Computer Vision and CNN

E. Fatemizadeh

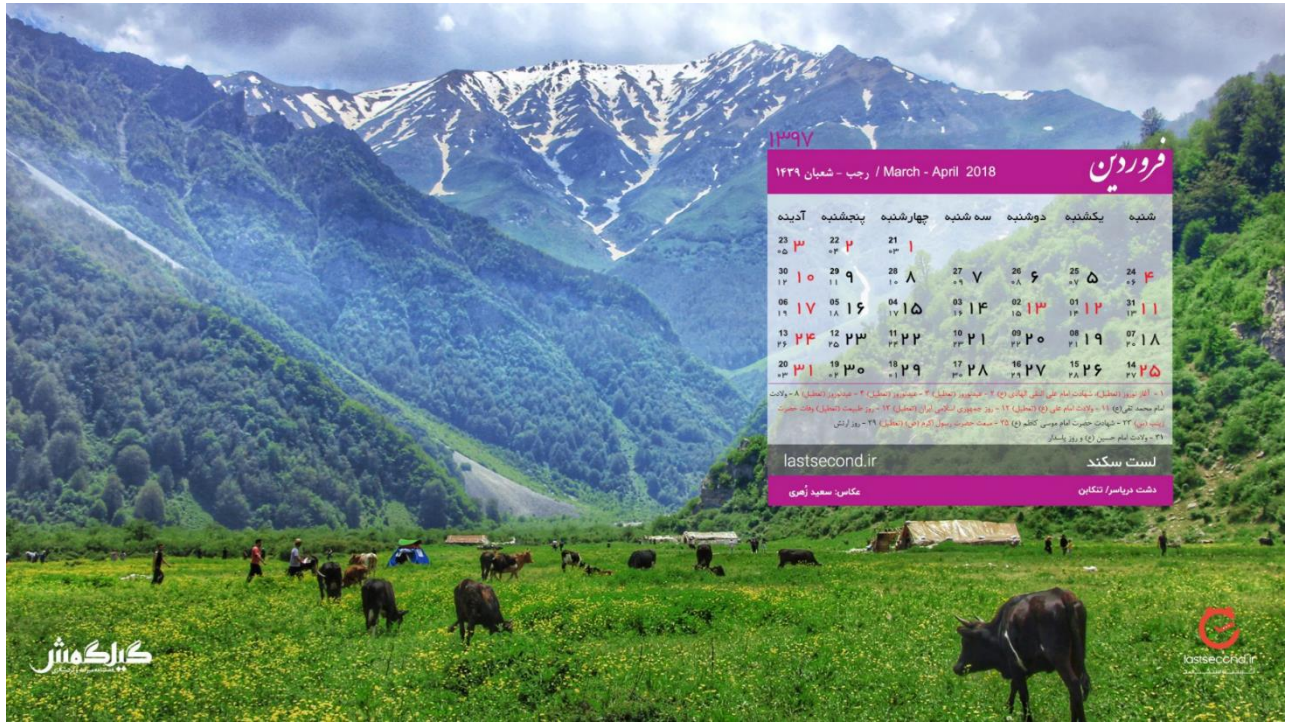
Fall 2020

Computer Vision Problems

- Classification
- Classification + Localization
- Object Detection
- Instance Segmentation

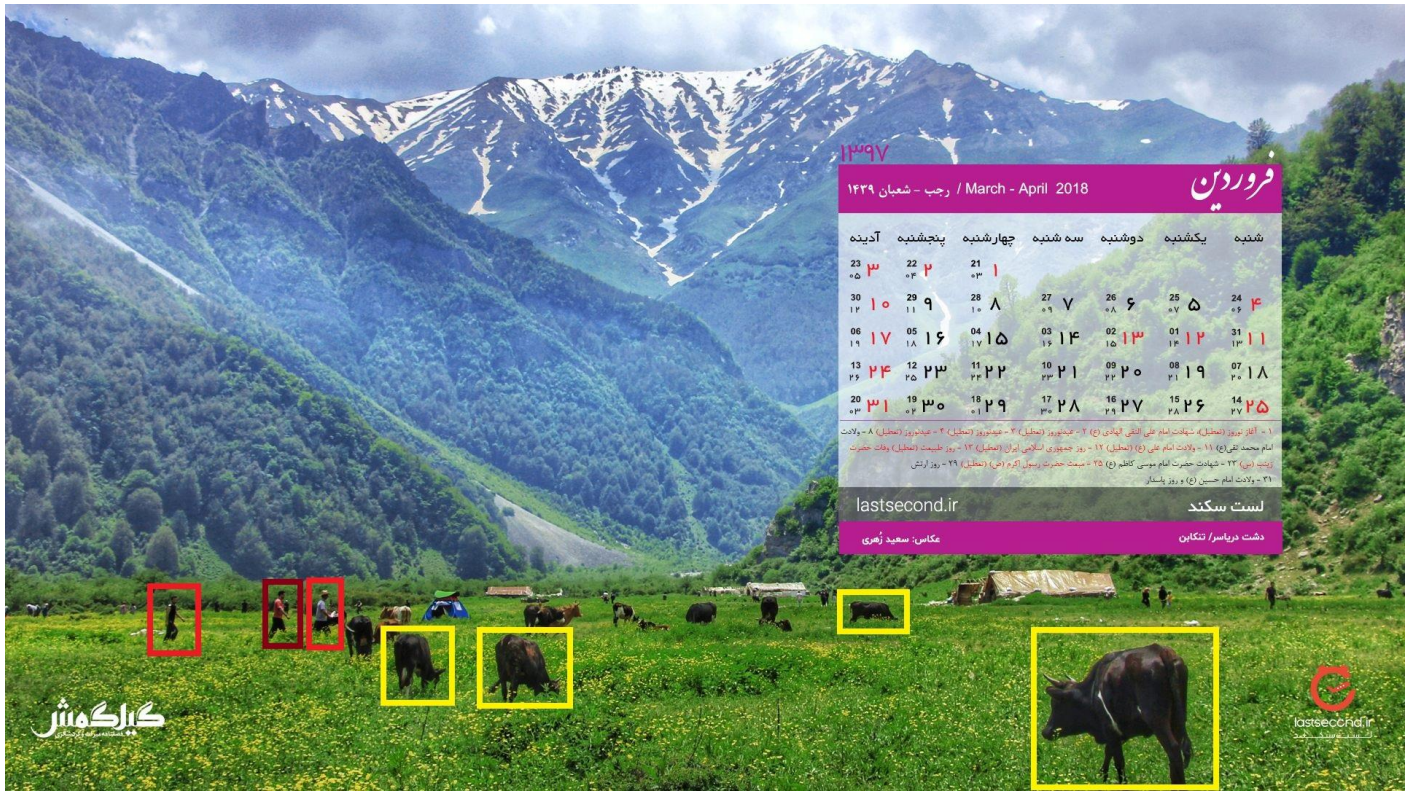
Classification

- Output:
  - Single/Multiple Label
  - Nature/Farm/Mountain…



Object Detection

- Output:
  - Bounding Box(es)
    - 4-touple(s)
  - Cow#n+Man#m

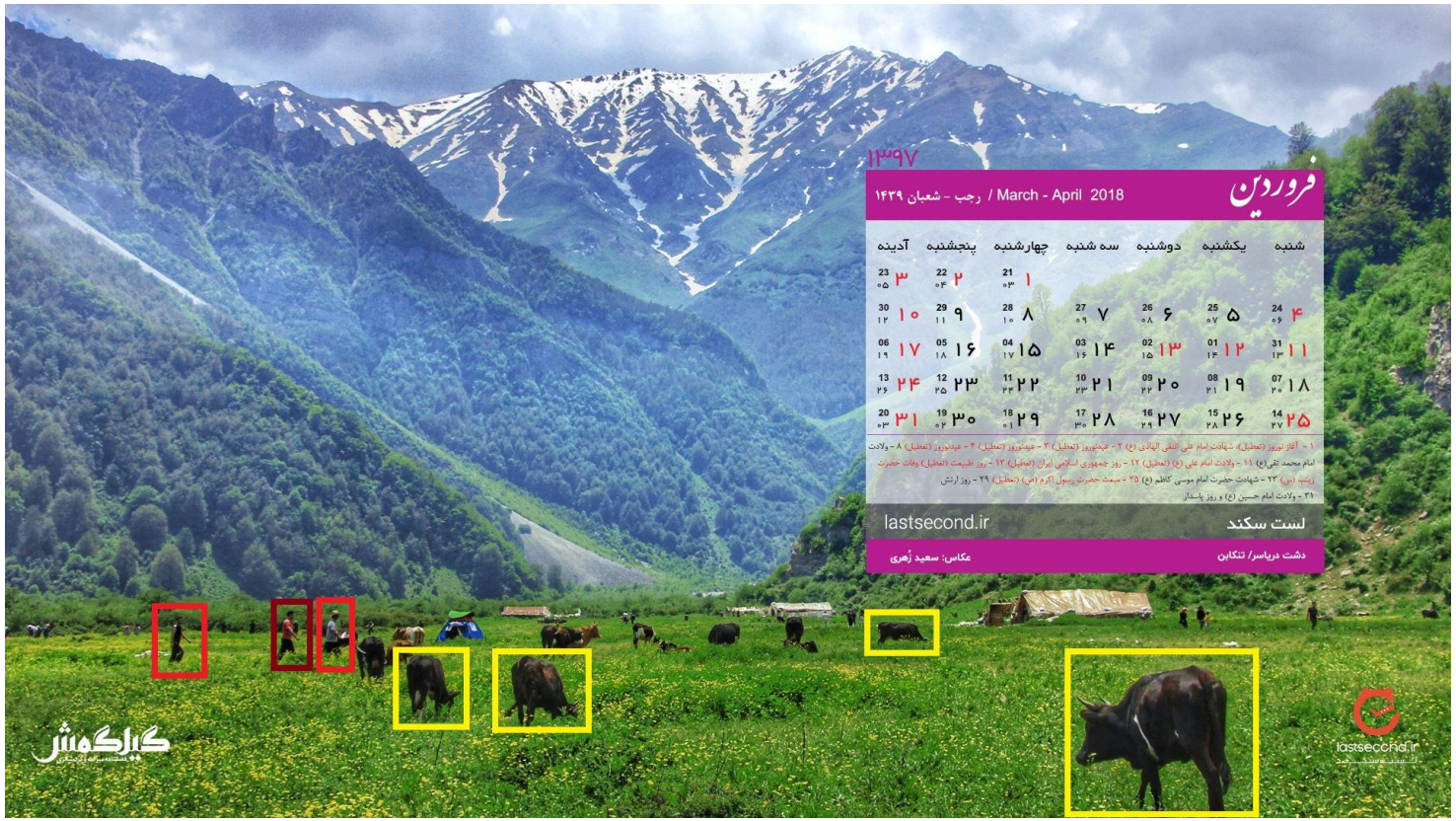


Segmentation- Pixel Classification

- Output:
  - A group of pixels

Classifiction+Localization

- Output:
  - Label
  - Bounding Box



Object Detection/Localization Applications!

- Robotics
- Self-Driving Cars
- Better Classification (True Label)
- CBIR (Content Based Image Retrieval)
- Medical Diagnosis
- …

Object Detection – General Approach

- Training:
  - Handy Crafted Bounding Box Selection
  - Feature Extraction
  - Train a Classifier (SVM) for each category
- Test:
  - Select Possible Bounding Boxes (sliding windows/selective search/….)
  - Crop each one
  - Assign score (for each category) for all box
  - Keep boxes with score more than a threshold
  - Success: > 50% overlap with ground truth

Faster R-CNN

Microsoft

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015

Faster R-CNN **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015**

› Main Idea: Region Proposal Network (RPN)

Bunch of bounding box that will be scored by a classifier and regressor to check the occurrence of objects

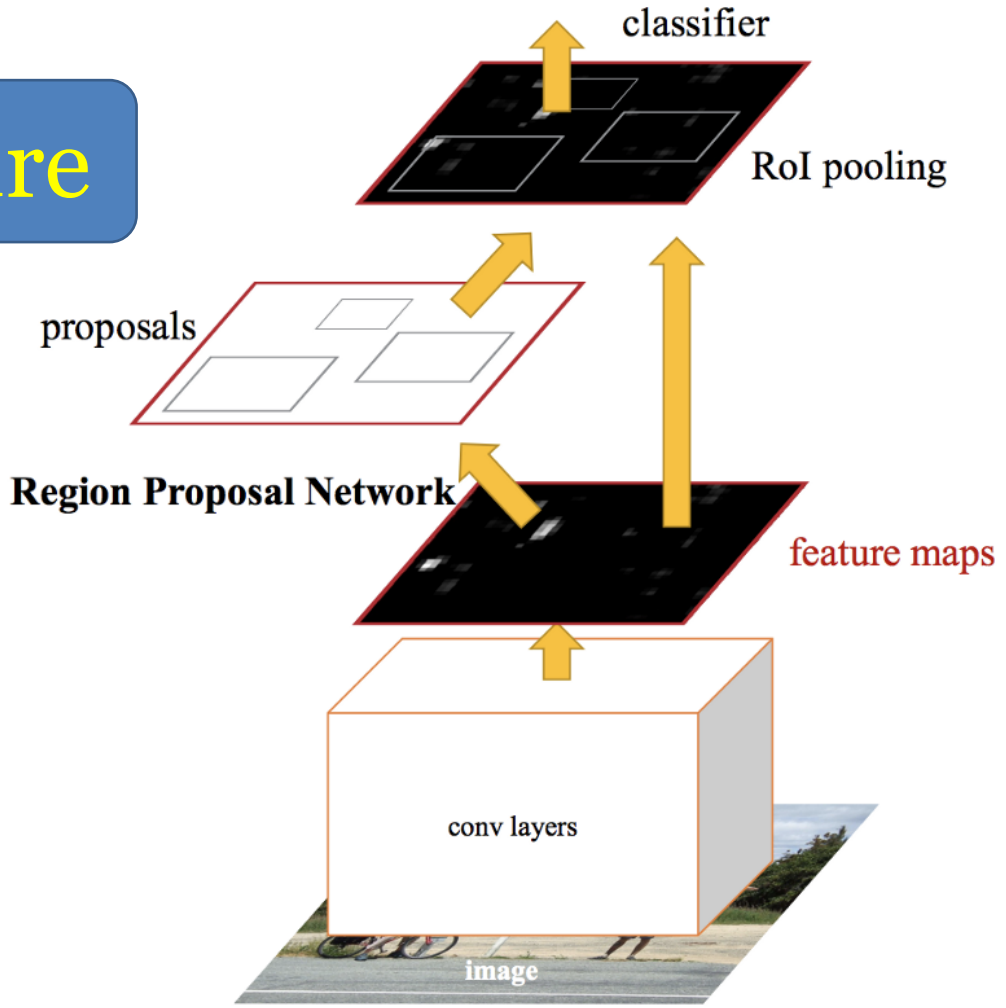
› More precise:

RPN **predicts** the possibility of an **anchor** being background or foreground, and refine the anchor.

Faster R-CNN

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015

Architecture



Faster R-CNN Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015

- Region Proposal Network (RPN):
  - RPN has a classifier and a regressor
- RPN Classifier (two classes):
  - Determine the probability of a proposal having the target object (LogLoss)
- RPN Regressor (4 outputs):
  - Regresses the coordinates of the proposals.

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

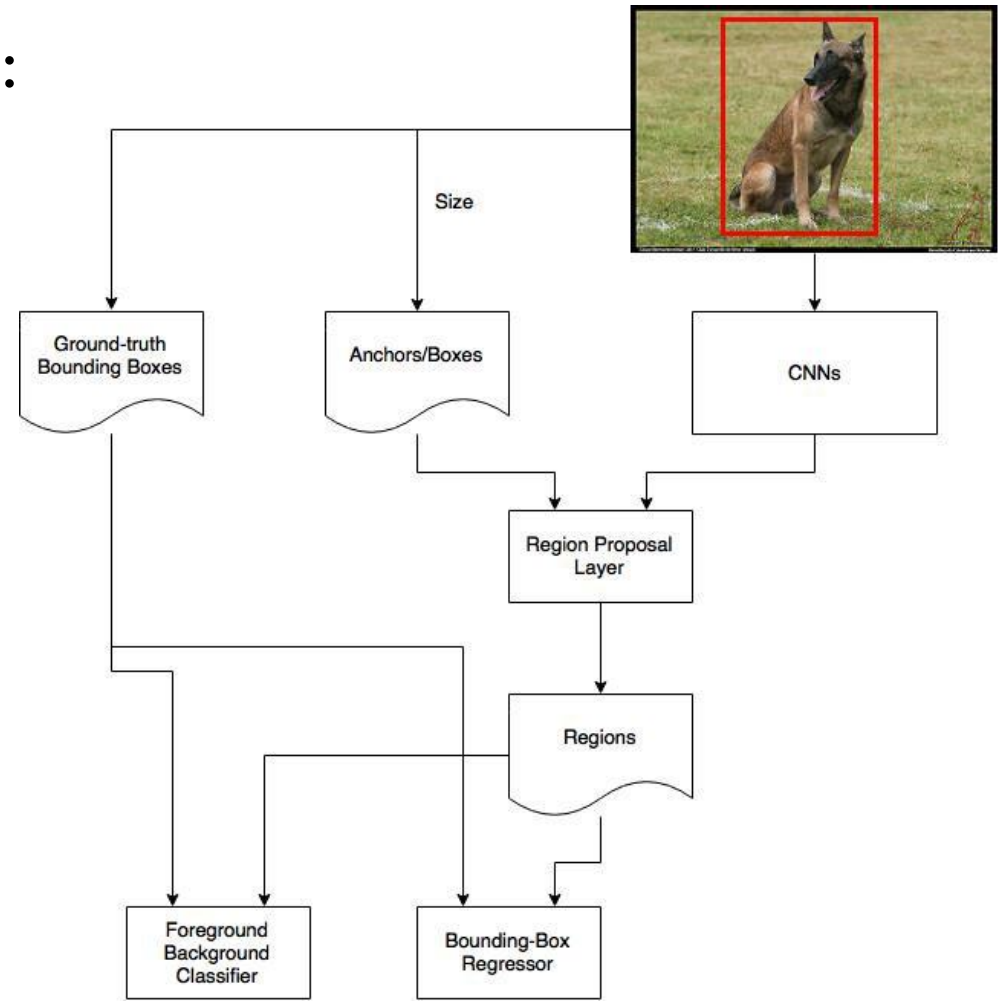
in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

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

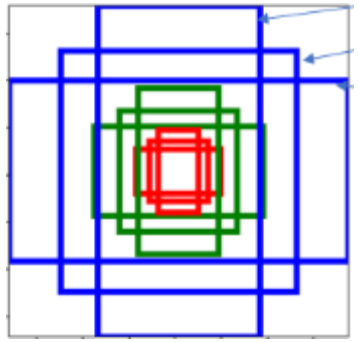
Faster R-CNN Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015

› Region Proposal Network (RPN):



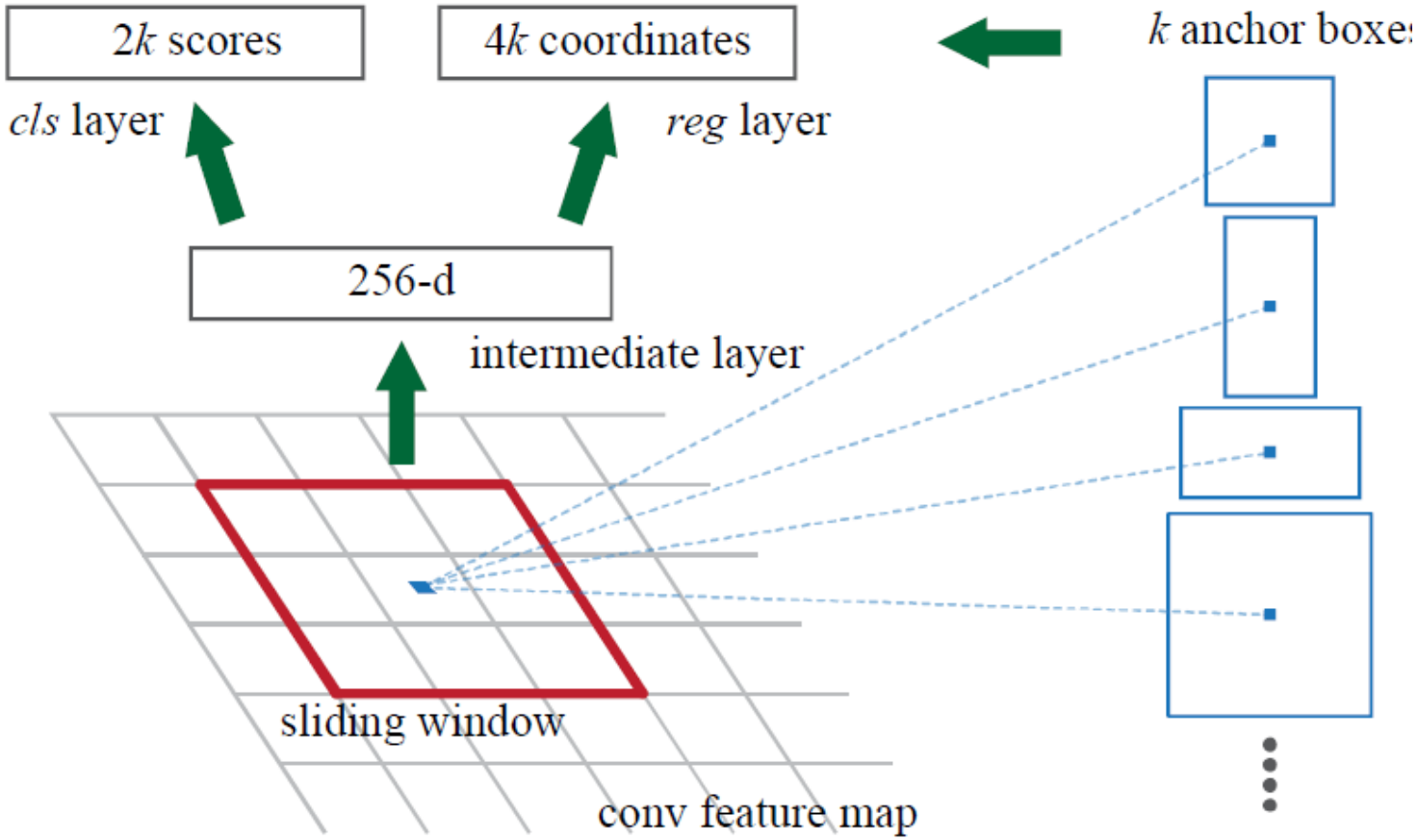
Faster R-CNN Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015

- RPN:
  - Anchor Generator Layer produce a set of bounding box (3 size, 3 aspect ratio):
  - At each position, one of these (9 anchor) is better than others!
  - Aspect ration: {0.5, 1, 2}
  - Sliding Stride: 16 (in image plane)
  - Anchor Scales: {8, 16, 32}, in feature map plane



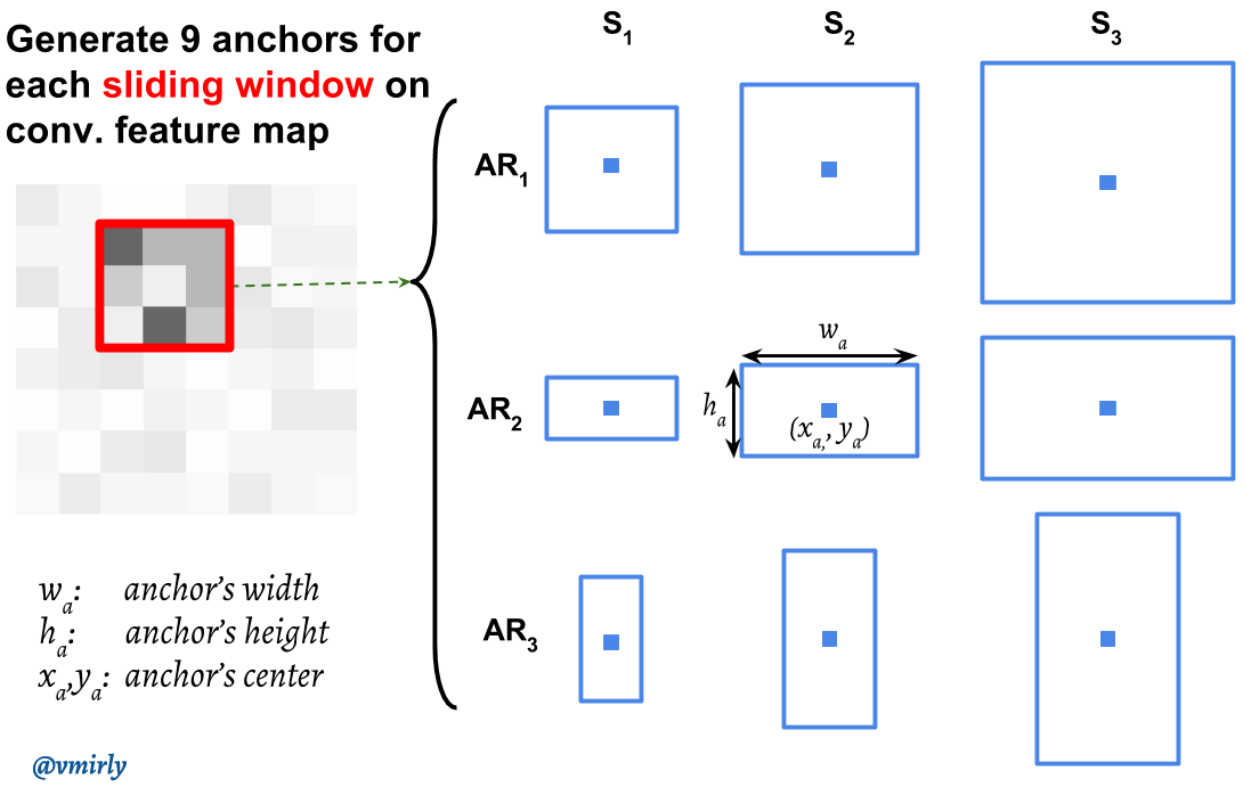
Faster R-CNN **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015**

› RPN training (k=9):



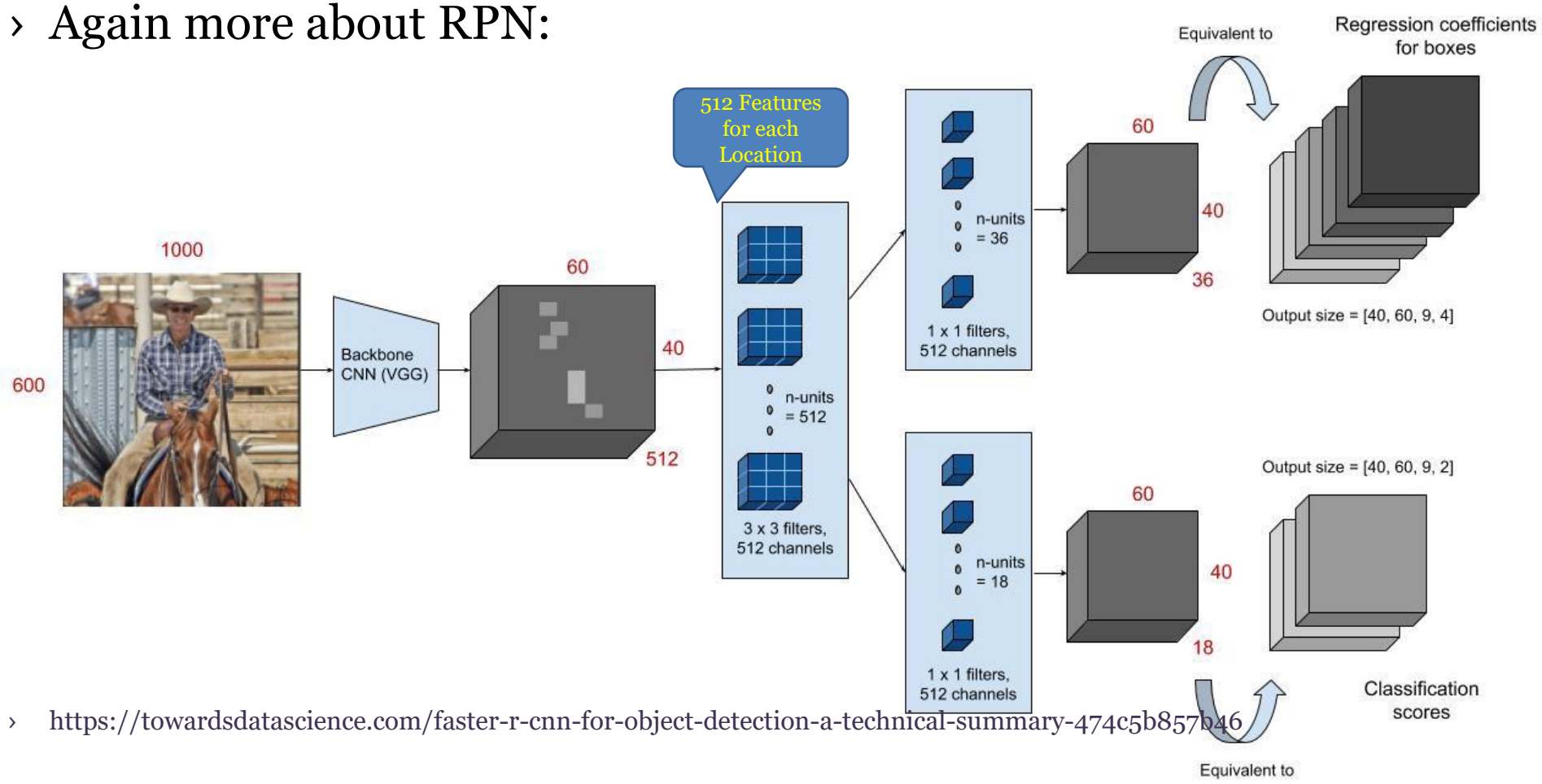
Faster R-CNN Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015

- More About RPN
- Anchor are possible objects position!



Faster R-CNN Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015

› Again more about RPN:



› https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46

Faster R-CNN **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015**

- Training (RPN):
  - 40*60*9 = 21600 anchors in total
  - Ignore anchors that cross the boundary (20k→6K)
  - Positive anchor: IoU (Intersection-Over-Union), $\frac{\text{True}\cap\text{Estimated}}{\text{True}\cup\text{Estimated}} > 0.7$
  - Negative anchor: IoU (Intersection-Over-Union), $\frac{\text{True}\cap\text{Estimated}}{\text{True}\cup\text{Estimated}} < 0.3$
  - The remaining anchors (neither Positive nor Negative) are discarded
  - Each minibatch of size 256: 128 Positive and 128 Negative from same images
  - The regression loss is activated iff the anchor actually contains an object
  - All k (=9) anchor have independent (not weight sharing) regressor

Faster R-CNN **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015**

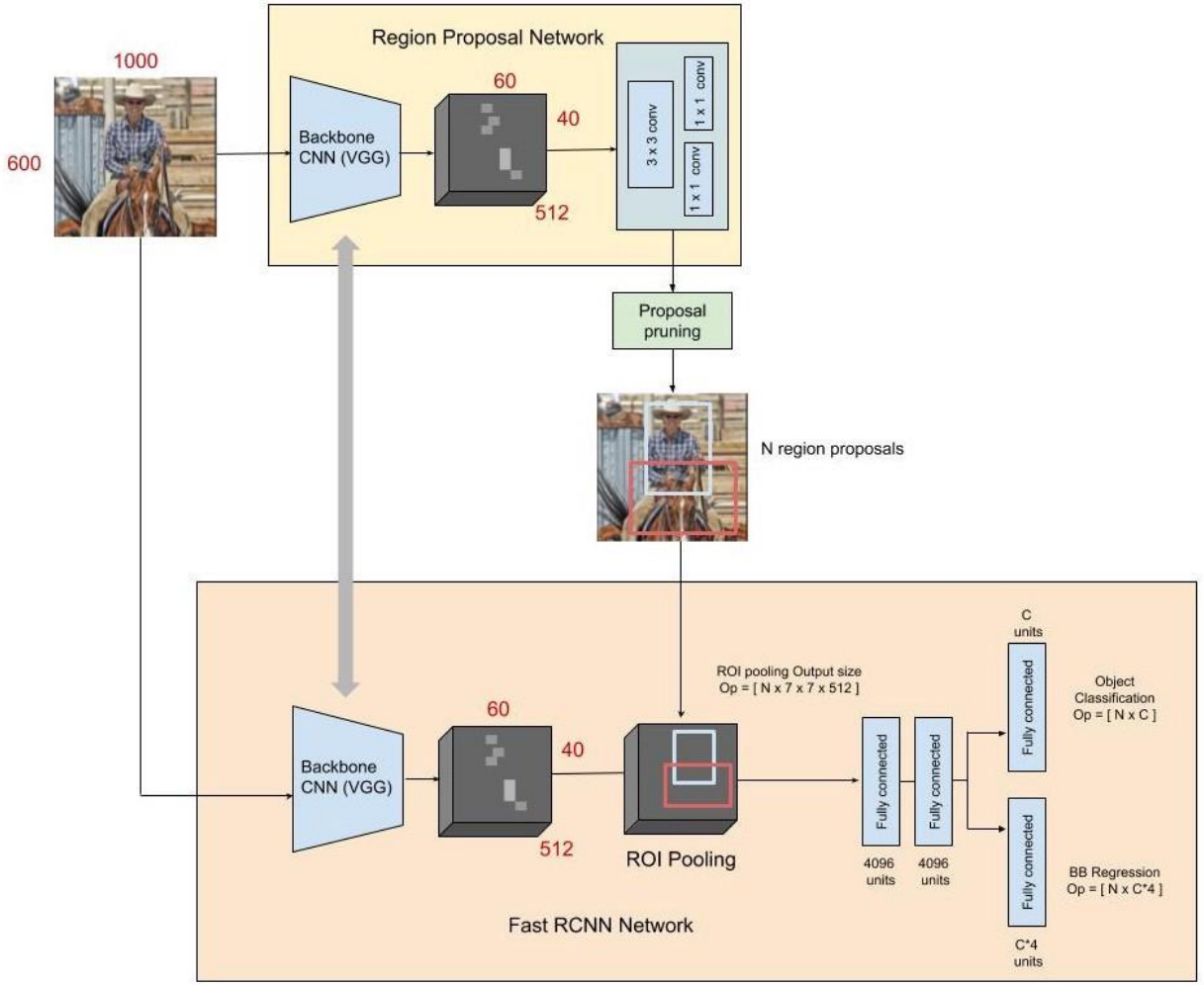
› Test (RPN):
- All 21600 anchors are used and passed to RPN network
- The regressors, fine tune localization
- All the region are sorted according to their *class* probability (P_i)
- Discard low probability boxes (<0.7)

Faster R-CNN

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networ

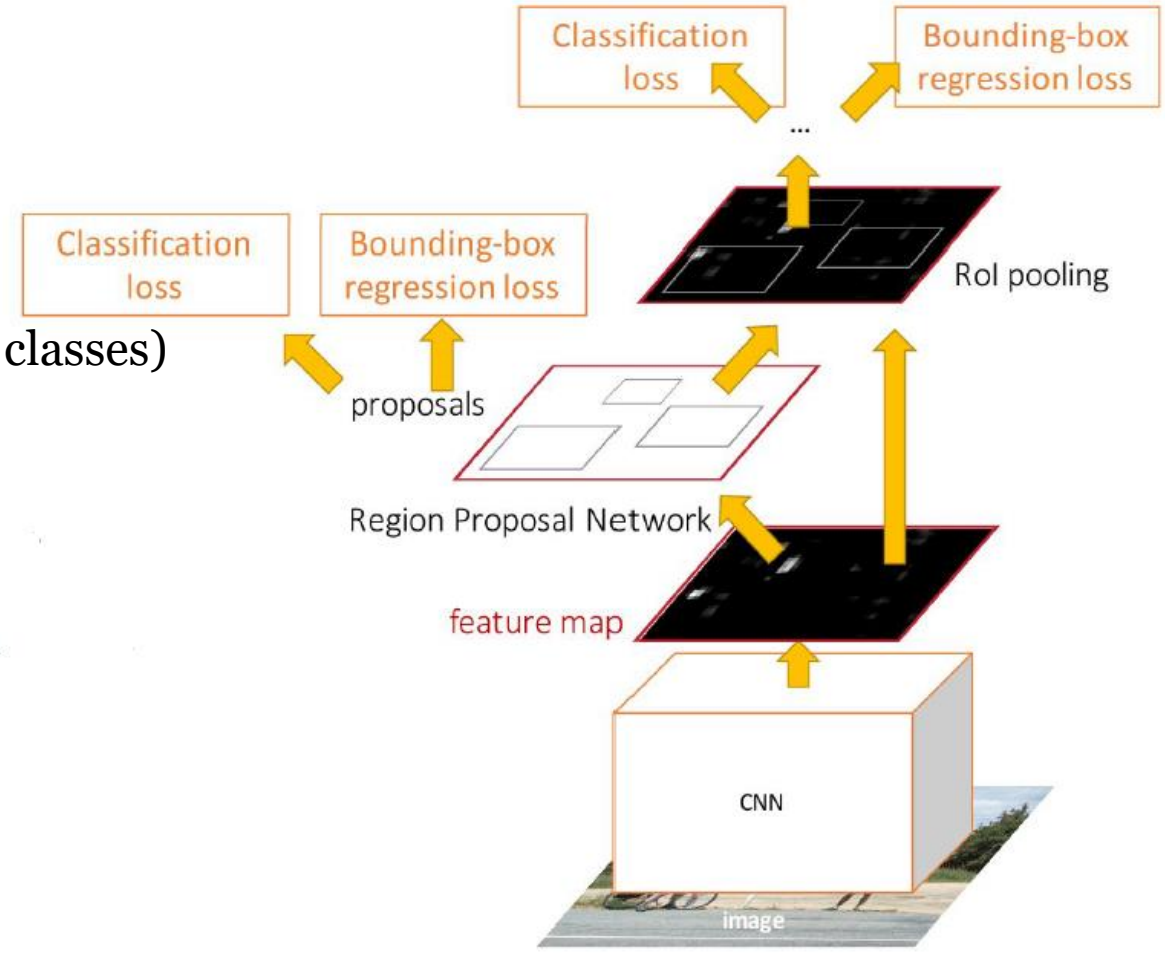
› Object detection:

› RPN + Fast R-CNN



Faster R-CNN **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS2015**

- Stage (option #2):
  - Jointly train with 4 losses:
    - RPN classify object / not object
    - RPN regress box coordinates
    - Final classification score (object classes)
    - Final box coordinates



Two New Operation

- You know upsampling in DSP! (↑):
  - Resampling and Interpolation
  - Fixed functionality and not-learnable
- New Operation:
  - Unpooling (Max Unpooling)
  - Deconvolution

Unpooling

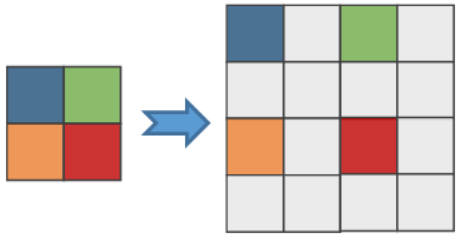
› Unpooling:

Approximate inverse of Max pooling

› A simple approach:

– Max unpooling:

2x2 unpooling:

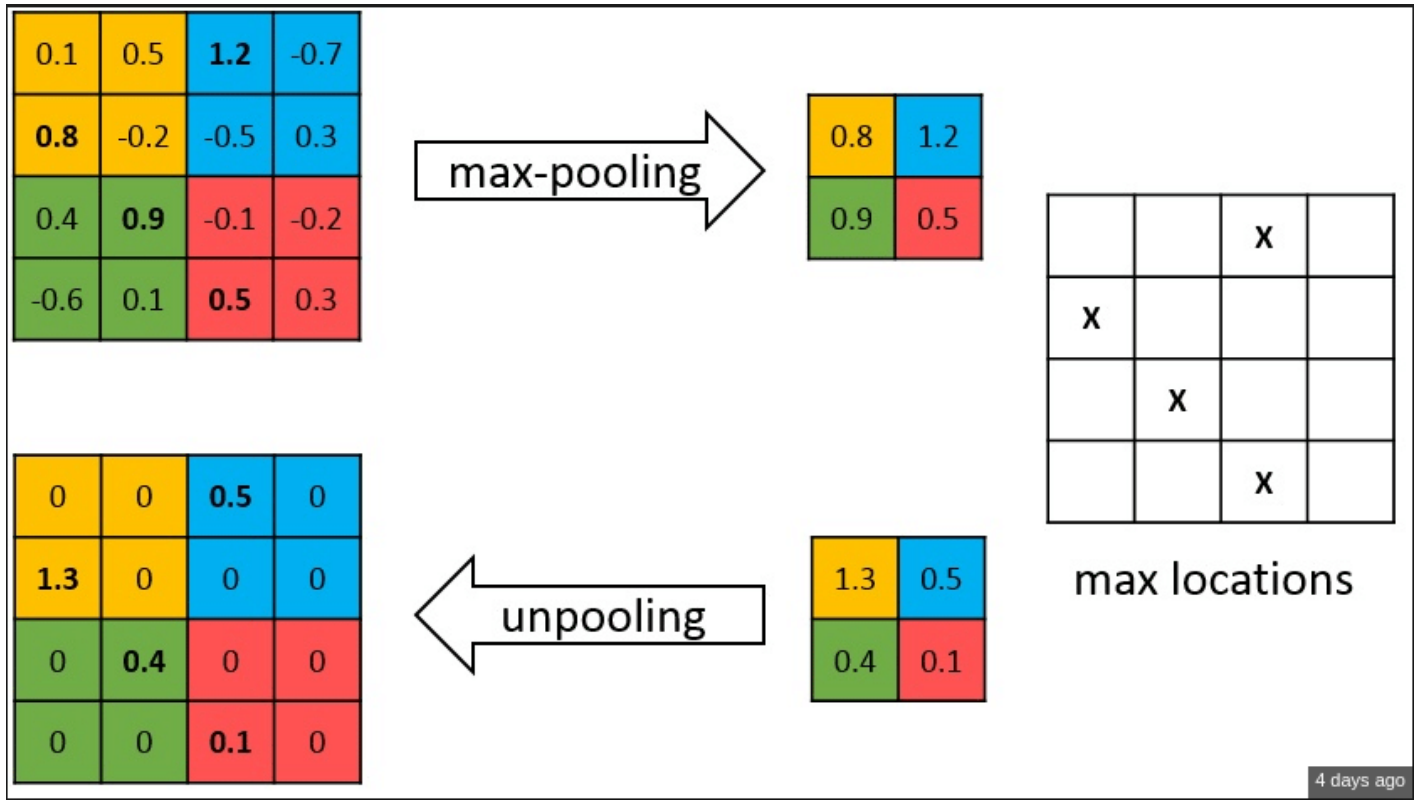


– Mean/Nearest Neighbor unpooling:

$$\begin{bmatrix} 2 & 4 \\ 1 & 6 \end{bmatrix} \Rightarrow \begin{bmatrix} 0.50 & 0.5 & 1 & 1 \\ 0.5 & 0.5 & 1 & 1 \\ 0.25 & 0.25 & 1.5 & 1.5 \\ 0.25 & 0.25 & 1.5 & 1.5 \end{bmatrix} \quad or \quad \begin{bmatrix} 2 & 2 & 4 & 4 \\ 2 & 2 & 4 & 4 \\ 1 & 1 & 6 & 6 \\ 1 & 1 & 6 & 6 \end{bmatrix}$$

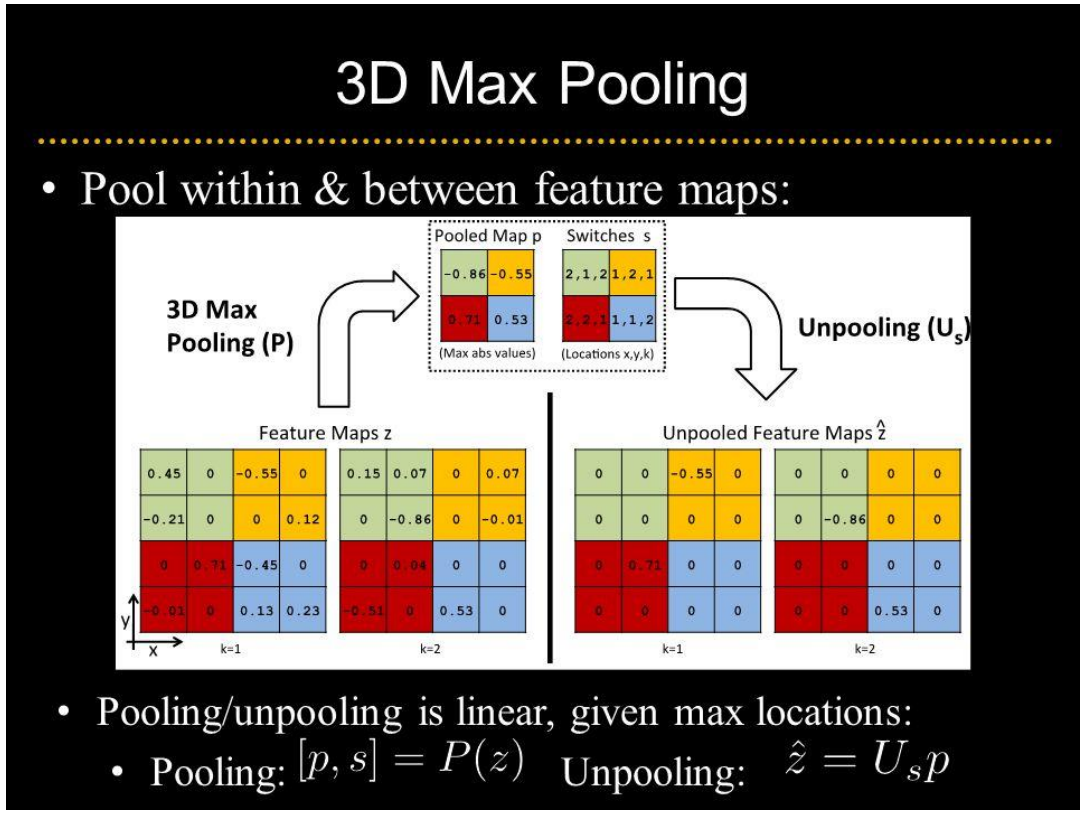
Unpooling -https://github.com/tiny-dnn/tiny-dnn/issues/612

› More popular approach: Max Unpooling (using index/switch map):



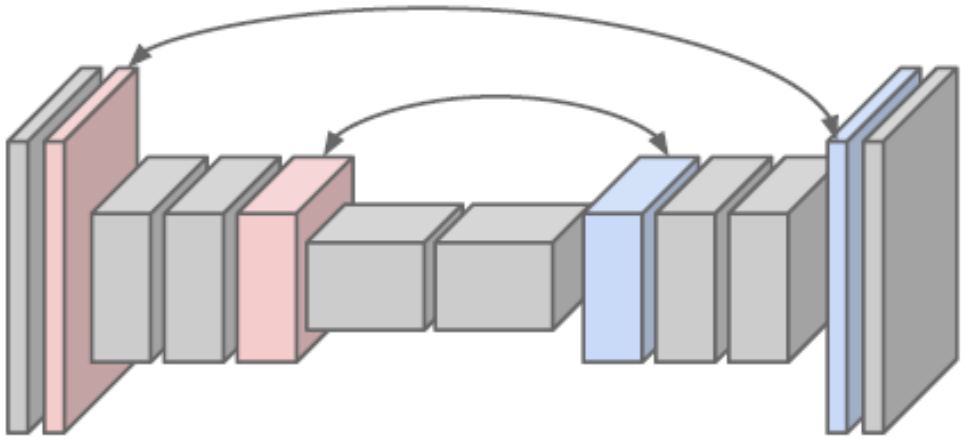
Unpooling -https://github.com/tiny-dnn/tiny-dnn/issues/612

› More popular approach: Max Unpooling (using index/switch map):



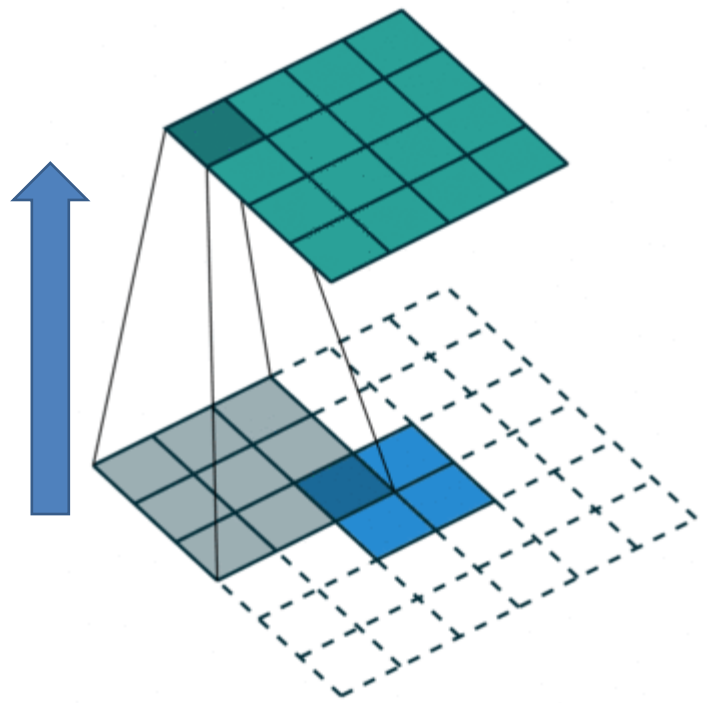
Unpooling

- Embed in a CNN
  - Create Corresponding Pooling and Unpooling!



Deconvolution (☹)/*Transpose Convolution* (☺)

- Deconvolution is not suitable name, confusion for DSP man!
- From 2×2 to 4×4 via filter



Transpose Convolution

- Other names:
  - Deconvolution
  - Upconvolution
  - Fractionally Strided convolution
  - Backward strided Convolution

Transpose Convolution

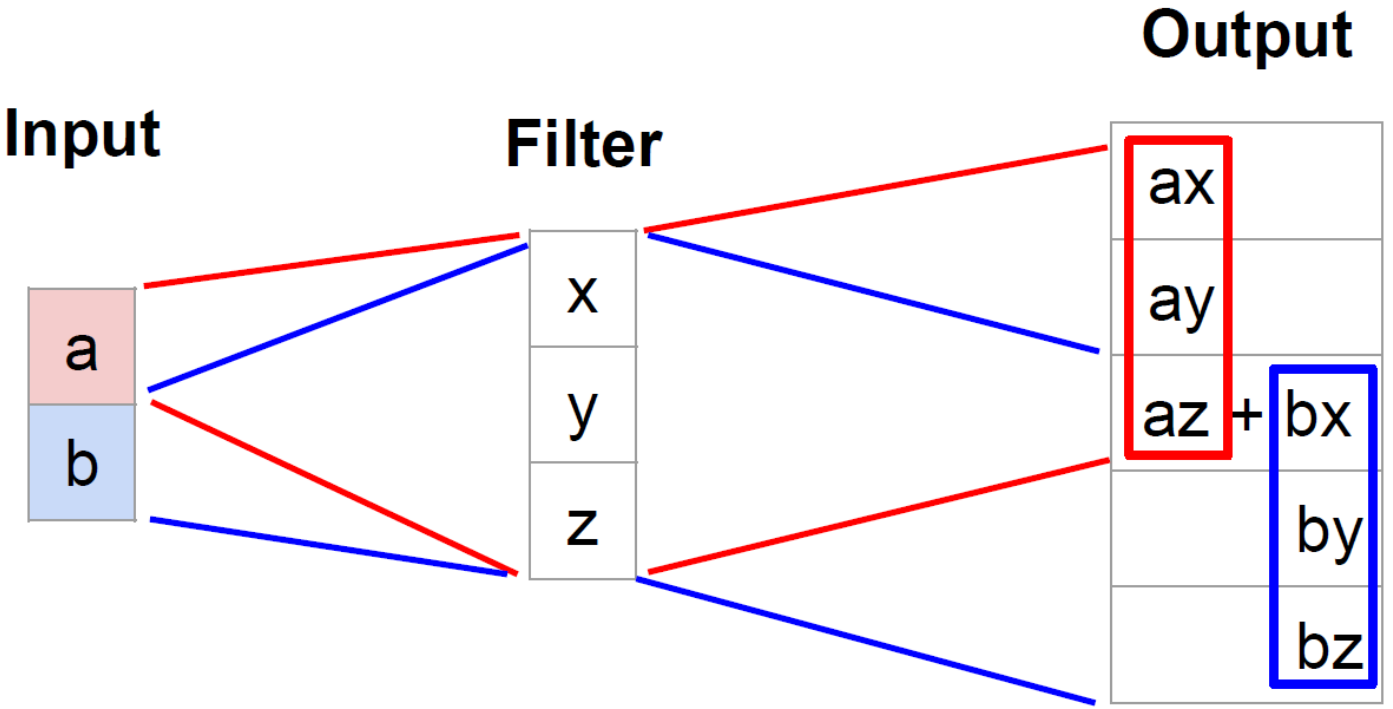
- Why Transpose Convolution:
  - Remember: Convolution ~ Matrix Multiplication: y=Mx (Left, n=3, s=1, p=1)
  - Transpose Convolution: is also Matrix Multiplication: $z=M^T t$ (Right, s=1)

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay+bz \\ ax+by+cz \\ bx+cy+dz \\ cx+dy \end{bmatrix} \qquad \begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay+bx \\ az+by+cx \\ bz+cy+dx \\ cz+dy \\ dz \end{bmatrix}$$

  - How to upsample by 2?

Transpose Convolution

› Learnable Upsampling:
  – Crop output to make *output size* exactly 2x *input size*



Transpose Convolution

› *Samples:*

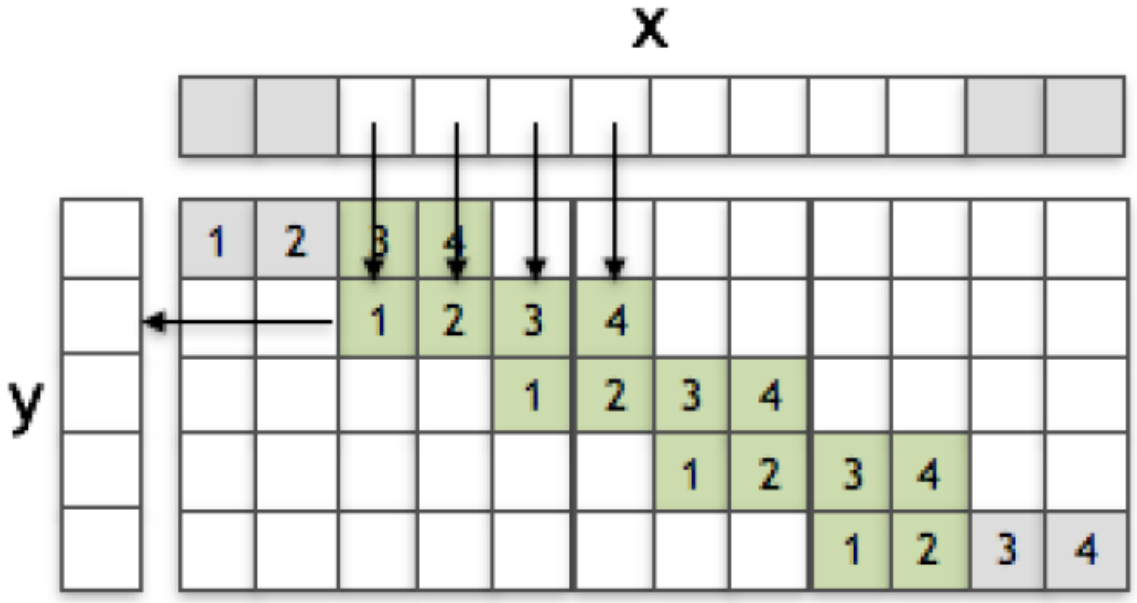


Figure 2: Convolution with stride 2 in 1D

Transpose Convolution

› *Samples (Convolution):*

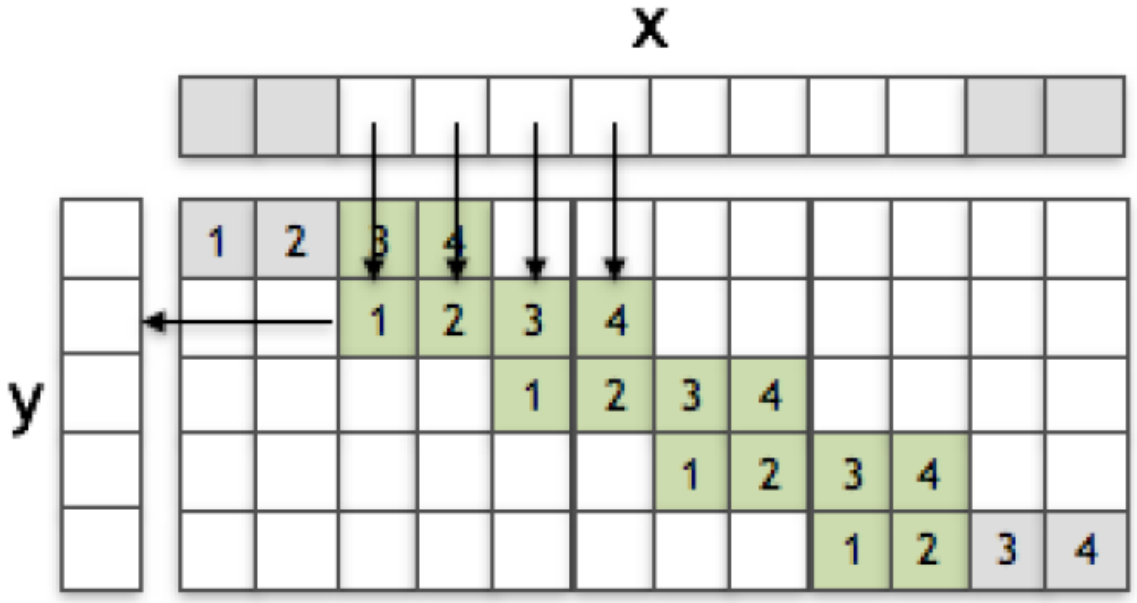


Figure 2: Convolution with stride 2 in 1D

Transpose Convolution

› *Samples (Transposed Convolution):*

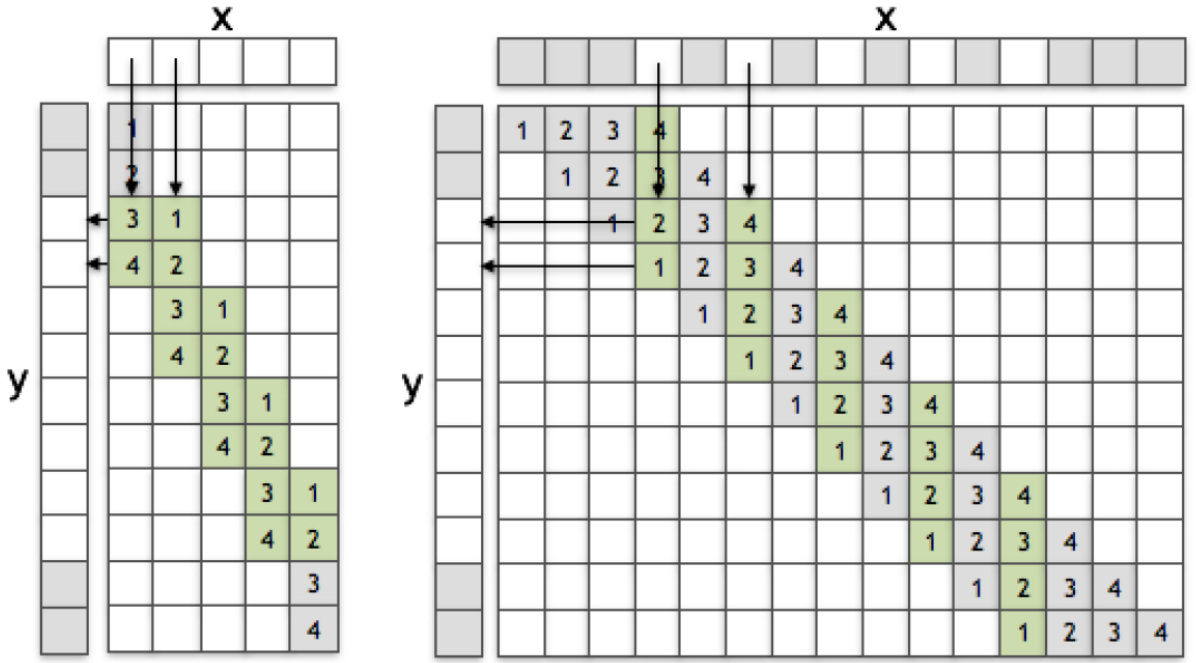
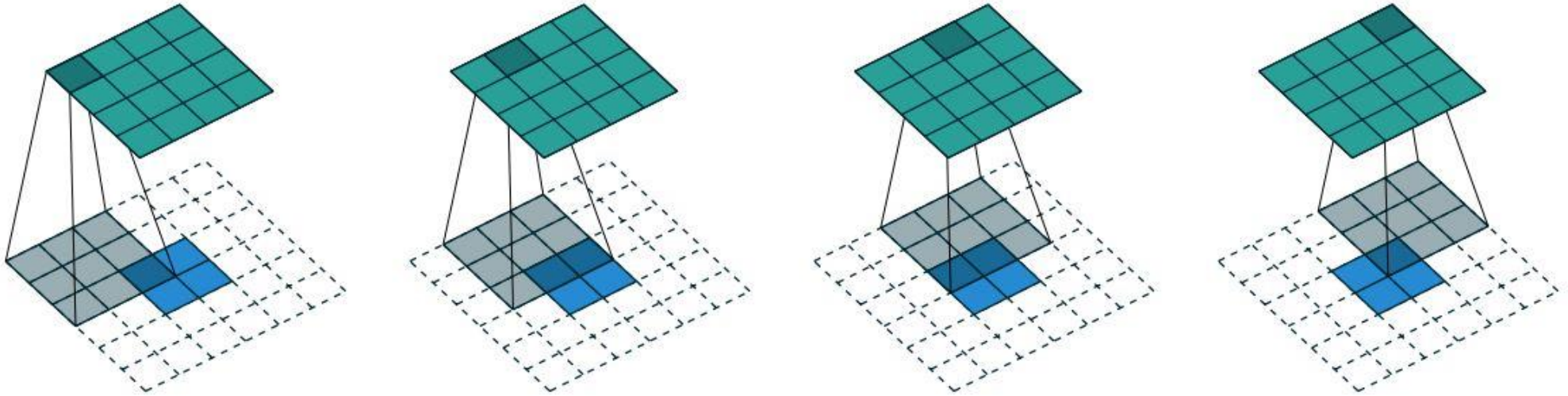
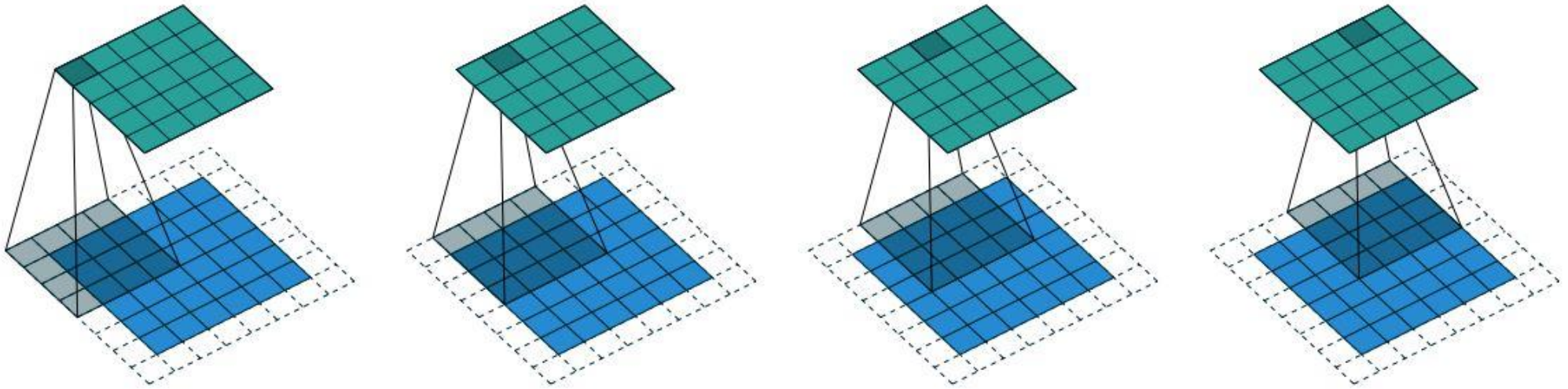


Figure 3: (a) Transposed convolution with stride 2 and (b) sub-pixel convolution with stride ½ in 1D

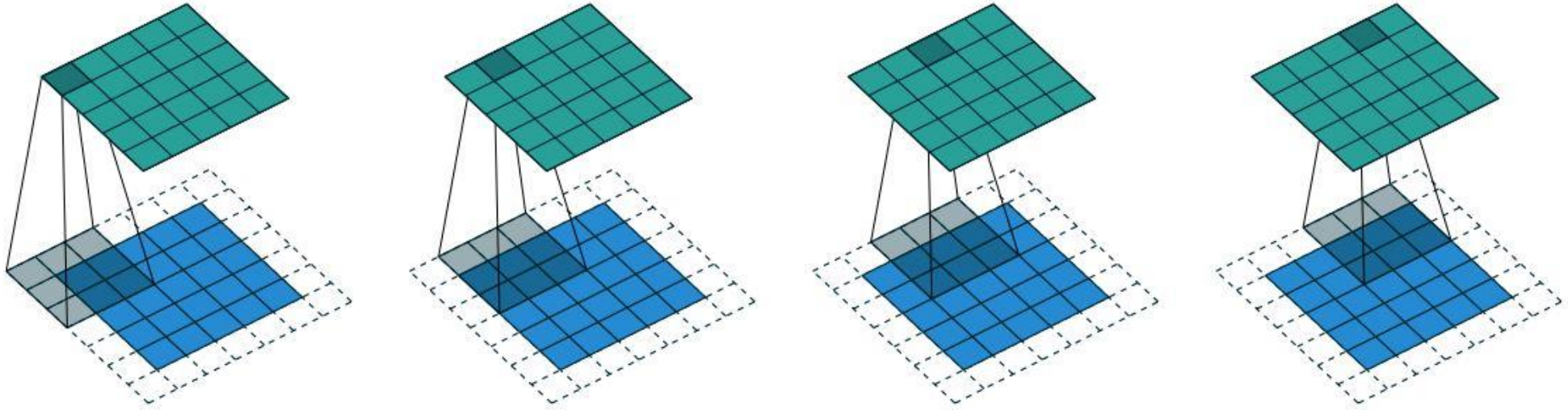
Examples - A guide to convolution arithmetic for deep learning



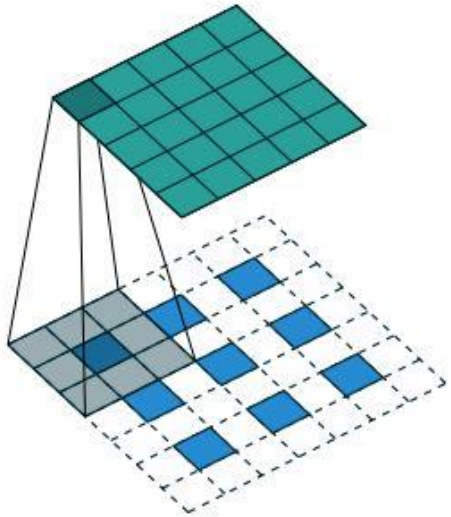
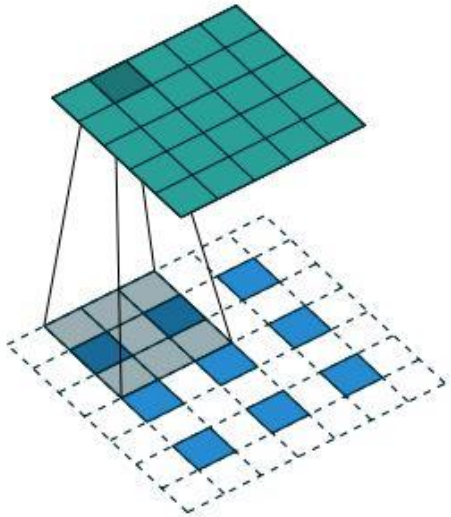
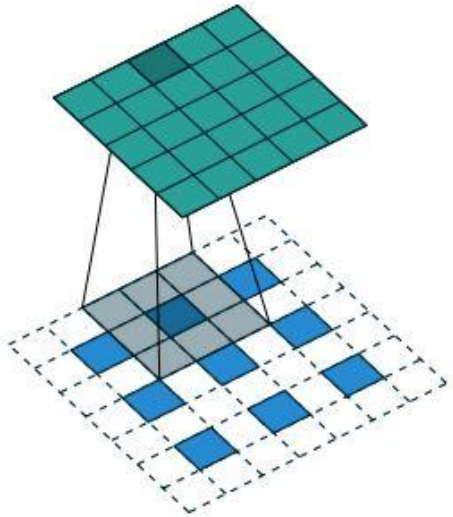
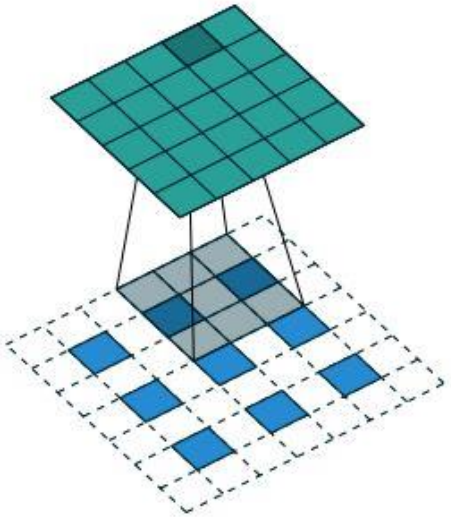
Examples - A guide to convolution arithmetic for deep learning



Examples - A guide to convolution arithmetic for deep learning



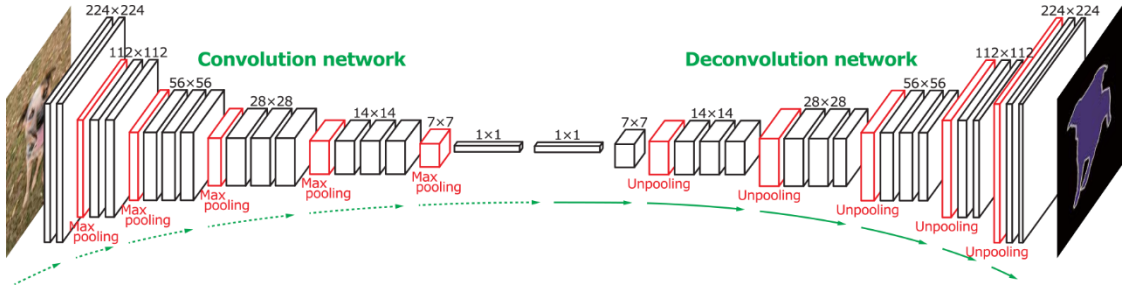
Examples - A guide to convolution arithmetic for deep learning

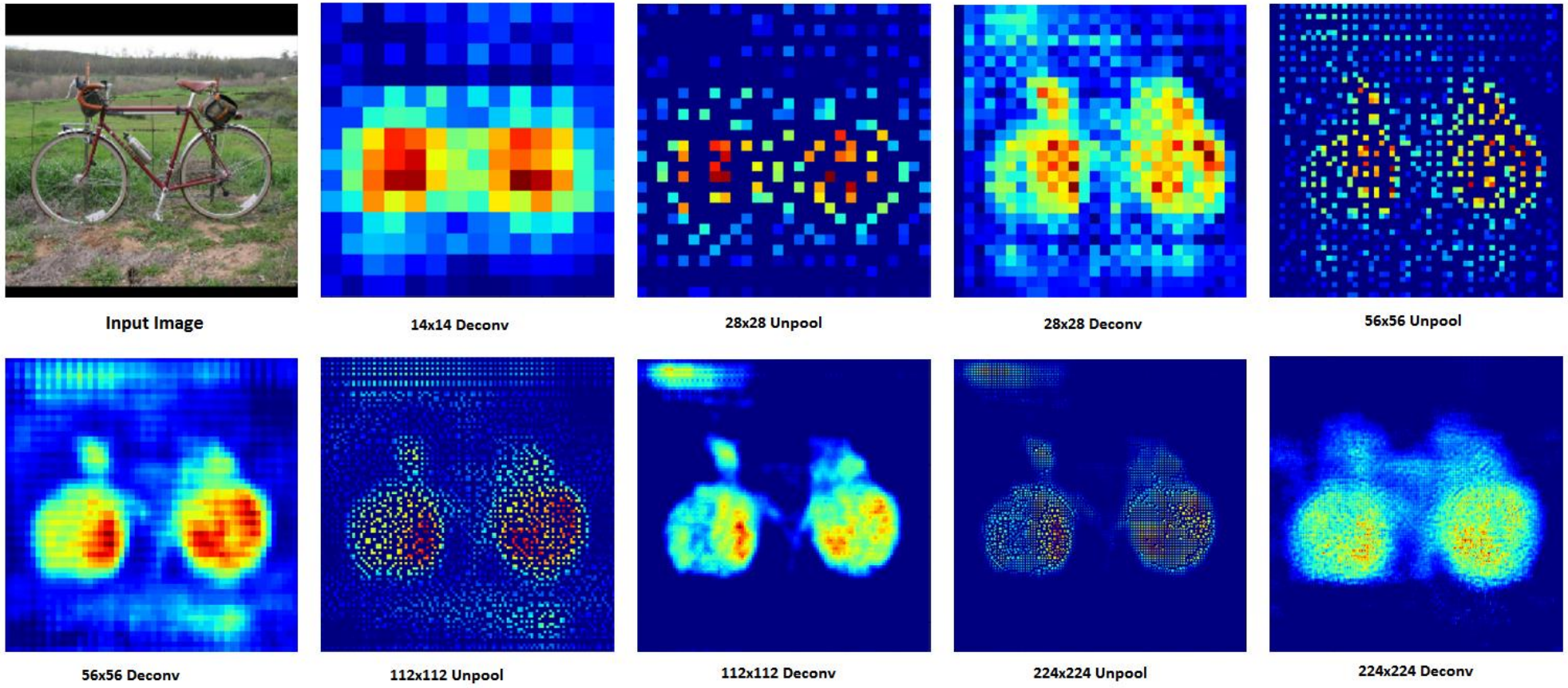
A Useful animation - **Deconvolution and Checkerboard Artifacts**

› https://distill.pub/2016/deconv-checkerboard/

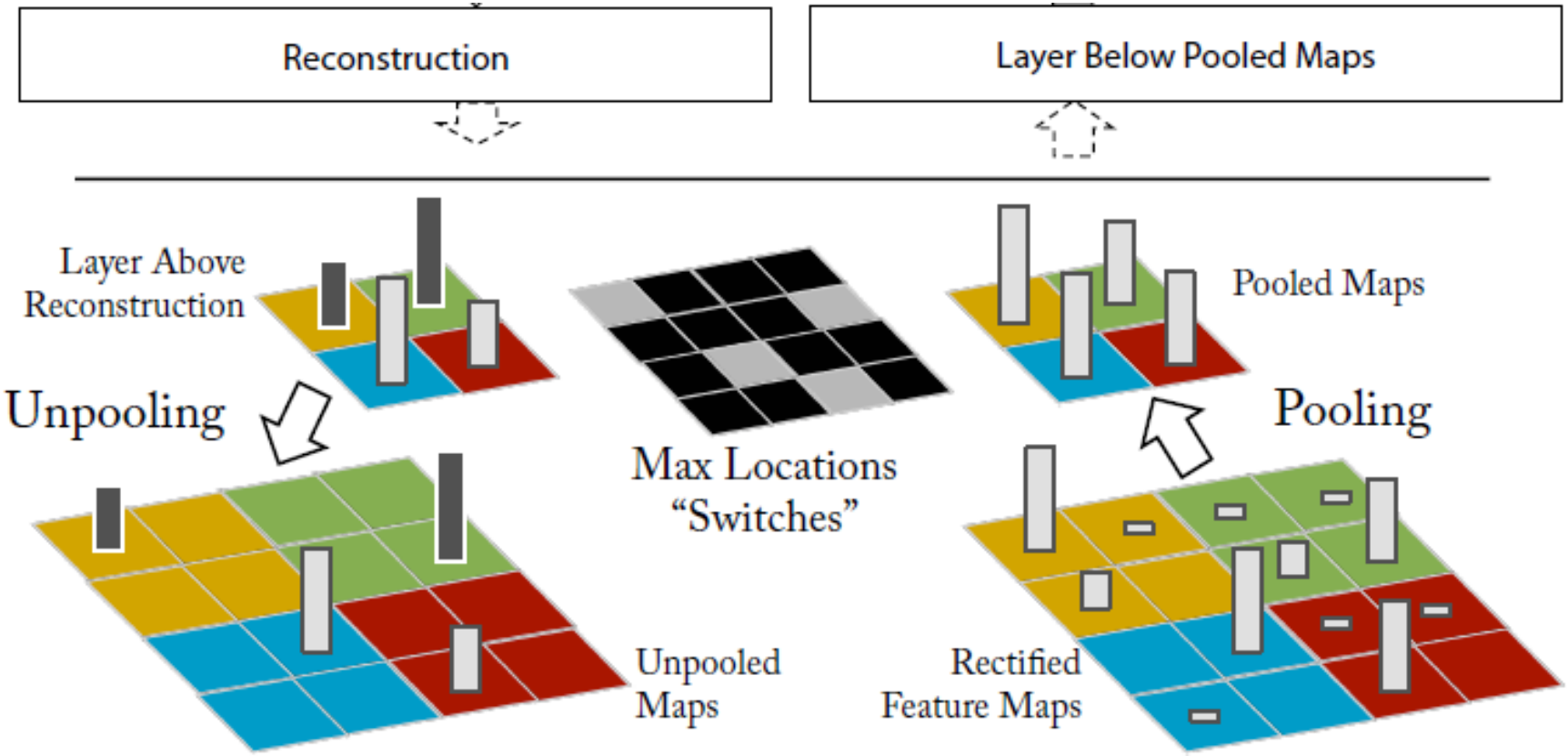
Deconvolutional Network (deconvnet)



Deconvnet visualization! - Learning Deconvolution Network for Semantic Segmentation

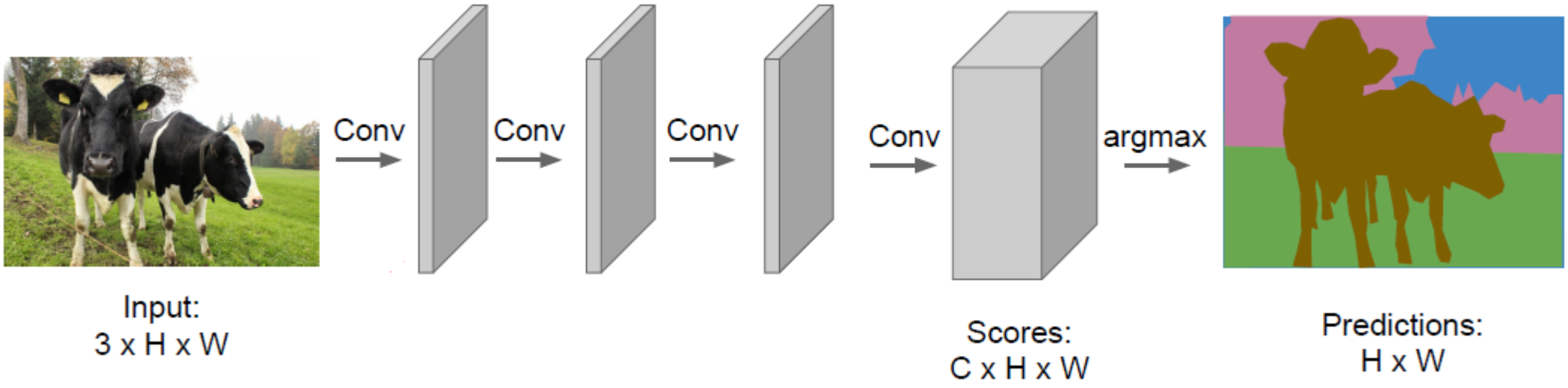


Deconvnet - Visualizing and Understanding Convolutional Networks



Semantic Segmentation

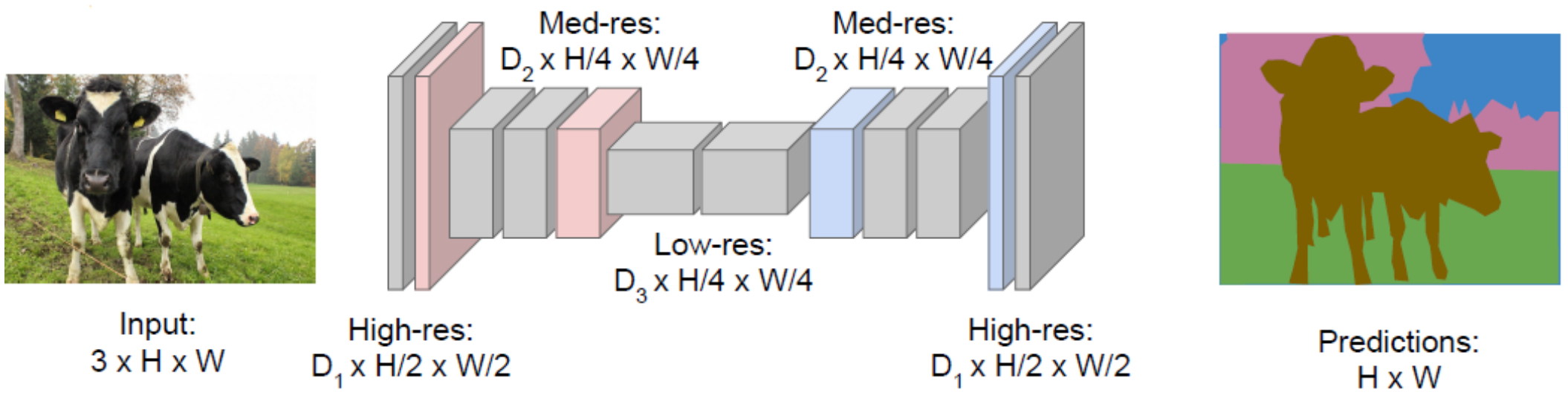
› Raw Idea:
  – Forget FC Layer (classification) → Fully Convolutional Network (FCN)



› Image from Fei-Fei Li & Justin Johnson & Serena Yeung 2017

Semantic Segmentation

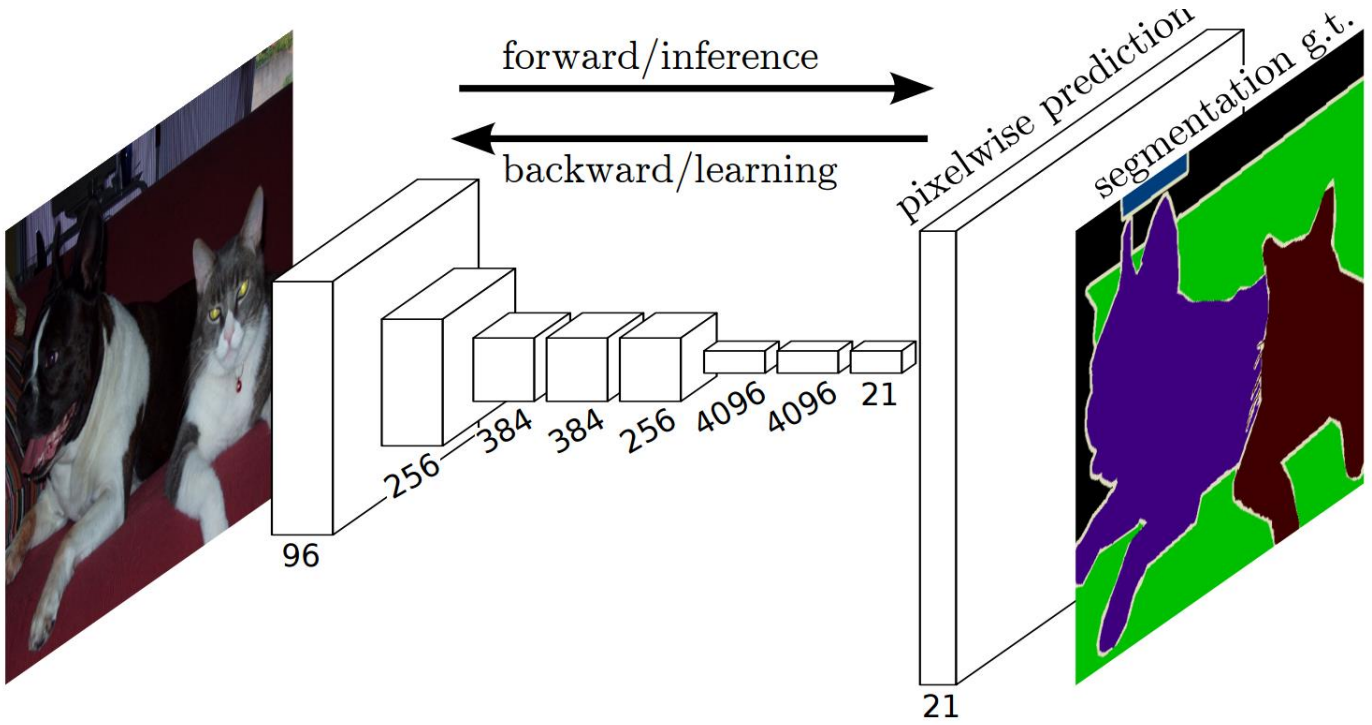
› Working Idea:
  – Forget FC Layer (classification) → Fully Convolutional Network (FCN)



› Image from Fei-Fei Li & Justin Johnson & Serena Yeung 2017

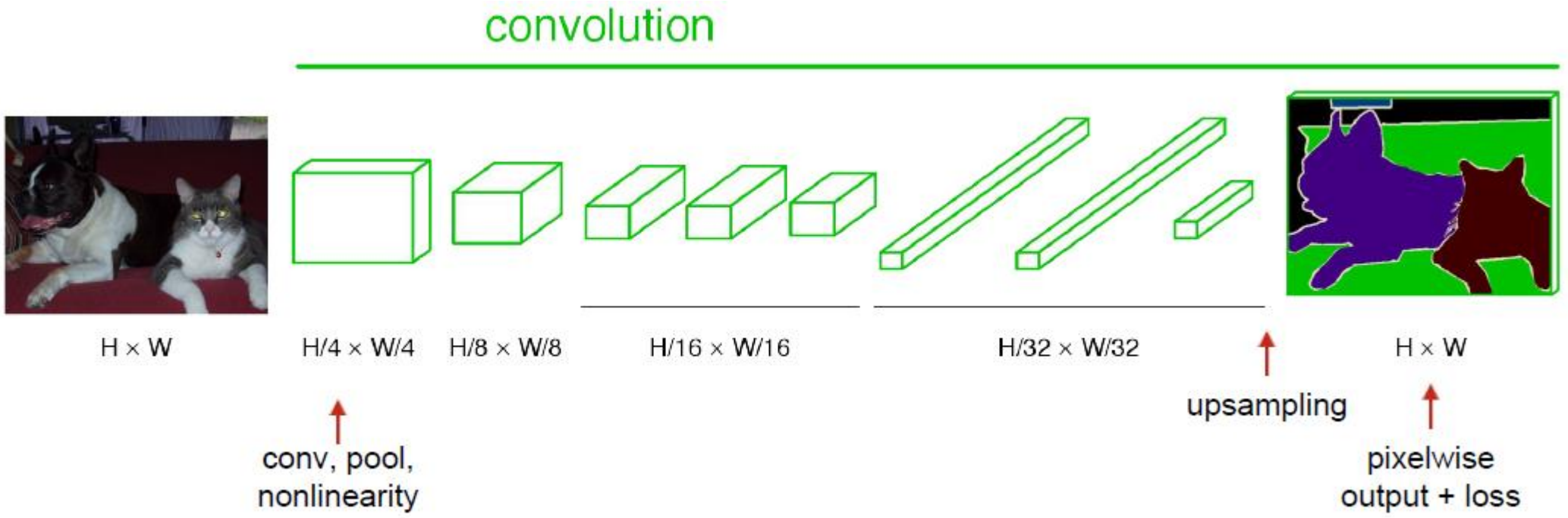
FCN - Fully Convolutional Networks for Semantic Segmentation – CVPR2015/PAMI2016

› Replace Fully Connected Layer to Convolution Layer!



FCN - Fully Convolutional Networks for Semantic Segmentation – CVPR2015/PAMI2016

› Replace Fully Connected Layer to Convolution Layer!

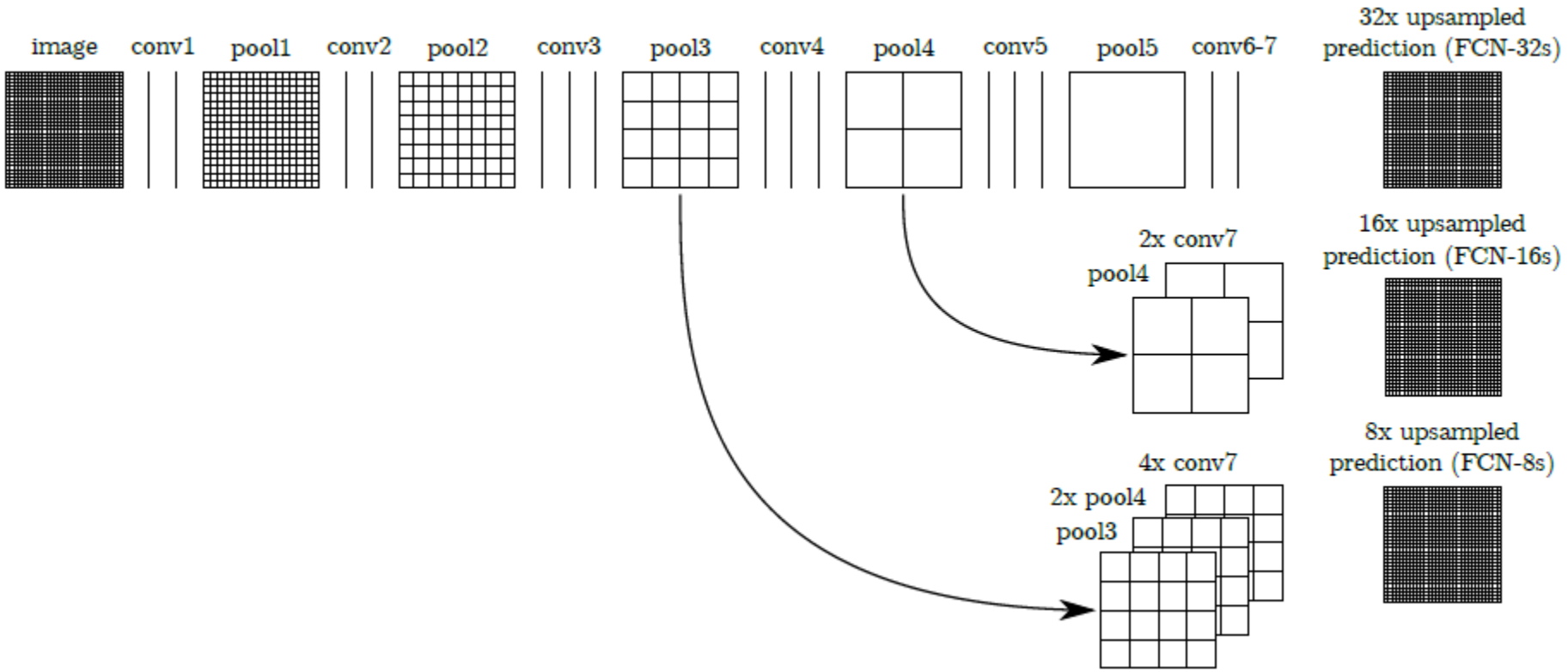


FCN - Fully Convolutional Networks for Semantic Segmentation – CVPR2015/PAMI2016

- Upsampling with DeconvLayer
- ConvLayer is pre-trained (VGG)!
- Training: Regular grid of large, overlapping patches.
- Skip Connection

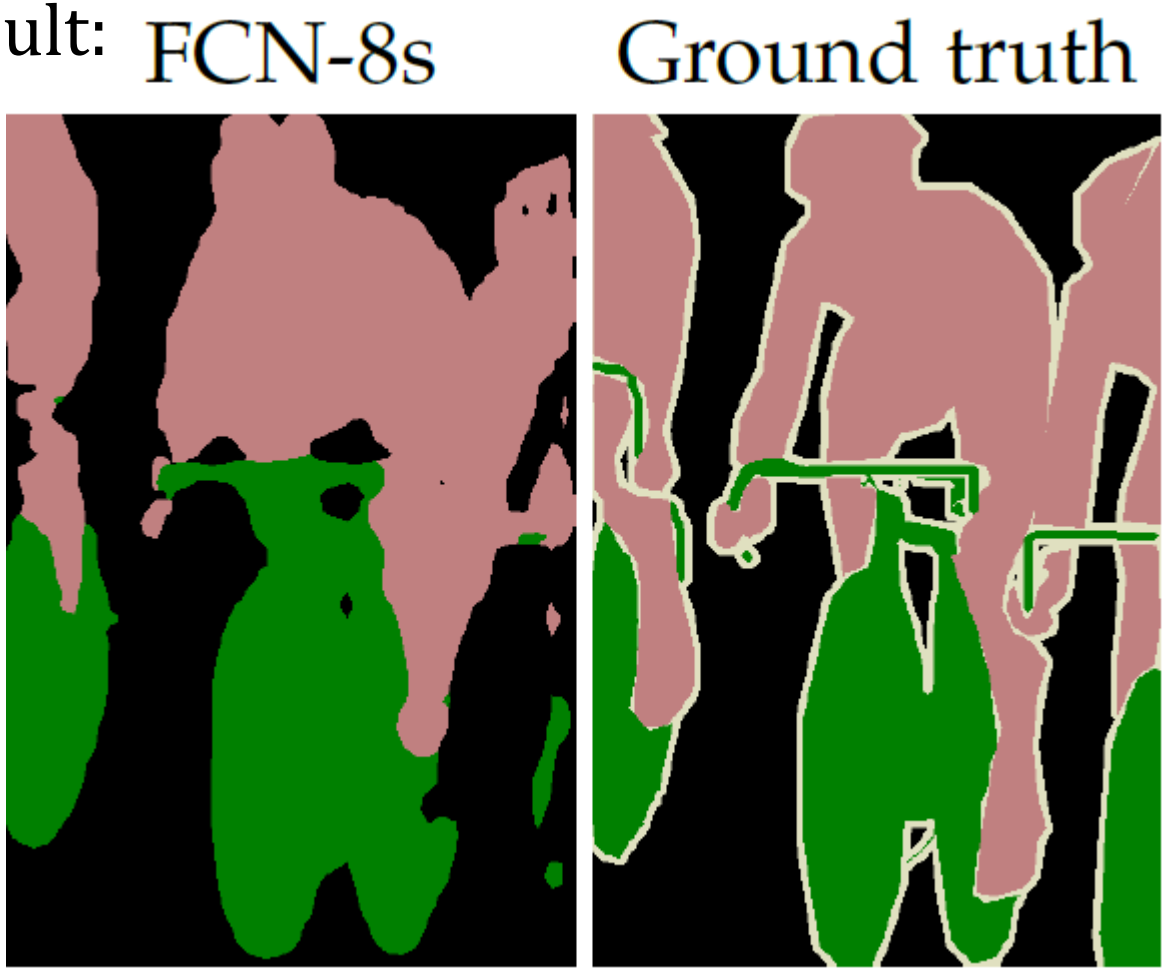
FCN - Fully Convolutional Networks for Semantic Segmentation – CVPR2015/PAMI2016

› Skip Connection:



FCN - Fully Convolutional Networks for Semantic Segmentation – CVPR2015/PAMI2016

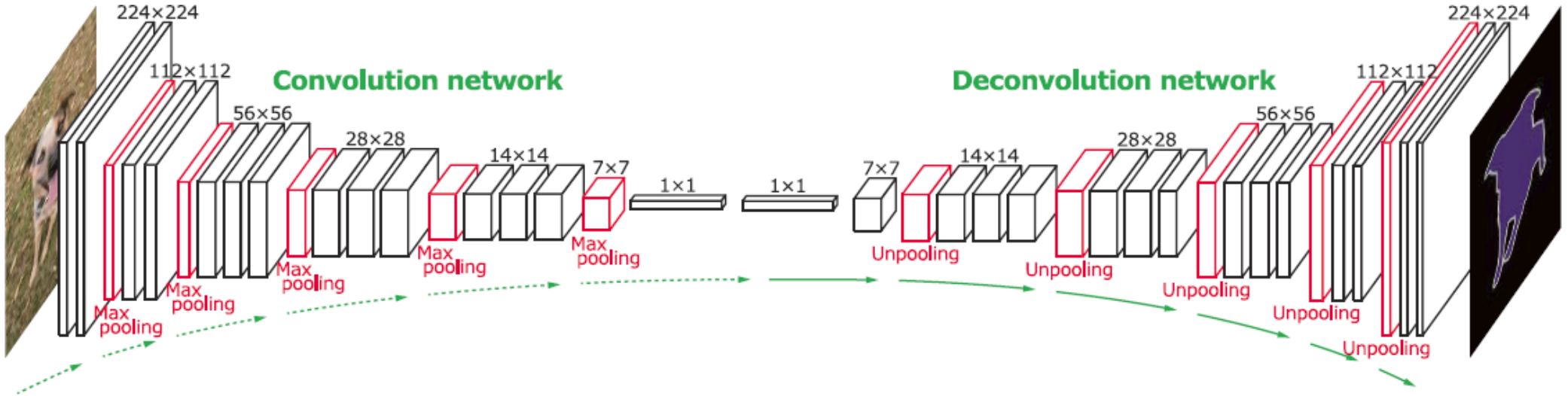
› Sample Result:



Learning Deconvolution Network for Semantic Segmentation – ICCV2015

› Unpooling (index map)+DeconvLayer

› Encoder+Decoder

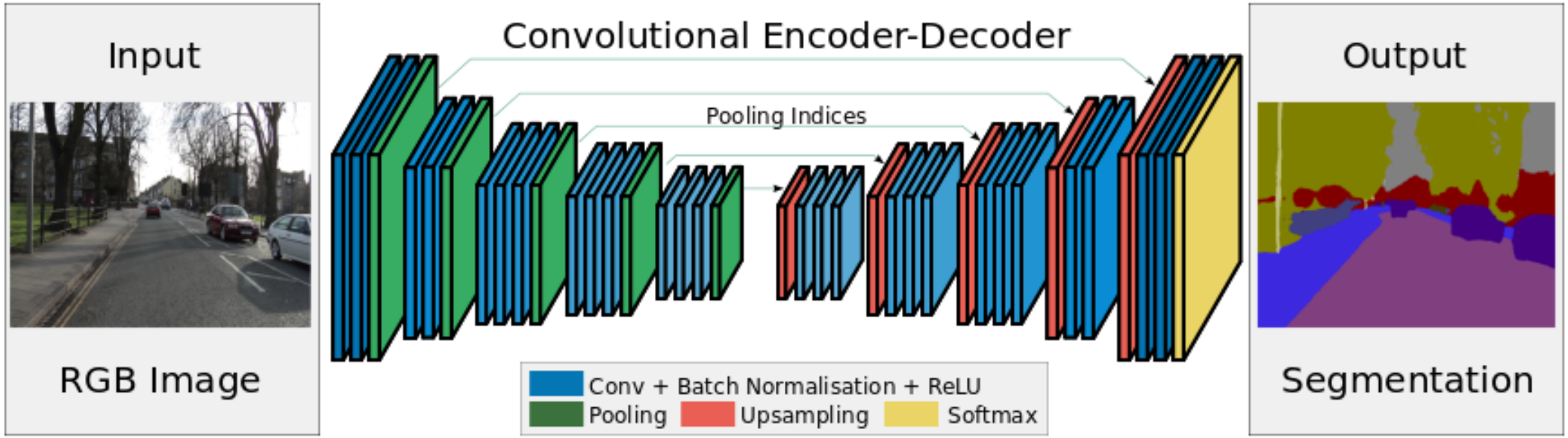


Learning Deconvolution Network for Semantic Segmentation – ICCV2015

- Details:
  - VGG16 (Encoder)
  - Batch Normalization
  - Two Stage Training (Easy then Hard samples)
    - Easy: Cropped-Centered
    - Hard: Image proposal
  - Test:
    - Image Proposal
    - Aggregate Results

SegNet - SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, PAMI 2017

› Architecture (Again Encoder-Decoder pair)

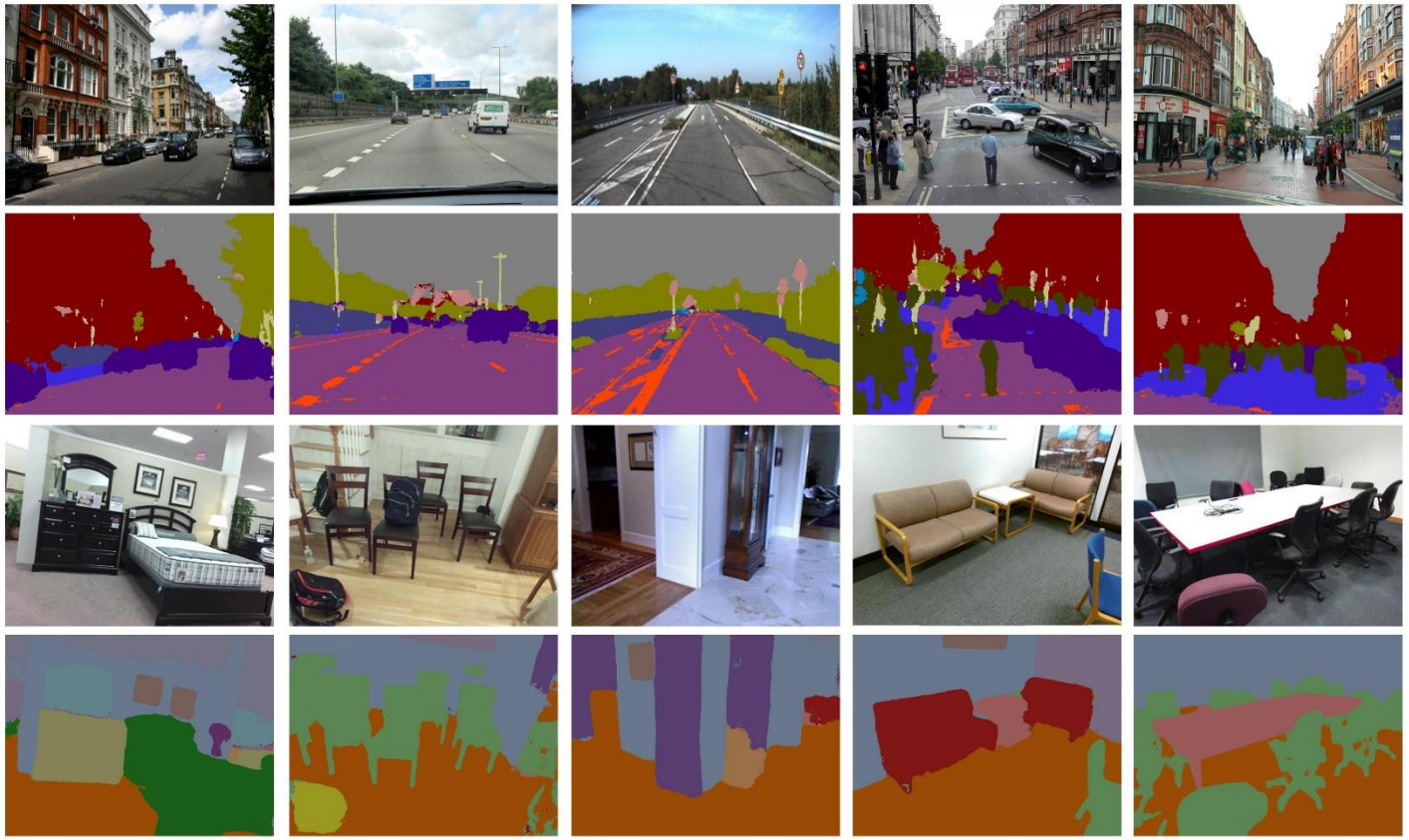


SegNet - SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, PAMI 2017

- Encoder:
  - Filter Bank
  - Batch Normalization
  - ReLu
  - Max-Pooling (2×2 and stride 2), indices are stored!
- Decoder:
  - Max-Unpooling and trainable decoder filter creates dense feature maps,
  - Batch Normalization,
  - Multi channel feature map,
  - Output: trainable K-channel SoftMax

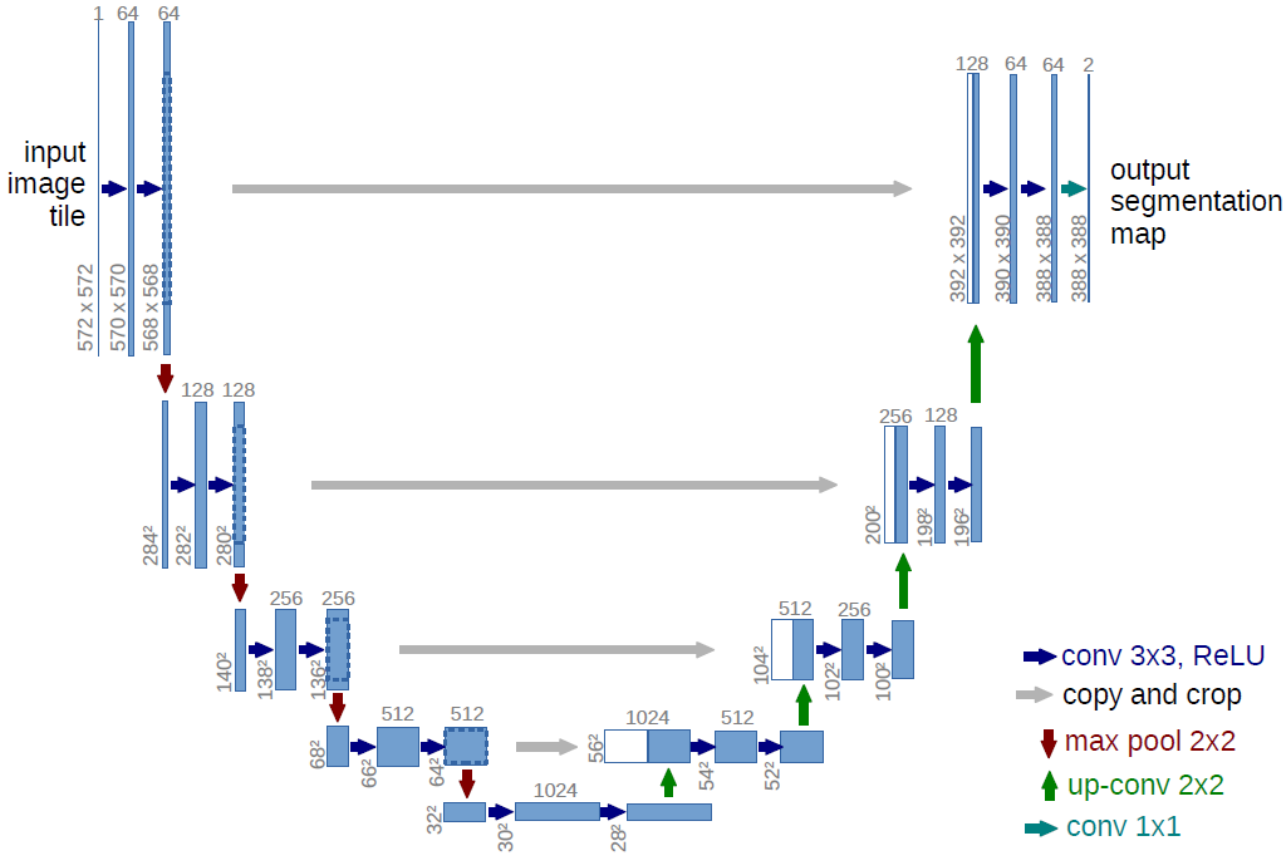
SegNet - SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, PAMI 2017

› Sample Results:



U-Net - U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI2015

› Architecture (Again Encoder-Decoder pair) + (Skip Connection)



U-Net - U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI2015

- Encoder:
  - Repeated application of two 3×3 Convolutions (unpadded)+ReLU
  - 2×2 Max Pooling with stride 2
- Decoder:
  - 2×2 up-convolution
  - Repeated application of two 3×3 Convolutions (unpadded)+ReLU
  - Cropping: Loss of border pixel (see in/out size)
  - 64 channel feature vector
- Data Augmentation:
  - Shift, Rotation, and Random Deformation

Computer Vision and CNN

- Other application:
  - Recognition
  - Verification
  - Super Resolution
  - Facial Expression
  - …

Notice!

Slides 55-98 are supplementary material and not covered in class.

Regional CNN: **R-CNN**

Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR2014

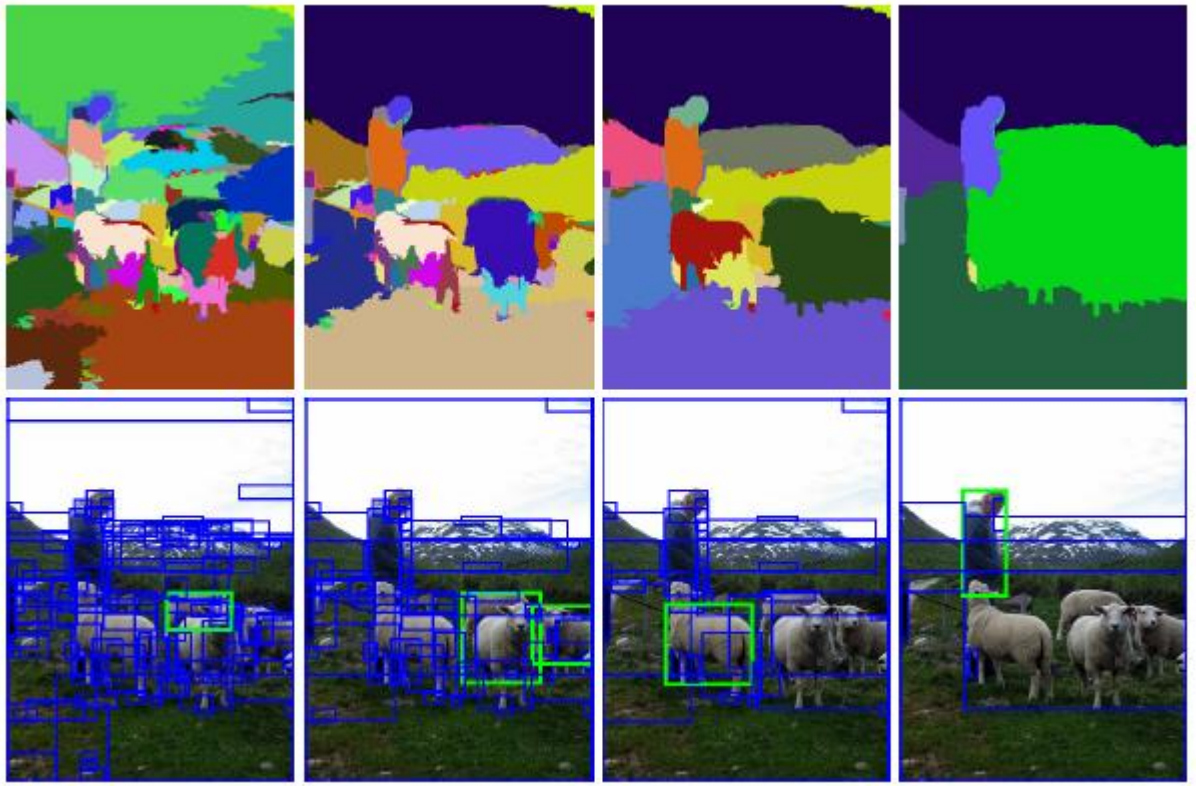
Region-based Convolutional Networks for Accurate Object Detection and Segmentation, PAMI2016

R-CNN- Rich feature hierarchies for accurate object detection and semantic segmentation

- Three (independent) Modules:
  - Region proposals:
    - Category-Independent Region Proposals (~2K candidates)
  - Feature Extraction:
    - 4096-D from a pre-trained CNN (AlexNet/ResNet) for image classification task (N=1000)
  - Classification:
    - A set of class specific SVMs, which trained for each category of objects, K=20+1

Selective Search- Selective Search for Object Recognition, IJCV2013

› Region Proposal – Selective Search
  – Over segmentation (connected region)
  – Merge Similar Region



Selective Search- Selective Search for Object Recognition, IJCV2013

› Algorithm:

Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image
Output: Set of object location hypotheses L

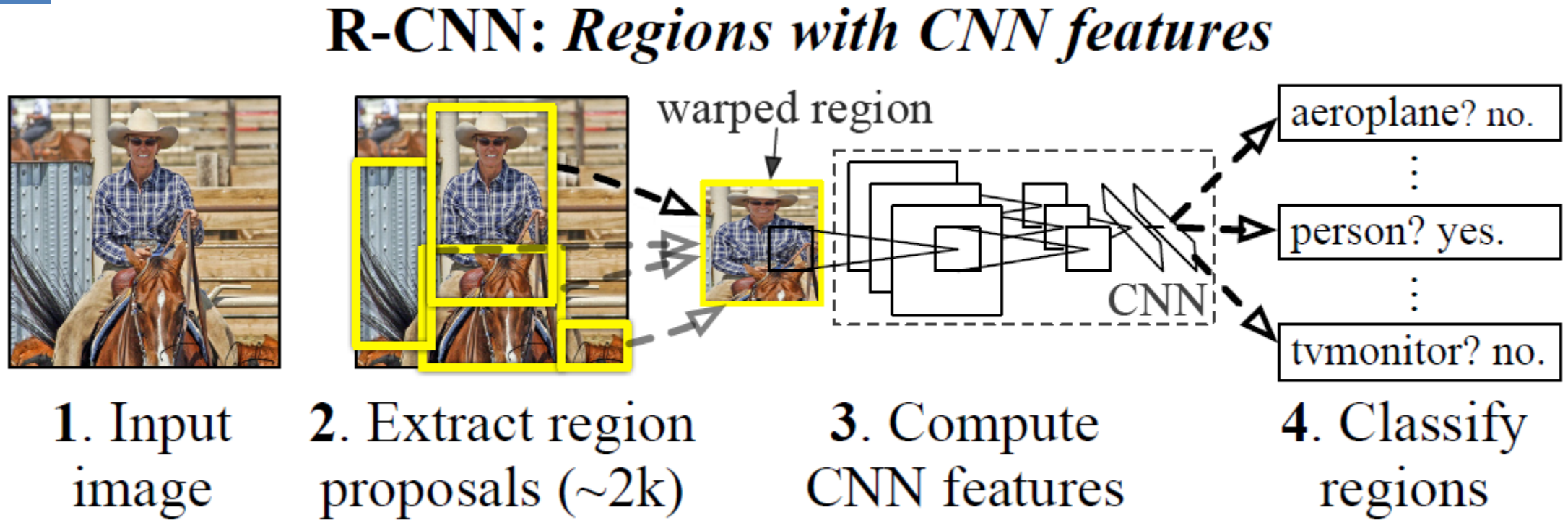
Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach *Neighbouring region pair* (r_i, r_j) **do**
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**
    Get highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set S_t between r_t and its neighbours
    $S = S \cup S_t$
    $R = R \cup r_t$

Extract object location boxes L from all regions in R

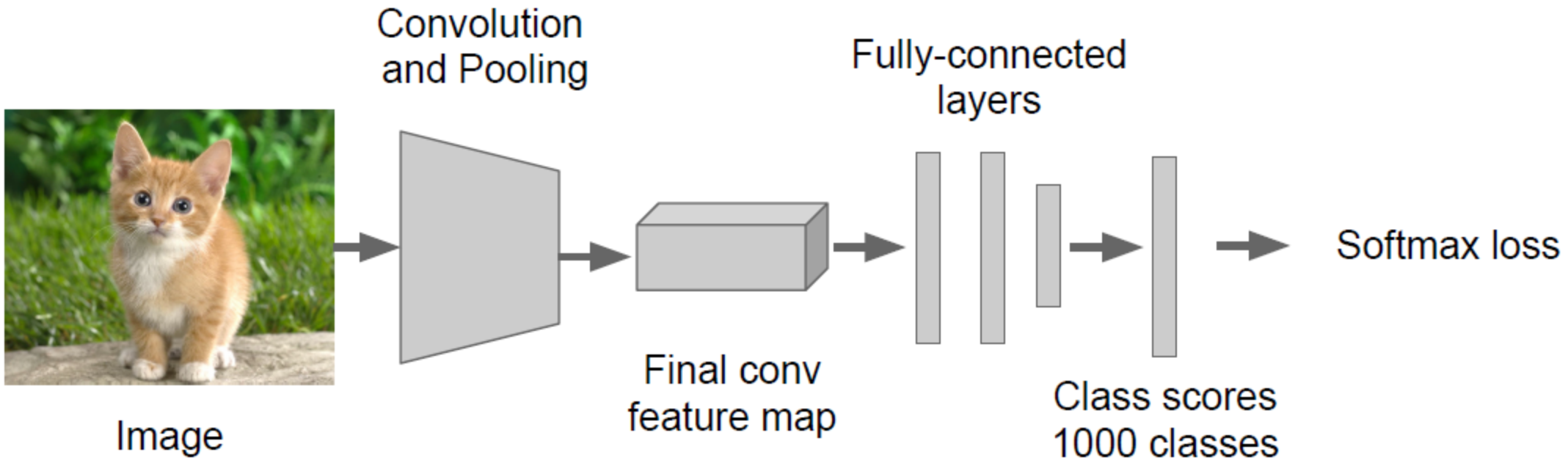
R-CNN- Rich feature hierarchies for accurate object detection and semantic segmentation

› Three (independent) Modules:



R-CNN Training

› **Step #1**: Train (or use) a classifier deep net for ImageNet (AlexNet), **1000 classes**

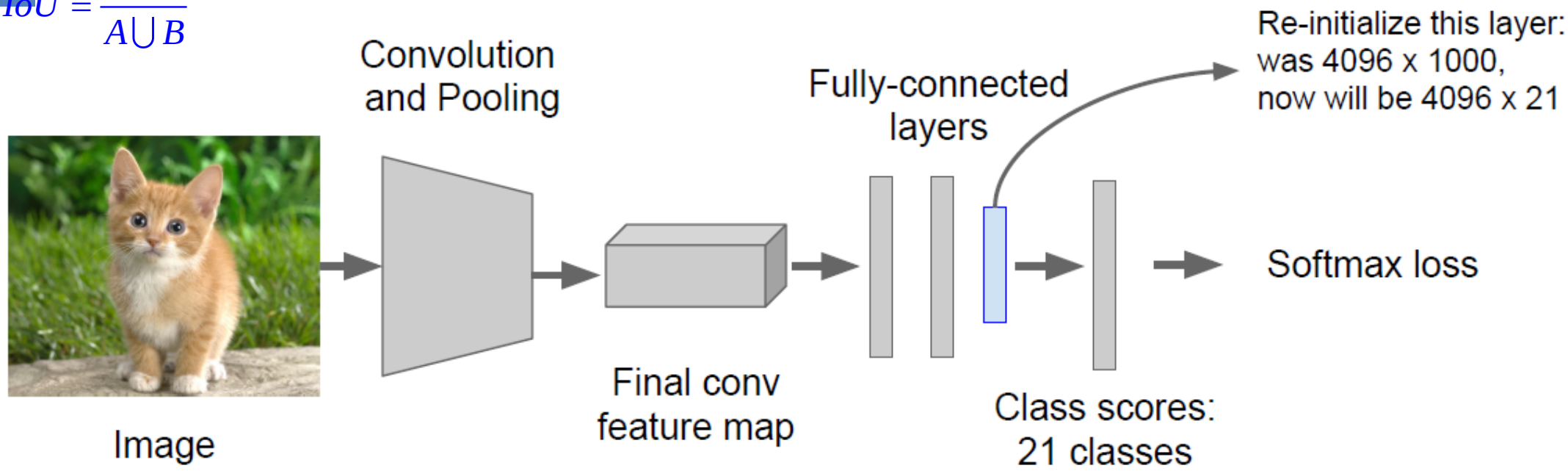


› Figure belong to: Fei-Fei Li & Andrej Karpathy & Justin Johnson

R-CNN Training

› **Step #2**: Fine tune net for detection, **20+1** (Background) classes (IoU>0.5 for positive)

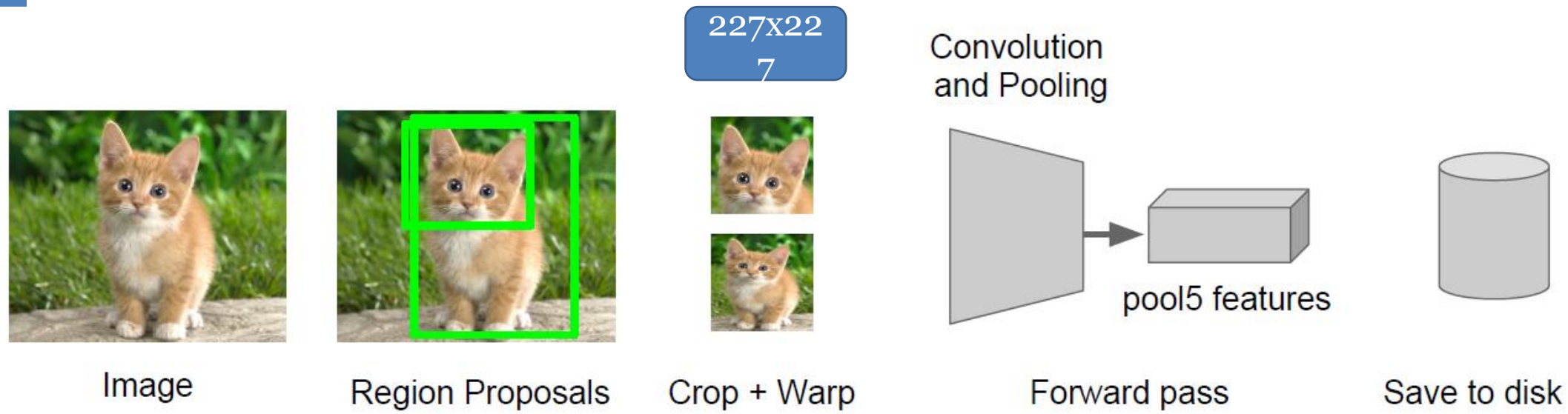
$$IoU = \frac{A \cap B}{A \cup B}$$



› Figure belong to: Fei-Fei Li & Andrej Karpathy & Justin Johnson

R-CNN Training

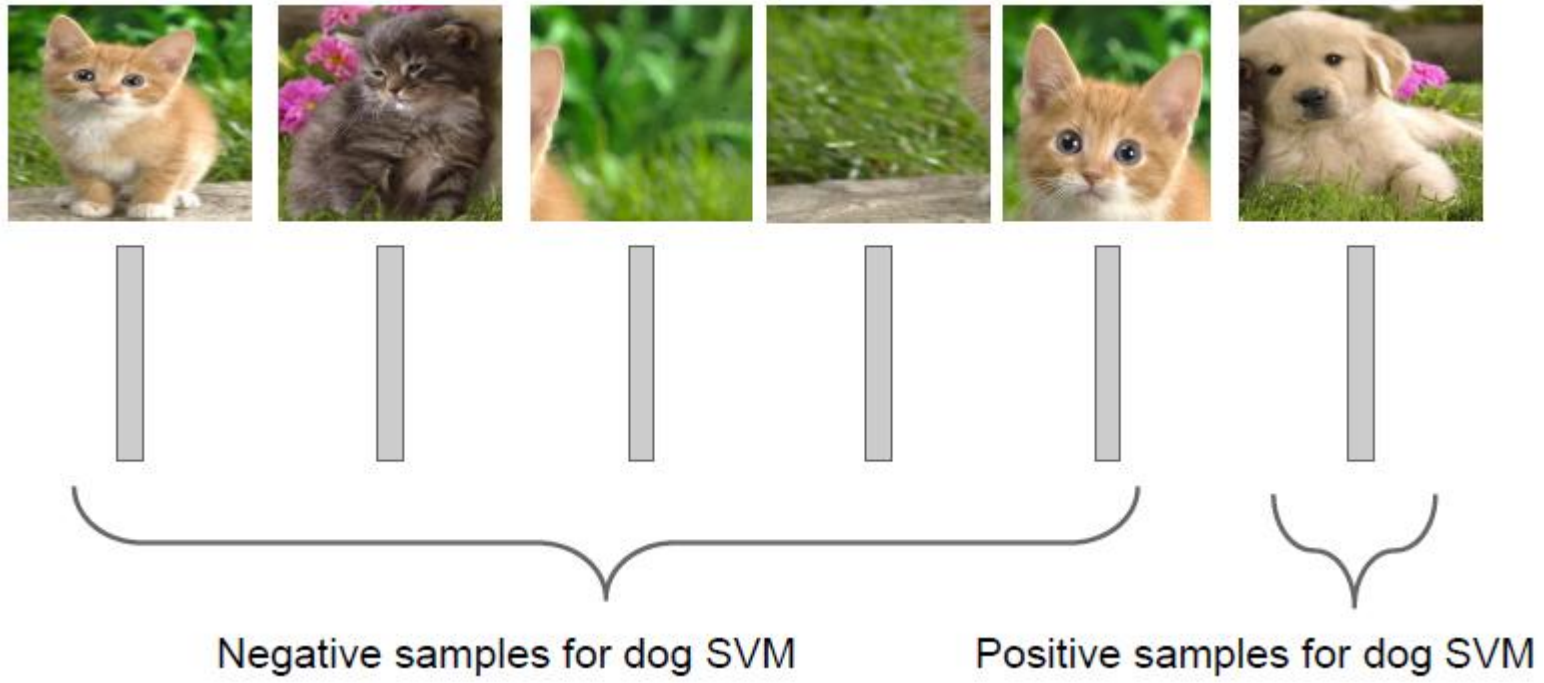
› **Step #3**: Extract Feature for warped bounding box, by CNN forward calculation



› Figure belong to: Fei-Fei Li & Andrej Karpathy & Justin Johnson

R-CNN Training

- **Step #4**: Train one binary SVM for each class to classify **region features**
- IoU>0.3 for positive sample



- Figure belong to: Fei-Fei Li & Andrej Karpathy & Justin Johnson

R-CNN Training

› **Step #5**: Bounding Box regression (*feature* to *4-touples offset* map)

› Figure belong to: Fei-Fei Li & Andrej Karpathy & Justin Johnson

R-CNN - Bounding Box Regression

- Regress Using Generated Feature by CNN (Φ_5)
  - N Training Pairs: Proposal (**P**) and Ground Truth (**G**)

$$\left\{\left(P^i, G^i\right)\right\}_{i=1}^{N}, P^i = \left(P_x^i \quad P_y^i \quad P_w^i \quad P_h^i\right), G^i = \left(G_x^i \quad G_y^i \quad G_w^i \quad G_h^i\right)$$

  - (x,y): center of windows, (w,h): width and height of windows
- Input: CNN Generated Features
- Output: offsets of windows

R-CNN - Bounding Box Regression

› Regression Model:

$$d_\star(P) = \mathbf{w}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P)$$

$$\hat{G}_x = P_w d_x(P) + P_x \qquad t_x = (G_x - P_x)/P_w$$
$$\hat{G}_y = P_h d_y(P) + P_y \qquad t_y = (G_y - P_y)/P_h$$
$$\hat{G}_w = P_w \exp(d_w(P)) \qquad t_w = \log(G_w/P_w)$$
$$\hat{G}_h = P_h \exp(d_h(P)). \qquad t_h = \log(G_h/P_h).$$

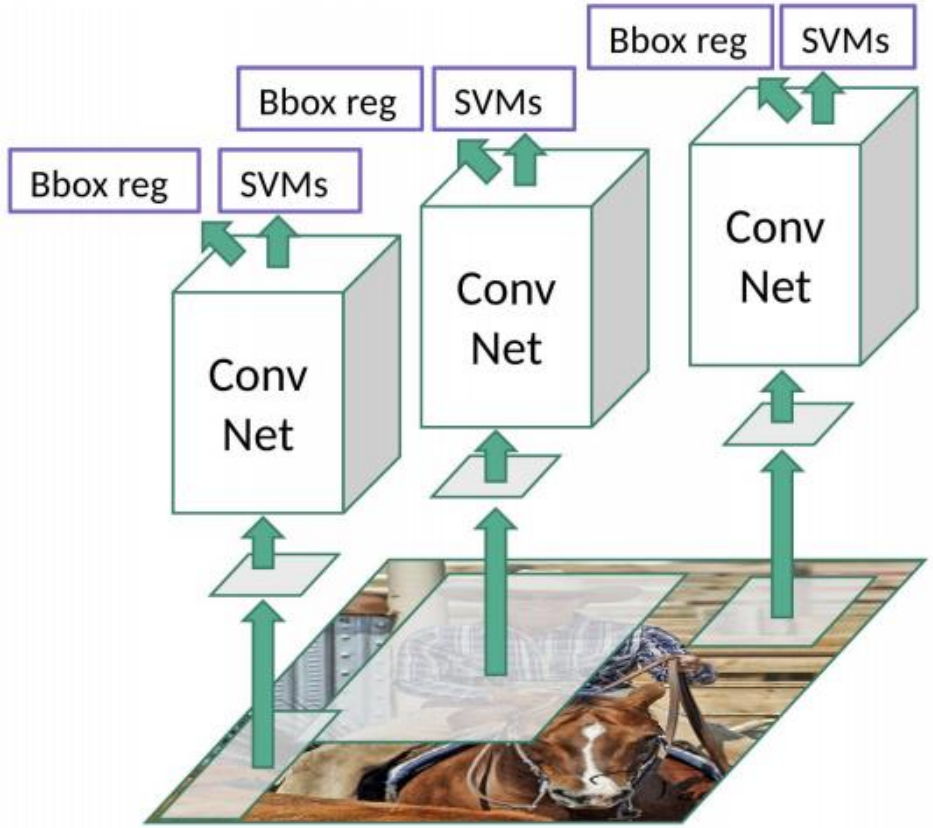
$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2$$

R-CNN- Rich feature hierarchies for accurate object detection and semantic segmentation

- Test time Detection:
  - Selective Search on input image (~2K region proposal),
  - Resize each region size to CNN-compatible input (e.g. 224x224)
  - Extract feature vector (4096) from CNN FC layer,
  - Feed the feature vector to all SVMs classifier
  - Reject region with low score

R-CNN- Rich feature hierarchies for accurate object detection and semantic segmentation

› Test time Detection:



R-CNN- Rich feature hierarchies for accurate object detection and semantic segmentation

- R-CNN Problems:
  - Triple Objective Functions:
    - SoftMax: Pre-trained CNN (AlexNet/ResNet/…)
    - Hinge Loss(+1/-1): SVM
    - L_2 Loss: Regression
  - Slow Training, Lots of storage
  - Slow Test (Detection/Inference):
    - Running selective search to find ~2k prposal for every image
    - CNN feedforward to find feature vector for every image region (~2K)

Fast R-CNN
FAIR (Facebook AI Lab)

Fast R-CNN, ICCV2015

Fast R-CNN- Fast R-CNN, ICCV 2015

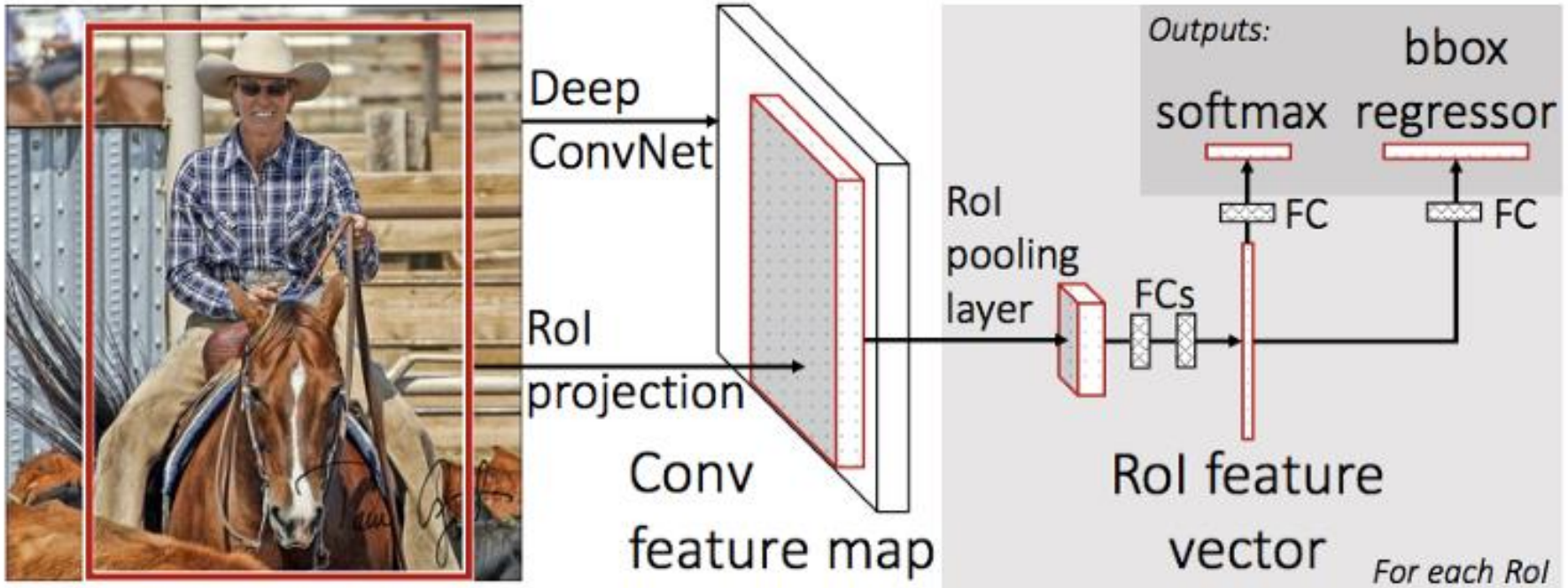
› Training is single-stage, using a multi-task loss,

› Training can update all network layers

› No disk storage is required for feature caching

Fast R-CNN- Fast R-CNN, ICCV 2015

› Fast R-CNN Architecture



Fast R-CNN- Fast R-CNN, ICCV 2015

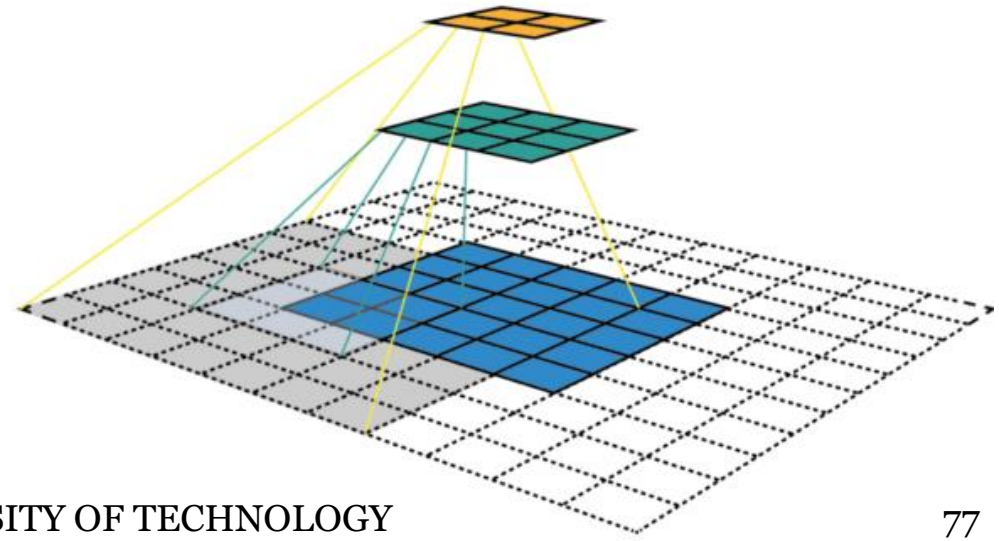
› Two Critical Idea:
  – Receptive Field
  – Region of Interest Pooling, RoI (Max) Pooling

Fast R-CNN – Receptive Field

› Receptive Field:

The receptive field is defined as the ***region*** in the ***input space*** that a particular CNN's ***feature*** is looking at

- Characterize by: center and size (Rect*Rect=Triangle!)
- Central pixel has more effect → Effective Receptive Field (ERF)
- How:
  - › Exact (RF/ERF)
  - › Use Input/output Aspect Ratio



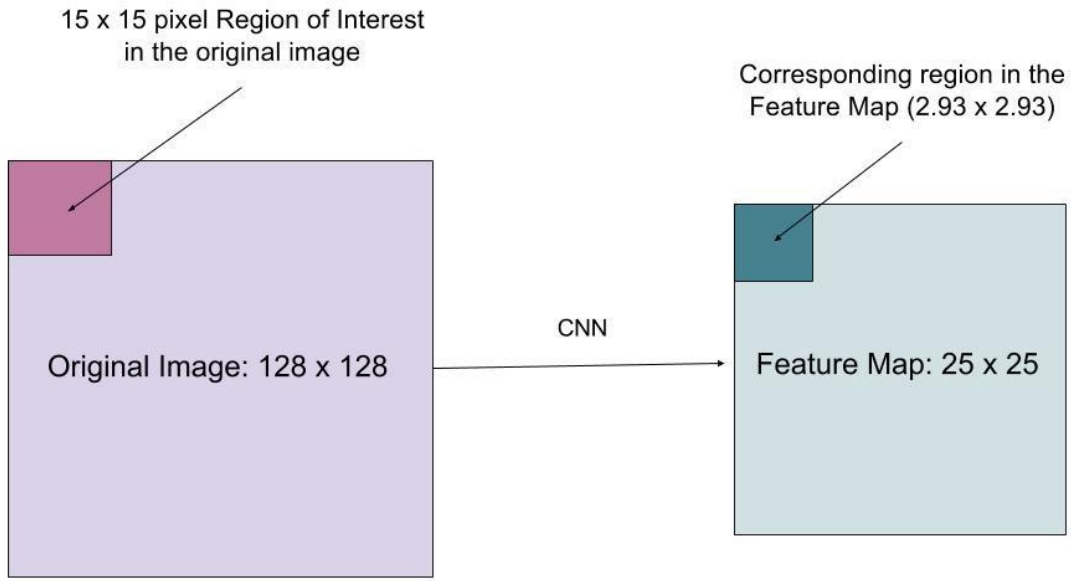
Fast R-CNN – RoI Max pooling

› RoI Max Pooling:

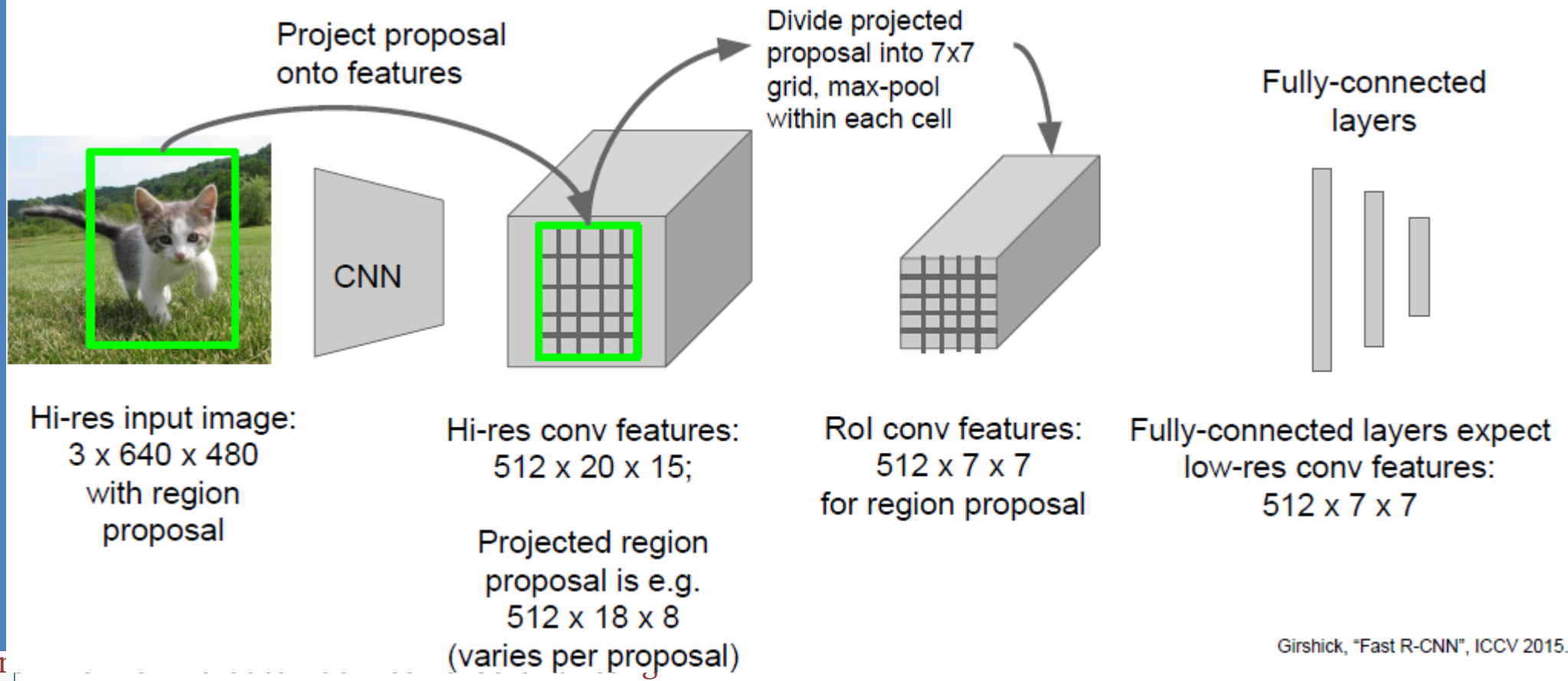
It is a type of max pooling to ***convert features*** in the ***projected region*** of the image of ***any*** size, $h \times w$, into a small fixed window, $H \times W$.

– Quantization in feature map space

– Quantization in$[h/H]\times[w/W^{-}$



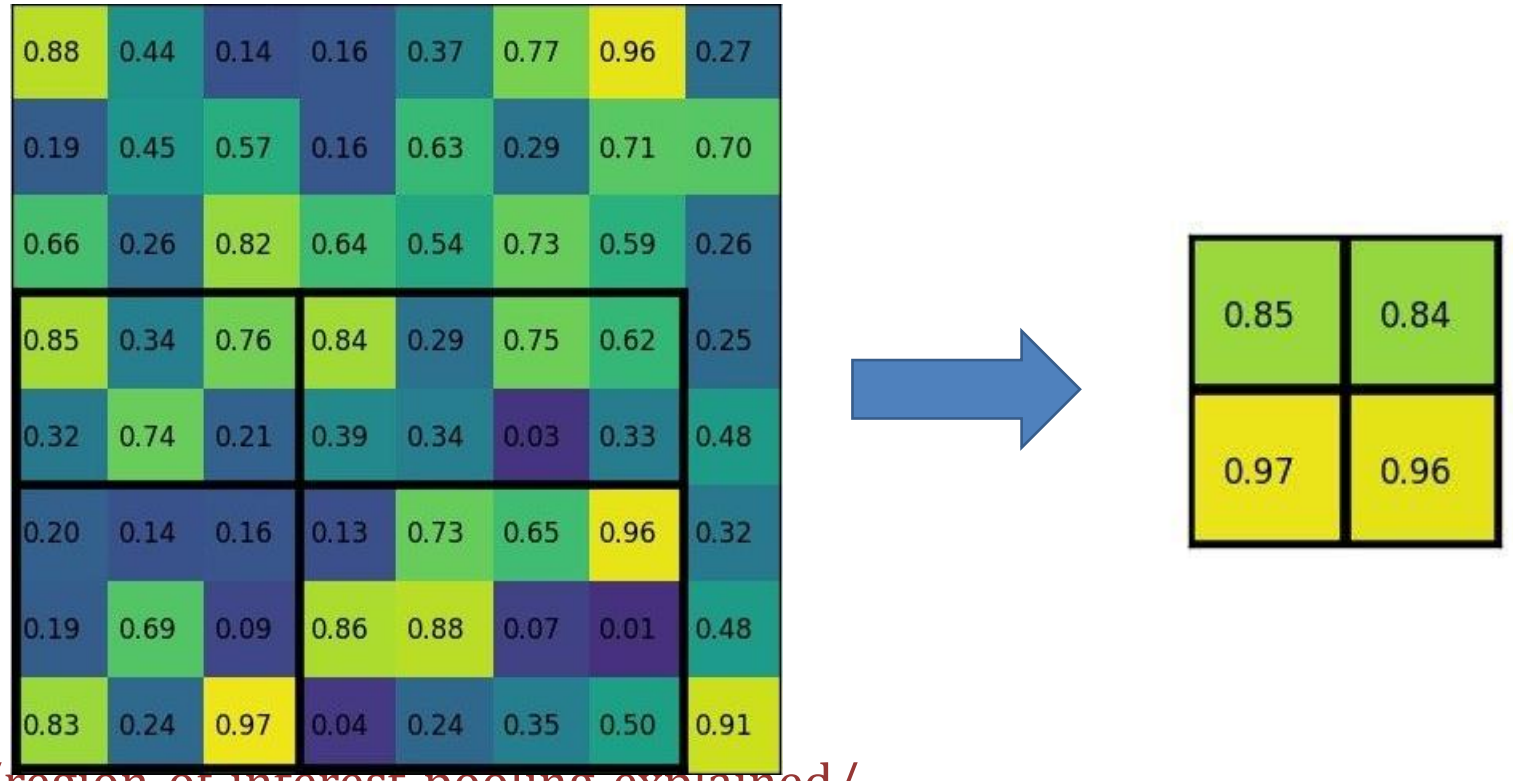
Fast R-CNN – RoI Max pooling



Girshick, "Fast R-CNN", ICCV 2015.

Fast R-CNN – RoI Max pooling

› Example: 5×7 → 2×2



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

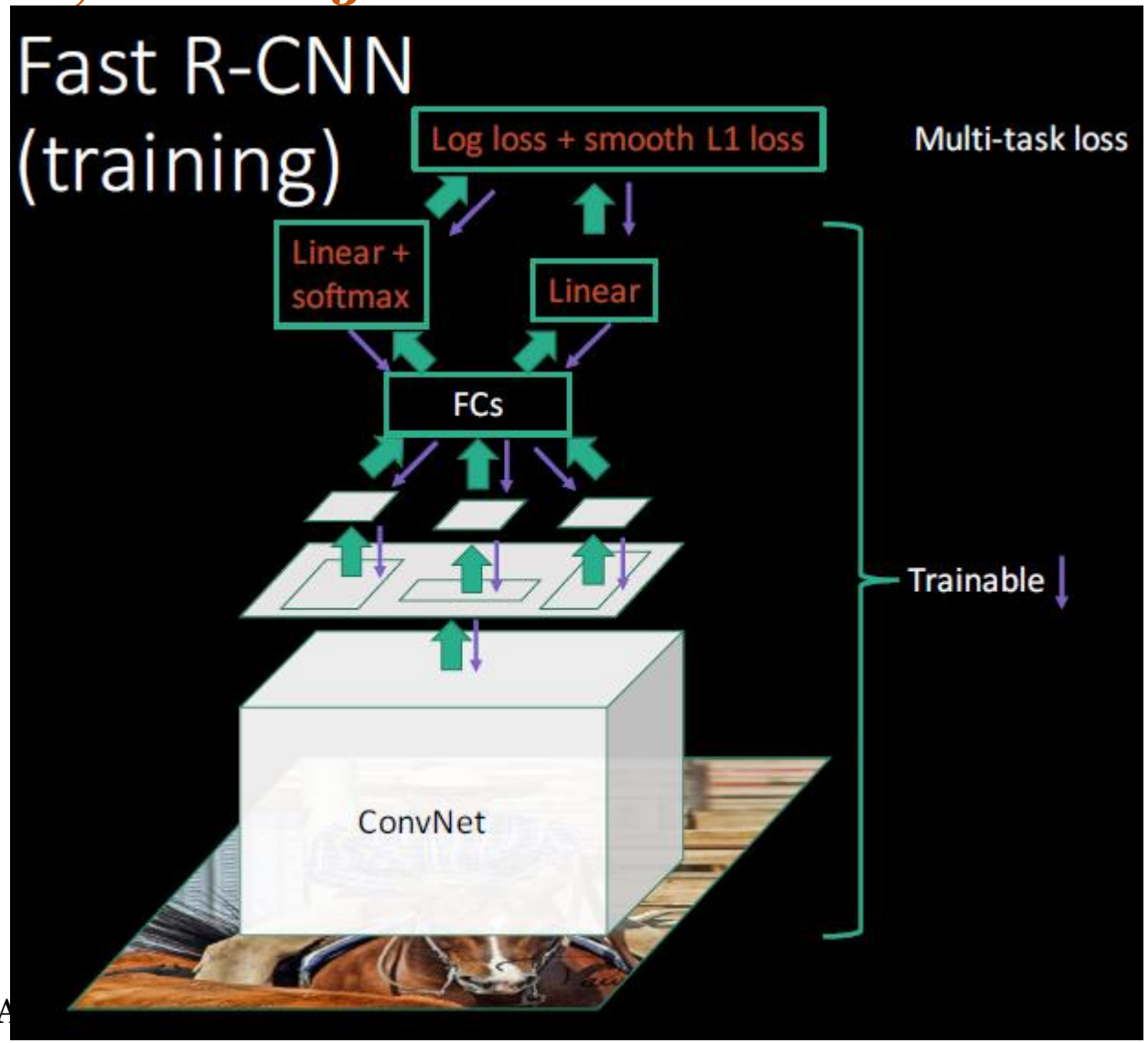
Fast R-CNN- Fast R-CNN, ICCV 2015

- Stages:
  - A pre-trained a CNN for image classification (VGG16)
    - Last max pooling layer is *replaced* by a *RoI pooling layer*
    - Last FC (1000) replaced with *two* outputs ("K+1 classes" and "bounding box regression")
    - Input data modification (one image and a list of RoI)!
  - A set of proposal regions (selective search)
- SGD+minBatch:
  - (N=2) images + (R/N=64) RoI
  - Positive RoI: IoU≥0.5
  - Negatibe (Background): IoU≤0.5 and IoU≥0.1

Fast R-CNN Fast R-CNN, ICCV 2015

› Training

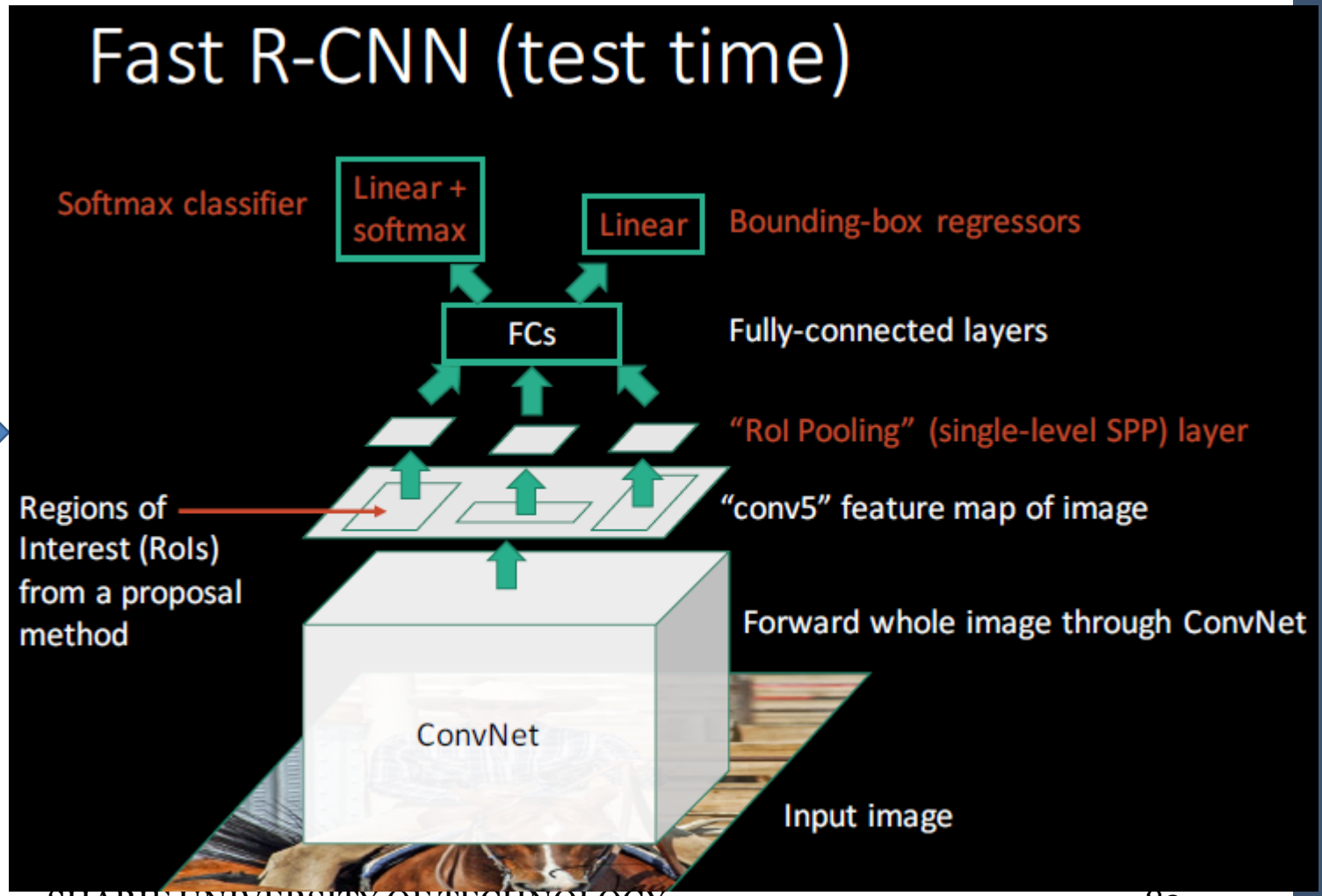
RoI pooling



Fast R-CNN Fast R-CNN, ICCV 2015

› Test

RoI pooling



Fast R-CNN Fast R-CNN, ICCV 2015

› Loss Functions:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v)$$

› First term: Loss function for true class label u

› u: True Class: Label {0,1,2,…,k}

› p: Discrete probability distribution (per RoI) over K + 1 classes: $p=(p_0,\ldots,p_K)$, computed by a softmax over the K + 1 outputs of FC.

› v: True bounding box: $v=(v_x,v_y,v_w,v_h)$

› t: predicted bounding box correction: $t^u=(t_x,t_y,t_w,t_h)^{(u)}$

› $[u\geq 1]$= one for *objects* and zero for background

Fast R-CNN Fast R-CNN, ICCV 2015

› Loss Functions:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v)$$

$$L_{\text{cls}}(p, u) = -\log p_u$$

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

$$t_x = (G_x - P_x)/P_w$$
$$t_y = (G_y - P_y)/P_h$$
$$t_w = \log(G_w/P_w)$$
$$t_h = \log(G_h/P_h).$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

You Only Look One: YOLO

- You Only Look Once: Unified, Real-Time Object Detection, CVPR2016

YOLO9000: Better, Faster, Stronger, CVPR2017

YOLOv3: An Incremental Improvement, Tech. Rep. (https://pjreddie.com)

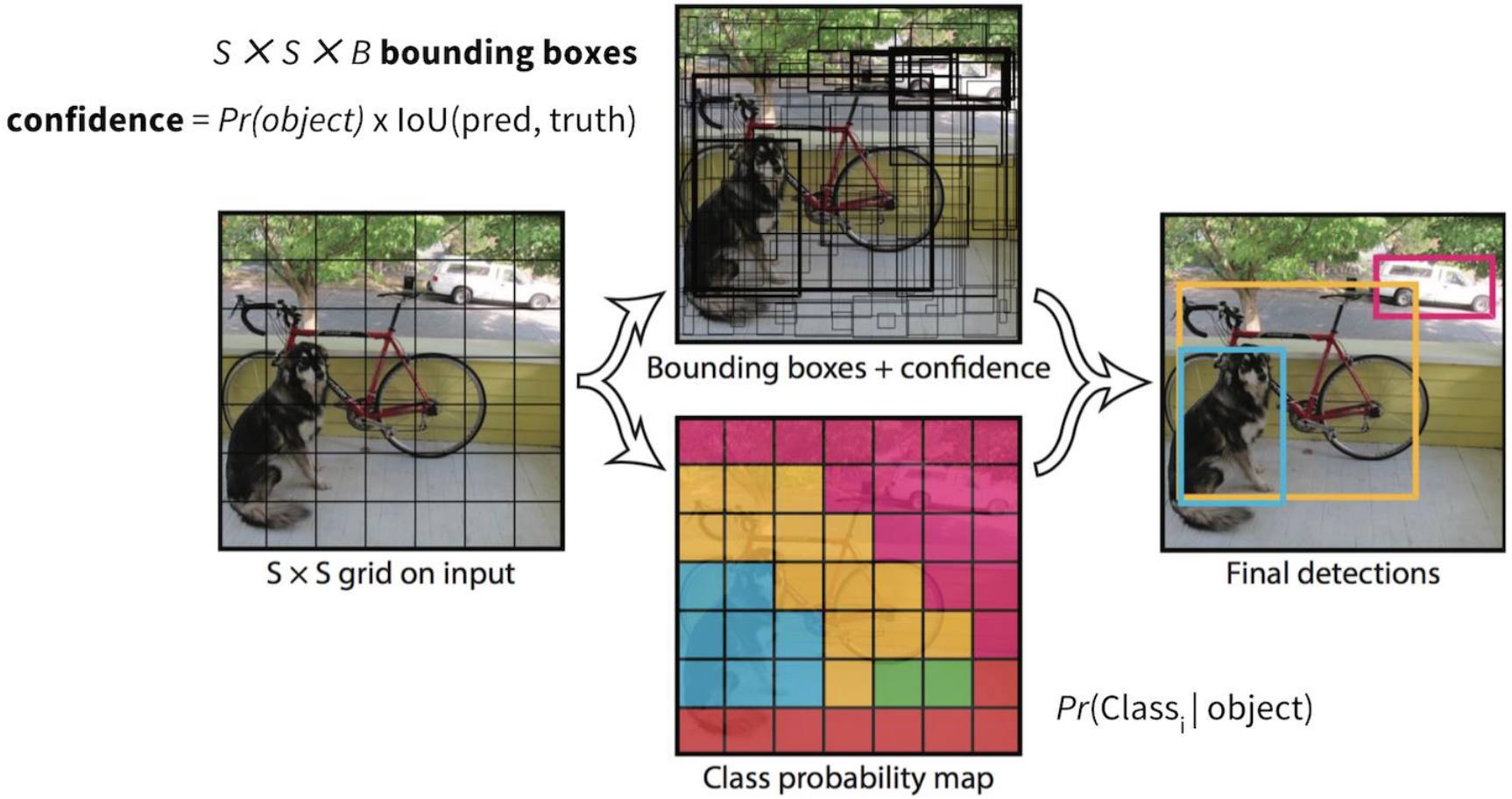
YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

- Solve Object recognition task as a *unified regression problem*!
- YOLO Look @ the entire image during training and Generate Bounding Box in output

YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

- YOLO:
  - Split image to “S by S” Cell/Patch
  - Each cell is responsible for identifying the object (if any) with its center located in this cell.
  - Each cell predicts:
    - The location of B (=2) bounding boxes,
    - The confidence score
    - The probability of object class conditioned on the existence of an object in the bounding box.

YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016



YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

- Bounding Box:
  - 4-touple: (x,y,w,h),
    - x and y normalized to offset of cell center
    - w and h are normalized by the image width and height, between (0, 1].

$$\begin{aligned}\text{box confidence score} &\equiv P_r(object) \cdot IoU \\ \text{conditional class probability} &\equiv P_r(class_i|object) \\ \text{class confidence score} &\equiv P_r(class_i) \cdot IoU \\ &= \text{box confidence score} \times \text{conditional class probability}\end{aligned}$$

where

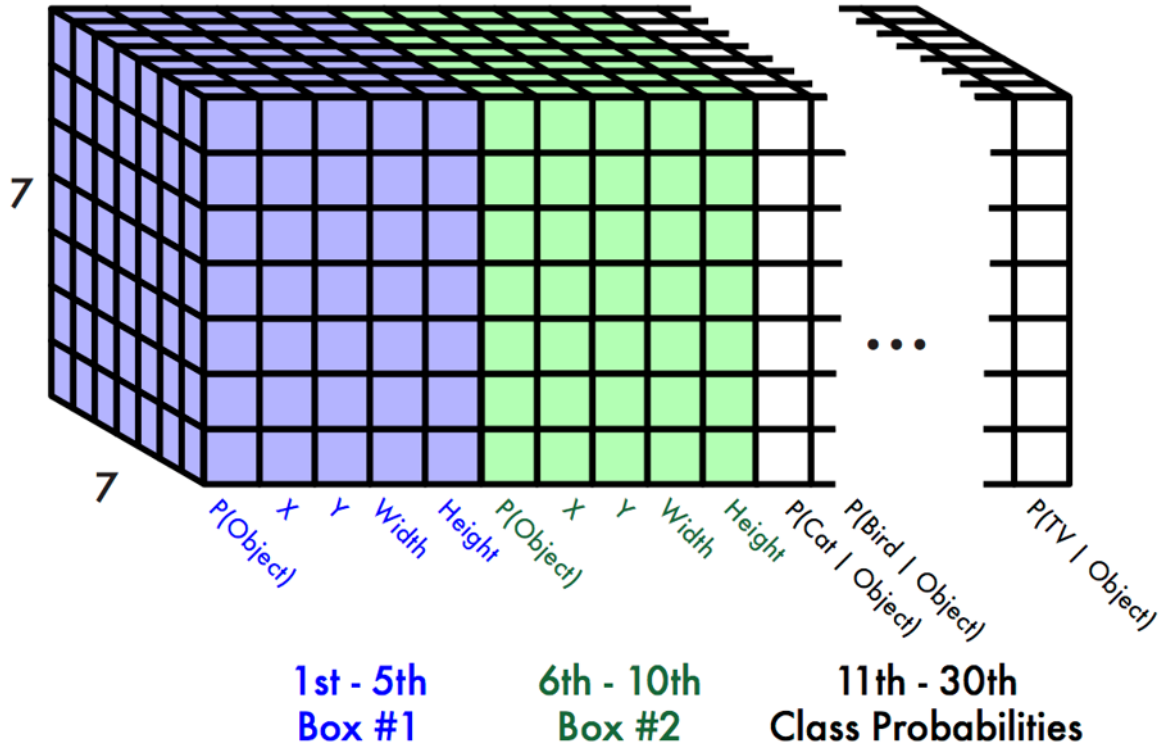
$P_r(object)$ is the probability the box contains an object.
IoU is the IoU (intersection over union) between the predicted box and the ground truth.
$P_r(class_i|object)$ is the probability the object belongs to $class_i$ given an object is presence.
$P_r(class_i)$ is the probability the object belongs to $class_i$

YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

- Each Cell:
  - C probability (# of possible objects), One object per Cell!
  - Responsible for B (=2) Bounding Box
- One image ~ S ×S × B bounding boxes
- Each Bounding Box: 4-touple+1 Confidence Score+C Probability
- The final layer: S×S×(5B+C) ~ 7×7× (5×2+20) =7×7×30 Tensor

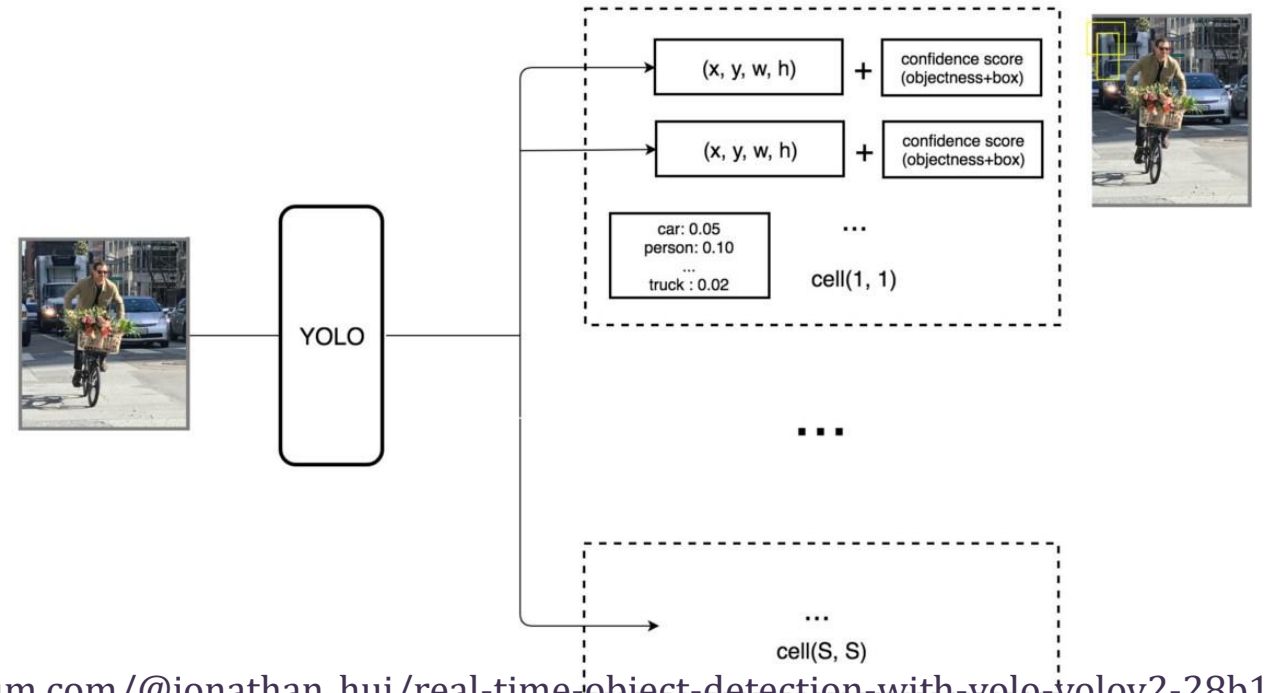
YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

› Output Tensor:



YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

› General View:

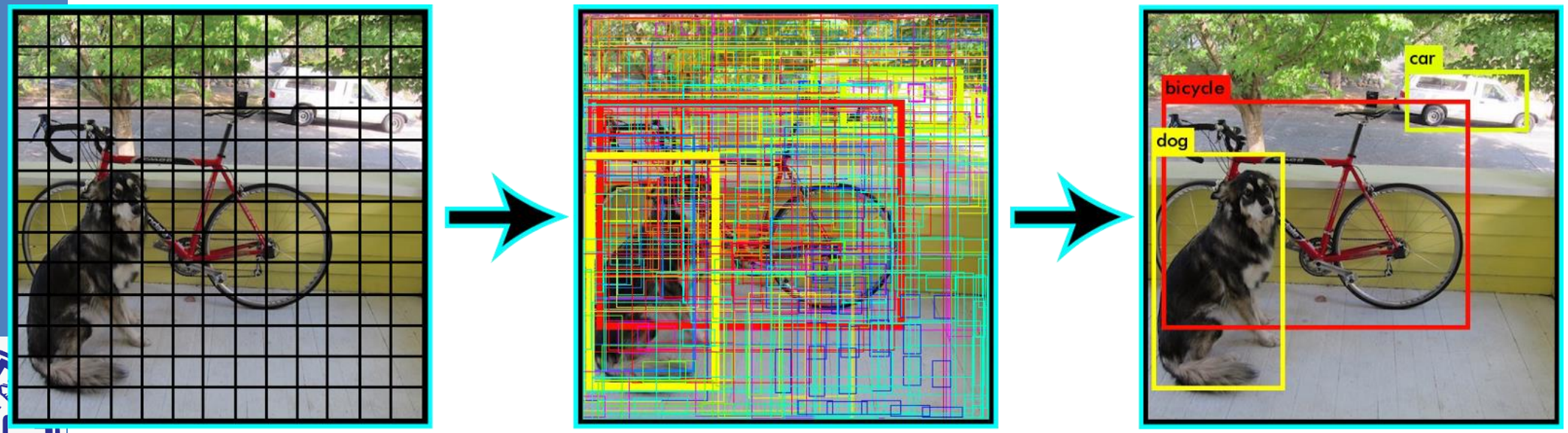


› https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

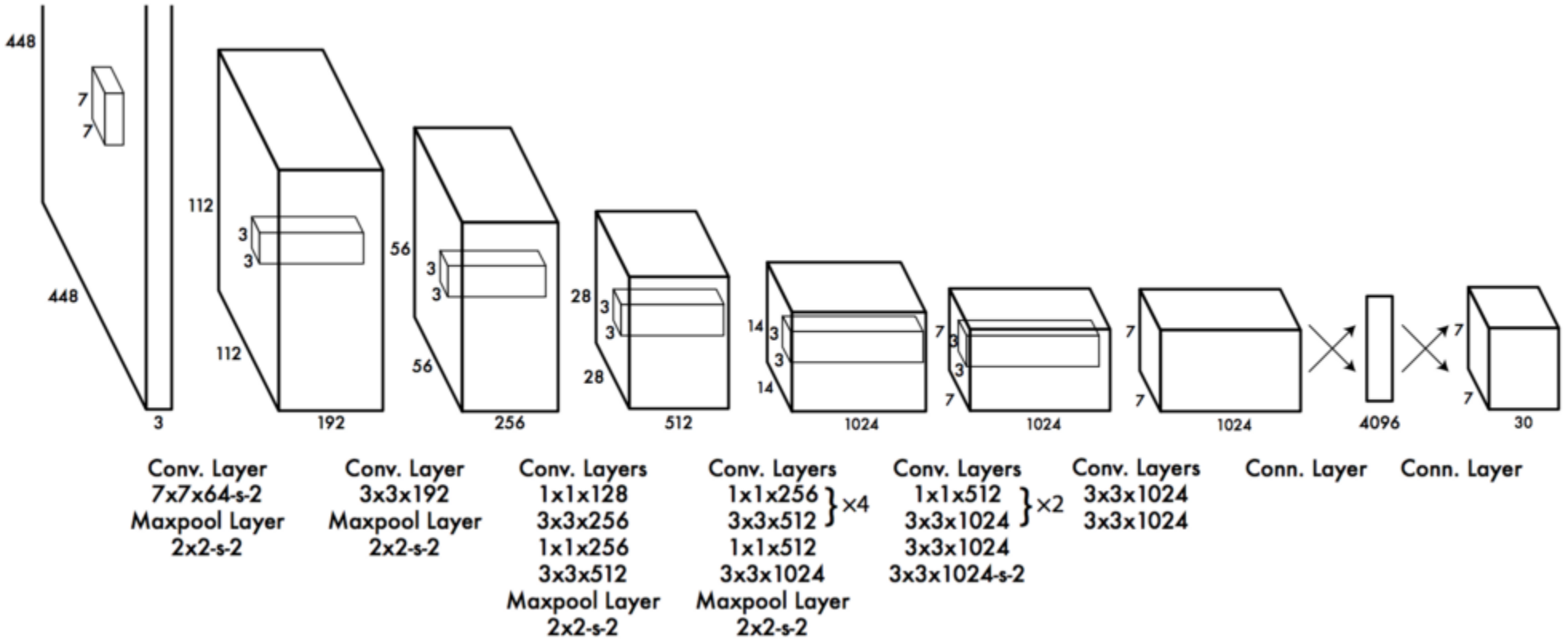
› Two FCs in last layers will produce 7×7×2 (predictions)

› Final prediction: keep those with high box confidence scores (≥0.25)



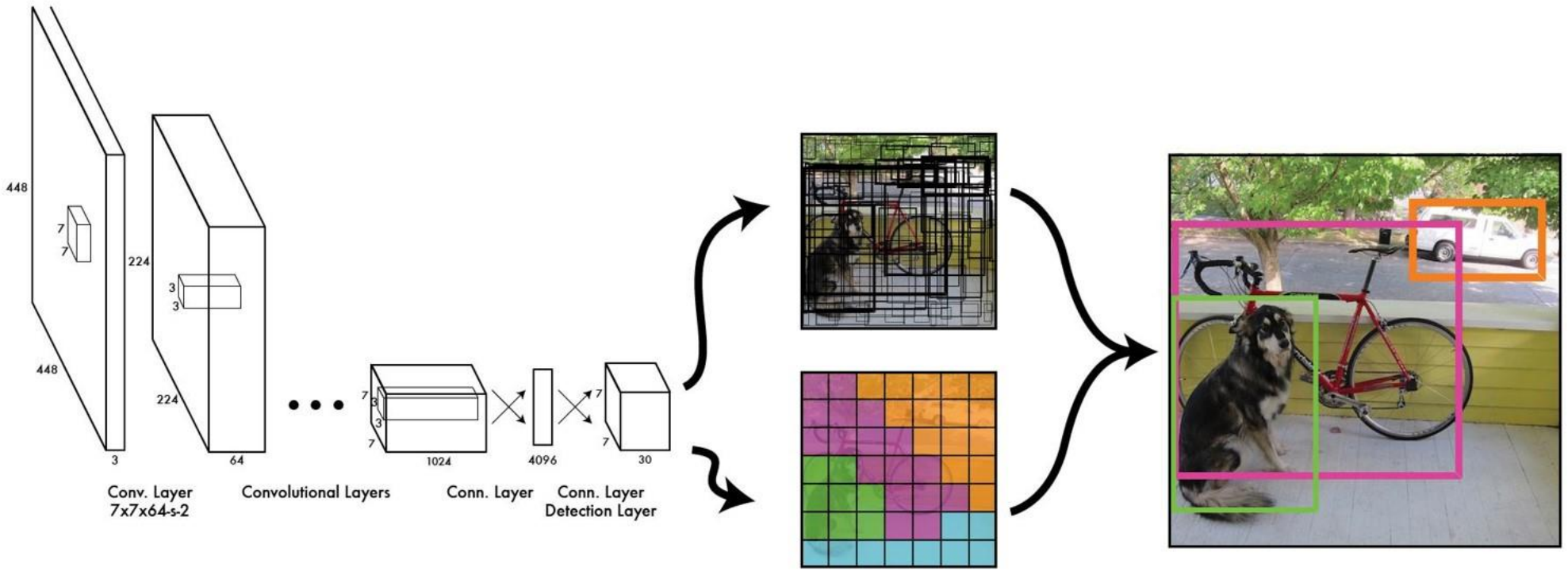
YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

› YOLO Architecture#1 (paper):



YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

› YOLO Architecture#2 (CVPR'16 Presentation):



YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

› Loss function(s):

› 1) Classification Loss:

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

where

$\mathbb{1}_i^{obj} = 1$ if an object appears in cell i, otherwise 0.

$\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.

YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

› Loss function(s):

› 2) Localization Loss:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

where

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

› Loss function(s):

› 3-1) Confidence Loss (for detected object):

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

where

$\hat{C}_i$ is the box confidence score of the box j in cell i.

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

› Loss function(s):

› 3-2) Confidence Loss (for not detected object):

$$\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

where

$\mathbb{1}_{ij}^{noobj}$ is the complement of $\mathbb{1}_{ij}^{obj}$.

$\hat{C}_i$ is the box confidence score of the box j in cell i.

λ_{noobj} weights down the loss when detecting background.

YOLO You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

› Total Loss function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$