Reproducible research – get the code!

http://git.io/vBqm5

Fast R-CNN

Ross Girshick

Facebook AI Research (FAIR)

Work done at Microsoft Research

Fast Region-based ConvNets (R-CNNs) for Object Detection

Localization
Where?

Recognition
What?

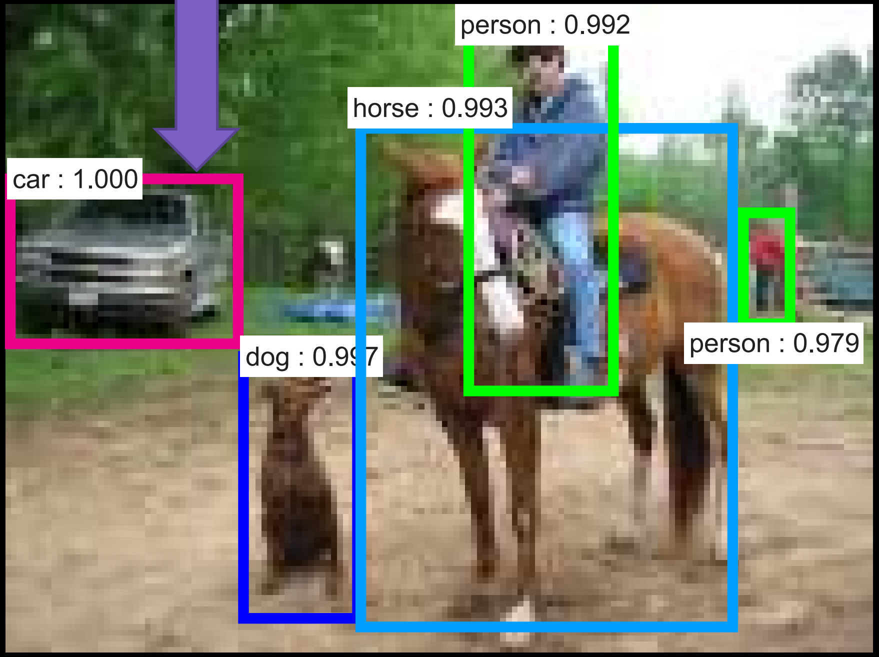
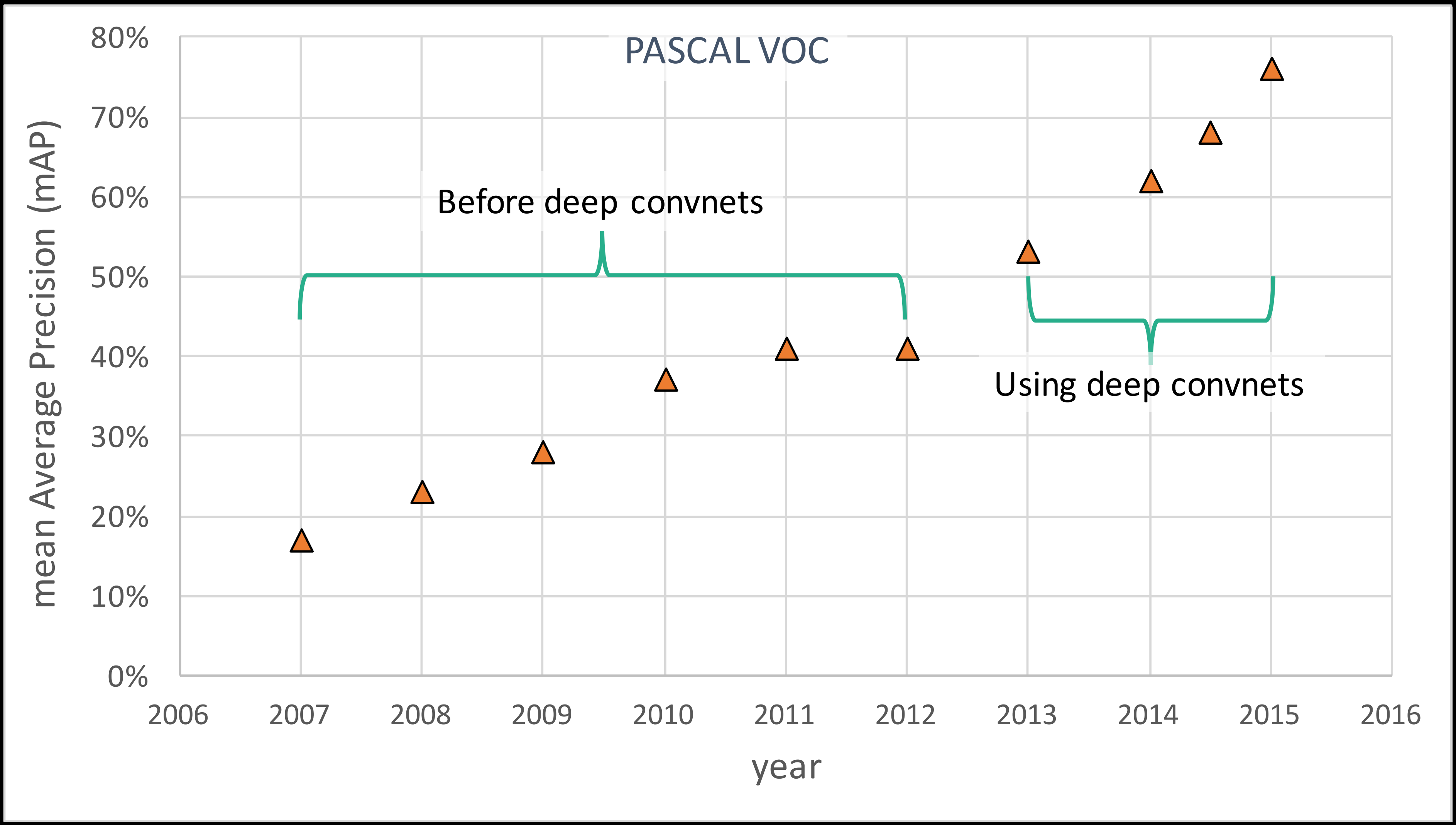
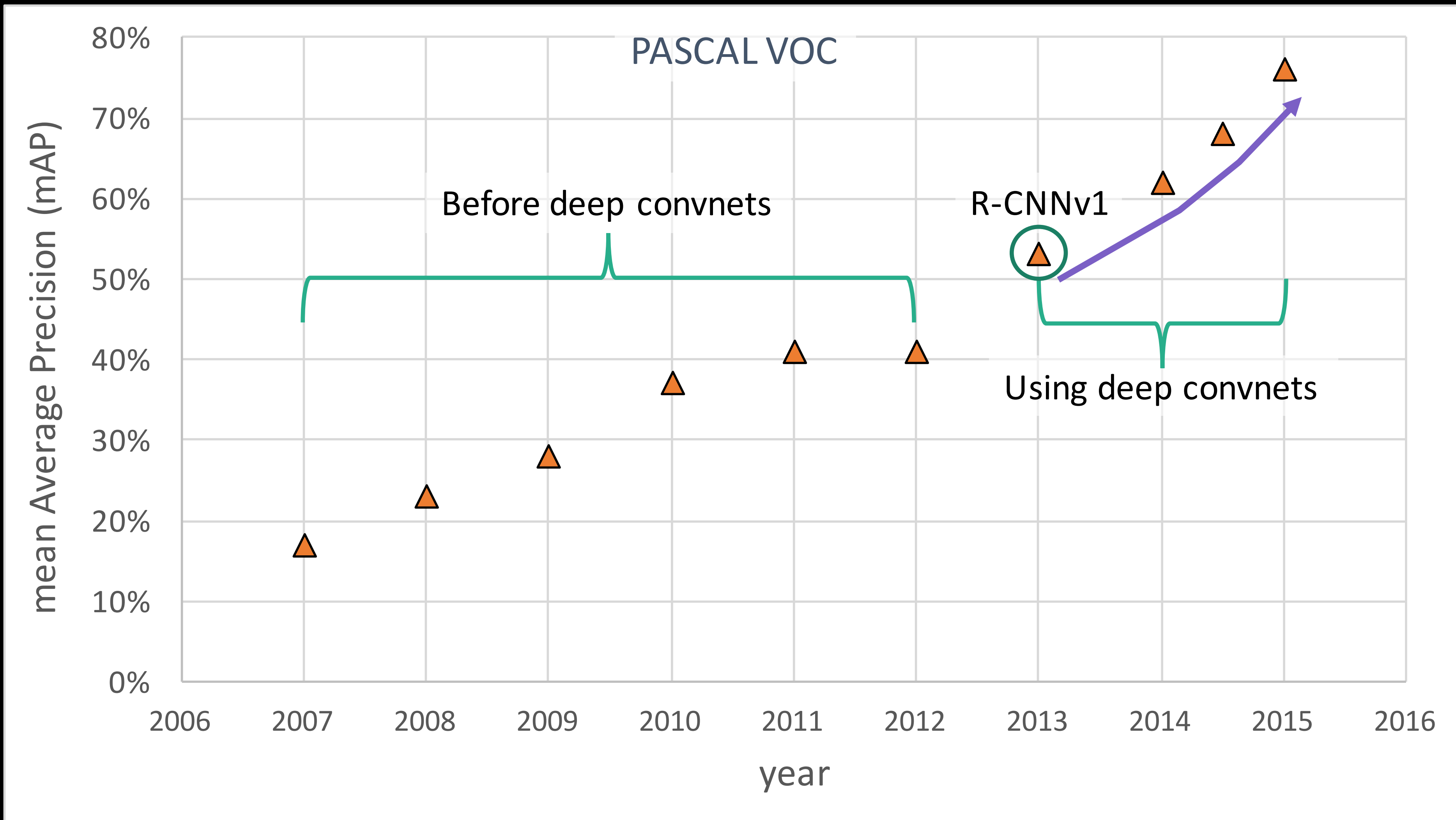


Figure adapted from Kaiming He

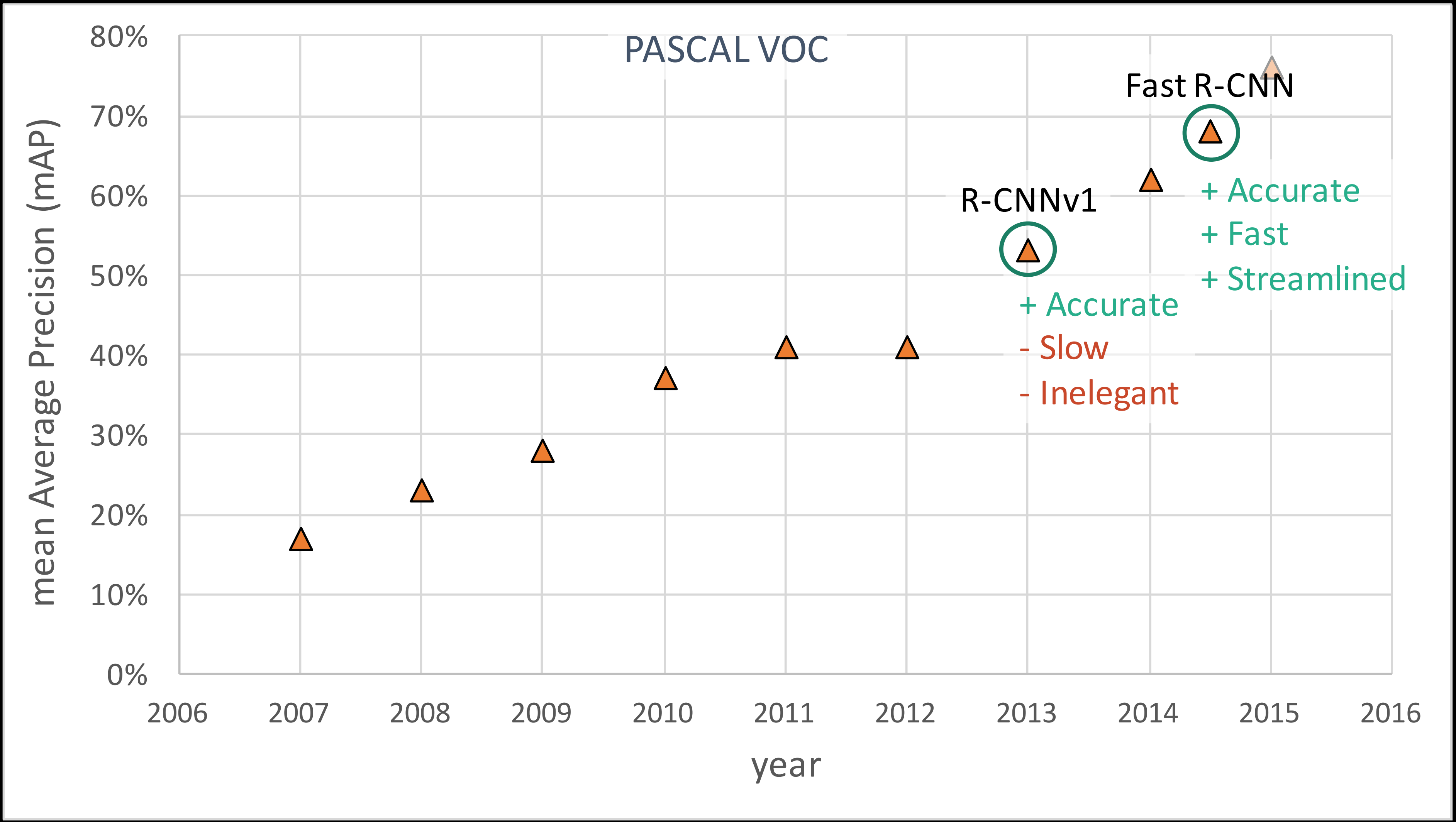
Object detection renaissance (2013-present)



Object detection renaissance (2013-present)



Object detection renaissance (2013-present)



Region-based convnets (R-CNNs)

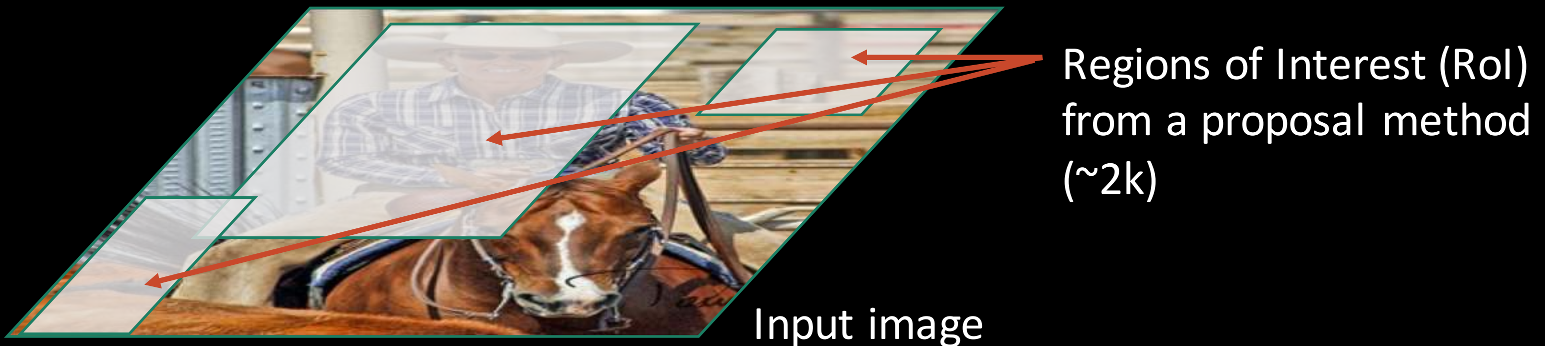
- R-CNN (aka “slow R-CNN”) [Girshick et al. CVPR14]
- SPP-net [He et al. ECCV14]

Slow R-CNN

Input image

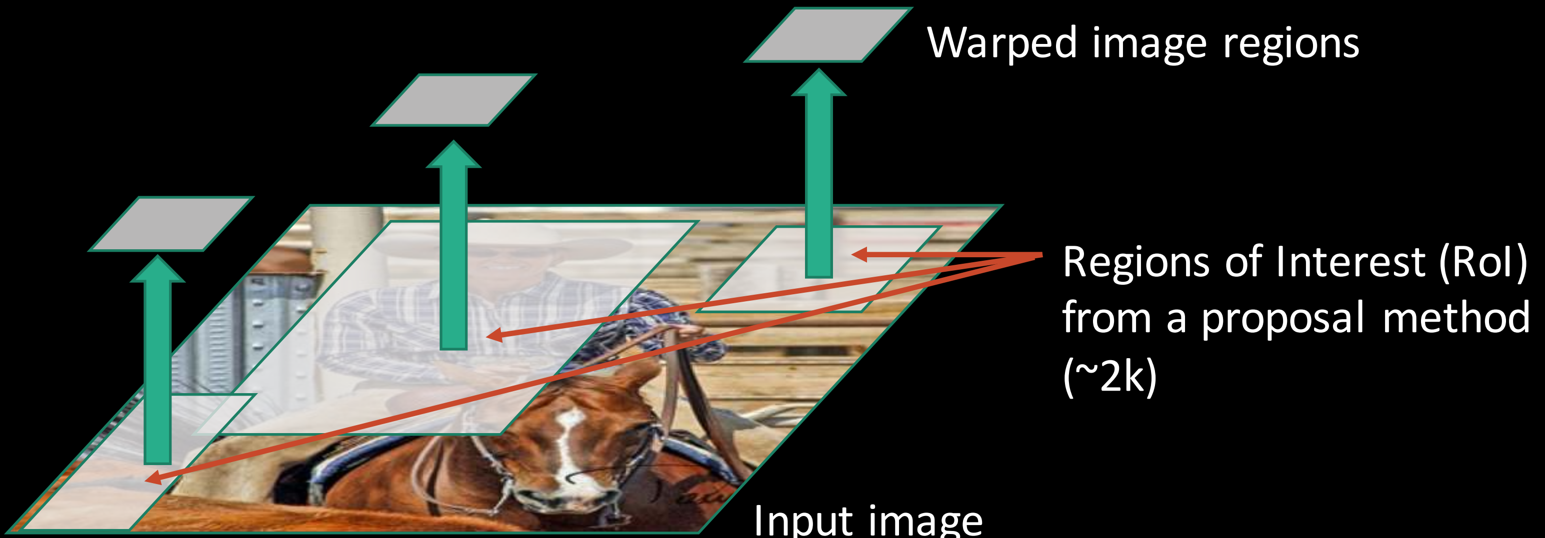
Girshick et al. CVPR14.

Slow R-CNN



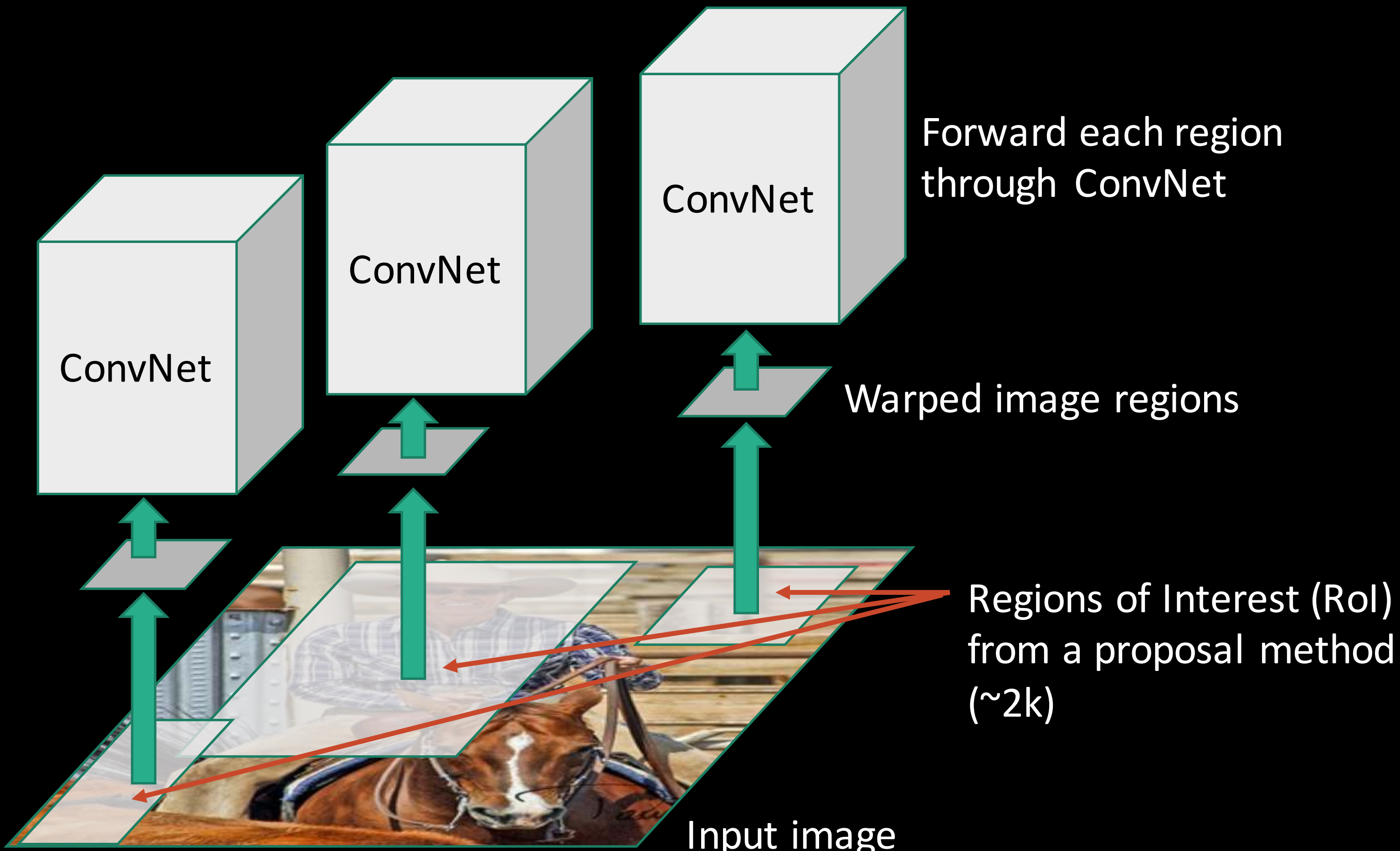
Girshick et al. CVPR14.

Slow R-CNN



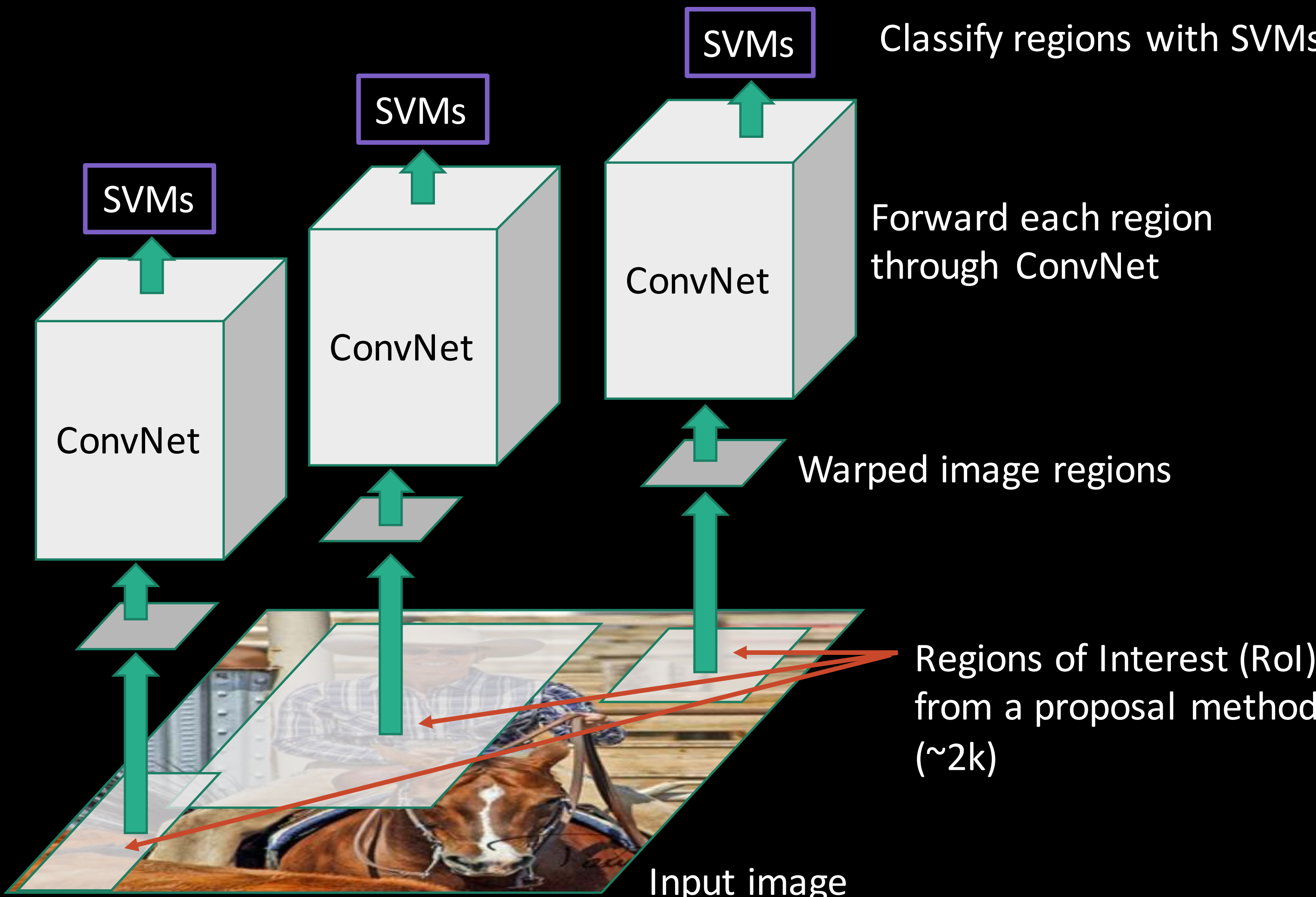
Girshick et al. CVPR14.

Slow R-CNN



Girshick et al. CVPR14.

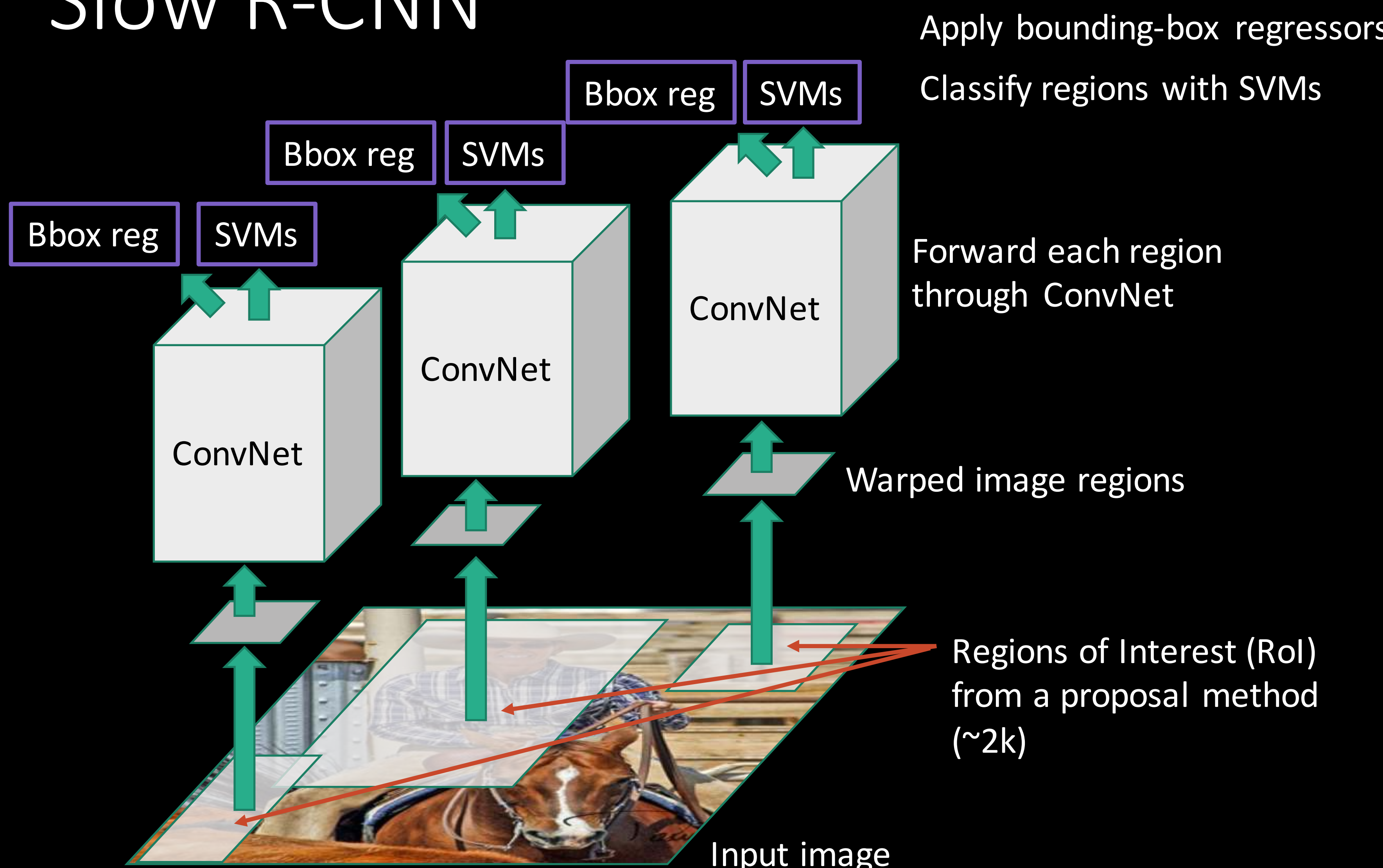
Slow R-CNN



Girshick et al. CVPR14.

Post hoc component

Slow R-CNN



Girshick et al. CVPR14.

Post hoc component

What's wrong with slow R-CNN?

What’s wrong with slow R-CNN?

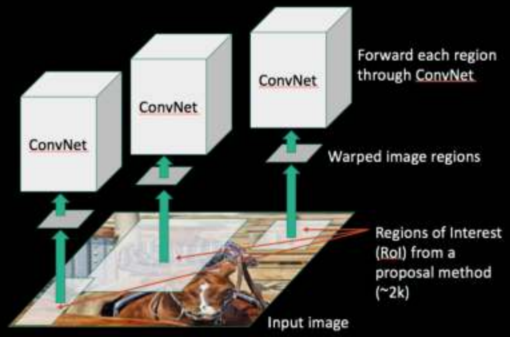
- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressors (squared loss)

What's wrong with slow R-CNN?

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressors (squared loss)
- Training is slow (84h), takes a lot of disk space

What's wrong with slow R-CNN?

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]



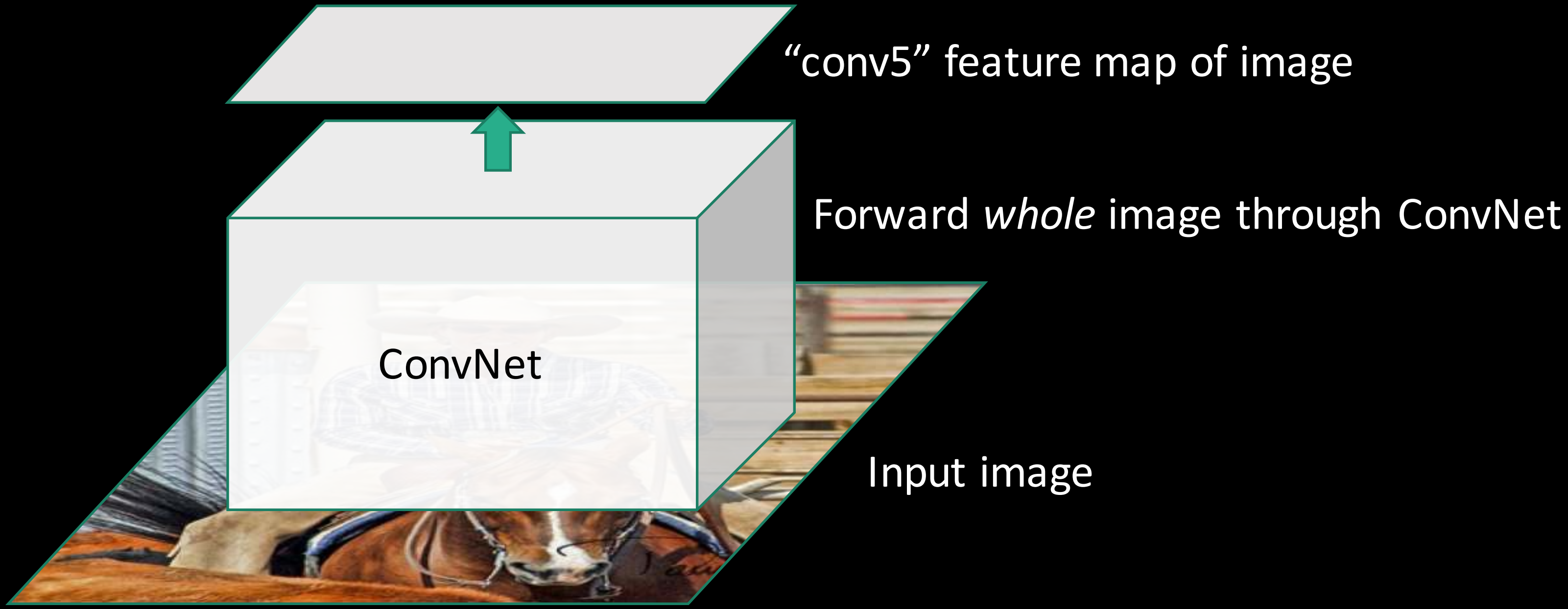
~2000 ConvNet forward passes per image

SPP-net

Input image

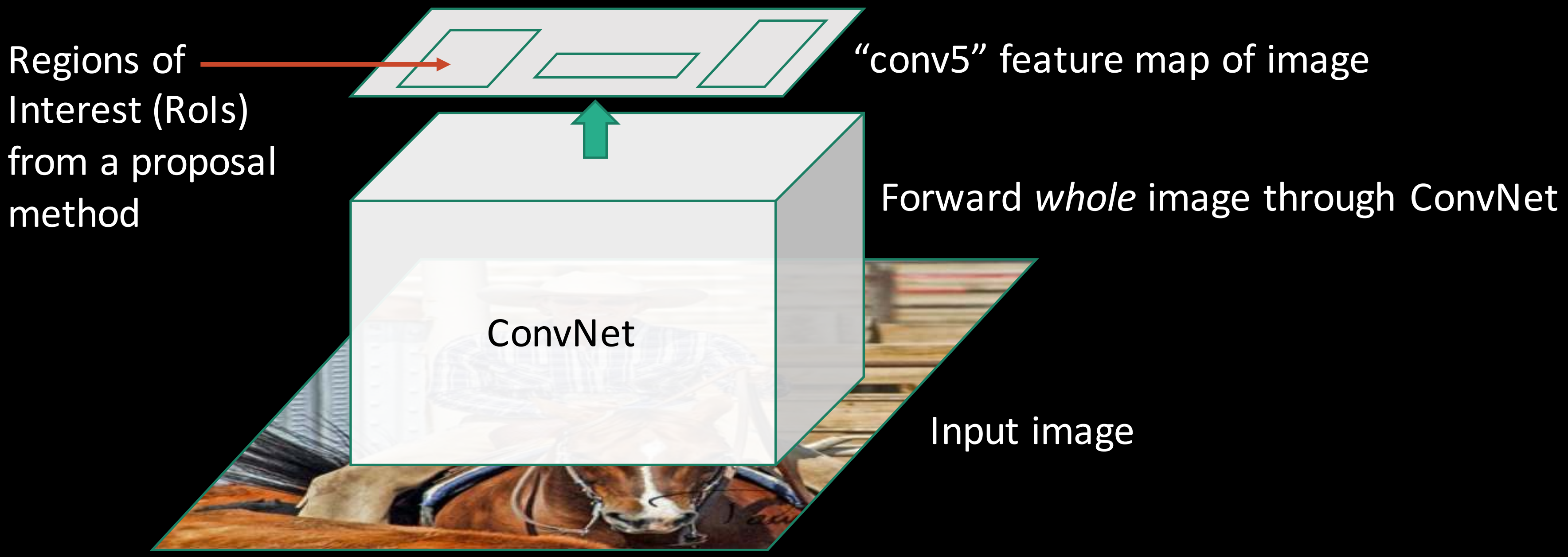
He et al. ECCV14.

SPP-net



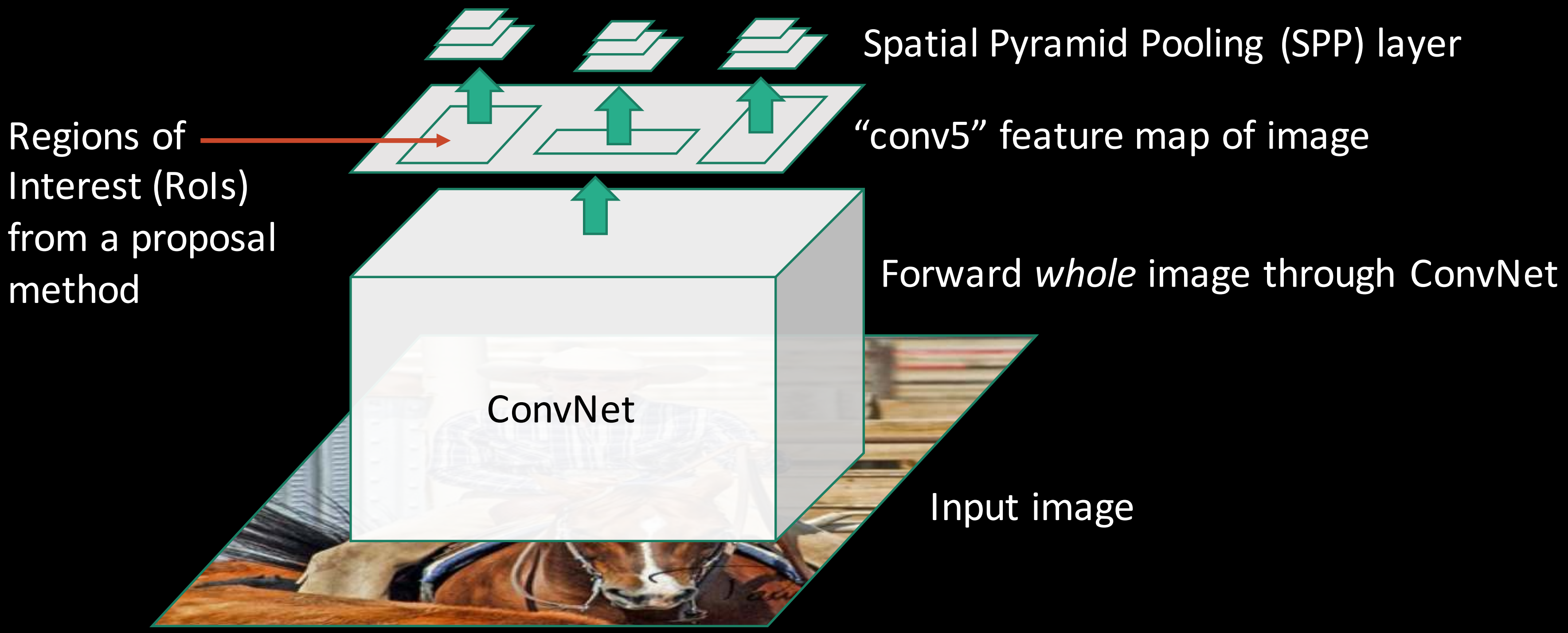
He et al. ECCV14.

SPP-net



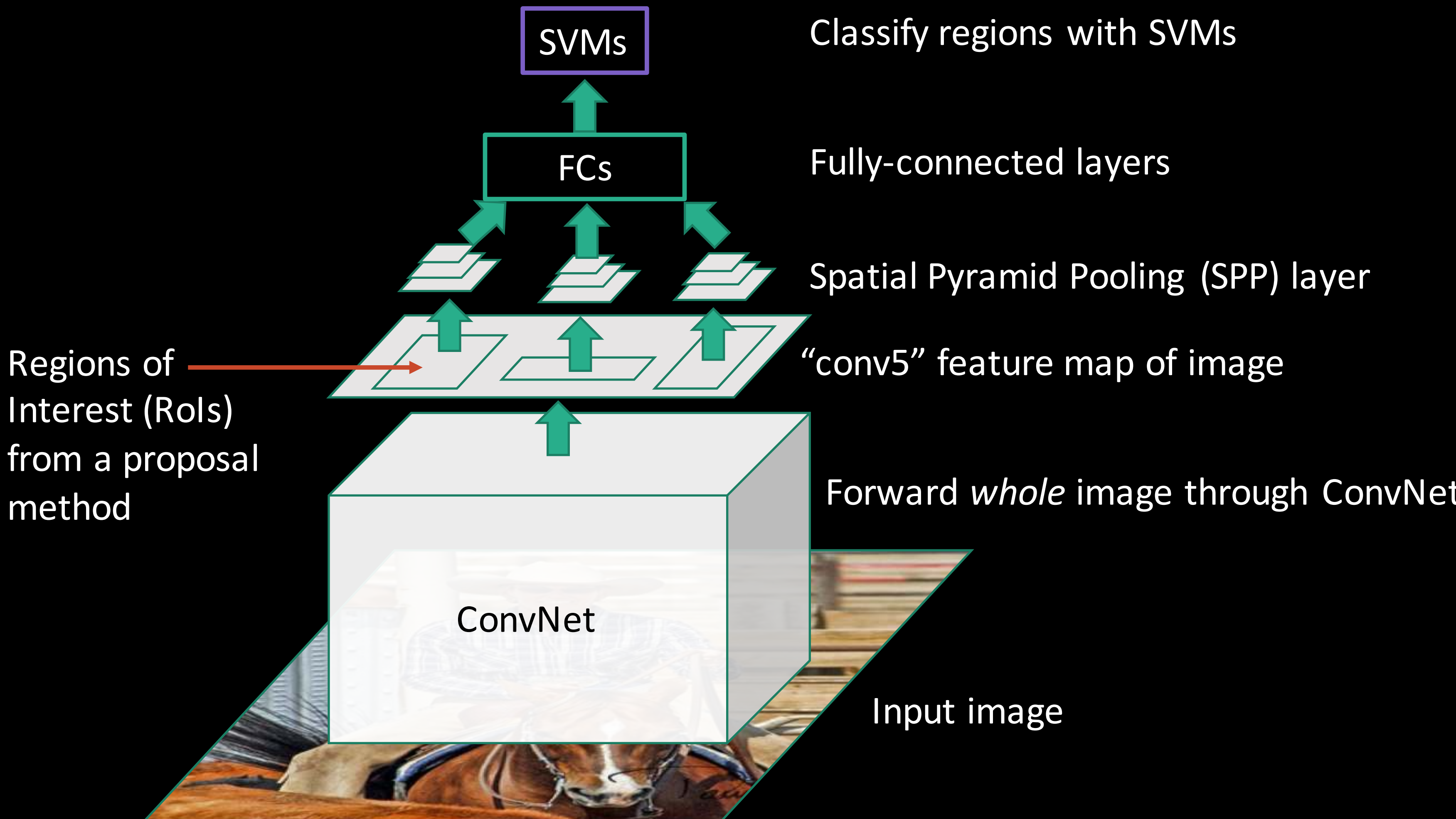
He et al. ECCV14.

SPP-net



He et al. ECCV14.

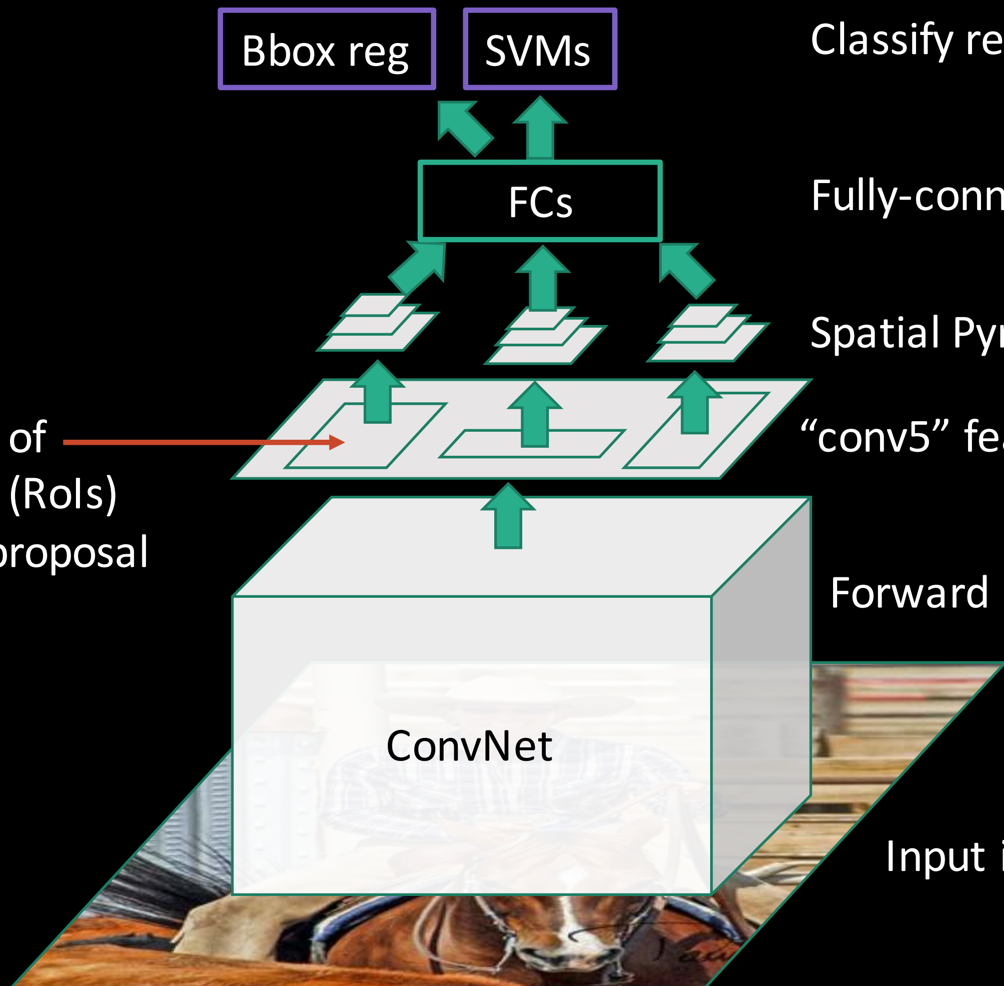
SPP-net



He et al. ECCV14.

Post hoc component

SPP-net

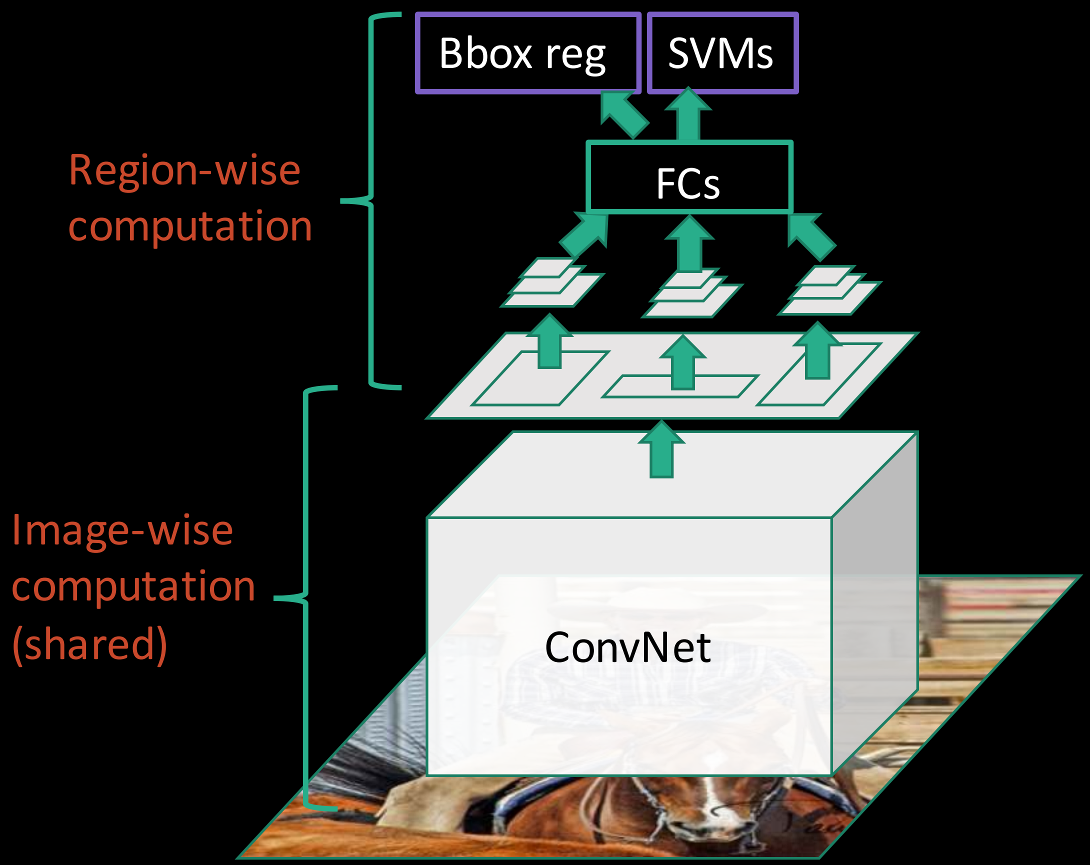


He et al. ECCV14.

Post hoc component

What’s good about SPP-net?

- Fixes one issue with R-CNN: makes testing fast



Post hoc component

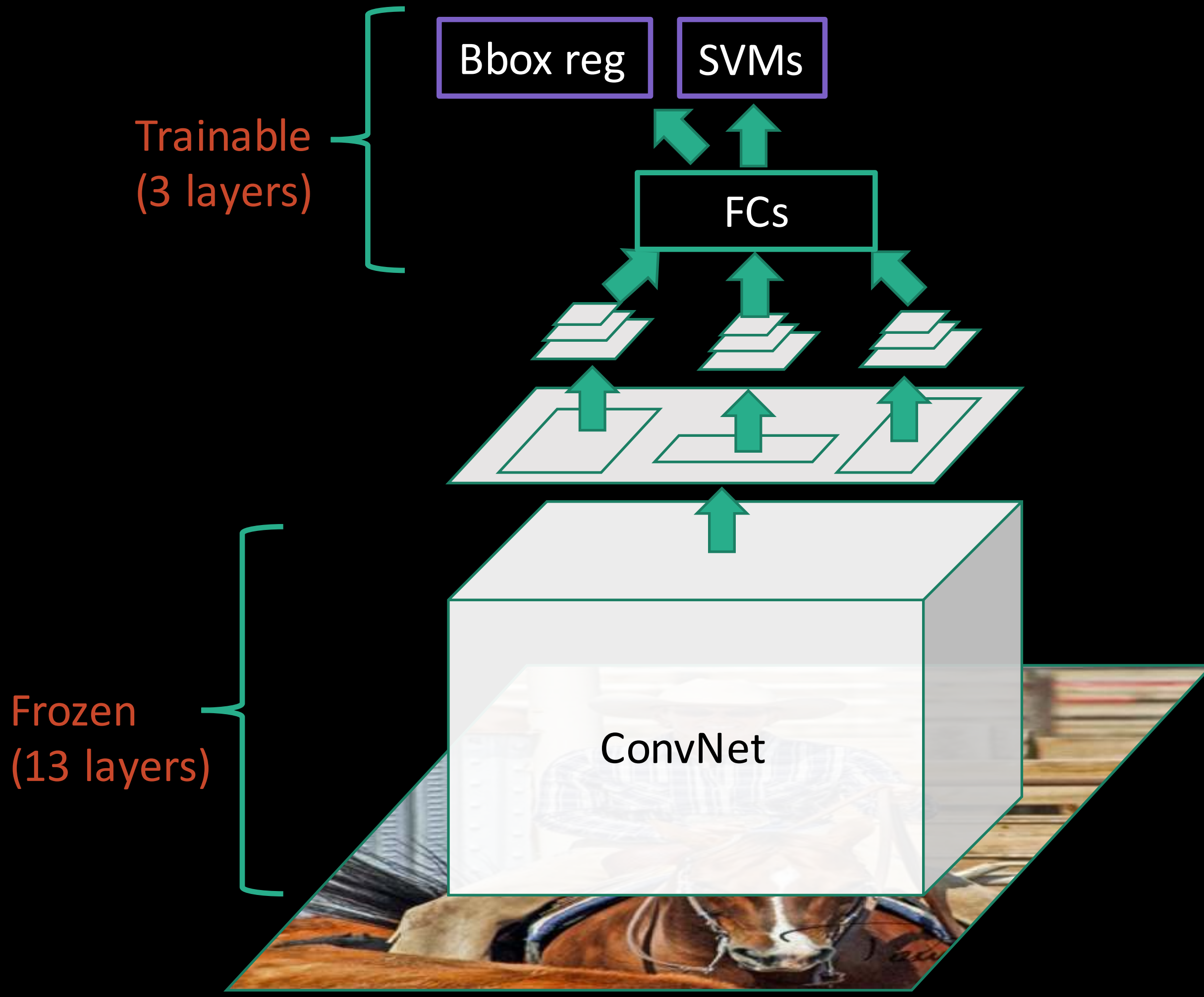
What’s wrong with SPP-net?

- Inherits the rest of R-CNN’s problems
  - Ad hoc training objectives
  - Training is slow (25h), takes a lot of disk space

What’s wrong with SPP-net?

- Inherits the rest of R-CNN’s problems
  - Ad hoc training objectives
  - Training is slow (though faster), takes a lot of disk space
- Introduces a new problem: cannot update parameters below SPP layer during training

SPP-net: the main limitation



He et al. ECCV14.

Post hoc component

Fast R-CNN

- Fast test-time, like SPP-net

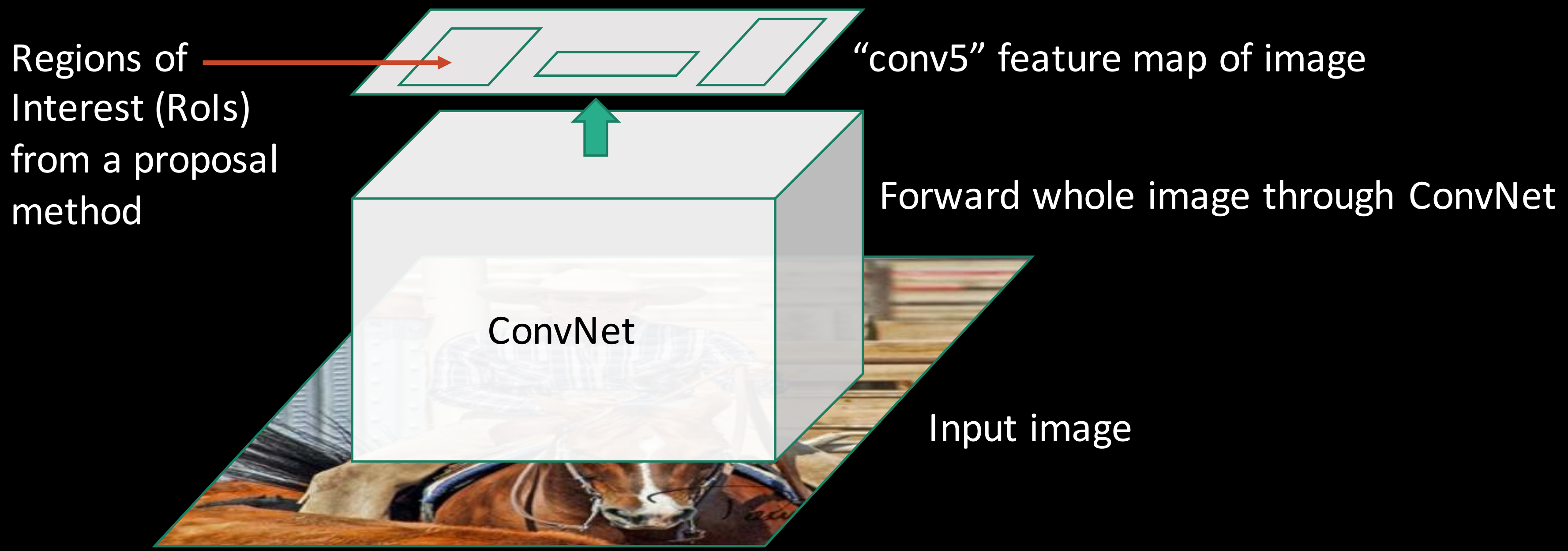
Fast R-CNN

- Fast test-time, like SPP-net
- One network, trained in one stage

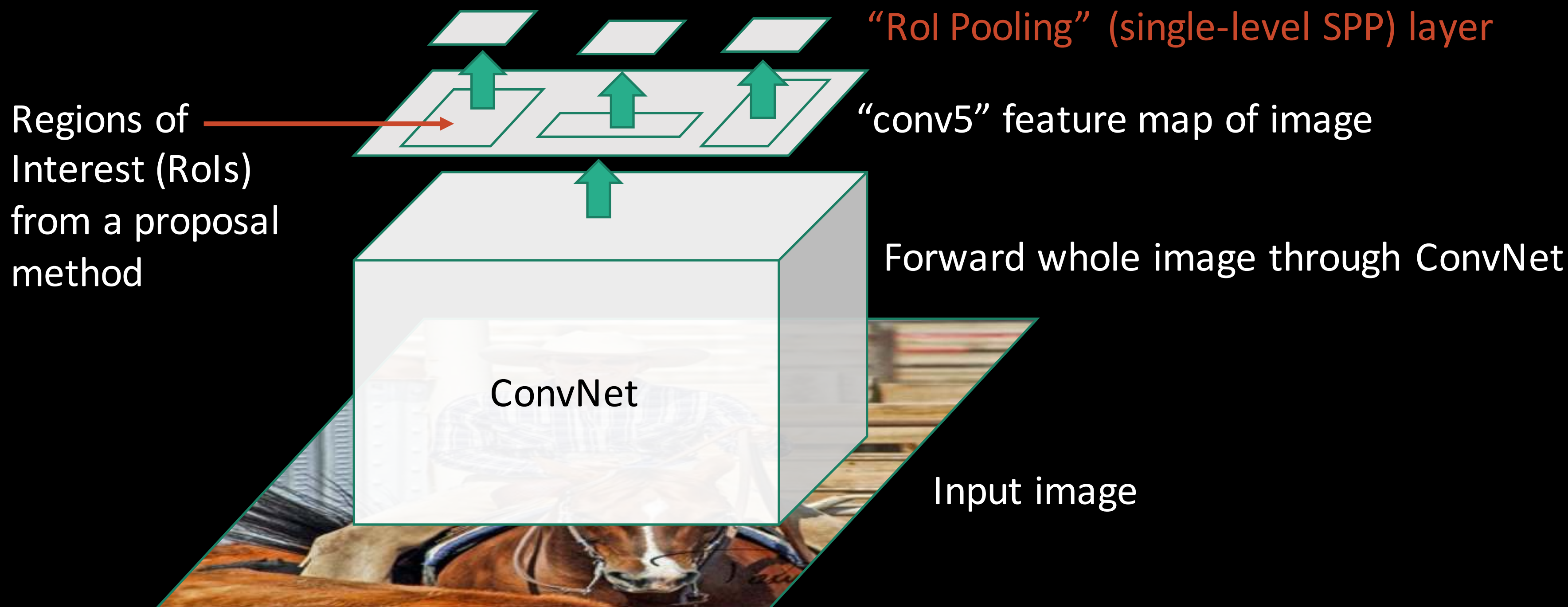
Fast R-CNN

- Fast test-time, like SPP-net
- One network, trained in one stage
- Higher mean average precision than slow R-CNN and SPP-net

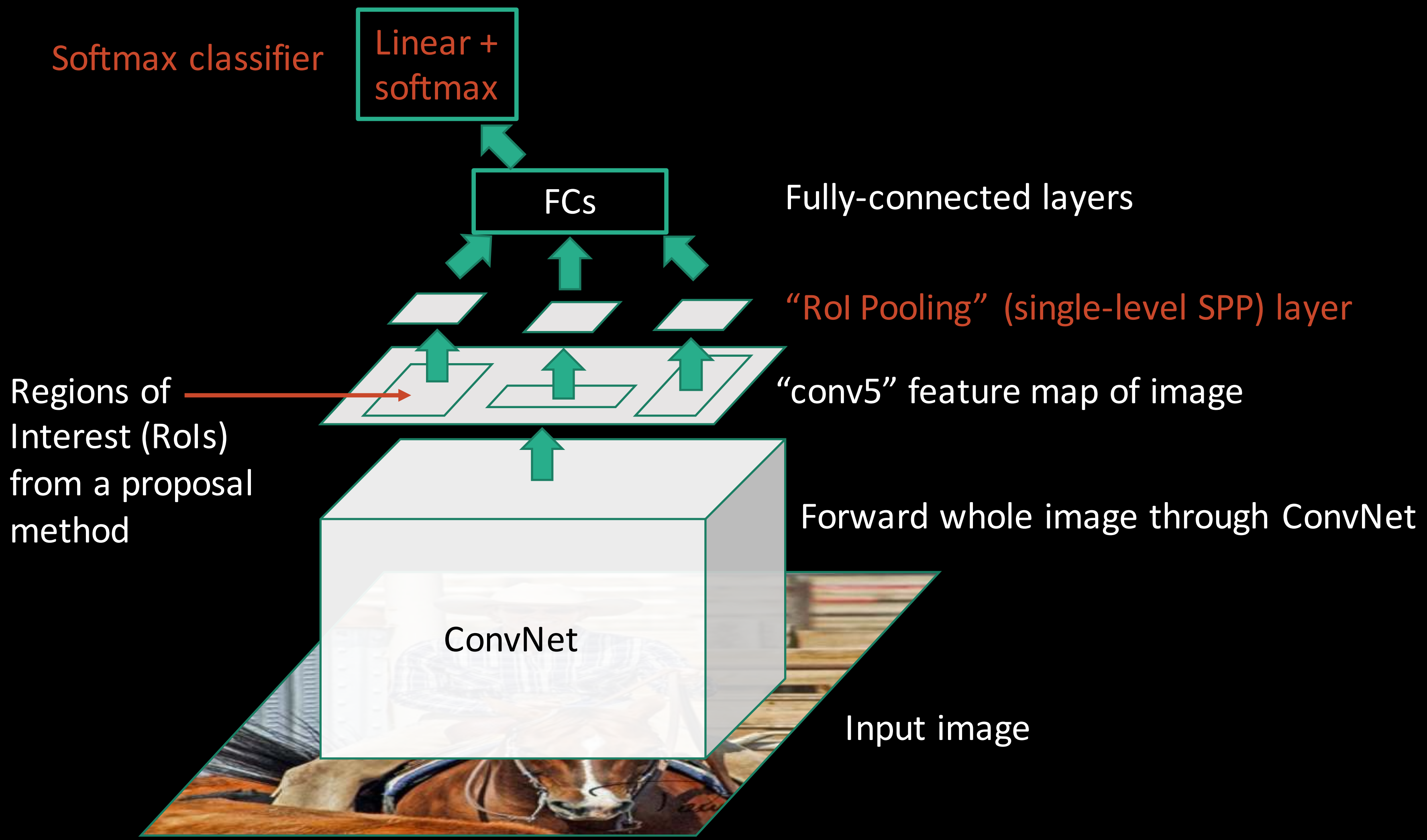
Fast R-CNN (test time)



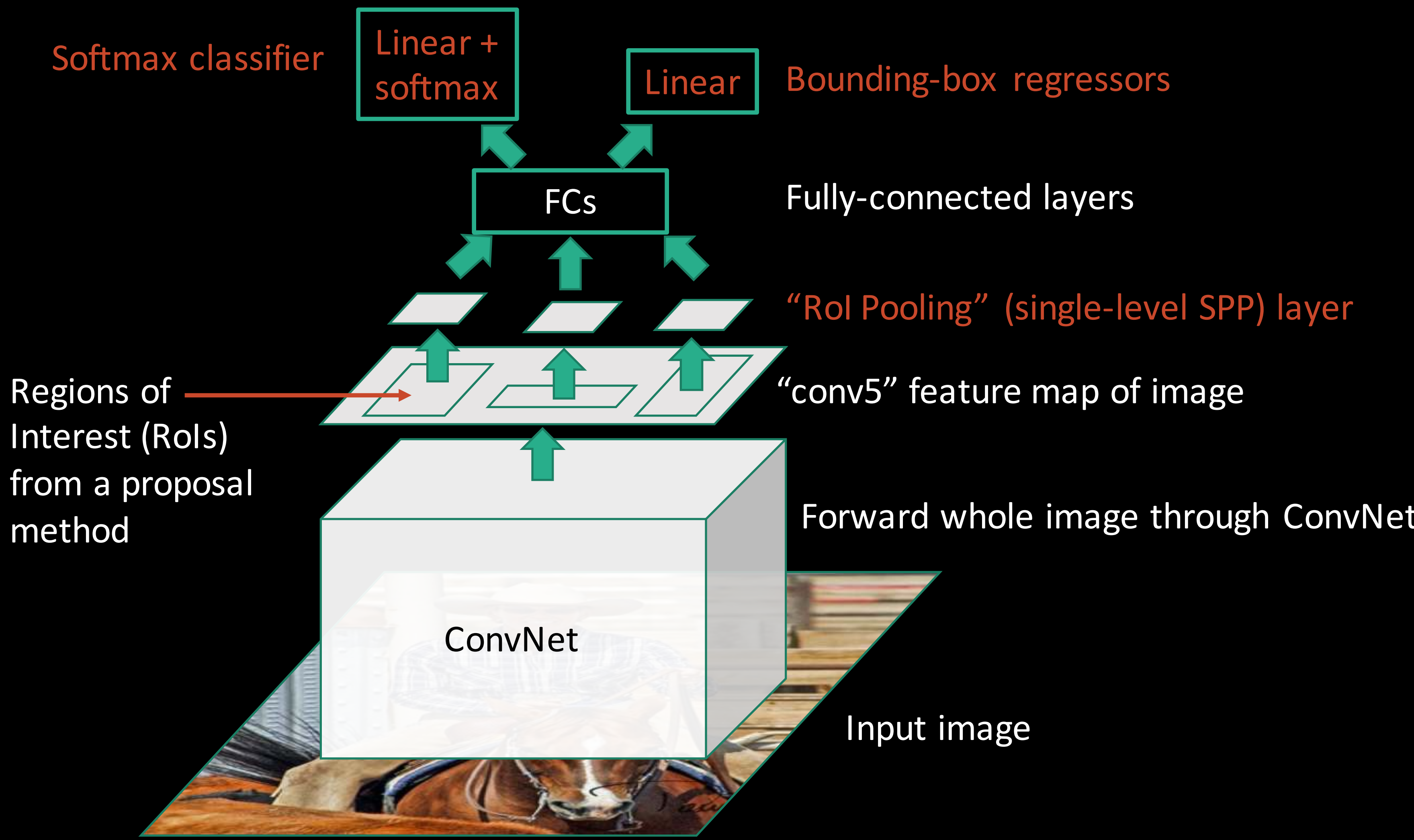
Fast R-CNN (test time)



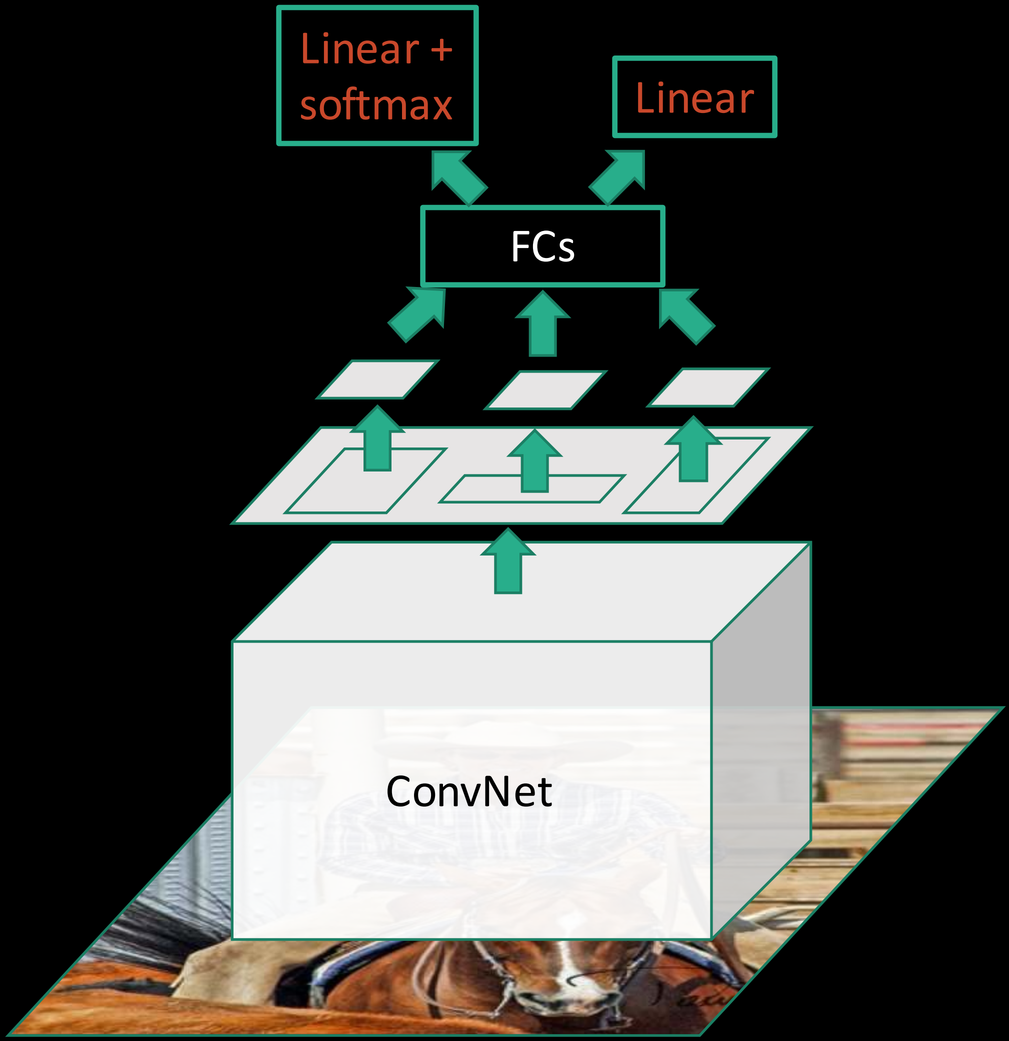
Fast R-CNN (test time)



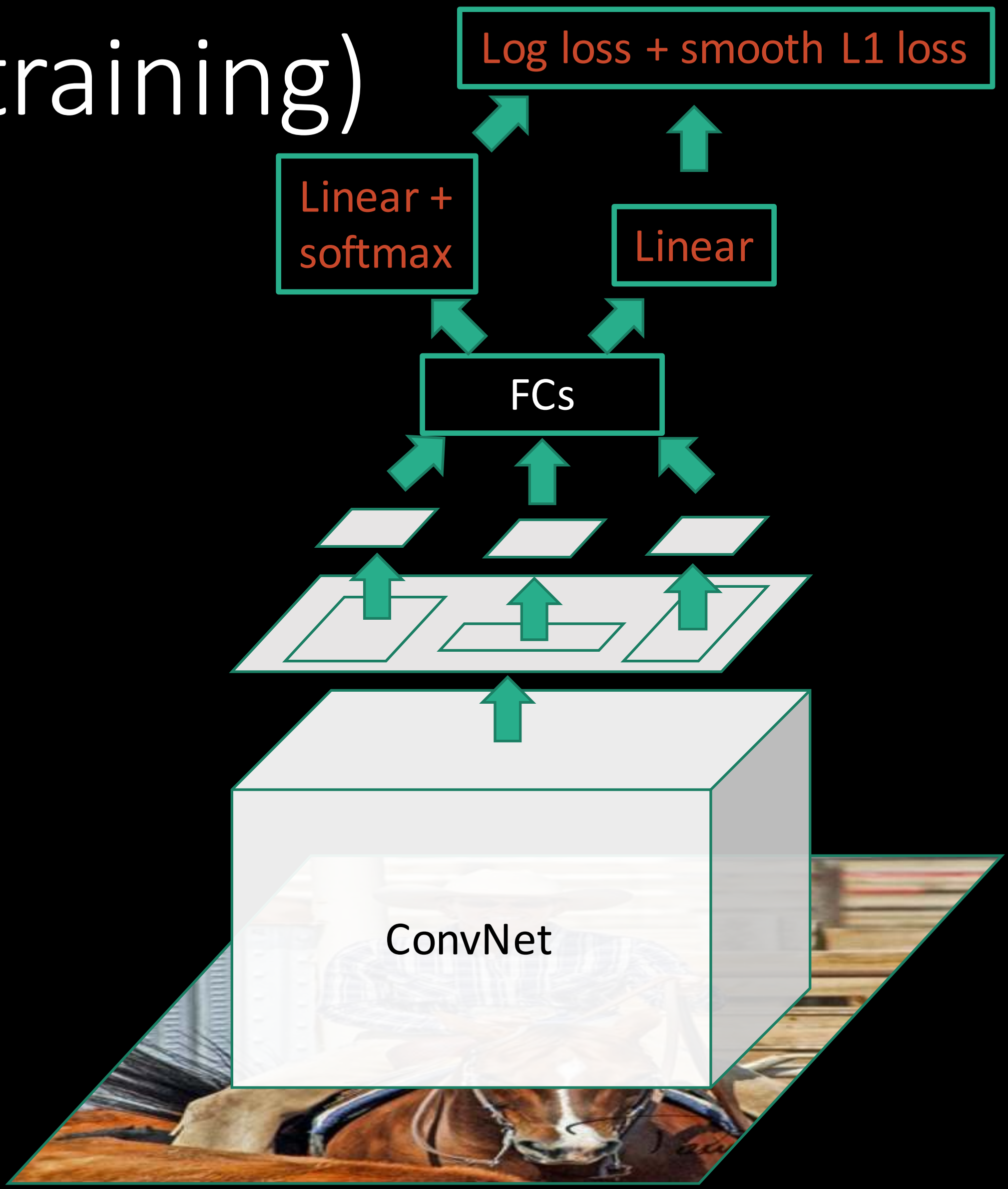
Fast R-CNN (test time)



Fast R-CNN (training)

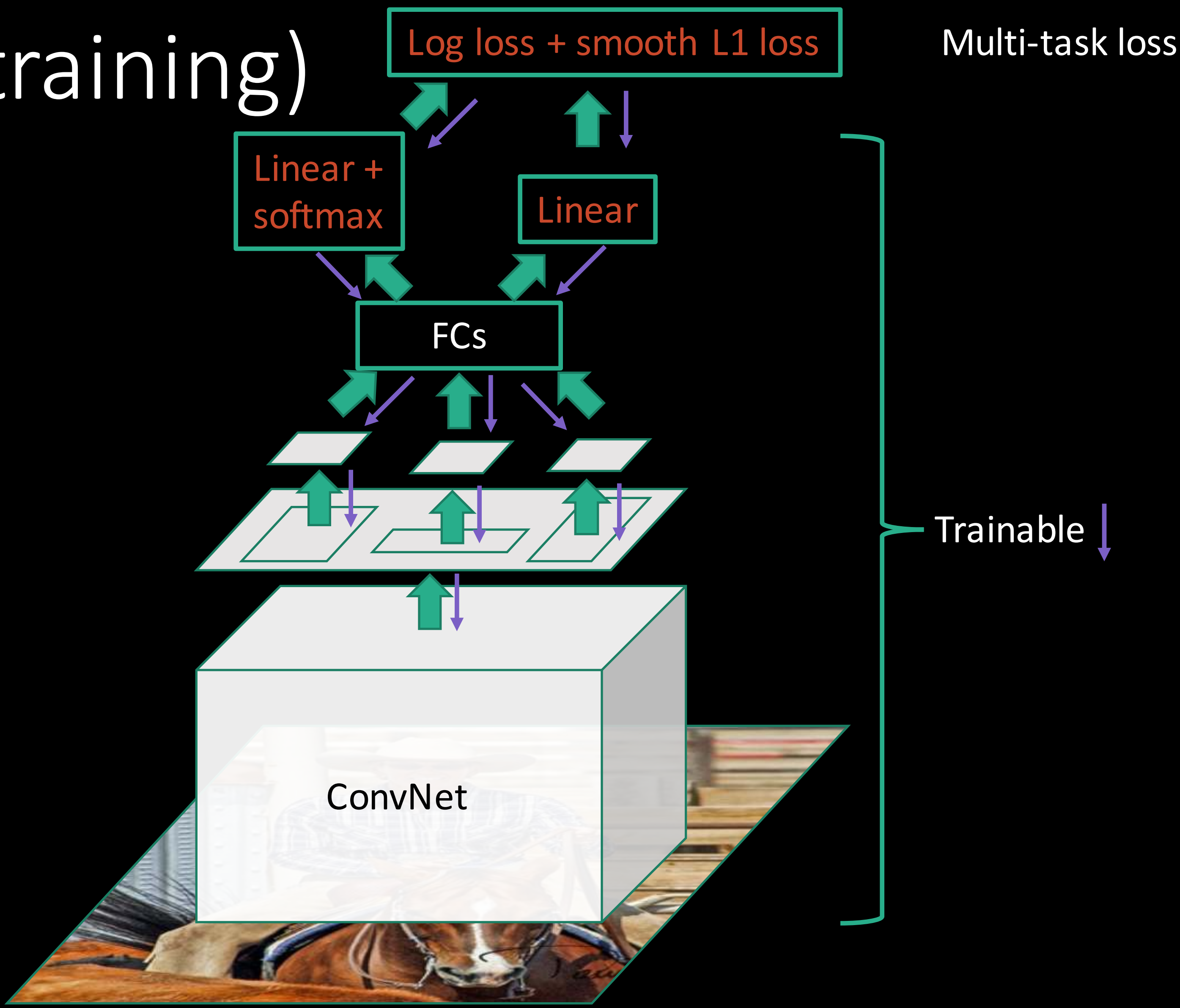


Fast R-CNN (training)



Multi-task loss

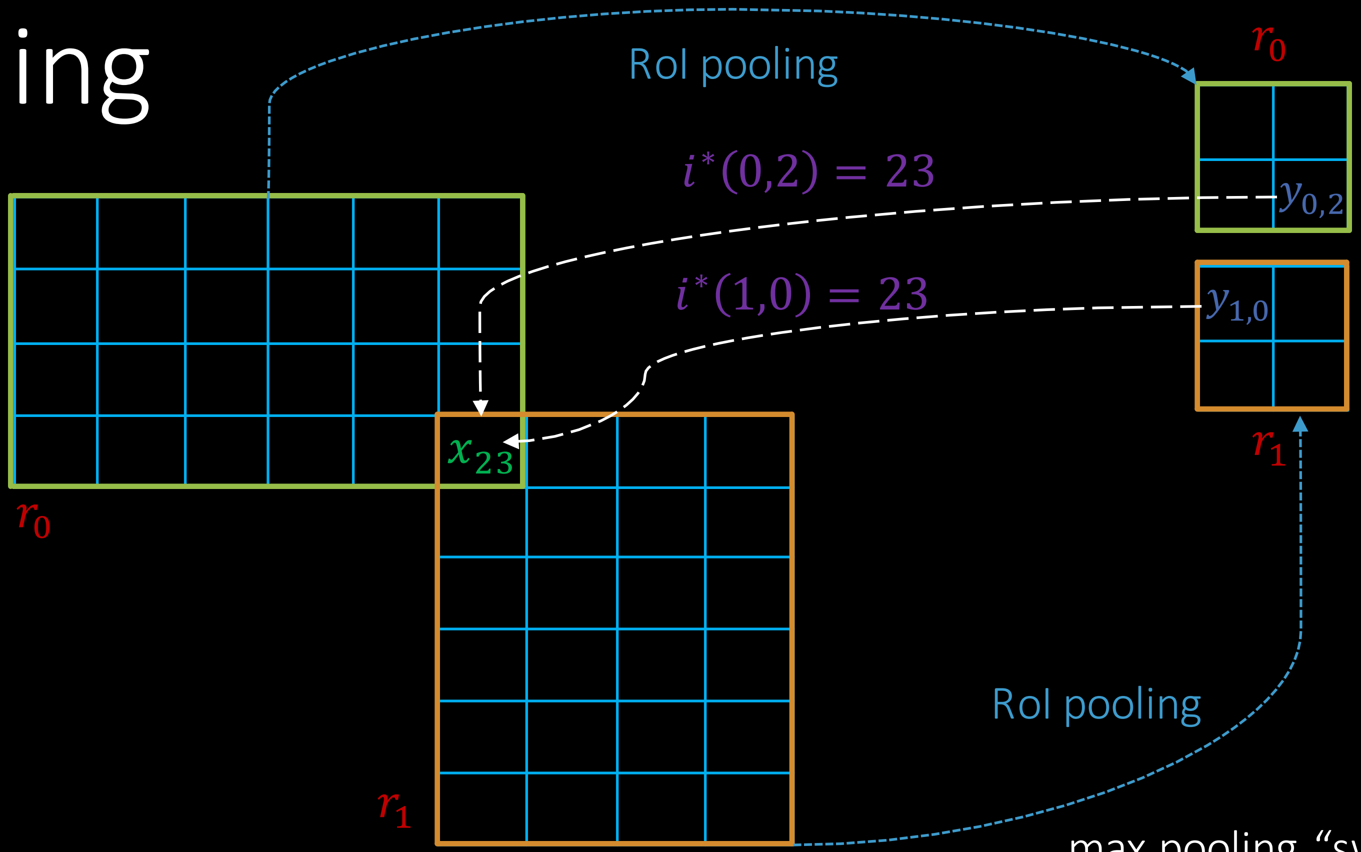
Fast R-CNN (training)



Obstacle #1: Differentiable RoI pooling

Region of Interest (RoI) pooling must be (sub-) differentiable to train conv layers

Obstacle #1: Differentiable RoI pooling



max pooling “switch”
(*i.e.* argmax back-pointer)

1 if r, j “pooled” input i; 0 o/w

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}}$$

Partial for x_i

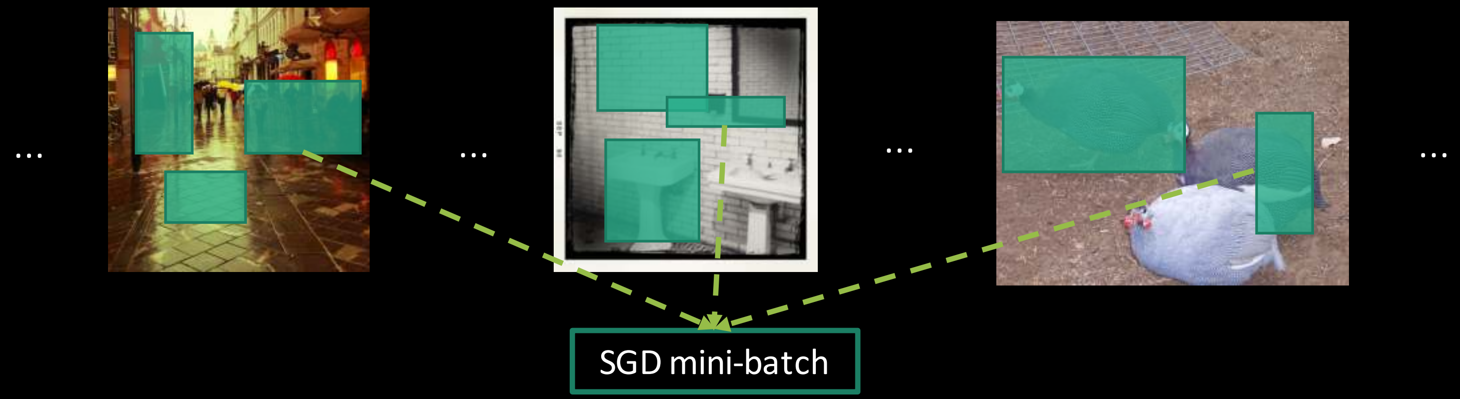
Over regions r, locations j

Partial from next layer

Obstacle #2: efficient SGD steps

Slow R-CNN and SPP-net use region-wise sampling to make mini-batches

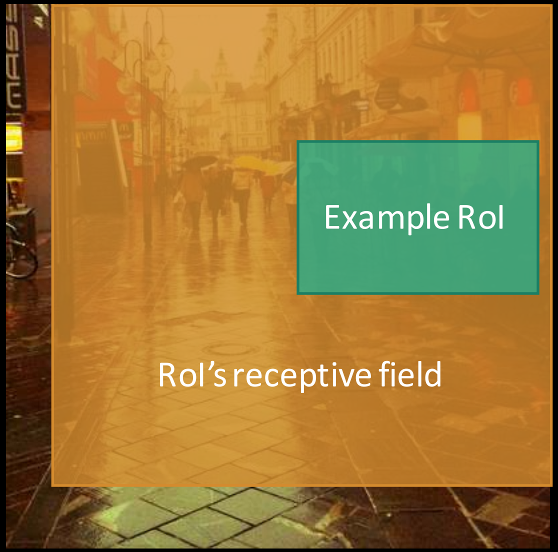
- Sample 128 example RoIs uniformly at random
- Examples will come from different images with high probability



Obstacle #2: efficient SGD steps

Note the receptive field for one example RoI is often very large

- Worst case: the receptive field is the entire image

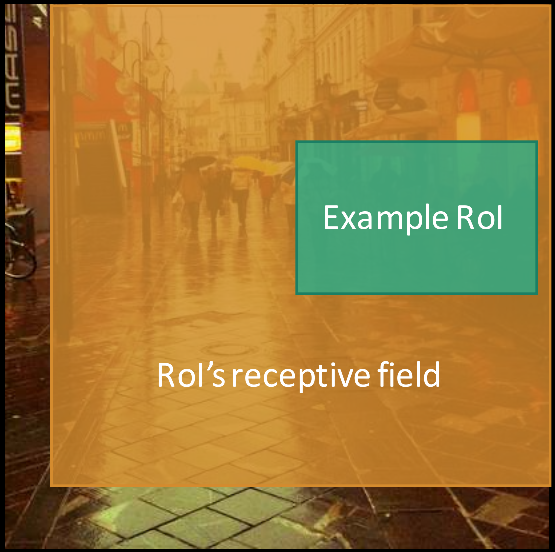


Obstacle #2: efficient SGD steps

Worst case cost per mini-batch (crude model of computational complexity)

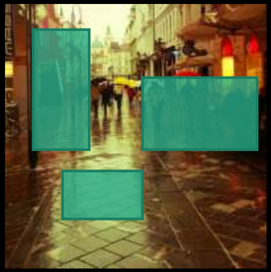
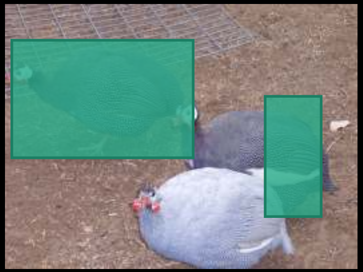
input size for Fast R-CNN input size for slow R-CNN

128*600*1000 / (128*224 *224) = 12x more computation than slow R-CNN



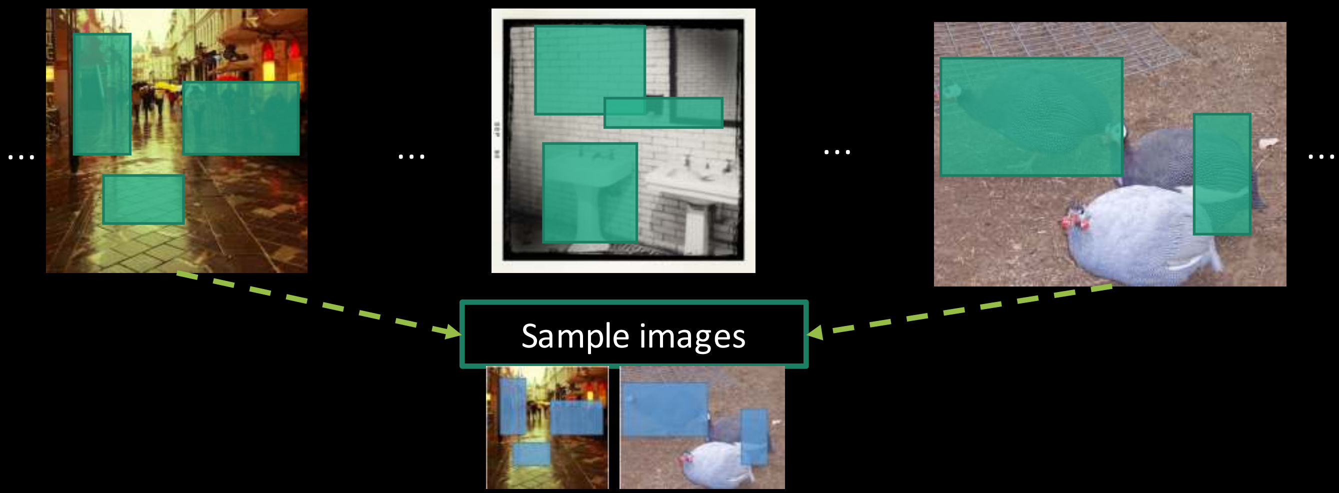
Obstacle #2: efficient SGD steps

Solution: use hierarchical sampling to build mini-batches

…  …  …  …

Obstacle #2: efficient SGD steps

Solution: use hierarchical sampling to build mini-batches



- Sample a small number of images (2)

Obstacle #2: efficient SGD steps

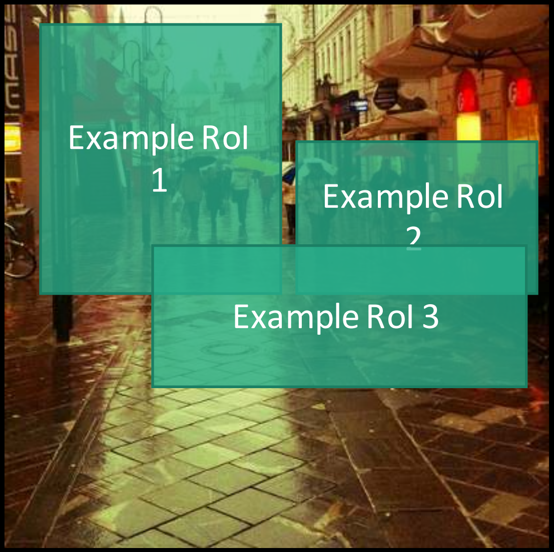
Solution: use hierarchical sampling to build mini-batches

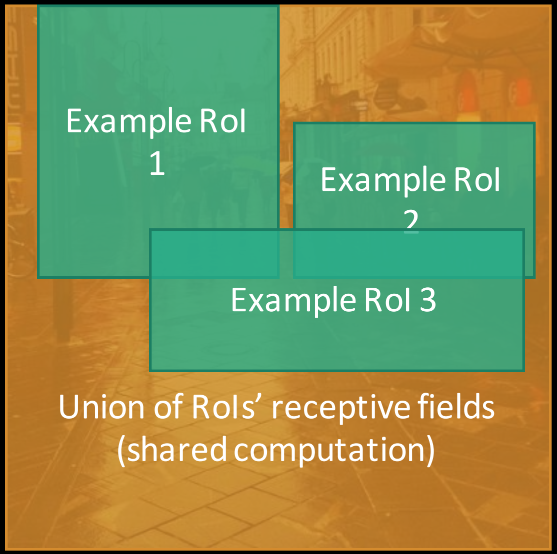
- Sample a small number of images (2)
- Sample many examples from each image (64)

Obstacle #2: efficient SGD steps

Use the test-time trick from SPP-net during training

- Share computation between overlapping examples from the same image

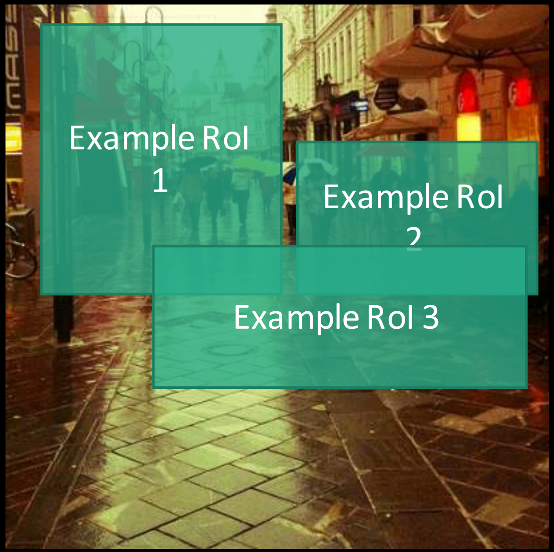


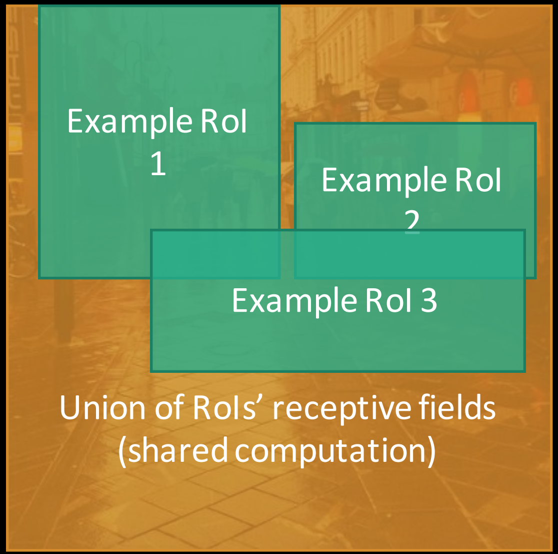


Obstacle #2: efficient SGD steps

Cost per mini-batch compared to slow R-CNN (same crude cost model)

- $\underbrace{2*600*1000}_{\text{input size for Fast R-CNN}} / (\underbrace{128*224*224}_{\text{input size for slow R-CNN}}) =$ 0.19x less computation than slow R-CNN





Main results

	Fast R-CNN	R-CNN [1]	SPP-net [2]
Train time (h)	**9.5**	84	25
- Speedup	**8.8x**	1x	3.4x
Test time / image	**0.32s**	47.0s	2.3s
Test speedup	**146x**	1x	20x
mAP	**66.9%**	66.0%	63.1%

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

[1] Girshick et al. CVPR14.
[2] He et al. ECCV14.

Main results

	Fast R-CNN	R-CNN [1]	SPP-net [2]
Train time (h)	**9.5**	84	25
- Speedup	**8.8x**	1x	3.4x
Test time / image	**0.32s**	47.0s	2.3s
Test speedup	**146x**	1x	20x
mAP	**66.9%**	66.0%	63.1%

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

[1] Girshick et al. CVPR14.
[2] He et al. ECCV14.

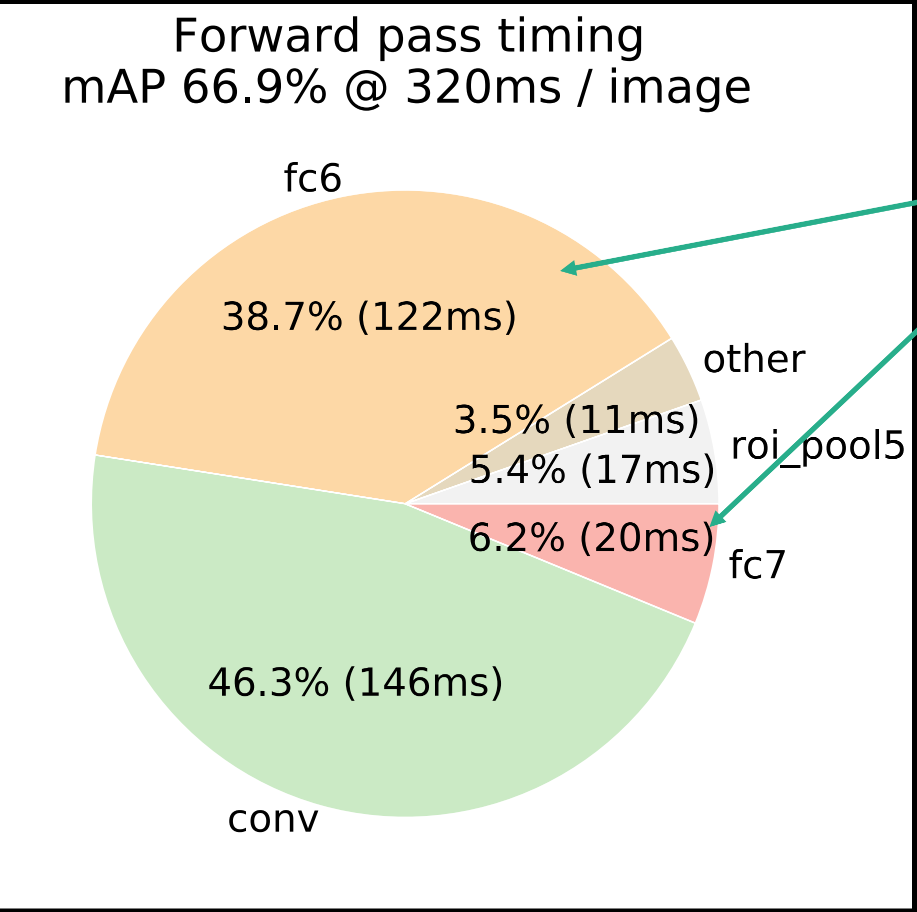
Main results

	Fast R-CNN	**R-CNN [1]**	**SPP-net [2]**
Train time (h)	**9.5**	84	25
- Speedup	**8.8x**	1x	3.4x
Test time / image	**0.32s**	47.0s	2.3s
Test speedup	**146x**	1x	20x
mAP	**66.9%**	66.0%	63.1%

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

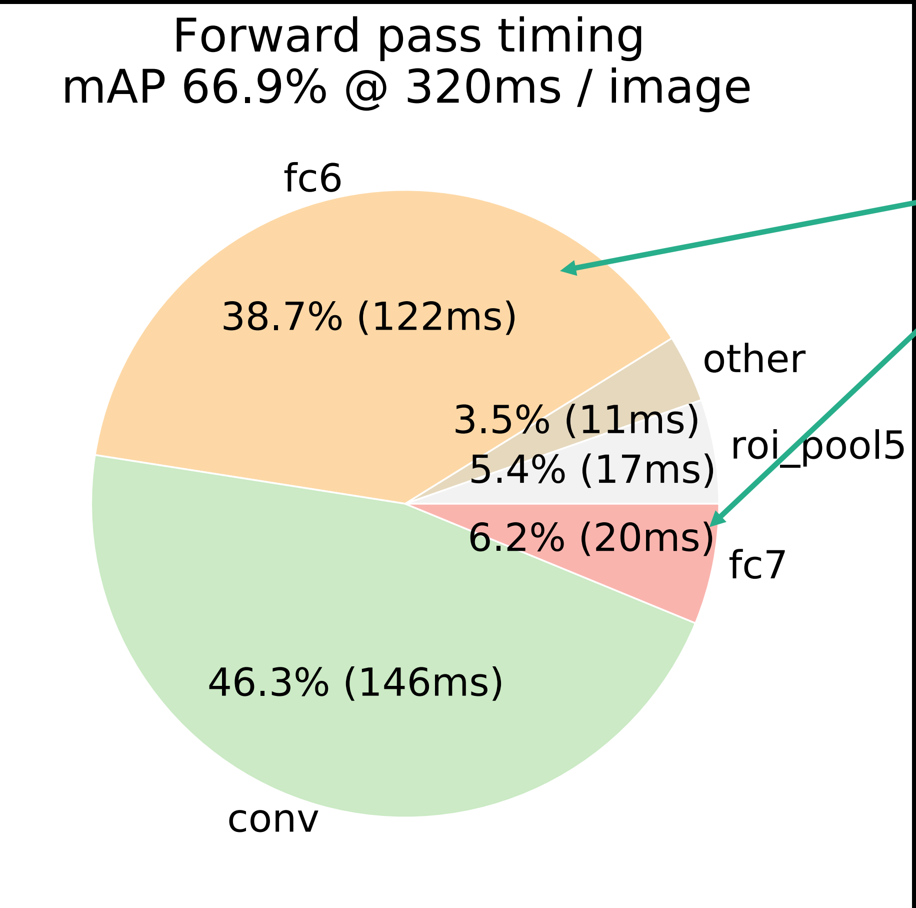
[1] Girshick et al. CVPR14.
[2] He et al. ECCV14.

Further test-time speedups



Fully connected layers take 45% of the forward pass time

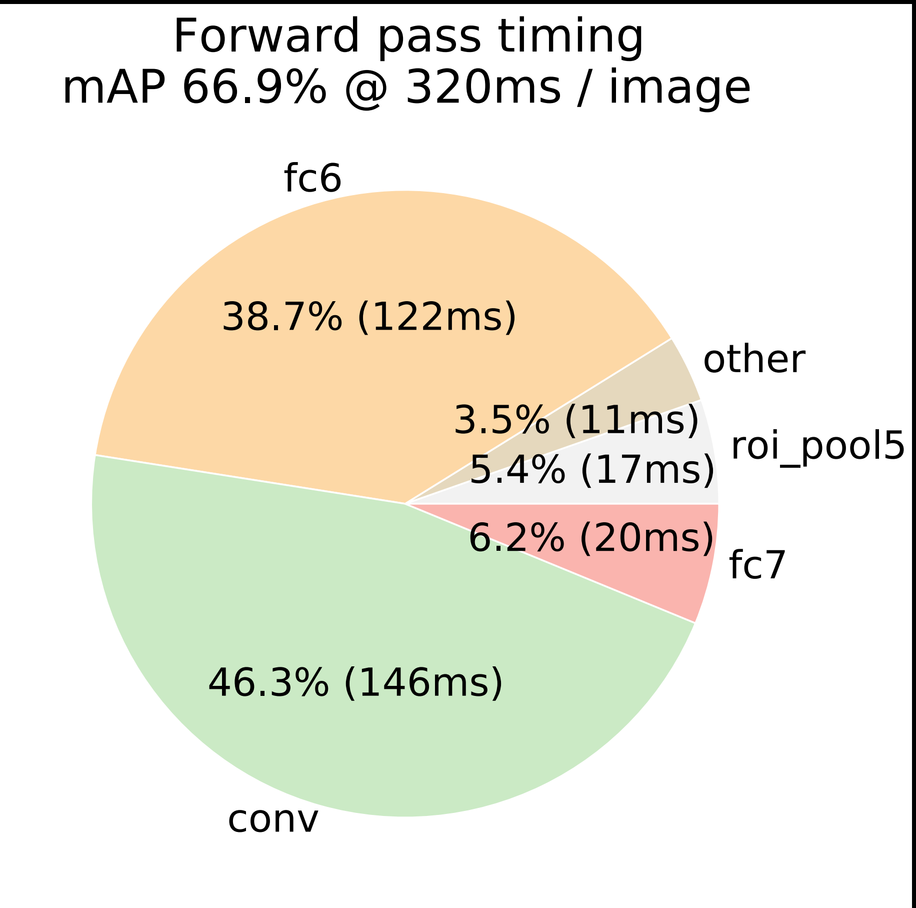
Further test-time speedups



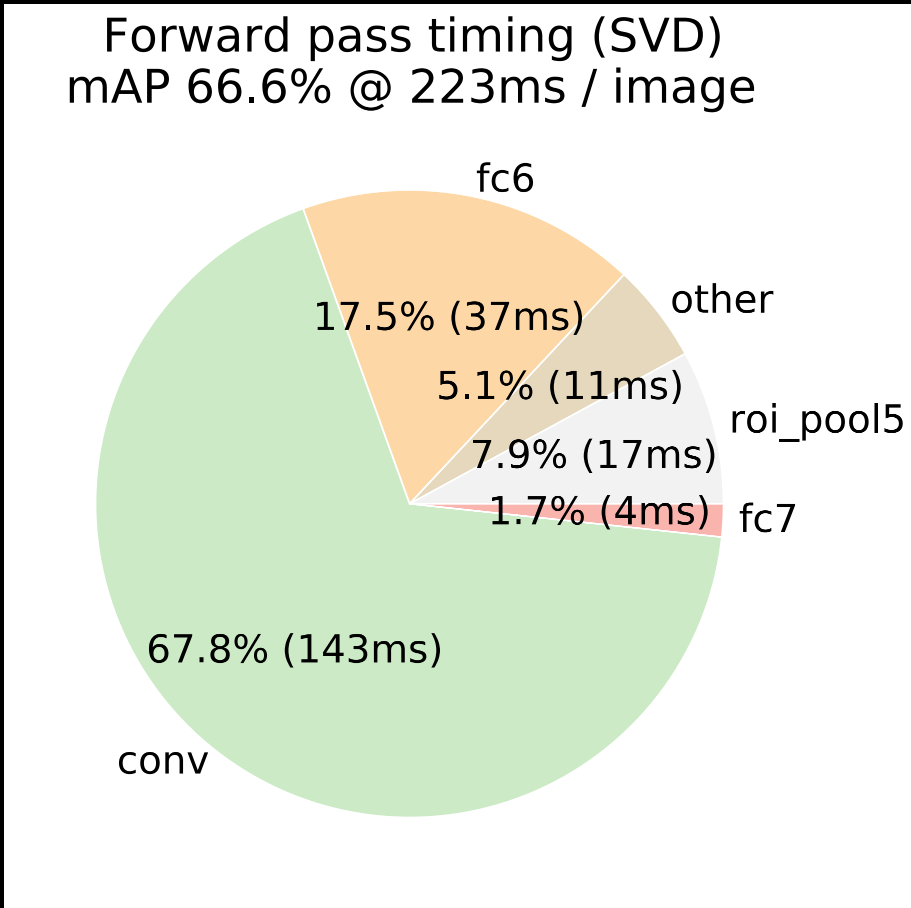
Compress these layers with truncated SVD

J. Xue, J. Li, and Y. Gong.
Restructuring of deep neural network acoustic models with singular value decomposition.
Interspeech, 2013.

Further test-time speedups



Without SVD



With SVD

Other findings

End-to-end training matters

	Fast R-CNN (VGG16)		
Fine-tune layers	$\geq$ fc6	$\geq$ conv3_1	$\geq$ conv2_1
VOC07 mAP	61.4%	66.9%	67.2%
Test time per image	0.32s	0.32s	0.32s

1.4x slower training

Multi-task training helps

	Fast R-CNN (VGG16)			
Multi-task training?		Y		Y
Stage-wise training?			Y	
Test-time bbox reg.			Y	Y
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

Multi-task training helps

	Fast R-CNN (VGG16)			
Multi-task training?		Y		Y
Stage-wise training?			Y	
Test-time bbox reg.			Y	Y
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

Trained without
a bbox regressor

Multi-task training helps

	Fast R-CNN (VGG16)			
Multi-task training?		Y		Y
Stage-wise training?			Y	
Test-time bbox reg.			Y	Y
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

Trained with
a bbox regressor,
but it's disabled at
test time

Multi-task training helps

	Fast R-CNN (VGG16)			
Multi-task training?		Y		Y
Stage-wise training?			Y	
Test-time bbox reg.			Y	Y
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

Post hoc bbox regressor, used at test time

Multi-task training helps

	Fast R-CNN (VGG16)			
Multi-task training?		Y		Y
Stage-wise training?			Y	
Test-time bbox reg.			Y	Y
VOC07 mAP	62.6%	63.4%	64.0%	66.9%

Multi-task objective,
using bbox regressors
at test time

What’s still wrong?

- Out-of-network region proposals
  - Selective search: 2s / im; EdgeBoxes: 0.2s / im
- Fortunately, we have a solution
  - Our follow-up work was presented last week at NIPS

Shaoqing Ren, Kaiming He, Ross Girshick & Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” NIPS 2015.

Fast R-CNN take-aways

- End-to-end training of deep ConvNets for detection
- Fast training times
- Open source for easy experimentation
  "I think [the Fast R-CNN] code is average-somewhat above average for what it is." – sporkles on r/MachineLearning
- A large number of ImageNet detection and COCO detection methods are built on Fast R-CNN
  Checkout the ImageNet / COCO Challenge workshop on Thursday!

Reproducible research – get the code!

http://git.io/vBqm5

Thanks!

rbg@fb.com

Softmax works well (vs. post hoc SVMs)

Method (VGG16)	classifier	VOC07 mAP
Slow R-CNN	Post hoc SVM	66.0%
Fast R-CNN	Post hoc SVM	66.8%
Fast R-CNN	Softmax	66.9%

More proposals is harmful