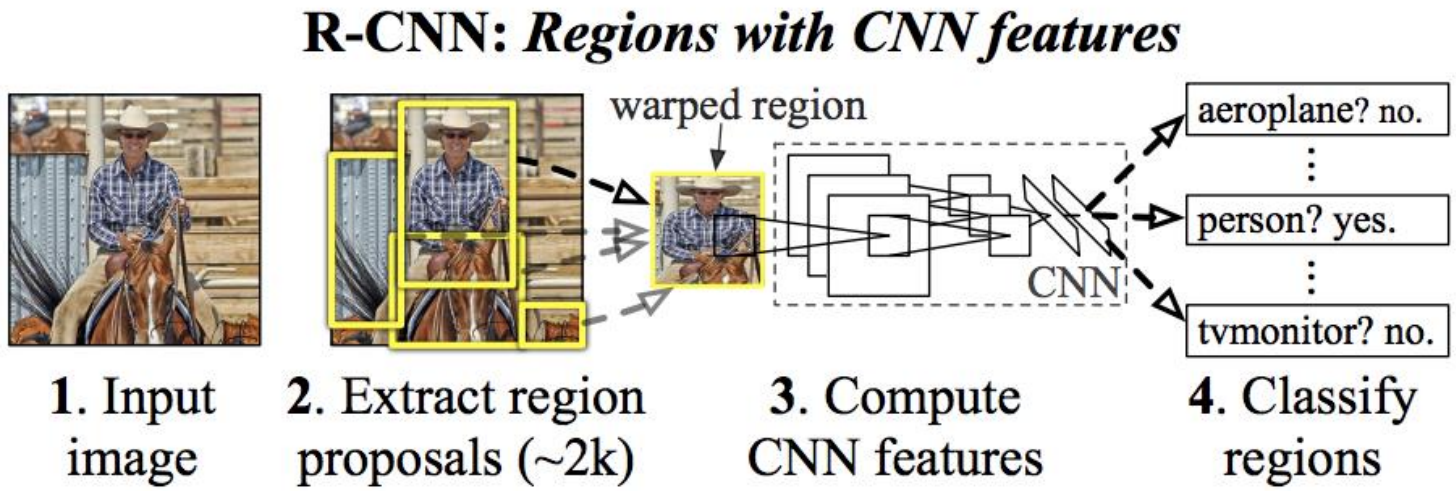
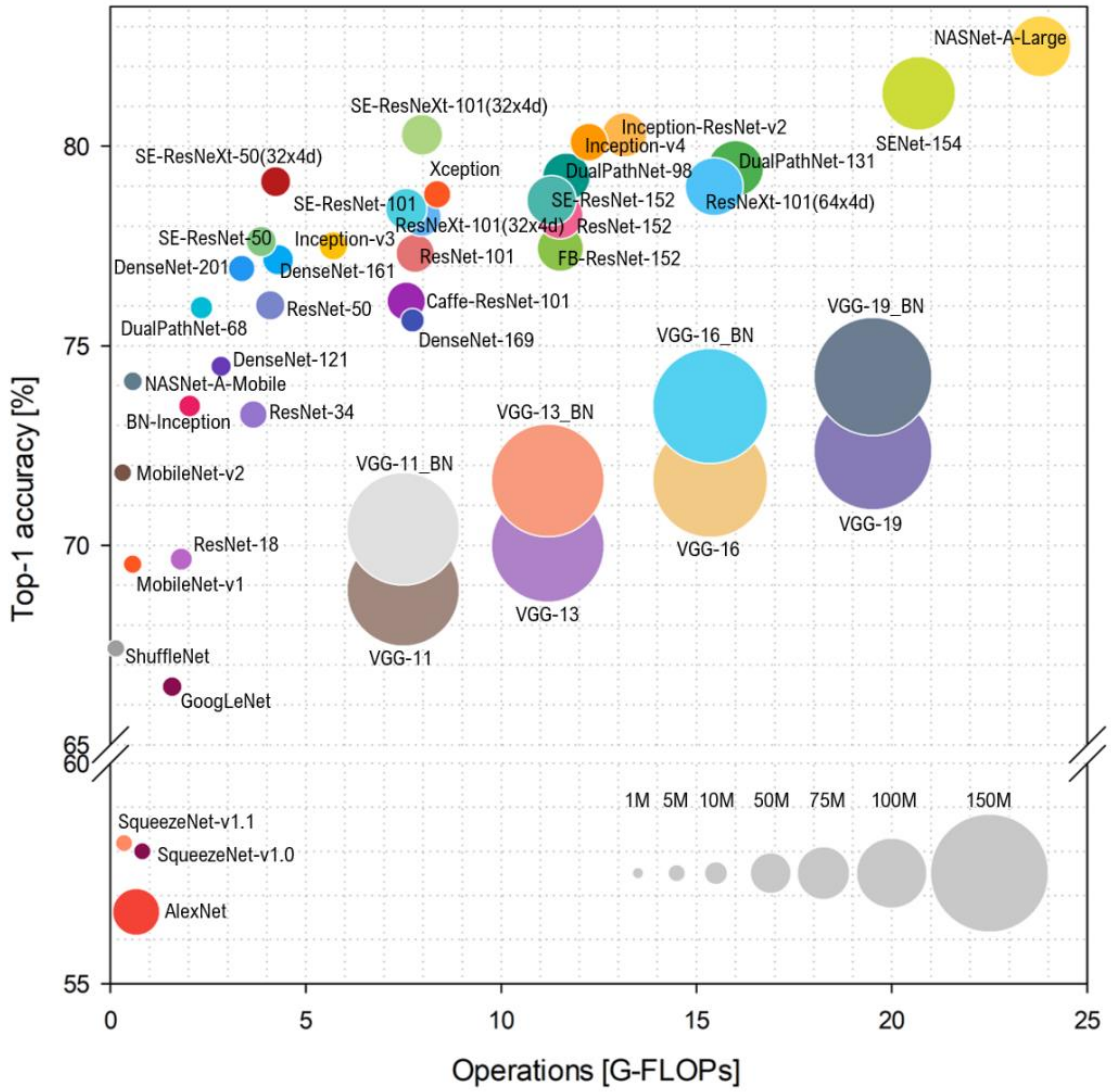
Specialized architectures



State-of-the-Art



Neural architecture search

- Why not learn the optimal neural architecture?
- How to backprop through discrete, non-differentiable variables?
  - Architectures/graphs, number of layers, number of nodes, connectivity
  - Several workarounds to circumvent the problem
- Should you run your own neural architecture search?
  - If you are Facebook or Google, yes!

Evolutionary search for NAS

- DARTS: Differentiable Architecture Search, Liu et al., 2018
- Efficient Neural Architectur Search via Parameter Sharing, Pham et al., 2018
- Evolving Space-Time Neura Architectures for Videos, Piergiovanni et al. 2018
- Regularized Evolution for Image Classifier Architecture Search, Real et al., 2019

Algorithm 1 Evolutionary search algorithm

function SEARCH
  Randomly initialize the population, P
  Evaluate each individual in P
  for $i <$ number of evolutionary rounds **do**
    $S =$ random sample of 25 individuals
    $parent =$ the most fit individual in S
    $child = parent$
    for $\max(\lceil d - \frac{i}{r} \rceil, 1)$ **do**
      $child = mutate(child)$
    end for
    evaluate *child* and add to population
    remove least fit individual from population
  end for
end function

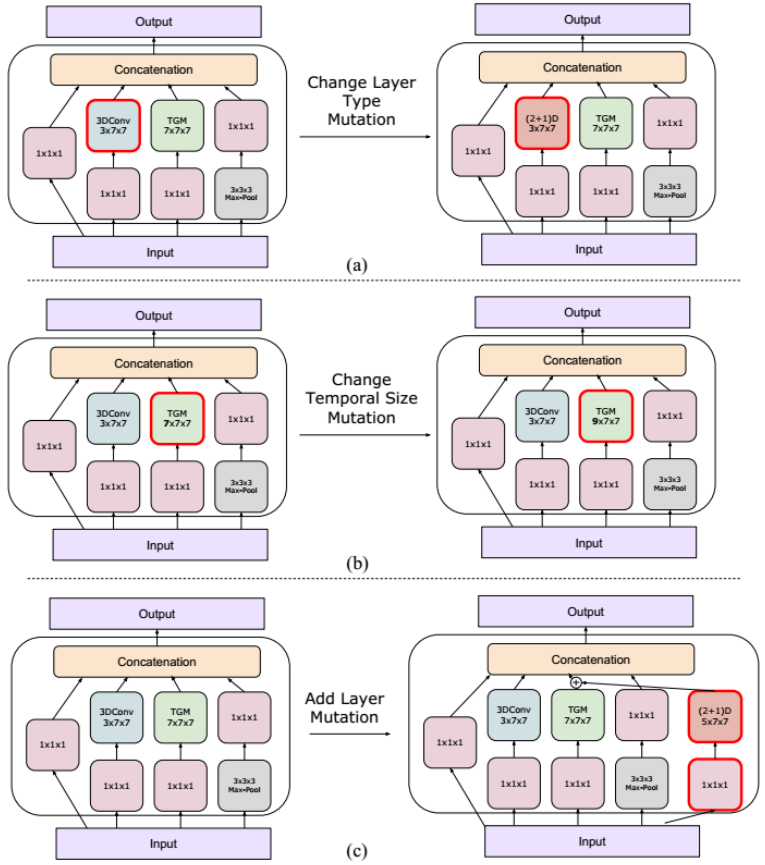
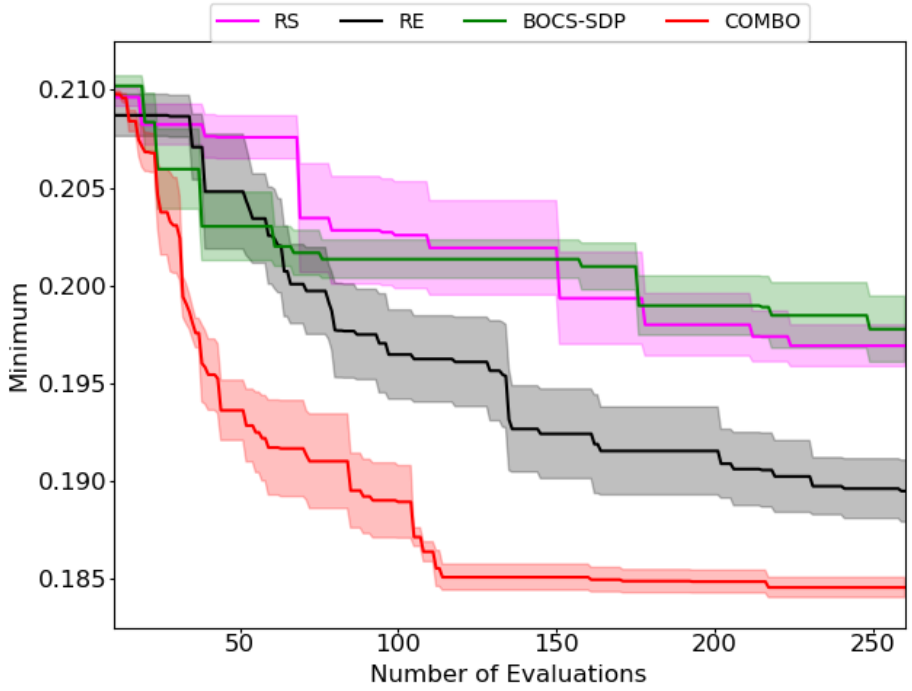
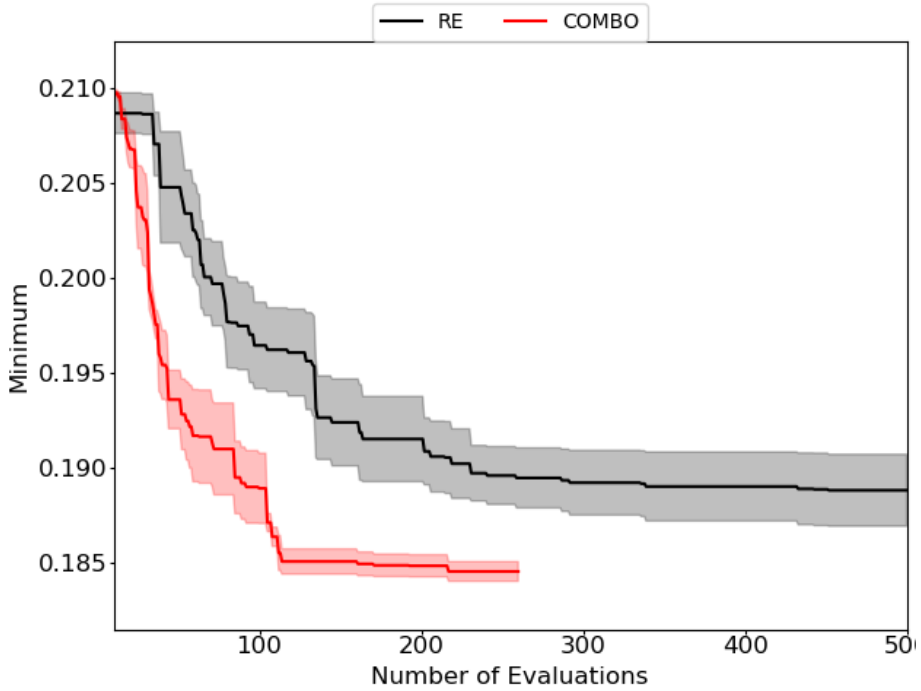


Figure 5. Example mutations applied to a module, including (a) layer type change, (b) filter length change, and (c) layer addition.

Combinatorial Bayesian Optimization for NAS

- Treat neural architecture search as hyperparameter optimization
- Learn bayesian model of architecture space and sample likely good architectures

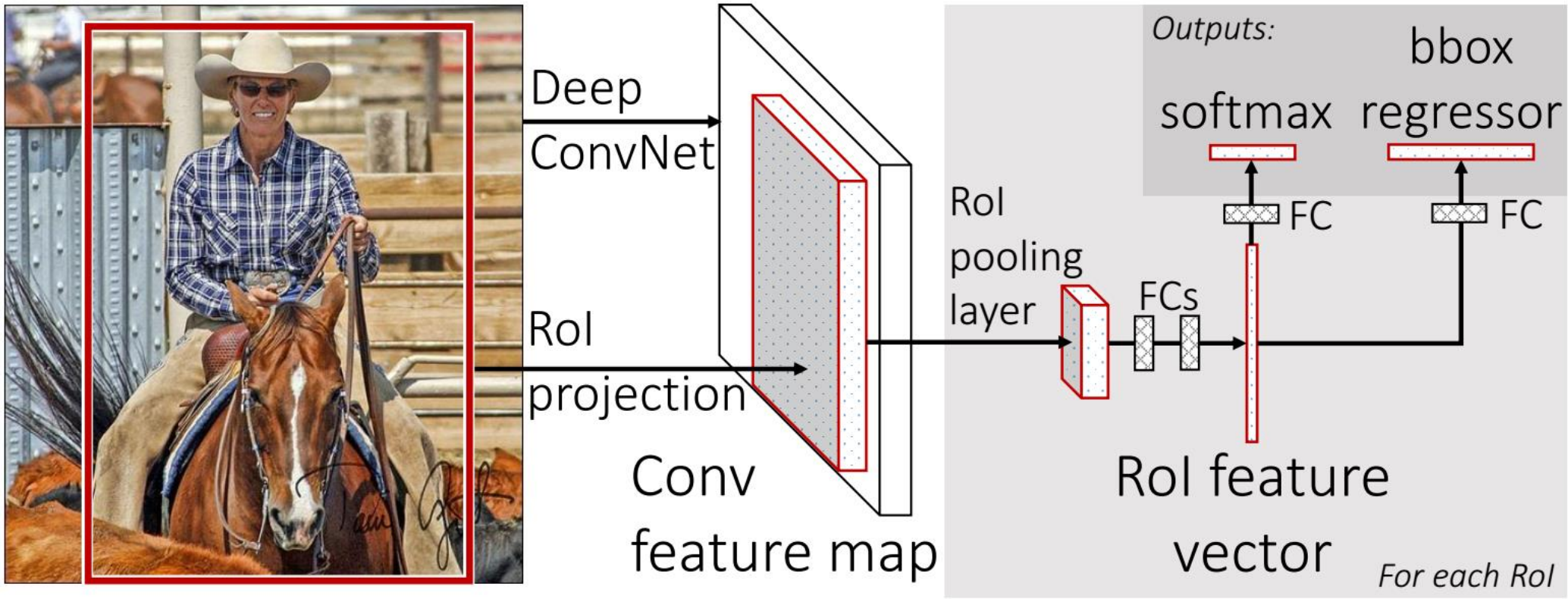




Combinatorial Bayesian Optimization using the Graph Cartesian Product, NeurIPS 2019, C. Oh, J. Tomczak, E. Gavves, M. Welling, NeurIPS 2019

Fast RCNN

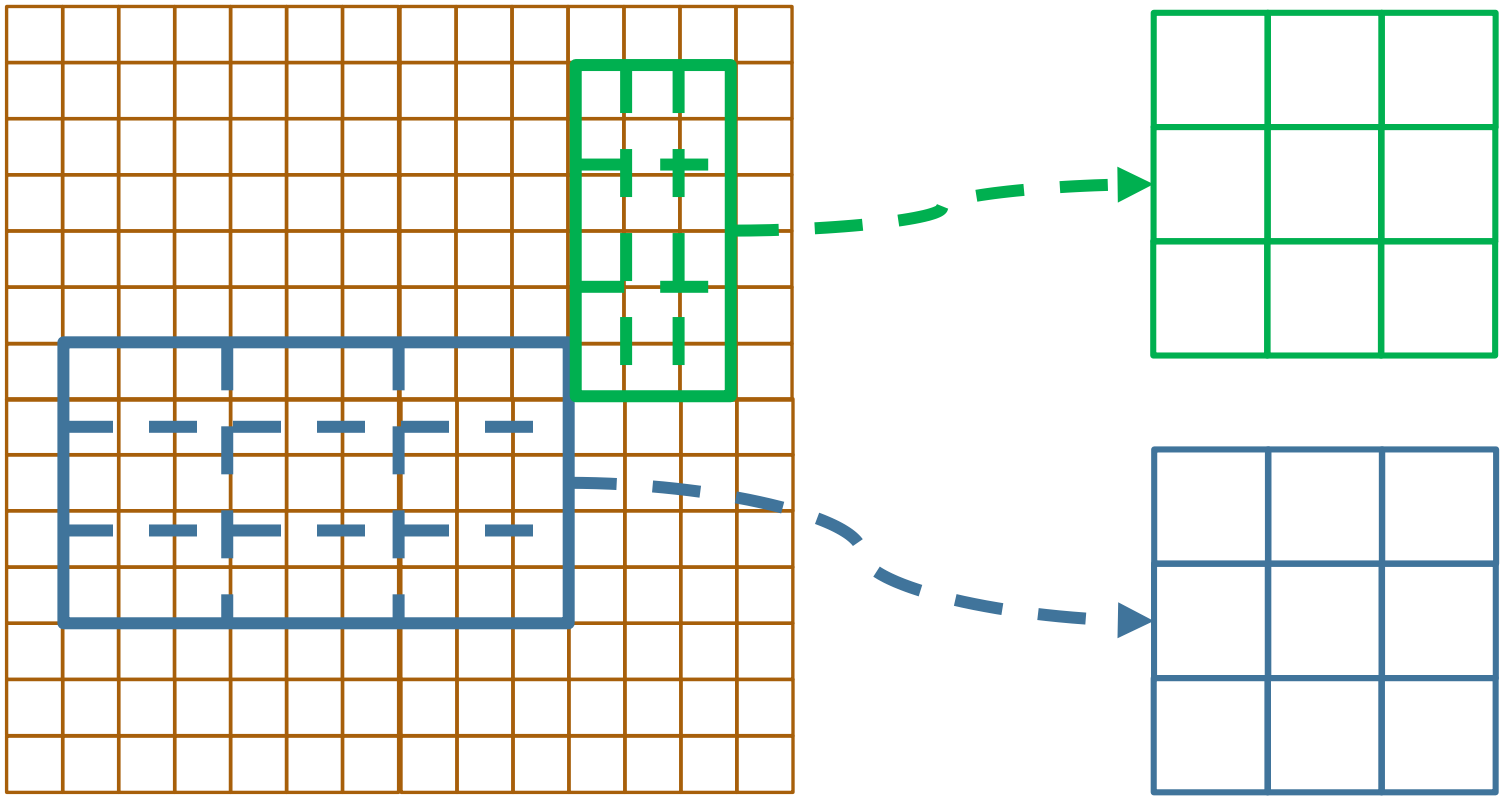
- Predictions from sliding windows on feature maps



He, Zhang, Ren, Sun, Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, 2014
Girshick, Fast R-CNN, 2015

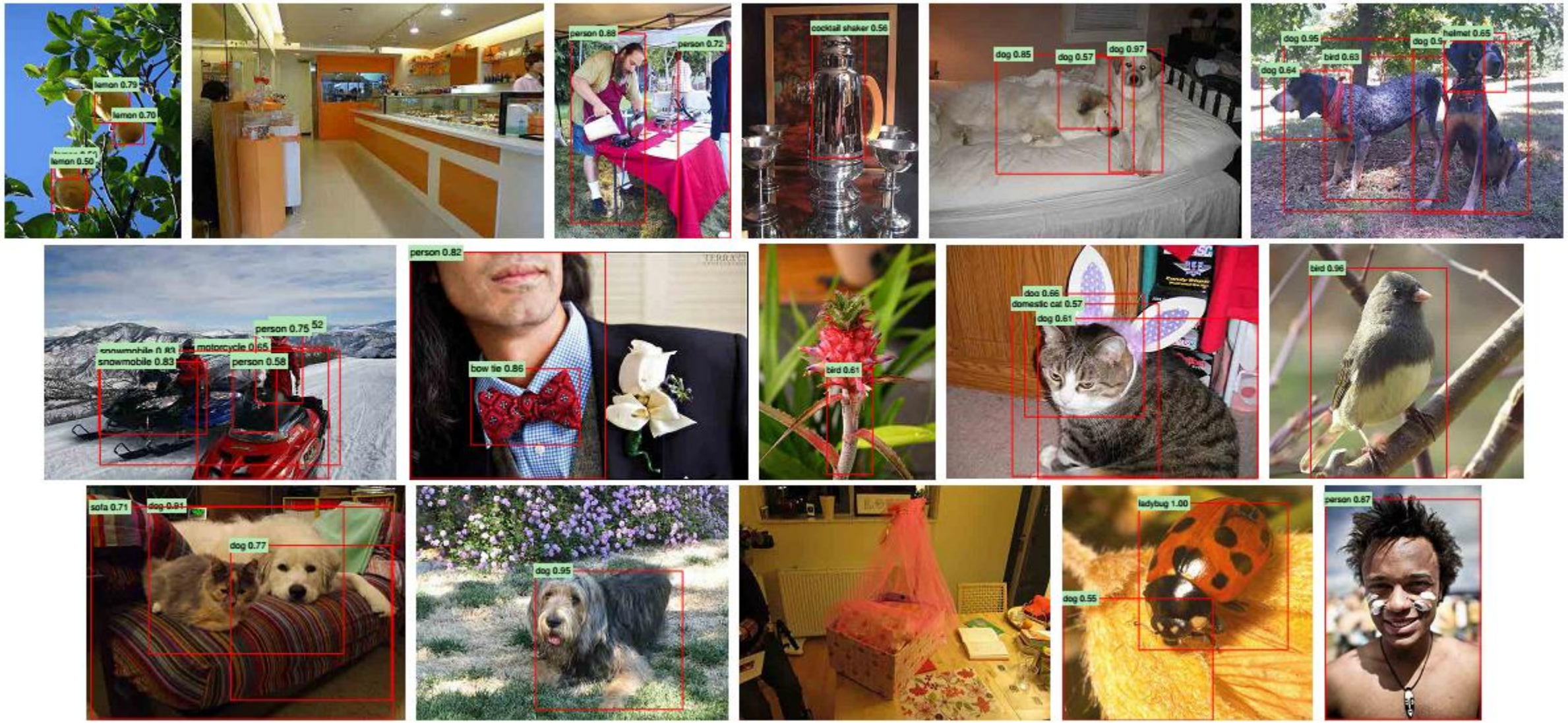
Region-of-Interest (ROI) Pooling

- Divide feature map in $T \times T$ cells
  - The cell size will change depending on the size of the candidate location



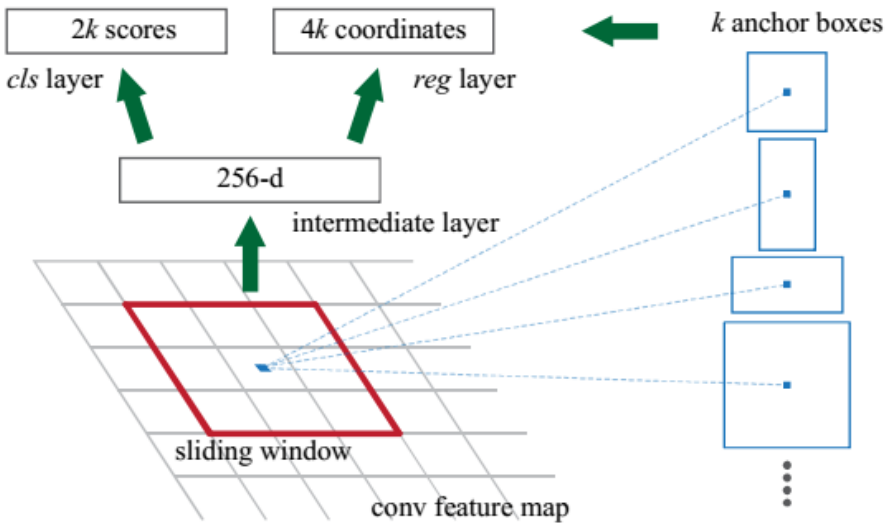
Always 3x3 no matter the size of candidate location

Some results



Faster R-CNN [Girshick2016]

- Generate also candidate locations



Region Proposal Network

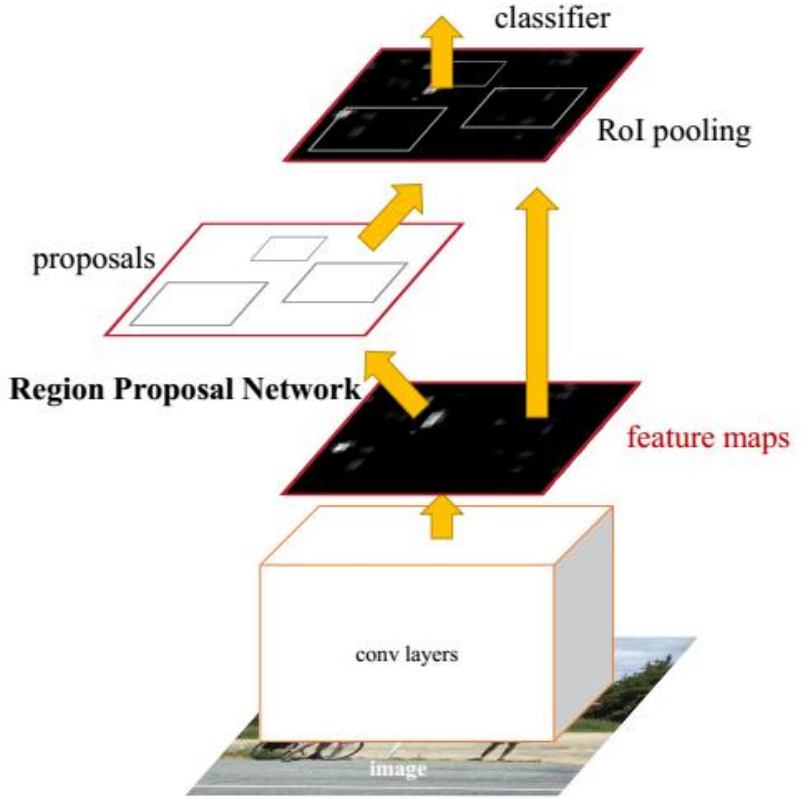
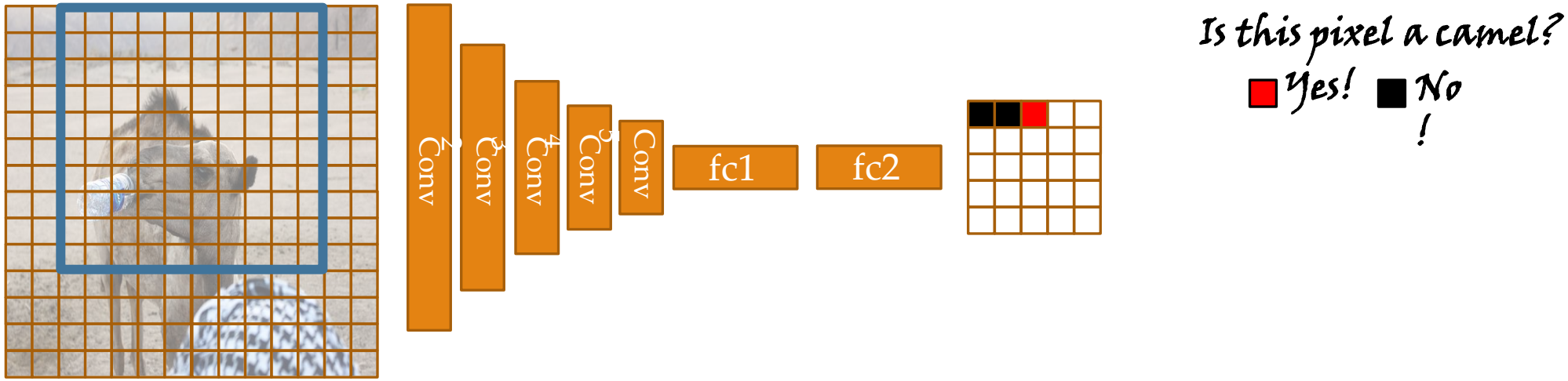


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Ren, He, Girshick, Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Fully Convolutional Networks

- Image larger than network input → slide the network

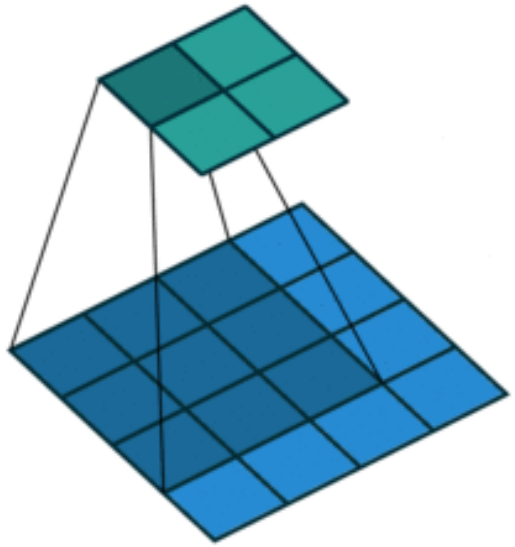


Long, Shelhamer, Darell, Fully Convolutional Networks for Semantic Segmentation, 2015

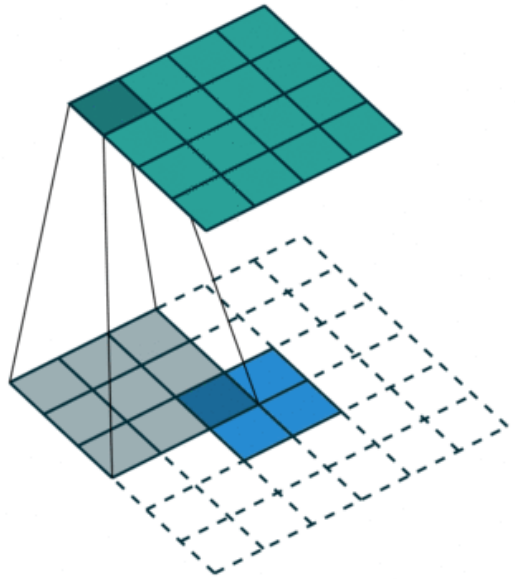
Deconvolutional modules

Output

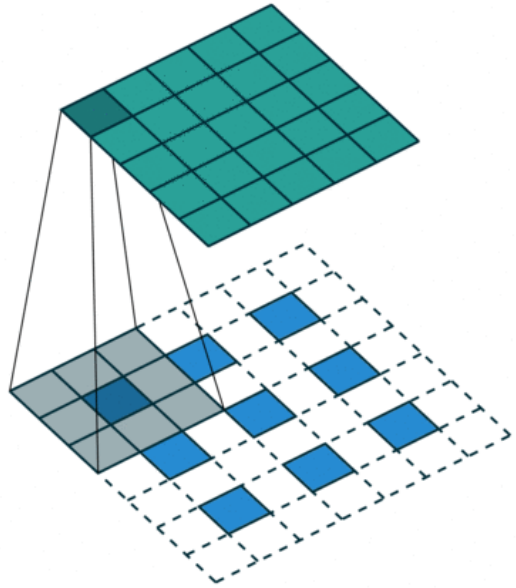
Input



Convolution
No padding, no strides



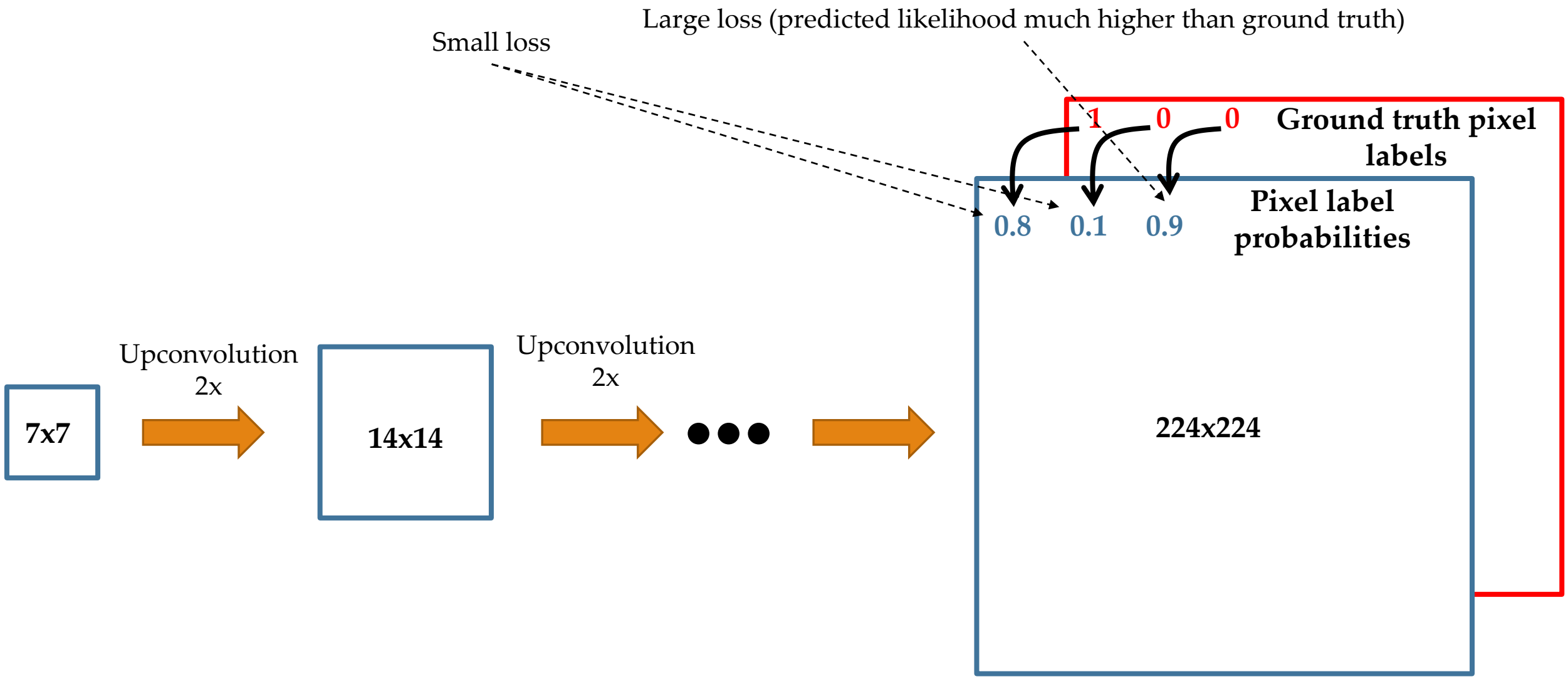
Upconvolution
Padding, no strides



Upconvolution
Padding, strides

More visualizations: https://github.com/vdumoulin/conv_arithmetic

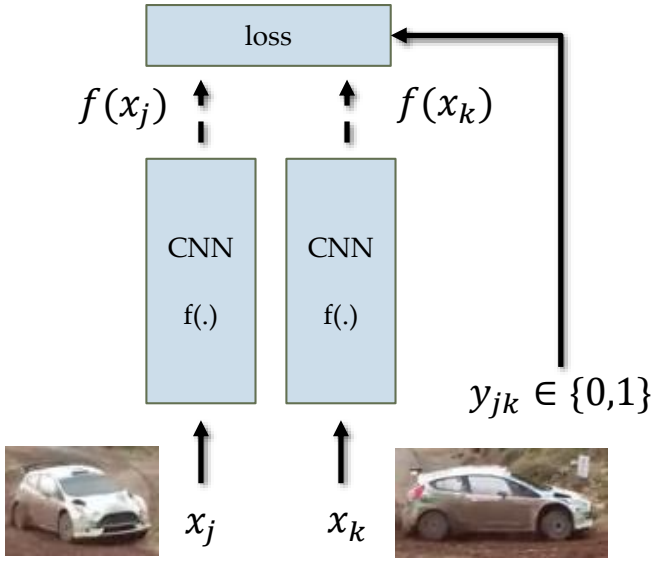
Upconvolving putput



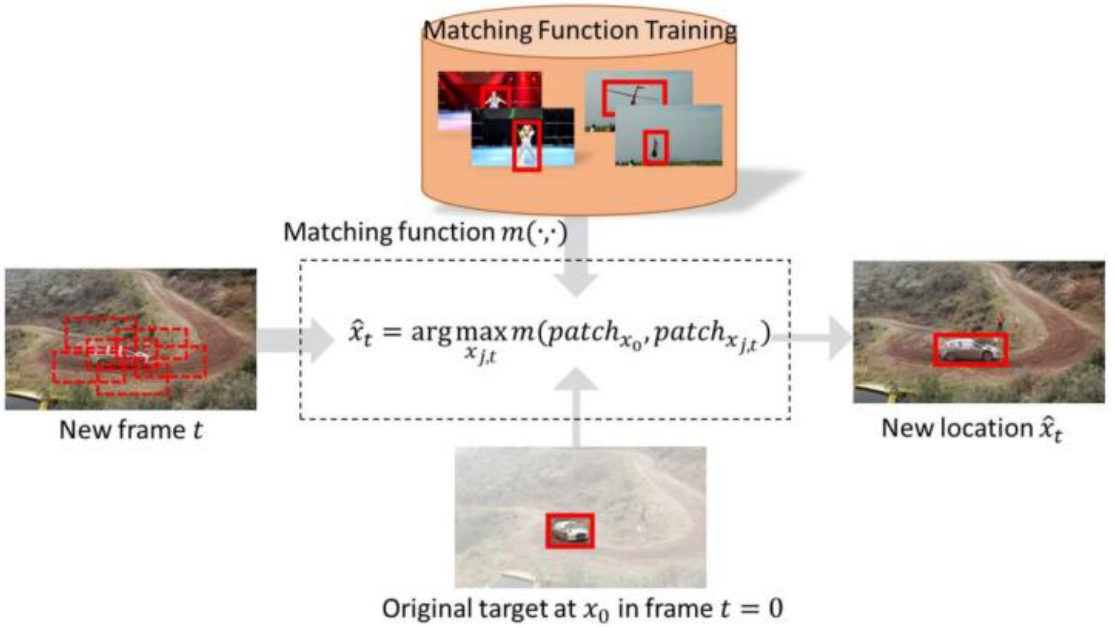
Siamese Networks for Tracking

- The backbone of modern trackers nowadays, especially for long-term tracking

Training



Test-time tracking



Marginal Contrastive Loss:

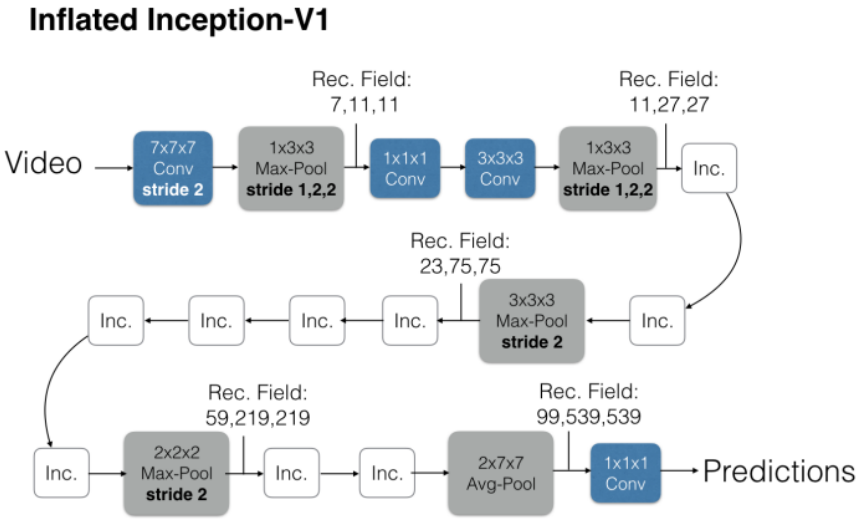
$$L(x_j, x_k, y_{jk}) = \frac{1}{2} y_{jk} D^2 + \frac{1}{2}(1 - y_{jk}) \max(0, \sigma - D)$$

$$D = \|f(x_j) - f(x_k)\|_2$$

Tao, Gavves, Smeulders, Siamese Instance Search for Tracking, CVPR, 2015

i3D

- i3D = C3D + Inception
  - Plus some neat tricks
- Take 2D filters and inflate them so that they become 3D filters
- Then, use them as initialization



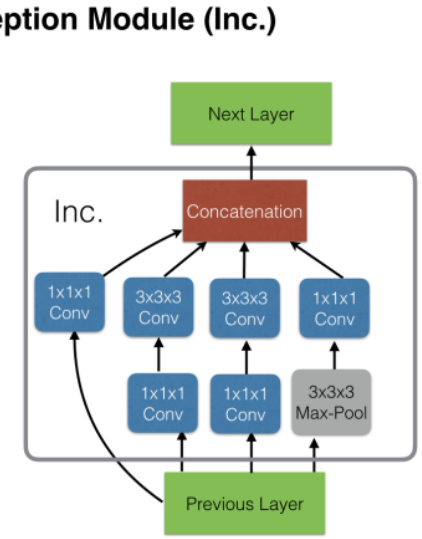


Figure 3. The Inflated Inception-V1 architecture (left) and its detailed inception submodule (right). The strides of convolution and pooling operators are 1 where not specified, and batch normalization layers, ReLu's and the softmax at the end are not shown. The theoretical sizes of receptive field sizes for a few layers in the network are provided in the format "time,x,y" – the units are frames and pixels. The predictions are obtained convolutionally in time and averaged.

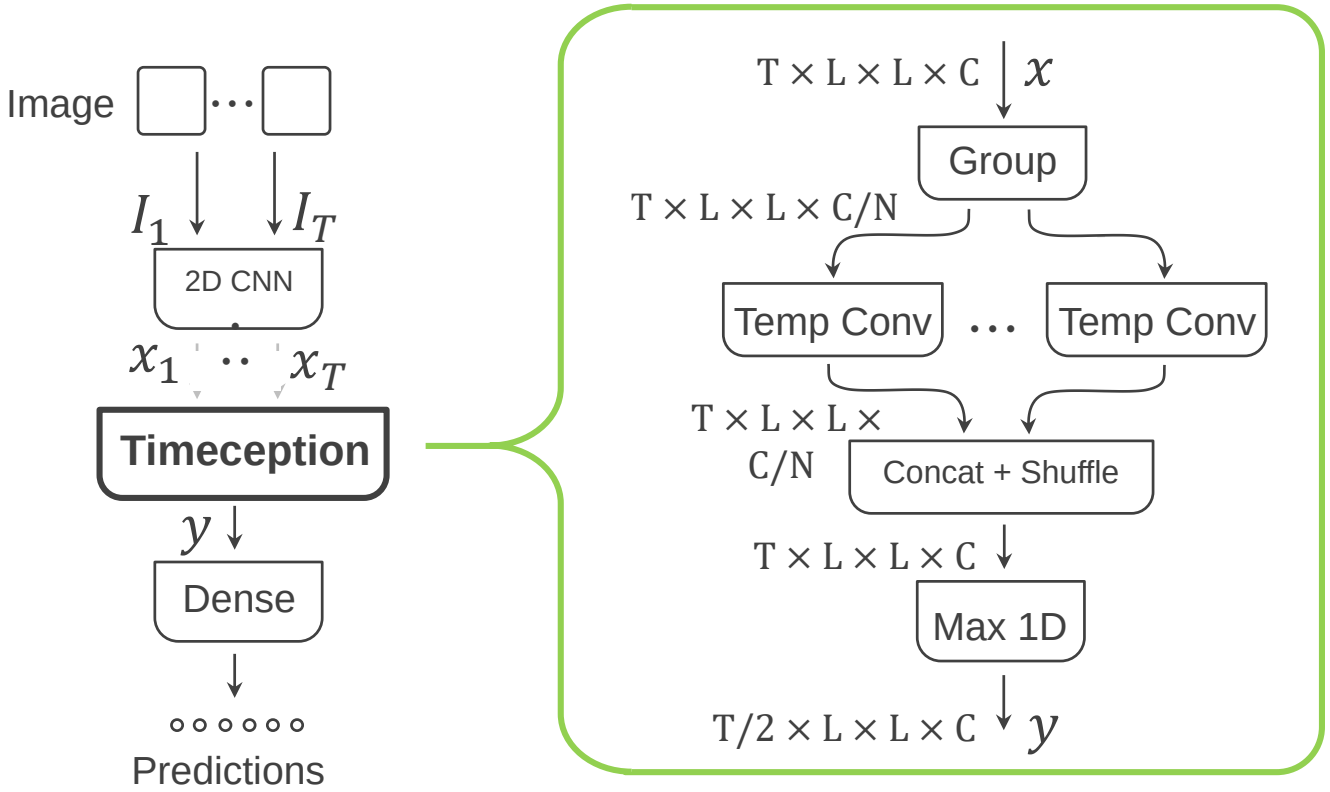
Architecture	UCF-101			HMDB-51			Kinetics		
	RGB	Flow	RGB + Flow	RGB	Flow	RGB + Flow	RGB	Flow	RGB + Flow
(a) LSTM	81.0	–	–	36.0	–	–	63.3	–	–
(b) 3D-ConvNet	51.6	–	–	24.3	–	–	56.1	–	–
(c) Two-Stream	83.6	85.6	91.2	43.2	56.3	58.3	62.2	52.4	65.6
(d) 3D-Fused	83.2	85.8	89.3	49.2	55.5	56.8	–	–	67.2
(e) Two-Stream I3D	**84.5**	**90.6**	**93.4**	**49.8**	**61.9**	**66.4**	**71.1**	**63.4**	**74.2**

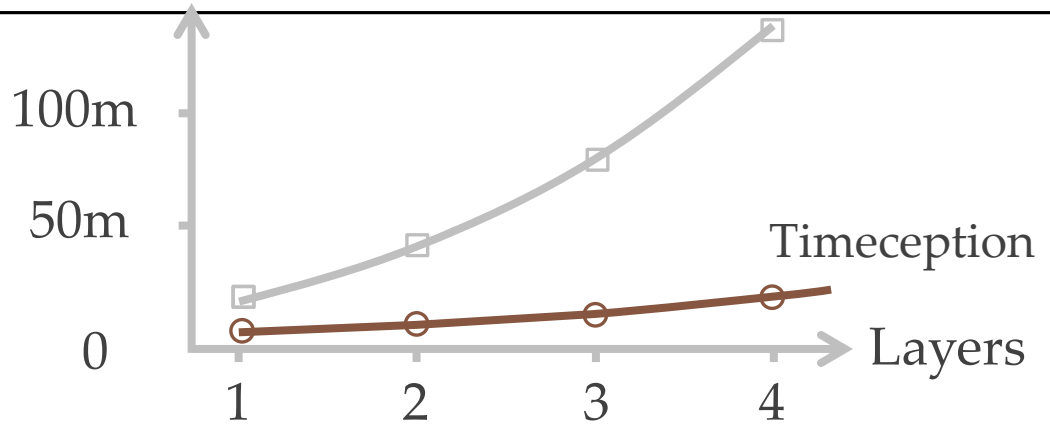
Table 2. Architecture comparison: (left) training and testing on split 1 of UCF-101; (middle) training and testing on split 1 of HMDB-51; (right) training and testing on Kinetics. All models are based on ImageNet pre-trained Inception-v1, except 3D-ConvNet, a C3D-like [31] model which has a custom architecture and was trained here from scratch. Note that the Two-Stream architecture numbers on individual RGB and Flow streams can be interpreted as a simple baseline which applies a ConvNet independently on 25 uniformly sampled frames then averages the predictions.

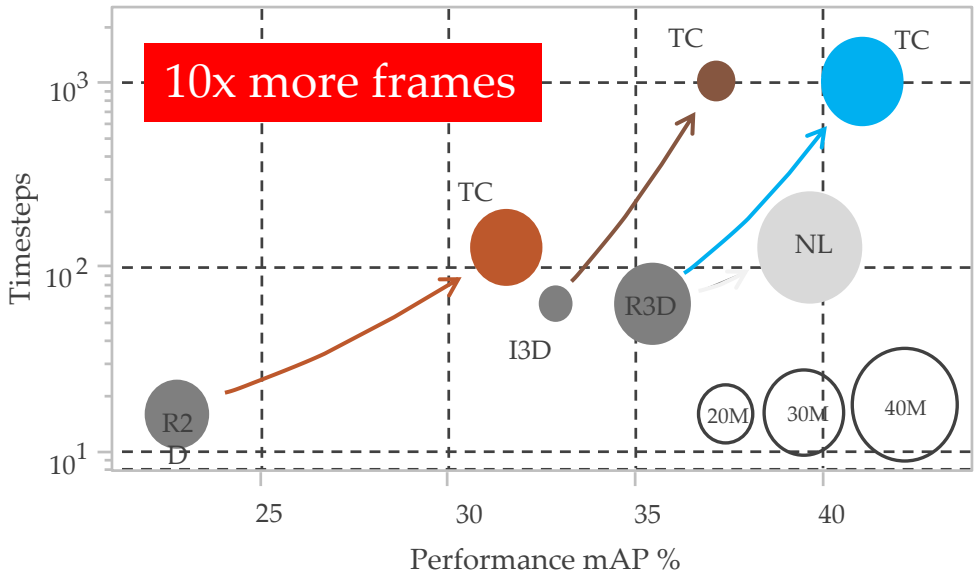
Carreira, Zisserman, Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset, CVPR, 2017

Timeception

- Decompose space-time in long temporal convolutions







Hussein, Gavves, Smeulders, Timeception for complex action recognition, CVPR, 2019

Summary

- Case study II: VGG
- Vanishing and exploding gradients
- Case study III: Inception
- Case study IV: Resnet, Denset, Highway Net
- Depth, trainability, generalization
- Specialized architectures

Reading material

- All the papers from the models presented