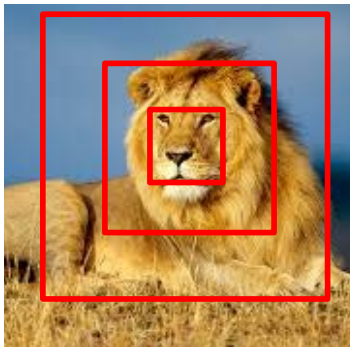
# CONVOLUTIONAL NEURAL NETWORKS --- ADVANCED W/ APPICATIONS

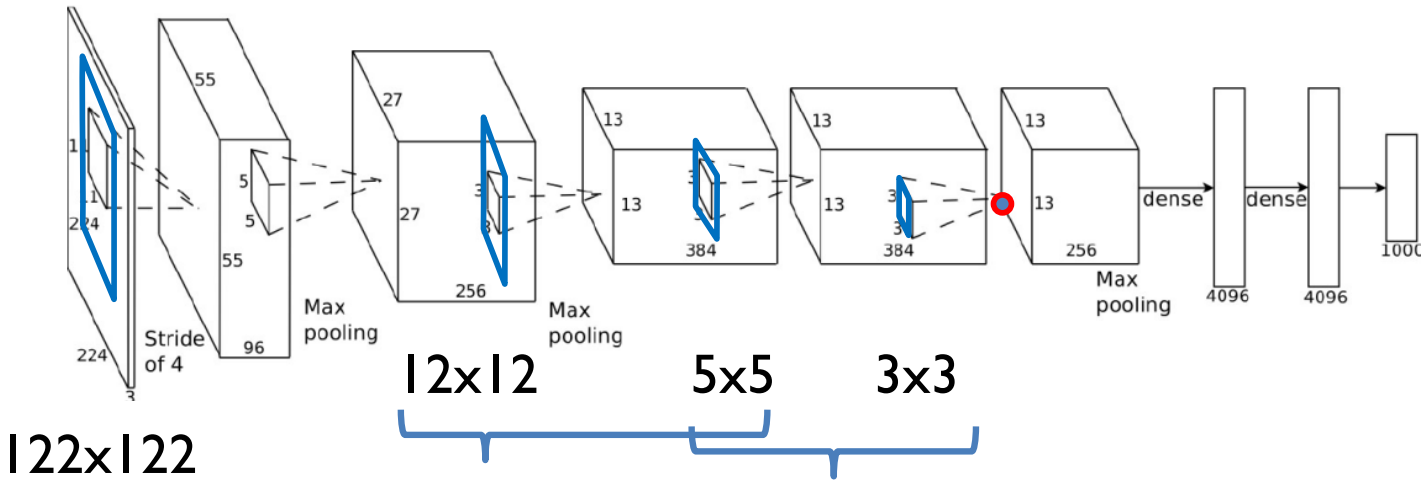
# Designing a CNN

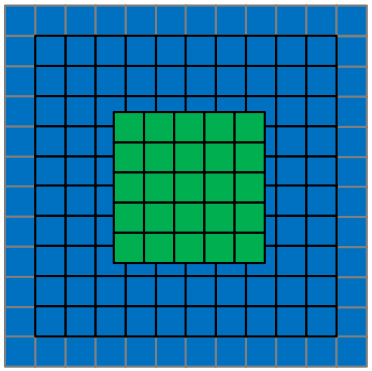
- Factors
  1. Number of layers, number of pooling/down-sampling
  2. Output layers w/ a loss
  3. Use of normalization layer such as batch normalization
  4. Use of additional structures, e.g., residual/skip connections, attention mechanism, etc.
- One (only?) thing we can (and should) control: the size of receptive field (of the last layer unit)
  - It is basically hard to predict the effects of your design
  - How large the local structures is that we need to consider to solve the task?
  - Set the receptive field size to their (maximum) size

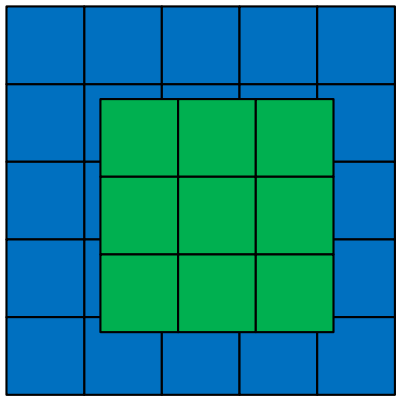


# Size of receptive field

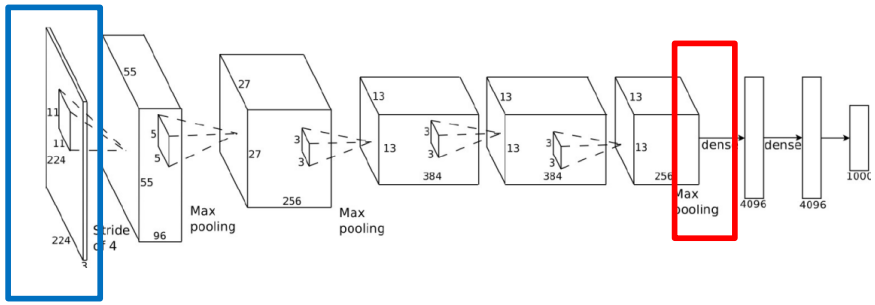
- Receptive field of a unit of a layer
  - The region in its input that will affect the unit

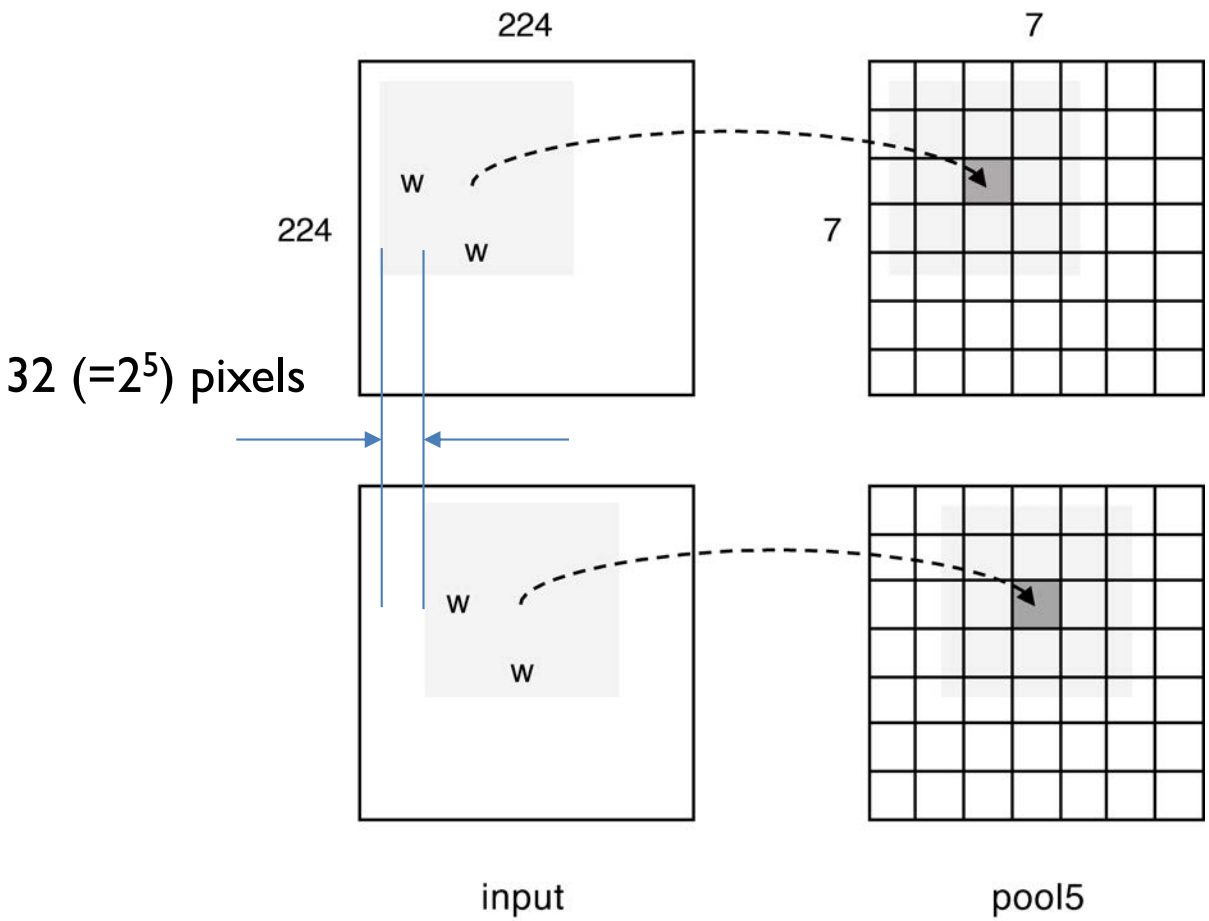




# Receptive field of the final pooling layer

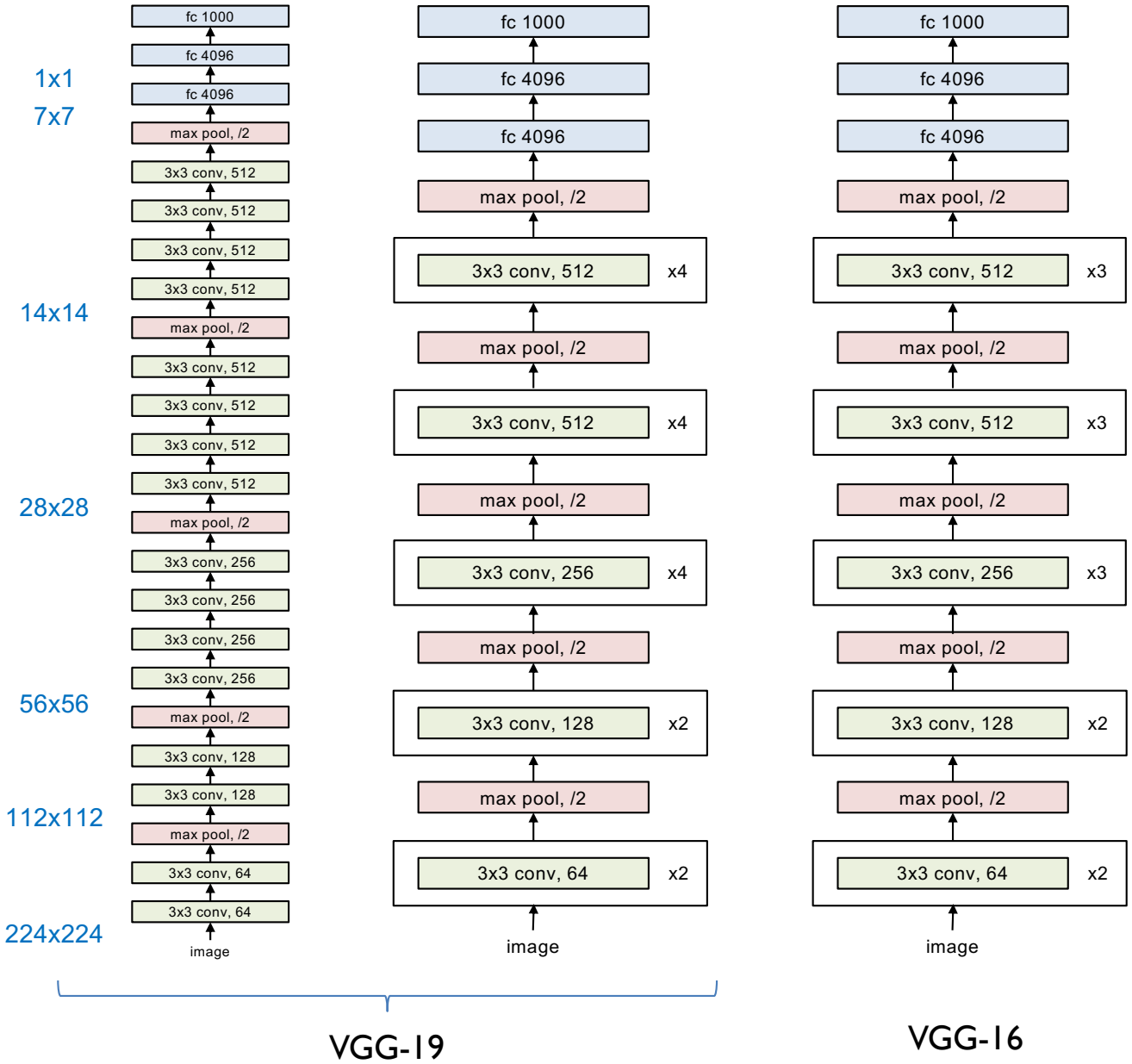


# Standard CNNs: VGG-Net

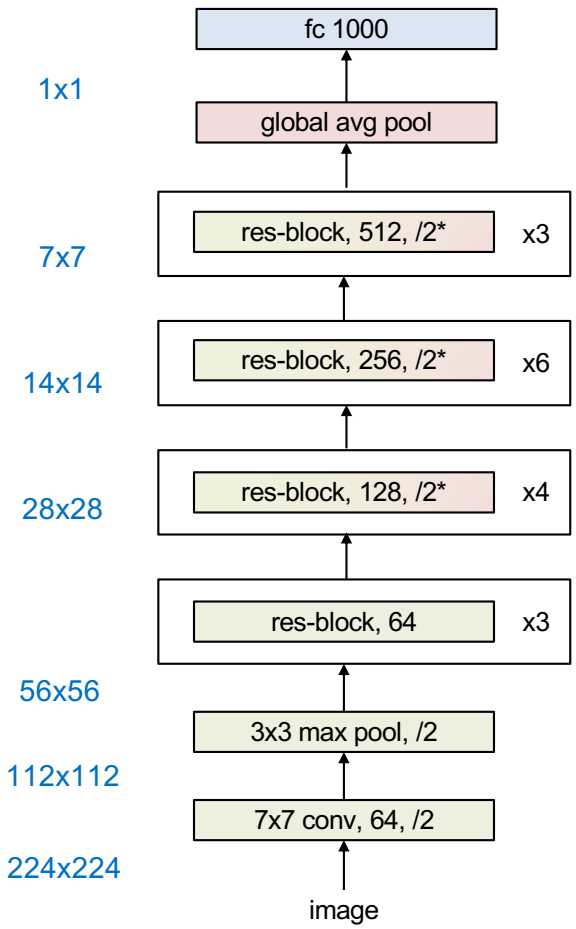
- '*Bigger and simplified* Alexnet'
- All the conv. filters are 3x3
- All conv. layers are followed by ReLU activation
- No local response normalization layer

Simonyan-Zisserman, VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, 2014

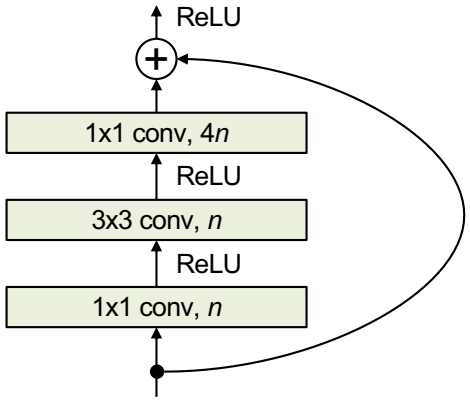


# Standard CNNs: ResNet

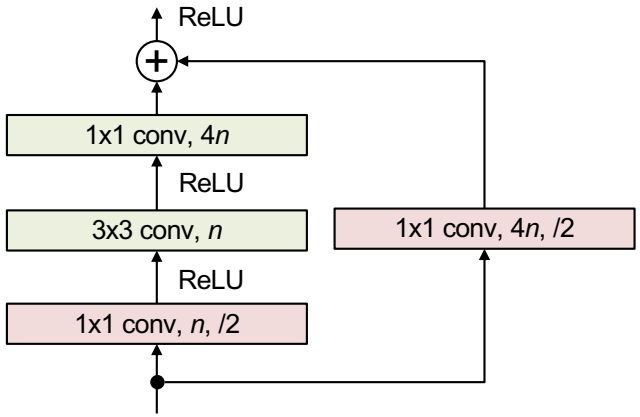
- Single FC w/ global average pooling
- Residual connection
- Block structure
- Bottleneck structure w/ 1x1 convolution
- Stride=2 convolution instead of pooling
- Extensive use of batch normalization




ResNet-50




res-block, *n*



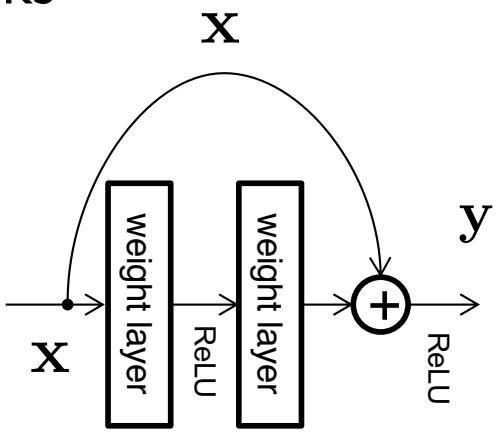

res-block, *n*, /2

Overall size is controlled by the repetition counts of the four blocks
ResNet-50: (3, 3, 5, 2)
ResNet-101: (3, 3, 22, 2)
ResNet-152: (3, 7, 35, 2)

* Only the 1st block has stride = 2

# Residual connection

- A core component of modern networks
  - Improved convergence
  - Deeper networks
  - Better performance

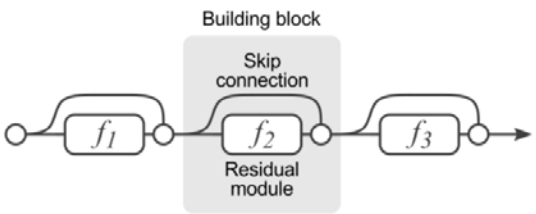



$$\mathbf{y} = F(\mathbf{x}) + \mathbf{x}$$

- Weight layers learn the *residual*:

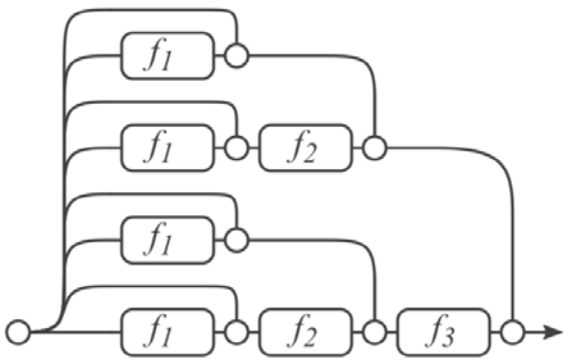
$$\mathbf{y} - \mathbf{x} = F(\mathbf{x})$$

- An interpretation of ResNet as an ensemble of networks [Veit+2016]
  - Many other explanations exist




(a) Conventional 3-block residual network

=

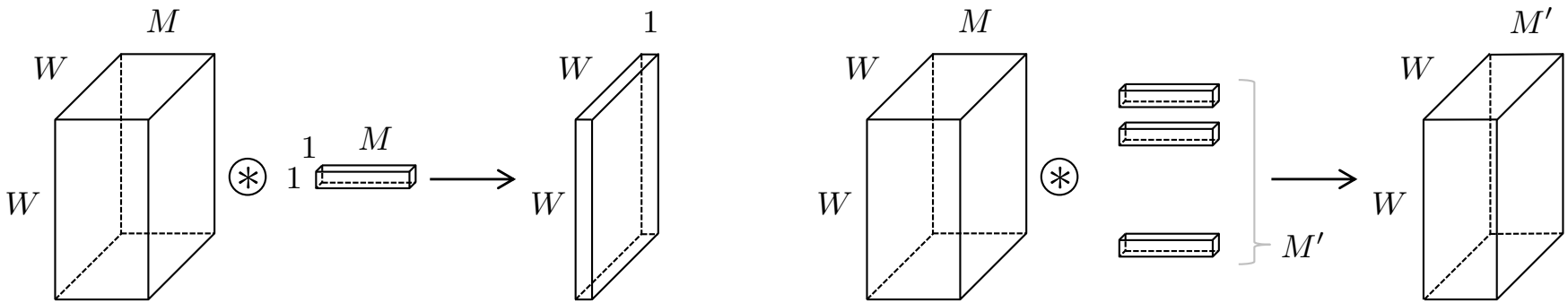



(b) Unraveled view of (a)

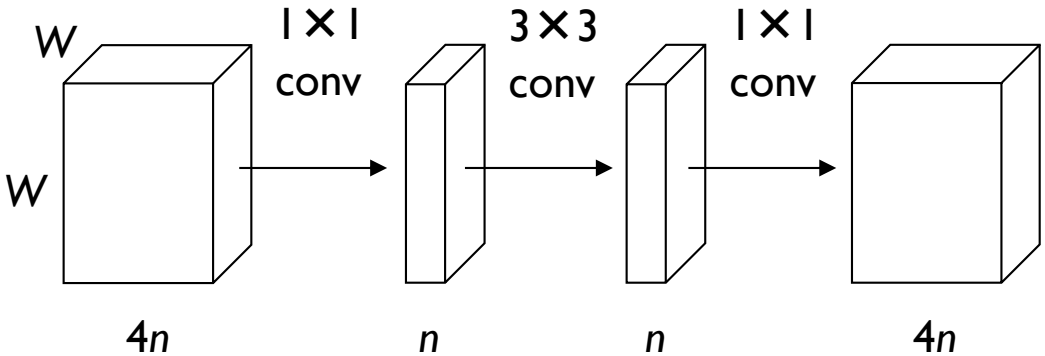
Veit-Wilber-Belongie, Residual Networks Behave Like Ensembles of Relatively Shallow Networks, 2016

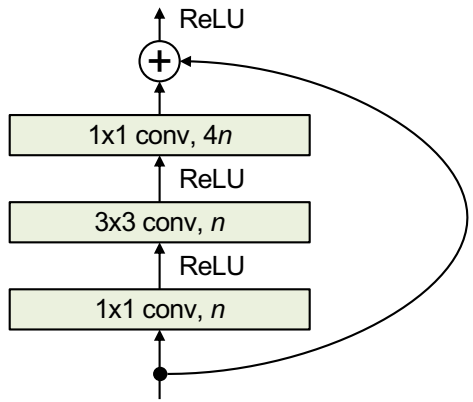
# 1x1 convolution

- 1x1 convolution = convolution w/ 1x1 filters
  - Aggregation only over channels; not over space
  - Can convert the number of channels between input and output



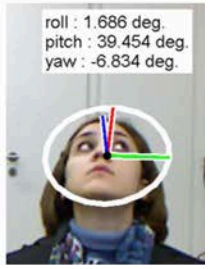

- Used in the bottleneck structure of resblock
  - Have many channels without increasing computational cost



# CNNs w/ various output types

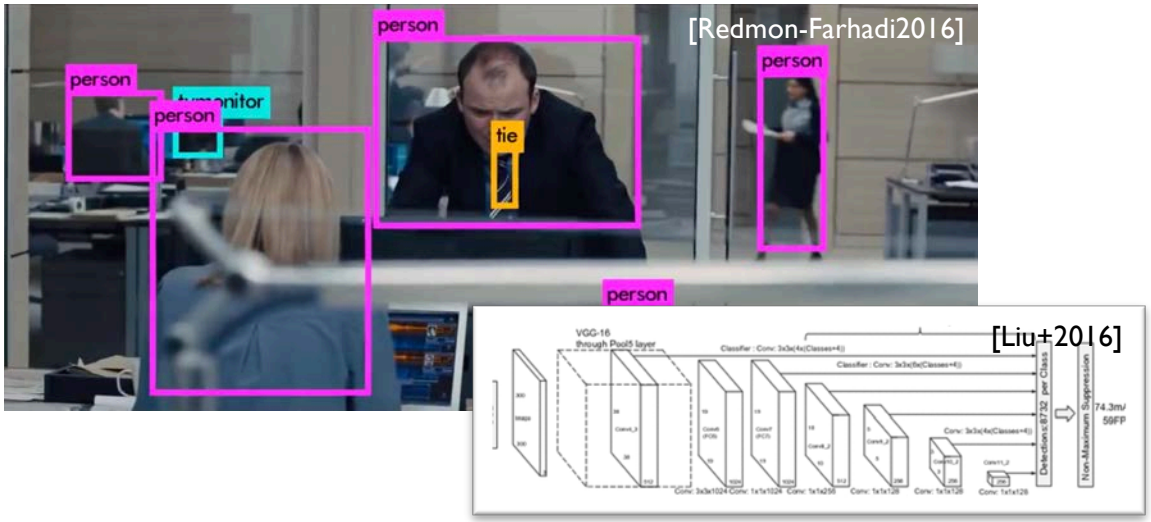
- Input = an image / Output = a vector
  - Object classification, Face orientation estimation



→ CNN → 'lion'

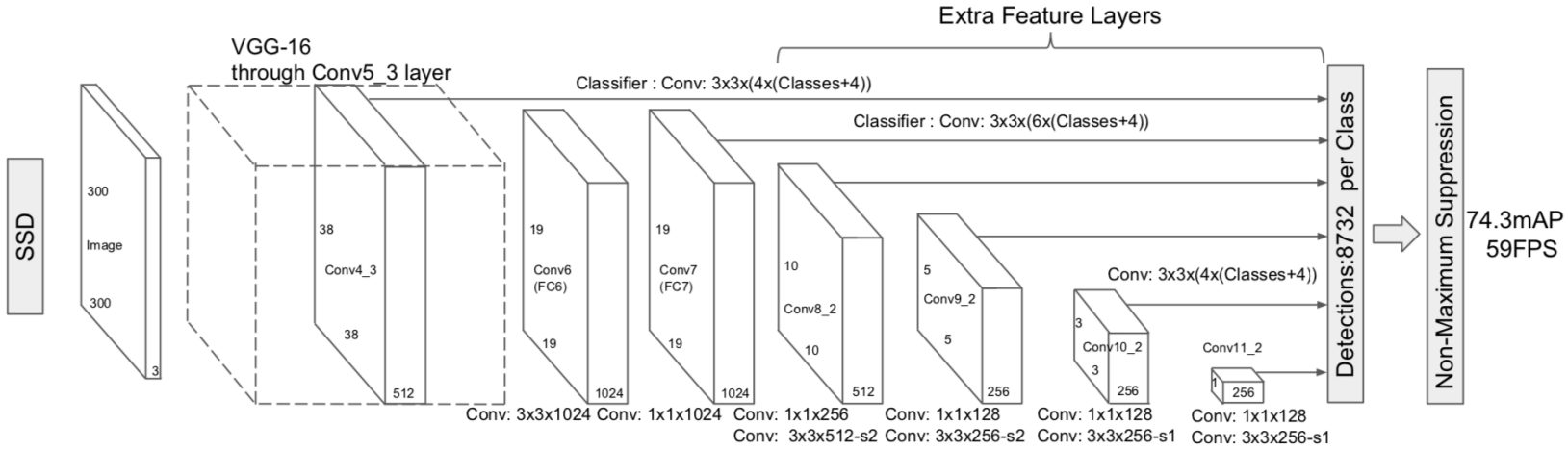
Image → CNN → roll pitch yaw

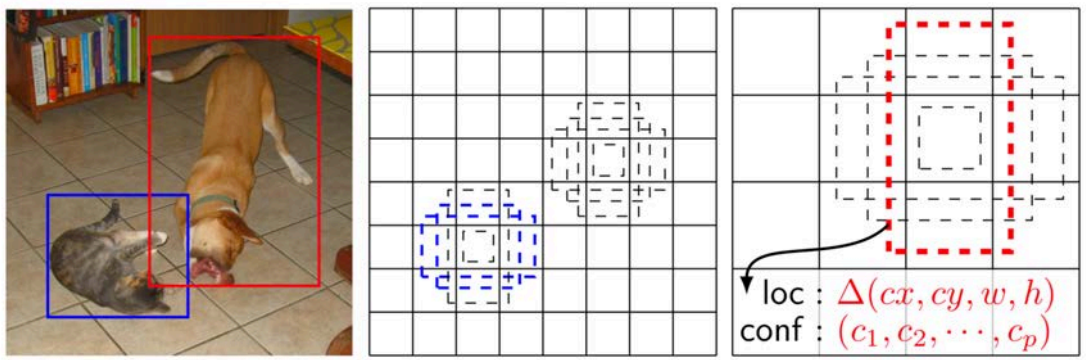
- Input = an image / Output = others
  - Object detection



# Object detection

- SSD predicts the bounding box (BB) for each object in an image
  - Each unit (cell) creates several BBs; a BB is represented by location, size, and confidence
  - SSD is trained so that the true BB is predicted for each object



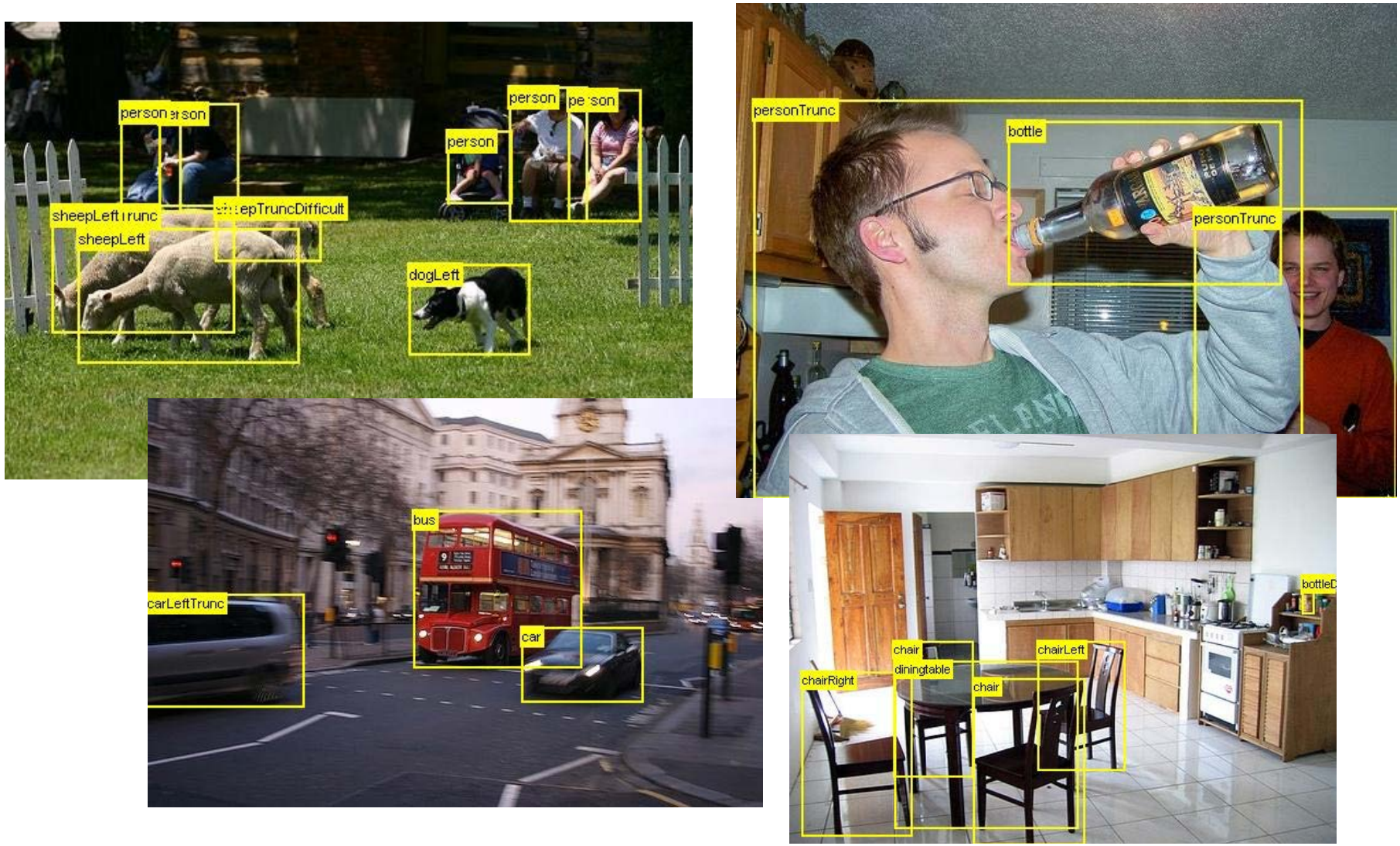




(a) Image with GT boxes (b) $8 \times 8$ feature map (c) $4 \times 4$ feature map

Liu+, SSD: Single Shot MultiBox Detector, 2015

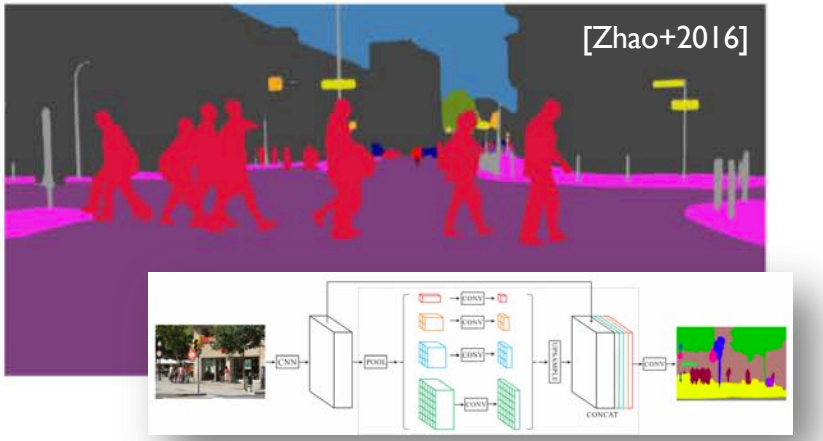
# Object detection: training data

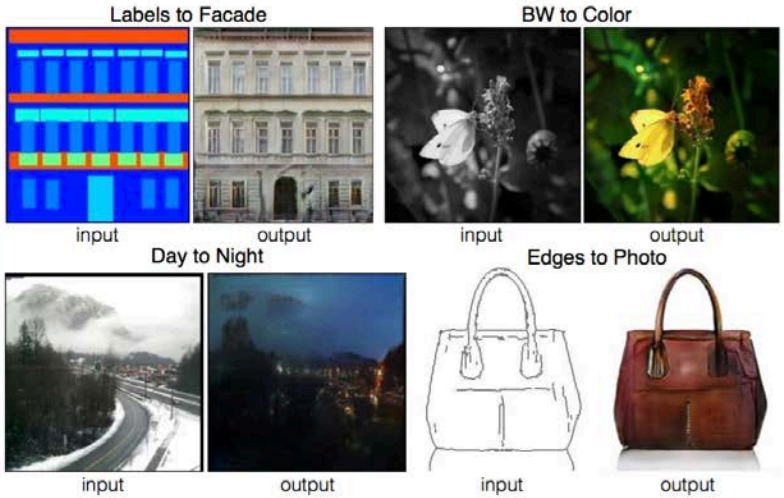
- Example dataset: PASCAL VOC(Visual Object Classes)
  - http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

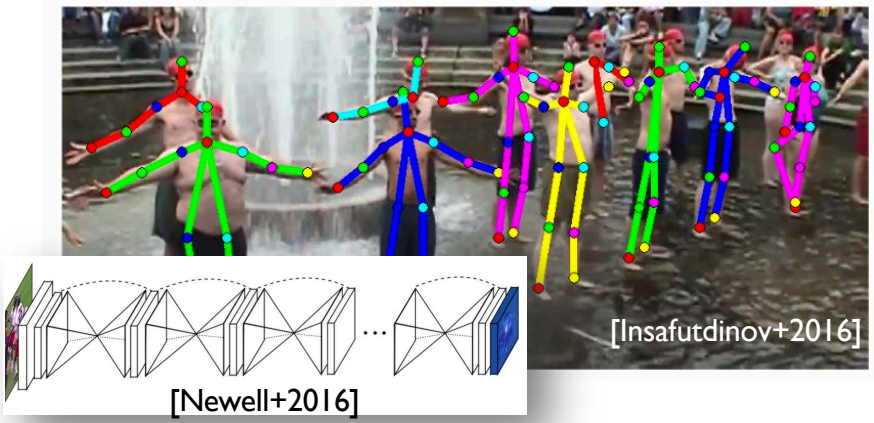


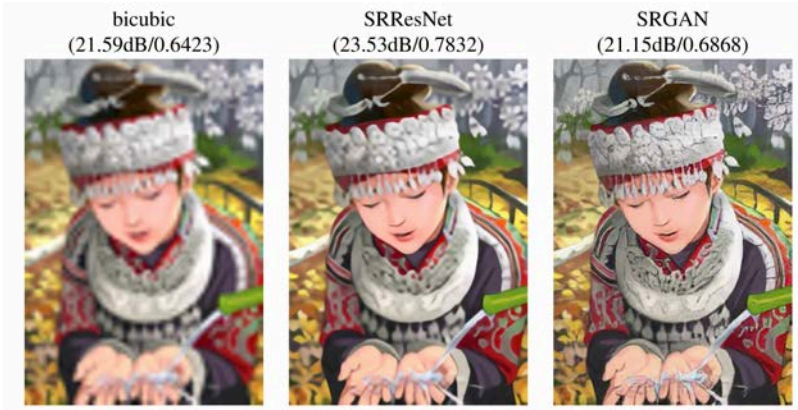
# CNNs w/ various output types

- Input = an image / Output = an image or a matrix/tensor
  - Semantic segmentation, Image-to-image translation / Super-resolution, Human pose estimation



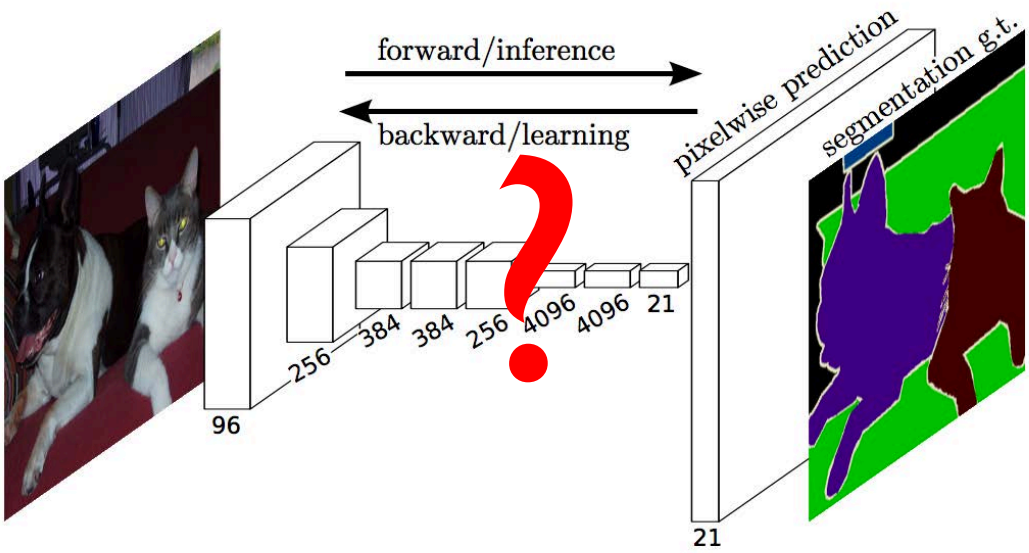




[Ledig+2016]

# CNNs that output a 2D map

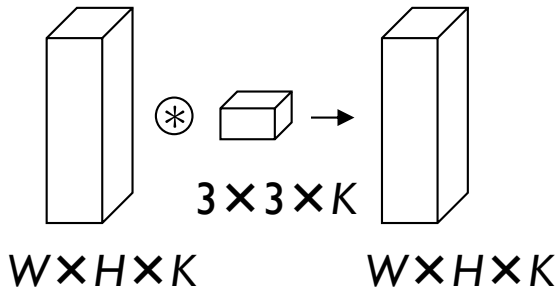
- In some tasks, CNNs need to have a 2D output
  - e.g., semantic segmentation
- How can we get 2D output instead of 1D output (class scores)?



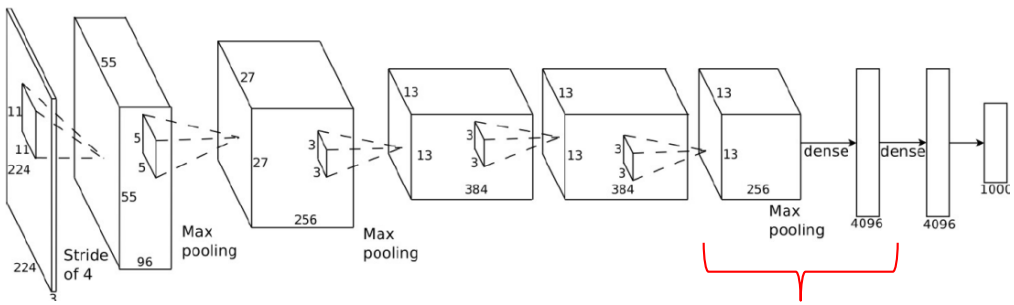

Long+, Fully Convolutional Networks for Semantic Segmentation, 2015

# Size of inputs to CNNs

- Conv. (and pooling) layers can handle inputs of arbitrary size
  - Output size is automatically determined by the input size



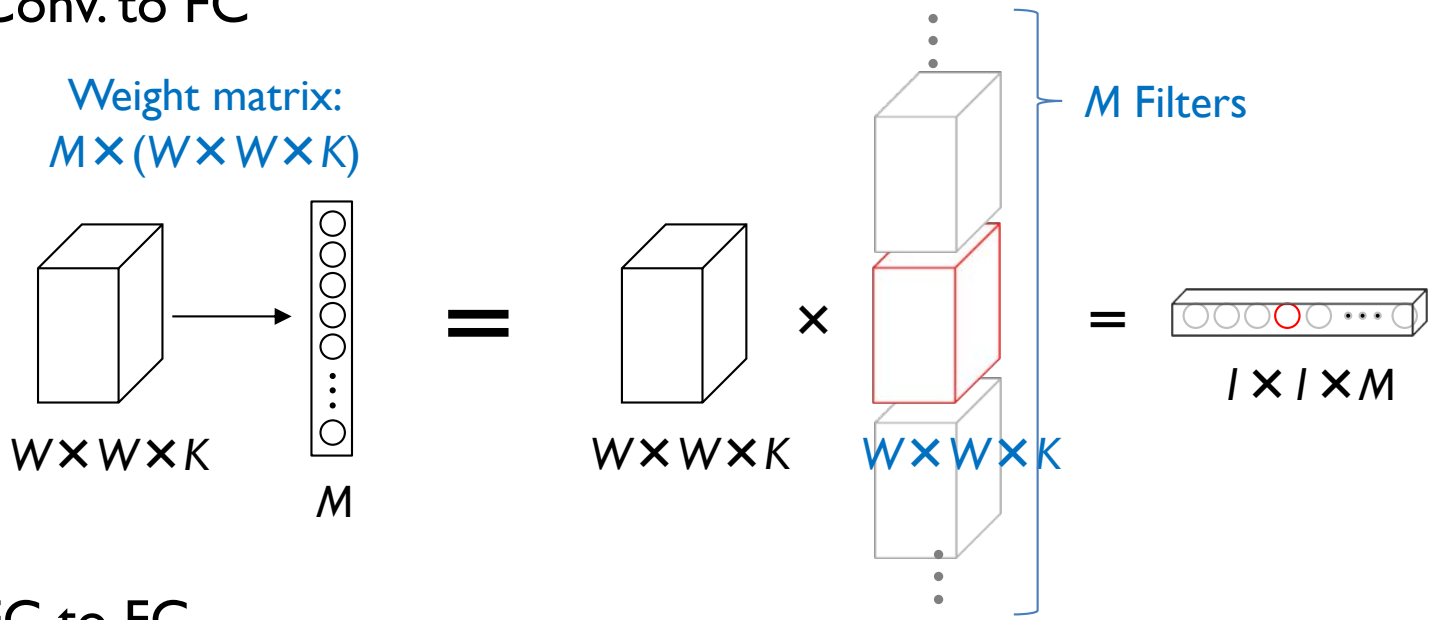

- Alexnet/VGG-net can handle only fixed-size inputs
  - Input size cannot be changed because of the last fc layers
  - This is not the case with ResNets, which has a global average pooling layer connecting the last conv. layer to the final fc layer(s)



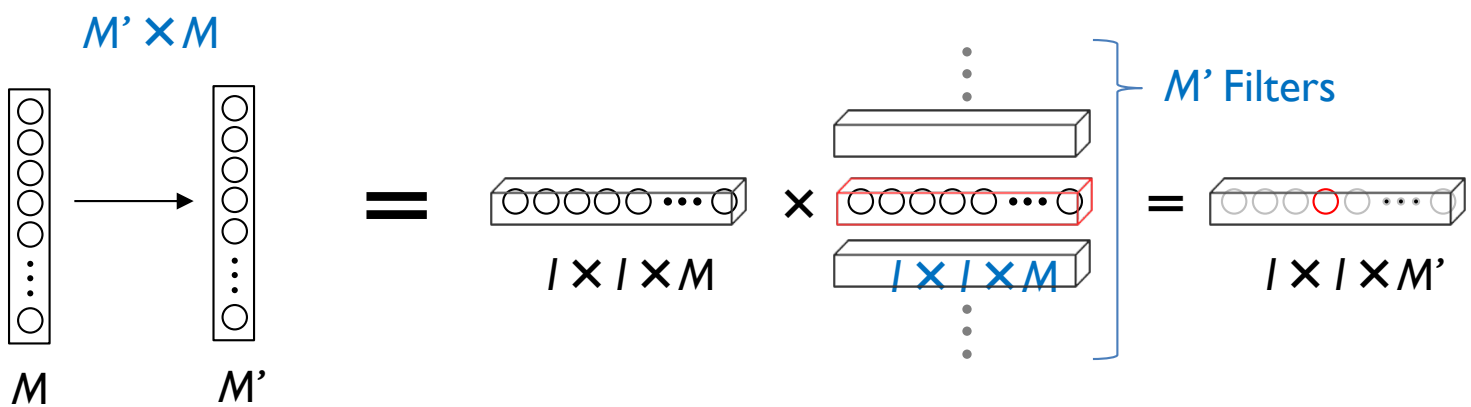

7x7x256 → 4096

# FC layers = conv. layers

- Fully-connected layers is reinterpreted as conv. layers of size 1x1
  - Conv. to FC

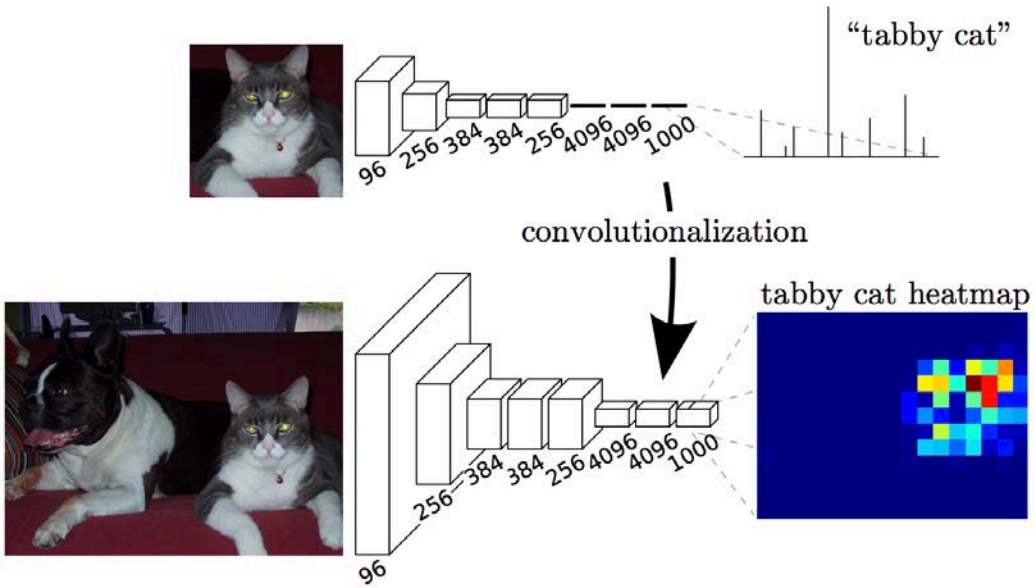



  - FC to FC



# Fully convolutional networks (FCNs)

- FC layers can be interpreted as a conv. layer of size 1x1
- Classification CNNs can be interpreted as CNNs w/ all conv. Layers




Long+, Fully Convolutional Networks for Semantic Segmentation, 2015

- Ratio of output/input size is determined by downsampling factor
  - In ImageNet CNNs, the output will be 1/32(=1/$2^5$) of the input
  - Too coarse; how can we increase resolution?

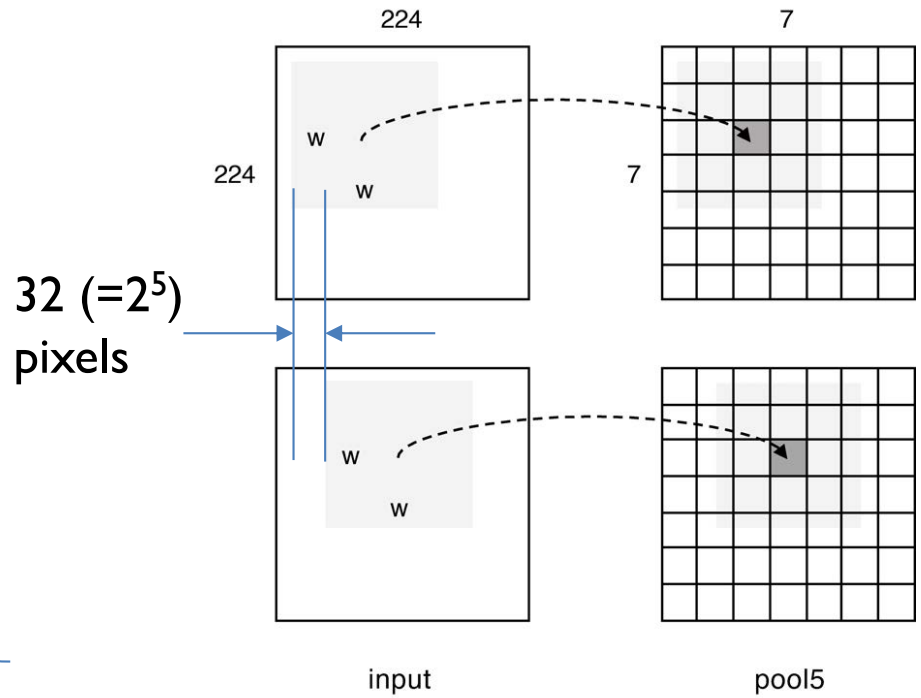
# How to get high-resolution outputs: four ways

- Upsampling
  - w/ or w/o interpolation
  - Plus convolution
  - Note: many different terminology (*subpixel conv., transposed conv., deconvolution*) for an identical concept
- Dilated convolution
- Shift & stitch
  - Very inefficient and not popular

---

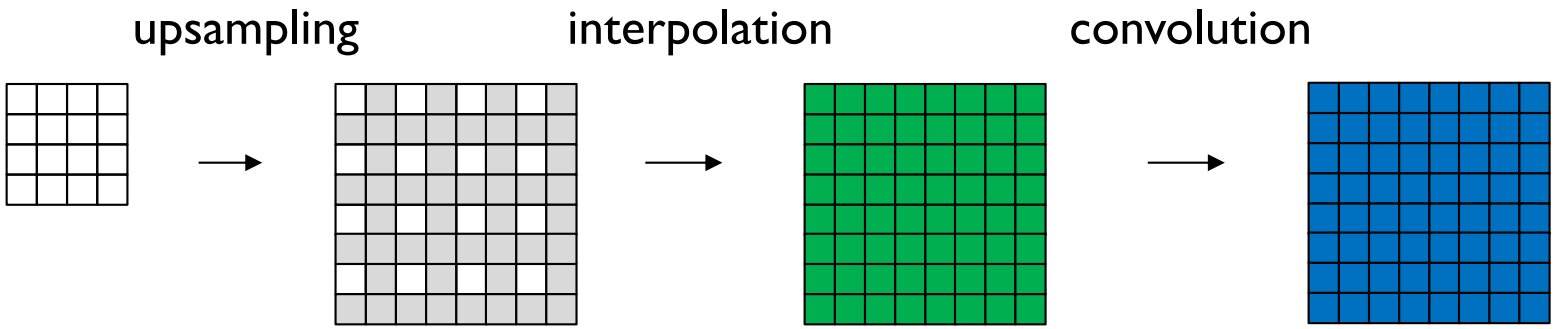
- Skip-connection
  - To recover spatial info that is lost due to downsampling

Shift the input by *i* (=0,…, 31) pixels and stitch their results → this yields output of the same size as the input

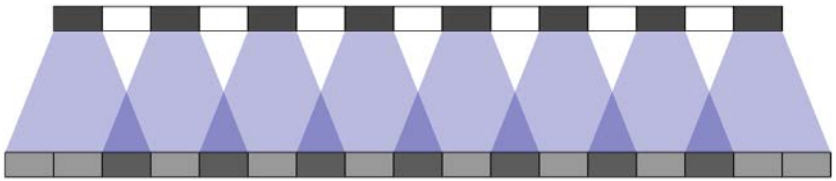


# Up-sampling

- Preferred method: upsampling w/ interpolation + convolution




- Artifacts will emerge if interpolation is not applied and the filter size is not mulitples of the upsampling factor



Stride = 2
Filter size = 3

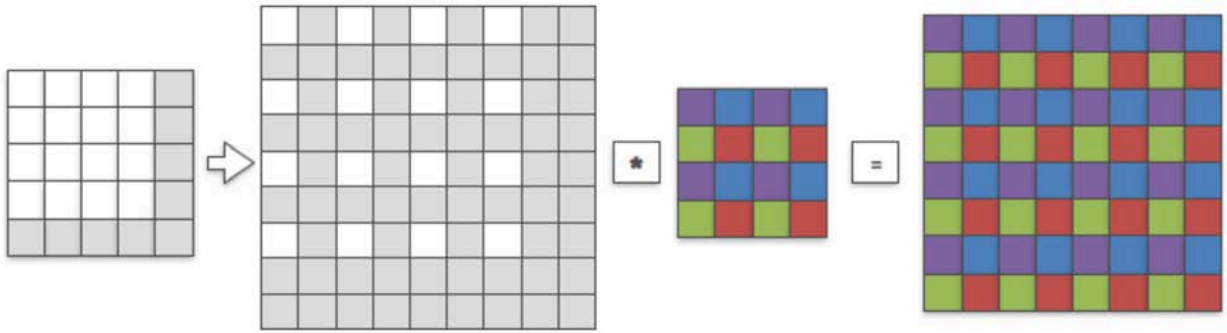
https://distill.pub/2016/deconv-checkerboard/

| | | |
|---|---|---|
| a | $\frac{a+b}{2}$ | b |
| $\frac{a+c}{2}$ | $\frac{a+b+c+d}{4}$ | $\frac{b+d}{2}$ |
| c | $\frac{c+d}{2}$ | d |

bilinear interpolation

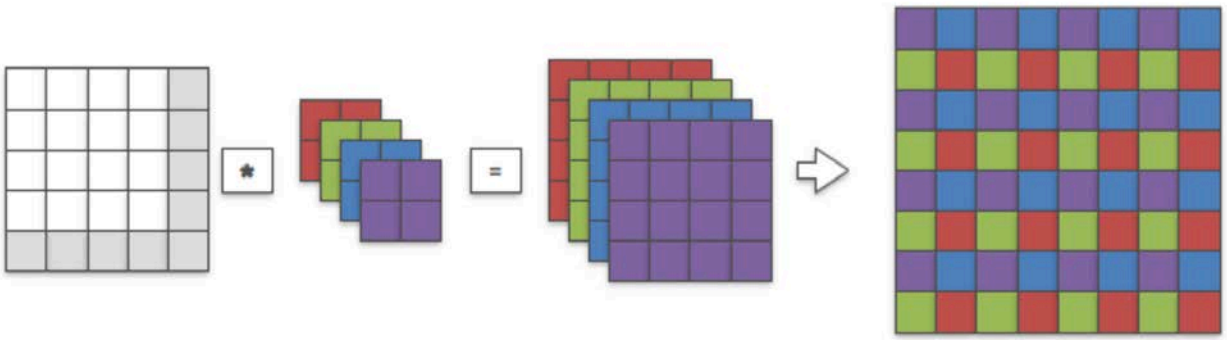
# Subpixel convolution

- Interpolation can be omitted if the filter size is multiples of the upsampling factor
  - Artifacts won't emerge



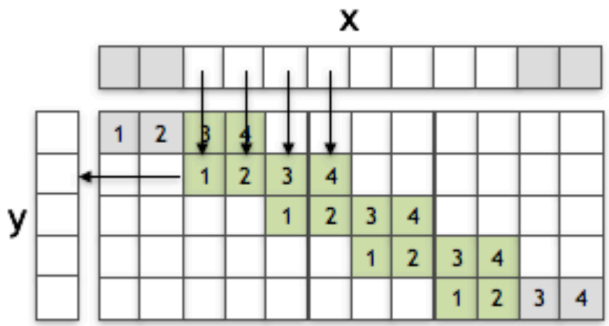
  - Computationally efficient method: *pixel shuffler*
    - Apply a multi-channel filter and reorganize the result into a single channel

Shi+, RealTime Single Image and Video Super-Resolution Using an Efficient SubPixel Convolutional Neural Network, 2016



# Transposed convolution

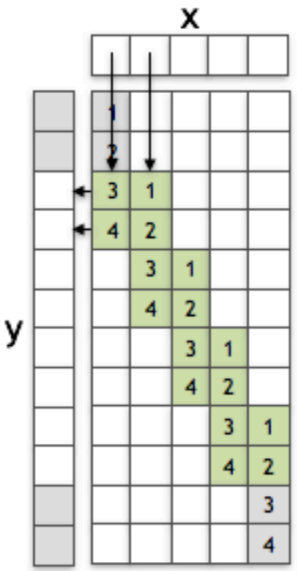
- Subpixel convolution is equivalent to *transposed convolution*




Standard convolution w stride=2

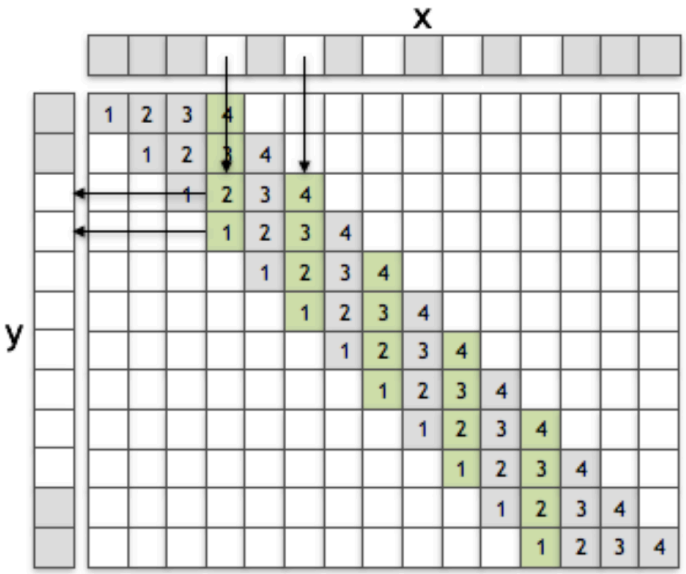
Summary

- If you want to use filter of arbitrary size, you must choose upsampling w/ interpolation + convolution
- If the size of your filter is multiples of your up-sampling factor, you can use transposed convolution = subpixel convolution




Transposed convolution w stride=2

=




x2 Upsampling w/o interpolation + convolution

What is worse, transposed convolution is sometimes referred to as *deconvolution;* but this is wrong terminology and you shouldn't use it

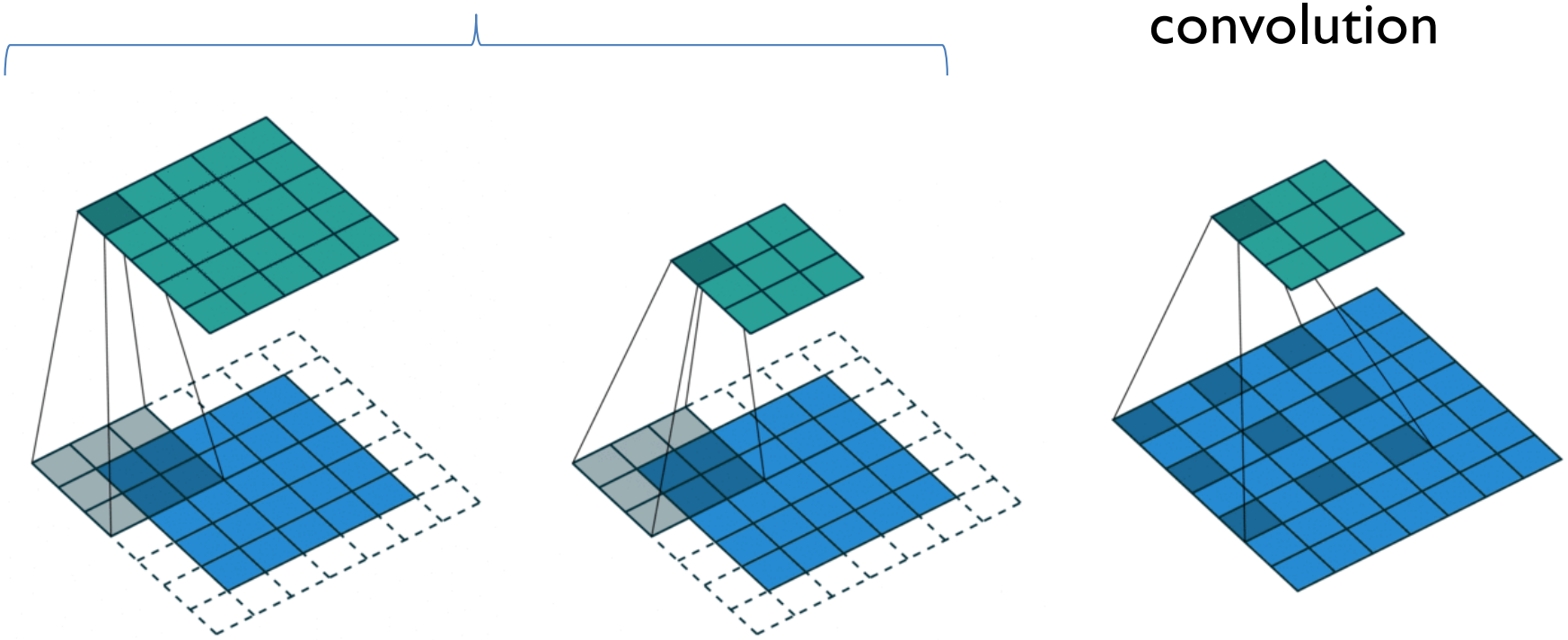
Shi+, arXiv1609.07009

# Dilated convolution (or atrous conv.)

- A filter is expanded w/ rate = *r* and applied to inputs

standard convolution

dilated (or atrous) convolution



stride = 1 w/ padding

stride = 2 w/ padding

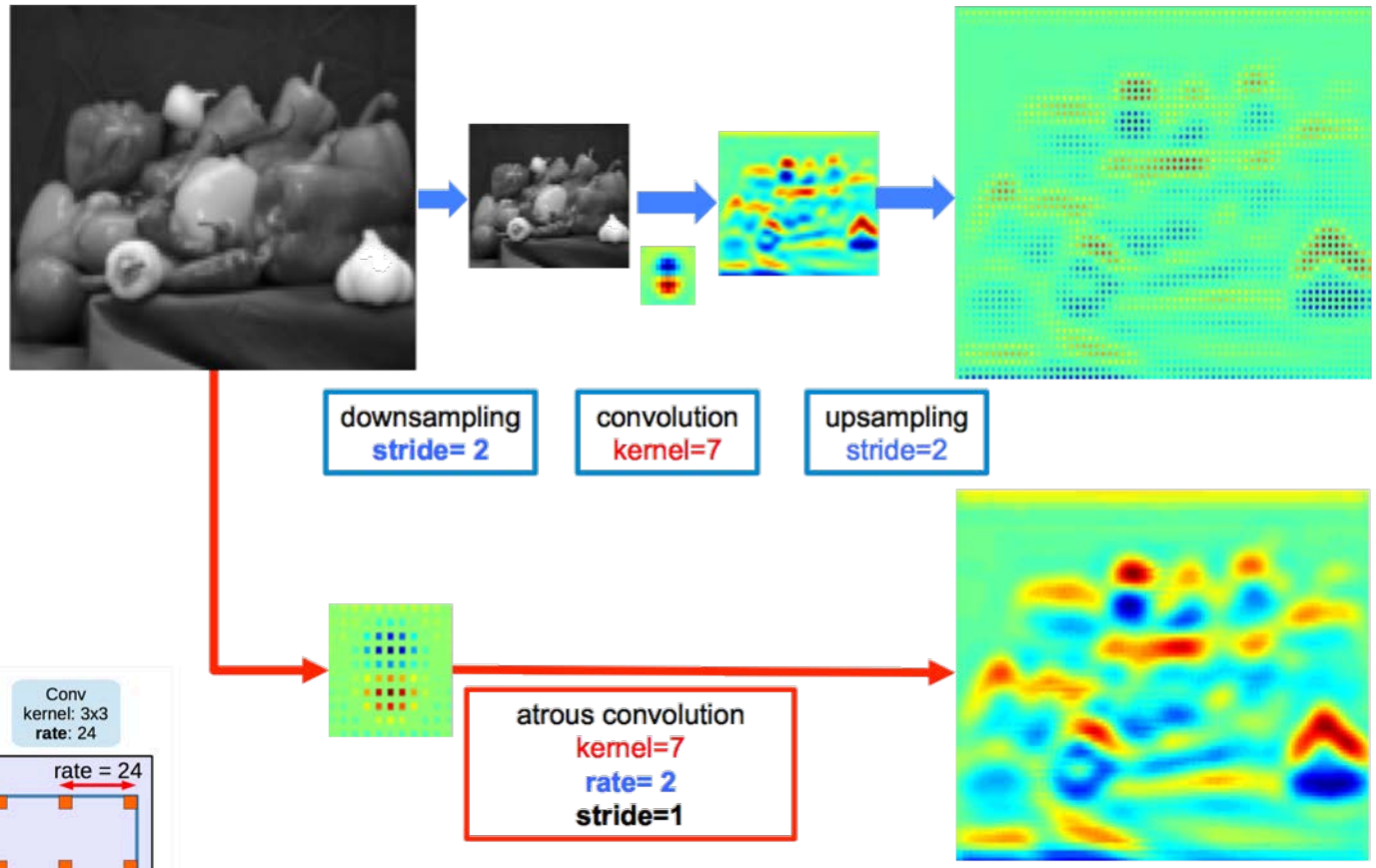
stride = 1 w/o padding

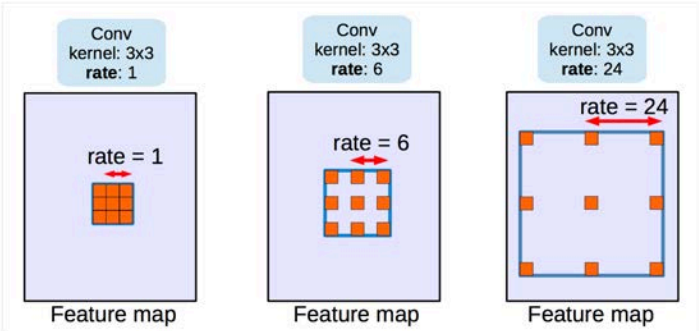
Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning (BibTeX)

# Dilated convolution (or atrous conv.)

Chen+, DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully connected CRFs, arXiv16

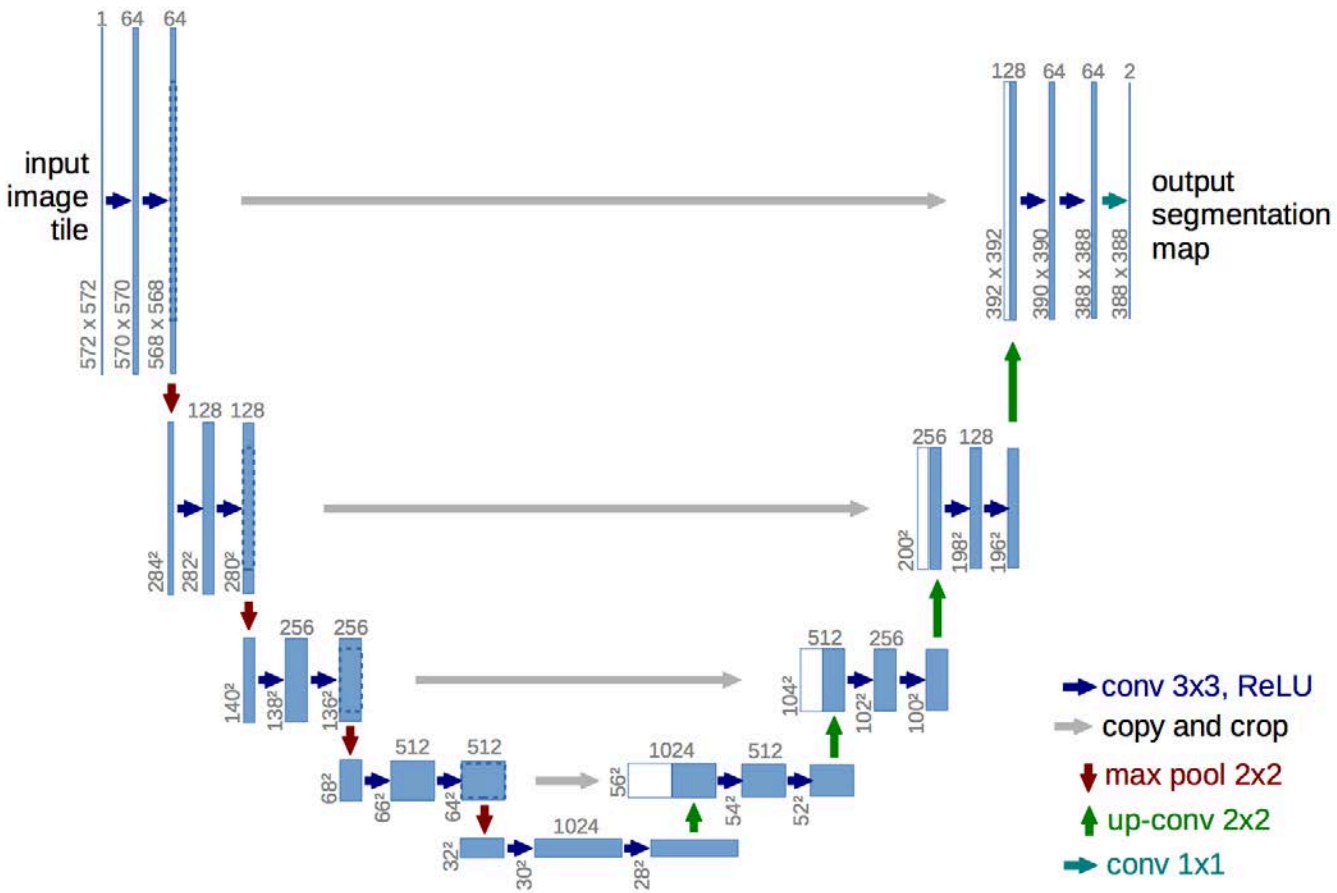
- Dilated conv. w/ expansion rate *r* is equivalent to
  *r* : 1 downsampling + convolution + 1 : *r* upsampling

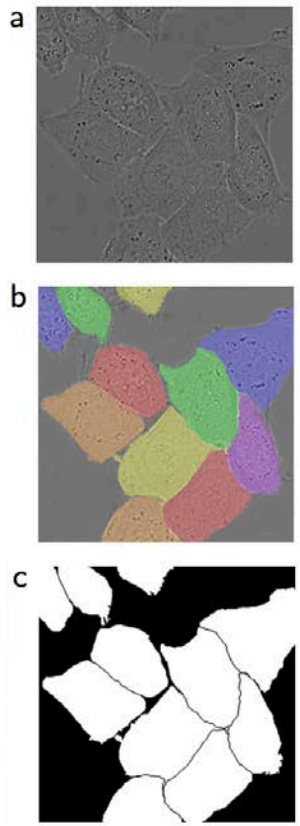


# Skip-connection: U-Net

- Upsampling + conv. does not recover lost resolution
- Transfer high-resolution feature from lower layers
  - Concatenate it in the channel dimension

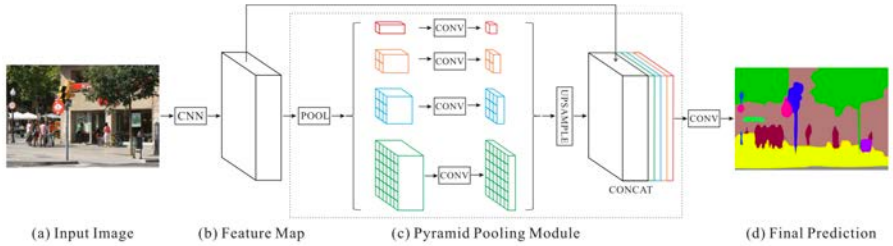


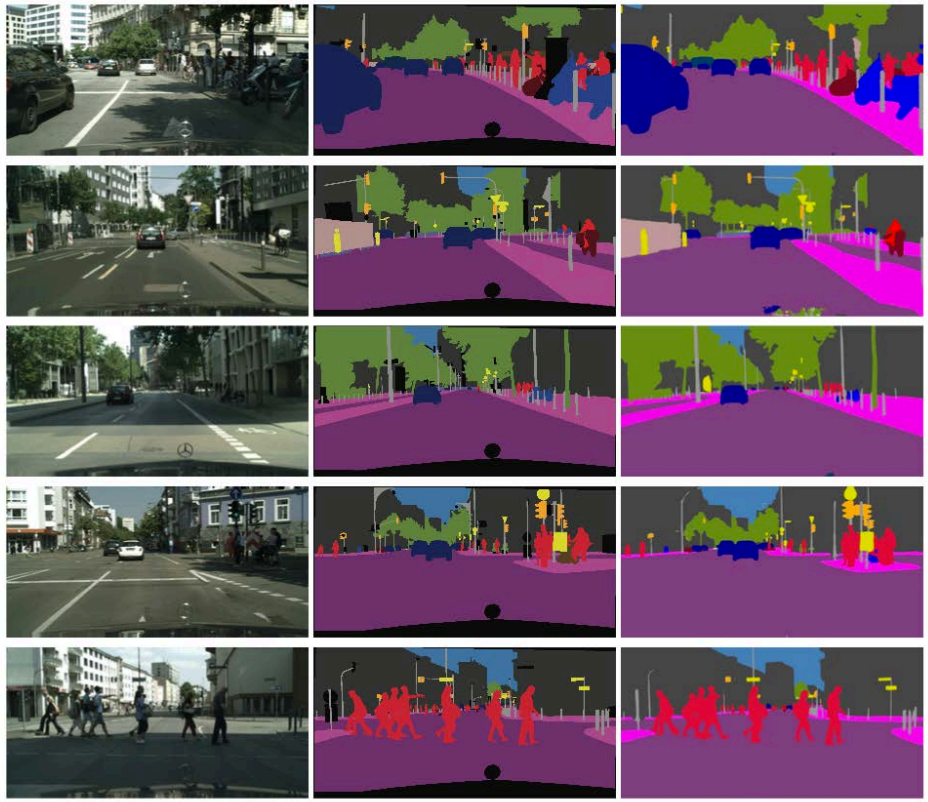




Ronneberger+, U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015

# Popular nets for semantic segmentation

## PSPNet

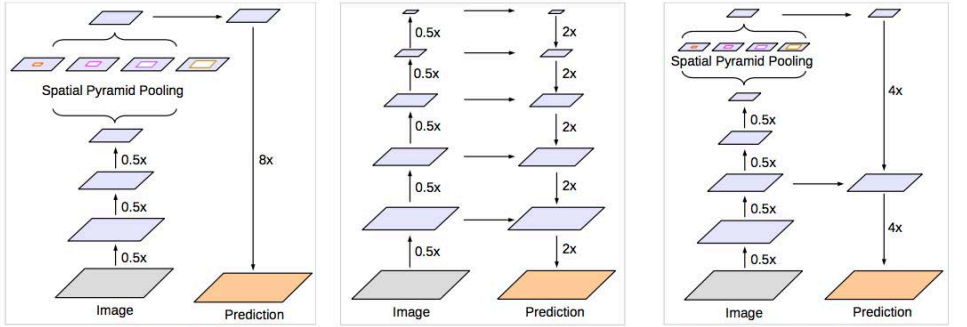


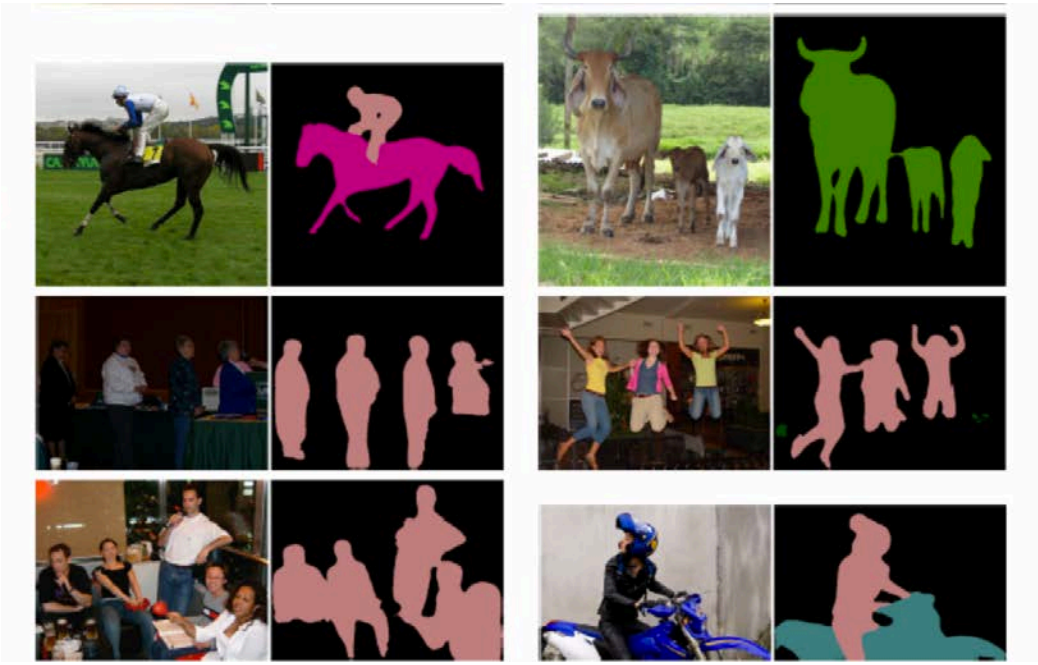



(a) Image (b) Ground Truth (c) PSPNet

Zhao+, Pyramid Scene Parsing Network, CVPR2017

## Deeplabv3+

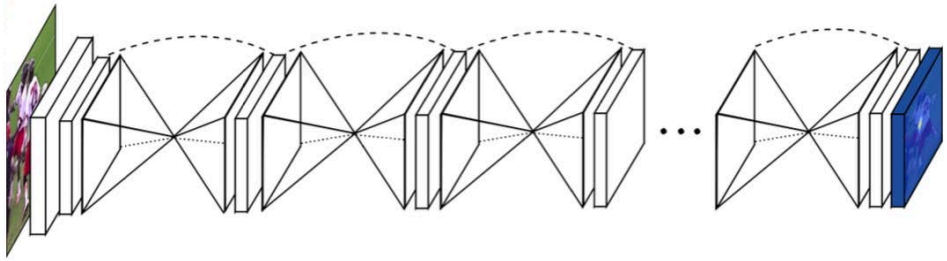


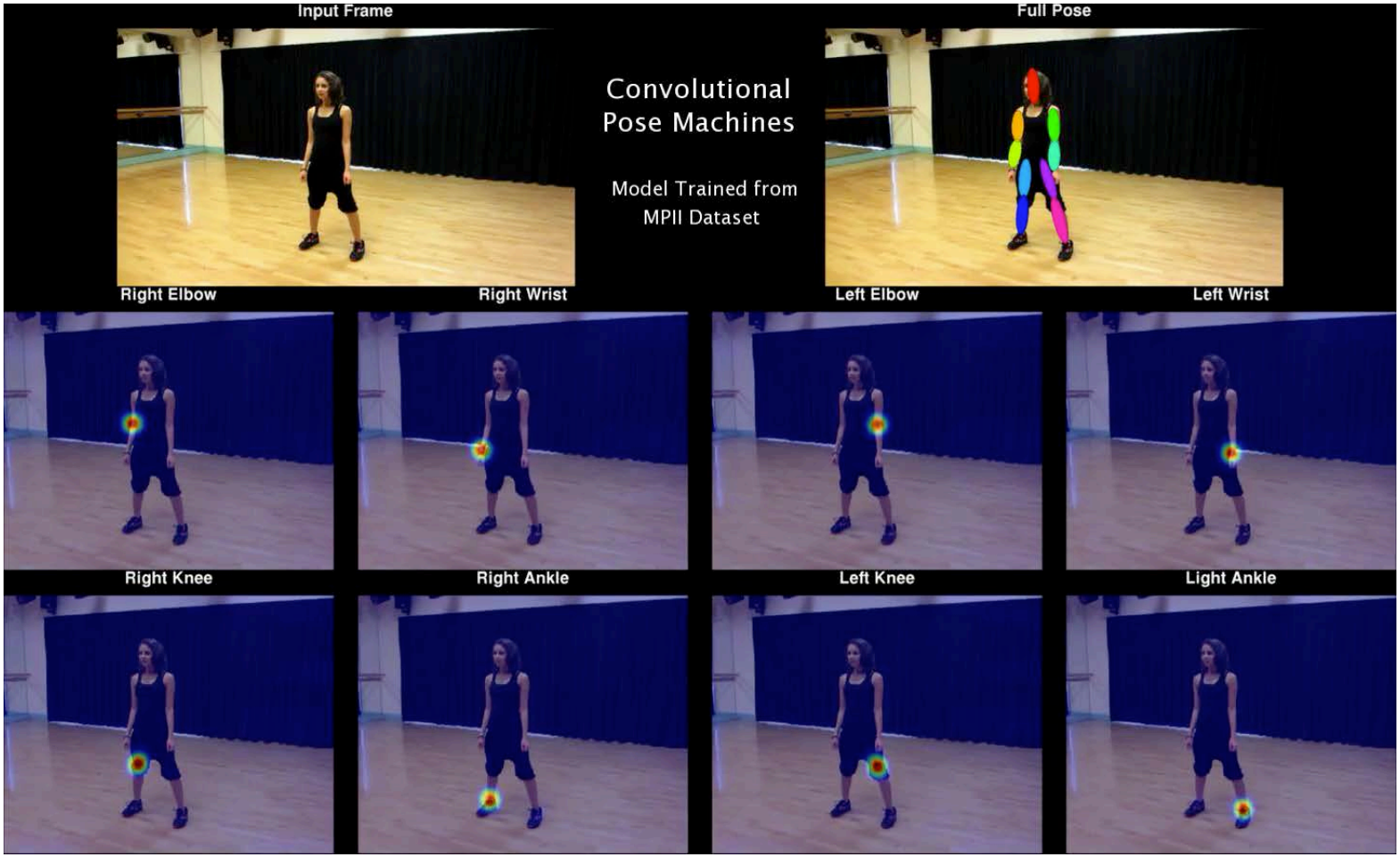



Chen+ (Google), Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation, arXiv2018

# Human pose estimation

Stacked Hourglass Networks
Newell+@ECCV2016

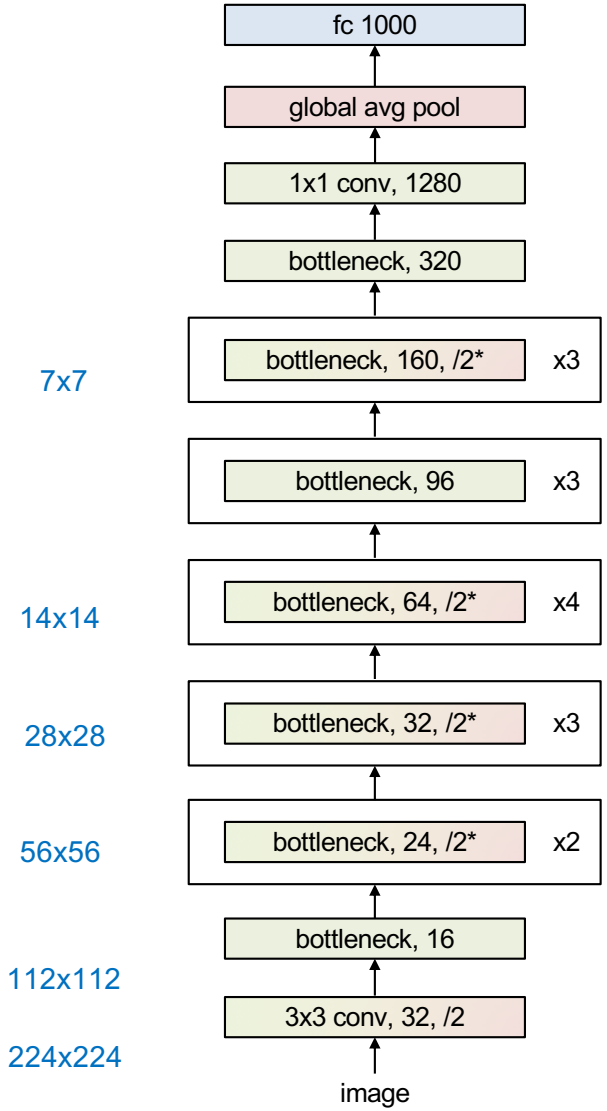




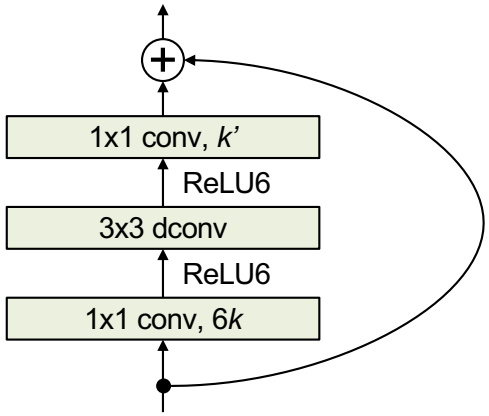

Shih-En Wei, Varun Ramakrishna, Takeo Kanade, Yaser Sheikh, Convolutional Pose Machines, 2016

# Standard CNNs: MobileNet(v2)

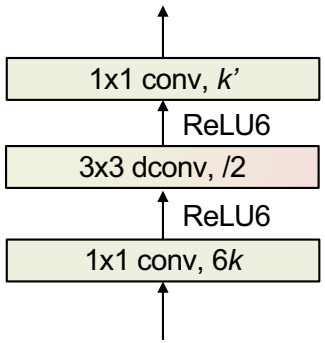
- Target: smartphones etc.
- Pursue computational efficiency while preserving prediction accuracy
- Depth-wise convolution




MobileNetV2




bottleneck, *k'*




bottleneck, *k'*, /2

* Only the 1st block has stride = 2

Sandler+, MobileNetV2: Inverted Residuals and Linear Bottlenecks, arXiv1801

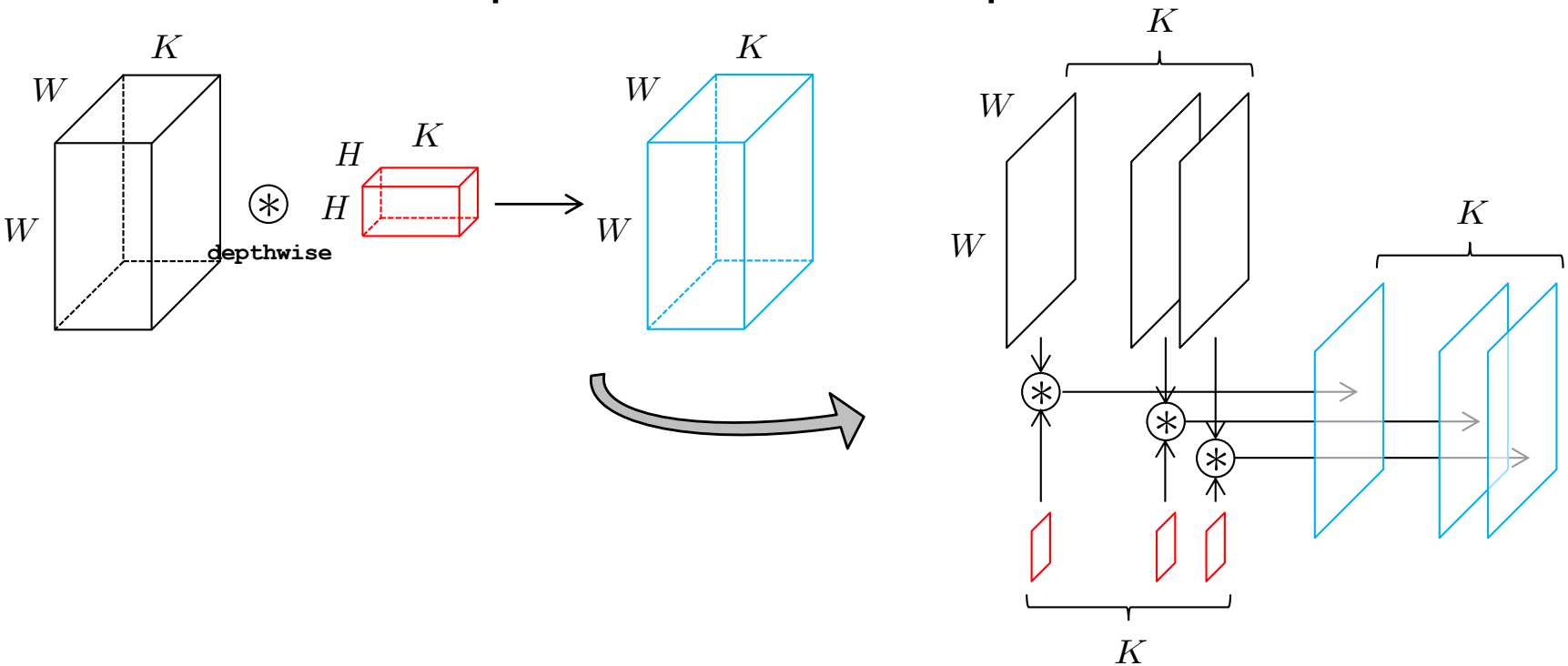
# Depthwise convolution

- Performs convolution at each channel independently

$$u_{ijm} = \sum_p \sum_q z_{i+p,j+q,k} h_{pqm} \quad \Leftrightarrow \quad u_{ijm} = \sum_k \sum_p \sum_q z_{i+p,j+q,k} h_{pqkm}$$

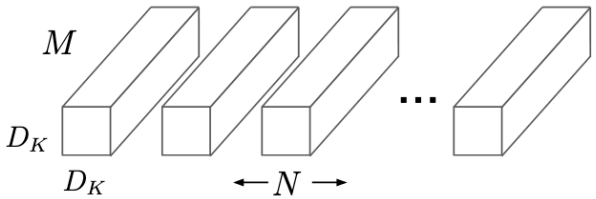
Depthwise conv. Standard conv.

  – A depthwise conv. layer usually has a single filter; the same number of output channels as the input

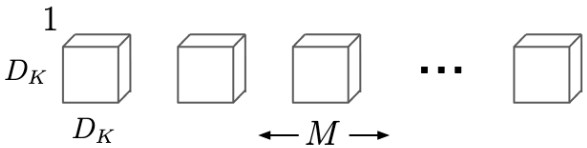


# Separable convolution

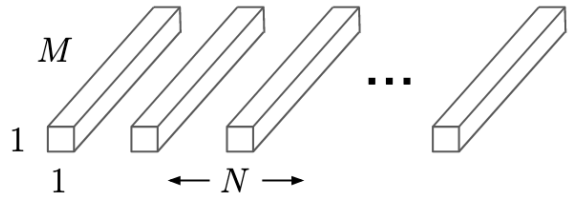
- Standard conv. is replaced by depth-wise conv. followed by 1x1 conv.
  - Mathematically not equivalent but works *comparably* well
  - Computationally faster ~ 1/(WxH of filters)



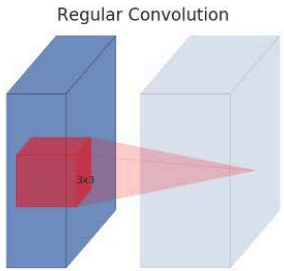

(a) Standard Convolution Filters

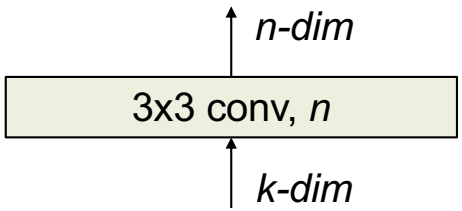



(b) Depthwise Convolutional Filters

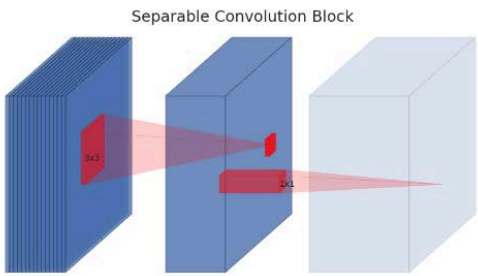


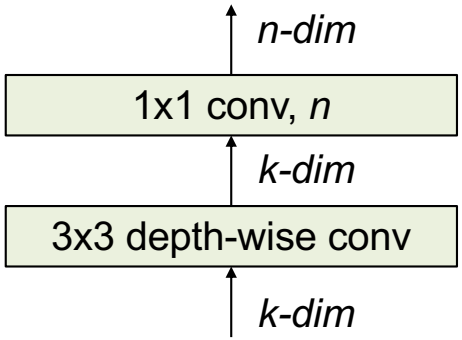

(c) $1 \times 1$ Convolutional Filters called Pointwise Convolution in the text of Depthwise Separable Convolution




$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$



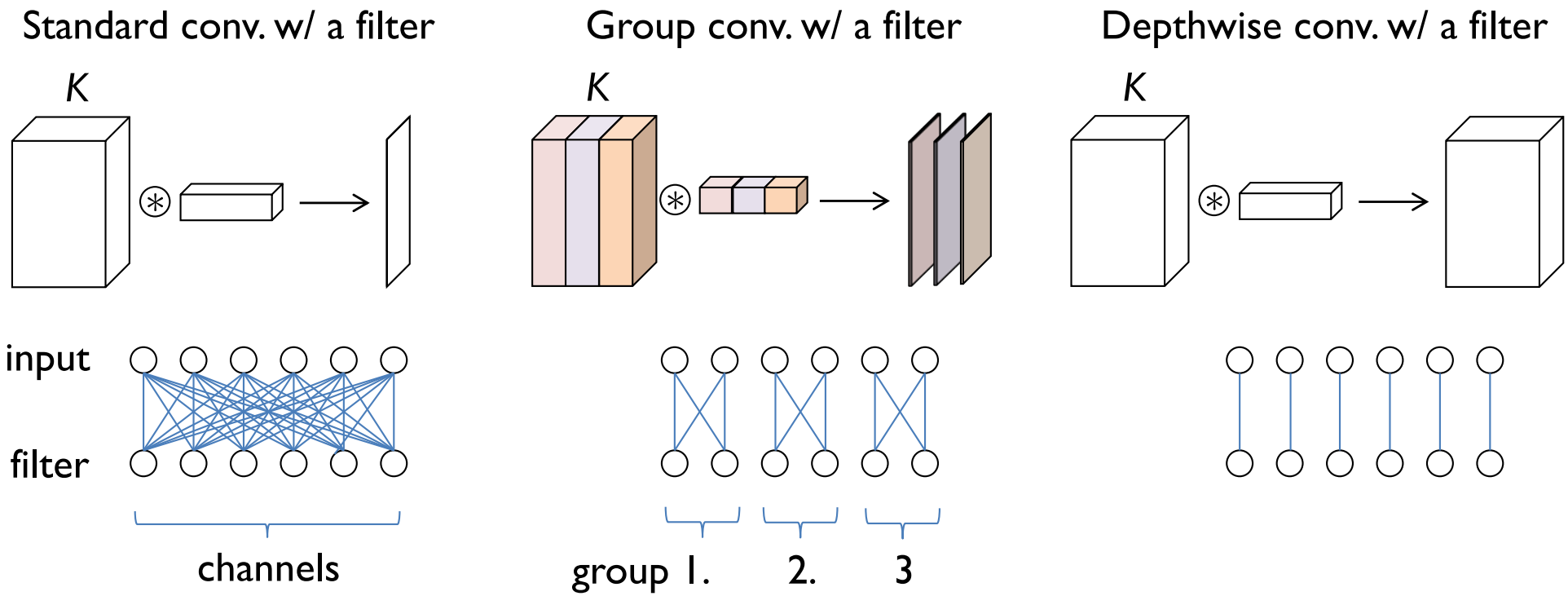




$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

Howard+(Google inc.), MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv1704

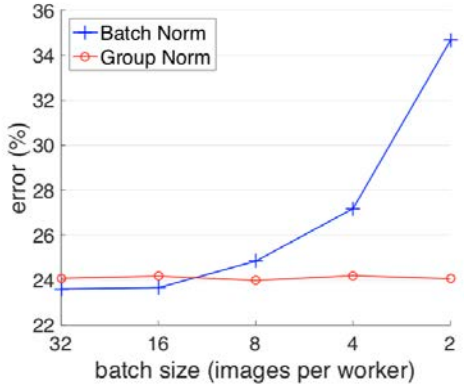
# Group convolution

- Input channels are split into groups
- Channels within each group are aggregated
  - No interaction between channels belonging to different groups
- Regarded as an intermediate between the standard conv. and the depthwise conv.



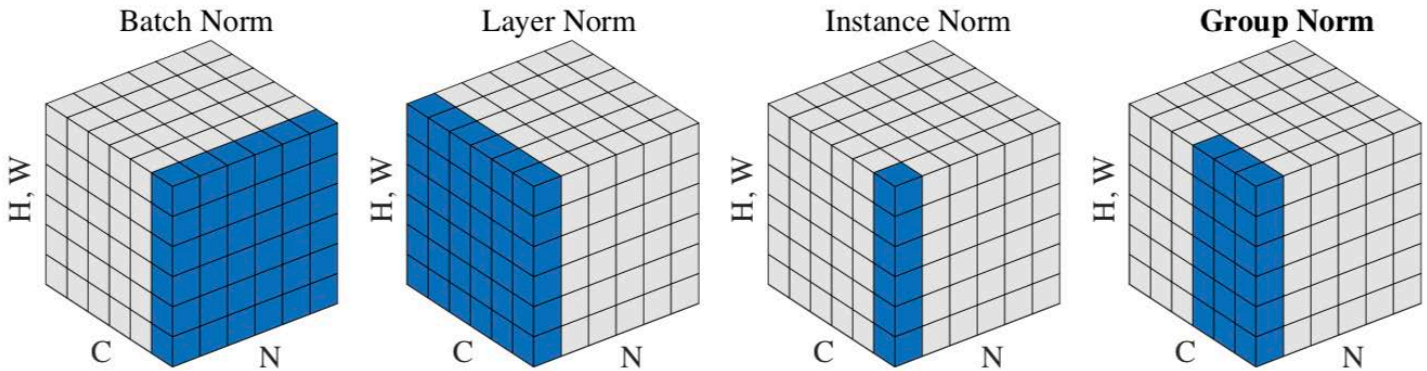
# Layer/Instance/Group Normalization

- Variants of Batch Normalization
- Issues with BN
  - BN does not work with small minibatches
  - BN is not fit for RNNs (inputs are sequences)




[Wu-He, ECCV2018]

- Layer norm/Instance norm/Group norm
  - They differ in the set for which mean and stddev are computed




[Wu-He, ECCV2018]