Spring 2024

CMPE 258-01 Deep Learning

Dr. Kaikai Liu, Ph.D. Associate Professor

Department of Computer Engineering

San Jose State University

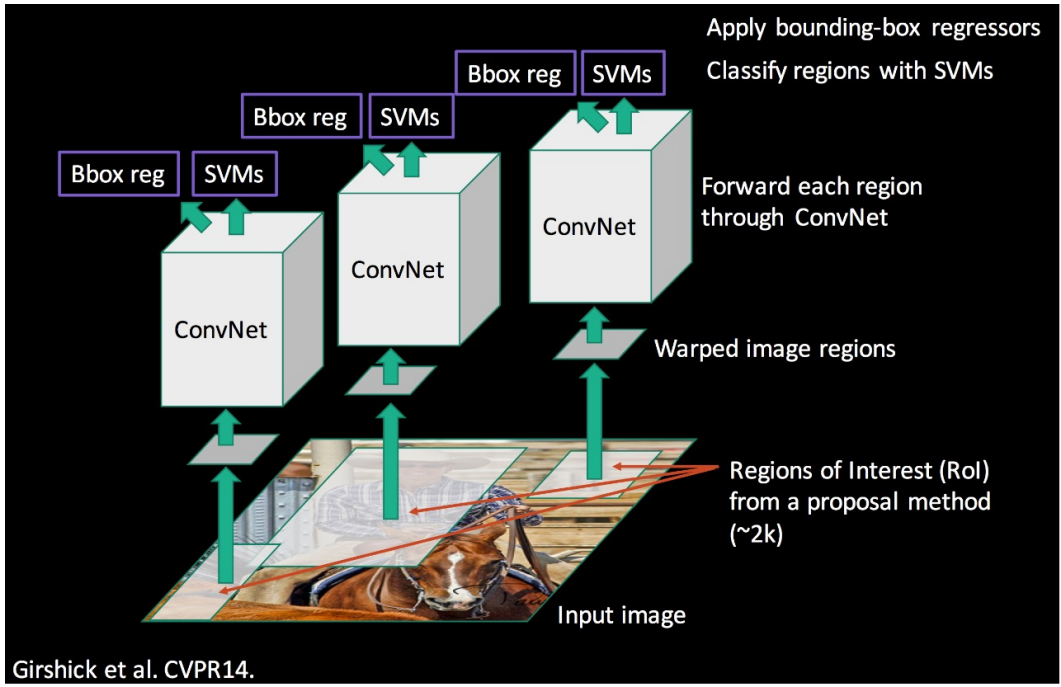
Email: kaikai.liu@sjsu.edu

Website: https://www.sjsu.edu/cmpe/faculty/tenure-line/kaikai-liu.php

Deep Learning based Object Detection

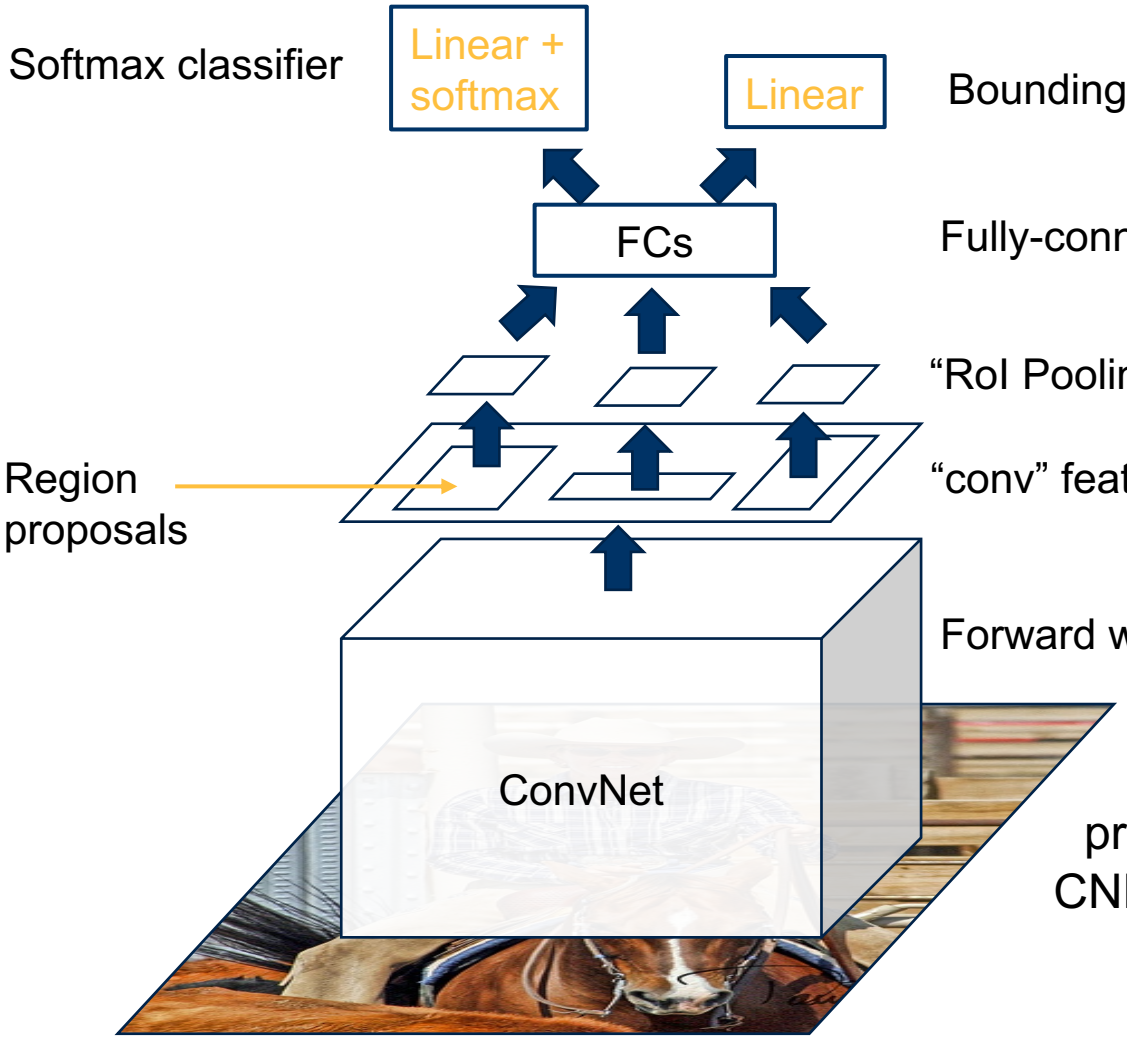
R-CNN pros and cons

- Pros
  - Accurate!
  - Any deep architecture can immediately be “plugged in”
- Cons
  - Ad hoc training objectives
    - Fine-tune network with softmax classifier
    - Train post-hoc linear SVMs
    - Train post-hoc bounding-box regressions (least squares)
  - Training is slow (84h), takes a lot of disk space
    - 2000 CNN passes per image
  - Inference (detection) is slow (47s / image with VGG16)



Number of CNN runs = number of proposals → high latency

Fast R-CNN

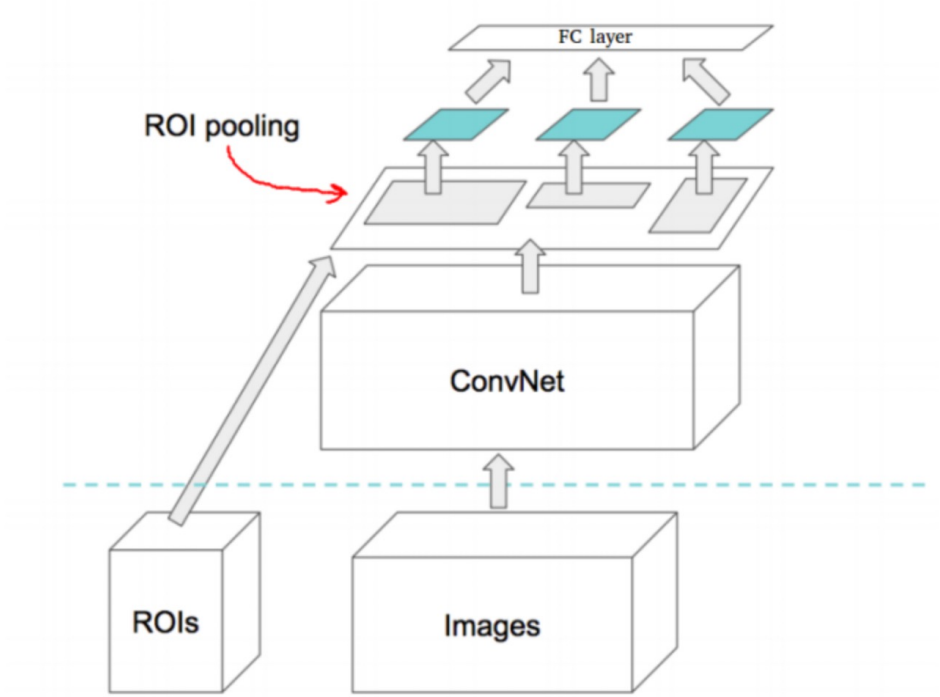


1 CNN run for whole image
proposed regions extracted from
CNN feature map, instead of image

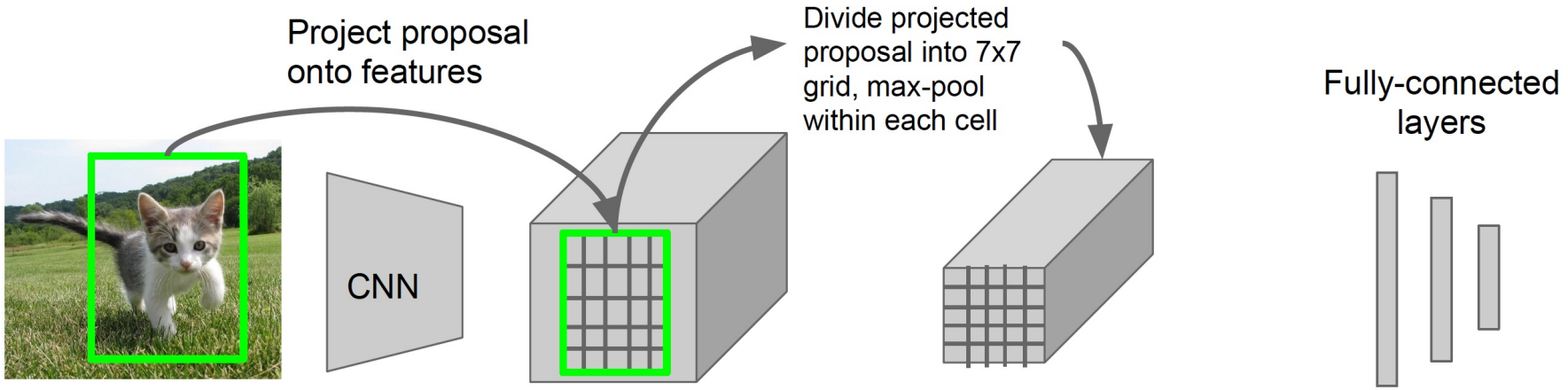
Source: R. Girshick

R. Girshick, Fast R-CNN, ICCV 2015

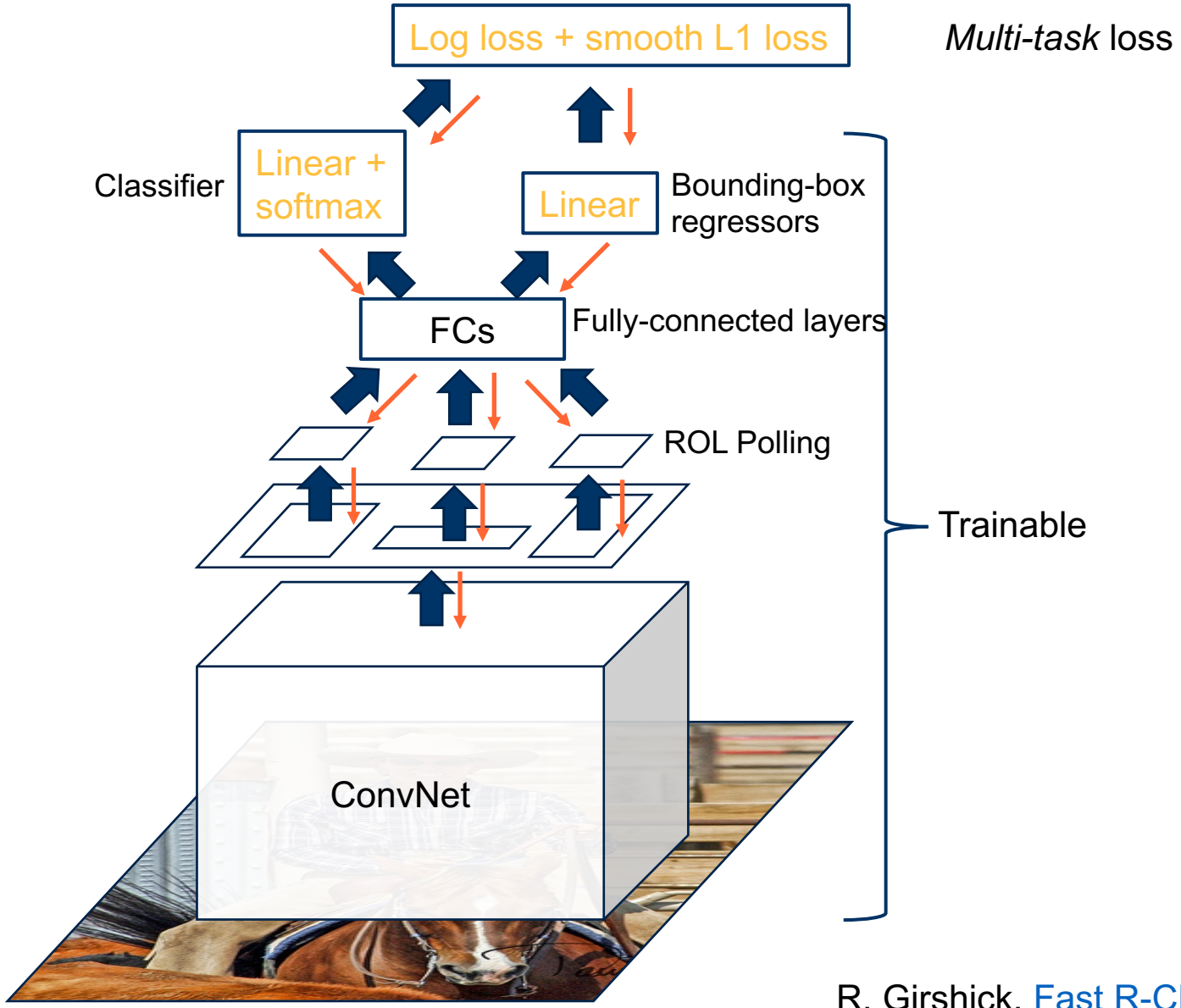
New operation: Region of Interest (ROI) pooling



Necessary to convert variable sized ROIs to fixed size inputs
expected by the following FC layers
(similar to warping original image patches in R-CNN
for fixed sized inputs to conv layers)



Fast R-CNN training



R. Girshick, Fast R-CNN, ICCV 2015

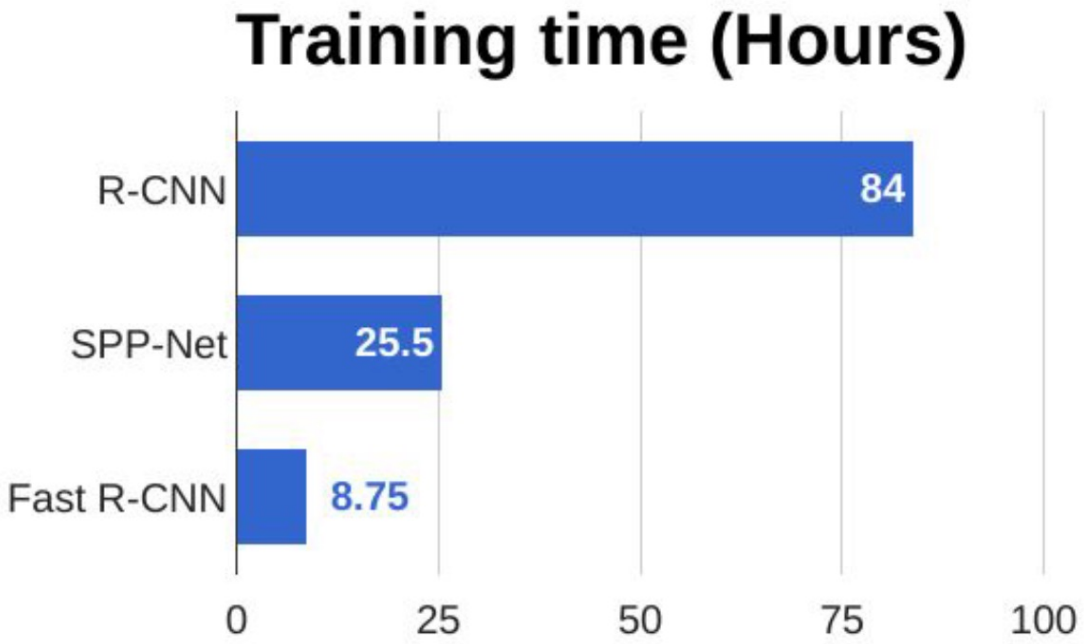
Fast R-CNN results

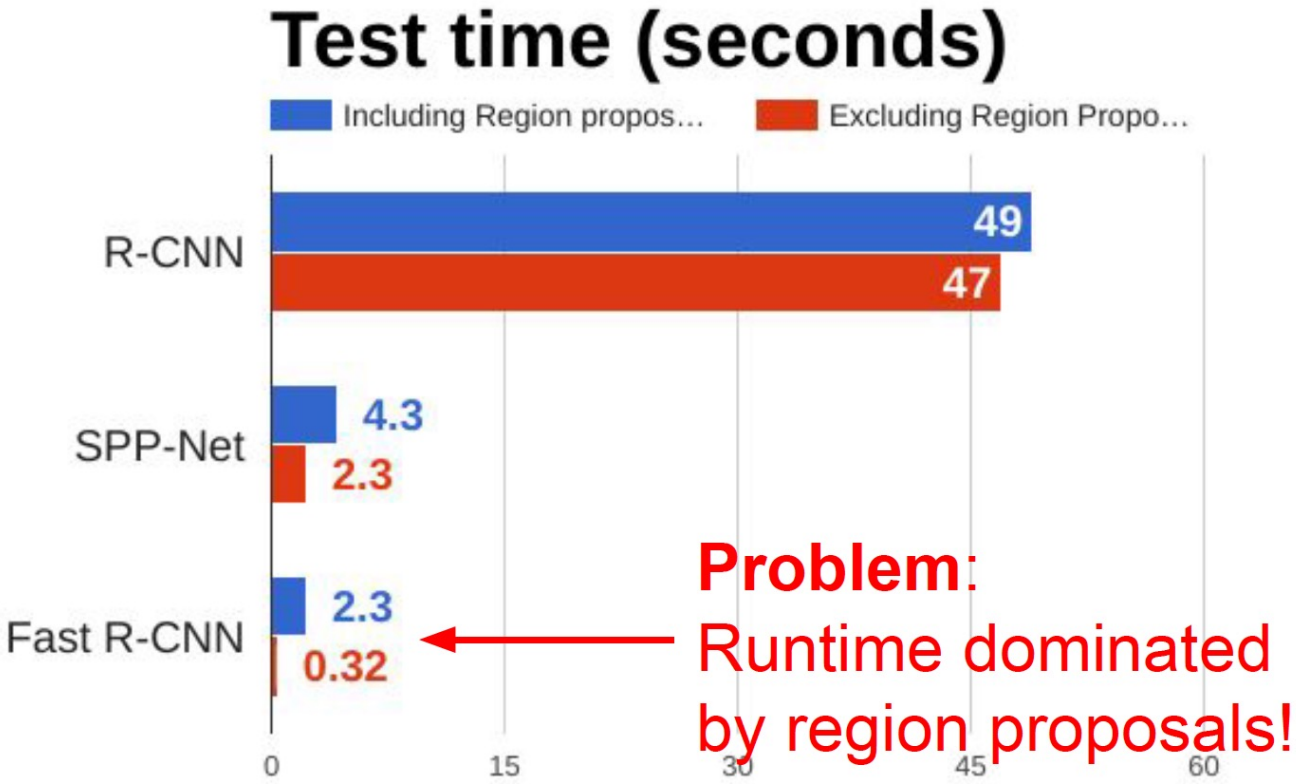
	Fast R-CNN	R-CNN
Train time (h)	**9.5**	84
- Speedup	**8.8x**	1x
Test time / image	**0.32s**	47.0s
Test speedup	**146x**	1x
mAP	**66.9%**	66.0%

Timings exclude object proposal time, which is equal for all methods.
All methods use VGG16 from Simonyan and Zisserman.

Source: R. Girshick

R-CNN vs SPP vs Fast R-CNN

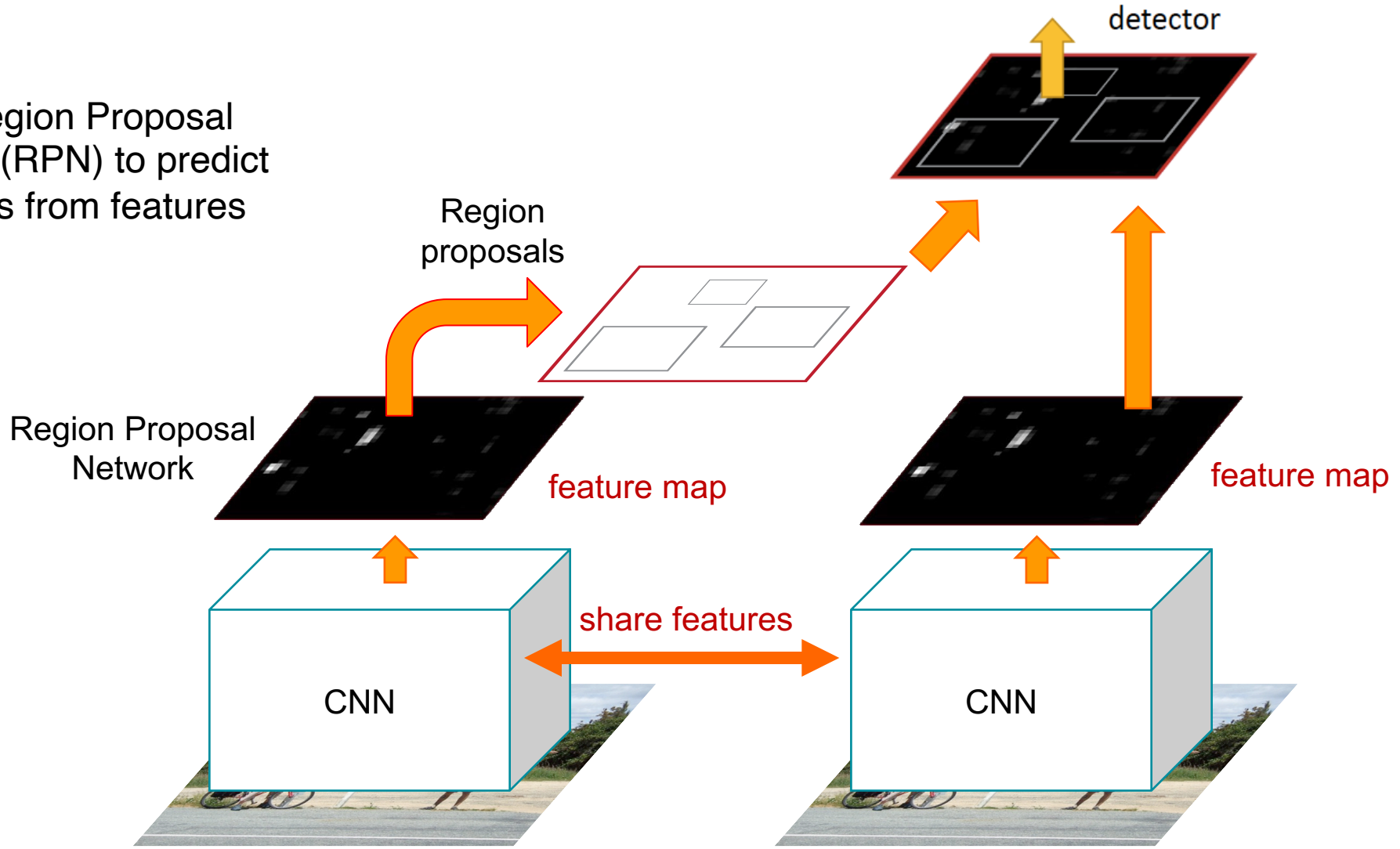




Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

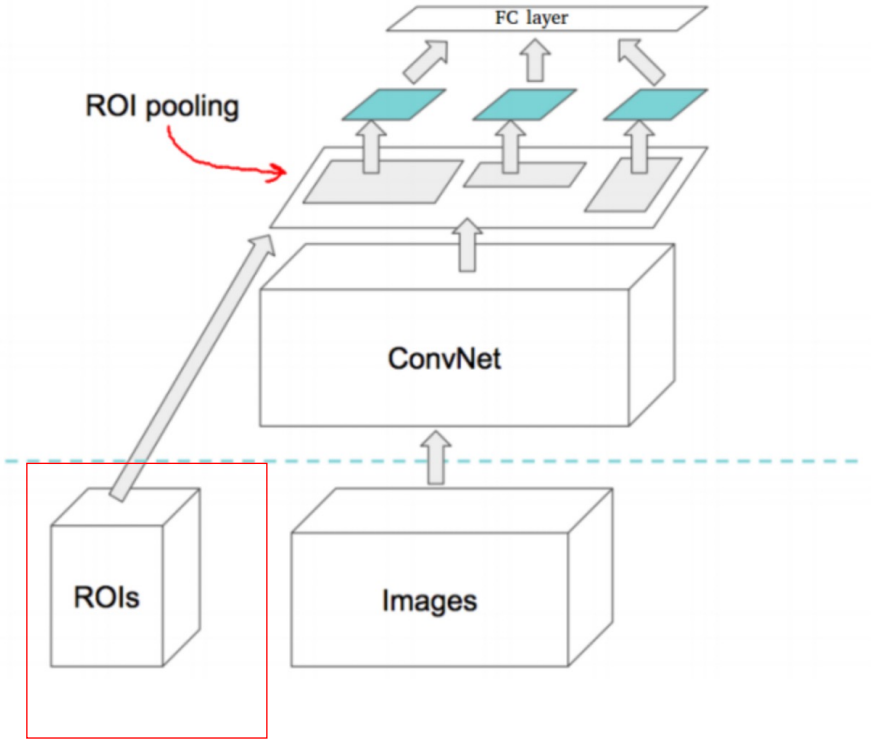
Faster R-CNN

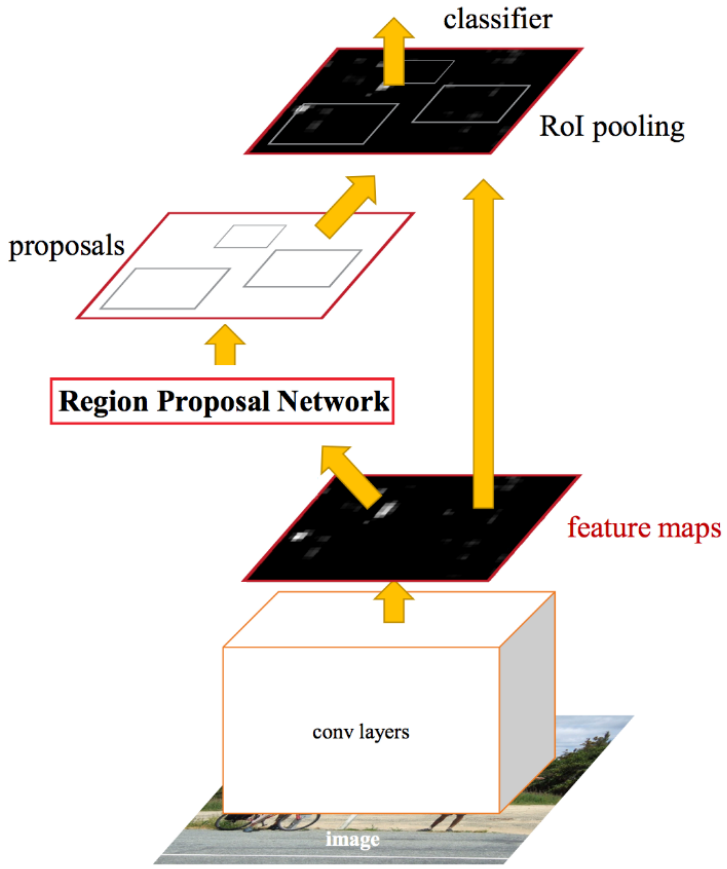
Insert Region Proposal Network (RPN) to predict proposals from features



S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS 2015

Region Proposal Network (RPN)

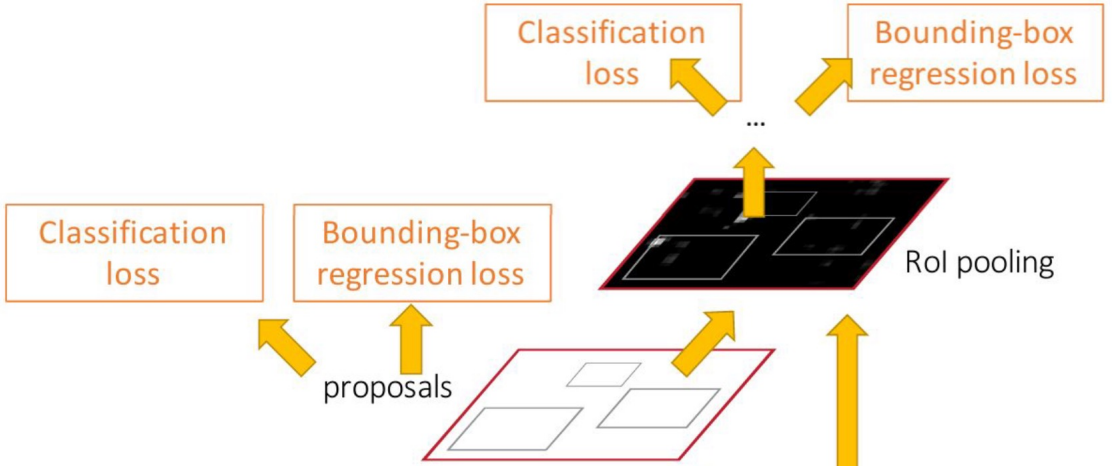


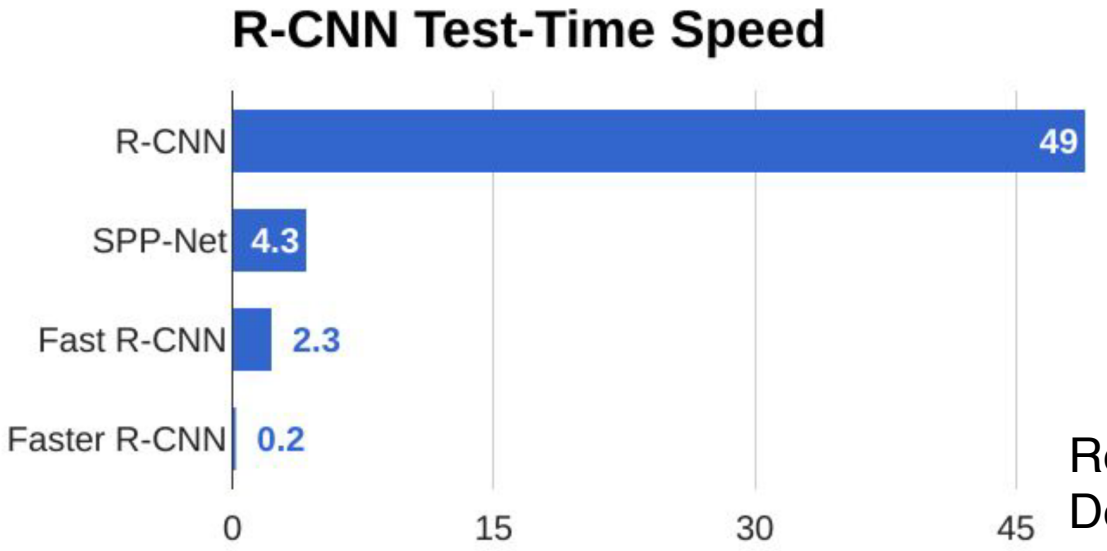


Instead of external region proposals in Fast R-CNN, Faster R-CNN generates proposals **re-using** the same initial convolutional feature map.

Region Proposal Network (RPN)

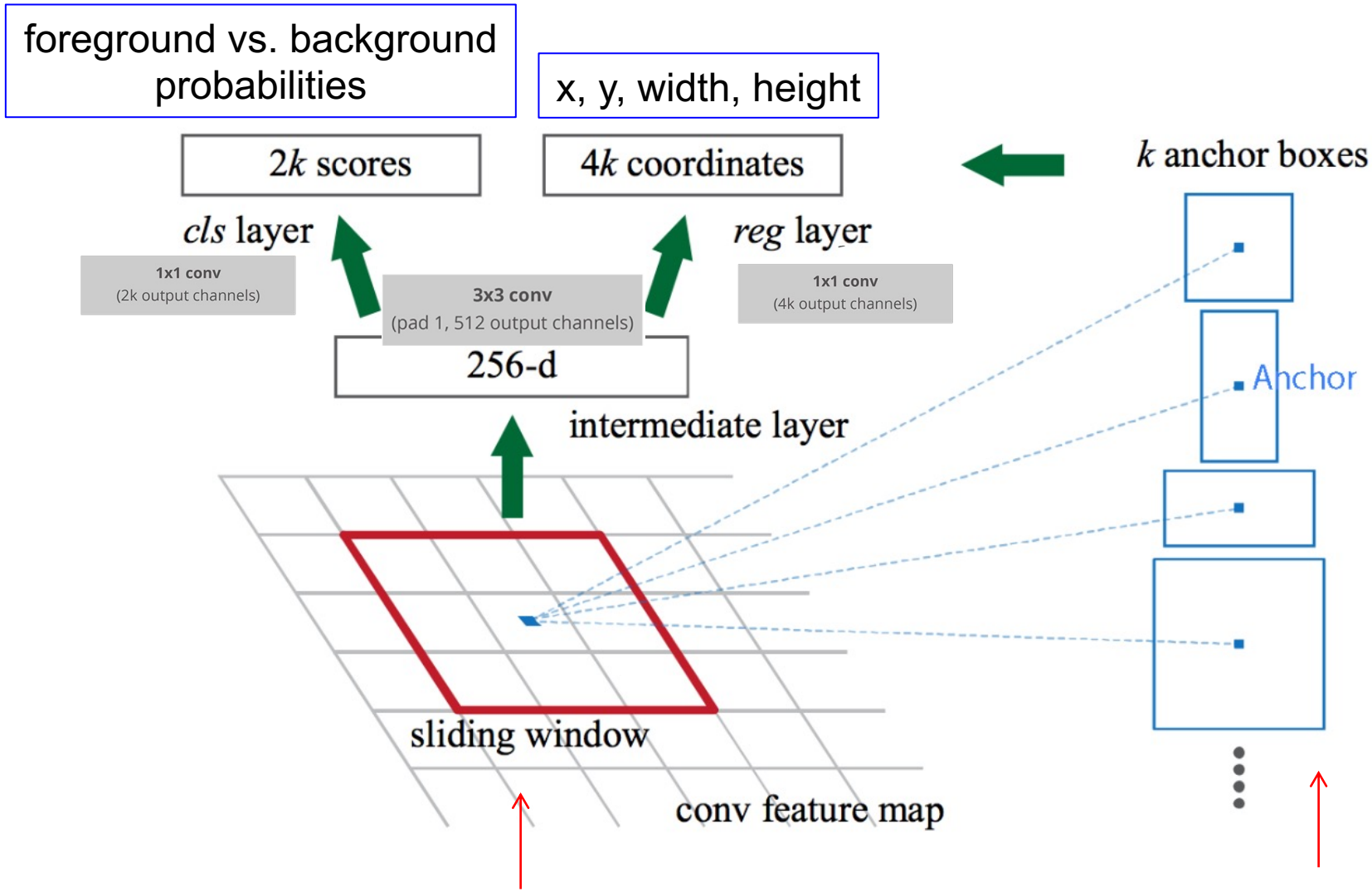
- Make CNN do proposals!
- Jointly train with 4 losses:
  - RPN classify object / not object
  - RPN regress box coordinates
  - Final classification score (object classes)
  - Final box coordinates





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

Region Proposal Network (RPN)

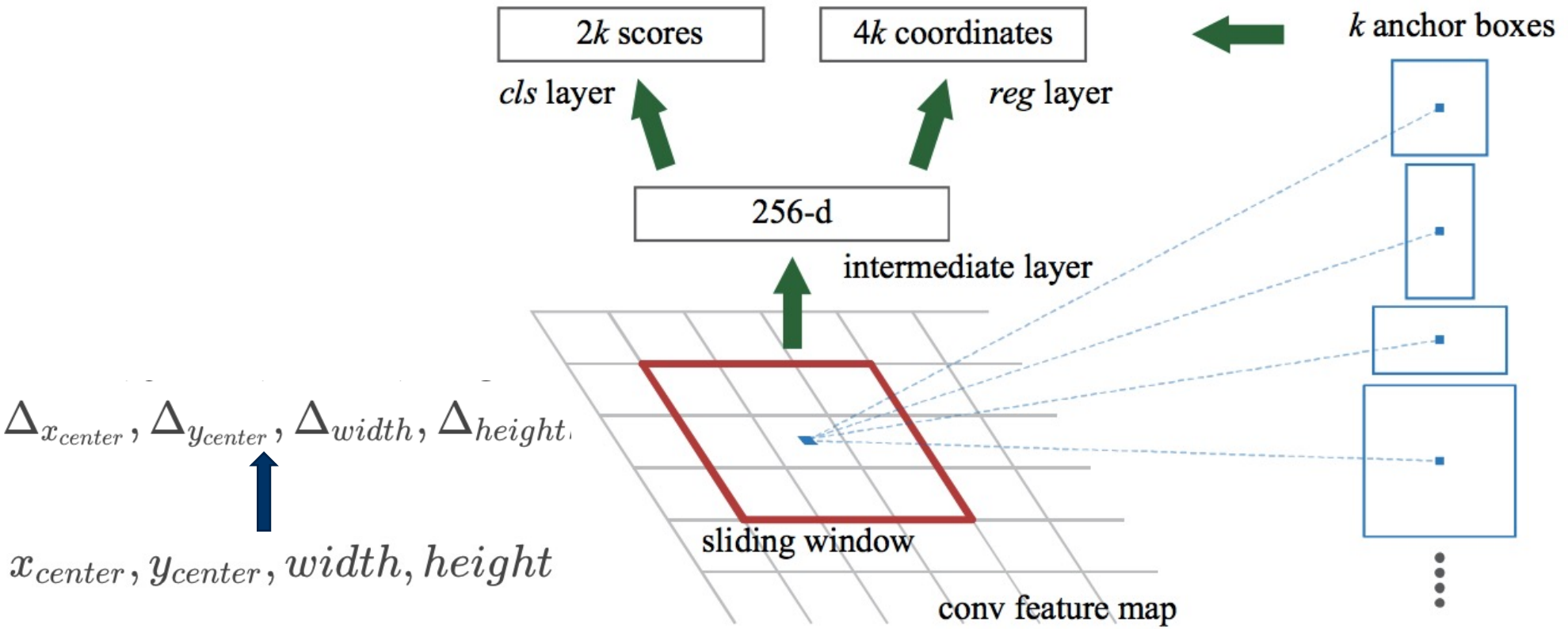


decides the location of the region proposal

decides the shape/size of the region proposal
(adjust if we are only looking for faces (square),
or cars (wide-short) or pedestrians (narrow-tall))

Region proposal network (RPN)

- Slide a small window over the conv layer
  - Predict object/no object
  - Regress bounding box coordinates
  - Box regression is with reference to *anchors* (3 scales x 3 aspect ratios)



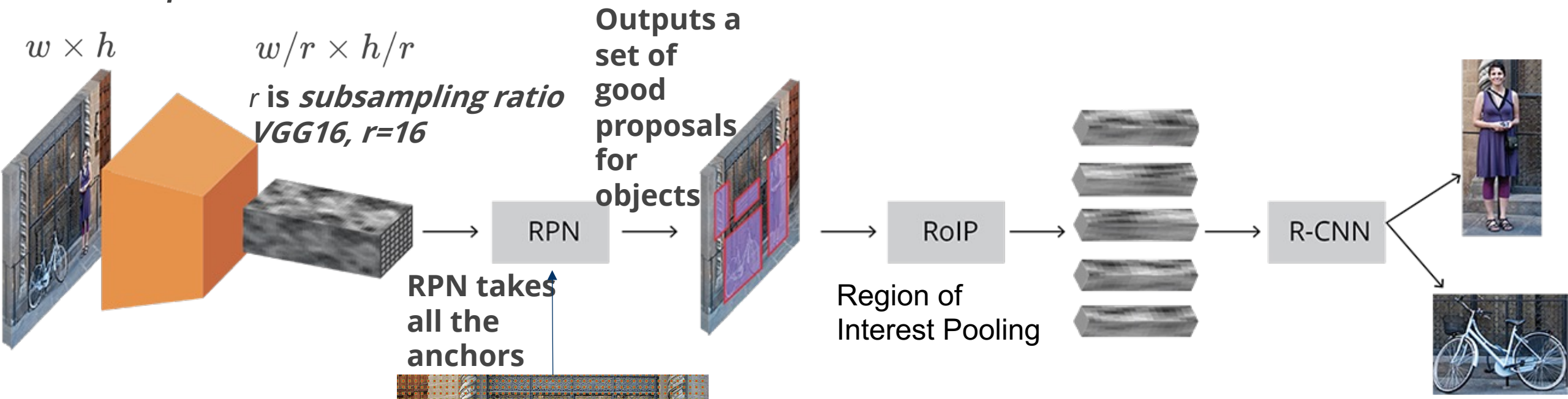
In order to choose the set of anchors we usually define a set of sizes (e.g. 64px, 128px, 256px) and a set of ratios between width and height of boxes (e.g. 0.5, 1, 1.5) and use all the possible combinations of sizes and ratios.

Faster R-CNN

- When modeling deep neural networks, the last block is usually a fixed sized tensor output. For example, in image classification, the output is a (*N*,) shaped tensor, with *N* being the number of classes, where each scalar in location *i* contains the probability of that image being *labeli*.
- One issue with object detection is generating a variable-length list of bounding boxes
- The variable-length problem is solved in the RPN by using anchors: fixed sized reference bounding boxes which are placed uniformly throughout the original image. Instead of having to detect where objects are, we model the problem into two parts. For every anchor, we ask:
  - Does this anchor contain a relevant object?
  - How would we adjust this anchor to better fit the relevant object?

Faster R-CNN

- *Complete Faster R-CNN architecture*



Anchor points (x_c, y_c)
Scales and ratios for (h, w)
N candidates per anchor

Faster R-CNN

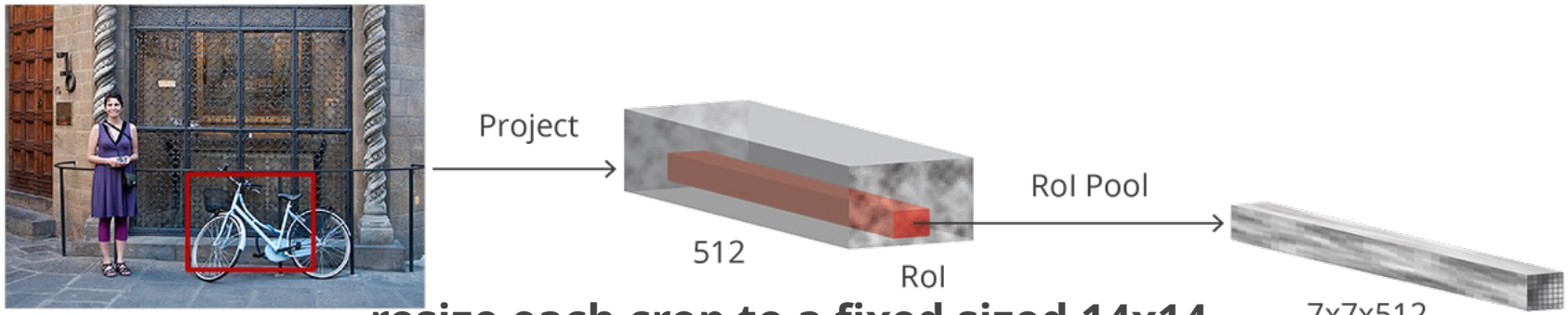
- RPN Training
  - For training, we take all the anchors and put them into two different categories. Those that overlap a **ground-truth object with an Intersection over Union (IoU)** bigger than 0.5 are considered “foreground” and those that don’t overlap any ground truth object or have less than 0.1 IoU with ground-truth objects are considered “background”.
  - Then, we randomly sample those anchors to **form a mini batch** of size 256 — trying to maintain a balanced ratio between foreground and background anchors.
  - The RPN uses all the anchors selected for the mini batch to calculate the classification loss using binary cross entropy.
  - Then, it uses only those minibatch anchors marked as **foreground** to calculate the regression loss. For calculating the targets for the regression, we use the foreground anchor and the closest ground truth object and calculate the correct Δ needed to transform the anchor into the object.
    - Instead of using a simple L1 or L2 loss for the regression error, the paper suggests using Smooth L1 loss. Smooth L1 is basically L1, but when the L1 error is small enough, defined by a certain σ, the error is considered almost correct and the loss diminishes at a faster rate.

Faster R-CNN

- Post Processing
  - Non-maximum suppression (NMS)
    - Since anchors usually overlap, proposals end up also overlapping over the same object.
    - NMS takes the list of proposals sorted by score and iterates over the sorted list, discarding those proposals that have an IoU larger than some predefined threshold with a proposal that has a higher score.
    - Proposal selection after applying NMS, we keep the top N proposals sorted by score. In the paper N=2000 is used, but it is possible to lower that number to as little as 50 and still get quite good results
  - The RPN can be used by itself without needing the second stage model. In problems where there is only a single class of objects, the objectness probability can be used as the final class probability. This is because for this case, “foreground” = “single class” and “background” = “not single class”.
  - One of the advantages of using only the RPN is the gain in speed both in training and prediction. Since the RPN is a very simple network which only uses convolutional layers, the prediction time can be faster than using the classification base network.

Faster R-CNN

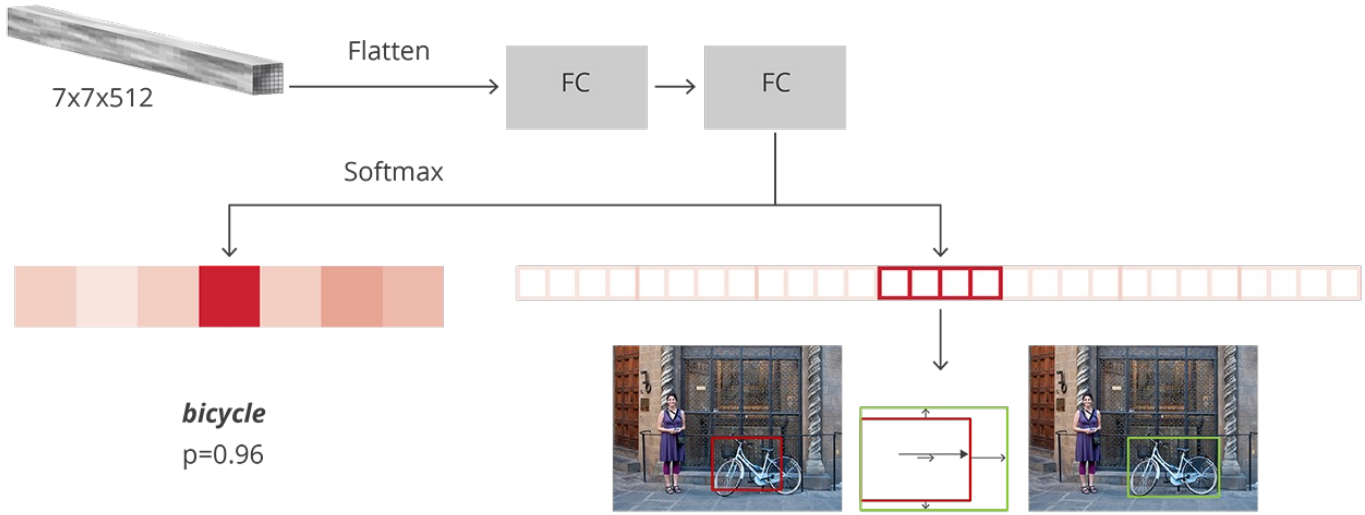
- Region of Interest Pooling
  - After the RPN step, we have a bunch of object proposals with no class assigned to them. Our next problem to solve is how to take these bounding boxes and classify them into our desired categories.
  - The main **problem** is that running the computations for all the 2000 proposals is really inefficient and slow.
    - Reusing the existing convolutional feature map. This is done by extracting fixed-sized feature maps for each proposal using region of interest pooling. Fixed size feature maps are needed for the R-CNN in order to classify them into a fixed number of classes.



resize each crop to a fixed sized 14x14, then max pooling with a 2x2 kernel

Faster R-CNN

- Then, it uses two different fully-connected layers for each of the different objects.
  - A fully-connected layer with N+1 units where N is the total number of classes and that extra one is for the background class.
  - A fully-connected layer with $4N$ units. We want to have a regression prediction, thus we need $\Delta centerx$,$\Delta centery$,$\Delta width$,$\Delta height$ for each of the N possible classes.



Faster R-CNN

- Training and targets for R-CNN
  - We take the proposals and the ground-truth boxes, and calculate the IoU
    - Those proposals that have a IoU greater than 0.5 with any ground truth box get assigned to that ground truth.
    - Those that have between 0.1 and 0.5 get labeled as background.
    - Different from RPN, we ignore proposals without any intersection. This is because at this stage we are assuming that we have good proposals
  - The targets for the bounding box regression are calculated as the offset between the proposal and its corresponding ground-truth box, only for those proposals that have been assigned a class based on the IoU threshold.
    - We randomly sample a balanced mini batch of size 64 in which we have up to 25% foreground proposals (with class) and 75% background.
    - The classification loss is now a multiclass cross entropy loss, using all the selected proposals and the Smooth L1 loss for the 25% proposals that are matched to a ground truth box.

Faster R-CNN

- Post processing
  - After getting the final objects and ignoring those predicted as background, we apply class-based NMS. This is done by grouping the objects by class, sorting them by probability and then applying NMS to each independent group before joining them again.
- Training
  - In the original paper, Faster R-CNN was trained using a multi-step approach, training parts independently and merging the trained weights before a final full training approach. Since then, it has been found that doing end-to-end, joint training leads to better results.
  - After putting the complete model together we end up with 4 different losses, two for the RPN and two for R-CNN. We have the trainable layers in RPN and R-CNN, and we also have the base network which we can train (fine-tune) or not. The four different losses are combined using a weighted sum.

Thank You

Address:
ENG257, SJSU

Email Address:
Kaikai.liu@sjsu.edu