Computer Vision and CNN

E. FATEMIZADEH

FALL 2023

Computer Vision Problems

- Classification
- Semantic Segmentation
- Object Detection
- Instance Segmentation
- Panoptic Segmentation (Semantic+ Instance Segmentation)

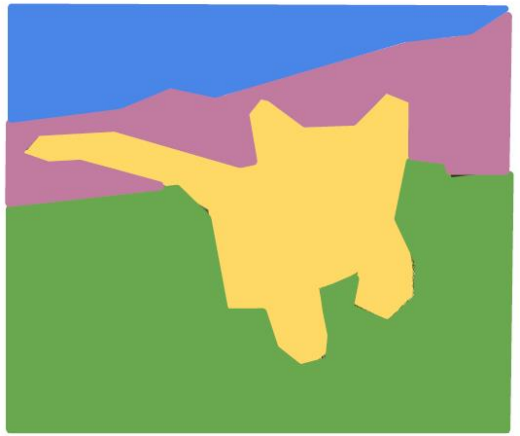
Computer Vision Problems

Classification

CAT

No spatial extent

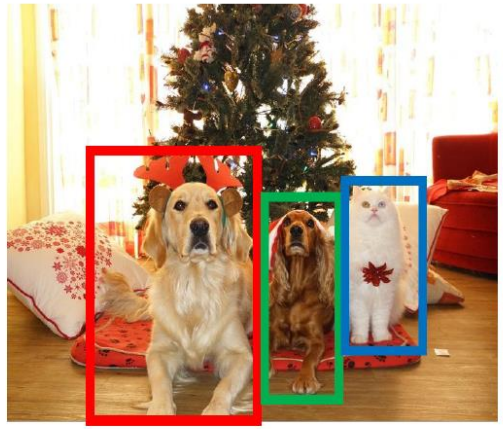
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



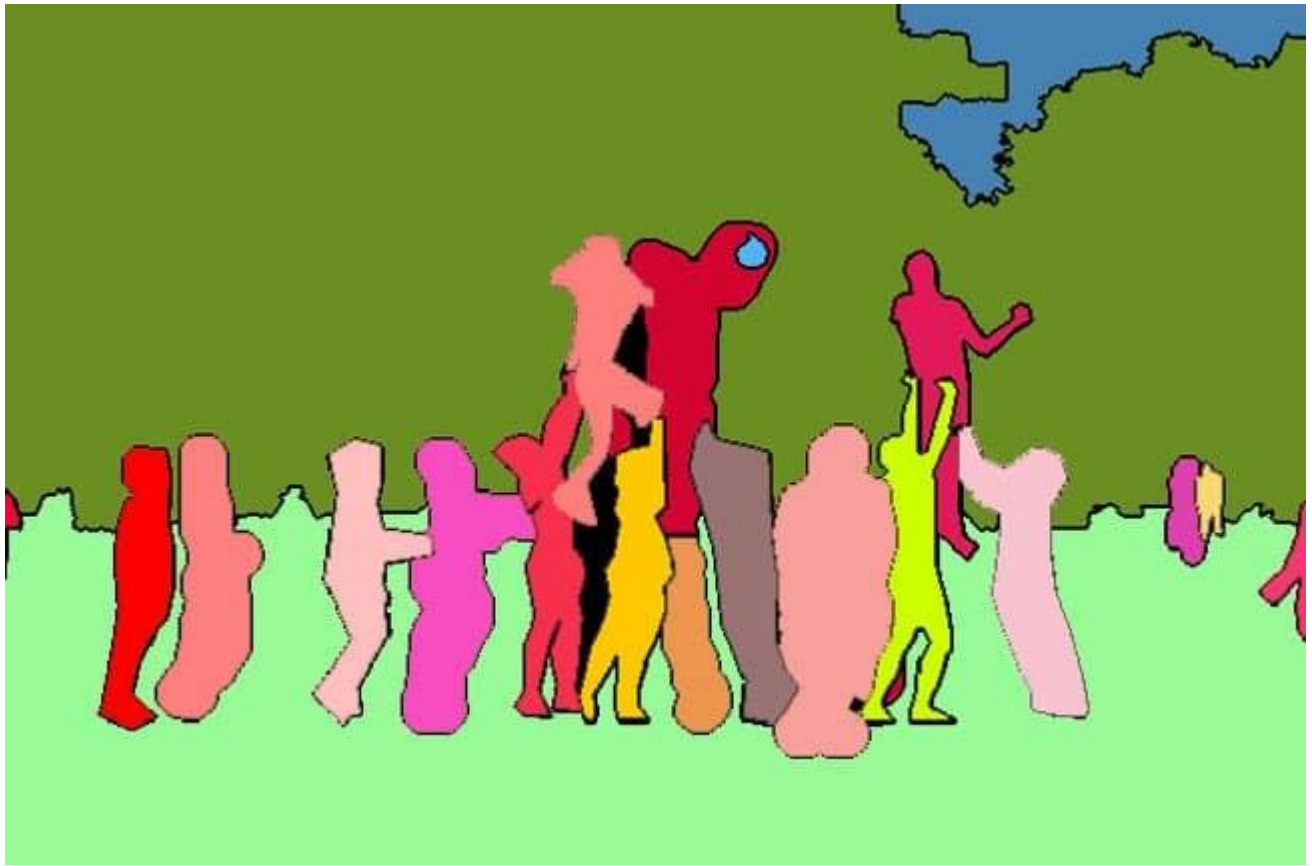
DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

Multiple Object

Panoptic Segmentation



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

size doesn't matter

- 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

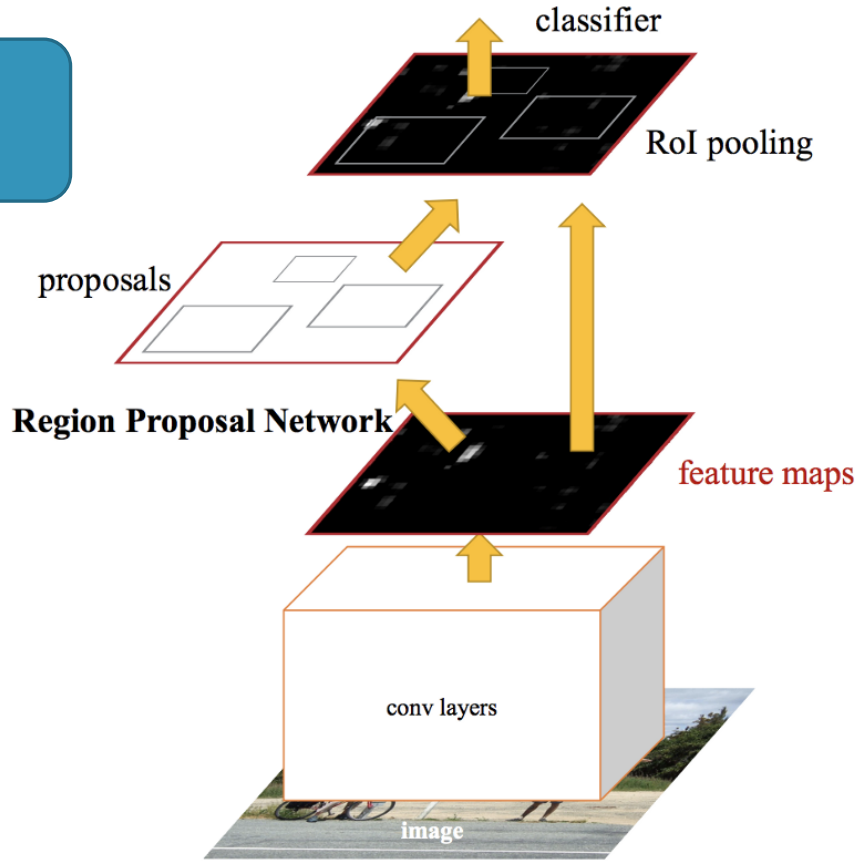
FASTER R-CNN: TOWARDS REAL-TIME OBJECT DETECTION WITH REGION PROPOSAL NETWORKS, NIPS2015

Faster R-CNN

- 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

Architecture



Faster R-CNN

- 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.

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

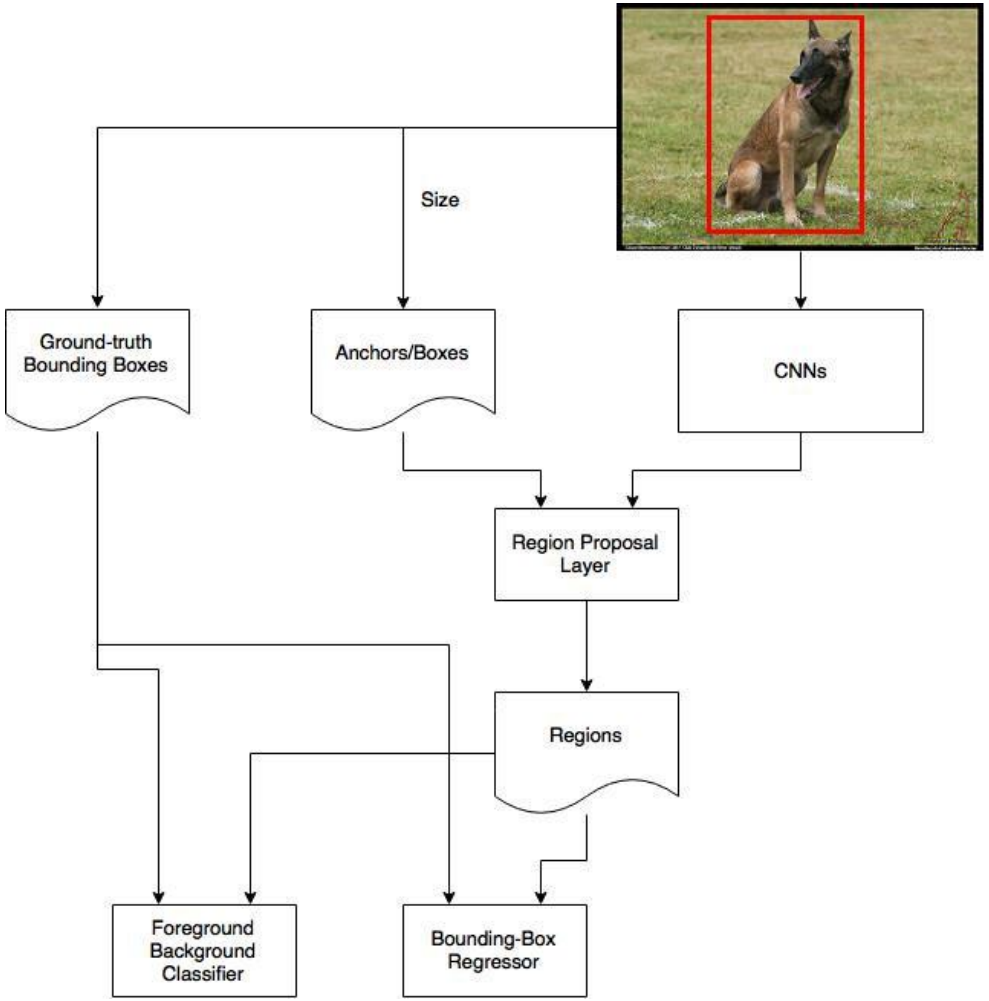
in which

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

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

Faster R-CNN

- Region Proposal Network (RPN):

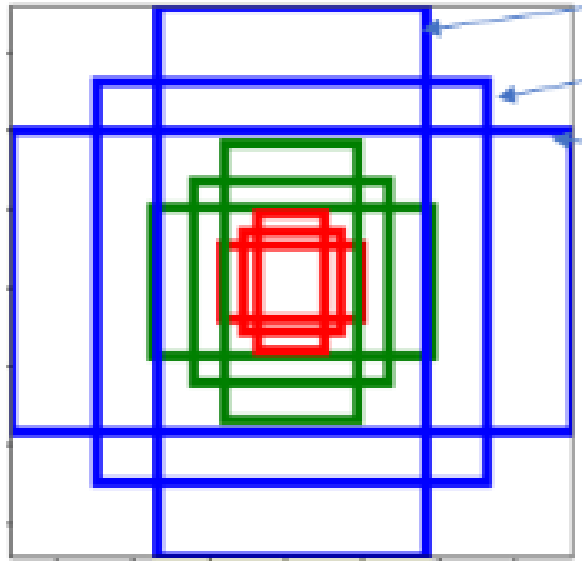


Faster R-CNN

- 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

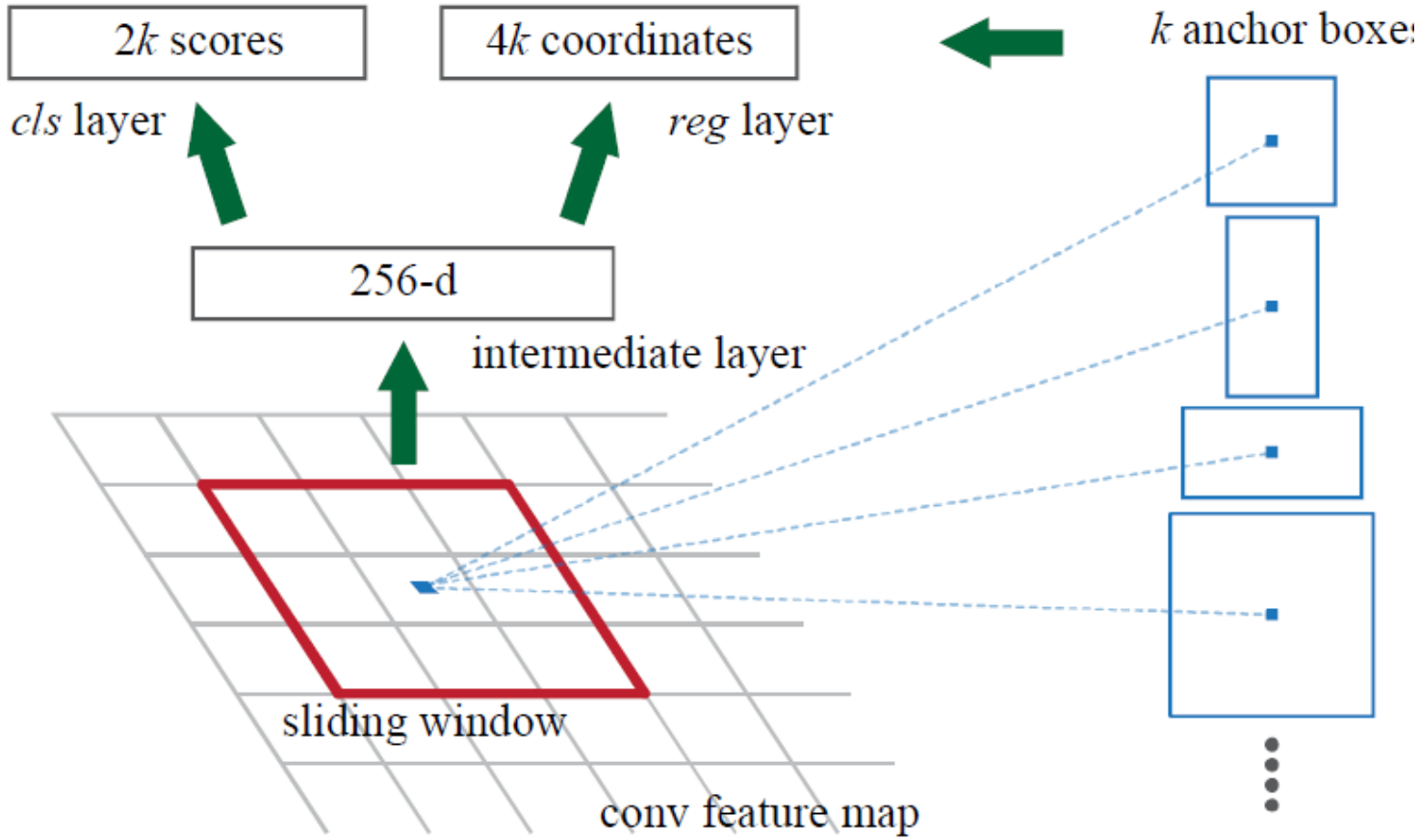
به از

به ازای هر نقطه روی لایه ویژگی ۹ تا کاندید در نظر می‌گیرد



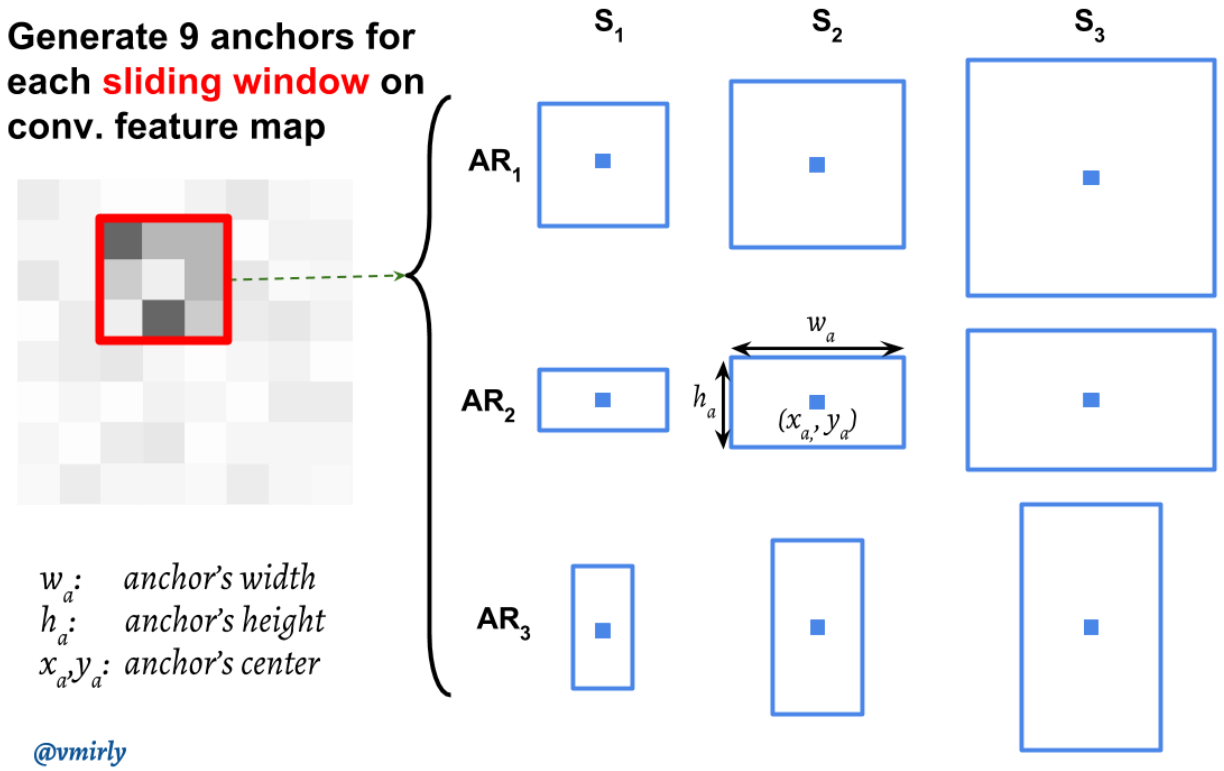
Faster R-CNN

- RPN training (k=9):



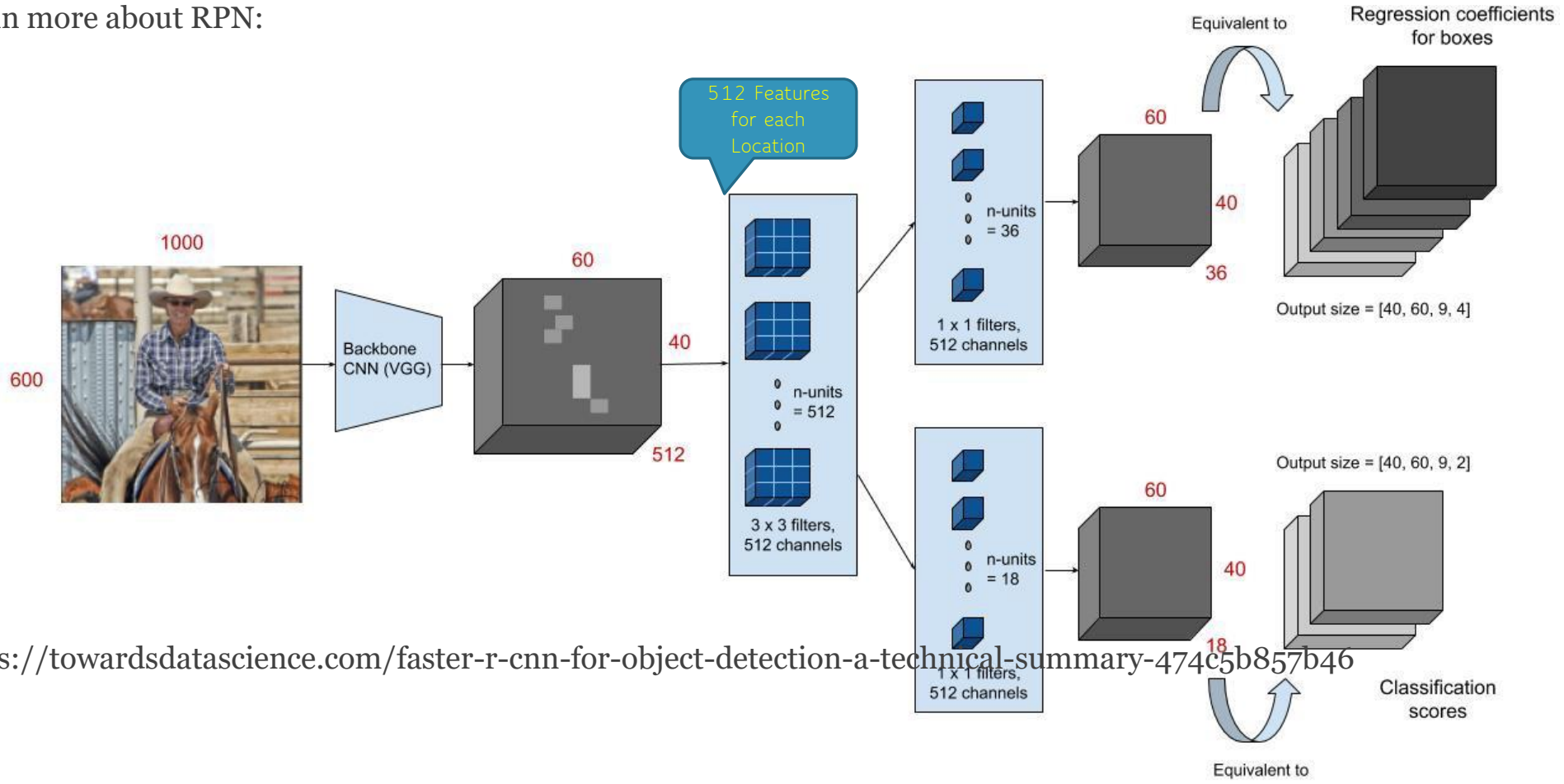
Faster R-CNN

- More About RPN
- Anchor are possible objects position!



Faster R-CNN

- Again more about RPN:



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

Faster R-CNN

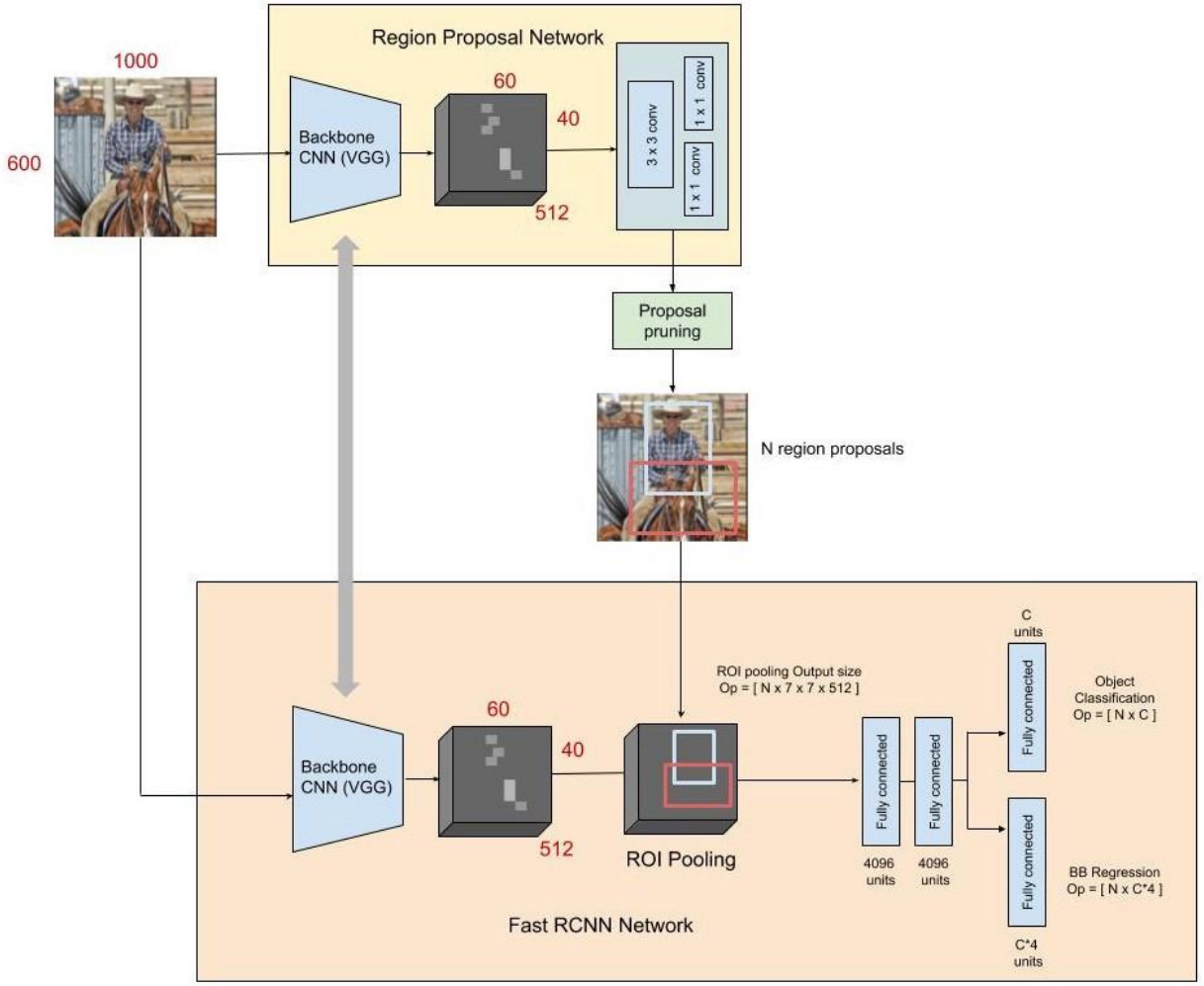
- Training (RPN):
  - 40×60×9 = 21600 anchors in total
  - Ignore anchors that cross the boundary (20k→6K)
  - Positive anchor: IoU (Intersection-Over-Union), $\frac{\text{True}\cap\text{Estimated}}{\text{True}\cup\text{Estimated}} > 0.7$
  - Negative anchor: IoU (Intersection-Over-Union), $\frac{\text{True}\cap\text{Estimated}}{\text{True}\cup\text{Estimated}} < 0.3$
  - 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

- 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 (Pi)
  - Discard low probability boxes (<0.7)

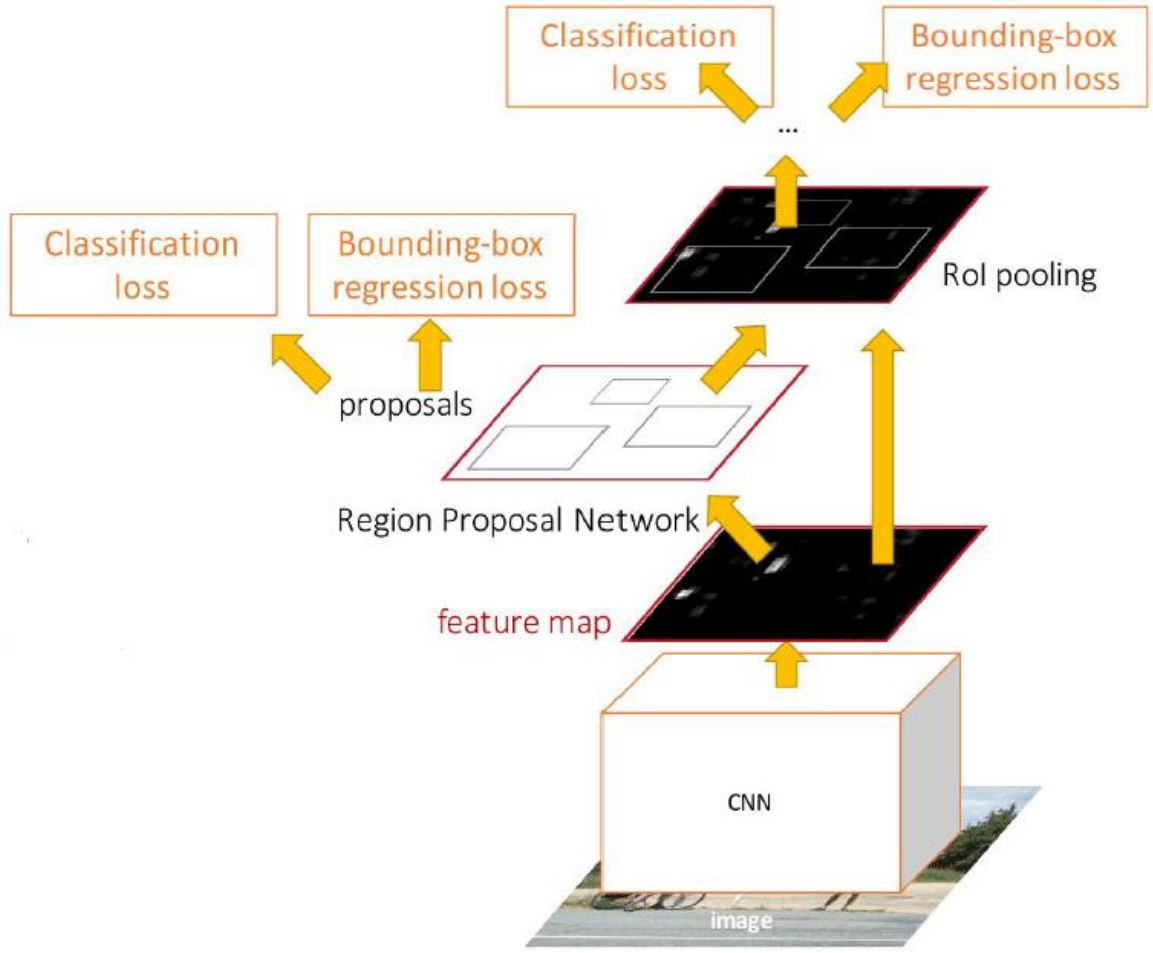
Faster R-CNN

- Object detection:
- RPN + Fast R-CNN



Faster R-CNN

- 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

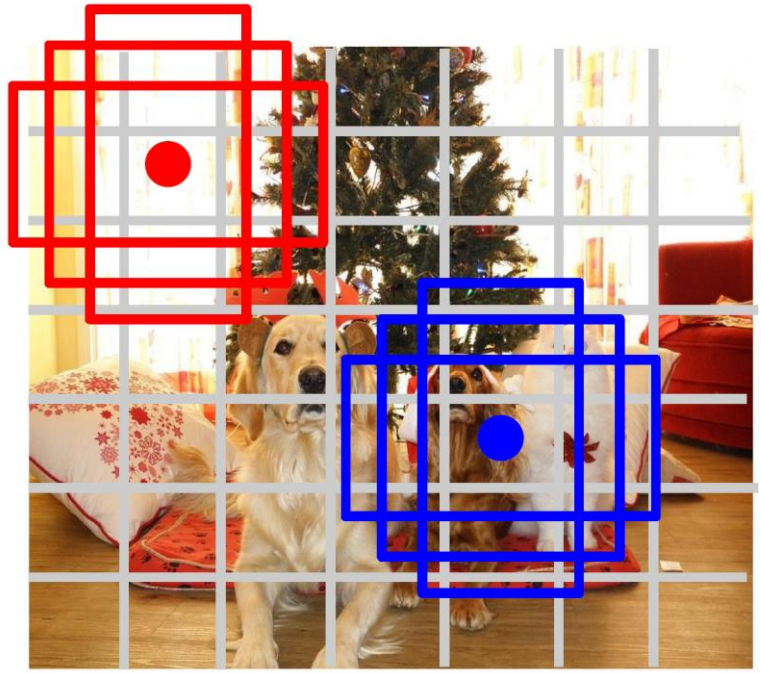


Top Known Model: YOLO famliy

- You Only Look Once (YOLO) is a state-of-the-art, real-time object detection system.
  - YOLO(2-3-4-5-…-7)
  - YOLOX
  - PP-YOLO

YOLO General Policy

- Divide image into grid 7×7
- Image a set of base boxes (B) centered at each grid cell
- Example for B=3;
- Within each grid cell:
- Regress from each of the B base boxes to a final box wit
- $(dx, dy, dh, dw, confidence)$
- Predict scores for each of C classes (+background)
- Looks a lot like RPN, but category-specific.
- Output: $7 \times 7 \times (5B + C)$

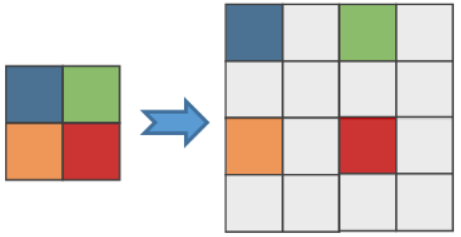


Two New Operation

- 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:

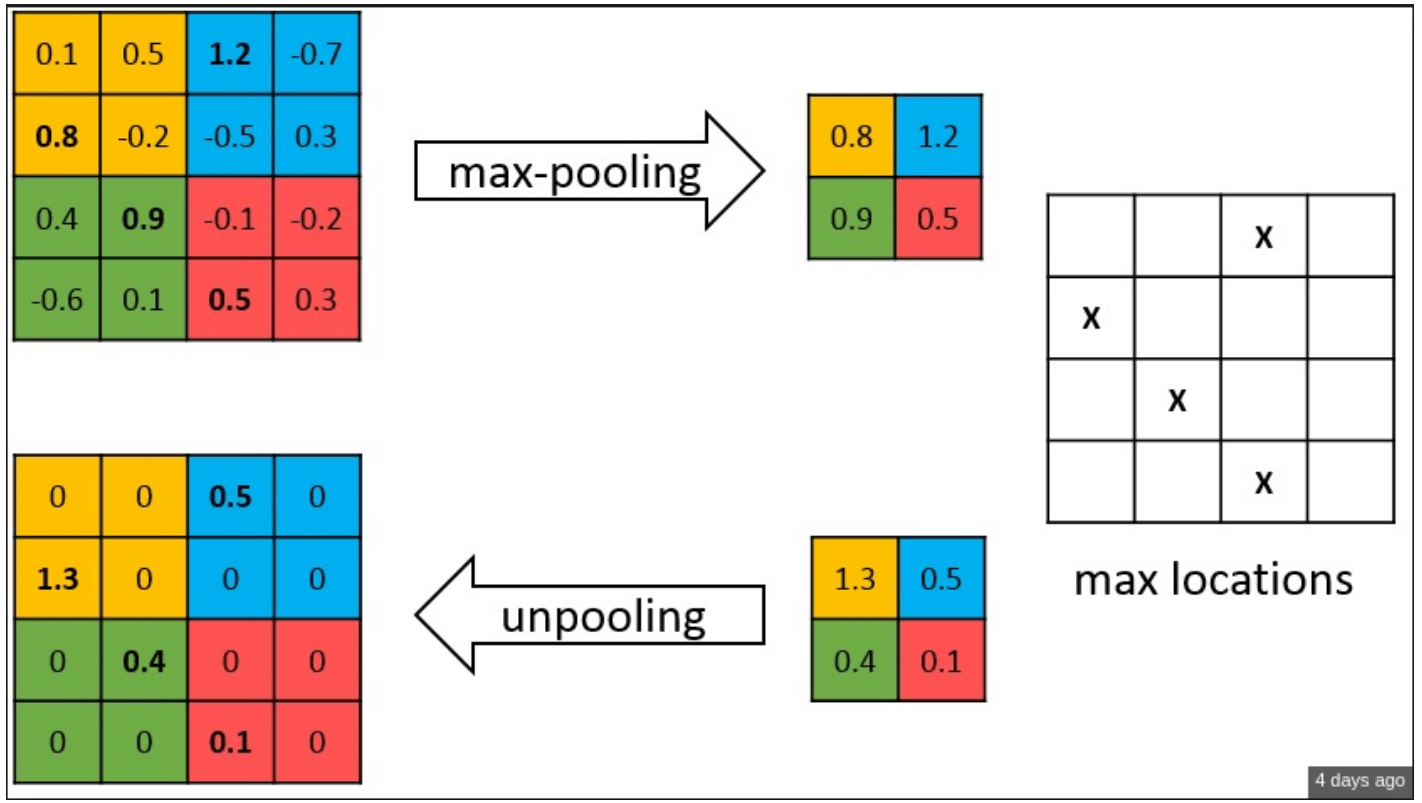
2x2 unpooling: 

  - Mean/Nearest Neighbor unpooling:

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

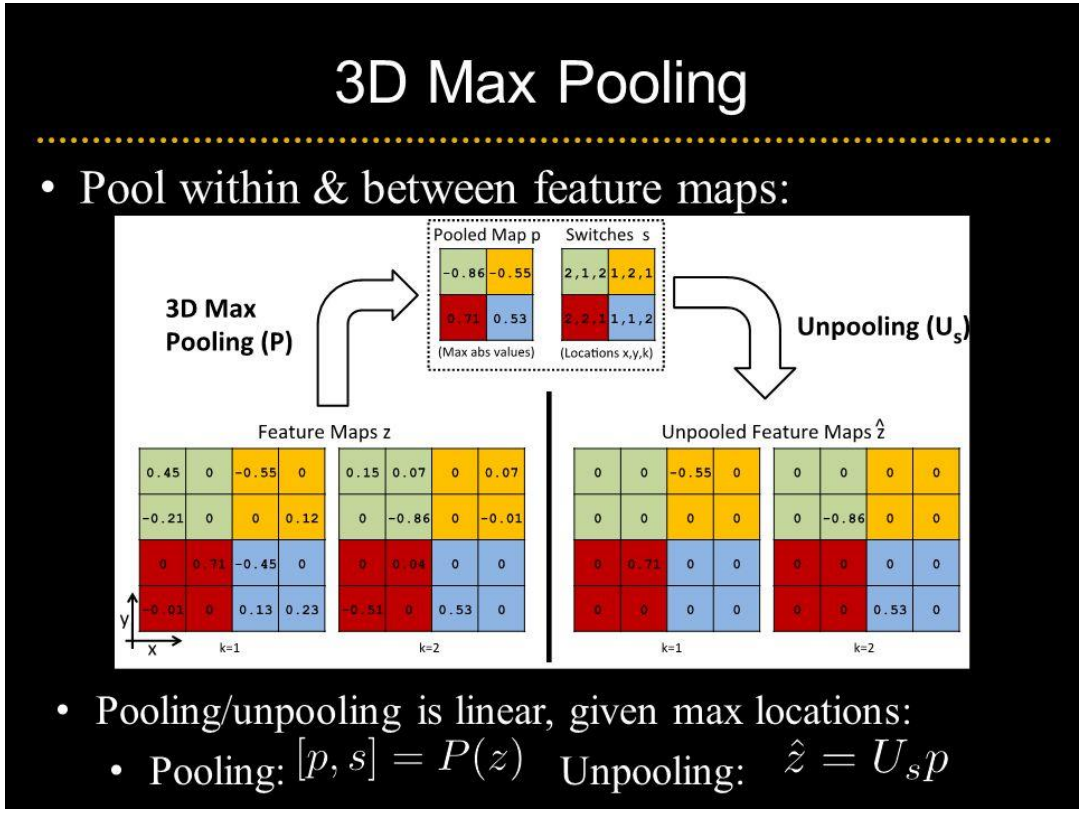
Unpooling

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



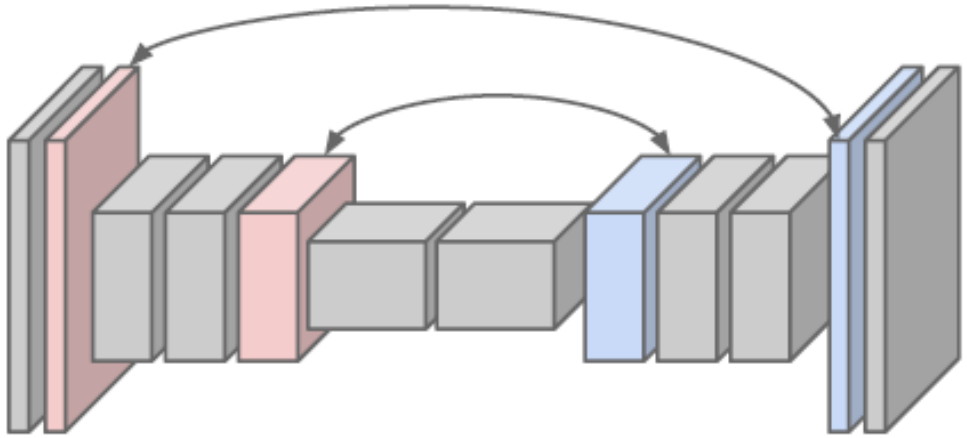
Unpooling

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



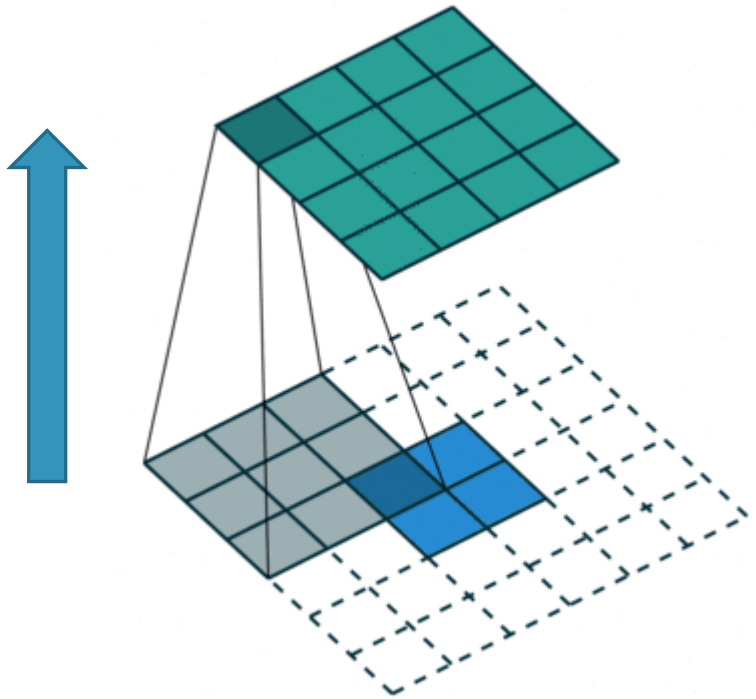
Unpooling

- 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

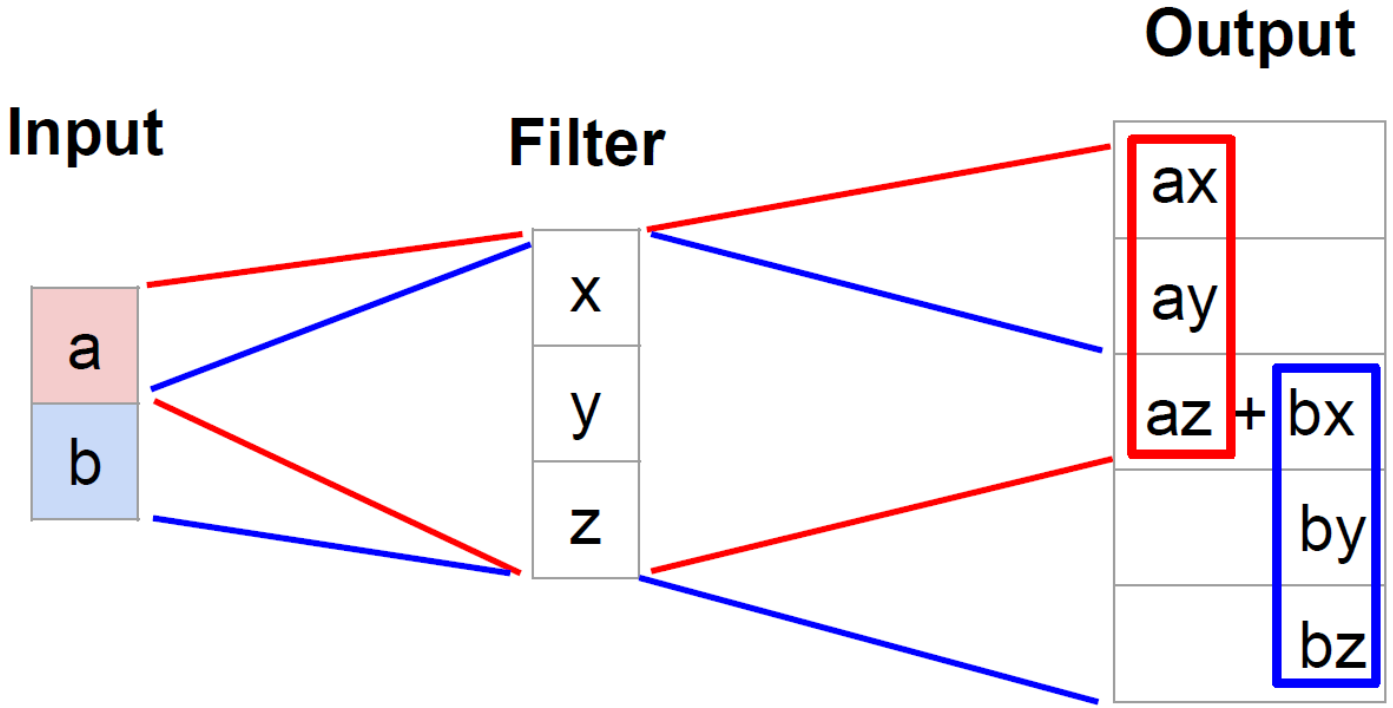
- Why Transpose Convolution:
  - Remember: Convolution ~ Matrix Multiplication: $y = Mx$ (Left, n=3, s=1, p=1)
  - Transpose Convolution: is also Matrix Multiplication: $z = M^T t$ (Right, s=1)

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

  - How to upsample by 2?

Transpose Convolution

- Learnable Upsampling:
  - Crop output to make output size exactly 2x input size



Transpose Convolution

- Samples:

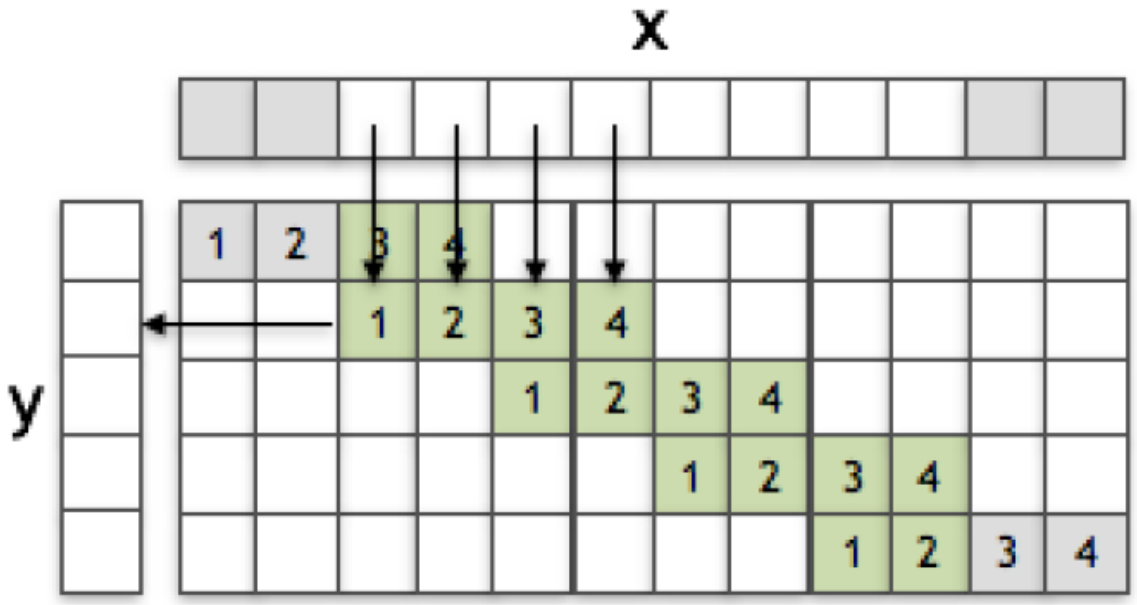


Figure 2: Convolution with stride 2 in 1D

Transpose Convolution

- Samples (Convolution):

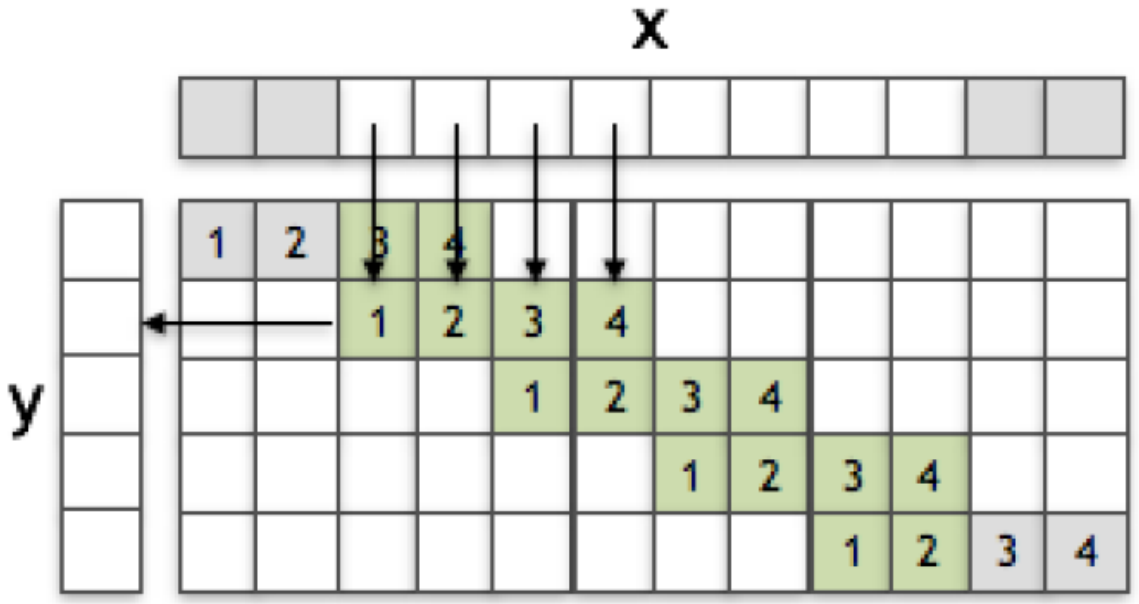


Figure 2: Convolution with stride 2 in 1D

Transpose Convolution

- Samples (Transposed Convolution):

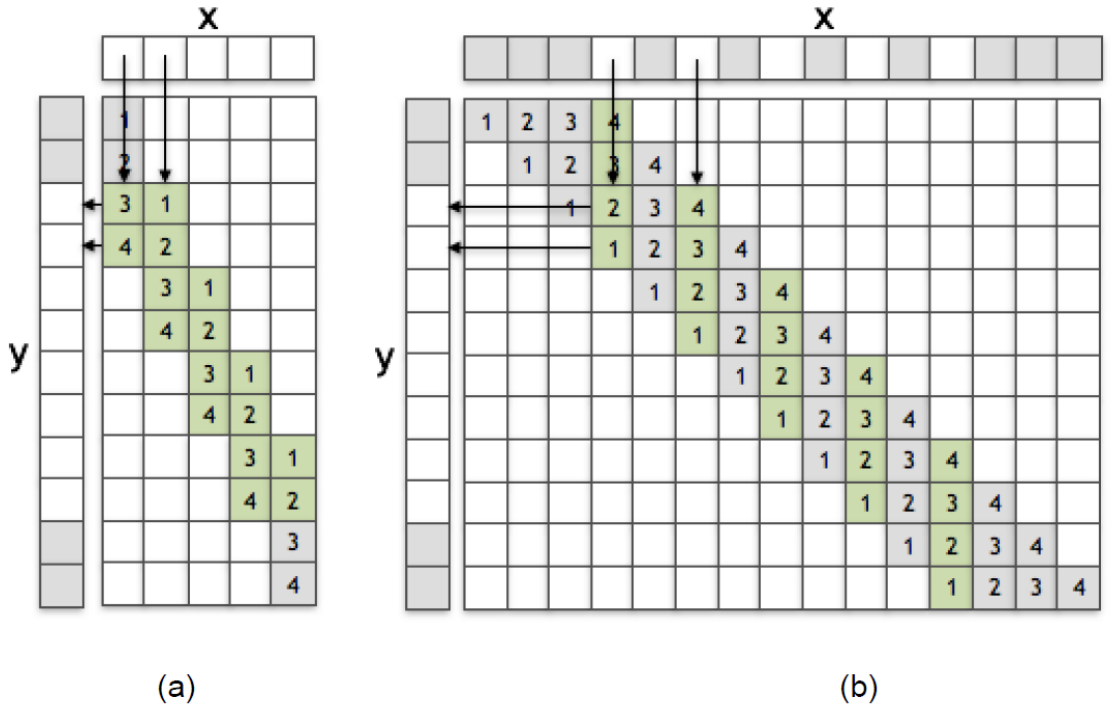
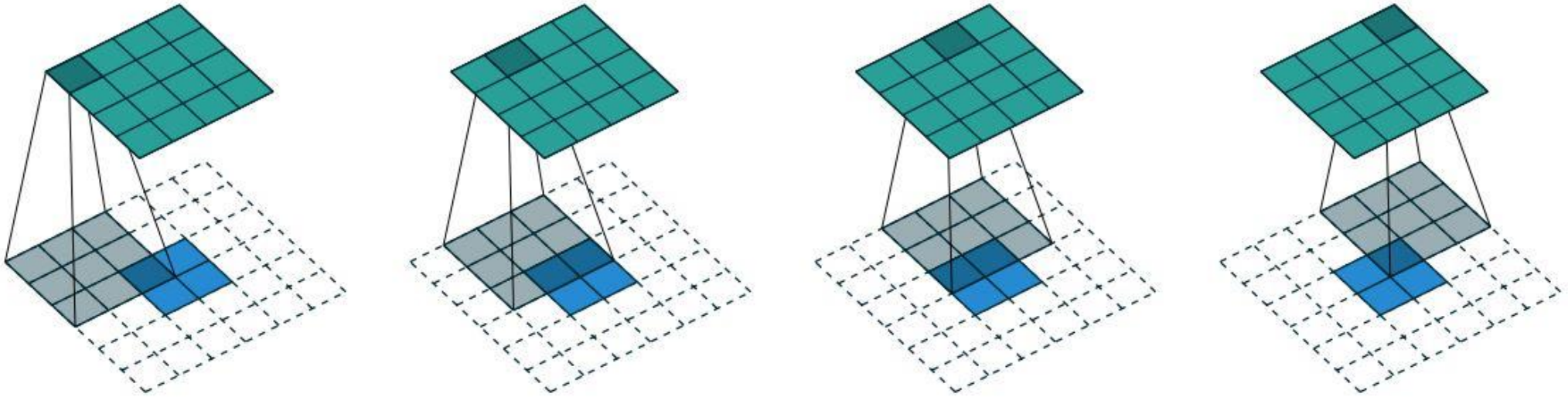
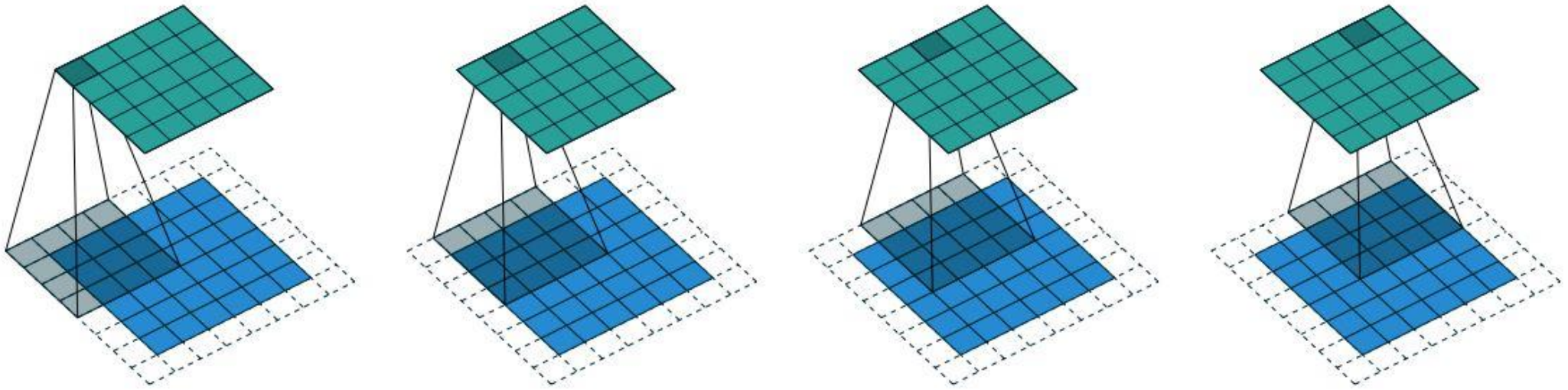


Figure 3: (a) Transposed convolution with stride 2 and (b) sub-pixel convolution with stride ½ in 1D

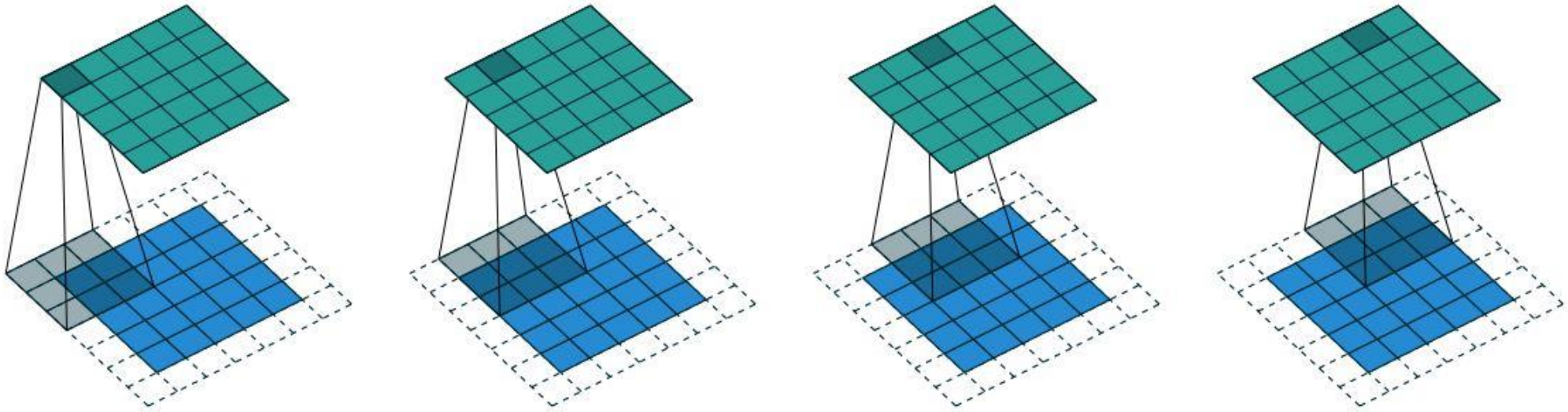
Examples - A guide to convolution arithmetic for deep learning



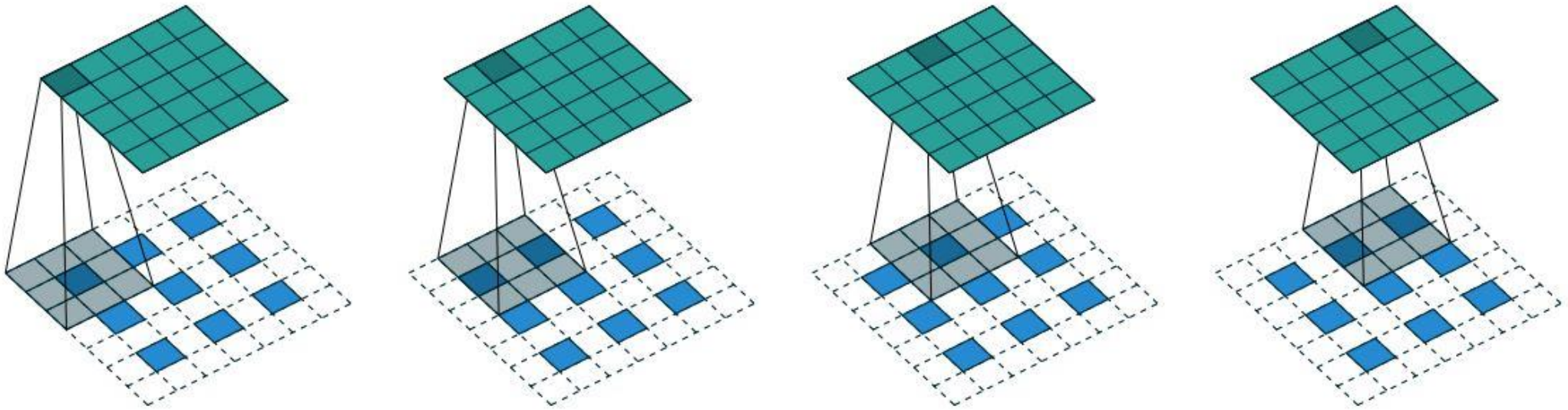
Examples - A guide to convolution arithmetic for deep learning



Examples - A guide to convolution arithmetic for deep learning



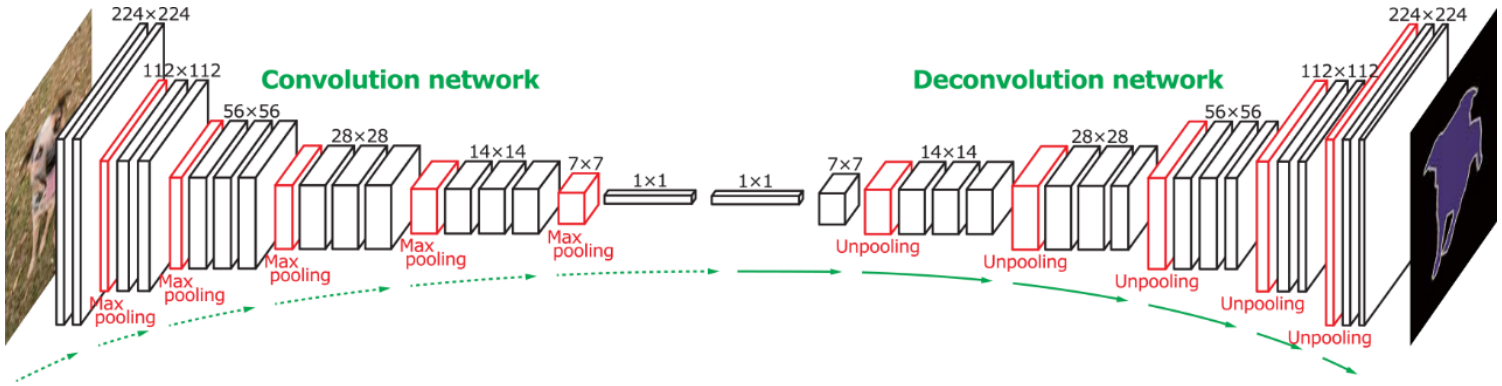
Examples - A guide to convolution arithmetic for deep learning



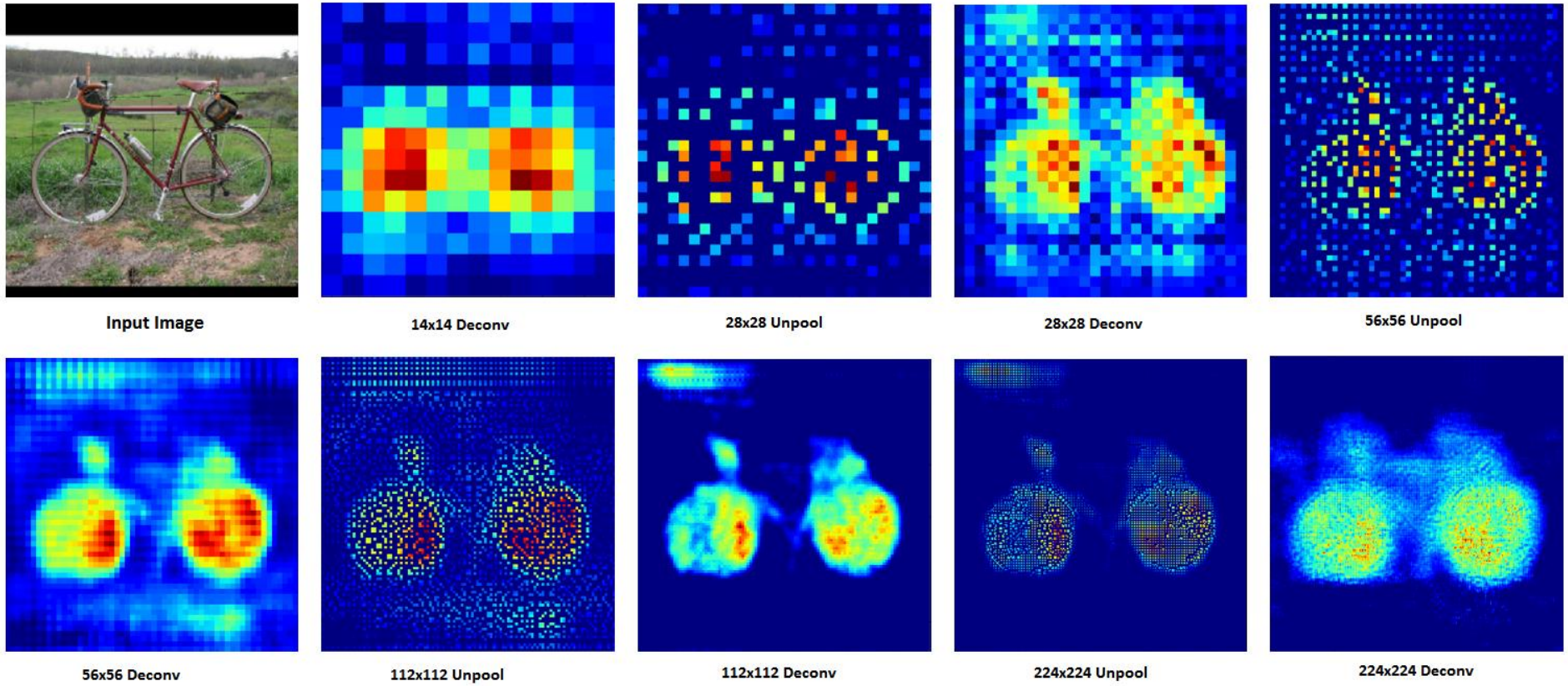
A Useful animation - Deconvolution and Checkerboard Artifacts

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

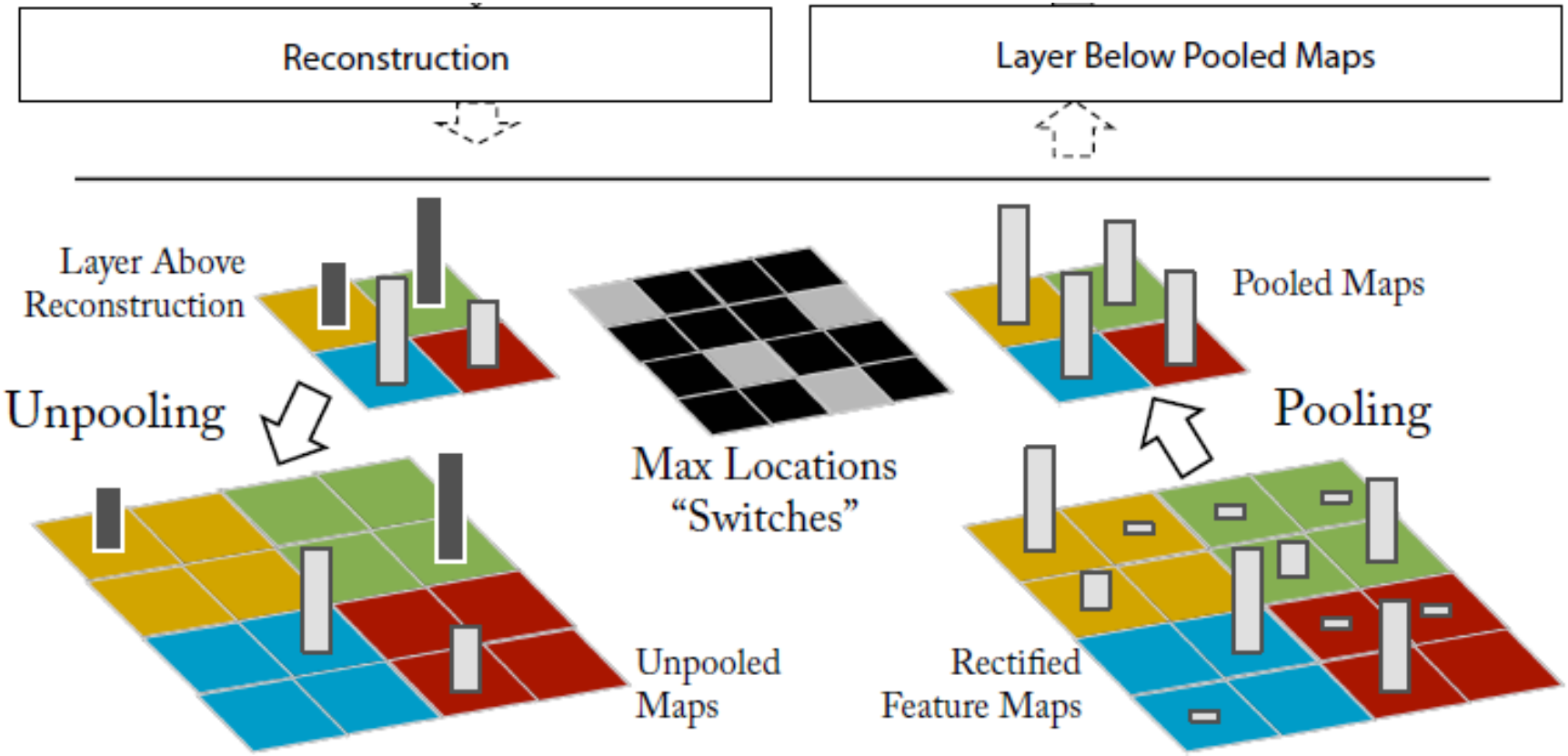
Deconvolutional Network (deconvnet)



Deconvolutional Network (deconvnet)

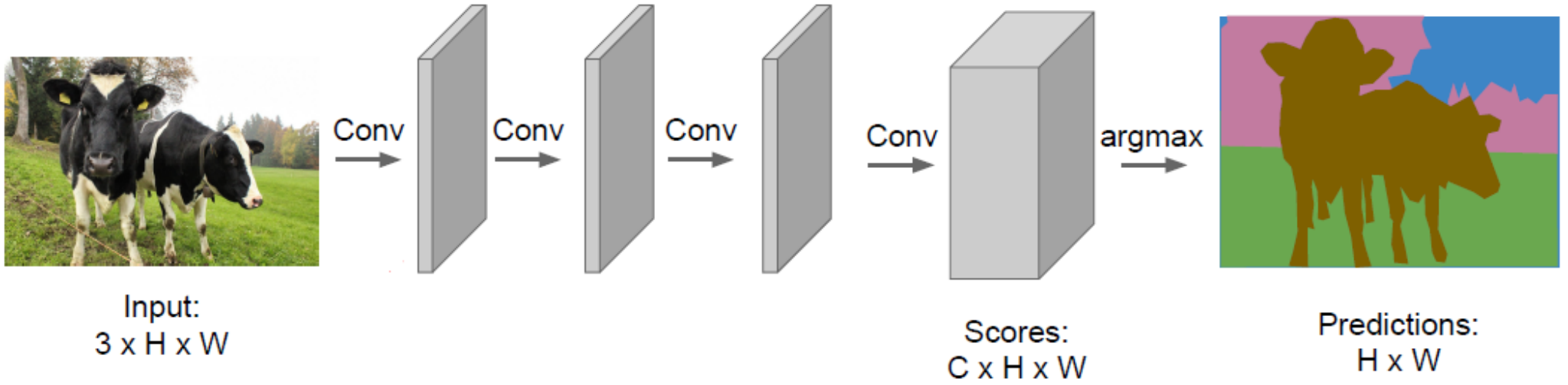


Deconvolutional Network (deconvnet)



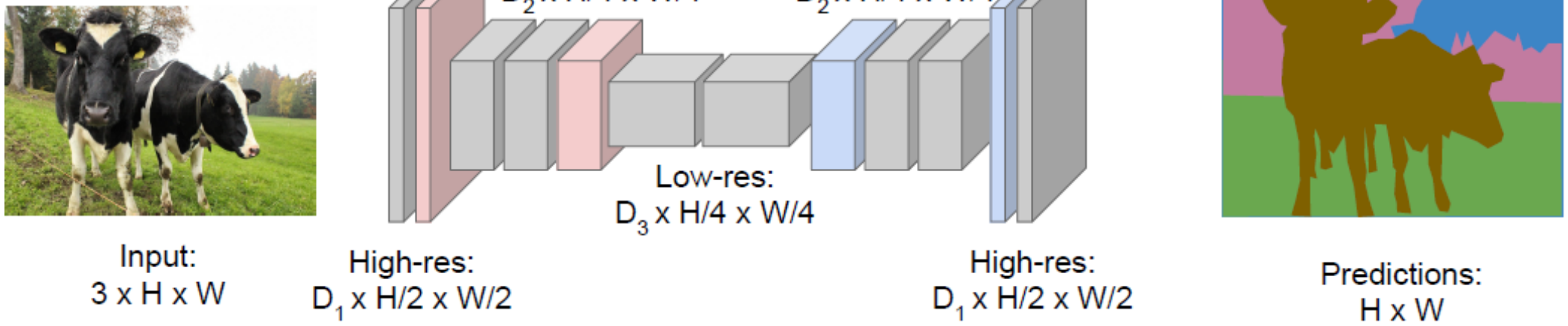
Semantic Segmentation

- Raw idea:
  - Forget FC Layer (classification) → Fully Convolutional Network (FCN)



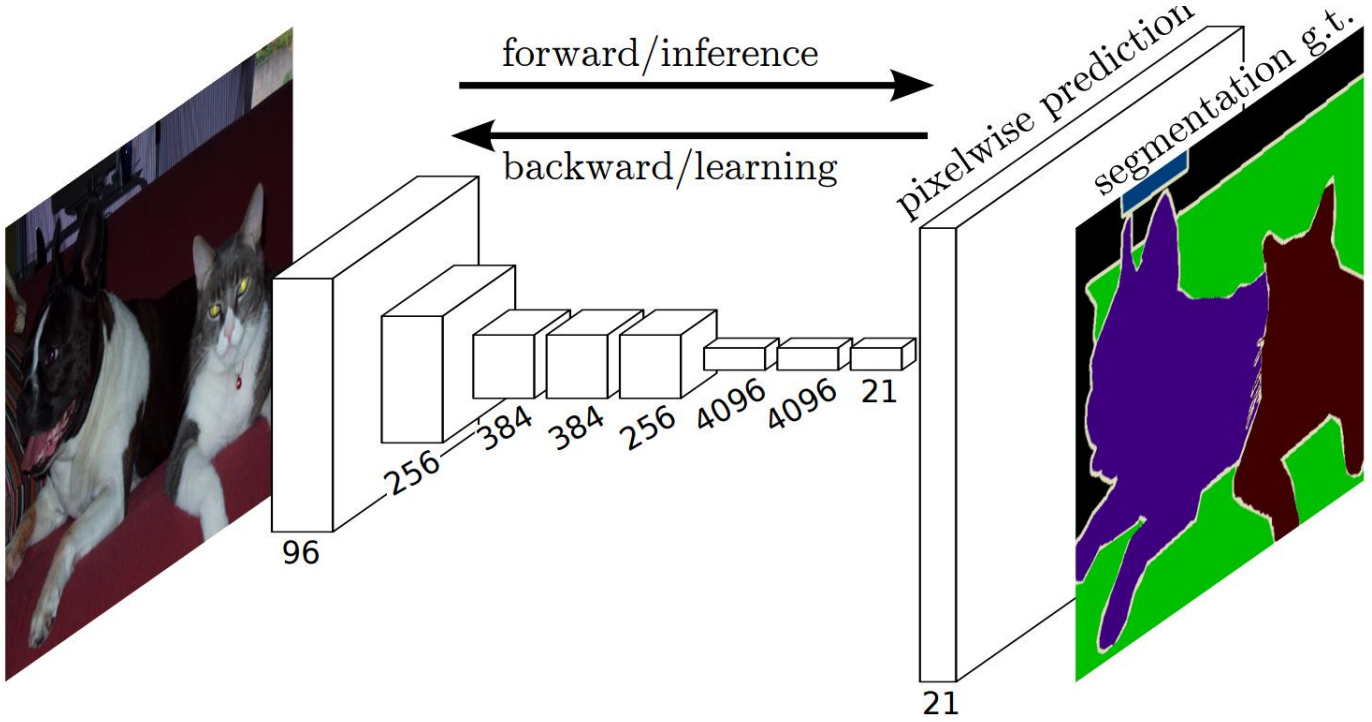
Semantic Segmentation

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



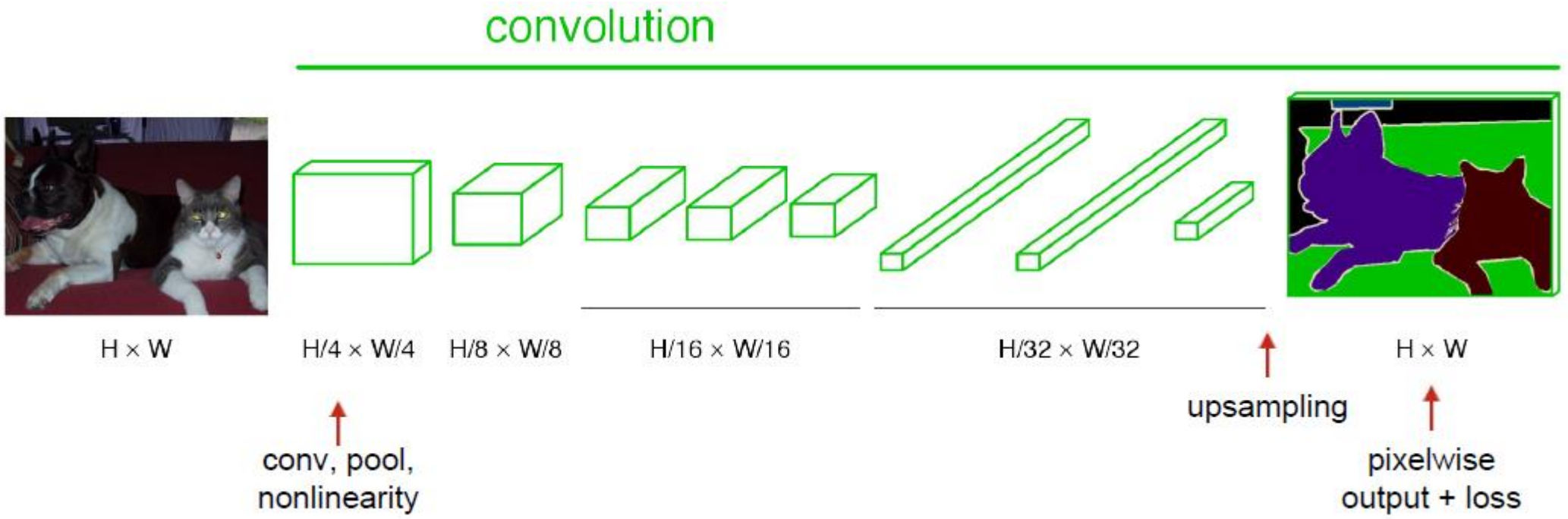
FCN (Fully Convolutional Networks)

- Replace Fully Connected Layer with Convolution Layer!



FCN (Fully Convolutional Networks)

- Replace Fully Connected Layer with Convolution Layer!

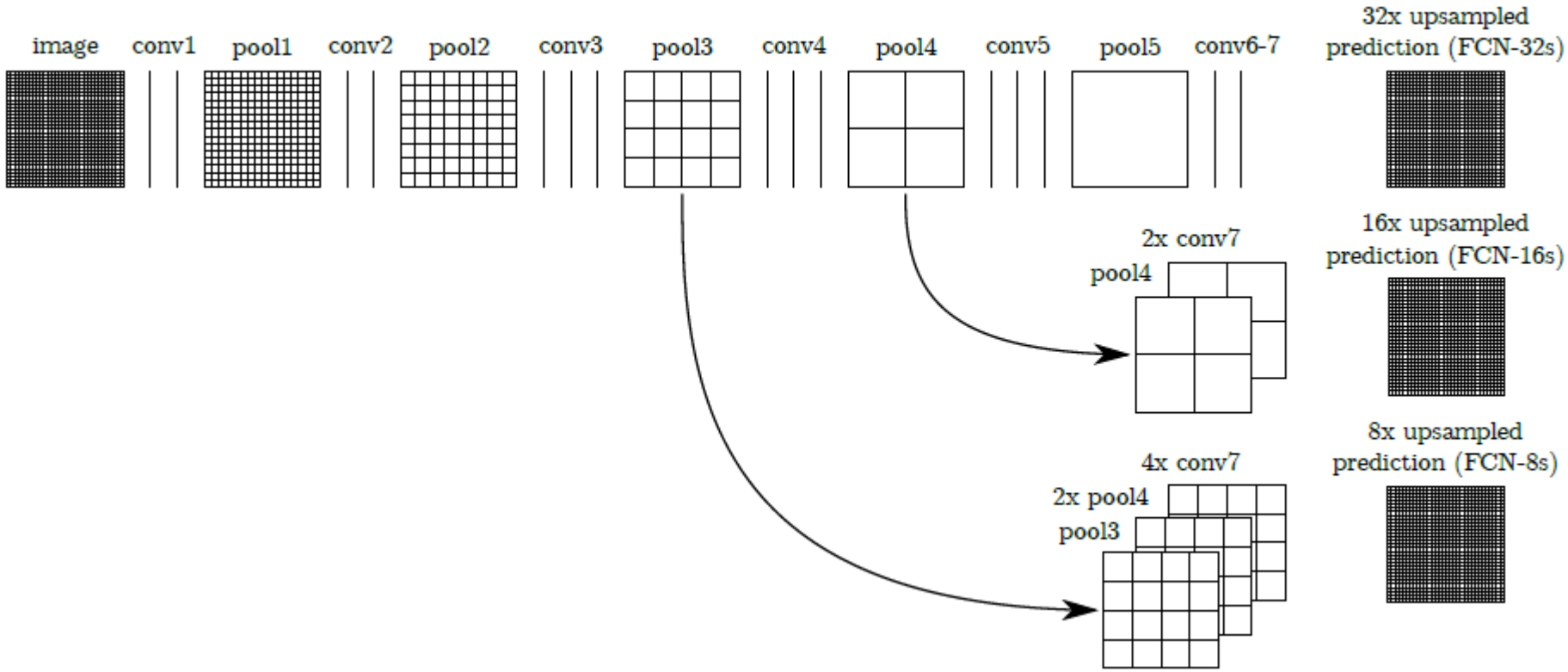


FCN (Fully Convolutional Networks)

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

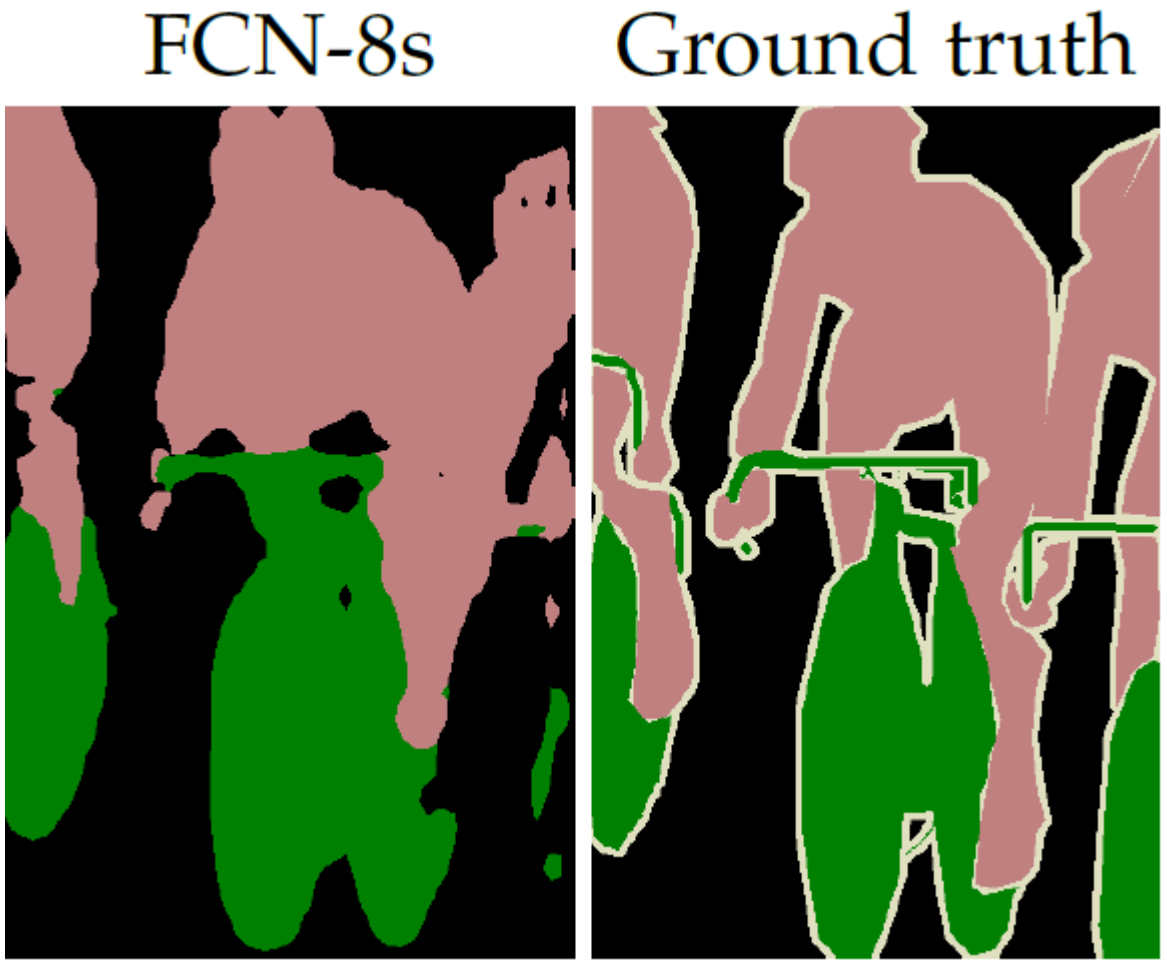
FCN (Fully Convolutional Networks)

- Skip Connection:



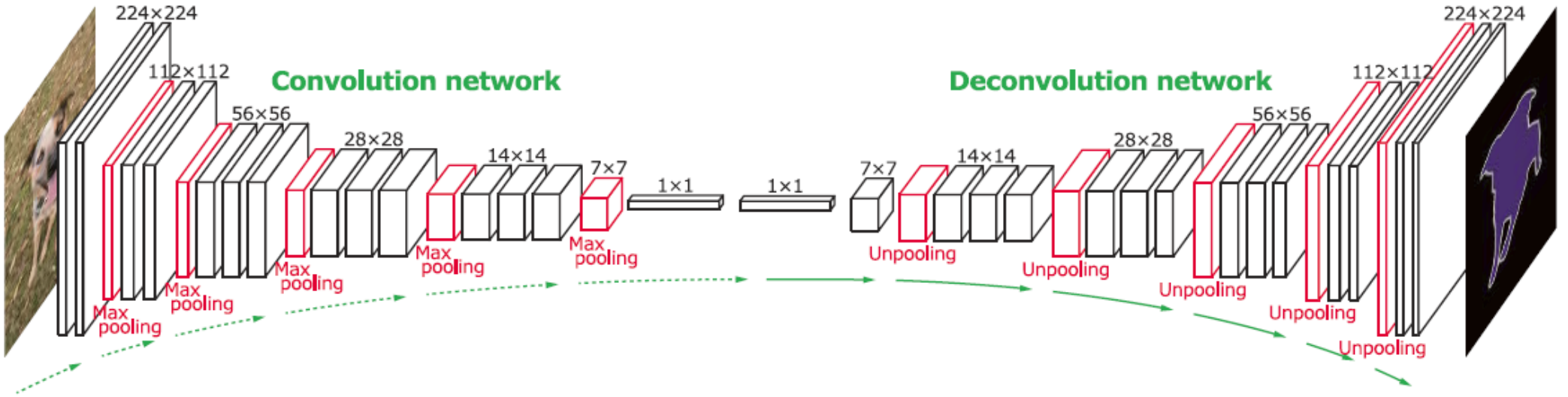
FCN (Fully Convolutional Networks)

- Sample Result:



Learning Deconvolution Network for Semantic Segmentation

- Unpooling (index map)+DeconvLayer
- Encoder+Decoder

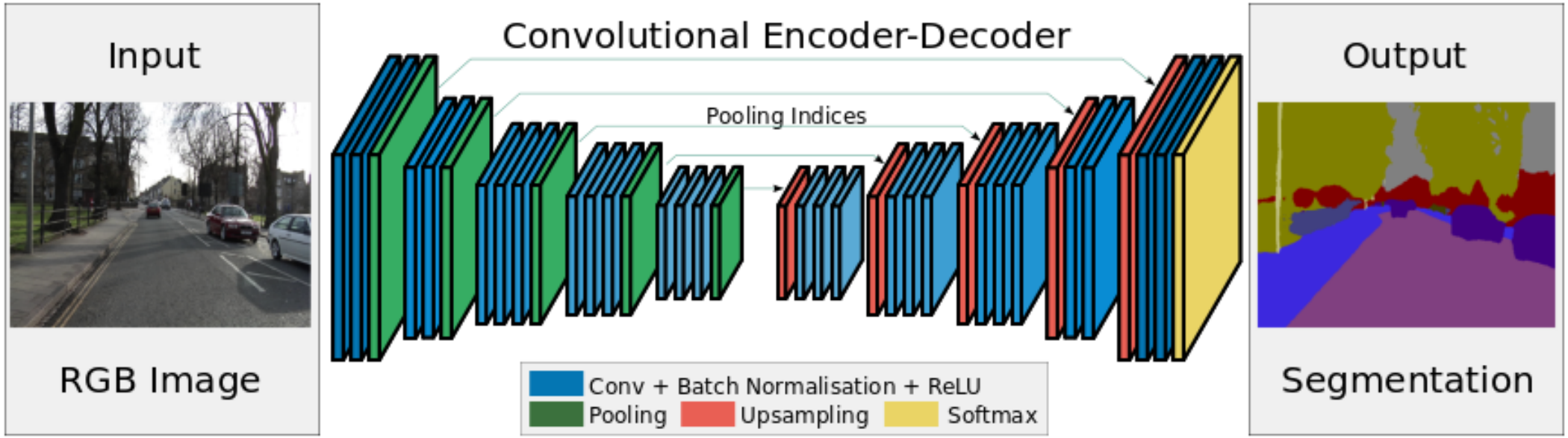


Learning Deconvolution Network for Semantic Segmentation

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

SegNet

- Architecture (Again Encoder-Decoder pair)

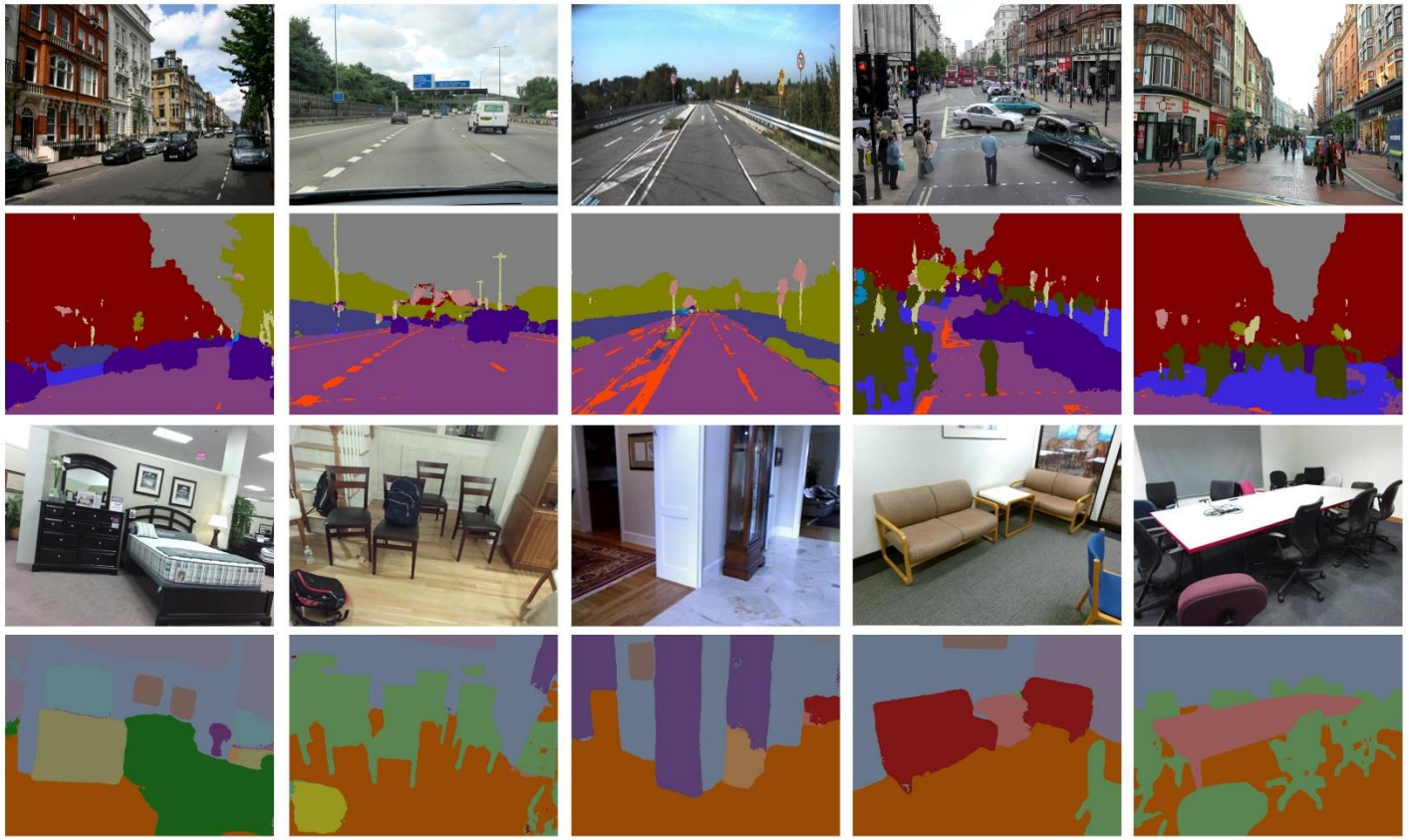


SegNet

- 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

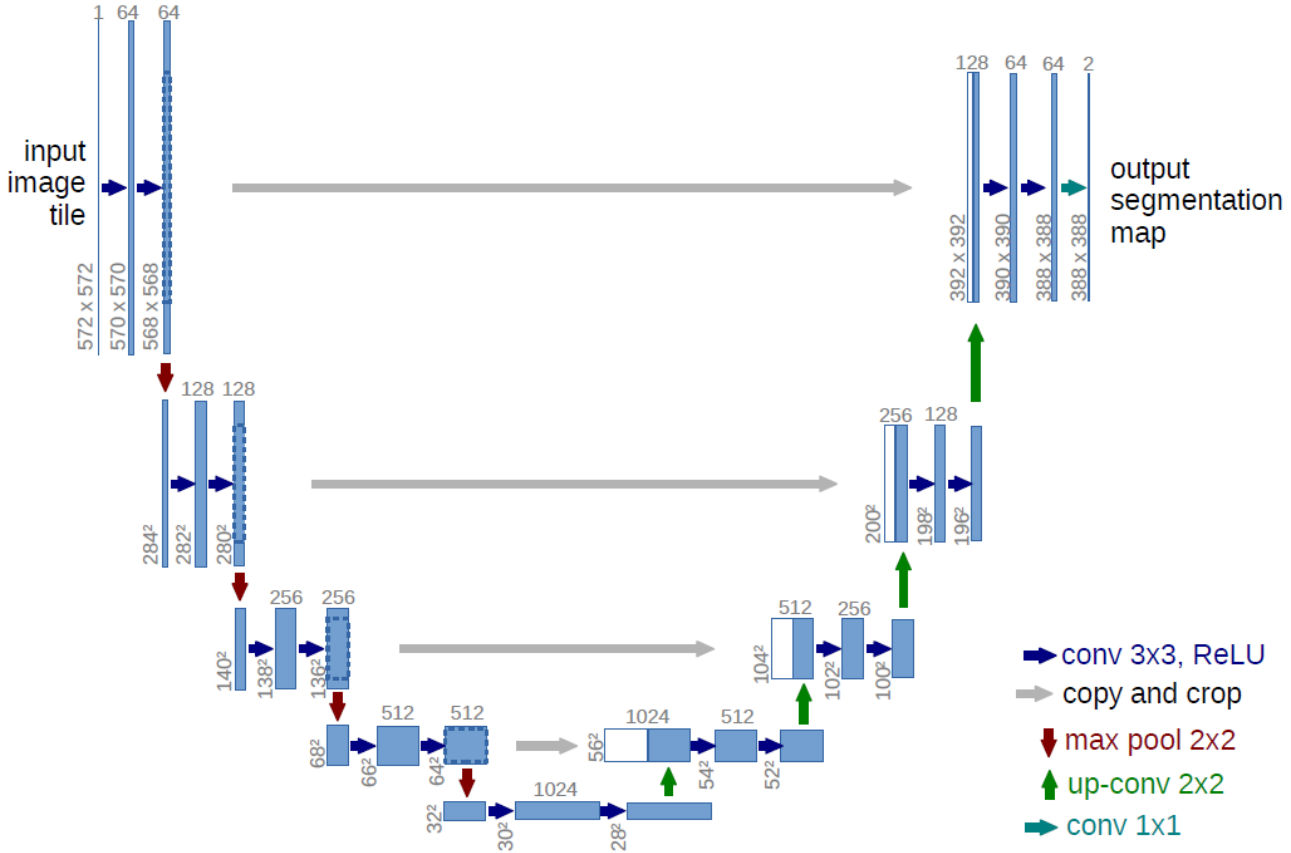
SegNet

- Sample Results:



U-Net

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



U-Net

- 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

Loss Function for image segmentation

- Problem is like data classification:
- For binary (two segments):

$$CE(p,\hat{p}) = -(p\log(\hat{p}) + (1-p)\log(1-\hat{p}))$$

- Weighted Cross Entropy (Two Classes):

$$WCE(p,\hat{p}) = -(\beta p\log(\hat{p}) + (1-p)\log(1-\hat{p}))$$

- Balanced Cross Entropy (Two Classes):

$$BCE(p,\hat{p}) = -(\beta p\log(\hat{p}) + (1-\beta)(1-p)\log(1-\hat{p}))$$

Loss Function for image segmentation

- Dice Loss (1):

$$\text{Dice}(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}, or\ \text{Dice}(A,B) = 1 - \frac{2|A \cap B|}{|A| + |B|}$$

- Dice Loss (2):

$$DL(p,\hat{p}) = 1 - \frac{2\langle p,\hat{p}\ \rangle}{\|p\|_1 + \|\hat{p}\|_1}$$

- where $\langle p,\hat{p}\ \rangle$ is the dot product of the ground truth and the prediction results matrix.

Loss Function for image segmentation

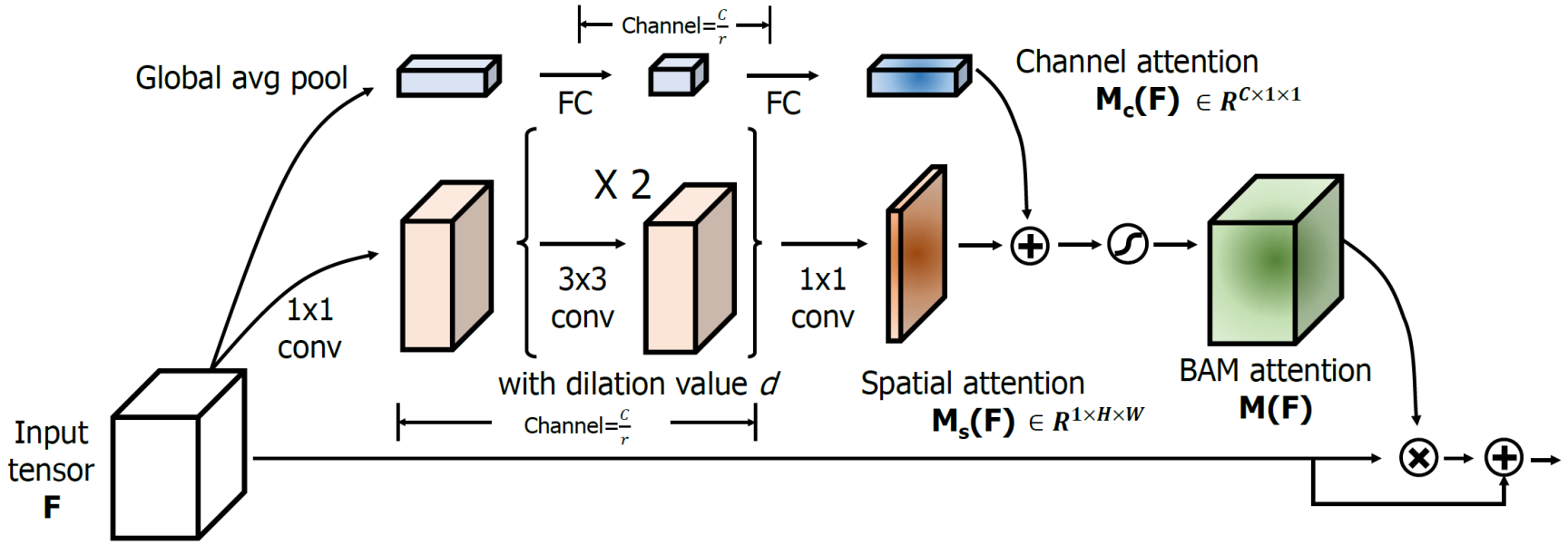
- Tversky Loss (TL):

$$TL(p,\hat{p}) = 1 - \frac{\langle p,\hat{p}\rangle}{\langle p,\hat{p}\rangle + \alpha\langle 1-p,\hat{p}\rangle + \beta\langle p,1-\hat{p}\rangle}$$

- A most used parameter selection: $\beta = 1-\alpha$
- Focused TL: $\mathrm{F}TL(p,\hat{p}) = \big(TL(p,\hat{p})\big)^{\gamma}, \alpha = 0.7, \beta = 0.3, \gamma = 0.75$
- There are many proposal for loss function to address imbalance problem:
  - Generalized Dice Loss (GDL)
  - Boundary Loss (BL)
  - Exponential Logarithmic Loss (EXP Loss)
  - …

BAM: Bottleneck Attention Module

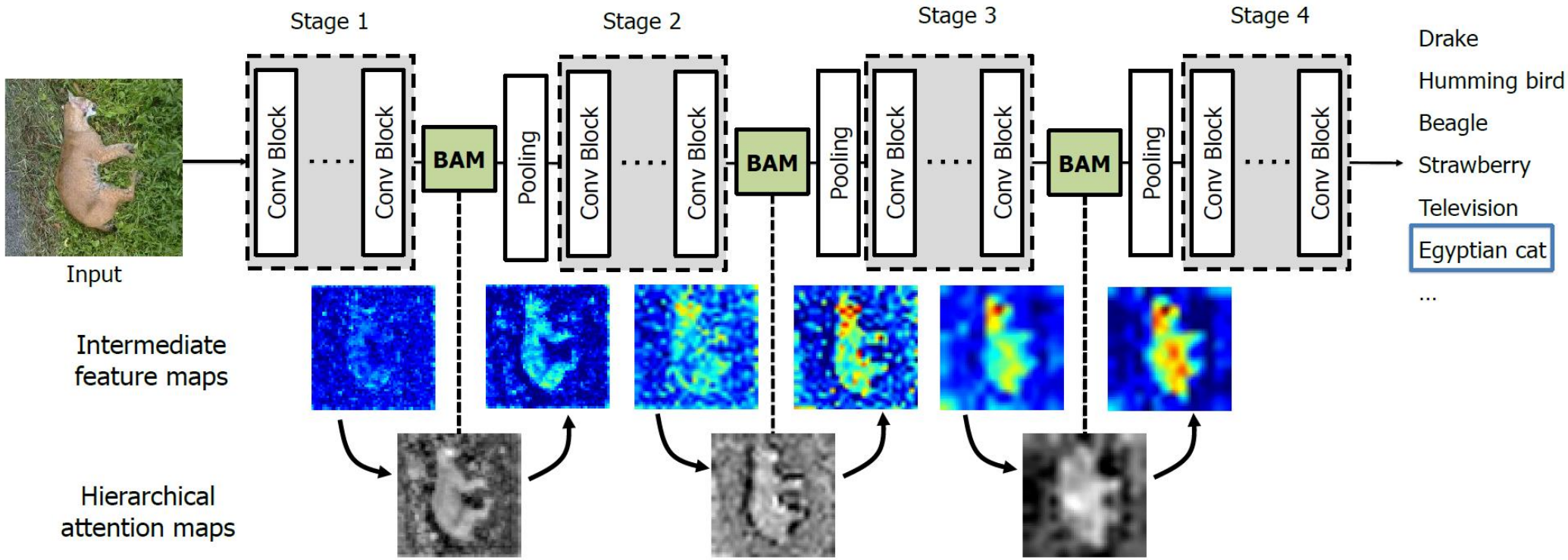
- A Method to boost performance of CNN via feature map enrichment



- How add $M_s(F) \in \mathcal{R}^{1\times H\times W}$ and $M_c(F) \in \mathcal{R}^{C\times1\times1}$: expand $M_s(F)$ to $\mathcal{R}^{C\times H\times W}$ and add elementwise

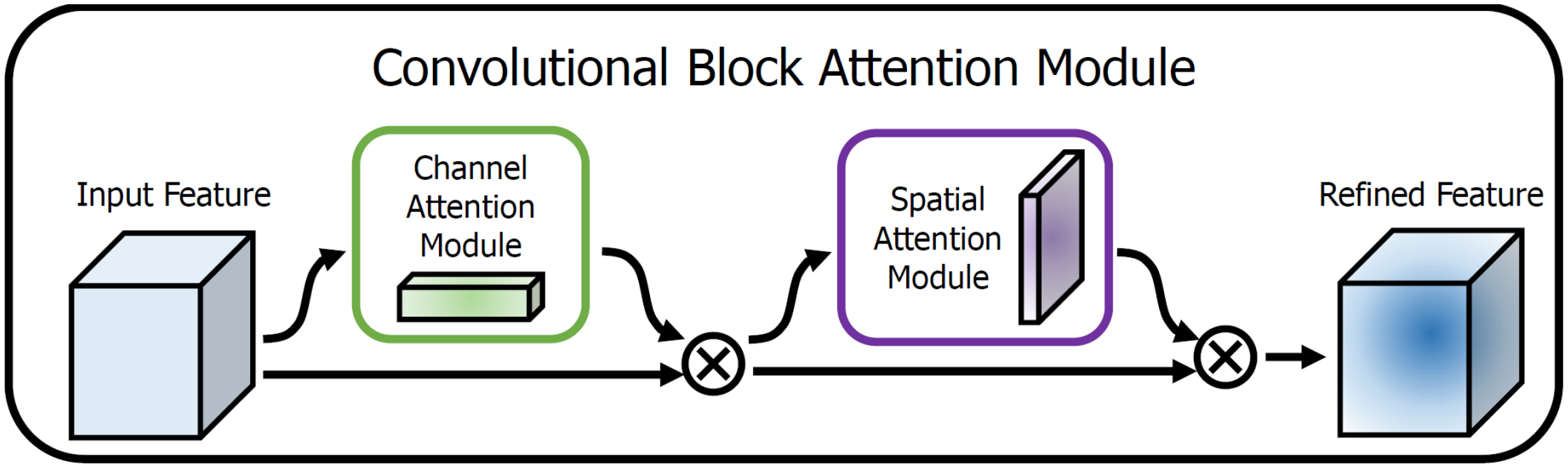
BAM: integrated with a general CNN architecture

- BAM integrated with a general CNN architecture



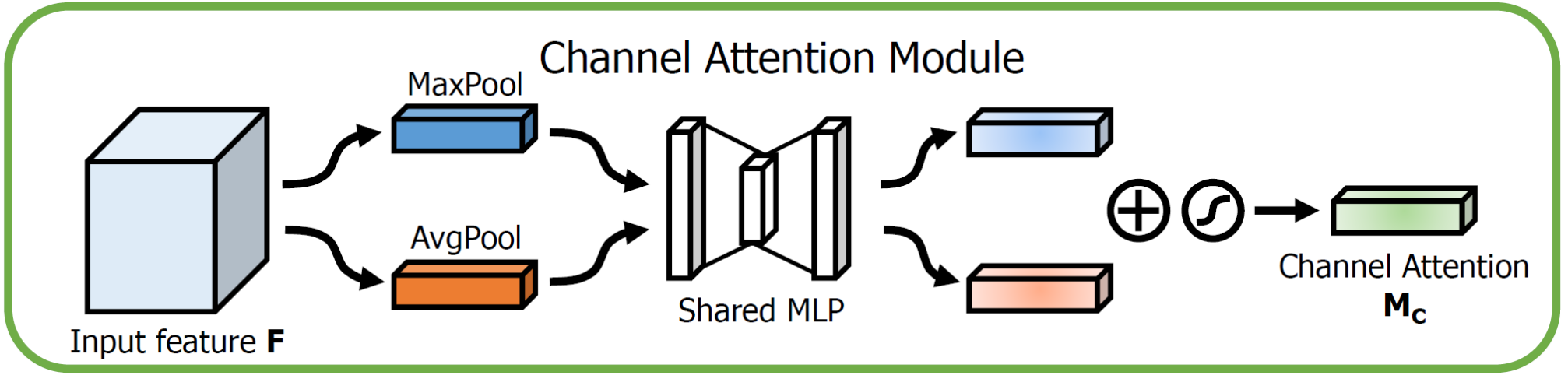
CBAM: Convolutional Block Attention Module

- Given an intermediate feature map, CBAM module sequentially infers attention maps along two separate dimensions, channel and spatial, then the attention maps are multiplied to the input feature map for adaptive feature refinement.



CBAM – Channel Attention Module

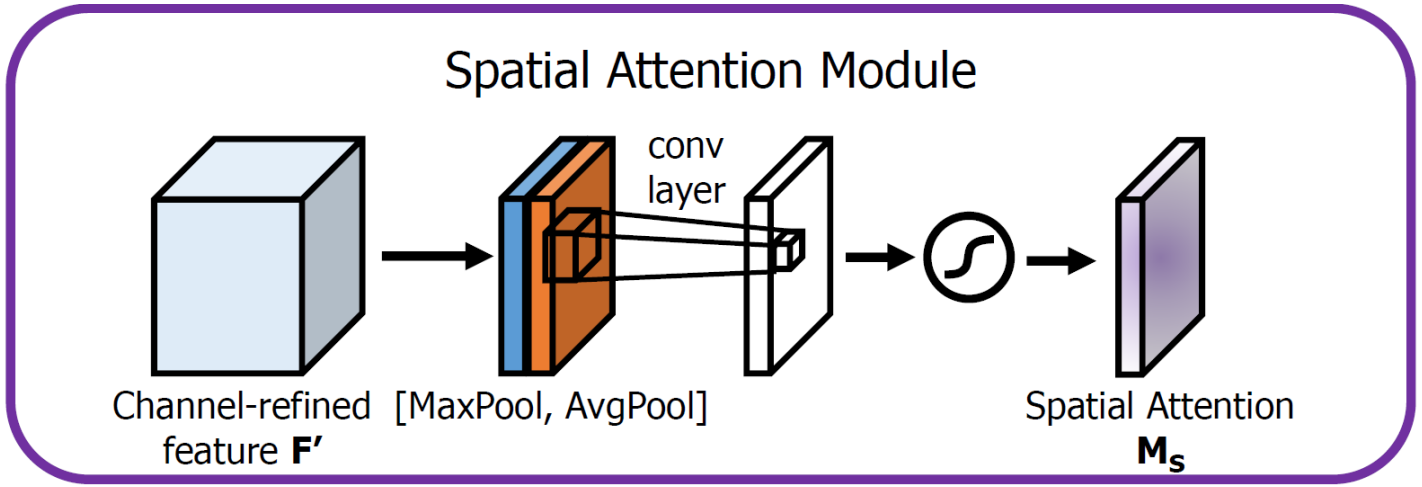
- Channel Attention Module:



- $\boldsymbol{M_c(F)} = \sigma(MLP(AvgPool(\boldsymbol{F})) + MLP(MaxPool(\boldsymbol{F})))$
- $\boldsymbol{M_c(F)} = \sigma(W_1\,(W_0\,(\mathbf{F}^c_{avg})) + W_1\,(W_0\,(\mathbf{F}^c_{max})))$

CBAM – Spatial Attention Module

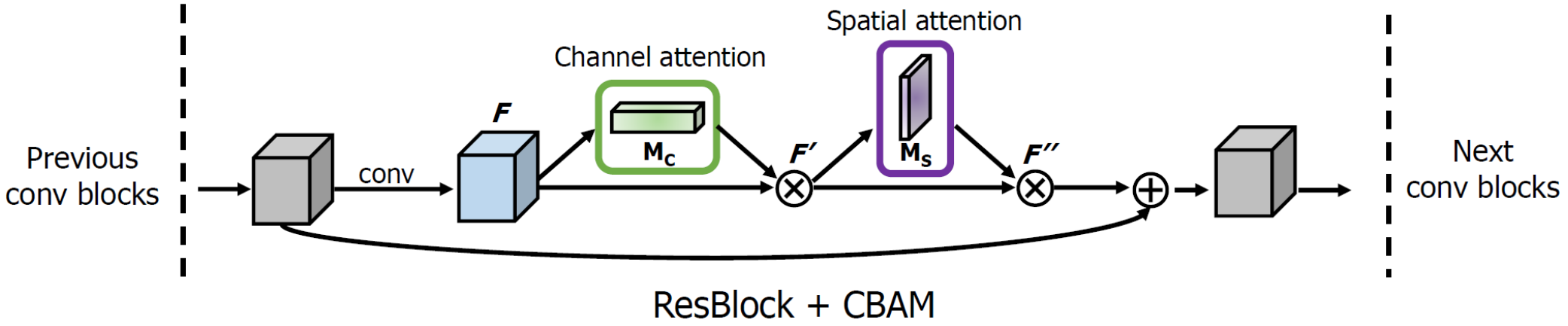
- Spatial Attention Module:



- $\boldsymbol{M_s(F)} = \sigma\left(f^{7\times7}([AvgPool(\boldsymbol{F}); MaxPool(\boldsymbol{F})])\right)$
- $\boldsymbol{M_s(F)} = \sigma\left(f^{7\times7}\left(\left[\mathbf{F}^c_{avg}; \mathbf{F}^c_{max}\right]\right)\right)$

CBAM – General arrangement

- CBAM integrated with a ResBlock in ResNet



Computer Vision and CNN

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