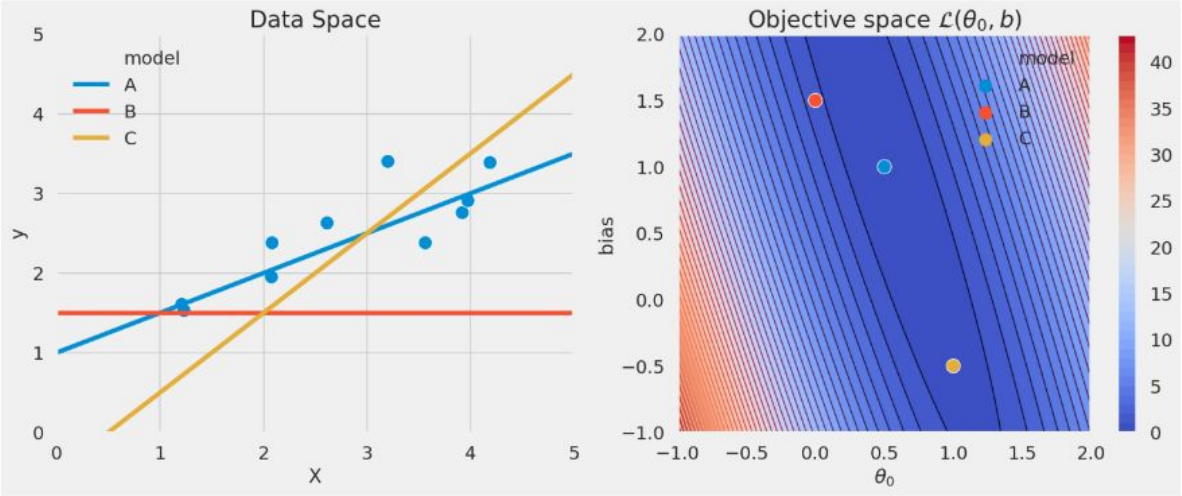
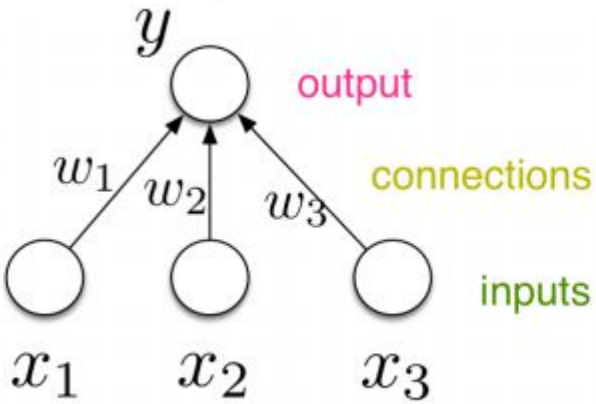
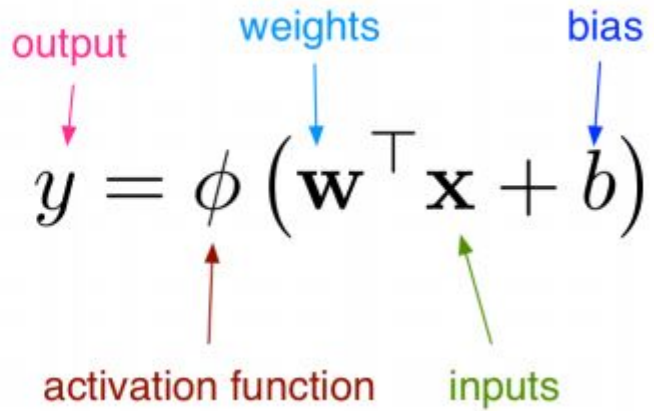
Deep Learning

6. Neural Networks Applications in Computer Vision

Recap

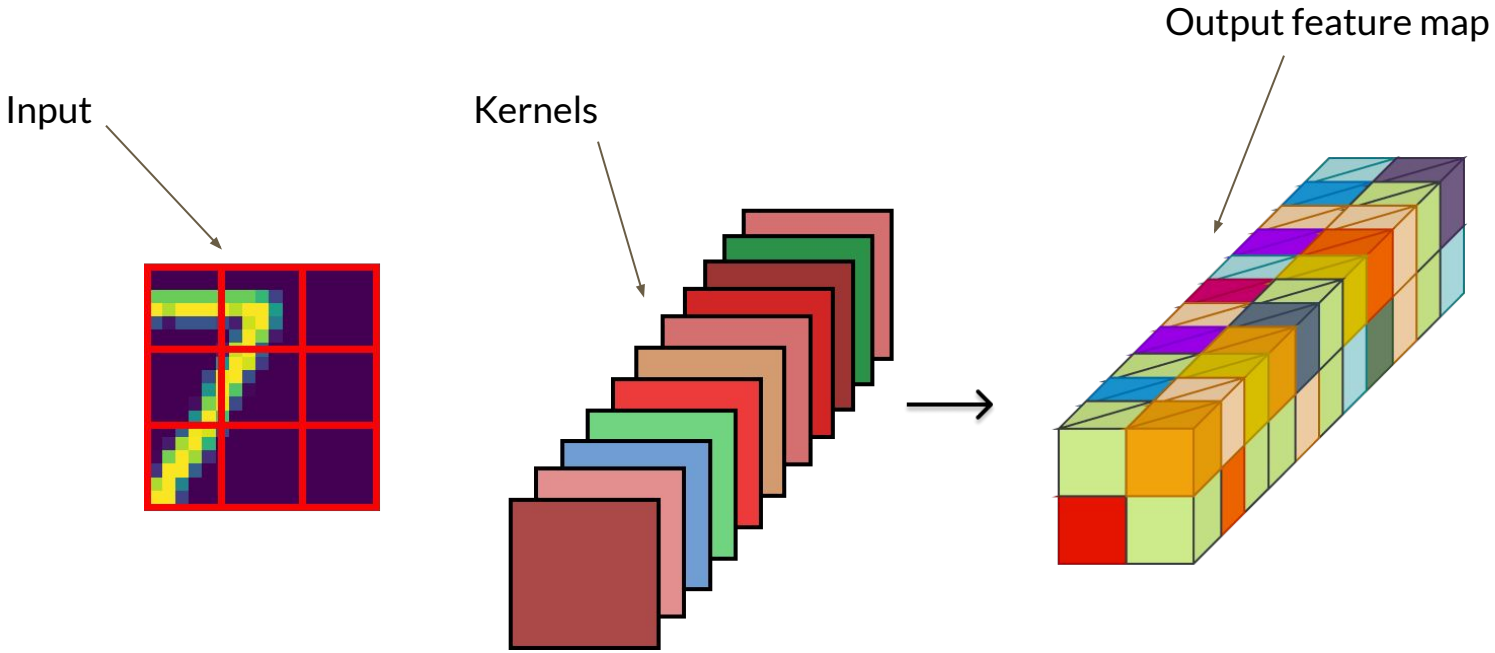

Data Space
model
A
B
C
X
y
Objective space $\mathcal{L}(\theta_0, b)$
model
A
B
C
bias
θ_0


y
output
w_1
w_2
w_3
connections
inputs
x_1
x_2
x_3

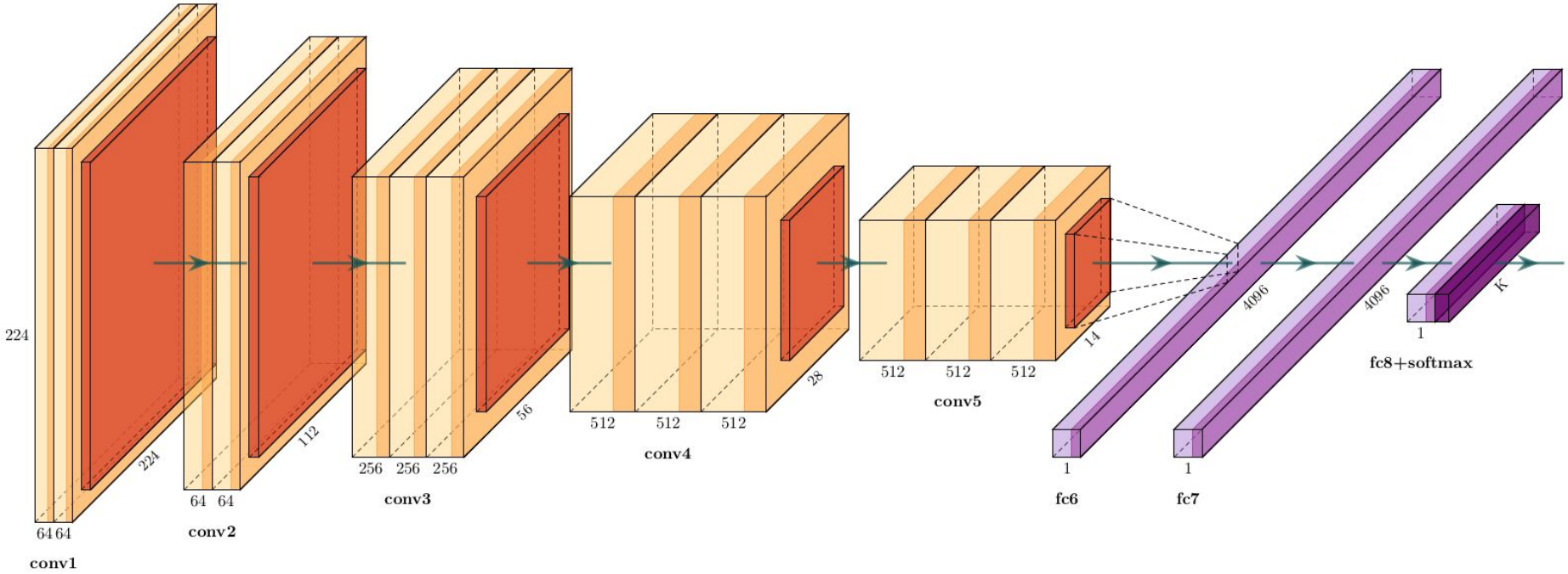

output
weights
bias
$y = \phi\left(\mathbf{w}^\top \mathbf{x} + b\right)$
activation function
inputs

Recap: Convolutional Networks

A convolutional operation:



Recap: Convolutional Networks



Simonyan and Zisserman [2014]: Very deep convolutional networks for large-scale image recognition.

Image from:
https://github.com/HarisIqbal88/PlotNeuralNet

Recap: Improve learning on a new task

- Transfer learning
  - Reuse parts of a previously learned model

- Self-supervised learning
  - Learn good representation without using annotations
  - Rely on self-supervision: tasks defined purely from data
  - You can transfer the self-supervised representation to a desired task

Stochastic Gradient Descent

Consider instead objective functions that are the sum of the losses -- $\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \mathcal{L}_n(\boldsymbol{\theta})$, resulting in the update:

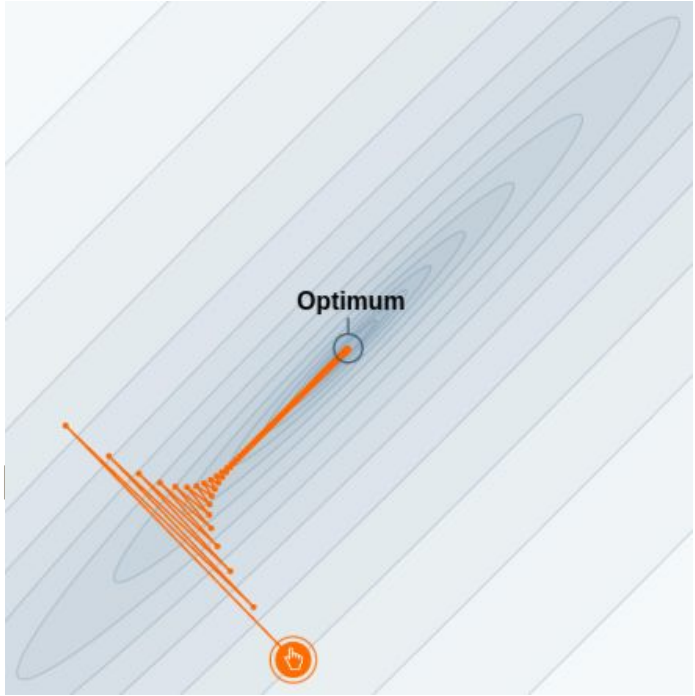
$$\boldsymbol{\theta}_{j+1} \leftarrow \boldsymbol{\theta}_j - \eta \sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}_j} \mathcal{L}_n(\boldsymbol{\theta})$$

Recap: Optimisation

- There are cases when different directions are steeper (easier) than others.

- This could lead to instabilities.
- Optimisations procedures could help: RMSpro or Adam.

- *General idea*: neural networks will optimise the easier path. We may not want this anytime and should find solutions to avoid it.



Shortcuts in Learning

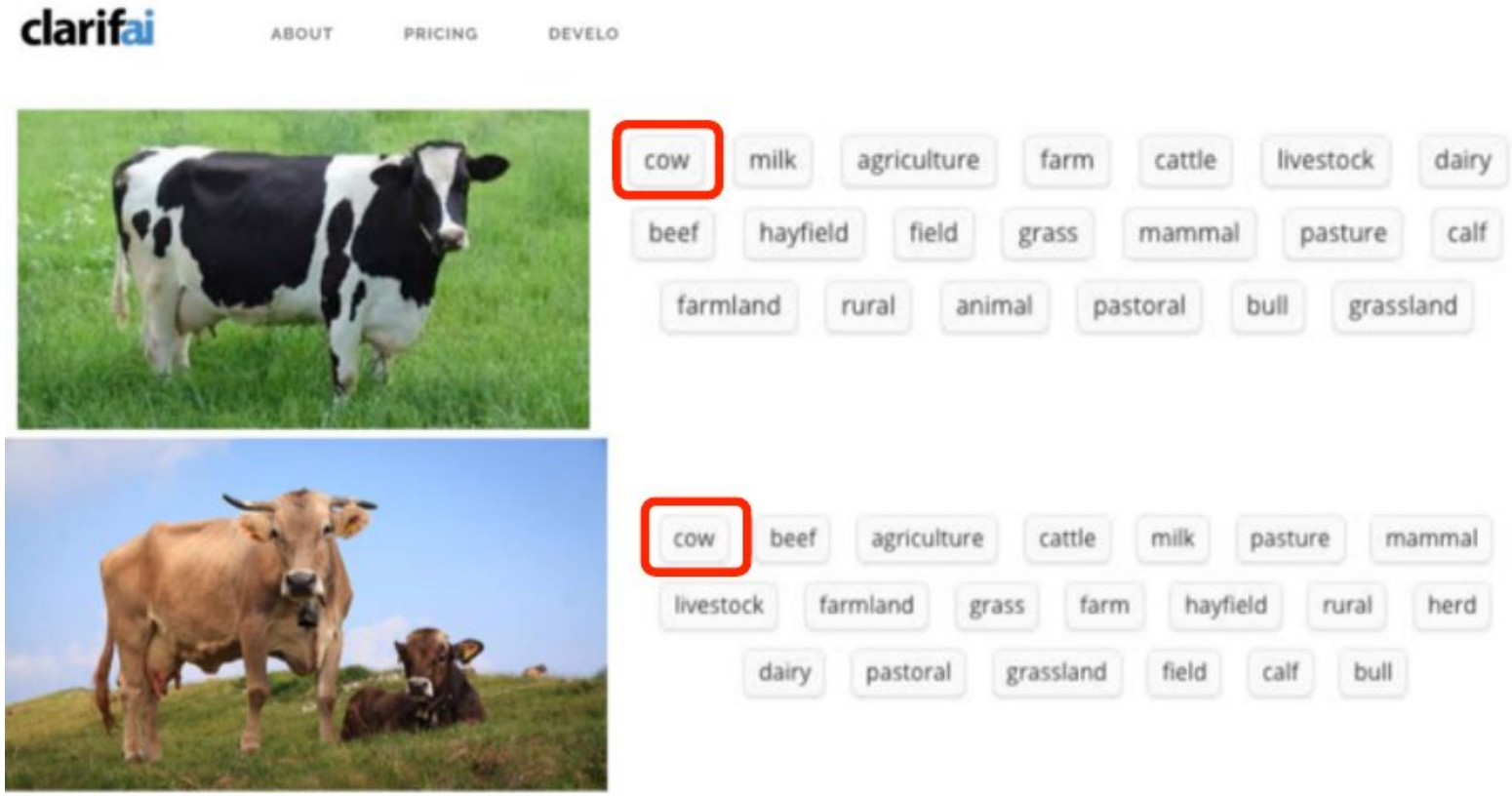


Image from Aaron Courville talk (EEML 2021) - who got it from Bernhard Schölkopf 's slides (2017) - who got it from Pietro Perona (2017) :)

Shortcuts in Learning

Shortcuts: learning strategies / rules that solve the task in an **unintended** way that is not based on true causes and **cannot generalize**

- These schotcuts are based on spurious (fake / misleading) features
  - Also called non-causal features
  - e.g. use color of the background to distinguish between cows and camels

(A) **Cow: 0.99**, Pasture: 0.99, Grass: 0.99, No Person: 0.98, Mammal: 0.98

(B) No Person: 0.99, Water: 0.98, Beach: 0.97, Outdoors: 0.97, Seashore: 0.97

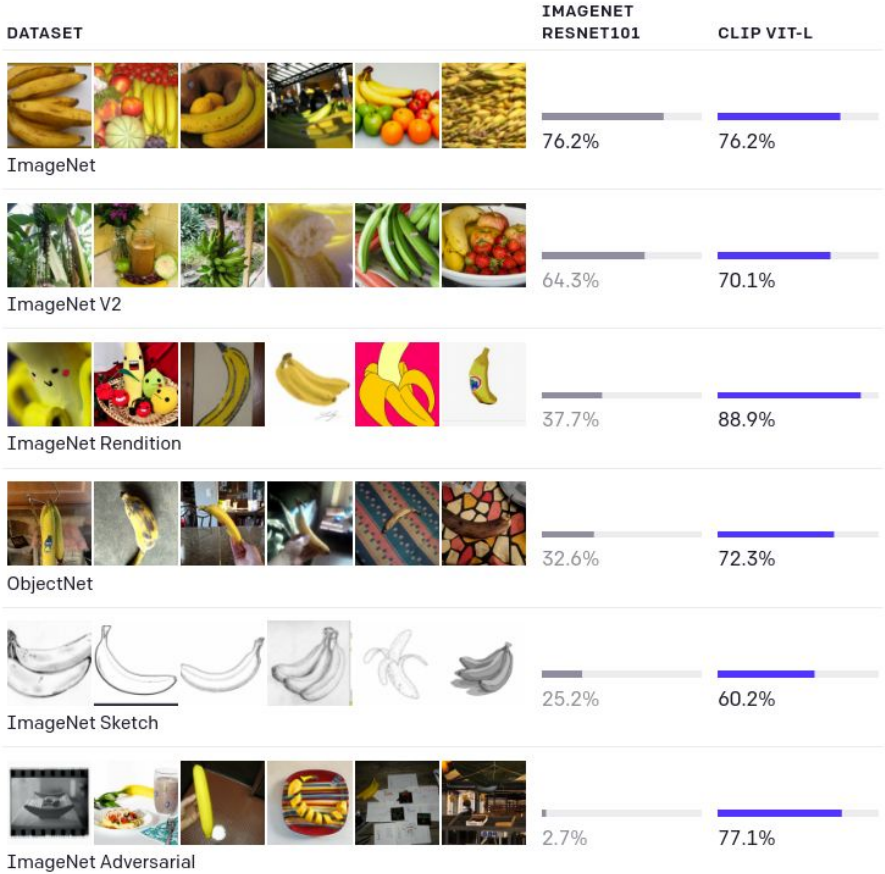
(C) No Person: 0.97, **Mammal: 0.96**, Water: 0.94, Beach: 0.94, Two: 0.94

Beery et al. "Recognition in terra incognita." *ECCV*. 2018.
Geirhos et al. "Shortcut learning in deep neural networks." *Nature Machine Intelligence* (2020).

Shortcuts in Learning

Large scale models like OpenAI's CLIP trained on 400M image-text pairs are:

- more robust to shortcuts
- capable of better generalising



https://openai.com/blog/clip/

Shortcuts in Learning

Attack text label iPod

Granny Smith	85.6%
iPod	0.4%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.1%

Granny Smith	0.1%
iPod	99.7%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.0%

But even CLIP models are still vulnerable to shortcuts

https://openai.com/blog/multimodal-neurons/

Systematic generalisation

Systematic Generalization: Generalization to examples that may not be drawn from the same distribution as the training data, but that obey the same **basic rules** of production or underlying structure.

Systematic generalisation - special king of **out-of-distribution generalisation**

Promoting systematic generalisation :

- Increase data diversity and potential use data augmentations
- Learn features that are do not change across different environments
  - They are more robust - and ideally represent the true causes
- Modularity and other Inductive biases

Case study: face analysis

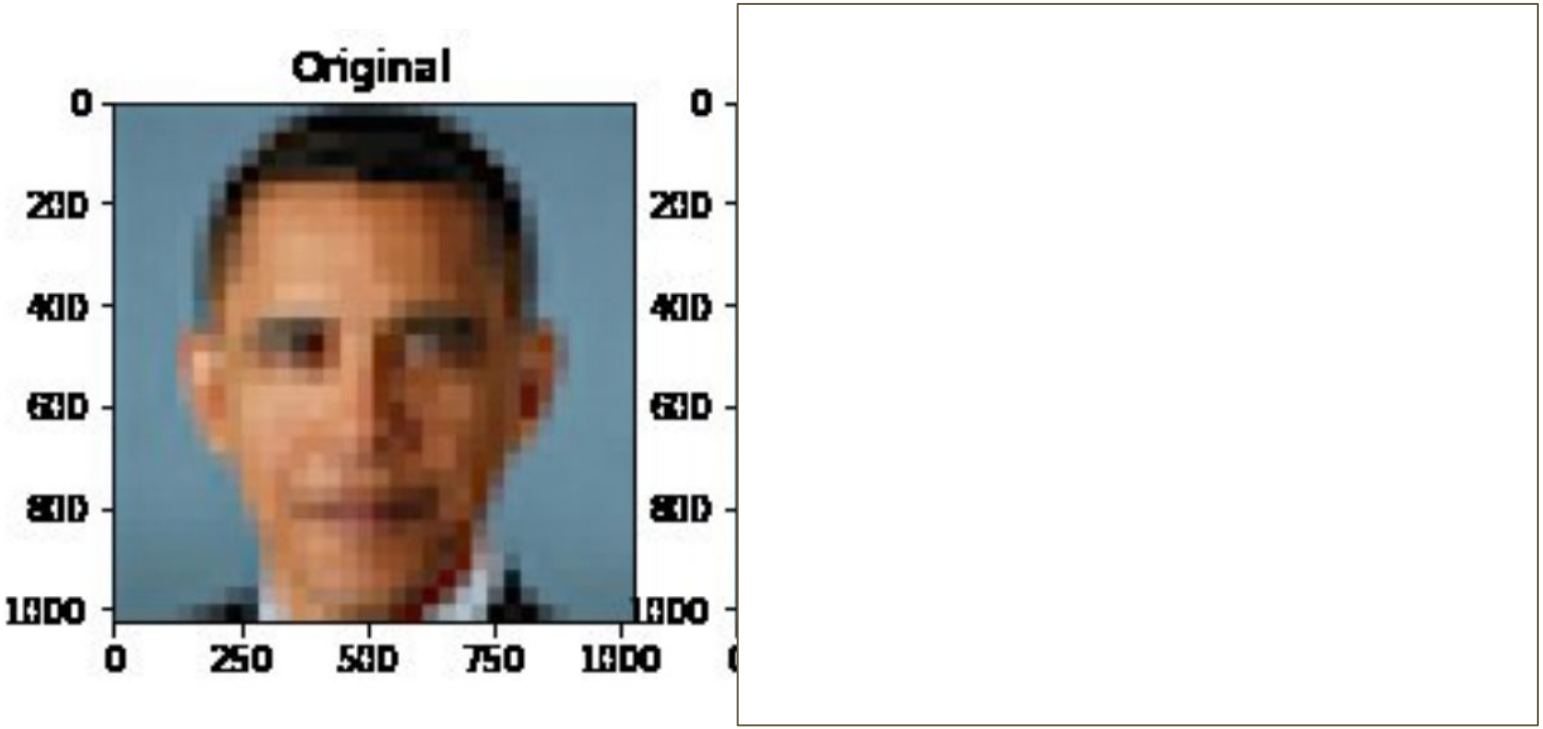
Buolamwini and Gebru [2018] study the performance of standard gender classifiers offered by API bundles by Microsoft, IBM and Face++

They show large differences in the performance on different subgroups

- All classifiers perform better on male faces than female faces
  - 8.1% − 20.6% difference in error rate
- All classifiers perform better on lighter faces than darker faces
  - 11.8% − 19.2% difference in error rate
- All classifiers perform worst on darker female faces
  - 20.8% − 34.7% error rate

Buolamwini and Gebru. "Gender shades: Intersectional accuracy disparities in commercial gender classification." FAccT 2018.

Case study: Super-Resolution

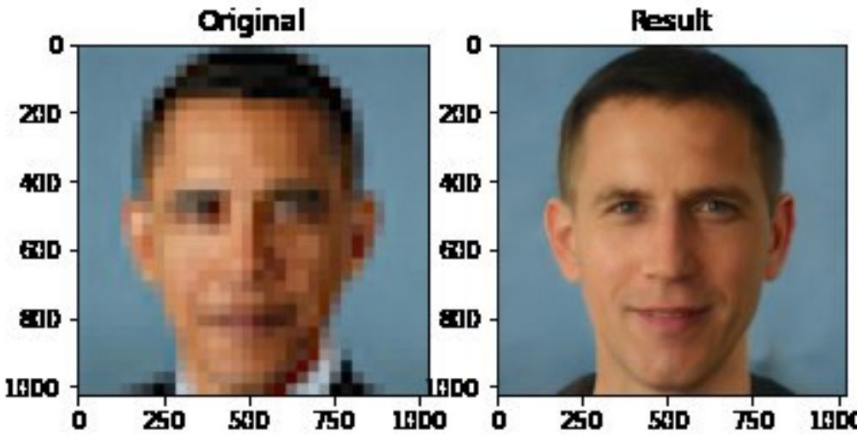


Case study: Super-Resolution

- Current ML methods could upsample low resolution images
- Keep in mind that some information cannot be recuperated from low-resolution
- ML methods inherently ‘guess’ the missing details

In certain cases this could be useful: photo sharing - video games

But this can also be used for dangerous applications - wrongly identifying suspects



Case study: Super-Resolution

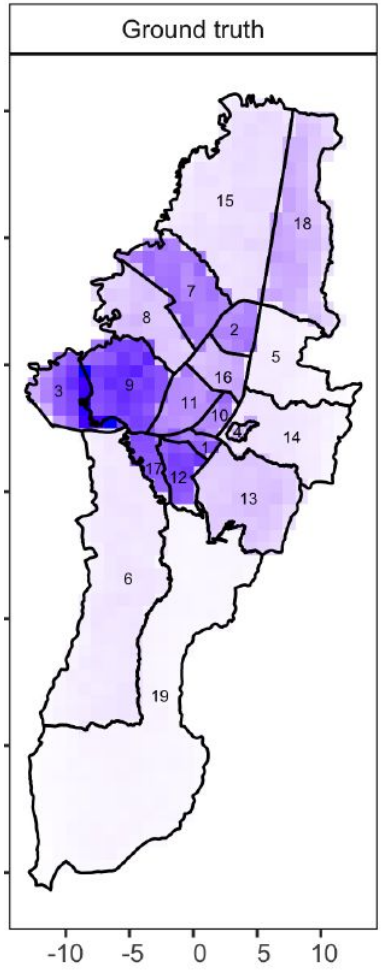
Google's New AI Engines Can Zoom and Enhance, Just Like in the Movies

SUZANNE HUMPHRIES @yeah_books
SEP 12, 2021, 5:00 PM EST | 2 min read

- Always present clearly what a method can do and cannot do
- Take care of unintended use cases

Case study: predictive policing

- Does this seem like a problem approachable with ML?
- Could spatio-temporal models for weather prediction or car-sharing prediction be used?
- When applied in real work these methods create feedback loops that increase the biases.
- Should we use them in the first place?



Case study: hiring and job searching

- Workflow of LinkedIn Talent Search (tool used by recruitos)
  - Retrieve large set of candidates based on rules
  - **Rank** candidates based on scores given by ML models
- Some care for re-ranking based on given protected attributes
  - Gender and age of the top results should reflect a desired distribution
- The method obtains: “tremendous improvement in the fairness metrics”

- But how about attributes that are not considered?
  - It is hard to define all possible protected attributes (ethnic group, place of birth, address, etc.)
  - Some biases are hidden and cannot be easily defined
- How can we be sure that the ranking is not entirely based on shortcuts?

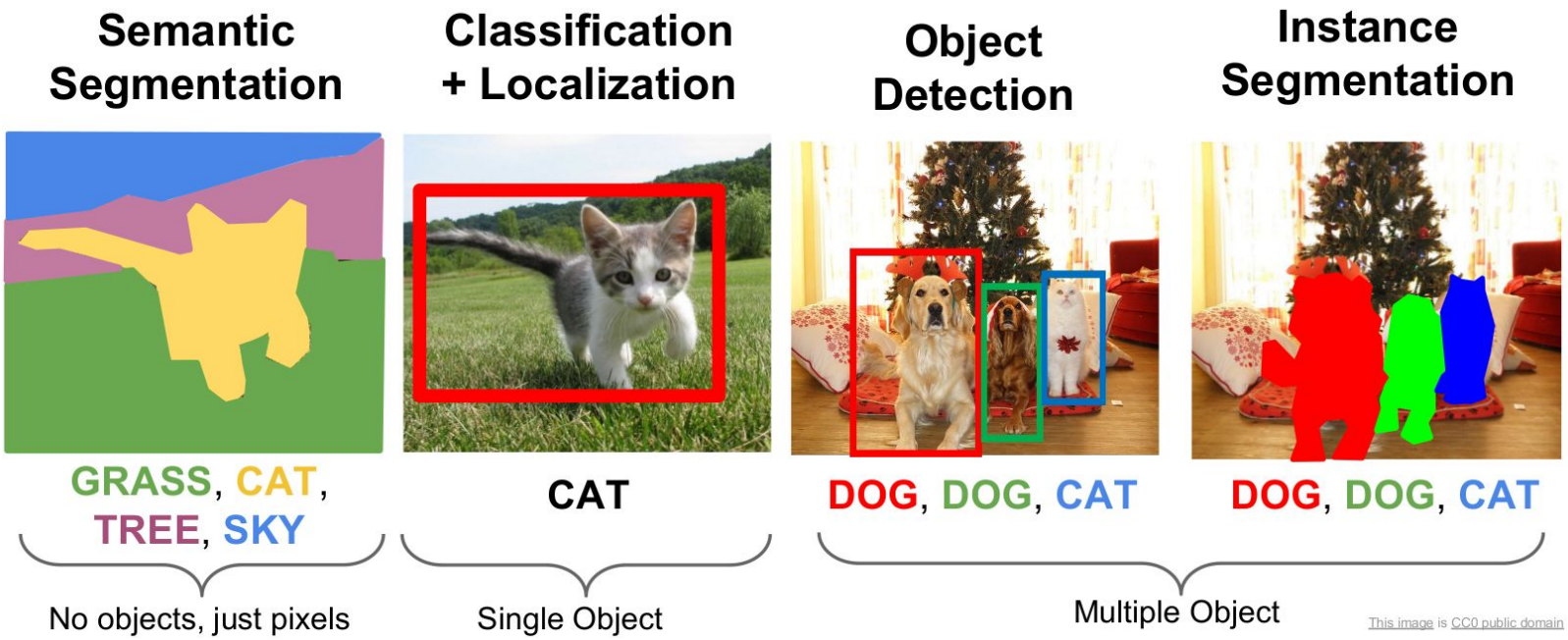
Geyik et al. "Fairness-aware ranking in search & recommendation systems with application to linkedin talent search." *ACM KDD*. 2019.

Recap

- Deep Learning suffer from learning shortcuts
- It is hard to eliminate spurious features
- Data has biases

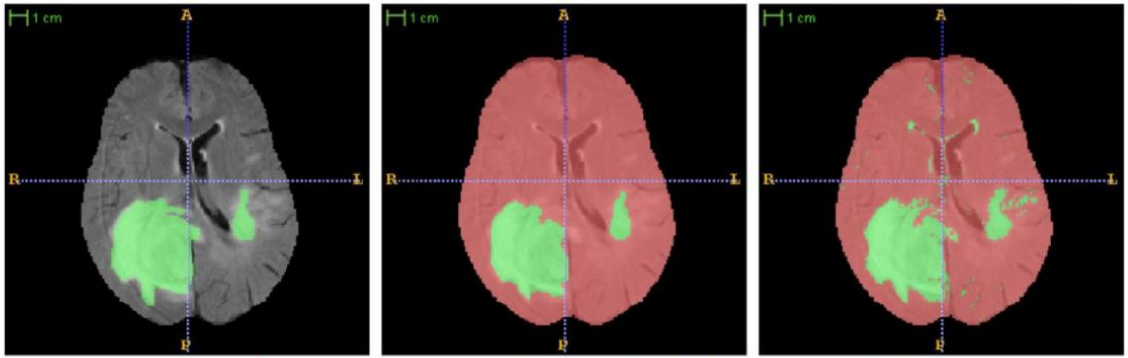
- DL systems could produce biassed and unintended results

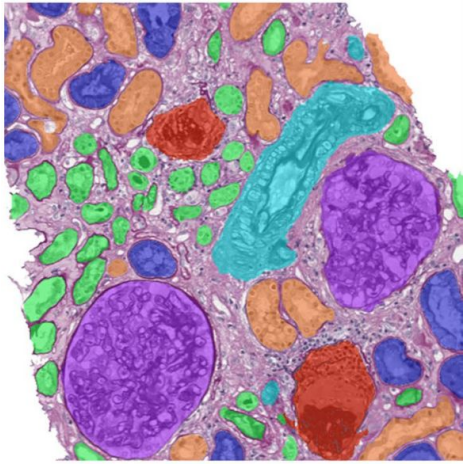
Computer Vision Tasks

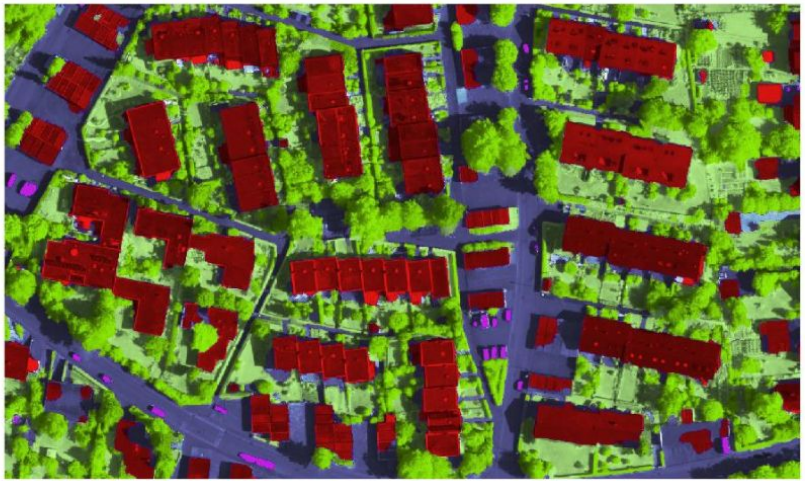


Slide from: CS231n Stanford Course

Segmentation





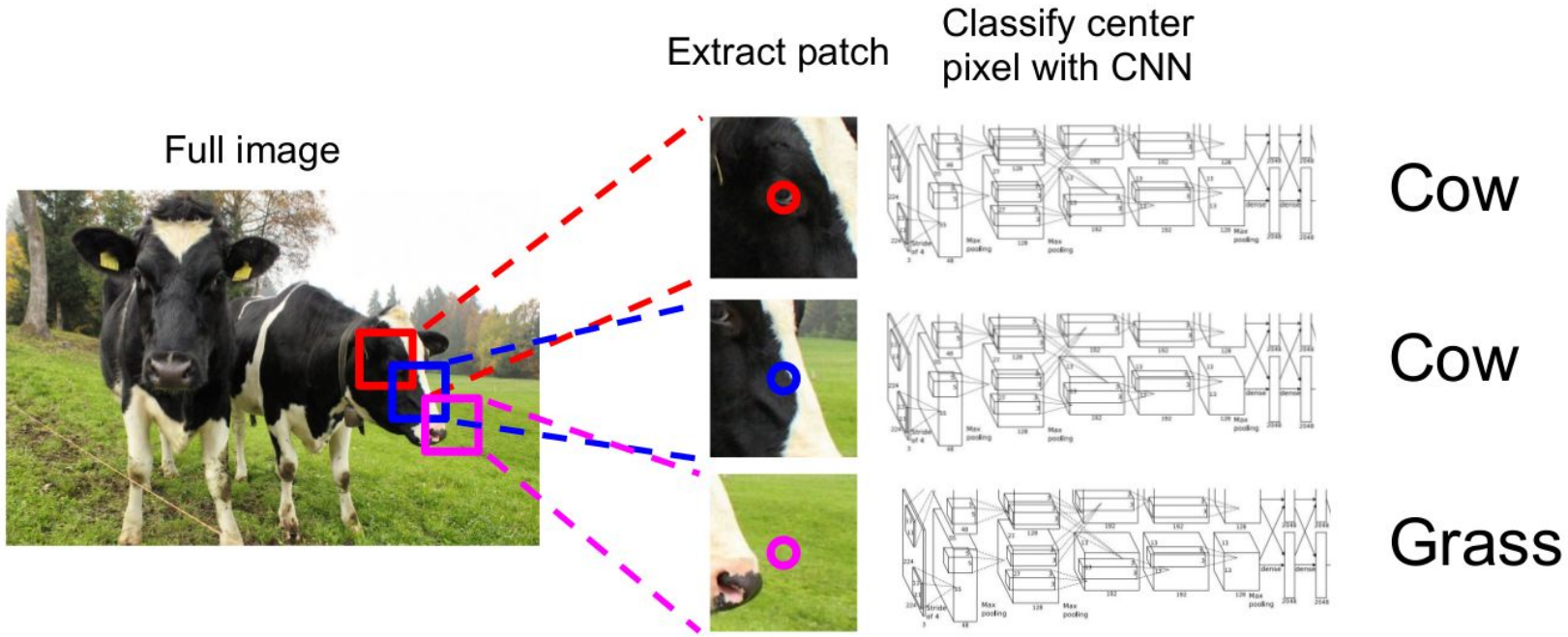


Definition:

- partition an image into regions each of which has a reasonably homogeneous visual appearance or which corresponds to objects or parts of an object [Forsyth and Ponce 2003]

Image from: Bauer et al. [2011], Marmanis et al. [2016], van der Laak [2021].

Segmentation: Sliding Window



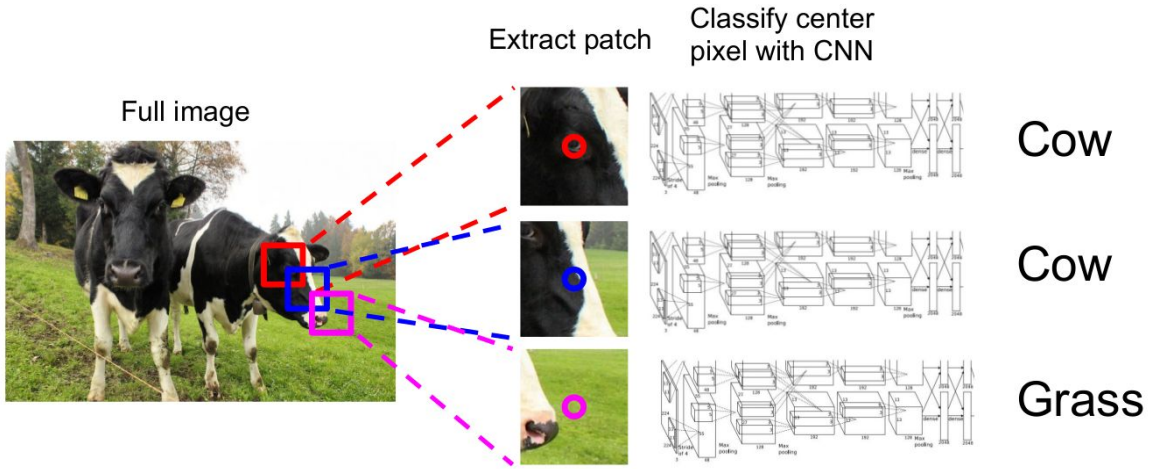
Slide from: CS231n Stanford Course

Segmentation: Sliding Window

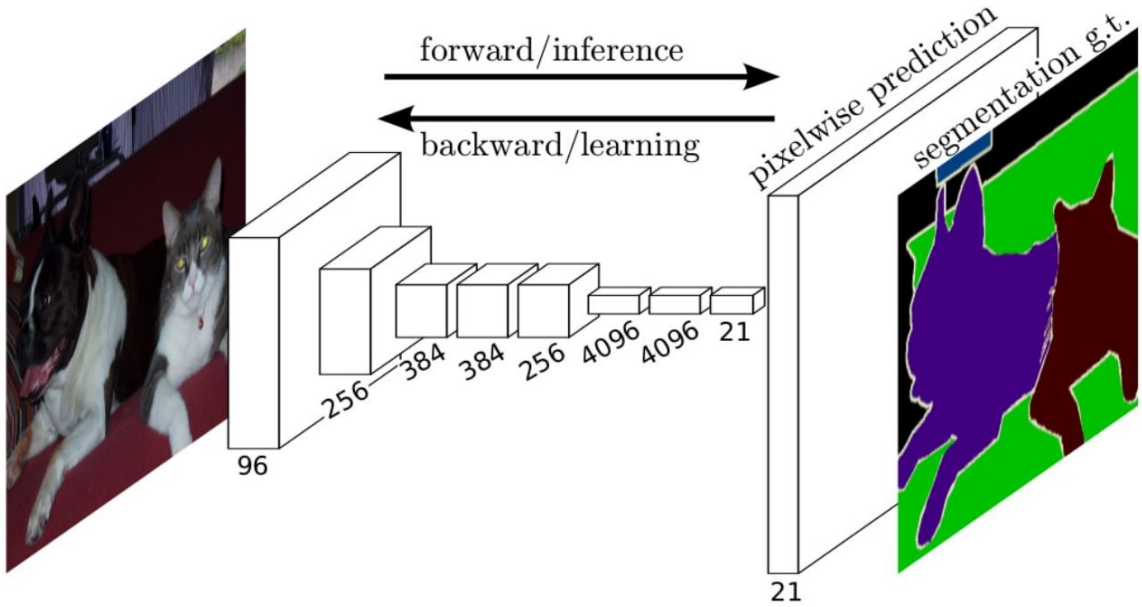
- Select patches obtained by sliding a window through the image
- Learn to classify each patch as the class of the object or region found at its center
- Create (patches, center labels) pairs
    - Learn the convolutional network to predict the class of the center of a patch

- Classify **each pixel** by selecting a patch around it and run the learnt network

- Problem: Inefficient



Segmentation



Goal:

- Generate a segmentation for the entire image **without redundant** computations
- Use information learned on classification tasks with more data, and be able to transfer it to segmentation task

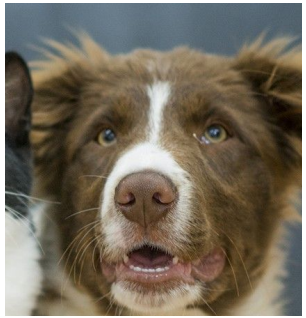
Image from: Long et al.[2014]: Convolutional networks for semantic segmentation.

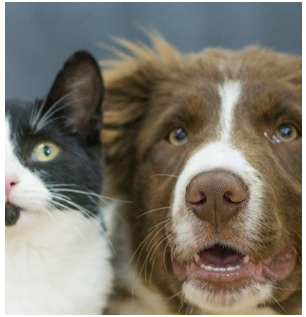
Segmentation

- convolution, pooling and activations layers are local operations
  - apply them on the whole image

- a fully-connected layer could be seen as convolutional layer
  - Instead of fc layers, use conv layers with filters of the same size as the input to the gc

Segmentation - sliding window

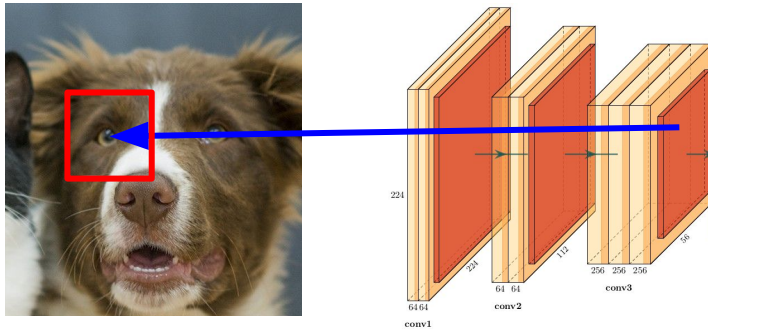


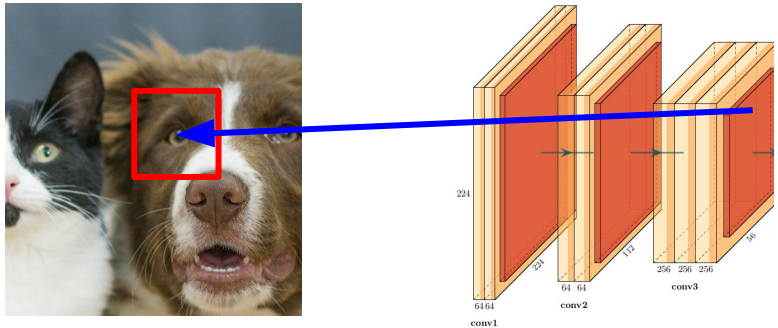


- Let's take two patches from a larger image and pass each of them through the **same** convolutional neural network

- We will observe that there exist points in the intermediate features maps of the net that:
  - Are obtained by the exact computations of the exact same input
  - Are redundant

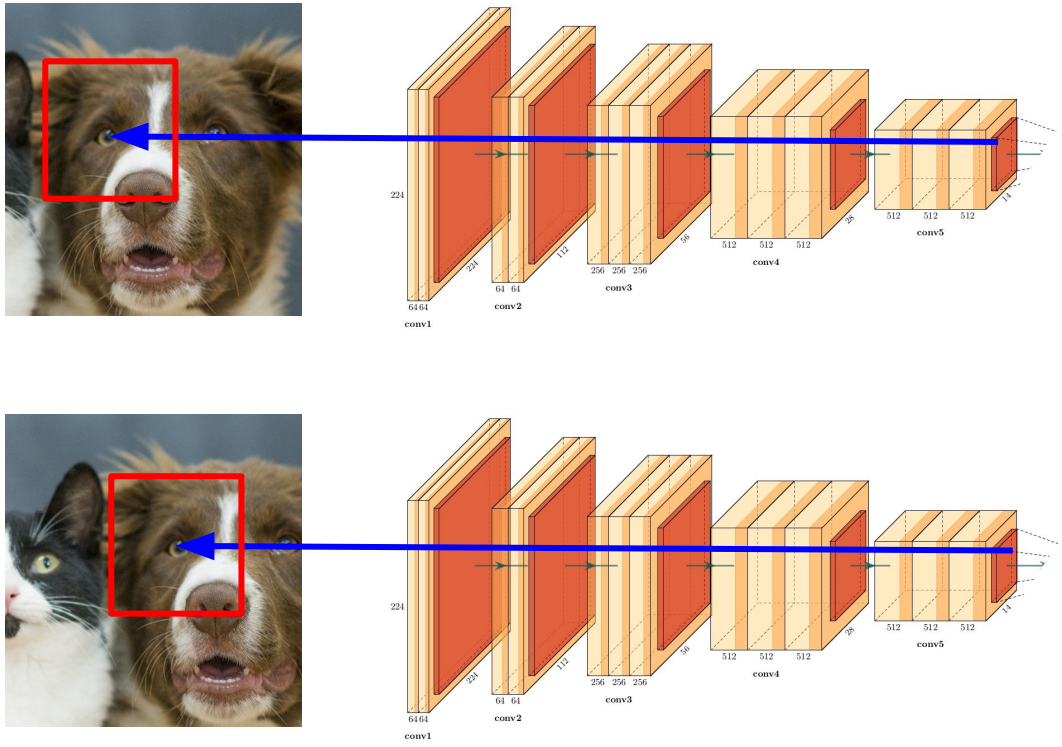
Segmentation - sliding window





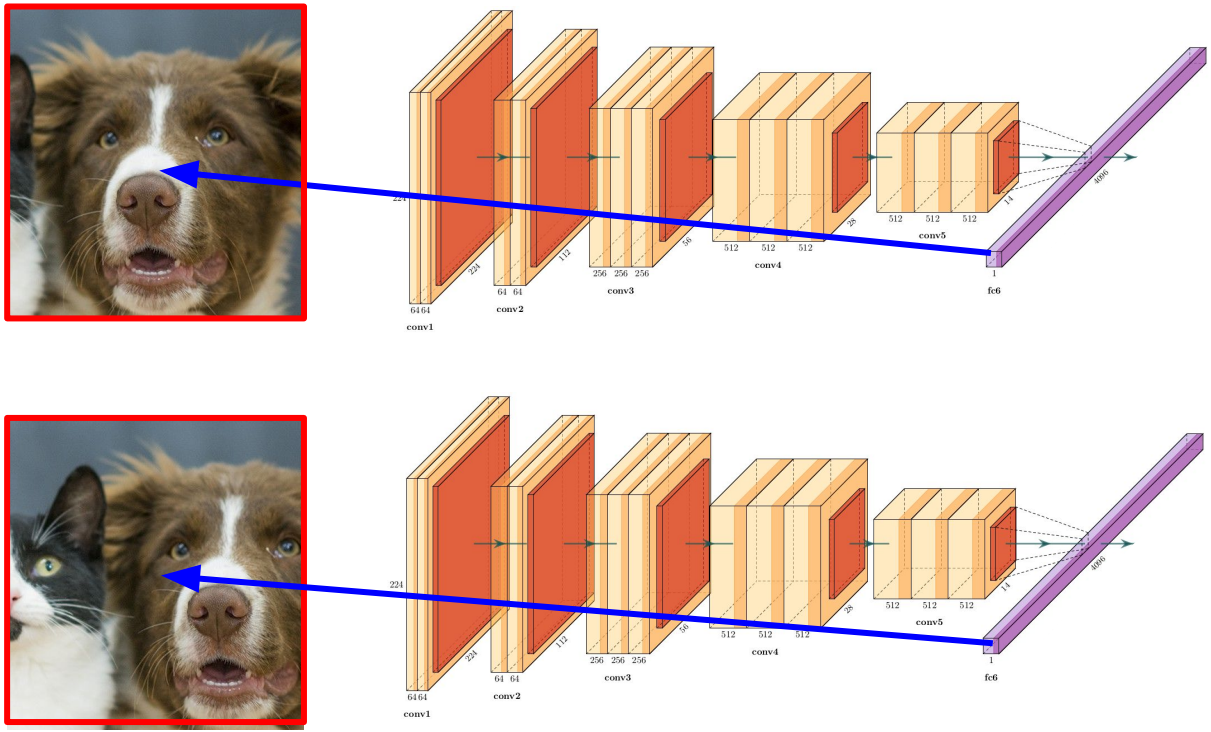
- Convolutional and pooling layers share computations
- There exists points in the feature maps of the two image patches that have the **same receptive field** in the input image
  - These points have the same values obtained by the same computations
- Results: for **neighbouring** patches most convolutional and pooling computations are **shared**

Segmentation - sliding window



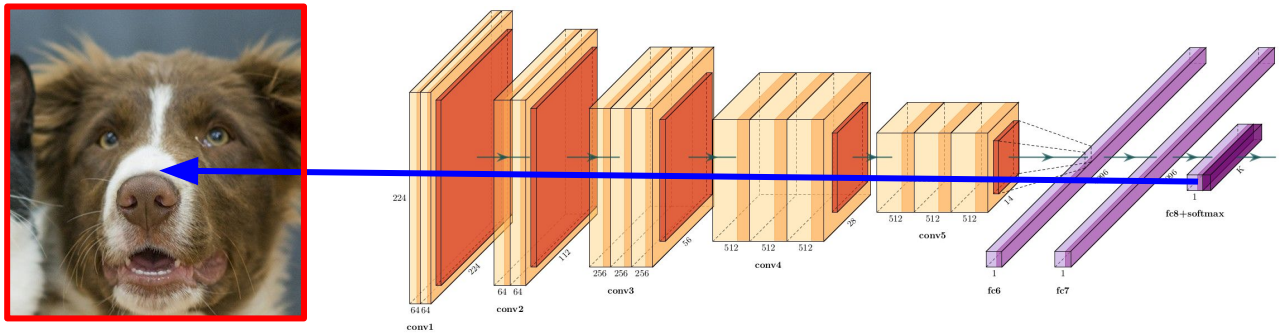
- Convolutional and pooling layers share computations
- There exists points in the feature maps of the two image patches that have the **same receptive field** in the input image
  - These points have the same values obtained by the same computations
- Results: for **neighbouring** patches most convolutional and pooling computations are **shared**

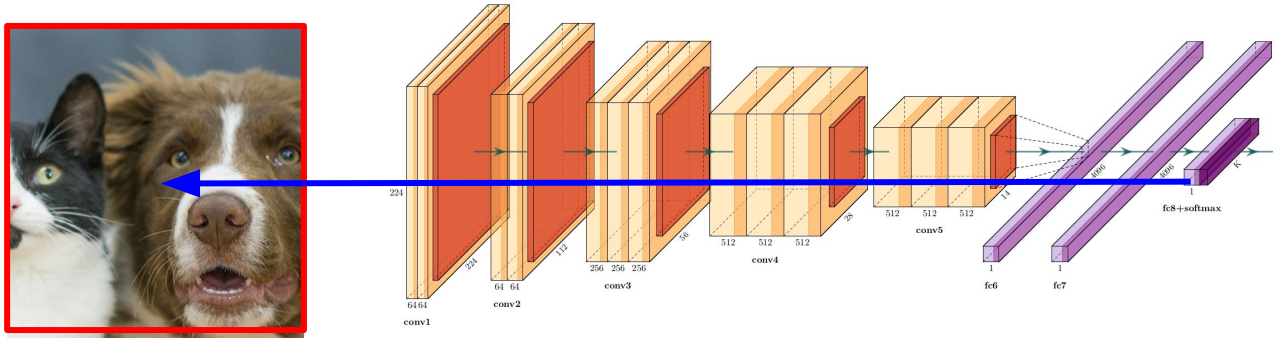
Segmentation - sliding window



- Fully connected layers depend on the whole input
- Output of the fully-connected layers for the two images are completely different
- Result: the two patches do **NOT share** any computations in the fully connected layers

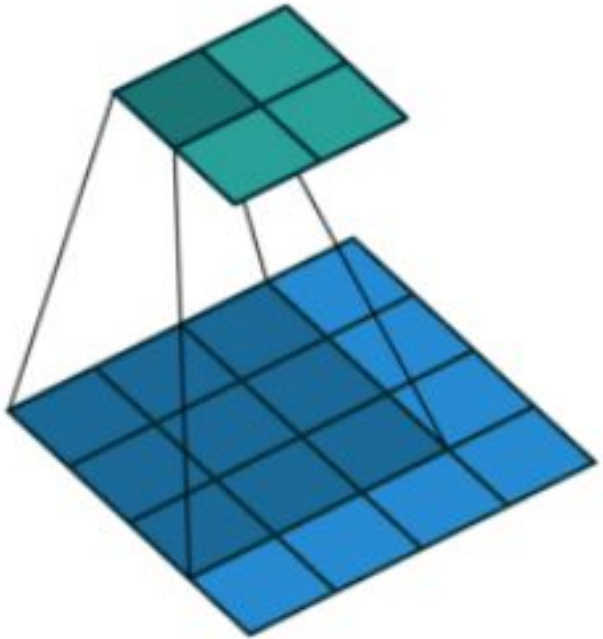
Segmentation





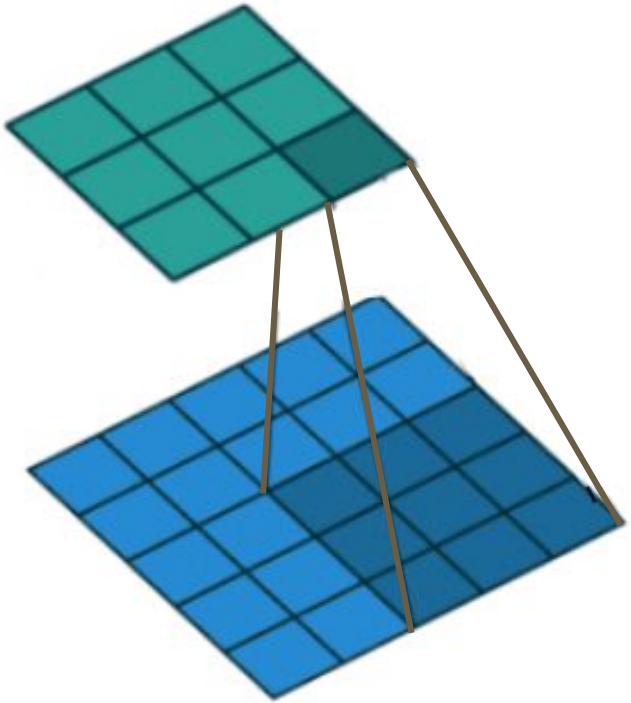
- Fully connected layers depend on the whole input
- Output of the fully-connected layers for the two images are completely **different**
- Result: the two patches do **NOT share** any computations in the fully connected layers

Apply convolutional layer on larger input



- convolution, pooling and activations layers are local operations
  - apply them on the whole image

Apply convolutional layer on larger input



- convolution, pooling and activations layers are local operations
  - apply them on the whole image

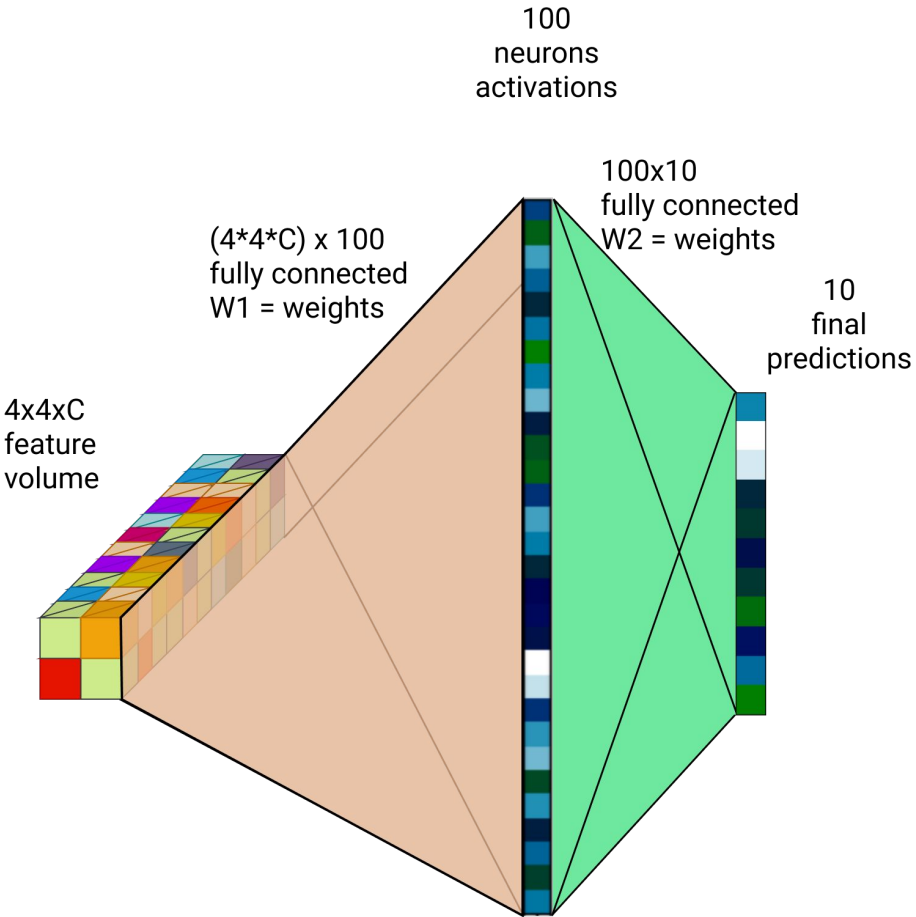
- enlarging the input (blue) does NOT influence the top 4x4 cells of the output

Fully-connected -> convolutional

- Instead of applying a fully connected layer on the output corresponding to each small patch, we will apply a conv layer on the entire image that would produce the same output

- Let’s consider first only the feature maps corresponding to a small patch, that would be fed to a fully connected layer

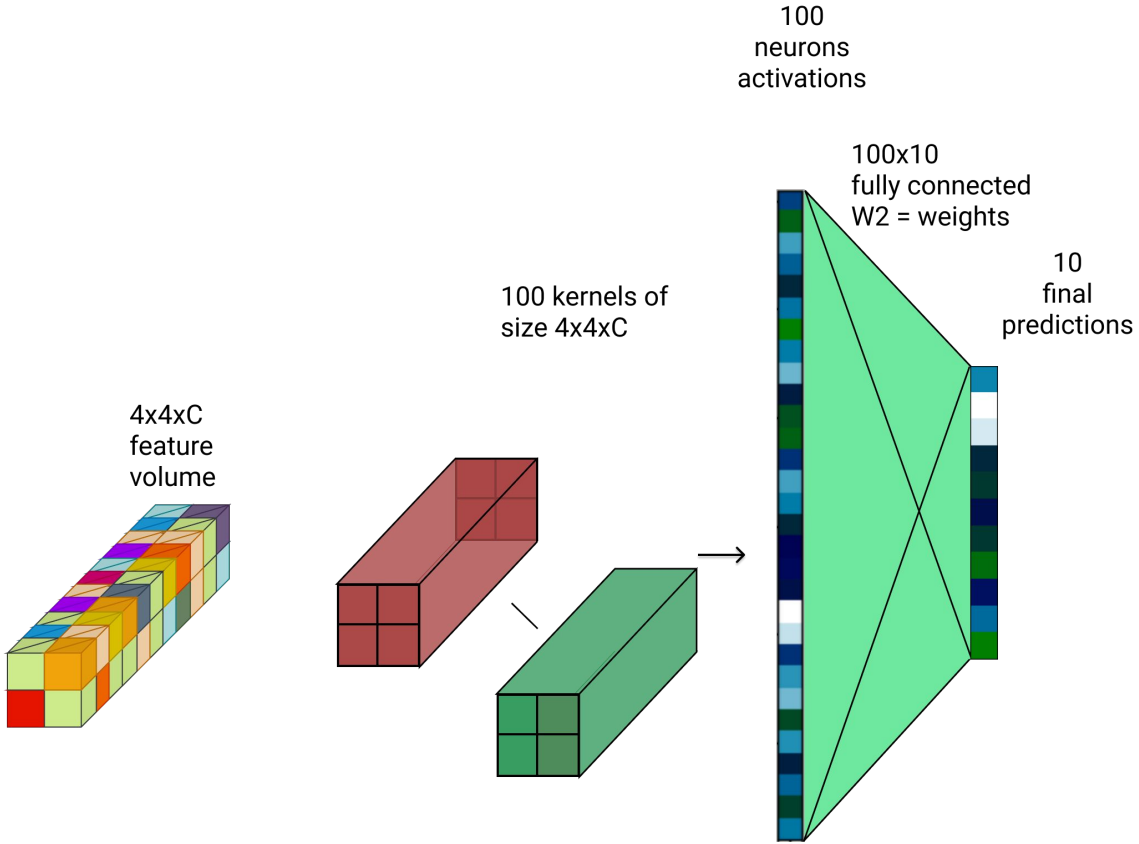
Fully-connected -> convolutional



- Input: feature volume of size 4x4xC
- Process the features into the final classification by two fully connected layers

- fully connected layers can be converted into convolutional layers
- Fully connected layer is **equivalent** with convolutional layer with kernels of the same size as the input

Fully-connected -> convolutional

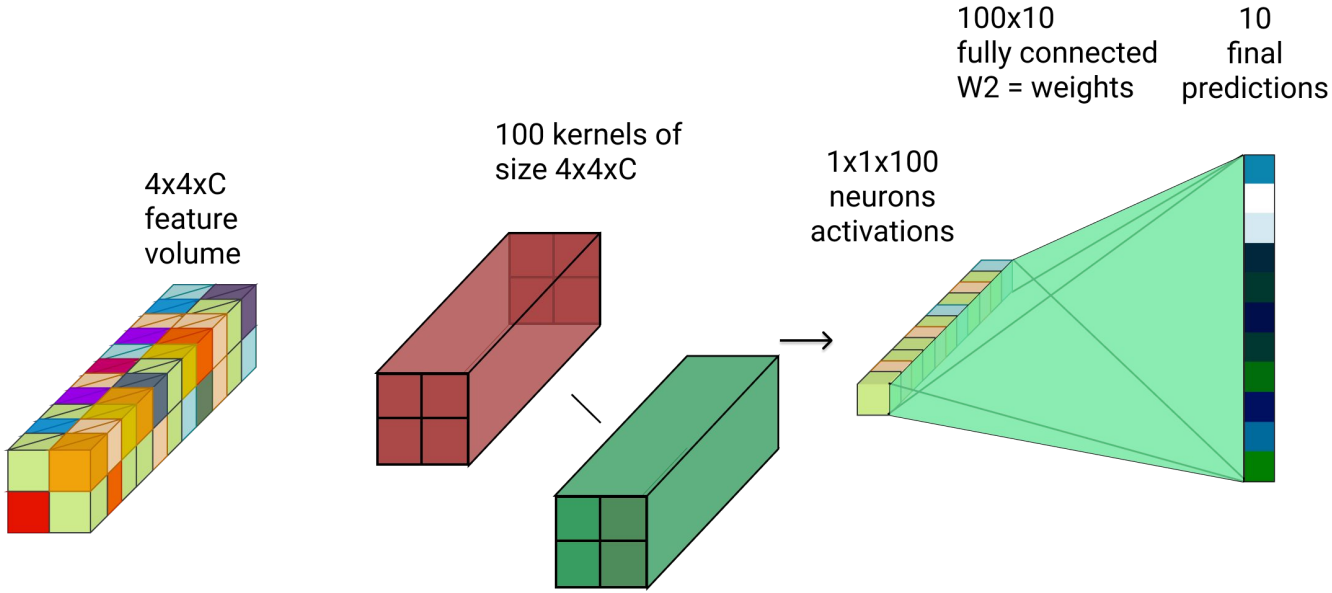


- Use the same parameters from the fully connected but reshape them to be used as convolutional parameters

- From (4*4*C)x100 matrix
  - To 100 kernels of size 4x4xC

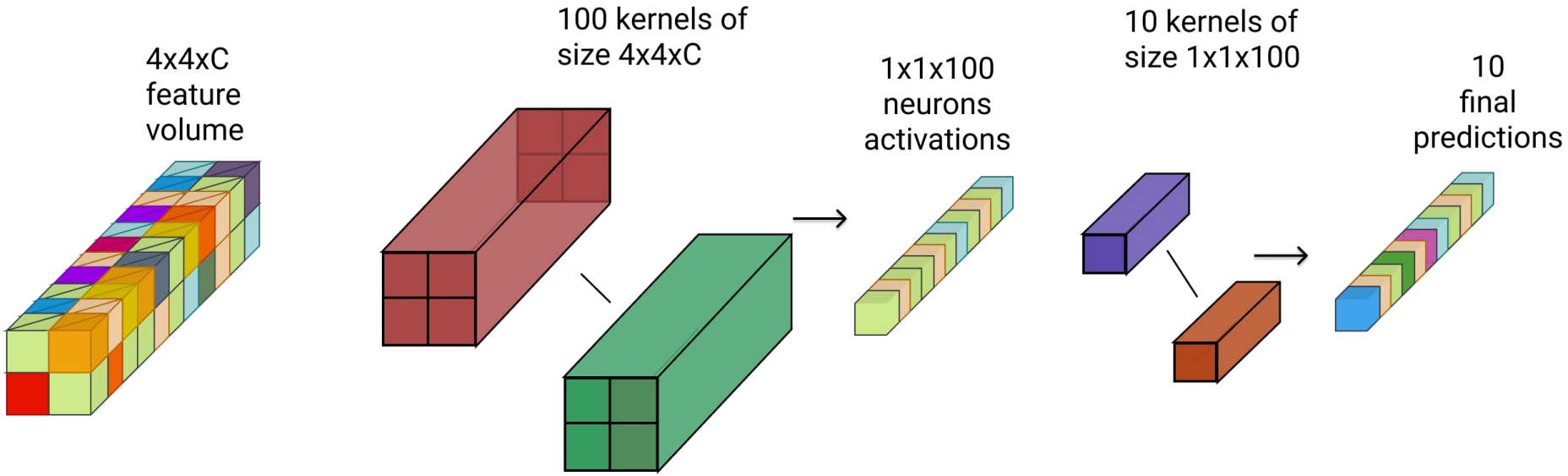
Fully-connected -> convolutional

- From a input with 2 spatial dimensions H x H and some channels C we obtain a single vector with 100 channels
- Resulting vector can be seen as a single 1x1 spatial location with 100 channels



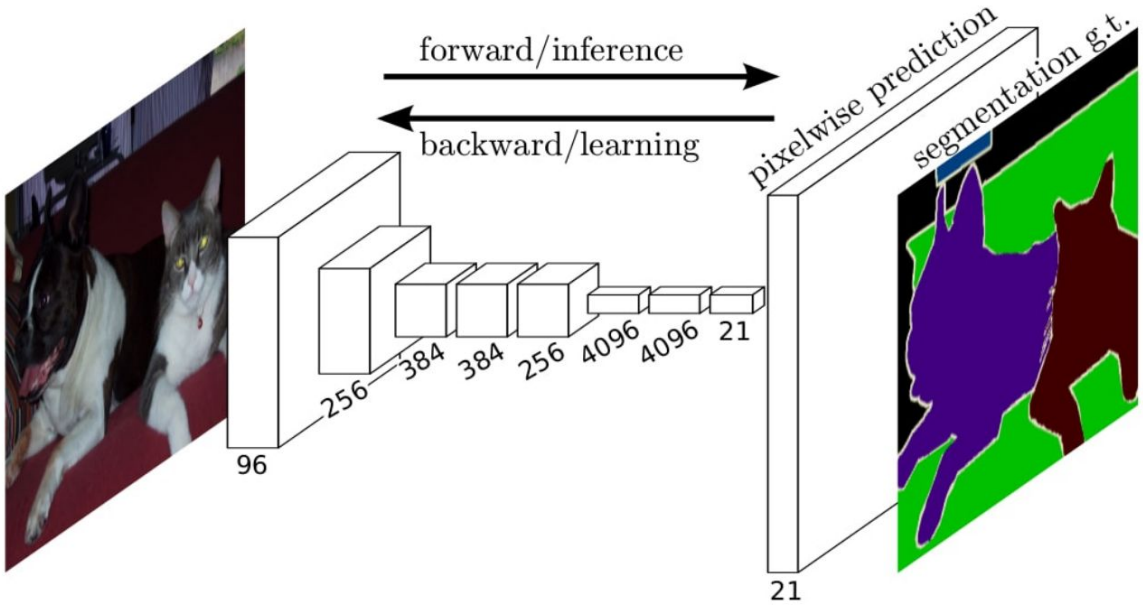
Fully Convolutional Network (FCN)

- All fully connected have been replaced by convolutions
- This network would work on any input size



Fully Convolutional Network (FCN)

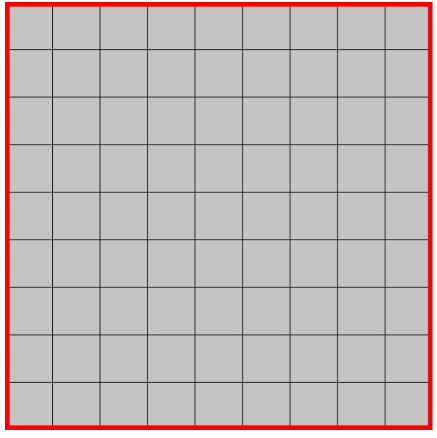
- All fully connected have been replaced by convolutions
- This network would work on any input size



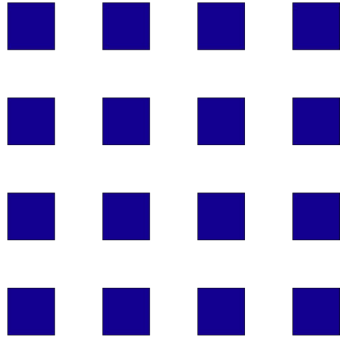
Problem: Sparse Output

- Due to the strides the output is sparser than the input
  - Every stride=k layer makes the output k^2 sparser
- For segmentation we want dense output
  - Every pixel should have a label

Input

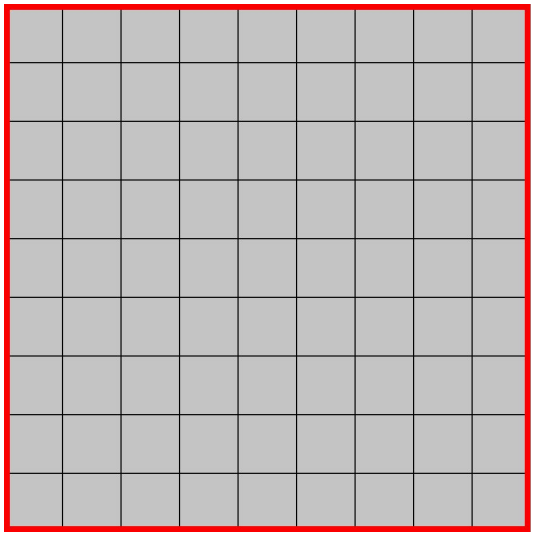


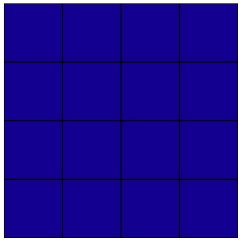
Output



Sparse Output

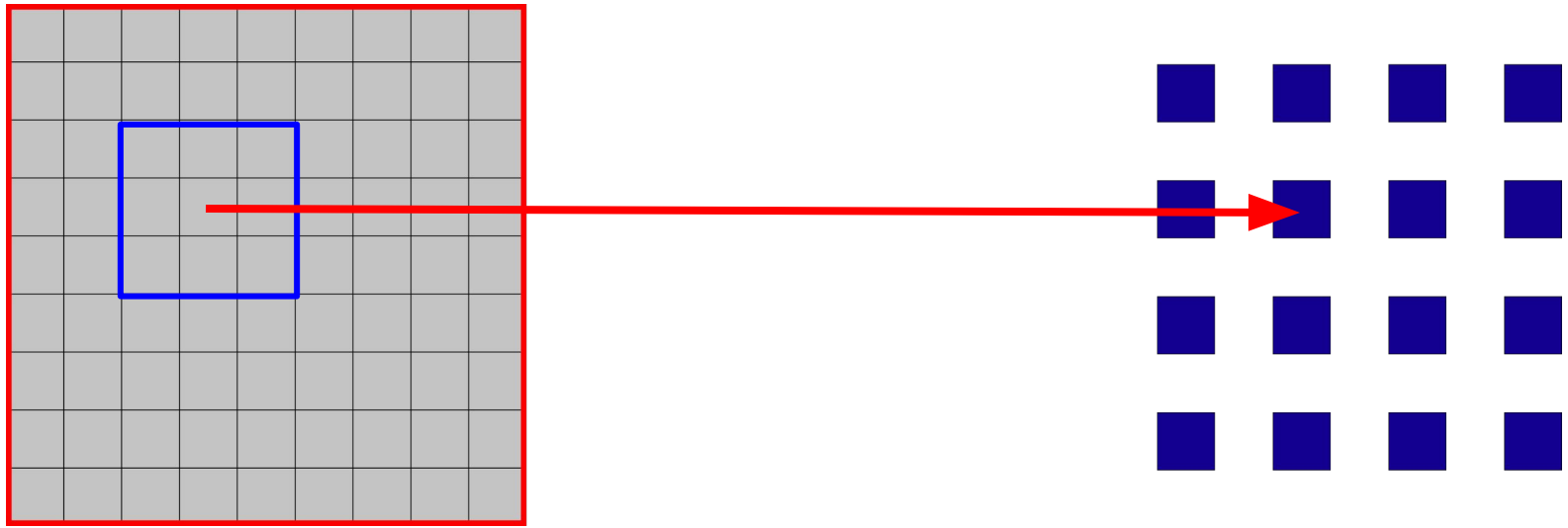
- Solution 1: shift and stitch
  - Shift the input by 1 pixel and get a different output





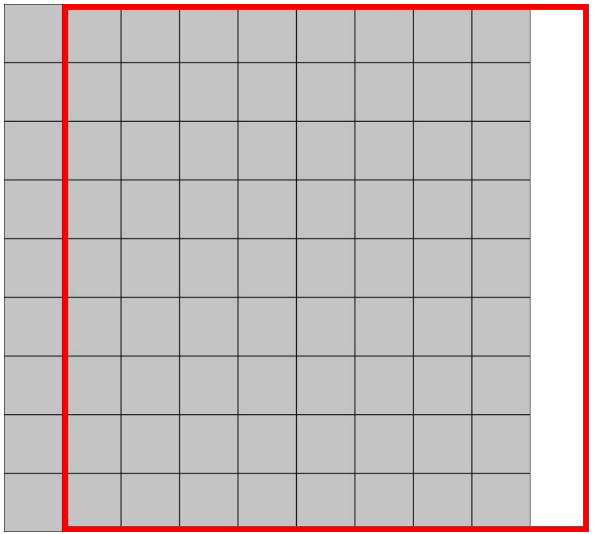
Sparse Output

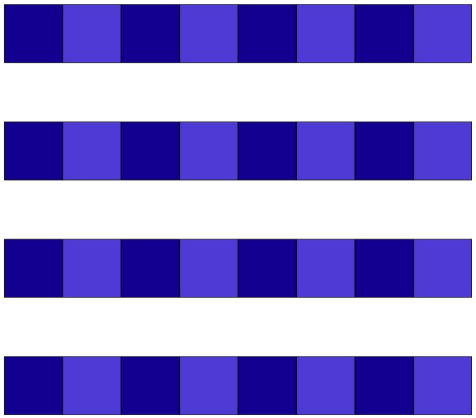
- Solution 1: shift and stitch
  - Shift the input by 1 pixel and get a different output



Sparse Output

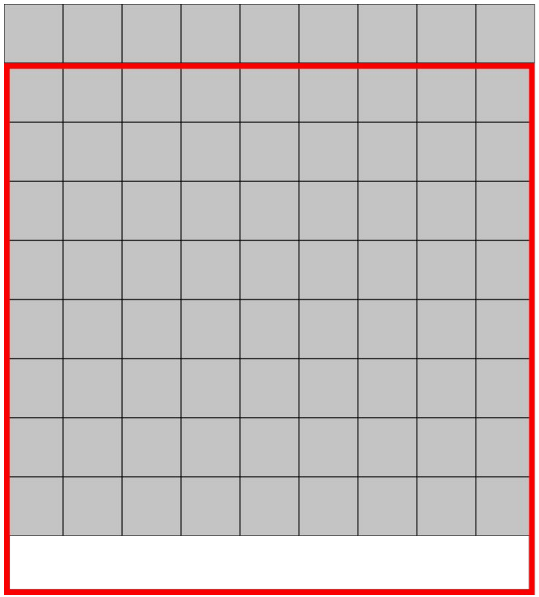
- Solution 1: shift and stitch
  - Shift the input by 1 pixel and get a different output

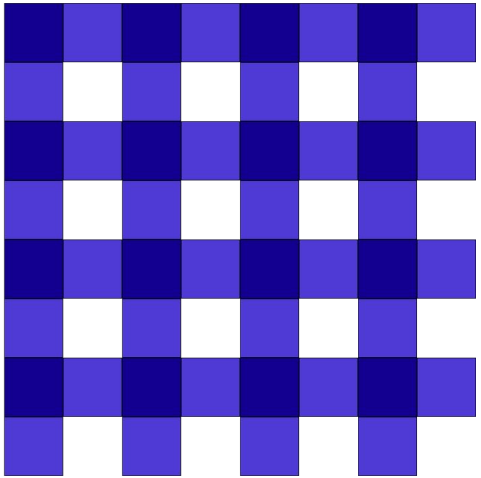




Sparse Output

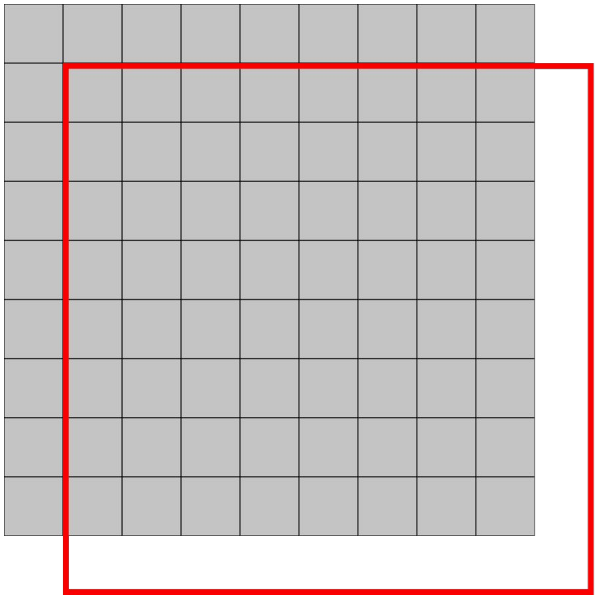
- Solution 1: shift and stitch
  - Shift the input by 1 pixel and get a different output

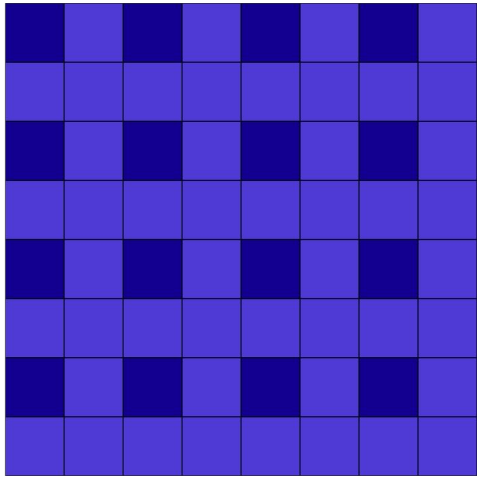




Sparse Output

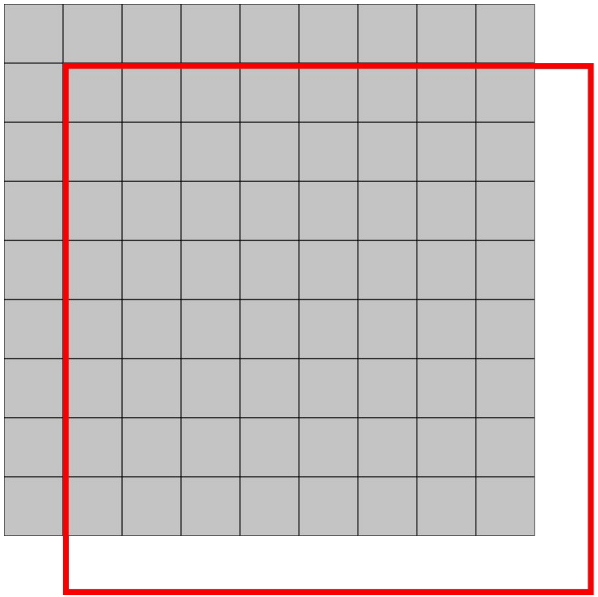
- Solution 1: shift and stitch
  - Shift the input by 1 pixel and get a different output

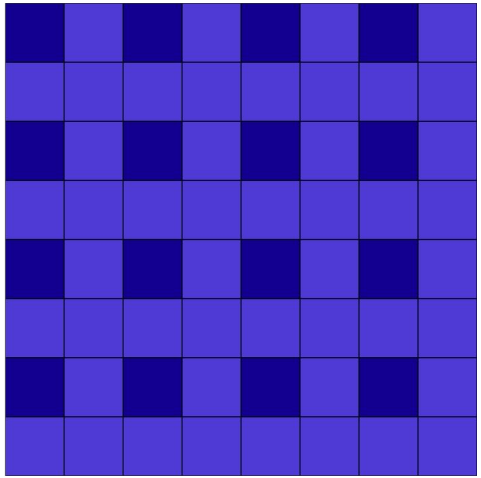




Sparse Output

- Solution 1: shift and stitch
  - Shift the input by 1 pixel and get a different output

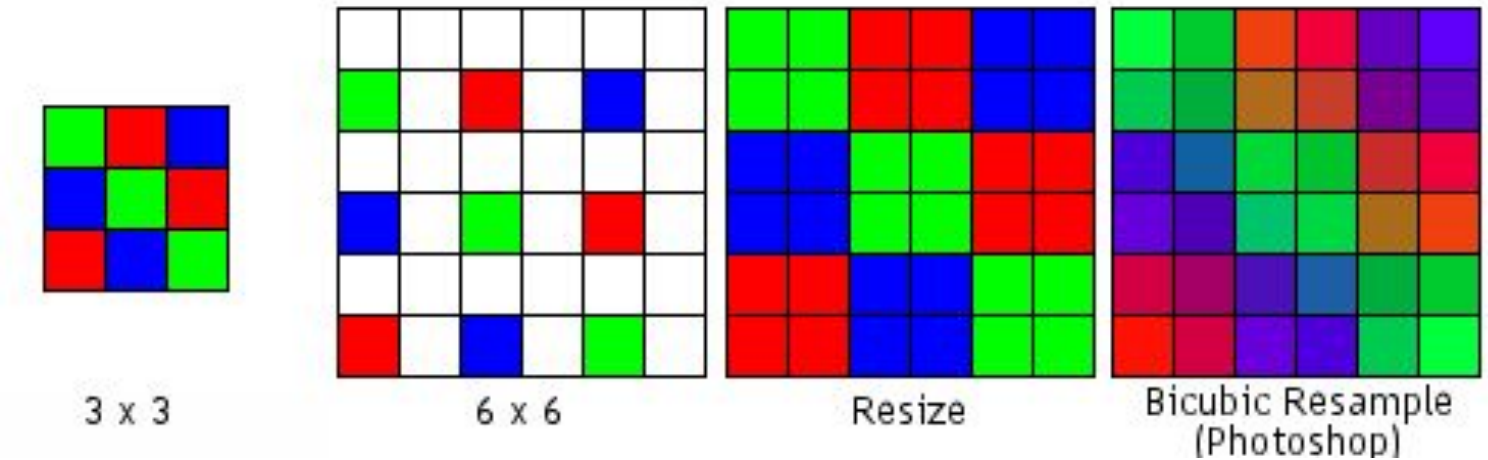




- Computational intensive: for t layers with stride=k
  - => k^{2t} passes

Sparse Output

- *Solution 2*: upsample via interpolation
- Pro: Simple and efficient
- Con: can not capture complex upscaling



Sparse Output: Deconvolutional Layers

- *Solution 3*: use deconvolutional layer
- Name: Deconvolution == Backwards Convolution == Transposed Conv == Fractionally strided Conv
- A deconv layer has ‘inverse’ operations as a convolution.
- A filter is multiplied with a point in the input and the resulting matrix is added in the corresponding region in the output
- Apply stride K in the output: => output is K^2 times larger

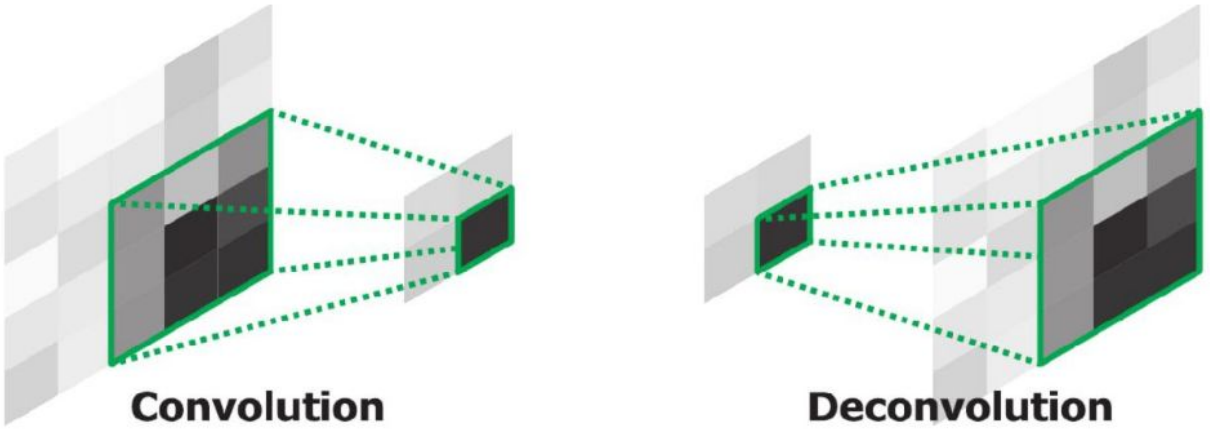
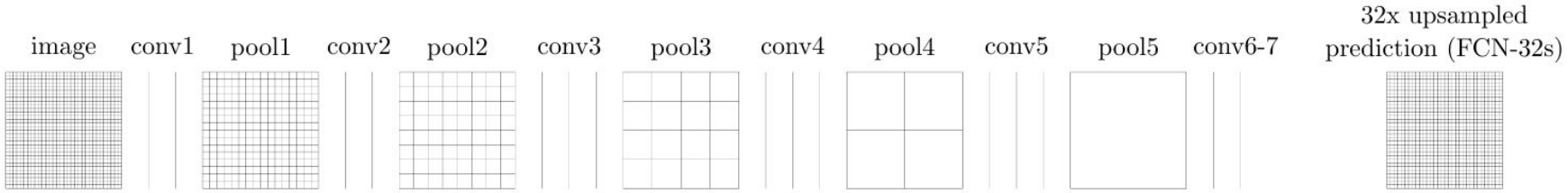


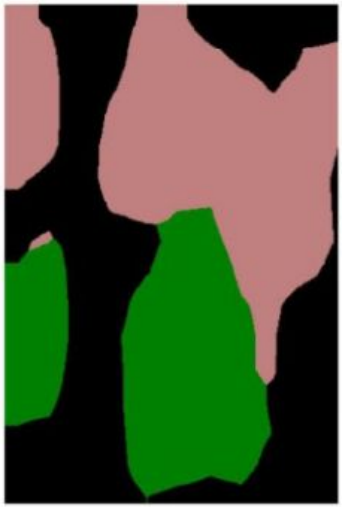
Image from: Noh et al. [2015]: Learning deconvolution network for semantic segmentation

Fully Convolutional Network (FCN)



- By converting the fully connected layers into convolutions we have a completely convolutional network that can be applied on images of any size
- The segmentation is rough because it results from upsampling *a law resolution* output

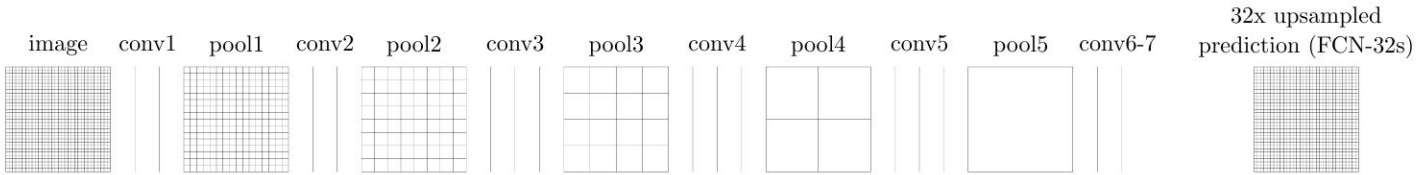
- *Solution*: use information from intermediate layers in the network, where low-level details are present



Long et al.[2014]: Convolutional networks for semantic segmentation.

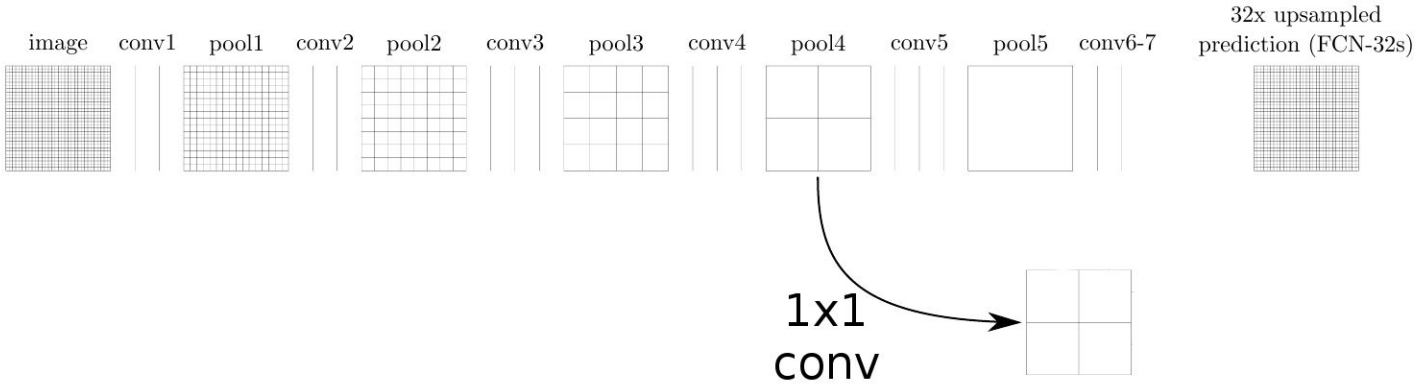
Skip Layers

- Combine the outputs of an upper and an intermediate layer
- Q: Why would low-level features help?
    - Useful for aligning the output with image boundaries, or with part of objects
- Use a 1x1 convolutional layer on features from lower levels to produce class predictions
- Upsample and add prediction scores from different levels then do a softmax



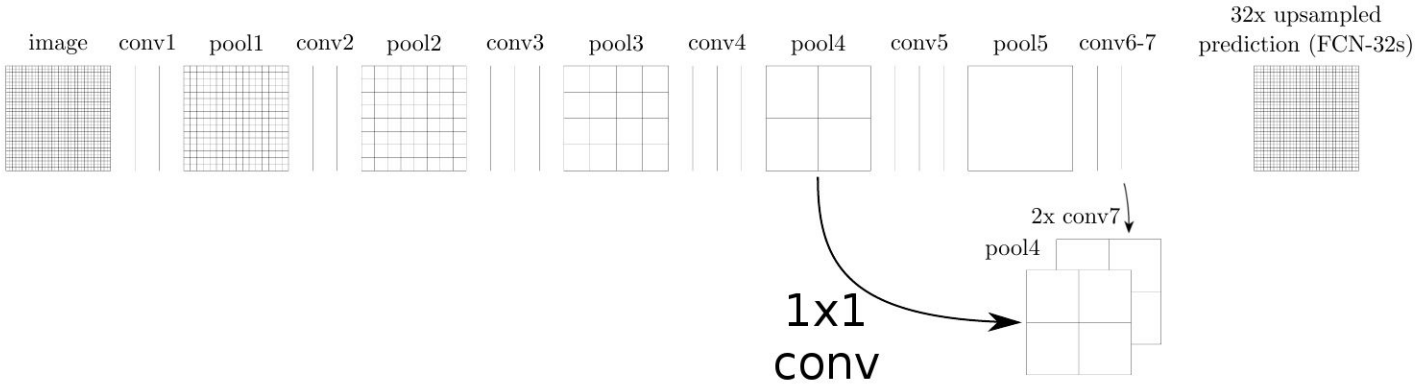
Skip Layers

- Combine the outputs of an upper and an intermediate layer
- Q: Why would low-level features help?
    - Useful for aligning the output with image boundaries, or with part of objects
- Use a 1x1 convolutional layer on features from lower levels to produce class predictions
- Upsample and add prediction scores from different levels then do a softmax



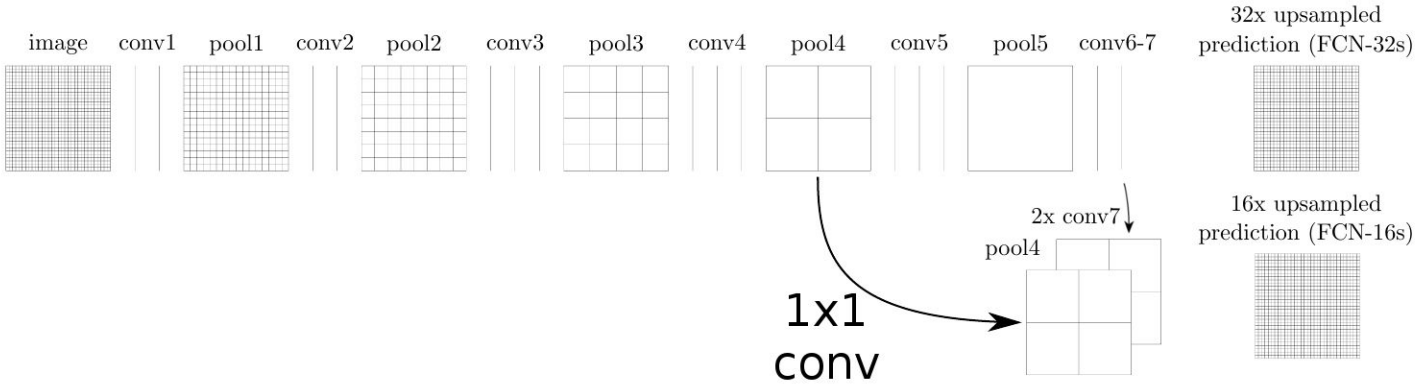
Skip Layers

- Combine the outputs of an upper and an intermediate layer
- Q: Why would low-level features help?
    - Useful for aligning the output with image boundaries, or with part of objects
- Use a 1x1 convolutional layer on features from lower levels to produce class predictions
- Upsample and add prediction scores from different levels then do a softmax



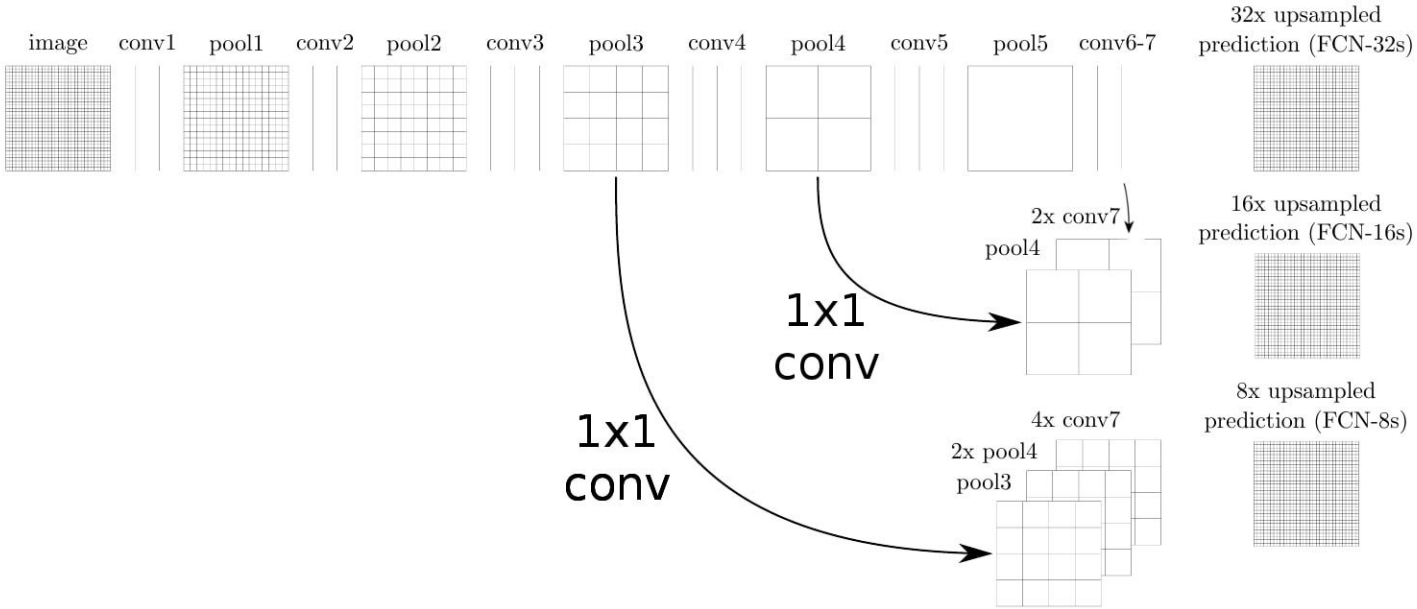
Skip Layers

- Combine the outputs of an upper and an intermediate layer
- Q: Why would low-level features help?
    - Useful for aligning the output with image boundaries, or with part of objects
- Use a 1x1 convolutional layer on features from lower levels to produce class predictions
- Upsample and add prediction scores from different levels then do a softmax



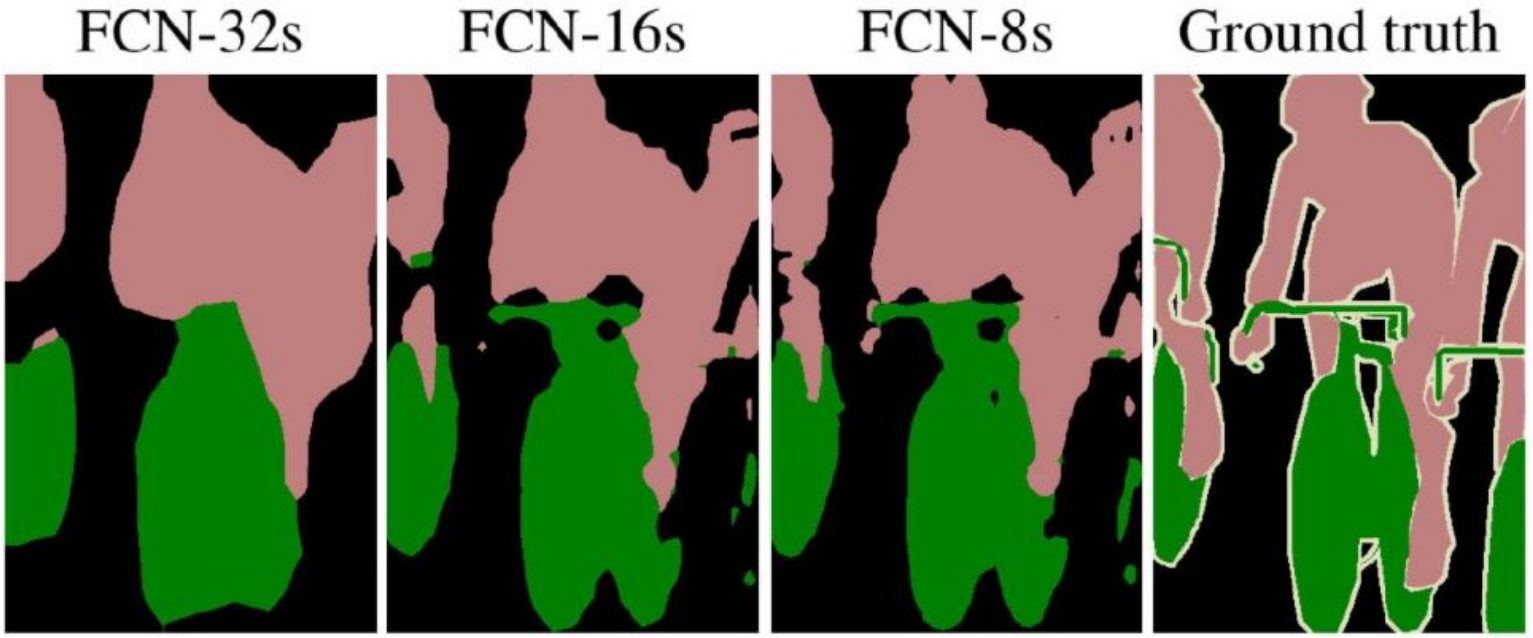
Skip Layers

- Combine the outputs of an upper and an intermediate layer
- Q: Why would low-level features help?
    - Useful for aligning the output with image boundaries, or with part of objects
- Use a 1x1 convolutional layer on features from lower levels to produce class predictions
- Upsample and add prediction scores from different levels then do a softmax



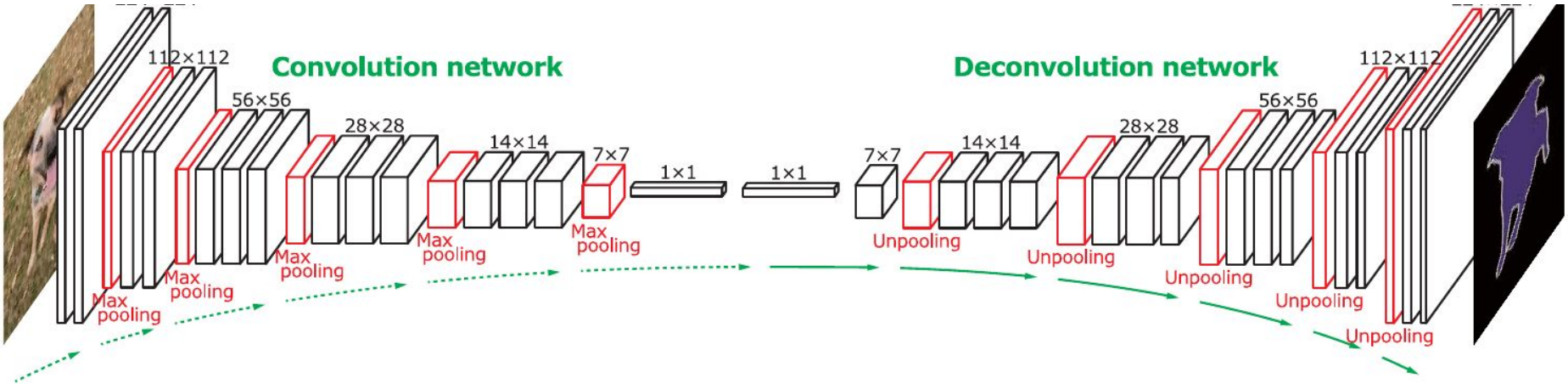
Results

- Using lower level features improves the level of details (resolution) of the output



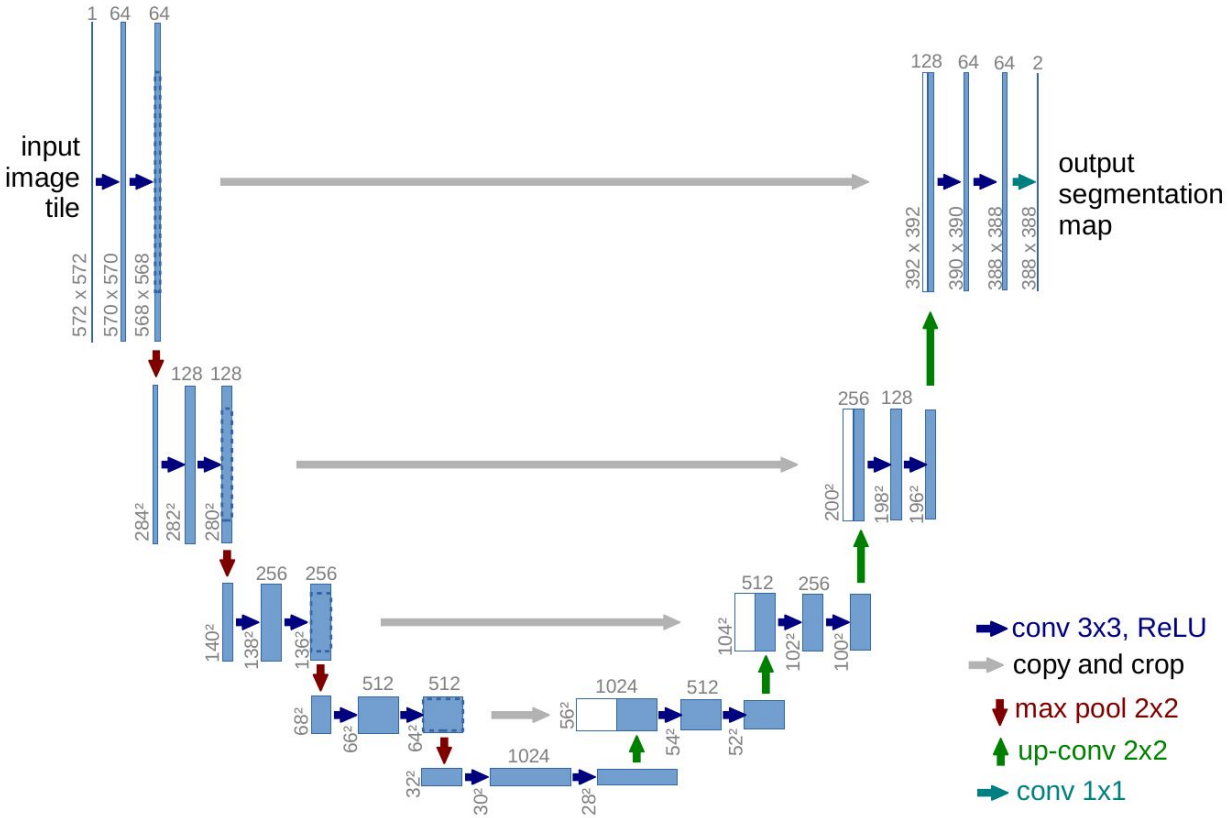
Deconvolutional Network

- Use deconvolutions to sample features maps
    - not just the final segmentation map
- Use an upsampling deconvolutional part that ‘mirrors’ the convolutional part



Noh et al. [2015]: Learning deconvolution network for semantic segmentation

U-net: using (again) intermediate features



- The contractive, convolutional part extracts information about the object
- Expansive part has the role of inferring the exact segmentation shape
- Upscale the convolutional maps in multiple layers
- The middle convolutional features are highly semantic but have low resolution - poor localisation
- Use previous convolutional features for fine-grained spatial information

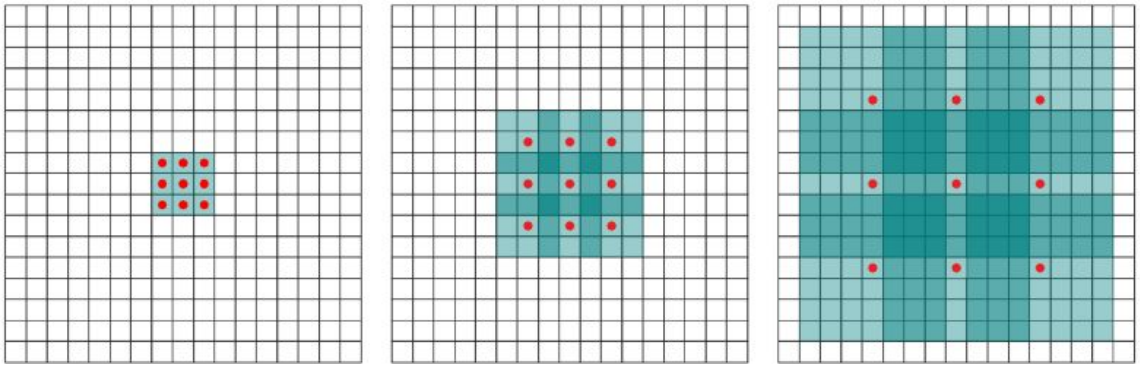
Ronneberger [2015]: U-net: Convolutional networks for biomedical image segmentation

Extra: Dilated Convolutions

- Operation designed for dense prediction
  - used in networks without downsample layers (pooling, strided comvs)
- Aggregate distant (contextual) information without losing input resolution

Normal Conv: $$(F * k)(p) = \sum_{s+t=p} F(s)k(t) \qquad t \in [(-1,-1),(-1,0),\ldots(1,1)]$$

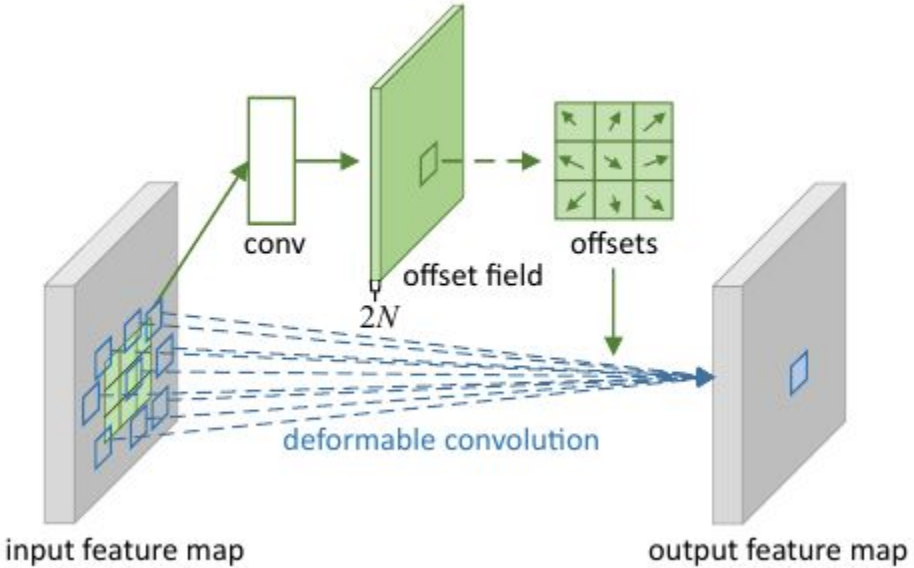
Dilated Conv: $$(F *_l k)(p) = \sum_{s+l\cdot t=p} F(s)k(t)$$ l - dilation factor

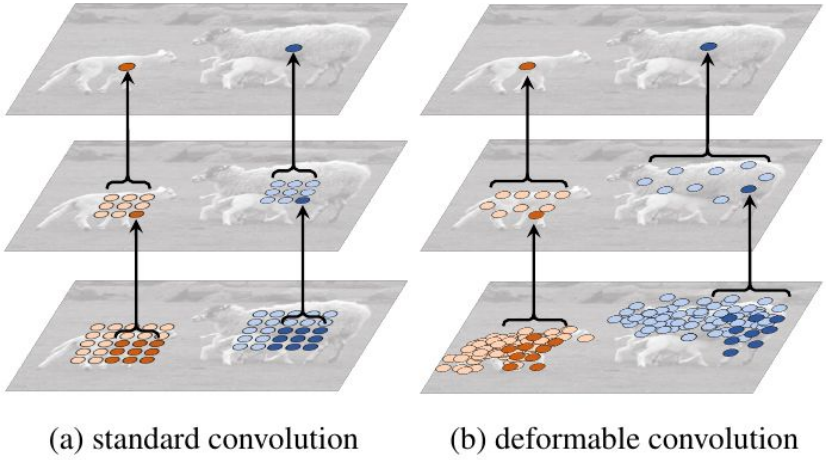


Yu and V. Koltun.[2015] Multi-scale context aggregation by dilated convolutions

Extra: Deformable Convolutions

- Learn to predict an offset for every point in the kernel



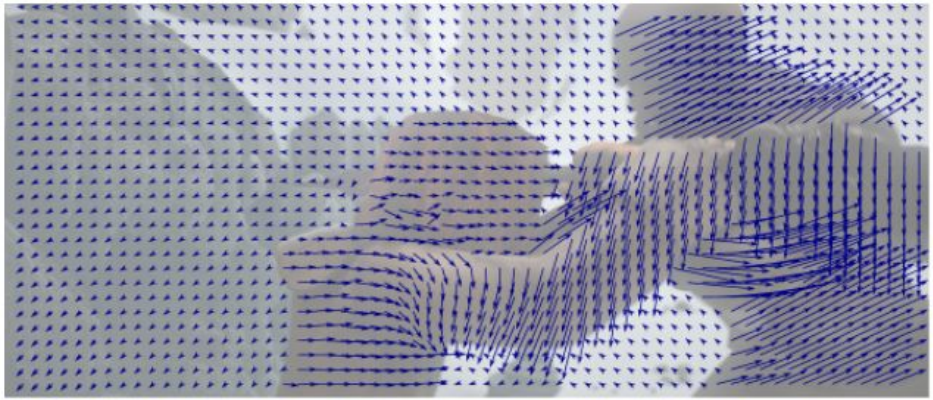


J. Dai et al. [2017] Deformable Convolutional Networks

Optical Flow

- Given 2 frames from a video, how does every pixel in the first image mode

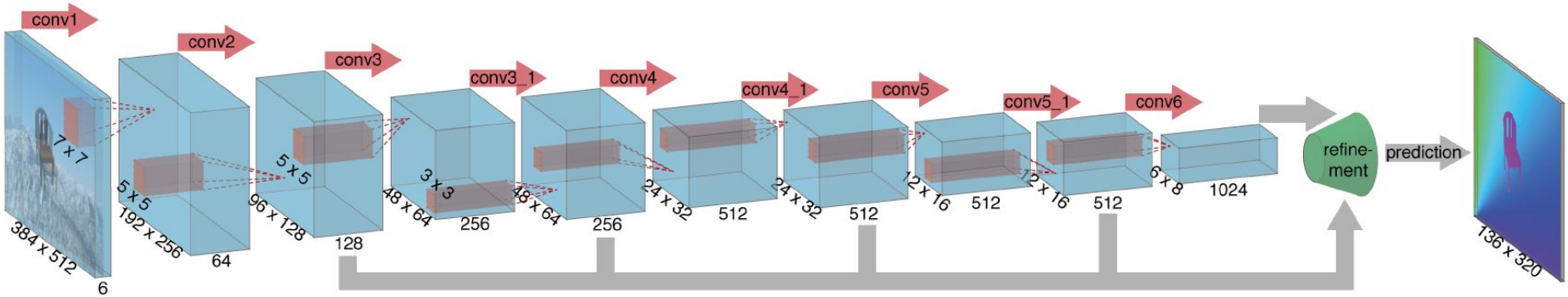
- For every position in the first image find an offset such such that it points to the same point in the second image



Flow Net

- Predict the optical flow (2 maps: for x,y coordinates)
- Take as input 2 images: process them with convolutional-deconvolutional model

- FlowNet Simple: concatenate the 2 images at input
- Let the network learn to process the two images to extract motion information



Dosovitskiy et al. [2015]: Flownet: Learning optical flow with convolutional networks

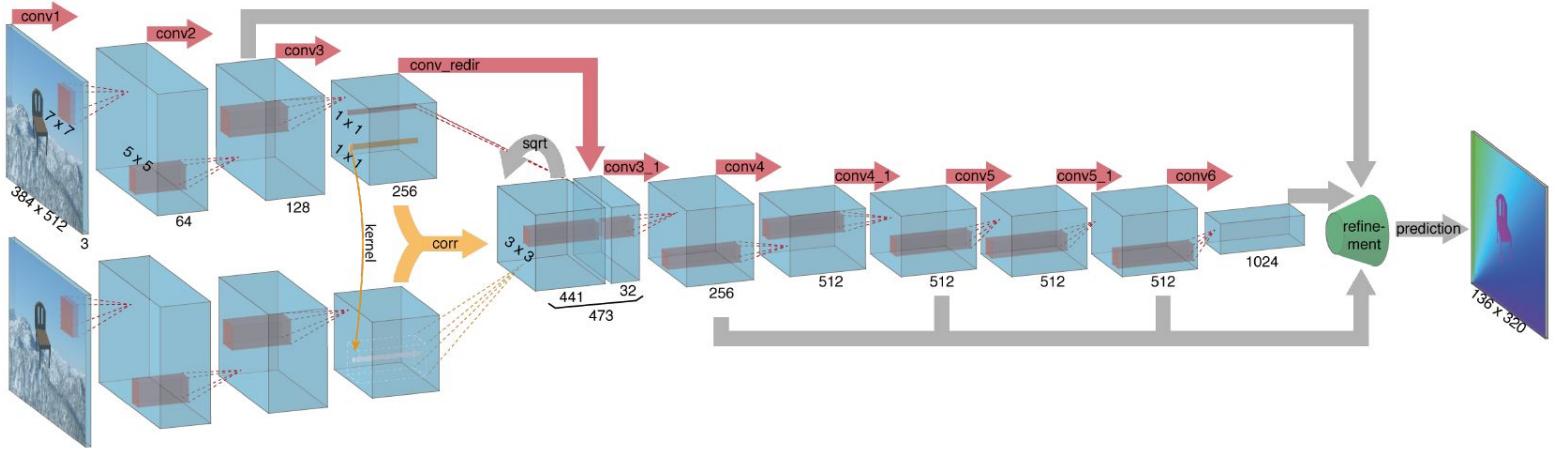
Flow Net-Correlations

- Explicit design the network to do matching
- Estimate correlations between patches in the first image and close patches in the second image

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle$$

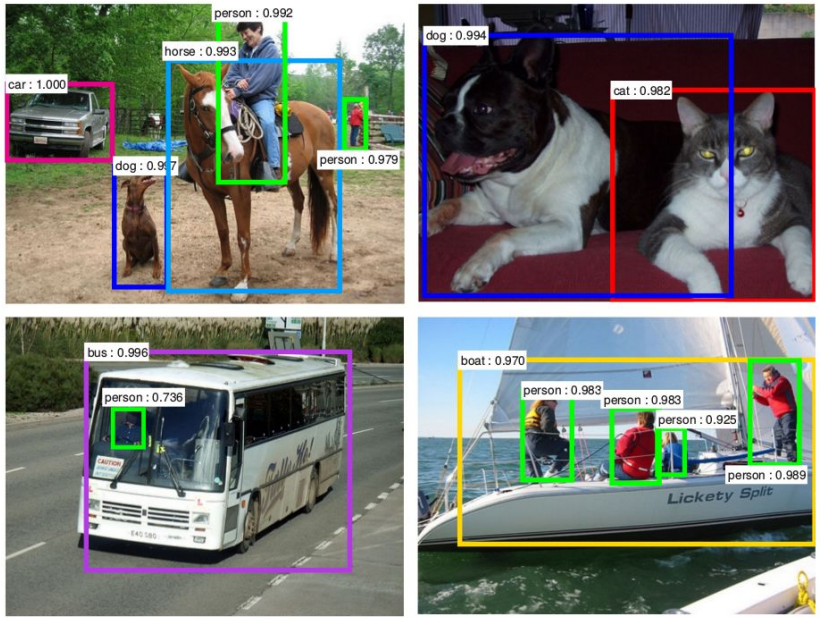
Dot product

- For every patch in the first image, compute correlations with the D^2 nearest patches from the second image
  - Keep all the resulted correlations as channels in a feature map



Detection

- Find every instance of certain objects in an image
- Draw a tight box around every object



Faster - RNN

- Learn to predict a box for every object using a neural network
- First learn to predict boxes of regions most probable to contain objects (**region proposals**)

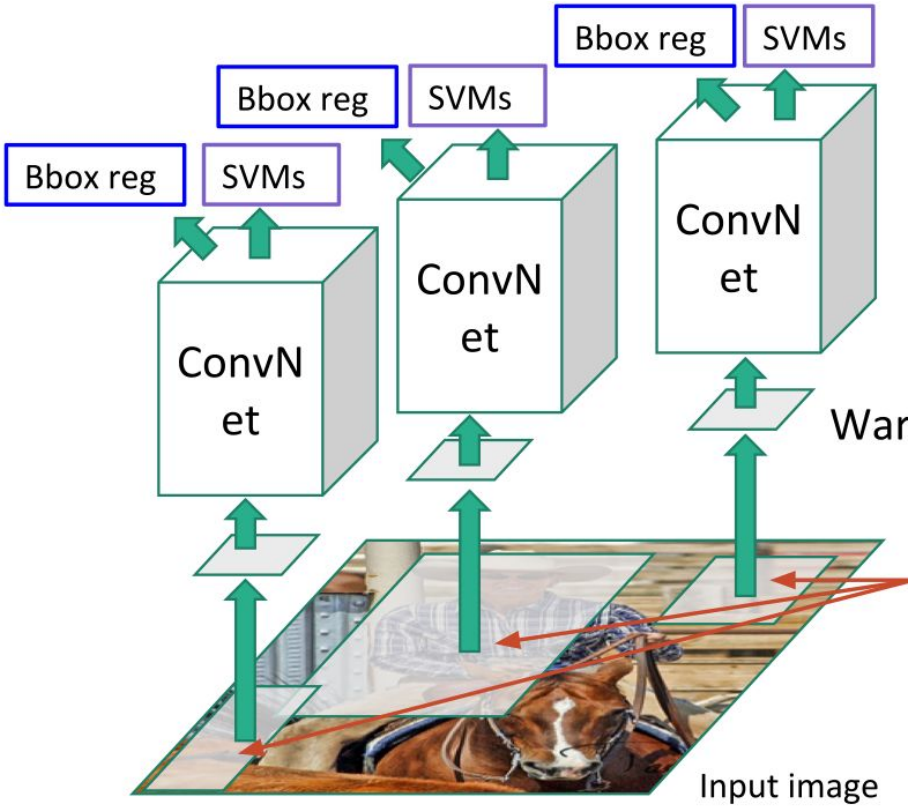
S. Ren et al.[2015]: Faster r-cnn: Towards real-time object detection with region proposal networks

Faster - RNN

- Learn to predict a box for every object using a neural network
- First learn to predict boxes of regions most probable to contain objects (**region proposals**)
- For the most probable regions predict if it contains an object and refine the box

S. Ren et al.[2015]: Faster r-cnn: Towards real-time object detection with region proposal networks

Detection: R-CNN

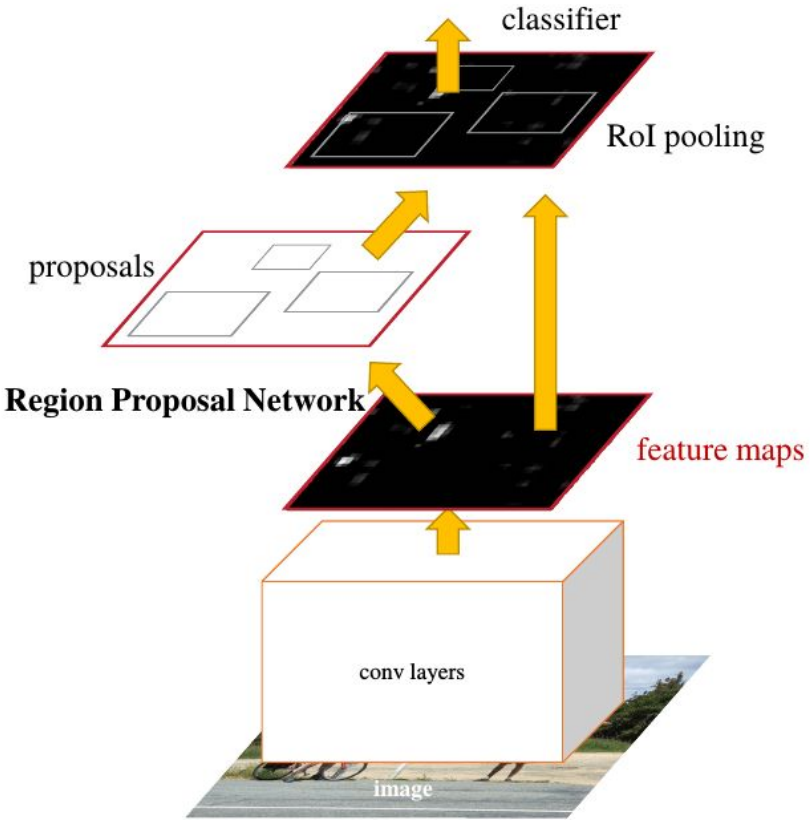


- Train a Network for classification
- Run network on **most probable** regions (computed by other methods)

- Problem:
  - SLOW
  - MANY redundant operations
- Solution:
  - For every image compute features for the entire image with fully convolutional network
  - Predict region proposals from the same conv network

Girshick et al. [2014]: Rich feature hierarchies for accurate object detection and semantic segmentation.

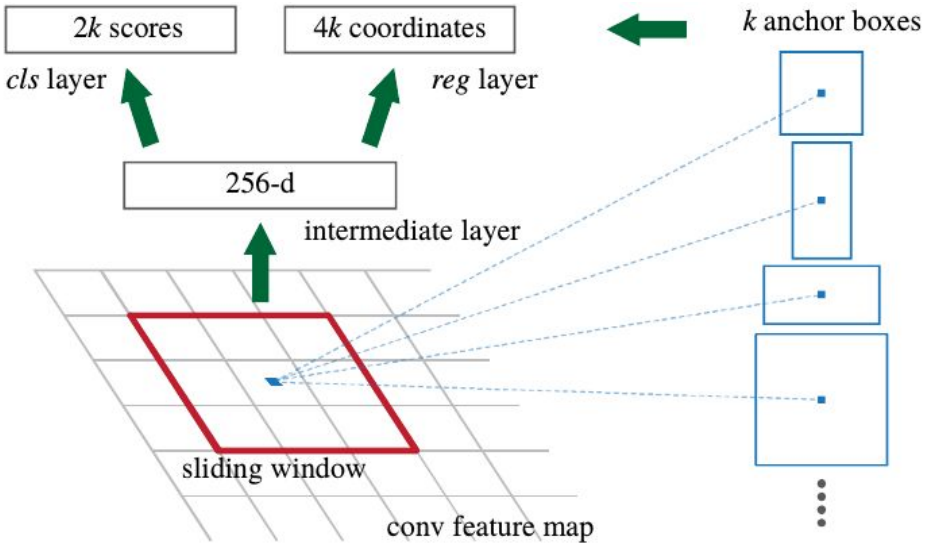
Detection: Faster - RNN



- Start with a convolutional network called **backbone** to extract features from the image
- At each point in the resulted feature maps predict a set of box proposals using a network called **Region Proposal Network**
- From the features corresponding the whole image **extract features** corresponding to the best proposals
- Process the extracted box features with fully connected layers and predict the object class and the final box

S. Ren et al.[2015]: Faster r-cnn: Towards real-time object detection with region proposal networks

Region Proposal Network (RPN)



- From every 3x3 patch of features predict a box
  - 3x3 patch has a sufficiently large receptive field to cover most objects (228 pixels wide compared to 600 pixels of the input)
- Predict the existence of boxes around k anchor boxes of different scales and aspect ratios

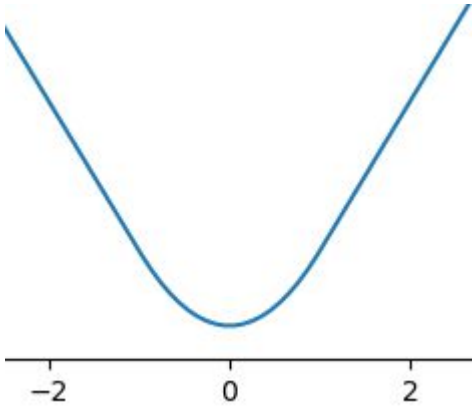
- Fully convolutional network:
  - 3x3 convolutional layer
  - 1x1 conv layer for predicting **objectness scores**
  - 1x1 conv layer for predicting **box coordinates**

S. Ren et al.[2015]: Faster r-cnn: Towards real-time object detection with region proposal networks

Region Proposal Network (RPN)

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

- L_{cls} : classification loss
  - Negative loss likelihood
- L_{reg}: regression loss
  - L_2 or L_1 loss
  - L_1 Smooth loss



p_i : predicted objectness score

t_i : predicted coordinates

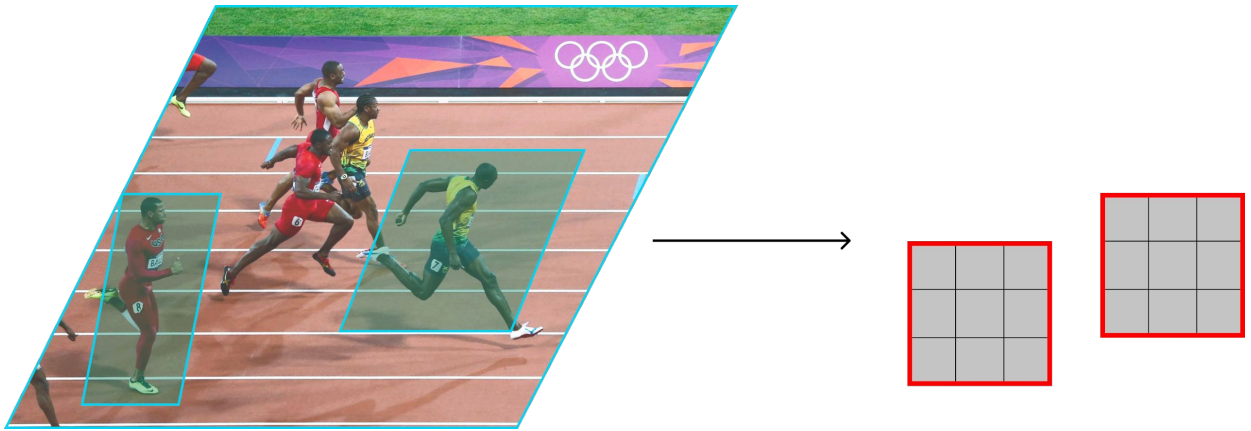
p_i^* : Ground Truth object existence in the region

t_i^* : GT coordinates of the region

RoI Pooling

- Select features corresponding to a region of interest (ROI)

- Represent that region as a grid of **fix size** (eg. 3x3):
  - Each point in the grid is formed by max pooling a small region of features



Final box prediction

- From the extracted ROI features predict the class of the objects and the final box coordinates
- Predict C+1 values for the C classes and one for background
- Predict a set of box coordinates for every class
- Like in RPN combine classification loss and coordinate regresion loss

- RPN and the backbone share the fully convolutional operations
- The final classification is **independent** for every proposal (but the parameters are shared)

Instance Segmentation

- Q: Semantic Segmentation vs Instance Segmentation?
  - Semantic segmentation: pixels of a certain class
  - Instance segmentation: pixels of each individual instance separately

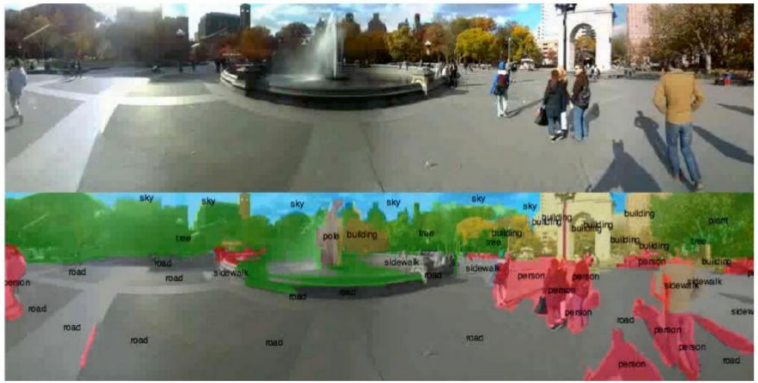
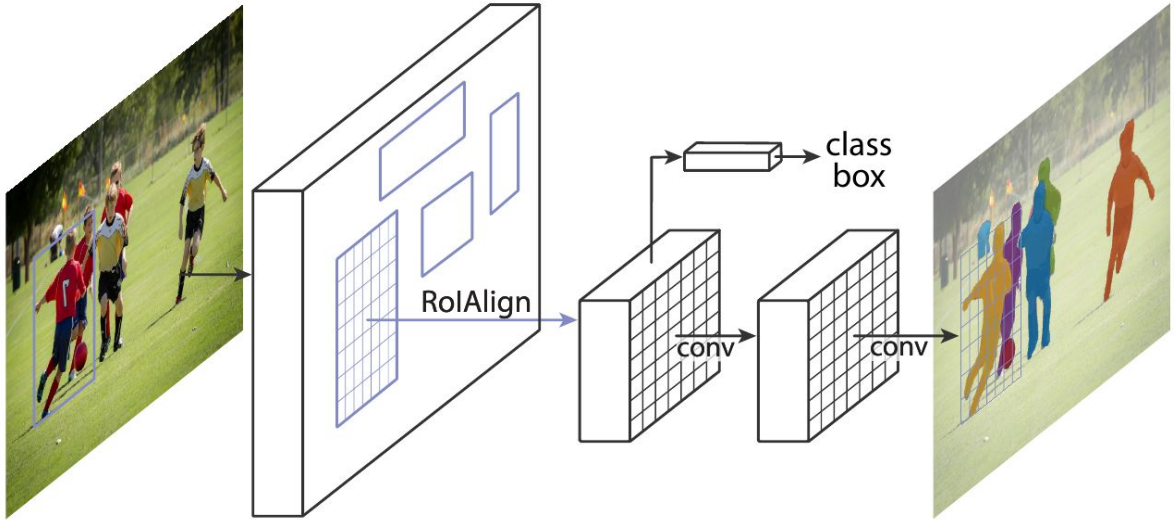


Image source: Pinheiro et al. [2016] and Farabet et al. [2013]

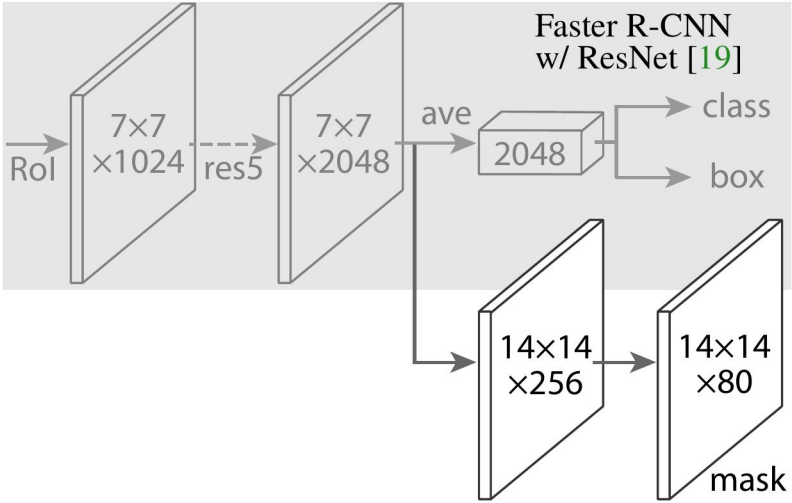
Mask R-CNN

- Use an additional segmentation branch in Faster R-CNN
- For each proposal given gy RPN estimate a segmentation mask



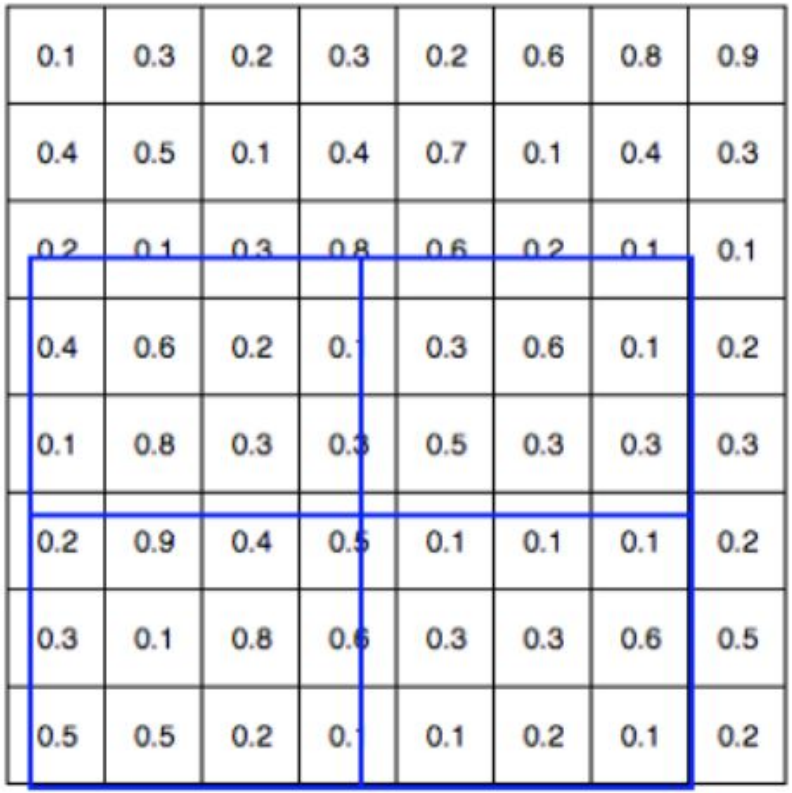
K He et al. [2017]:Mask r-cnn

Mask R-CNN



- After the RPN extract 7x7 patch features from **every** region of interest
- Predict the final **class** and the box **coordinates**
- Using fully convolutional network predict a segmentation mask corresponding to the object in the ROI

RoIPool vs RoIAlign:



Region of Interest (RoI): region in the input that we want to consider

- For a given ROI (blue)
  - Extract features from the feature map (gray)
- Features have lower resolution than the input
  - Each region in the input space does not have an exact correspondence at low resolution (e.g. the desired region falls between the feature map grid)

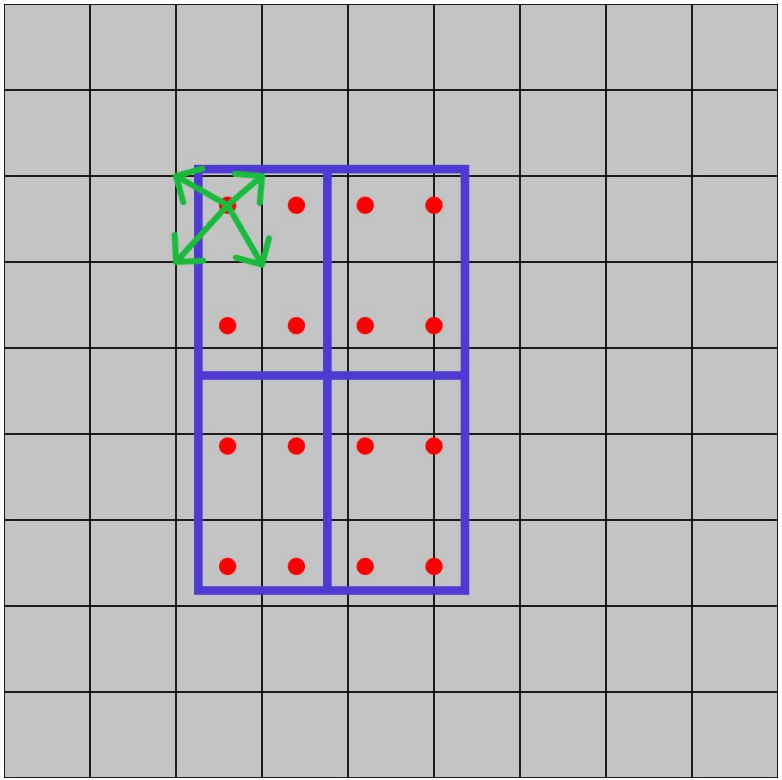
RoIPool vs RoIAlign:

0.1	0.3	0.2	0.3	0.2	0.6	0.8	0.9
0.4	0.5	0.1	0.4	0.7	0.1	0.4	0.3
0.2	0.1	0.3	0.8	0.6	0.2	0.1	0.1
0.4	0.6	0.2	0.1	0.3	0.6	0.1	0.2
0.1	0.8	0.3	0.3	0.5	0.3	0.3	0.3
0.2	0.9	0.4	0.5	0.1	0.1	0.1	0.2
0.3	0.1	0.8	0.6	0.3	0.3	0.6	0.5
0.5	0.5	0.2	0.1	0.1	0.2	0.1	0.2

Region of Interest (RoI): region in the input that we want to consider

- For a given ROI (blue)
  - Extract features from the feature map (gray)
- Features have lower resolution than the input
  - Each region in the input space does not have an exact correspondence at low resolution (e.g. the desired region falls between the feature map grid)
- **RoIPool**:
  - Discretize the correct RoI to fit the low resolution
  - From the discrete region extract a **2x2 patch** by mean / max pooling

RoIPool vs RoIAlign:



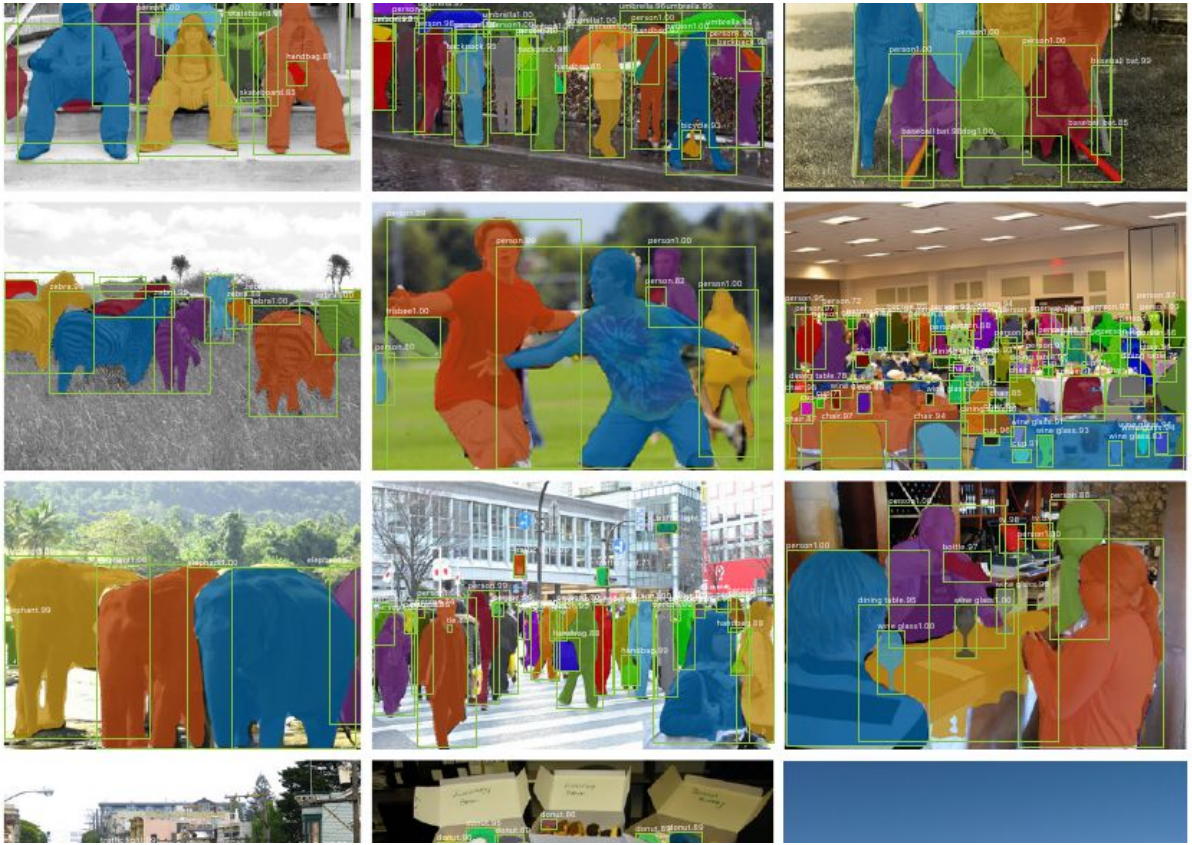
- For a given RoI (blue)
  - Extract features from the feature map (gray)

RoIAlign:

- Split the original RoI in NxN (e.g. 2x2) cells
- For each cell consider 4 points at fixed locations
  - **interpolate** their value by considering the closest neighbours in the feature grid
  - and aggregate them by max-pool or average

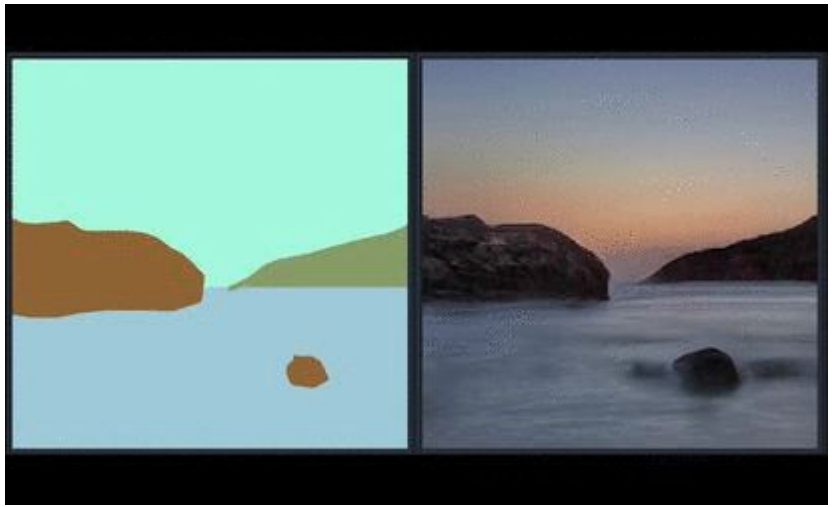
- By RoIAlign the extracted features are more aligned to the original object
  - The resulting segmentation would be more fine grained and more aligned to the object boundaries

Results



Bonus: Image Synthesis

- Generate image from segmentation Mask 



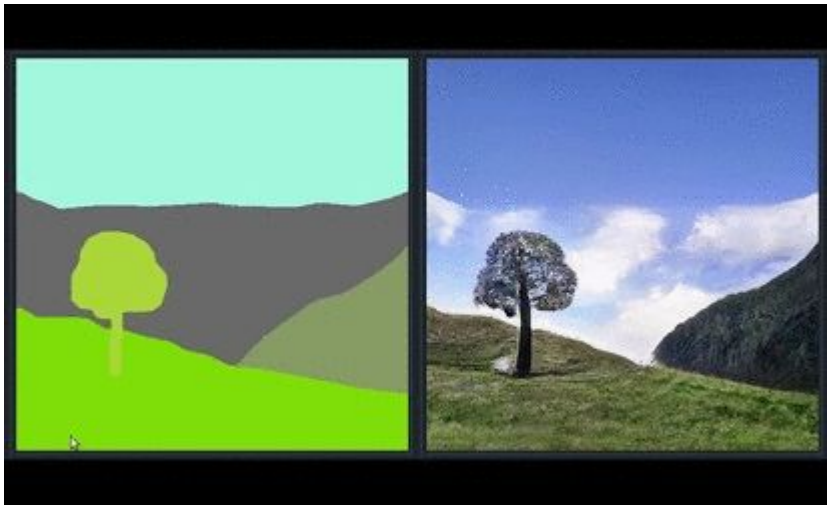
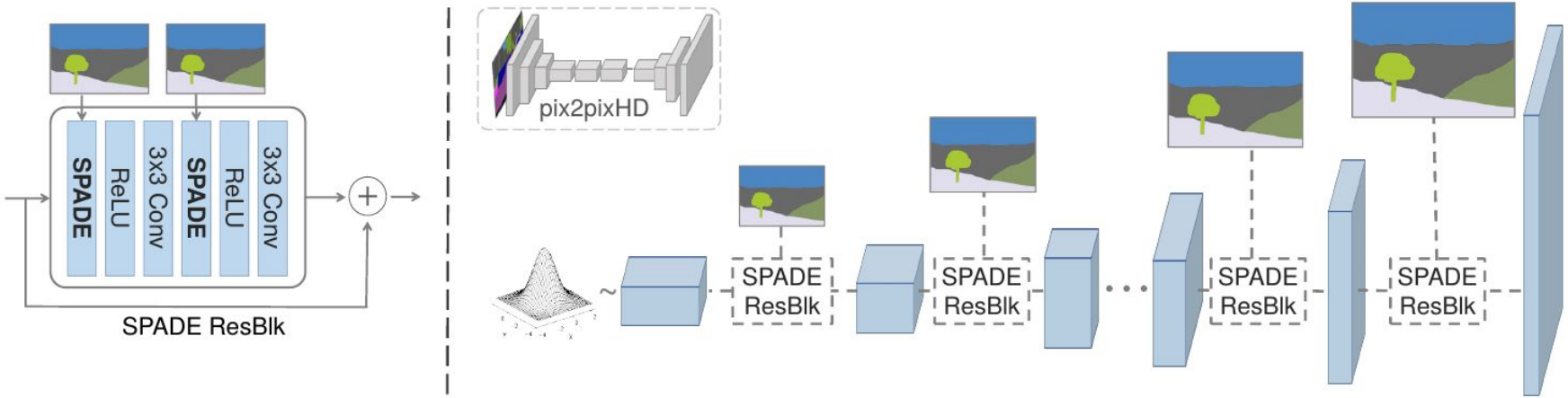
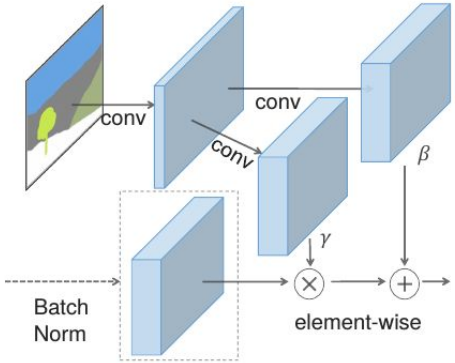


Image source: https://nvlabs.github.io/SPADE/

Bonus: Image Synthesis



- Use segmentation mask to modulate intermediate layers activations



Park et al. [2019]: Semantic image synthesis with spatially-adaptive normalization

Thank you!

(Next:Recurrent Neural Networks)

References

- A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas,V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox.Flownet: Learning optical flow with convolutional networks. InProceedings of the IEEE international conference on computer vision, pages 2758–2766, 2015.
- R. Girshick. Fast r-cnn. InProceedings of the IEEE international conference on computer vision, pages 1440–1448, 2015.
- R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. InProceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014.
- H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. InProceedings of the IEEE International Conference on Computer Vision, pages 1520–1528, 2015.
- T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019.
- S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99,2015.

References

- O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.
- F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions.arXiv preprint arXiv:1511.07122, 2015.
- Van der Laak, Jeroen, Geert Litjens, and Francesco Ciompi. "Deep learning in histopathology: the path to the clinic." Nature medicine 27.5 (2021)