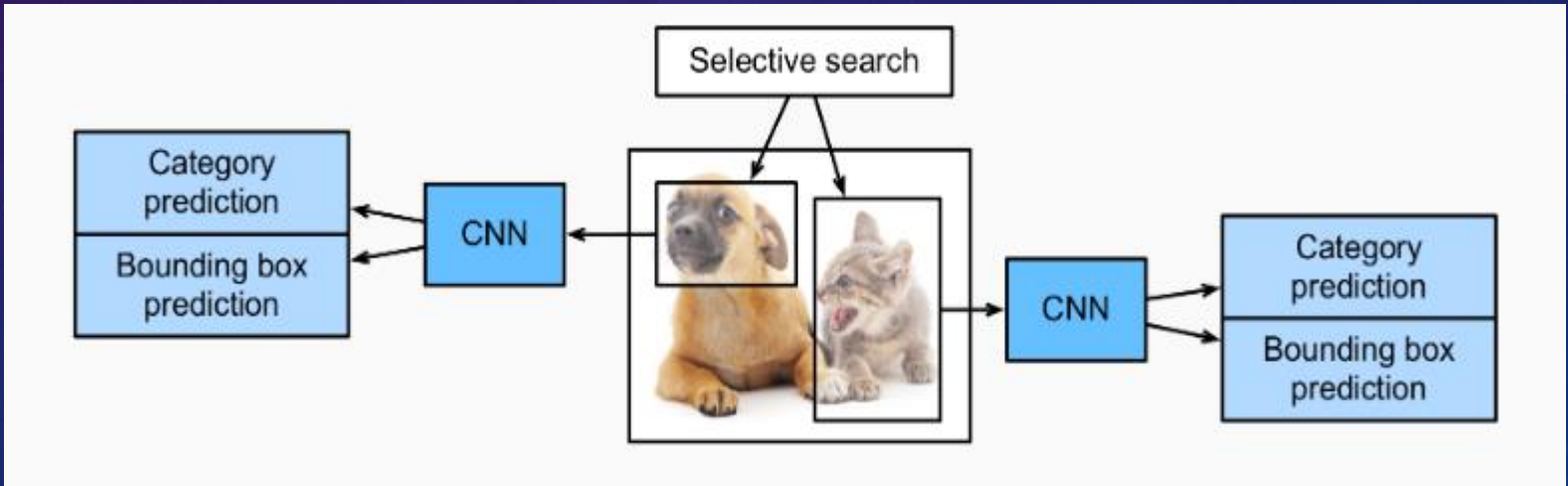
- Region-based CNN
- Fast R-CNN
- Faster R-CNN
- Mask R-CNN

Region-based CNN

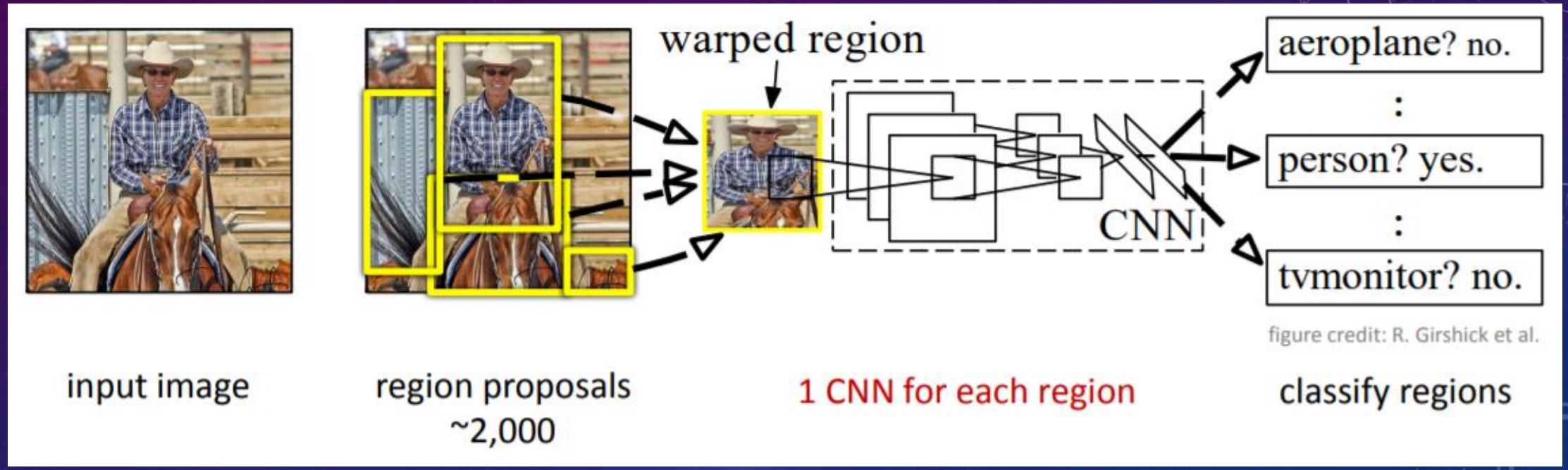
- Introduction
- Region-Based CNN
- R-CNN Algorithm
- Region Proposals – Selective Search

Region-based CNN

- R-CNN models first select several proposed regions from an image (for example, anchor boxes are one type of selection method) and then label their categories and bounding boxes (e.g., offsets).
- They use a CNN to perform forward computation to extract features from each proposed area. Afterwards, we use the features of each proposed region to predict their categories and bounding boxes.



Region-Based CNN Architecture



R-CNN pipeline

R-CNN Algorithm

Per-Image computation

Extract features by f_I

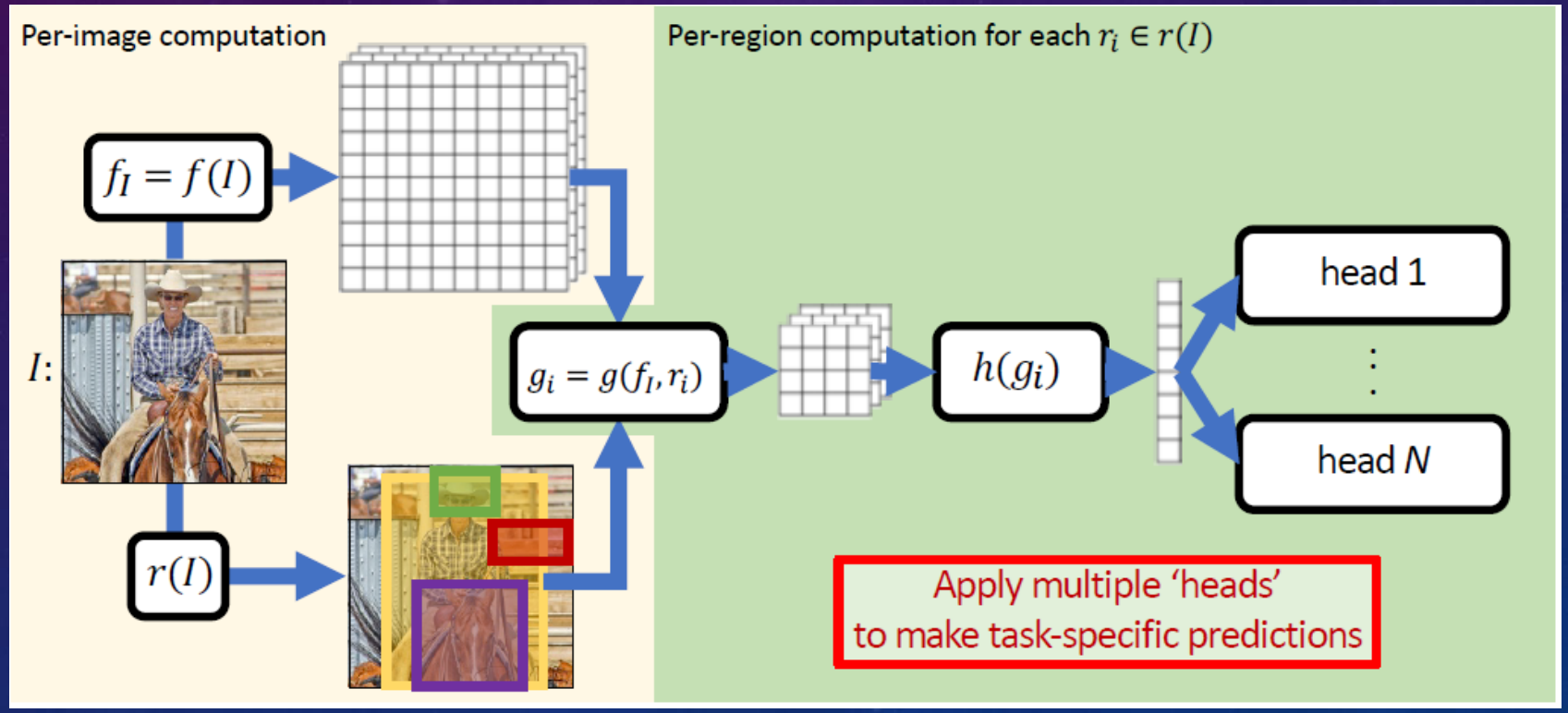
Region detection proposal $r(I)$

Per-Region computation

Extract region of Interest g_i

Pass each region to CovNet $h(g_i)$

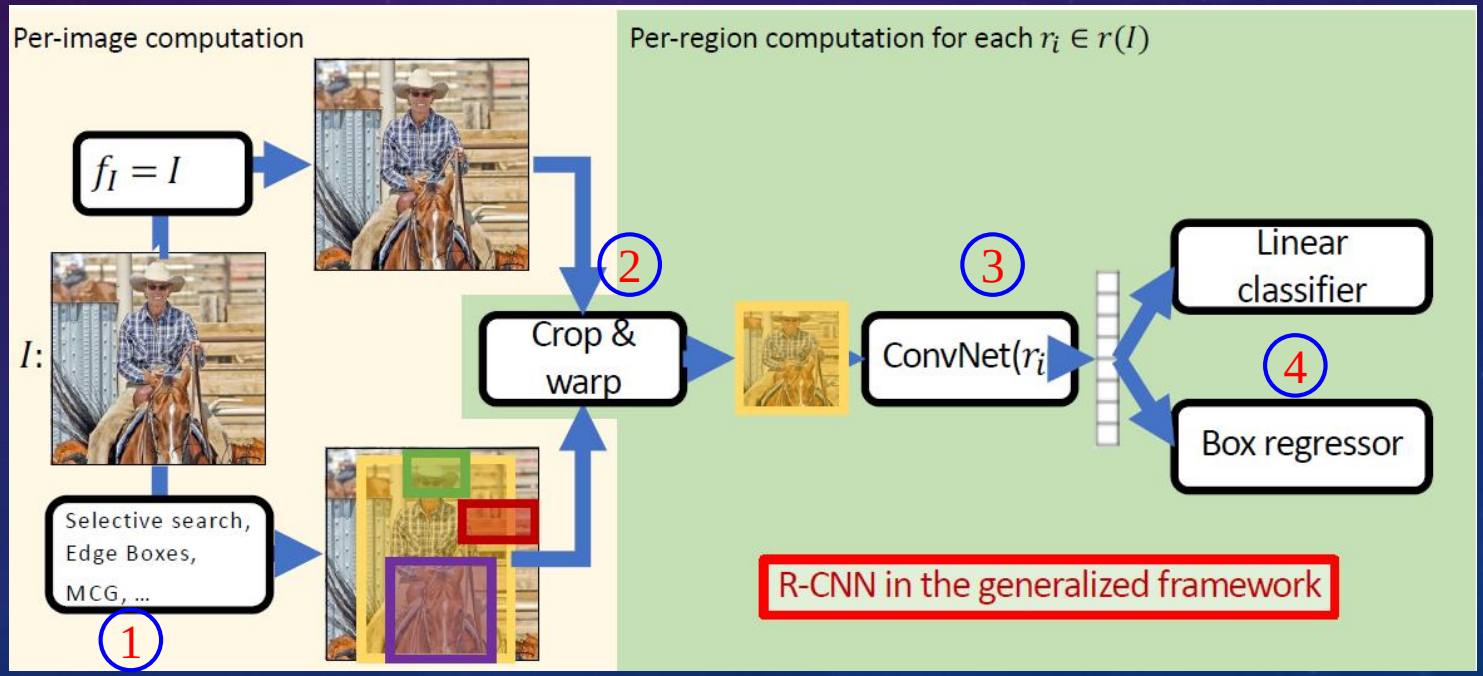
multiple heads to make prediction



R-CNN Algorithm

- Use selective search to find region proposal (~2k)
- Extract region of interest from original image
- Apply ConvNet to each region of interest
- Use support vector machine to find class label and linear regression to find bounding box offsets.

Problem:
- ✓ Very heavy per-region computation
- ✓ Ad hoc training objective
- ✓ Inference is very slow



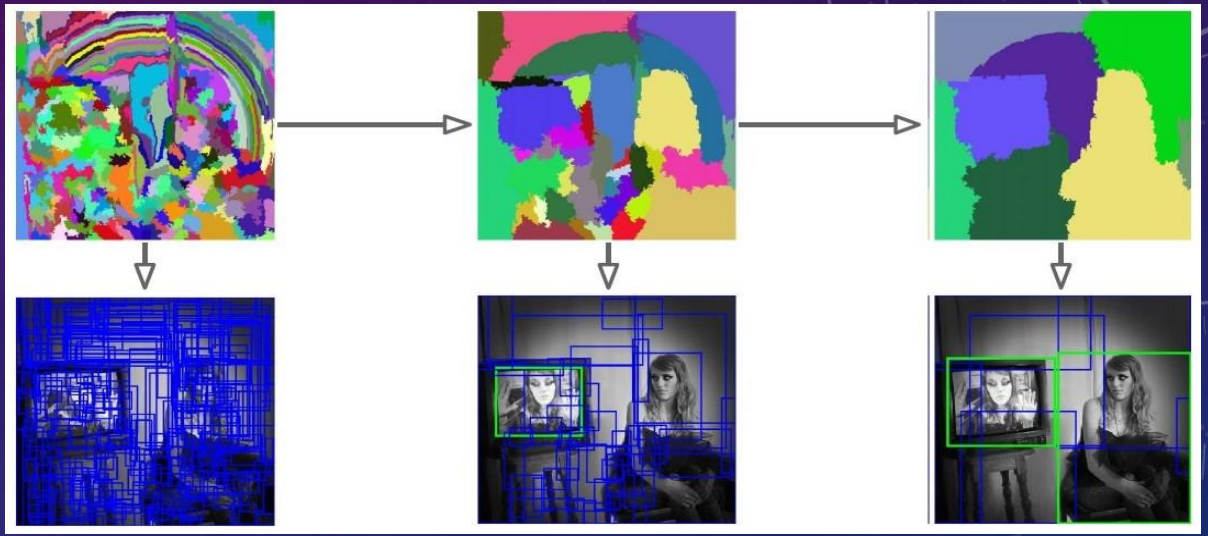
Detection and Segmentation – Stanford University School of Engineering

https://www.youtube.com/watch?v=nDPWywWRIRo&list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk&index=11

[Starting from 08:04] [1:14:25]

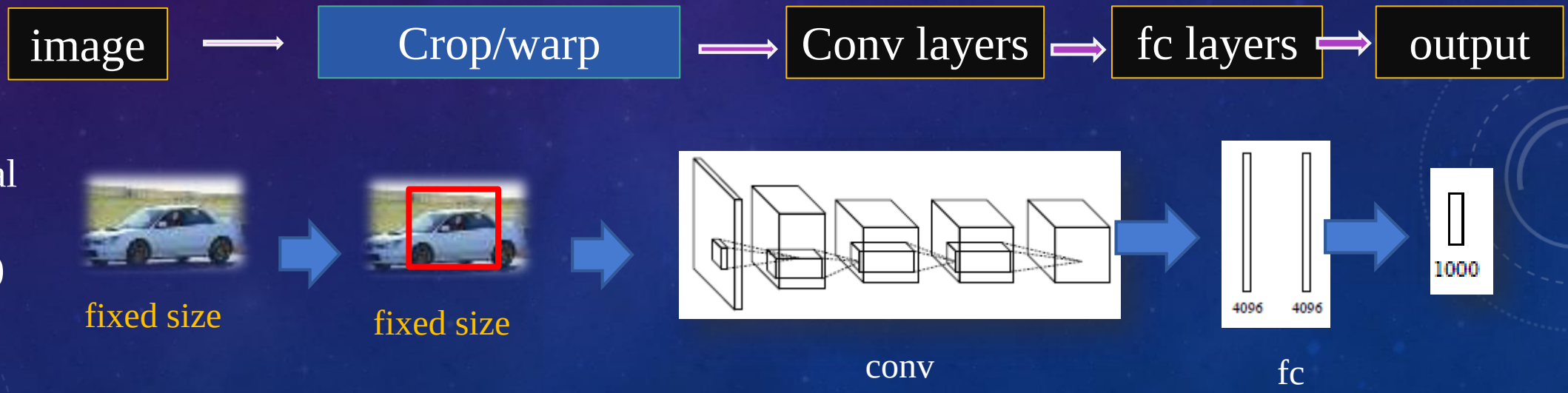
Region Proposals – Selective Search

- It is based on computing hierarchical grouping of similar regions based on color, texture, size and shape compatibility.
- Selective Search starts by over-segmenting the image based on intensity of the pixels using a graph-based segmentation method by Felzenszwalb and Huttenlocher.



R-CNN

- In a typical CNN structure, a full connection is usually connected after the convolutional layer. The number of features of the fully connected layer is fixed, so the input size (fixed-size) is fixed when the network is input. But in reality, the image size of our input is always unable to meet the size required for input. However, the usual methods are cropping and warping.

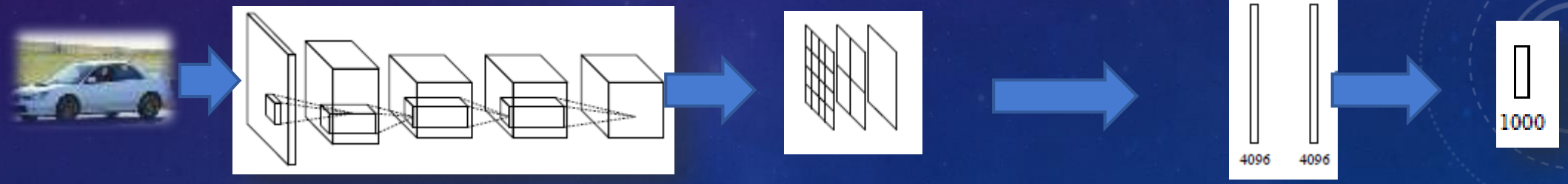


Spatial Pyramid Pooling net

- SPP can produce a fixed size output regardless of the input size
- Use multiple windows (pooling window)
- SPP can use the same image with different scales as input to get the same length of pooling features.

image → Conv layers → Spatial pyramid pooling → fc layer → output

SPP-net



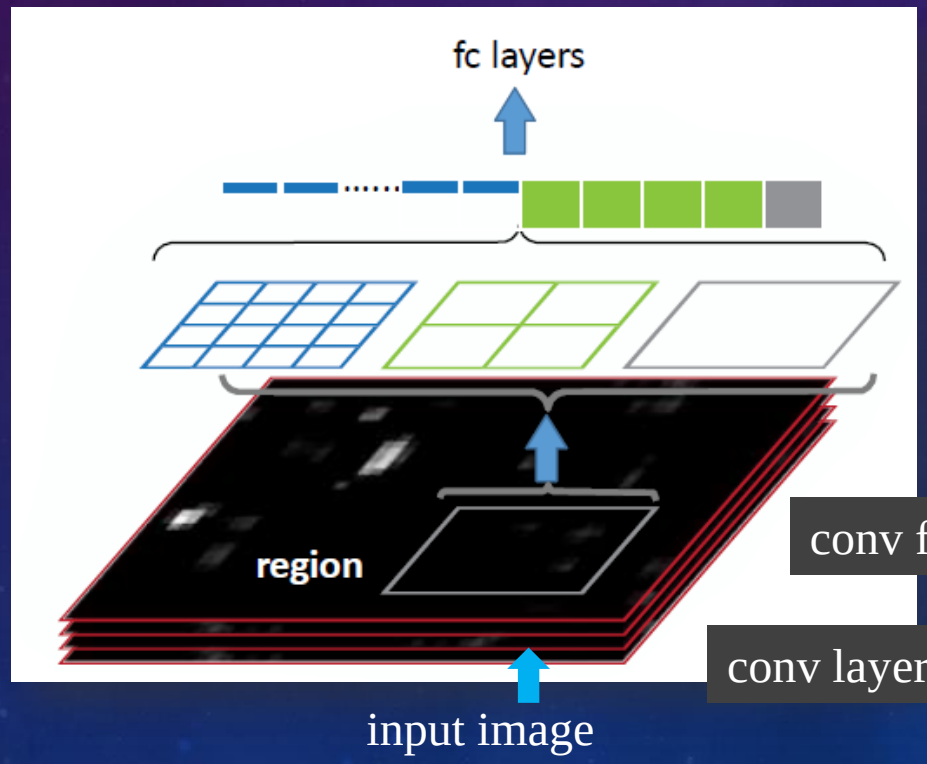
conv

Spatial Pyramid Pooling

Spatial Pyramid Pooling net

How do we get the fixed output?

SPP NET use multiple windows (pooling window, the blue, green, silver gray window in the input picture, then pooling the feature maps, the combined results will get a fixed length output).

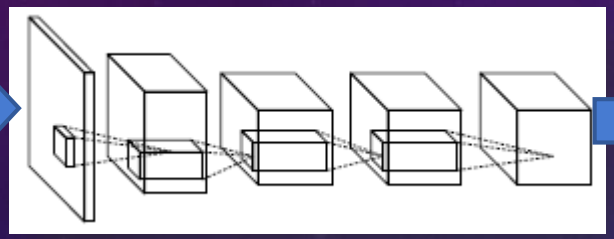


Spatial Pyramid Pooling

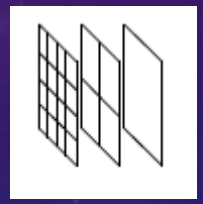
Spatial Pyramid Pooling (SPP)

image ⇒ Conv layers ⇒ Spatial pyramid pooling ⇒ fc layer ⇒ output

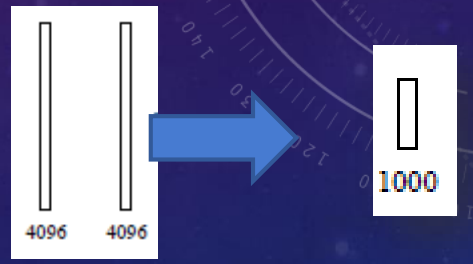
SPP-net



conv



Spatial Pyramid Pooling



Backward

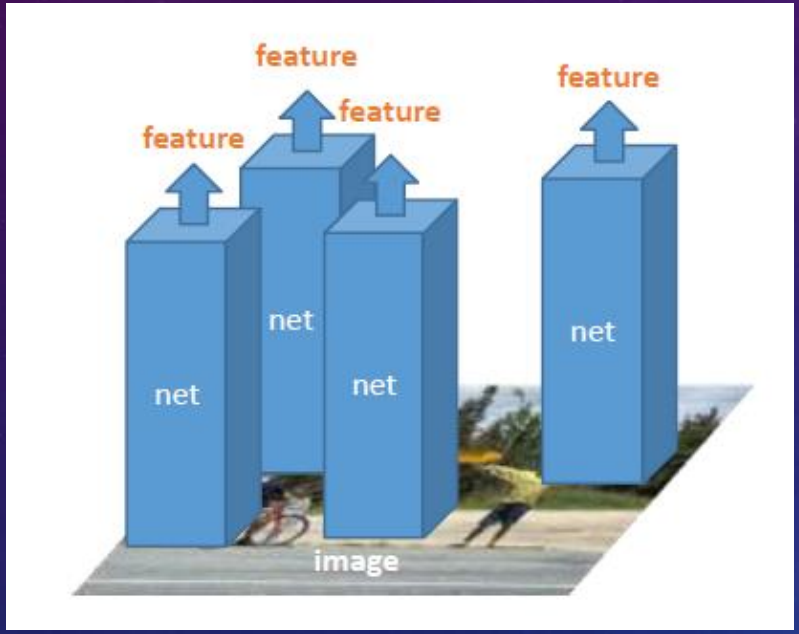
R times
Slow and Heavy
computation

R: # of RoIs

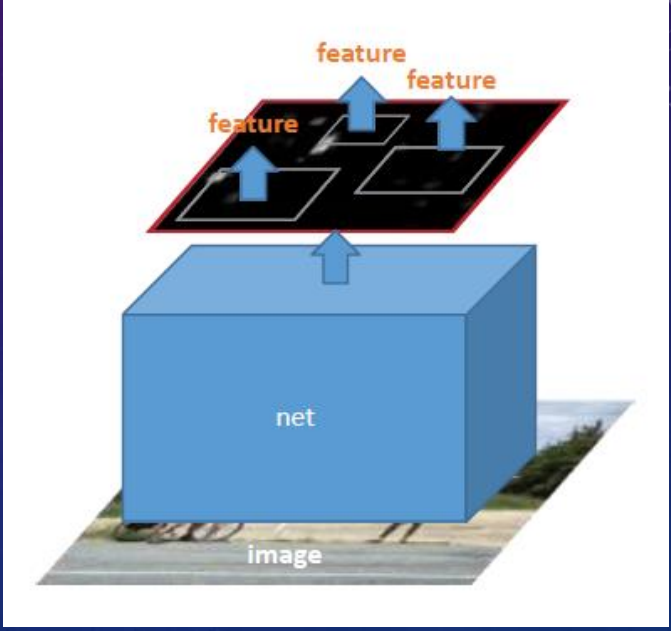
R times
back-propagation

RCNN vs. SPP

- image regions *vs*. feature map regions



R-CNN
2000 nets on image regions

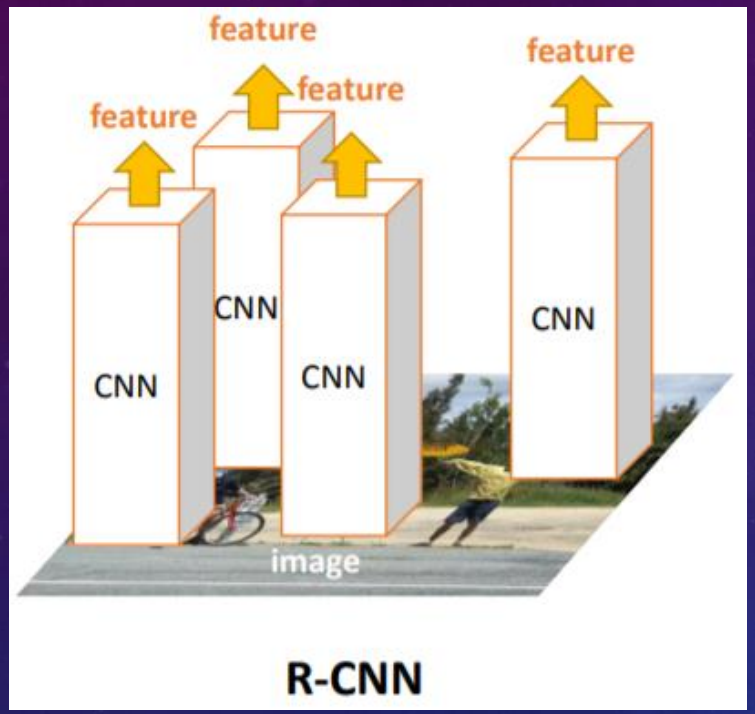


SPP-net
1 net on full image

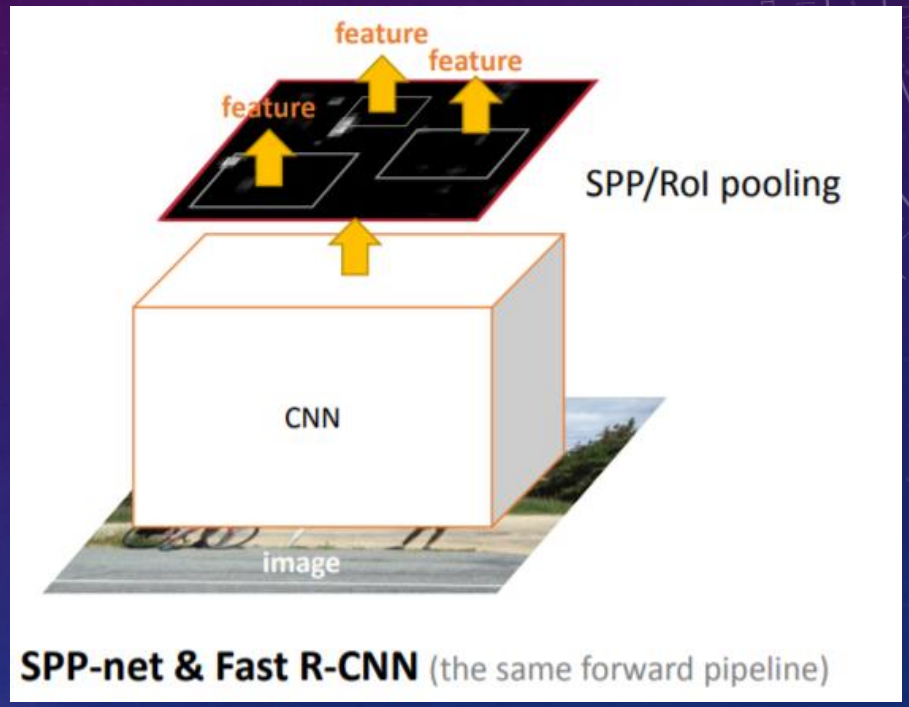
Fast R-CNN

- Spatial Pyramid Pooling net
- Fast R-CNN architecture
- What is the ROI pooling layer?
- ROI pooling Summary
- ROI pooling Summary
- Problems of Fast R-CNN

R-CNN vs. Fast R-CNN (Forward Pipeline)

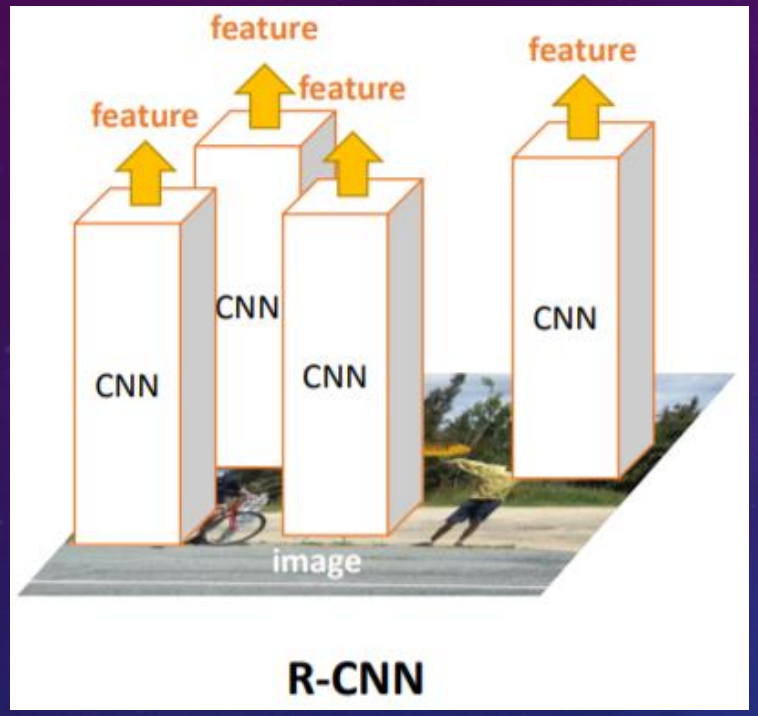


- Extract image regions
- 1 CNN per region (2000 CNNs)
- Classify region-based features

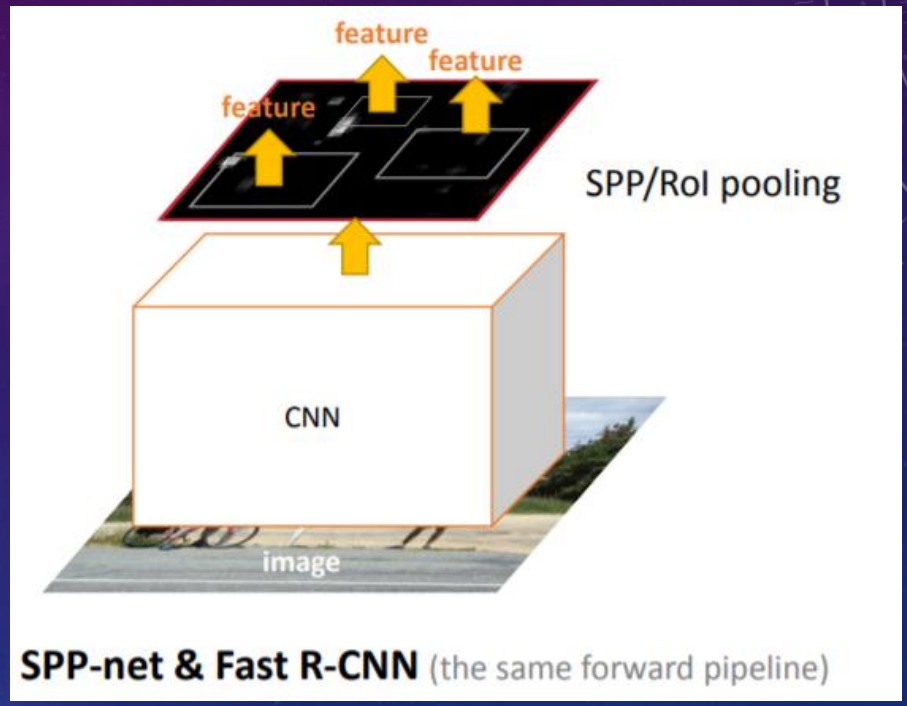


- 1 CNN on the entire image
- Extract features from feature map regions
- Classify region-based features

R-CNN vs. Fast R-CNN (Forward Pipeline)



- Complexity: ~224 × 224 × 2000



- Complexity: ~600 × 1000 × **1**
- ~160x faster than R-CNN

R-CNN vs. Fast R-CNN

- Fix most of what's wrong with R-CNN and SPP-net

- Train the detector in a single stage, end-to-end
  - No caching features to disk
  - No post hoc training steps

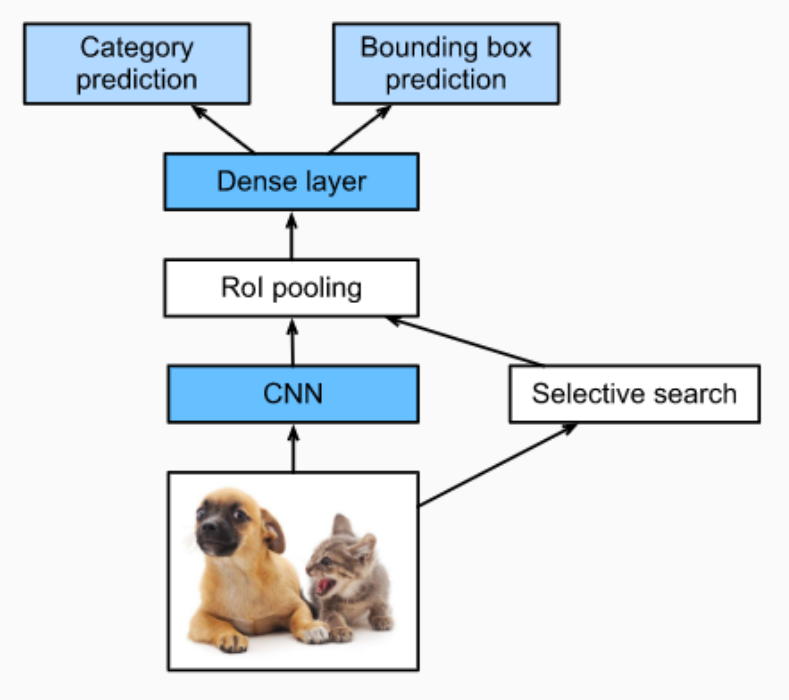
- Train all layers of the network

Problems of R-CNN

- **Slow at test-time**: need to run full forward path of CNN for each region proposal
  - 13s/image on a GPU(K40)
  - 53s/image on a CPU
- **SVM and regressors are post-hoc**: CNN features not updated in response to SVMs and regressors
- **Complex multistage training pipeline** (84 hours using K40 GPU)
  - Fine-tune network with softmax classifier(log loss)
  - Train post-hoc linear SVMs(hinge loss)
  - Train post-hoc bounding-box regressions(squared loss)

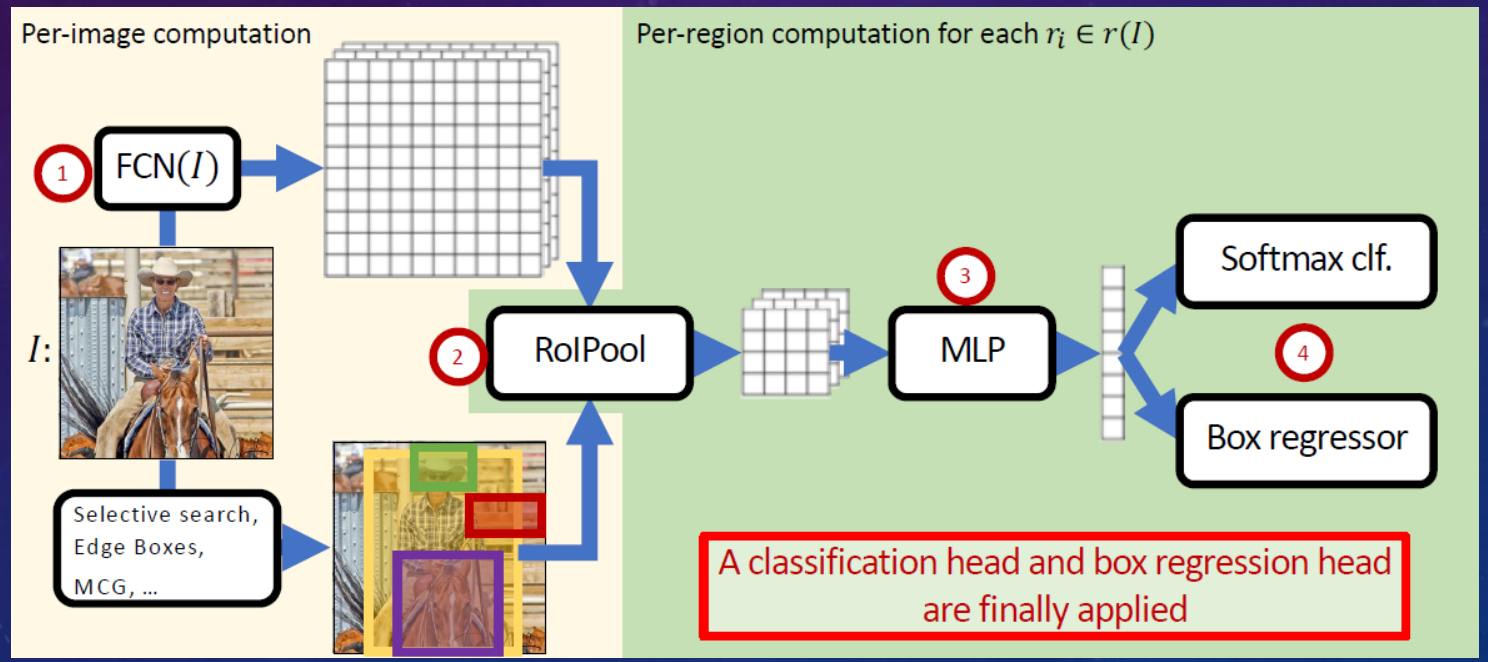
FAST R-CNN

- The main performance bottleneck of an R-CNN model is the need to independently extract features for each proposed region.
- As these regions have a high degree of overlap, independent feature extraction results in a high volume of repetitive computations. Fast R-CNN improves on the R-CNN by only performing CNN forward computation on the image as a whole.



FAST R-CNN

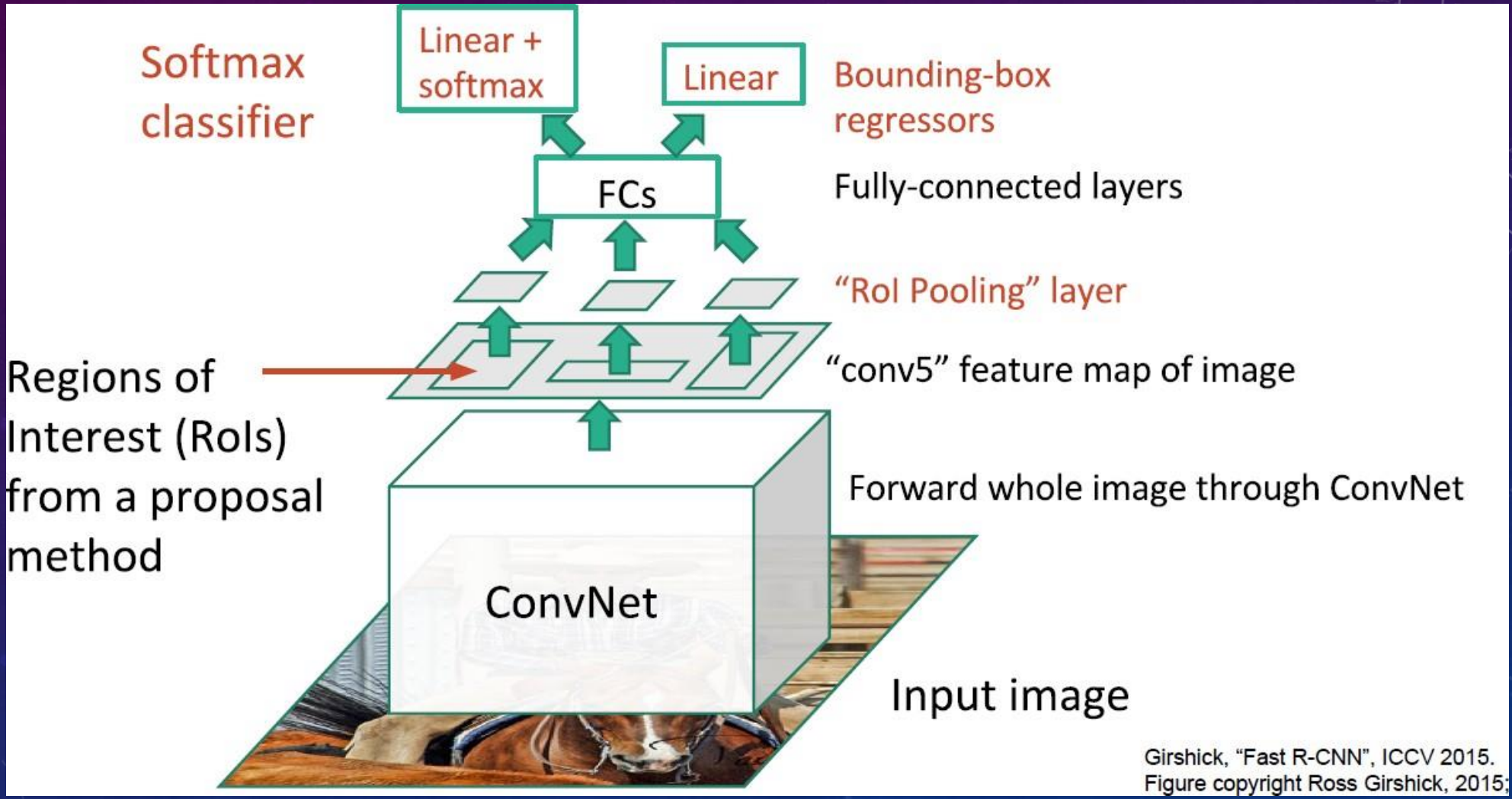
- Use fully convolutional network to extract features
- Extract region of interest from feature map and use region of interest (*RoI*) pooling layer extracts a fixed-length feature vector from the feature map.
- A sequence of fully connected (*fc*) layers that finally branch into two sibling output layers: one that produces softmax probability estimates over K object classes plus a "background" class and another layer that outputs four real-valued numbers for each of the K object classes.



Problem:

- ✓ Use selective search still slow
- ✓ Not end-to-end training

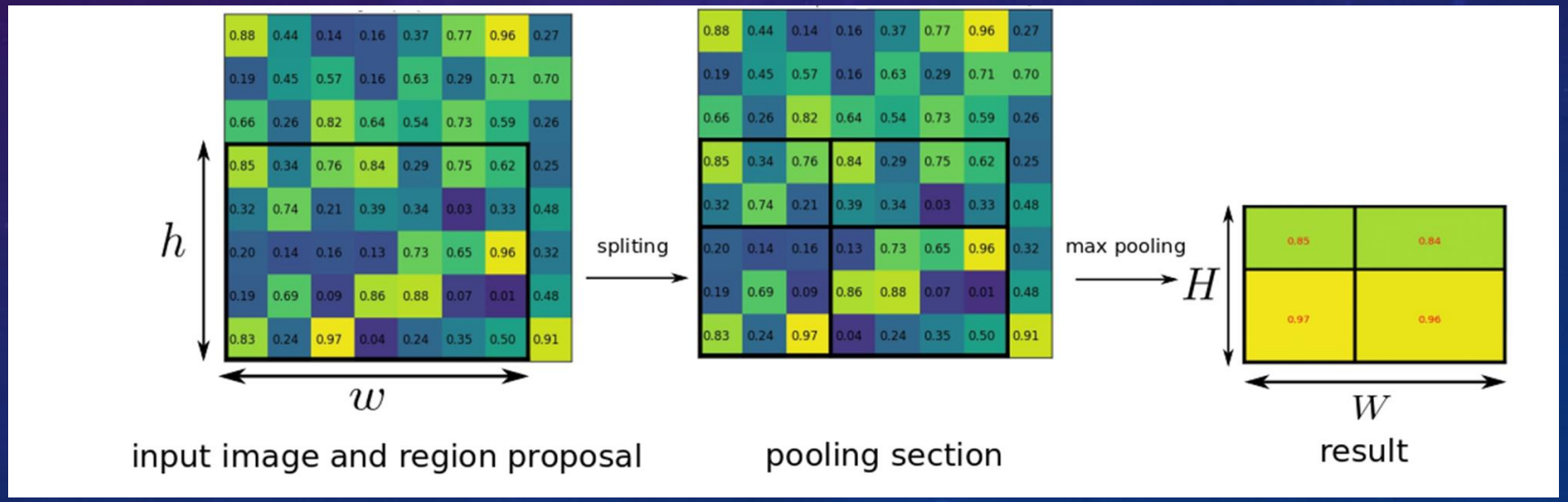
FAST R-CNN



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015;

What is the ROI pooling layer?

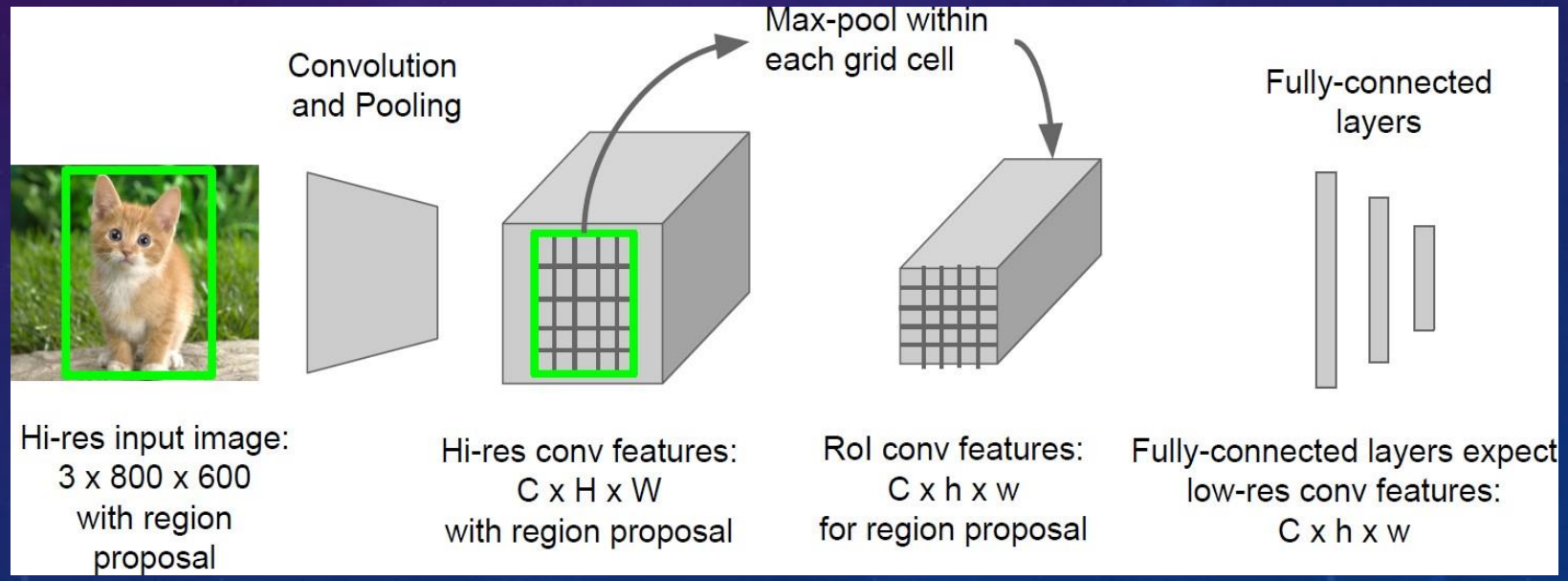
- Suppose we got the region proposal (left) with $h \times w$, and we would like to have an output (right) of H×W sizes of output layer after pooling. Then, the area for each pooling area (middle) = $h/H \times w/W$.
- And in the example above, with input ROI of 5×7, and output of 2×2, the area for each pooling area is 2×3 or 3×3 after rounding.
- And the maximum value within the pooling window is taken as output value for each grid which is the same idea of conventional max pooling layer.



How does ROI pooling work?

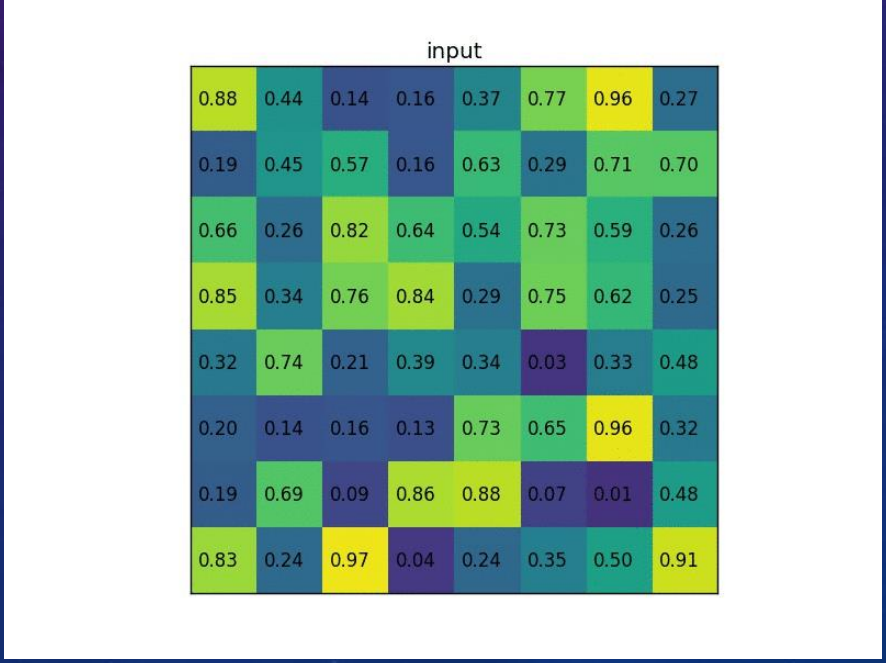
- RoI max pooling works by dividing the **h × w** RoI window into an **H × W** grid of sub-windows of approximate size **h/H × w/W** and then max-pooling the values in each sub-window into the corresponding output grid cell.
- i.e. the "max-pooling" filter size becomes **h/H × w/W**. Pooling is applied how it is normally applied, independently to each feature map channel.

HOW DOES ROI POOLING WORK?



ROI pooling Summary

- ROI pooling summary:
- It’s used for object detection tasks
- It allows us to reuse the feature map from the convolutional network
- It can significantly speed up both train and test time
- It allows to train object detection systems in an end-to-end manner



Problems of Fast R-CNN

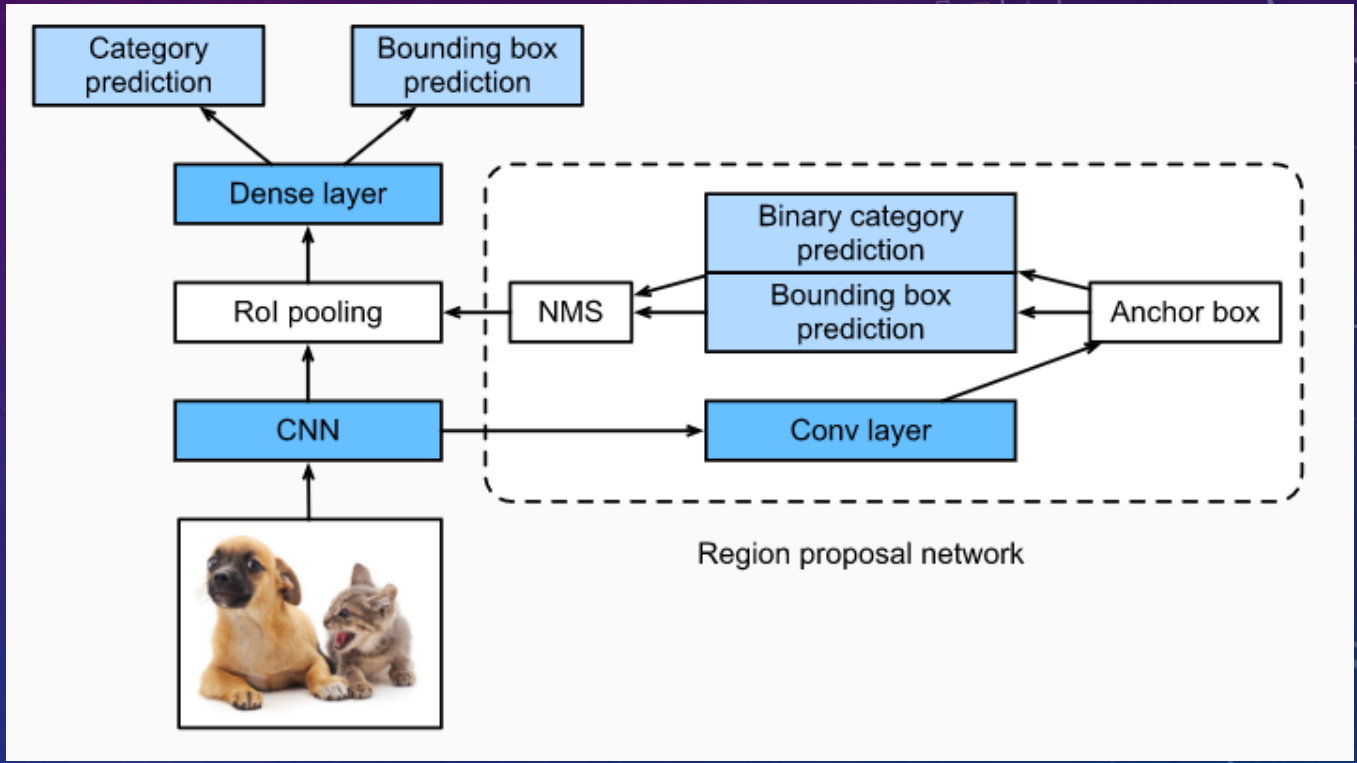
- Out-of-network region proposals are the test-time computational bottleneck
- Is it fast enough??

Faster R-CNN

- Faster R-CNN architecture
- Faster R-CNN(RPN + fast R-CNN)
- How to train faster R-CNN
- Training goal : share features
- How RPN (region proposal networks) works
- Region Proposal Network
- RPN(fully convolutional network)

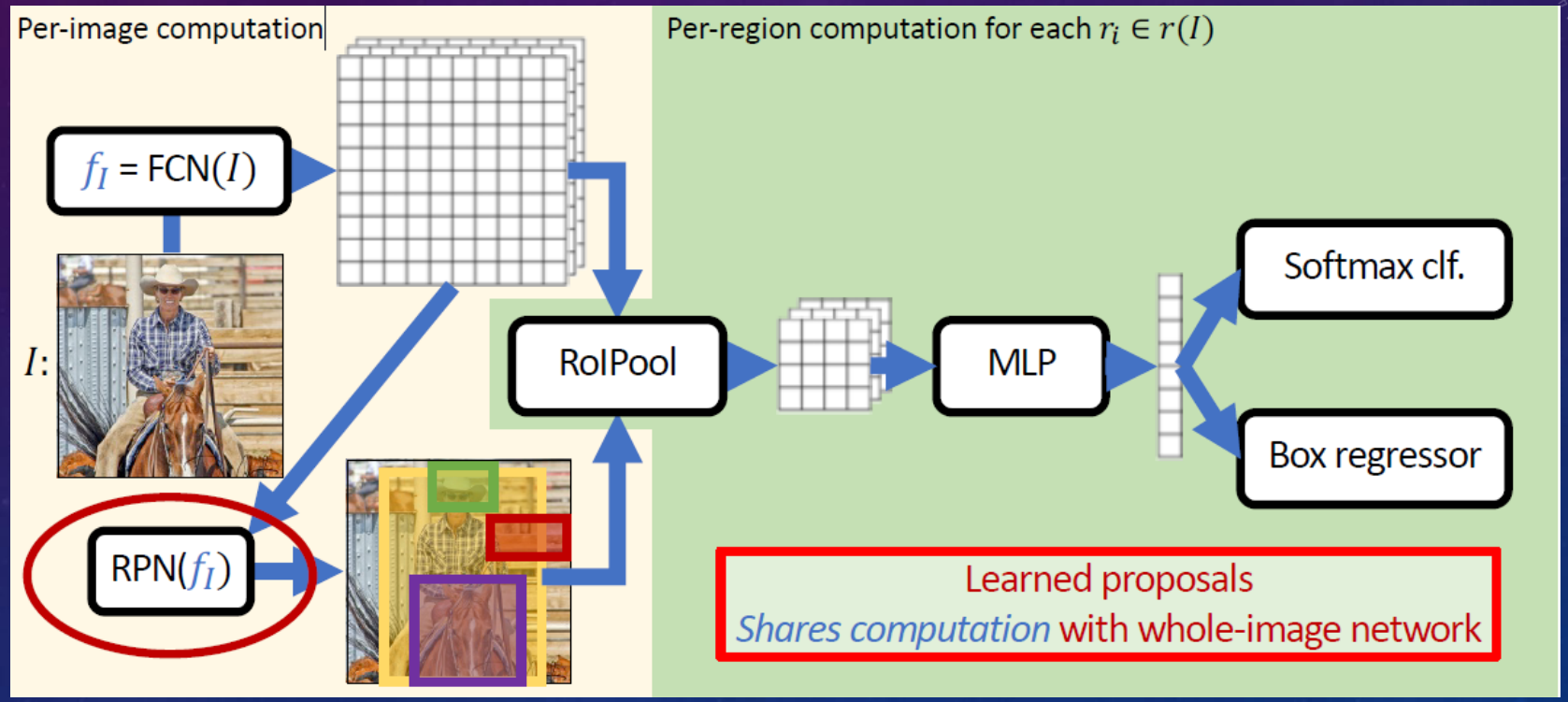
Faster R-CNN

- In order to obtain precise object detection results, Fast R-CNN generally requires that many proposed regions be generated in selective search. Faster R-CNN replaces selective search with a region proposal network. This reduces the number of proposed regions generated, while ensuring precise object detection.



Faster R-CNN

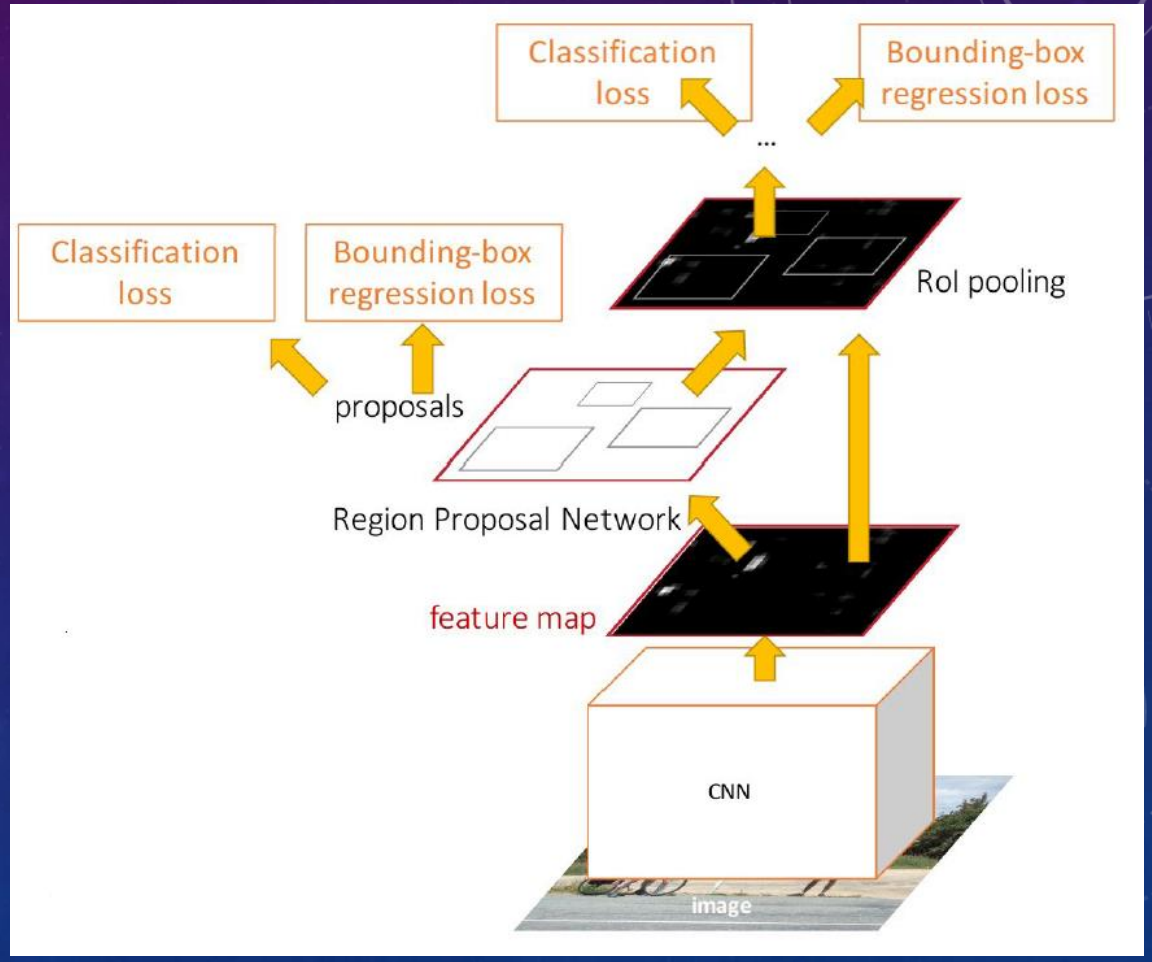
- Replace selective search with Region Proposal Network
- End-to-end training



Faster R-CNN

- Jointly train the whole network with 2 losses
  - RPN classifies object/non-object
  - RPN regresses bbox coordinates
- Results show the final classification score and bbox coordinates.

$$L = L_{cls} + L_{reg}$$



Loss Function

$$L_{cls} = -\frac{1}{N_{cls}} \sum_{\mathrm{i}} L_{cls}(p_i, p_i^*)$$

- Here i is the index of the anchor in the mini-batch.
- $L_{cls}(p_i, p_i^*)$: log loss over two classes (object vs non-object)
- p_i: output score from the classification branch for anchor i
- p_i^*: groundtruth label (1 or 0).

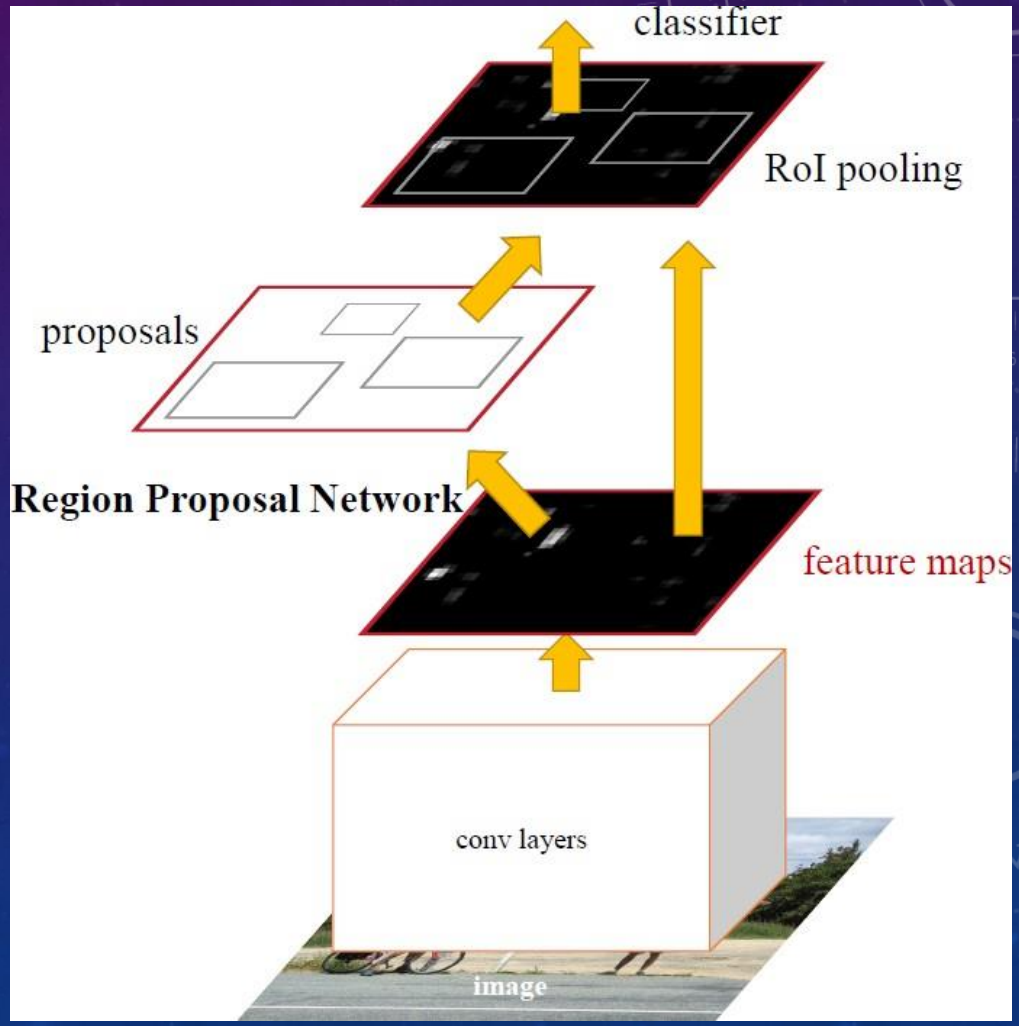
Loss Function

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

- The regression loss $L_{reg}(t_i, t_i^*)$ is activated only if the anchor actually contains an object.
- t_i: The output prediction of the regression layer and consists of 4 variables $[t_x, t_y, t_w, t_h]$.
- t_i^*: $t_x^* = (x^* - x_a)/w_a$, $t_y^* = (y^* - y_a)/h_a$, $t_w^* = \log(w^*/w_a)$, $t_h^* = \log(h^*/h_a)$

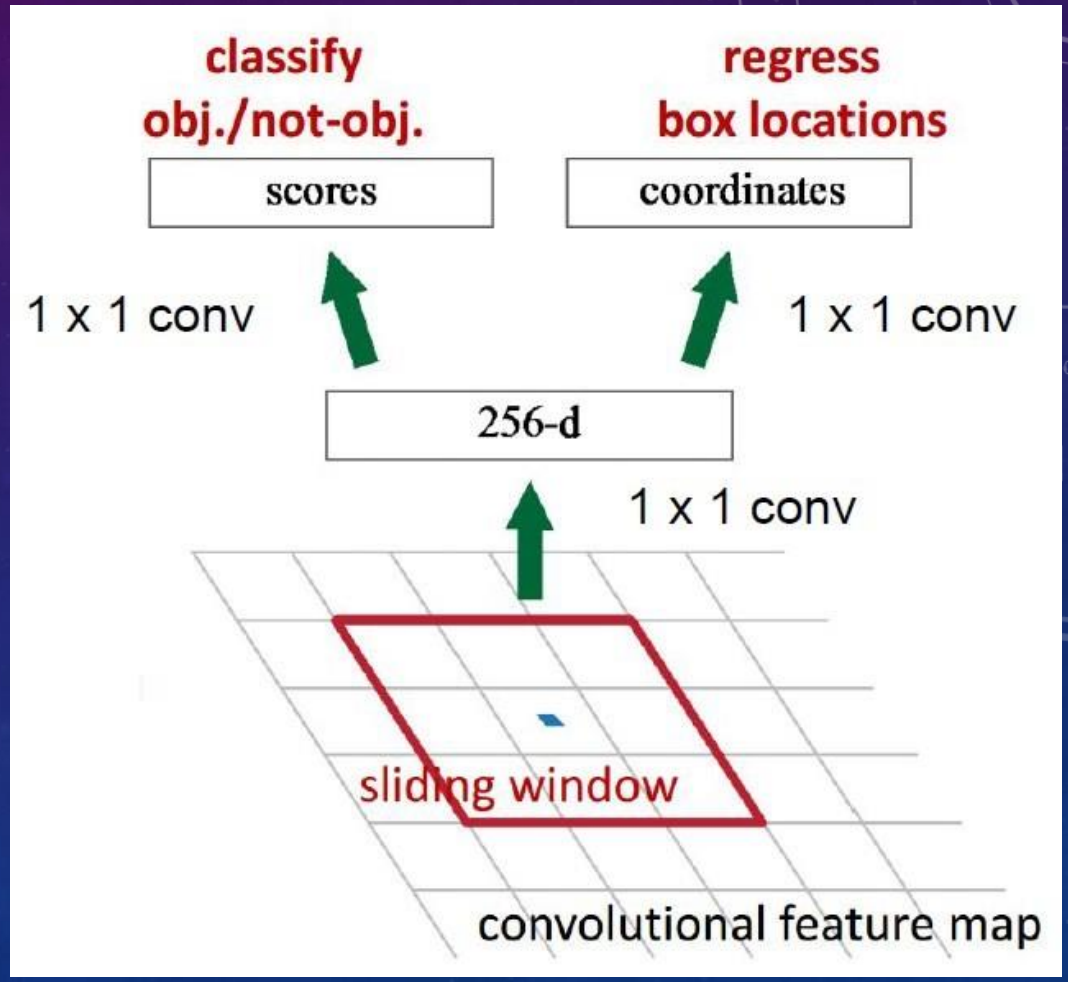
Faster R-CNN (RPN + Fast R-CNN)

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



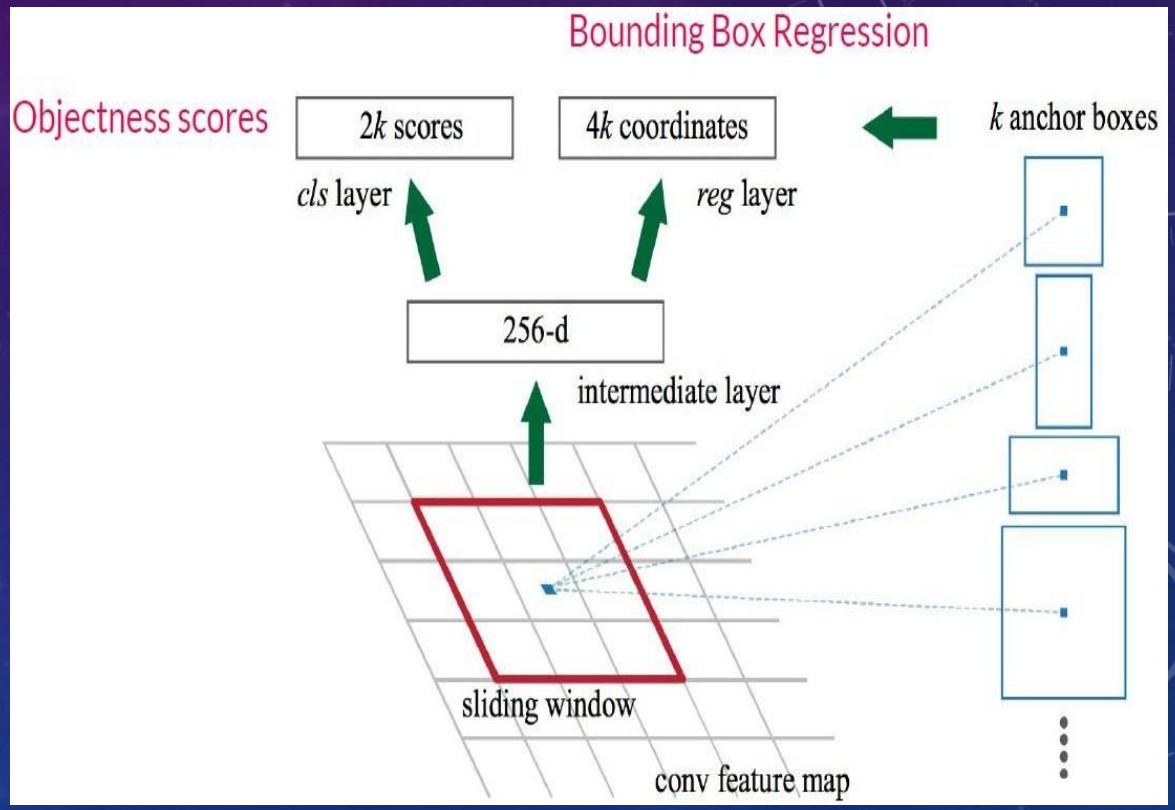
Region Proposal Network

- Slide a small window on the feature map
- Build a small network for
  - Classifying object or not-object
  - Regressing bbox locations
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window



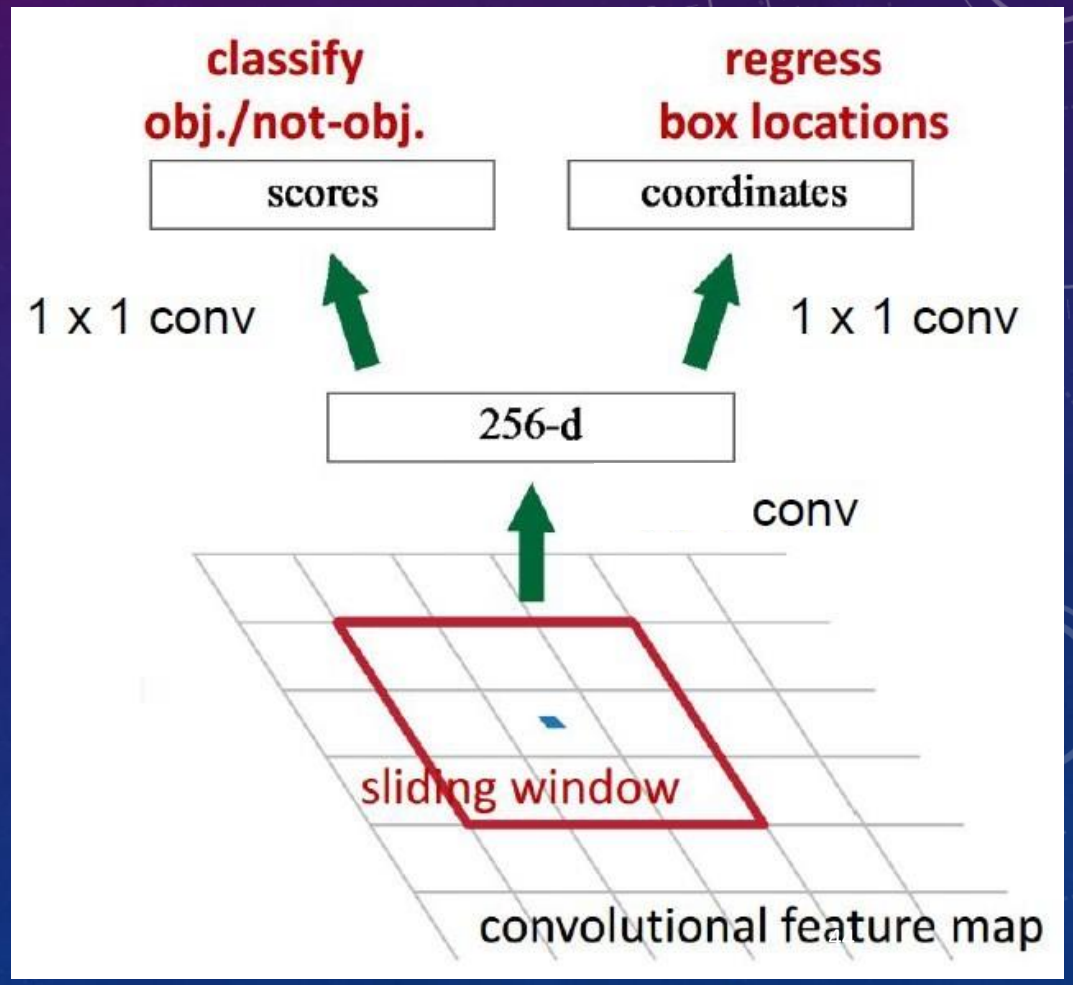
Region Proposal Network

- Use K anchor boxes at each location
- Anchors are translation invariant: use the same ones at every location
- Regression gives offsets from anchor boxes
- Classification gives the probability that each (regressed) anchor shows an object



RPN (Fully Convolutional Network)

- Intermediate Layer – 256(or 512) 3x3 filter, stride 1, padding 1
- Cls layer – 18(9x2) 1x1 filter, stride 1, padding 0
- Reg layer – 36(9x4) 1x1 filter, stride 1, padding 0



Anchors As References

- Anchors: pre-defined reference boxes
- Multi-scale/size anchors:
  - Multiple anchors are used at each position:
    3 scale(128x128, 256x256, 512x512) and 3 aspect rations(2:1, 1:1, 1:2) yield 9 anchors
  - Each anchor has its own prediction function
- Single-scale features, multi-scale predictions

Is Faster RCNN enough?

- RoI Pooling has some quantization operations
- These quantizations introduce misalignments between the RoI and the extracted features
- While this may not impact classification, it can make a negative effect on predicting bbox

See Coding Manual, Chapter 4, Page 2.

Mask R-CNN

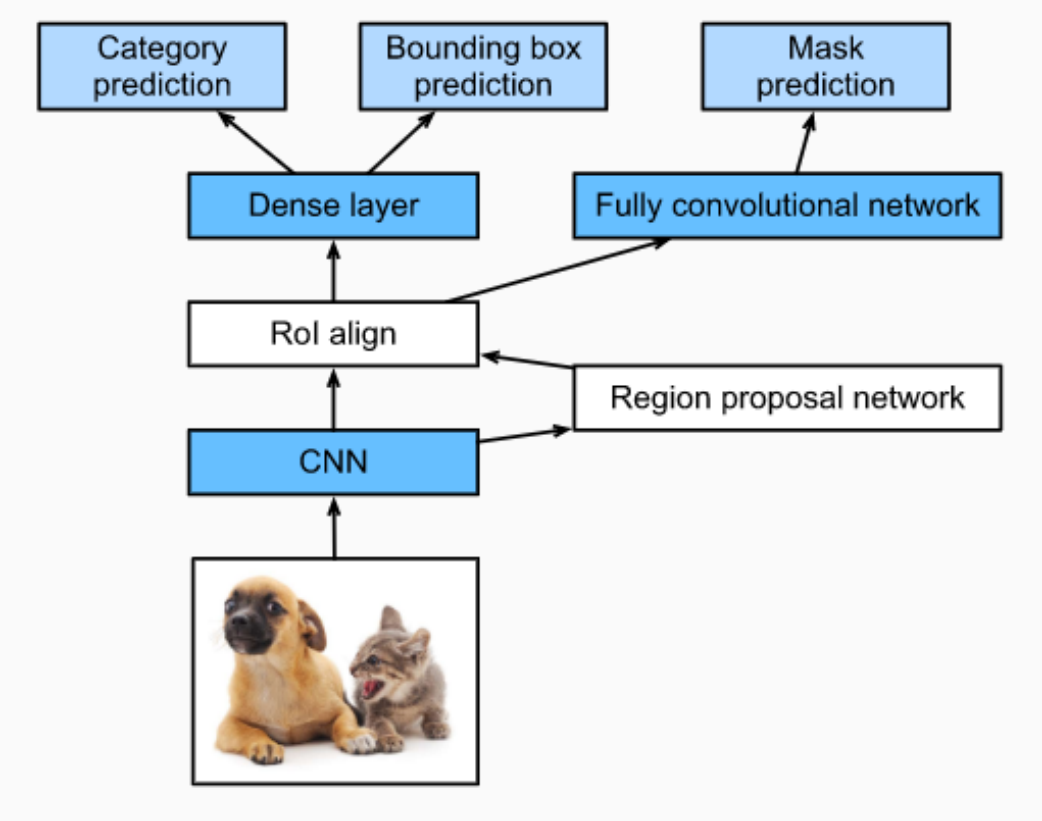
- Abstract
- Semantic Segmentation
- Instance Segmentation
- General Architecture
- Mask R-CNN RoI Alignment
- Mask R-CNN Network architecture

Mask R-CNN

https://www.youtube.com/watch?v=4tkgOzQ9yyo [9:34]

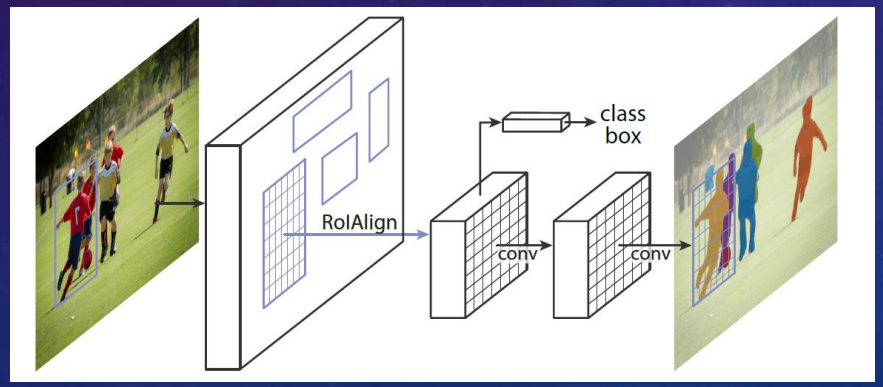
Mask R-CNN

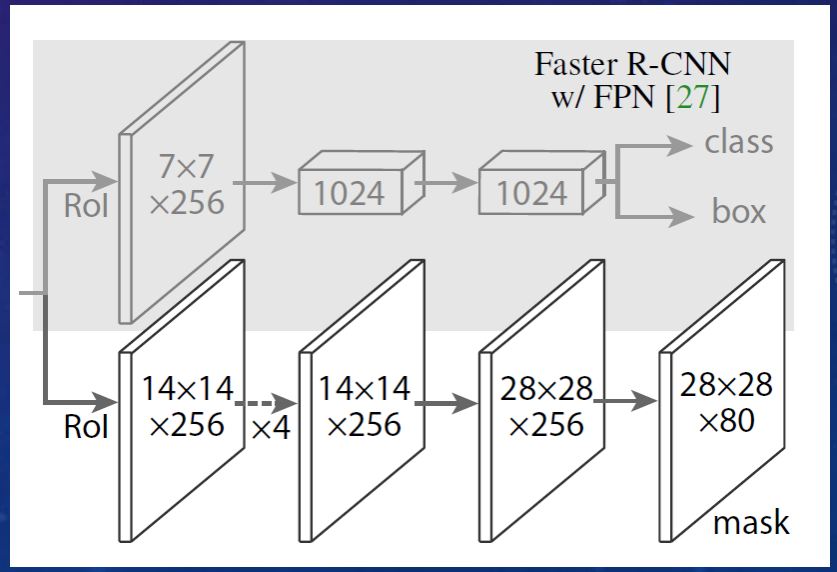
If training data is labeled with the pixel-level positions of each object in an image, a Mask R-CNN model can effectively use these detailed labels to further improve the precision of object detection.



Mask R-CNN

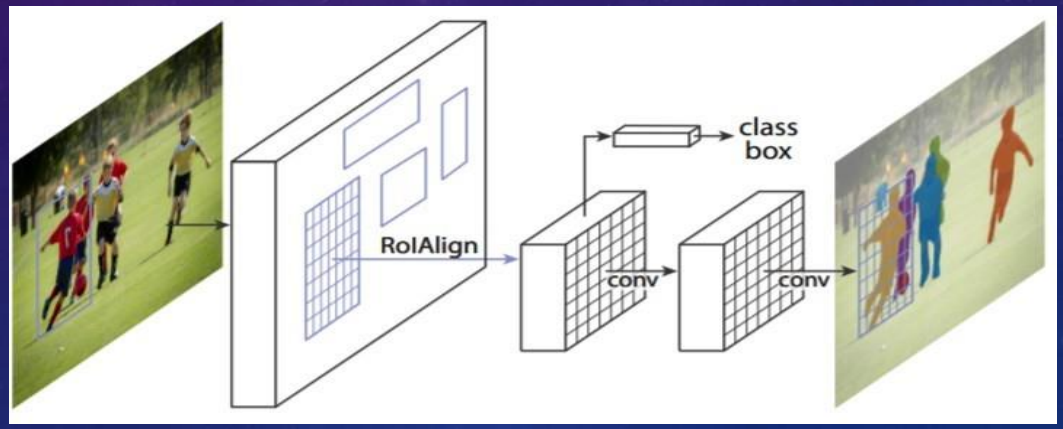
- Mask R-CNN, extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression
- The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner.
- RoI Alignment: use bilinear interpolation to compute the exact values of the input features at four regularly sampled locations in each RoI bin.





Abstract

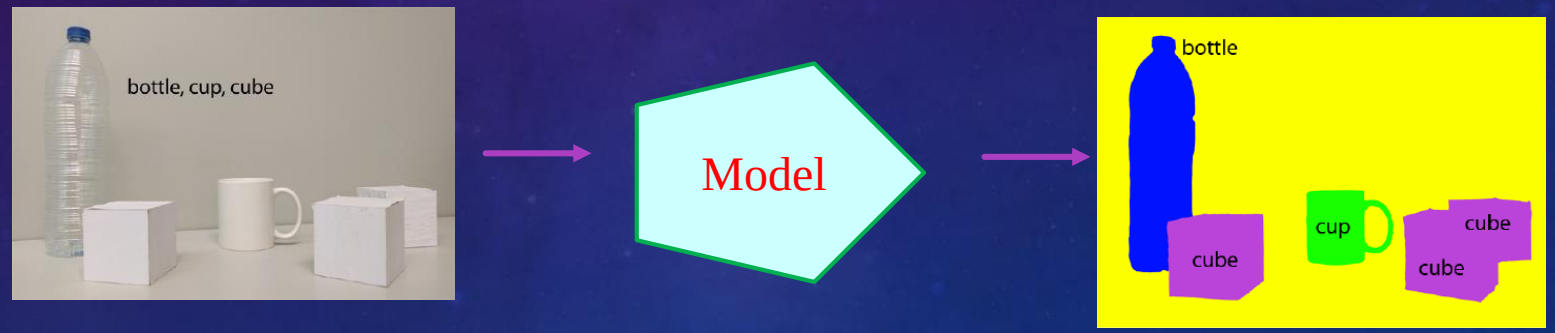
- Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance.
- Mask R-CNN extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.



Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5fps.

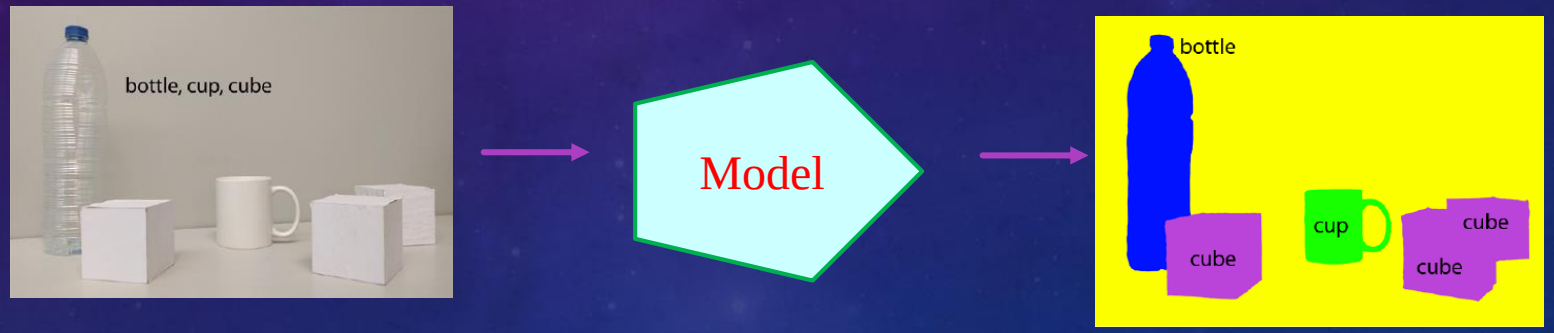
Semantic Segmentation

- Semantic segmentation refers to the process of linking each pixel in an image to a class label.
- These labels could include a person, car, flower, piece of furniture, etc., just to mention a few. We can think of semantic segmentation as image classification at a pixel level.



Instance Segmentation

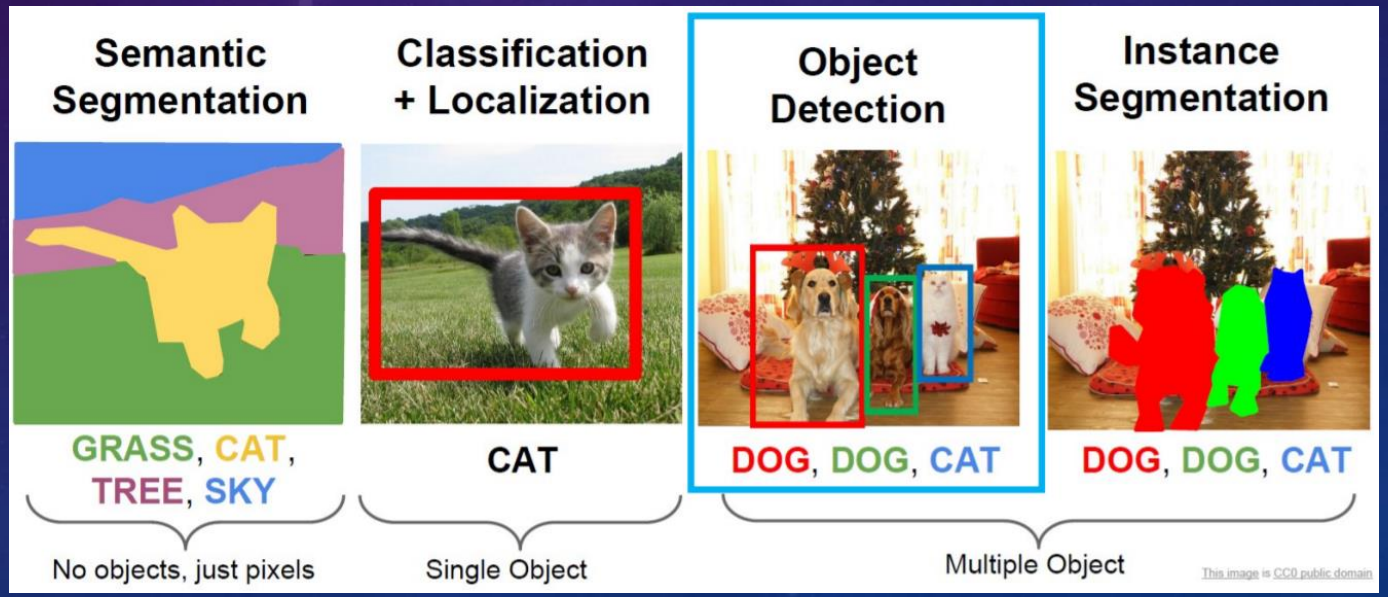
- For each pixel *I*, predict semantic label *l* and instance id *z*
- Instance segmentation = semantic segmentation + distinguished instances per class
- Closer to real scene understanding



Instance Segmentation

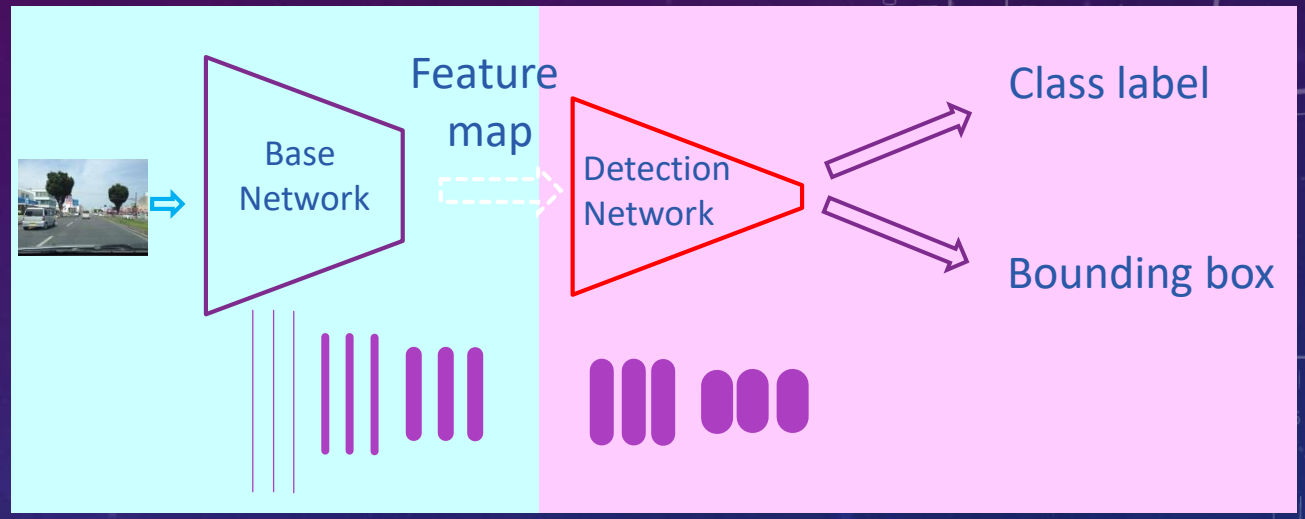
Instance segmentation combines elements from the classical computer vision tasks of object detection and segmentation.

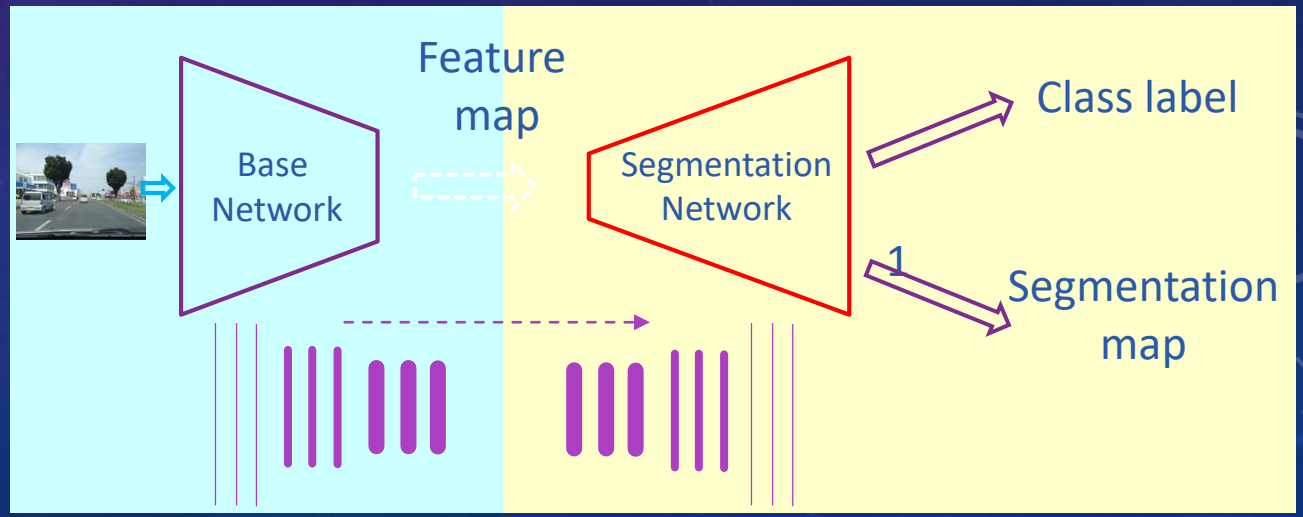
- Object detection is to classify individual objects and localize each with a bounding box
- Semantic segmentation, where the goal is to classify each pixel into a fixed set of categories without differentiating object instances.



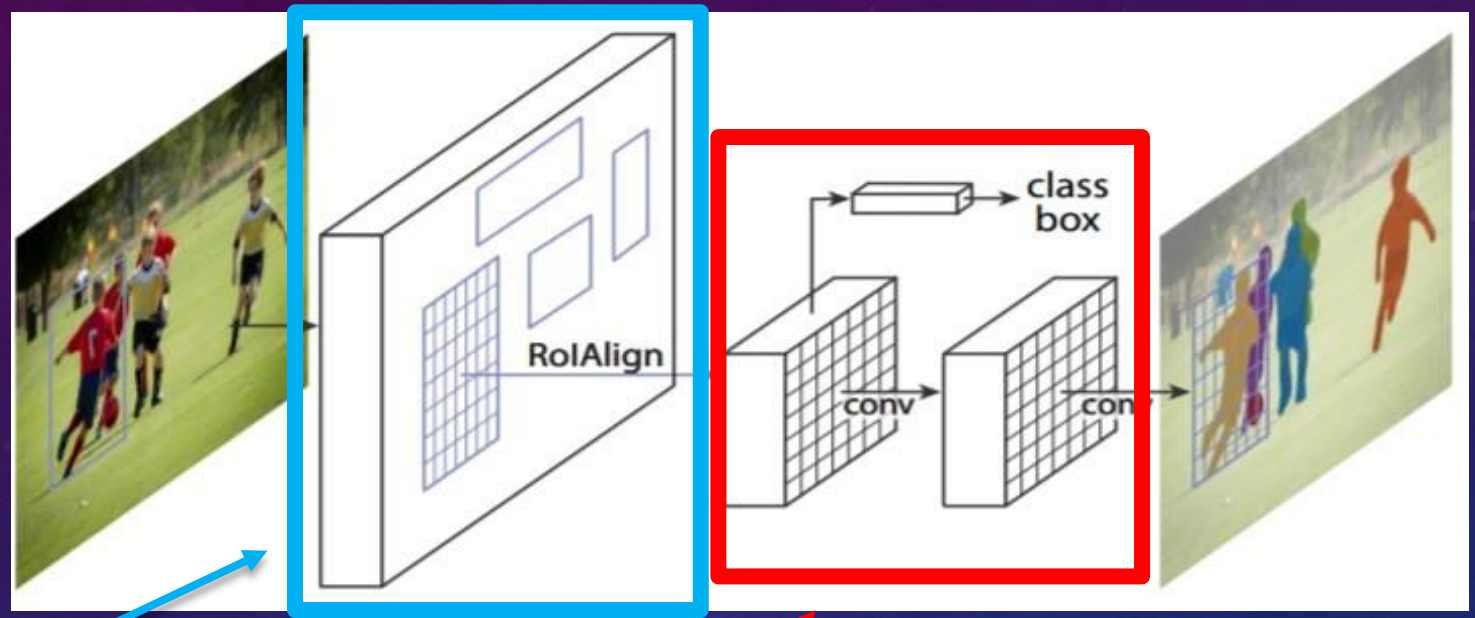
General Architecture

- Object Detection
  - Height and width decrease
  - Channel increase
  - Base network is pre-trained model
  - Output is a list of scalars or vectors
- Segmentation
  - Height and width decrease, then increase again.
  - Channel increases and then decreases
  - Base network is pre-trained model
  - Outputs are map and a lists of scalar.





Mask R-CNN Contribution



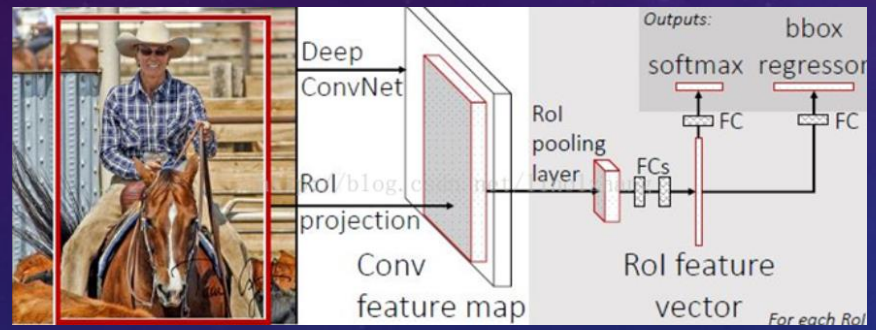
1. To fix the mis-alignment, we propose a simple, quantization-free layer, called RoI Alignment, that faithfully preserves exact spatial locations.

2. Adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression.

Mask R-CNN

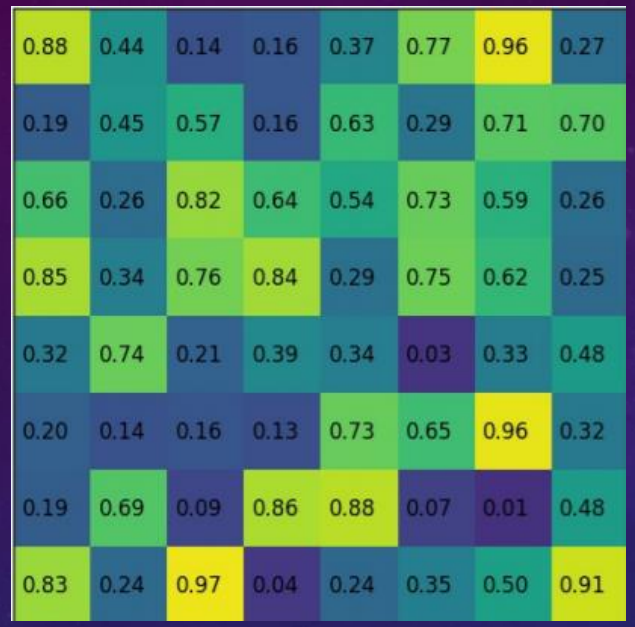
We propose an RoI Alignment layer that removes the harsh quantization of RoI Pooling, properly aligning the extracted features with the input. RoI Alignment improves mask accuracy by relative 10% to 50%, showing bigger gains under stricter localization metrics

- Previous works - RoIPool

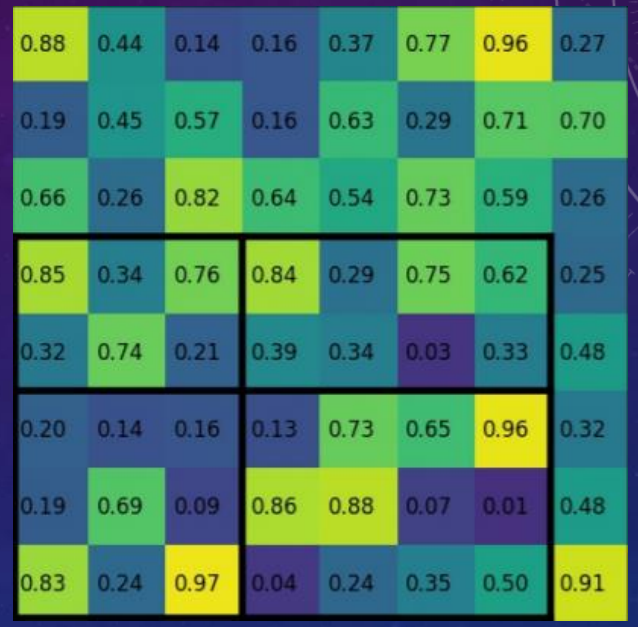


Max-pooling (Input is rounded of)

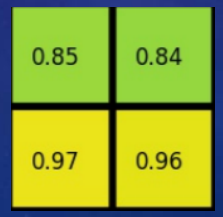
Faster R-CNN ROI Pooling



Input activation

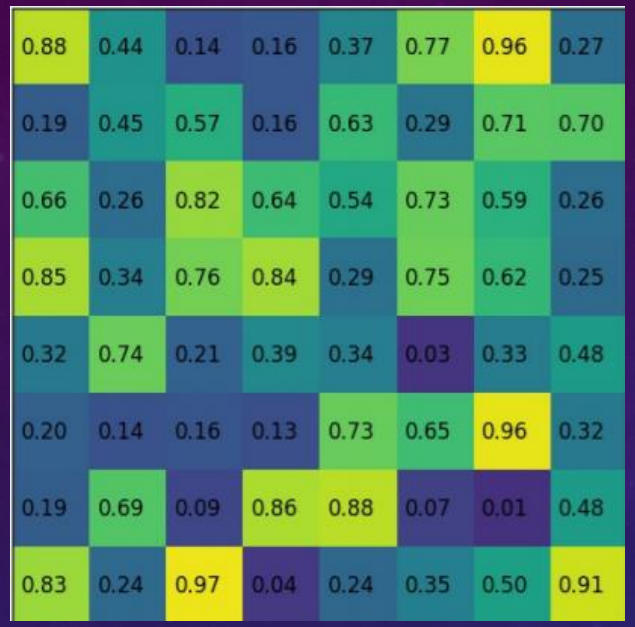


Region projection and pooling sections



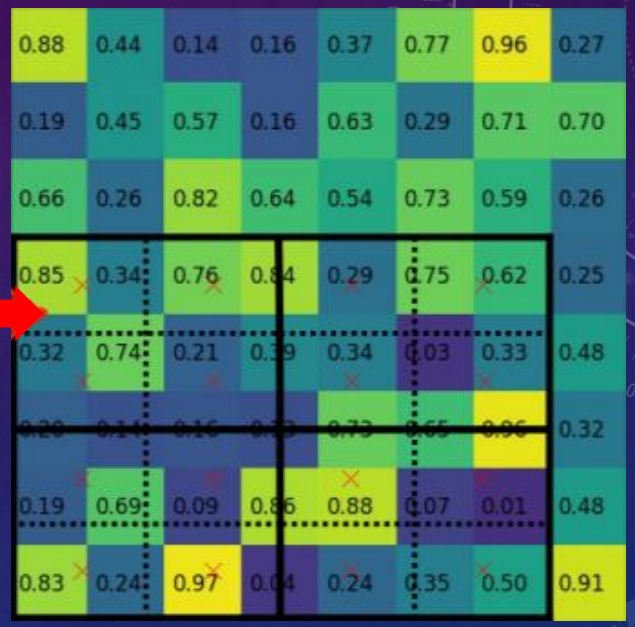
Max pooling output

Mask R-CNN RoI Alignment



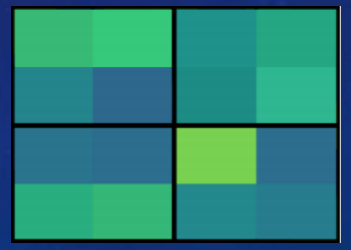
Input activation

Region projection and pooling sections



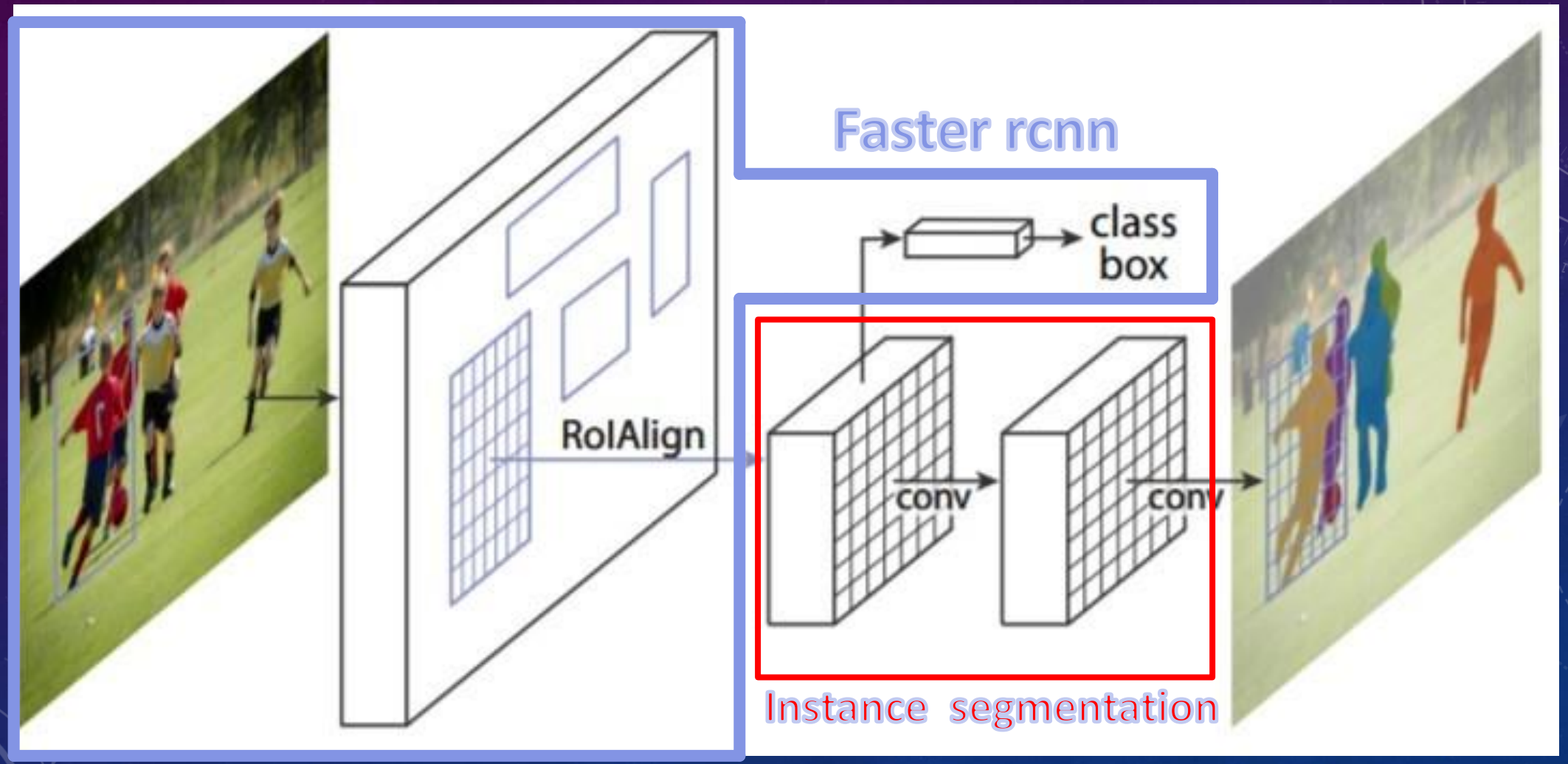
Sampling locations

Max pooling output



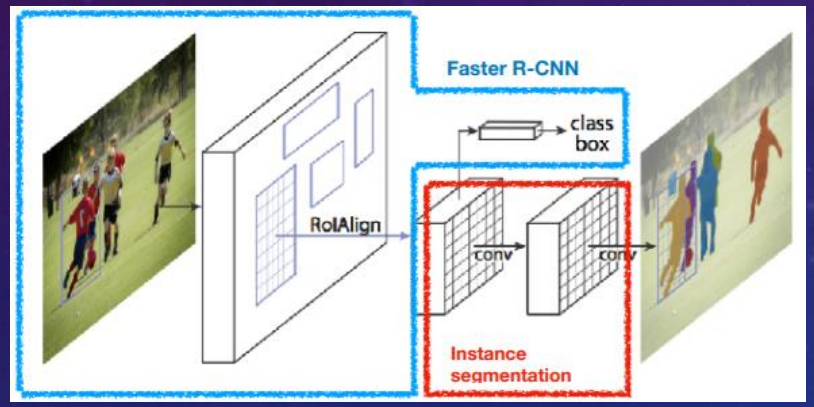
Bilinear interpolated values

Mask R-CNN Architecture



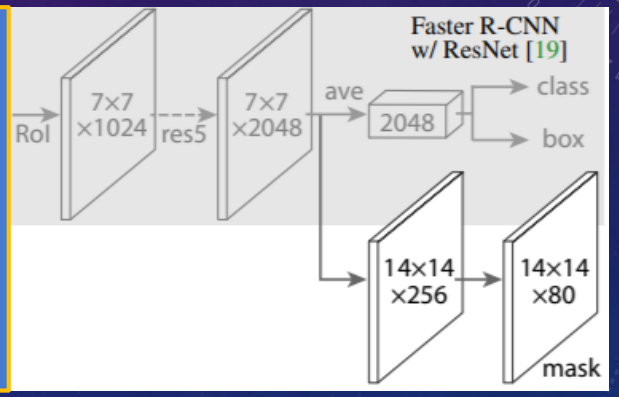
Mask R-CNN Network Architecture

- Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression

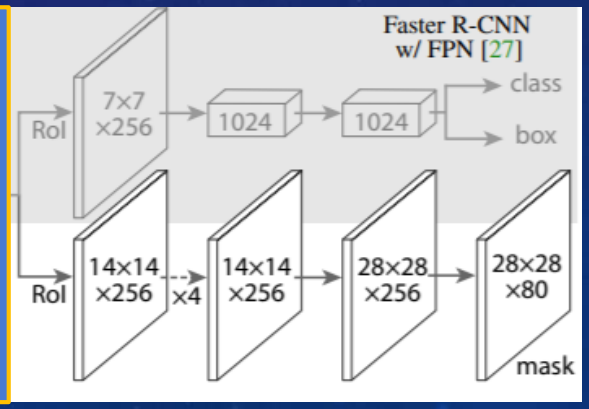


Faster R-CNN + Instance segmentation

ResNet or ResNeXt



FPN (Feature Pyramid Network)



Loss Function

$$L = L_{cls} + L_{reg} + L_{mask}$$

- L_{cls}, L_{reg} is the same in the faster R-CNN

Loss Function

$$L_{mask} = -\frac{1}{m^2}\sum_{1\le i,j\le m}\left[y_{i,j}\log\hat{y}_{i,j}^{k} + \left(1-y_{i,j}\right)\log\left(1-\hat{y}_{i,j}^{k}\right)\right]$$

- The mask branch has a K x m x m - dimensional output for each RoI and each class; K classes in total.
- L_{mask} is defined as the average binary cross-entropy loss, only including k-th mask if the region is associated with the ground truth class k

Summary

•R-CNN model selects several proposed regions and uses a CNN to perform forward computation and extract the features from each proposed region. It then uses these features to predict the categories and bounding boxes of proposed regions.

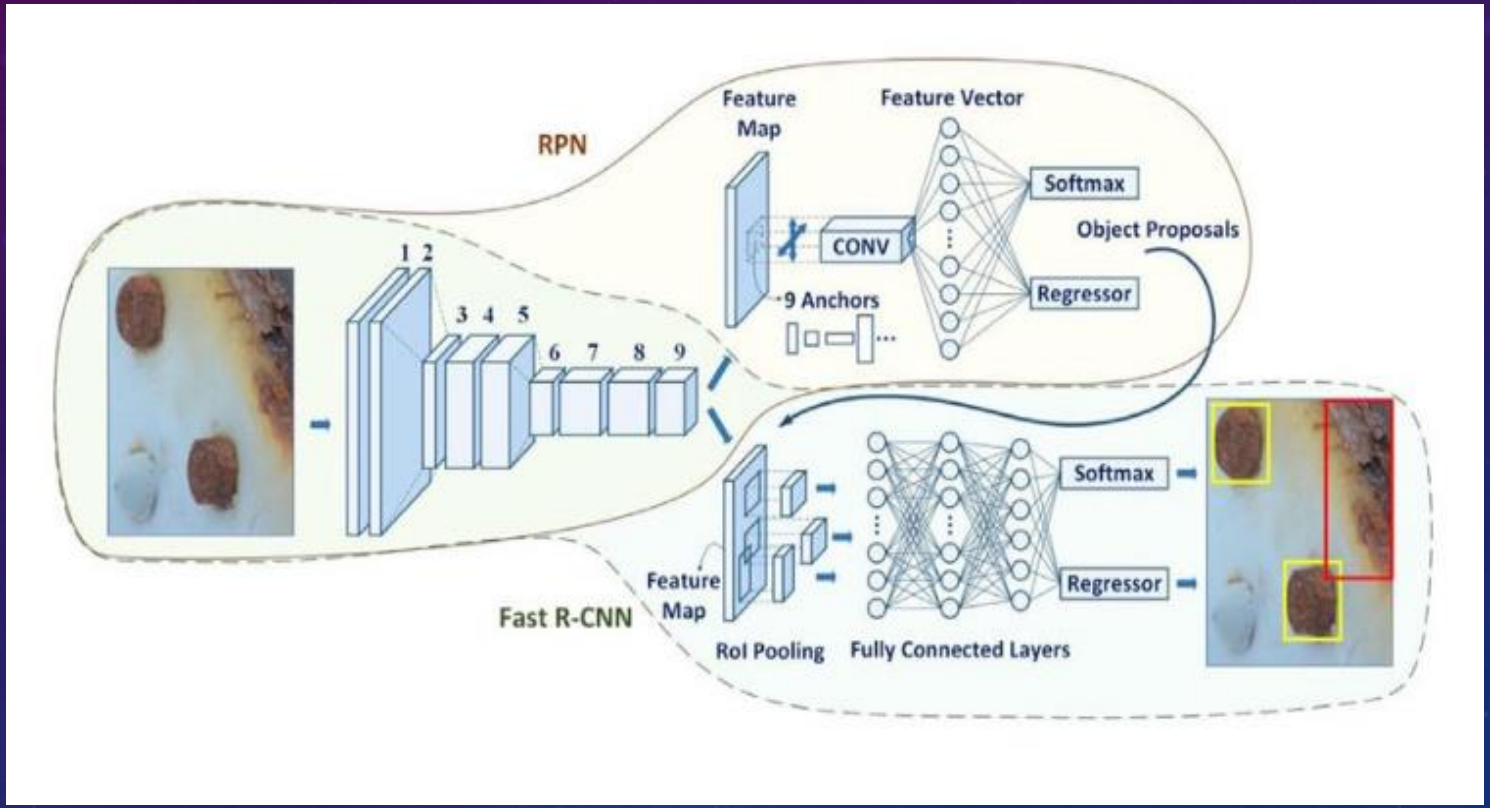
•Fast R-CNN improves on the R-CNN by only performing CNN forward computation on the image as a whole. It introduces an RoI pooling layer to extract features of the same shape from RoIs of different shapes.

•Faster R-CNN replaces the selective search used in Fast R-CNN with a region proposal network. This reduces the number of proposed regions generated, while ensuring precise object detection.

•Mask R-CNN uses the same basic structure as Faster R-CNN, but adds a fully convolution layer to help locate objects at the pixel level and further improve the precision of object detection.

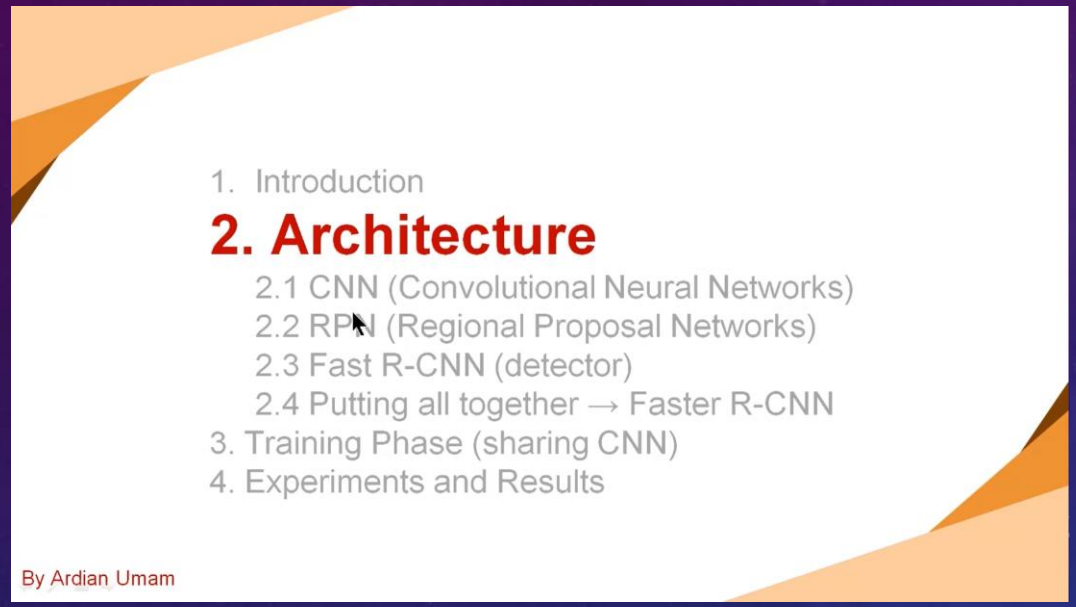
Appendix

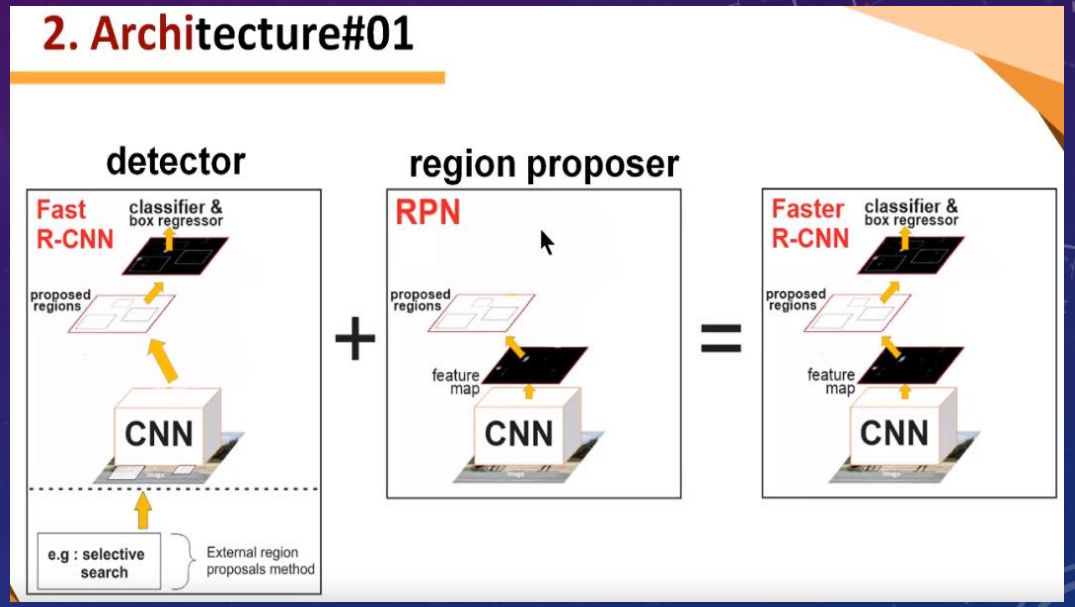
Introduction to How Faster R-CNN, Fast R-CNN and R-CNN Works



https://www.youtube.com/watch?v=v5bFVbQvFRk&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=1 [8:40]

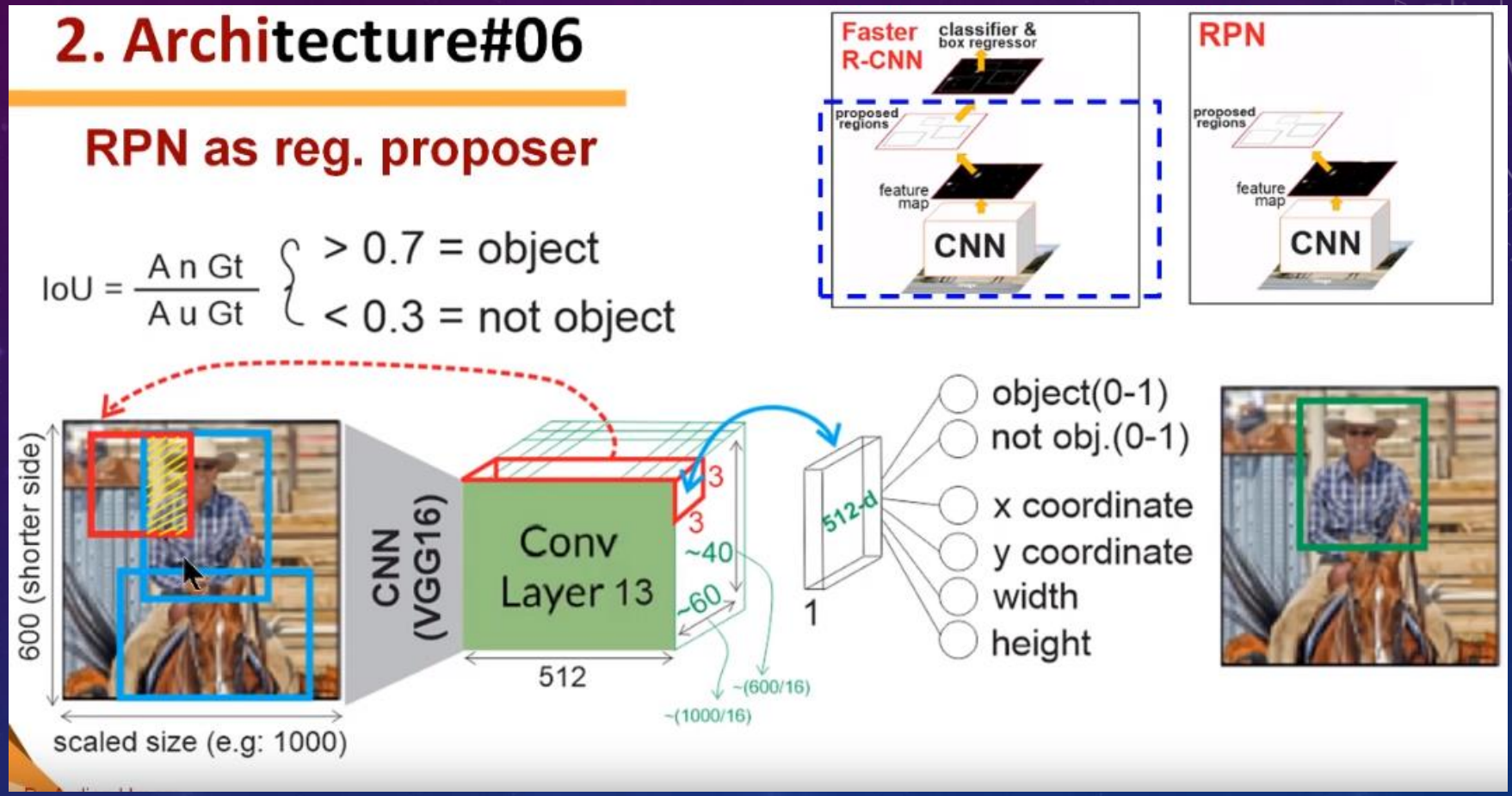
Faster R-CNN architecture





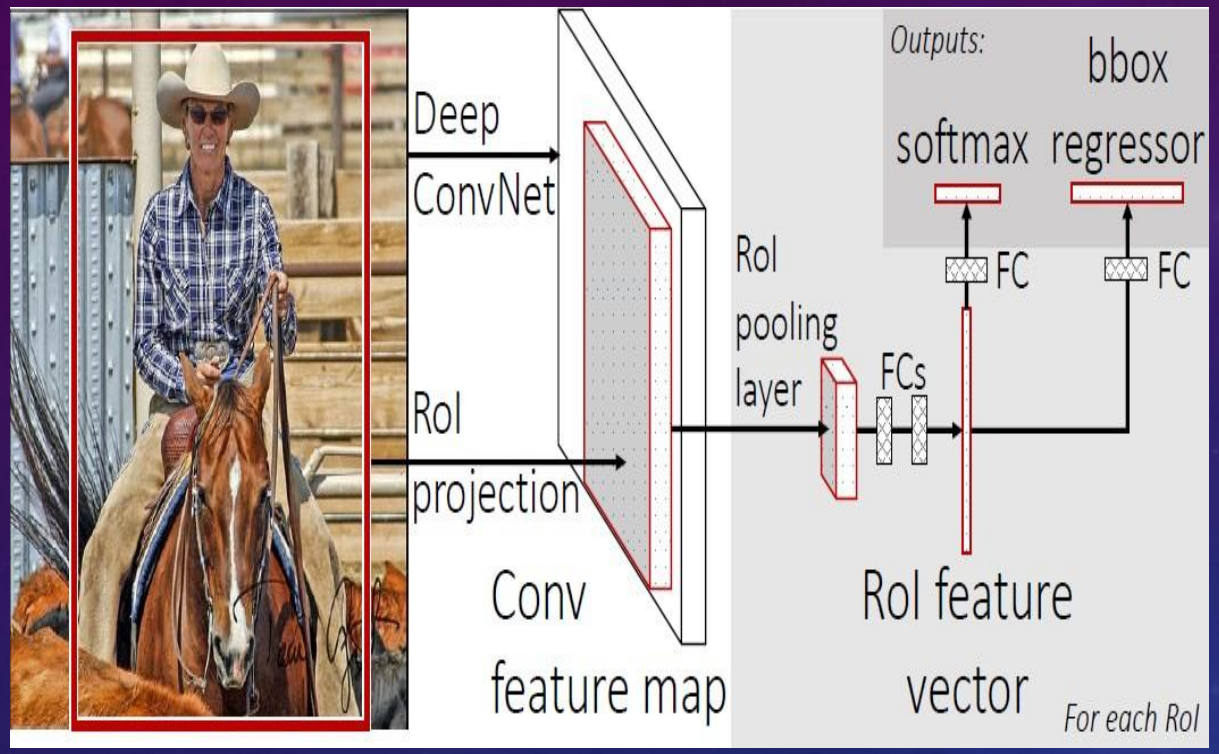
https://www.youtube.com/watch?v=c1_g6tw69bU&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=2 [5:07]

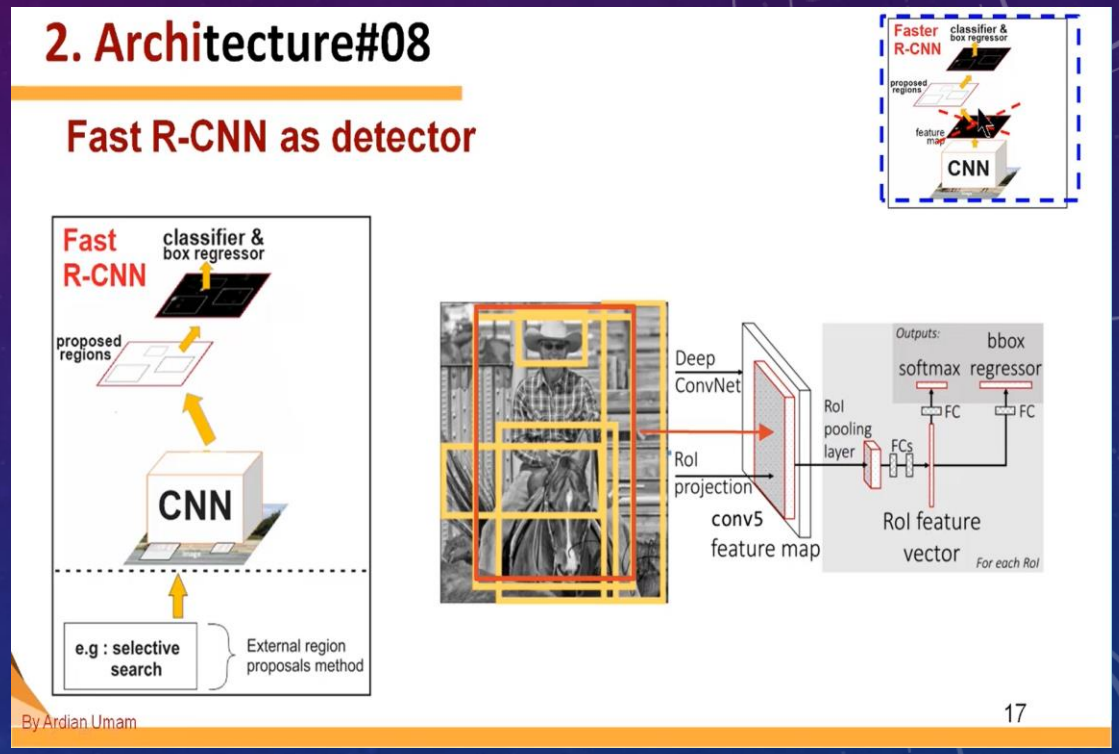
How RPN (Region Proposal Network) works



https://www.youtube.com/watch?v=X3IlbjQs190&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=3 [18:10]

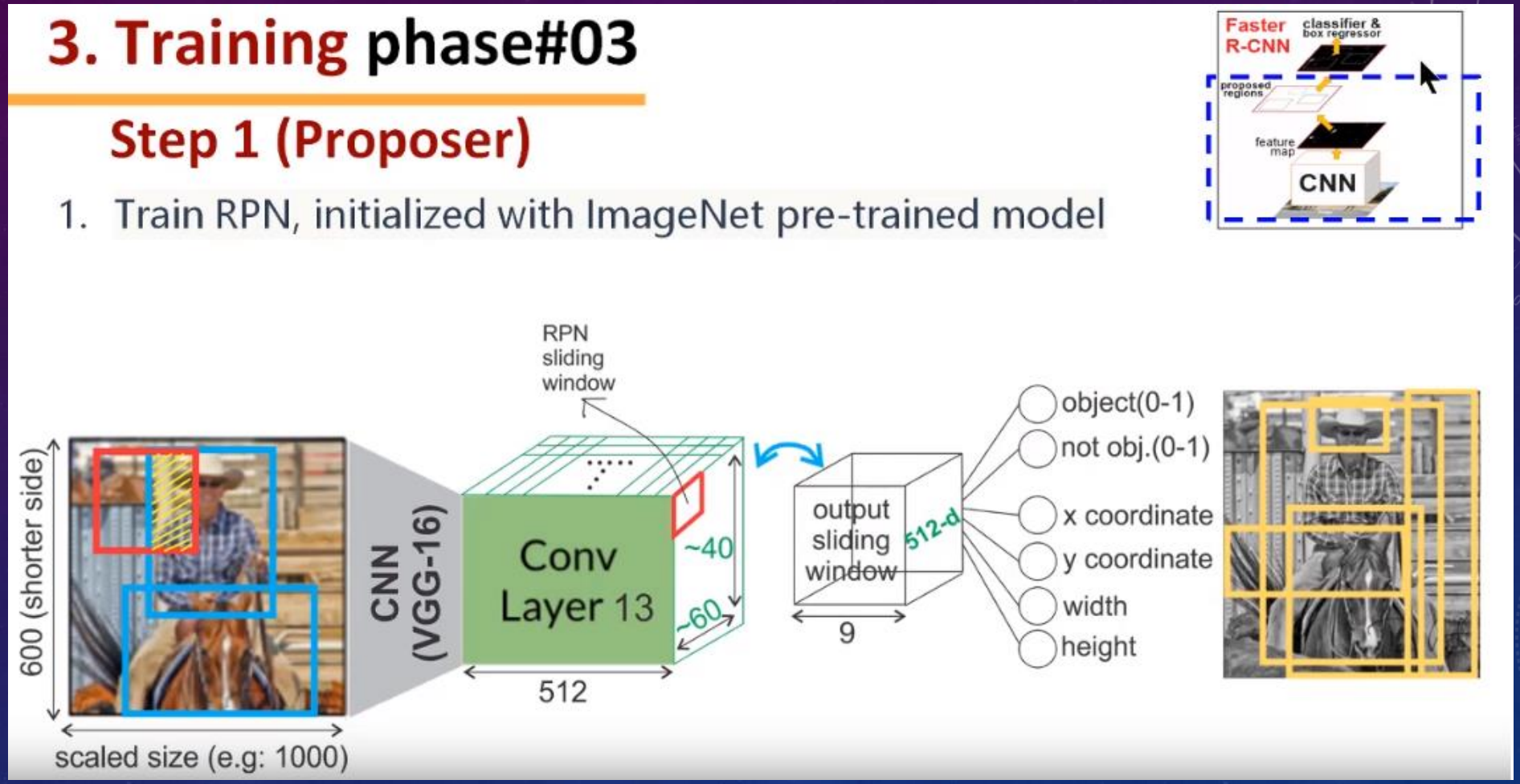
How Fast R-CNN Works





https://www.youtube.com/watch?v=xzw3lcdllOU&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=4 [2:51]

How To Train Faster R-CNN



https://www.youtube.com/watch?v=cSO1nUj495Y&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=5 [13:55]

Experiments and Results of Faster R-CNN

Experiments & Results#1

Experiments: Datasets

Visual Object Classes Challenge 2012 (VOC2012)

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

VOC 2007

5k trainval images
5k test images
20 object categories

80k training images
40k validation set images
20k test images
80 object categories

Microsoft COCO
Common Objects in Context

By Ardian Umam

29

https://www.youtube.com/watch?v=306ieamtvGM&list=PLkRkKTC6HZMzp28TxR_fJYZ-K8Yu3EQw0&index=6 [20:51]