The Great Ideas in CNNs

Recitation 5

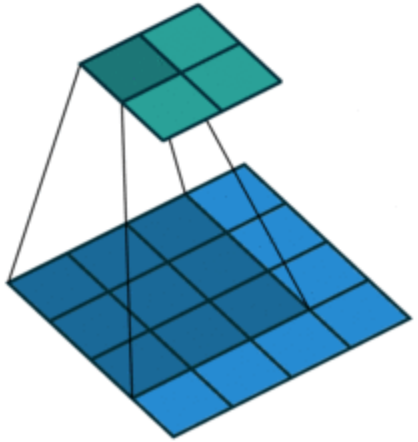
Credits to Kangrui for these amazing slides

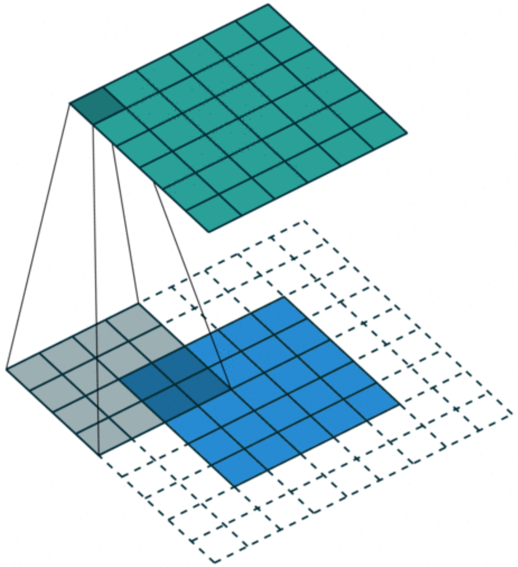
Content

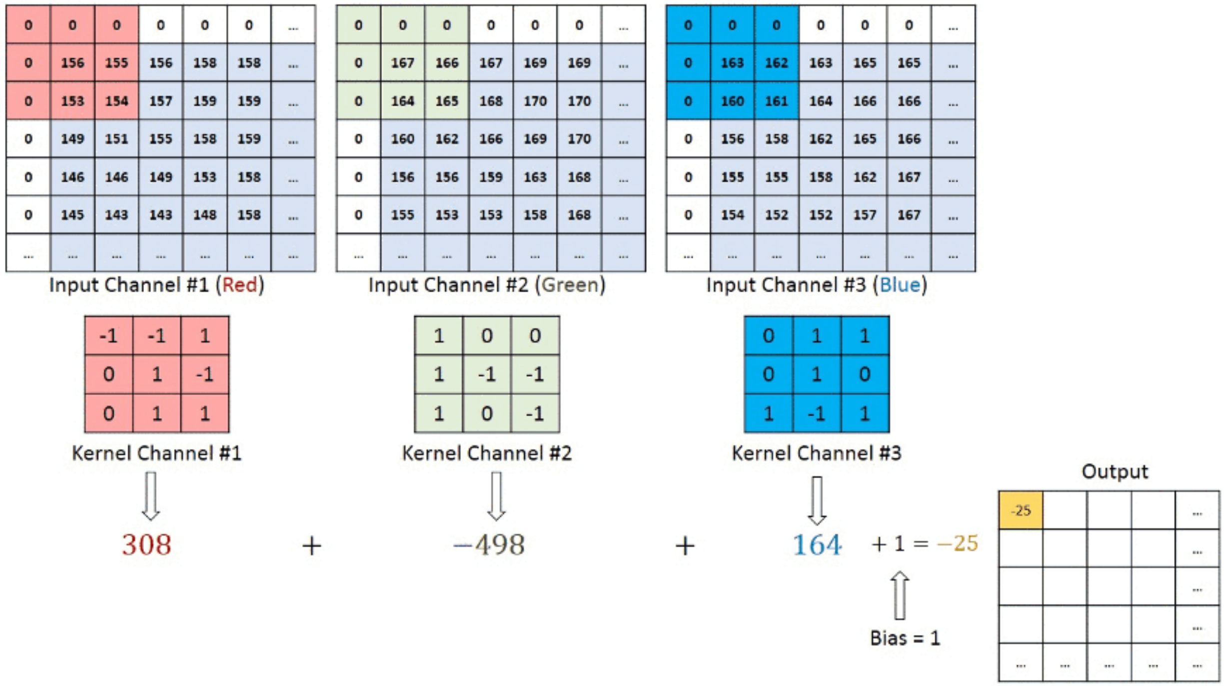
- The basic ideas of CNNs: 1d, 2d, 3d
- Transposed Convolutional
- 1x1 convolution (Network in Network)
- Fully Convolutional Network
- Skip Connection (ResNet, Densely Connected Convolutional Networks) and inverted residual structure (MobileNet v2)
- Dropout (Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift)
- Normalization: Batch Norm, Weight Norm, Layer Norm, Instance Norm, Group Norm (https://arxiv.org/pdf/1803.08494.pdf)
- Depth-wise Separable Convolution (MobileNet v1)
- Other visual tasks: Images Classification --- > Semantic Segmentation, Object Detection and Instance Segmentation

Credits to 11785 and CS231n

Convolution animations







Blue maps are **inputs**, and cyan maps are **outputs**.

https://github.com/vdumoulin/conv_arithmetic/blob/master/README.md

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

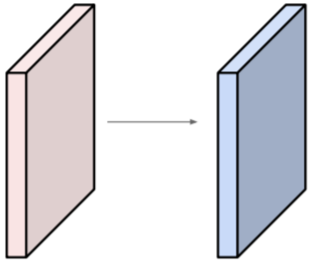
n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

Examples Time

Input volume: 32x32x3 10 5x5 filters with stride 1, pad 2

1. Output volume size: ?
2. Number of parameters in this layer?

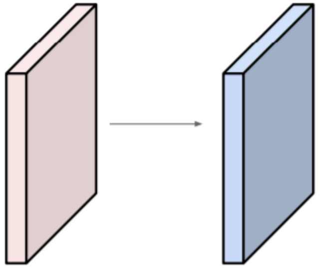
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Credits to cs231n

Why do we use Convolutional Neural Networks?

1. Local Connectivity: The connections are local in space (along width and height), but always full along the entire depth of the input volume. True or False ? (Depth-wise Separable Convolution)
2. Parameter Sharing: Parameter sharing scheme is used in Convolutional Layers to control the number of parameters.

Credits to cs231n

Convolution to reduce complexity

The input is a sequence of 1000 words. Each word is represented as a 100-dimensional vector.

The output is another 1000x100 matrix.

If we do this with a normal NN layer, how many parameters do we need?

Convolution to reduce complexity

The input is a sequence of 1000 words. Each word is represented as a 100-dimensional vector.

The output is another 1000x100 matrix.

If we do this with a normal NN layer, how many parameters do we need?

*Weight matrix: (1000x100) ** 2 = 10,000,000,000*

Biases: 1000x100 = 100,000

We need 10,000,100,000 parameters

Convolution to reduce complexity

The input is a sequence of 1000 words. Each word is represented as a 100-dimensional vector.

The output is another 1000x100 matrix.

If we use a convolutional layer with a kernel size of 5 (plus a bias term for each of the 100 output channels), how many parameters do we need?

Convolution to reduce complexity

The input is a sequence of 1000 words. Each word is represented as a 100-dimensional vector. --- > time length:1000, channels: 100

The output is another 1000x100 matrix. (With Padding) --- > output time length: 1000, output channels: 100

If we use a convolutional layer with a kernel size of 5 (plus a bias term for each of the 100 output channels), how many parameters do we need?

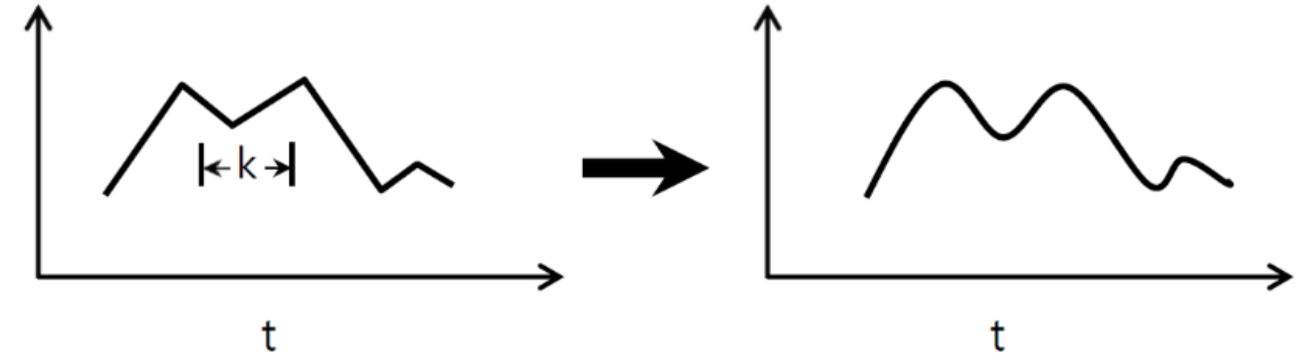
Convolution kernels: 100 x 100 x 5 = 50,000

Biases: 100

We need 50,100 parameters

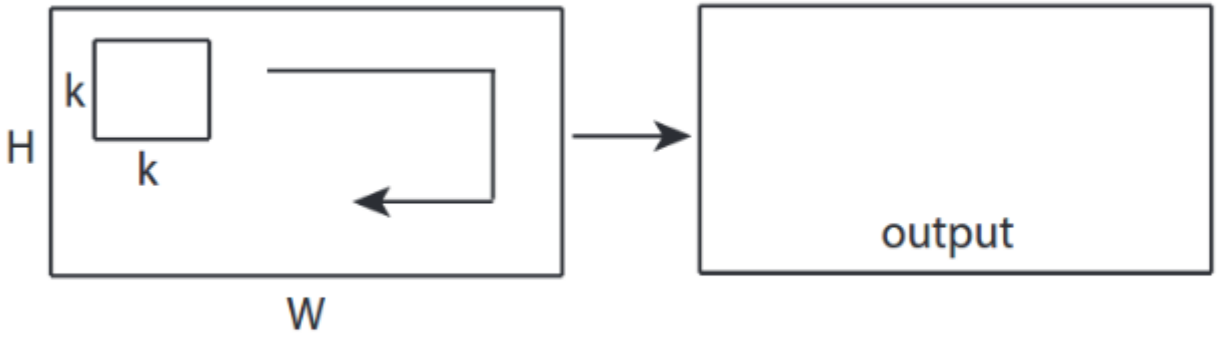
Conv 1d, 2d, 3d

- 1d:



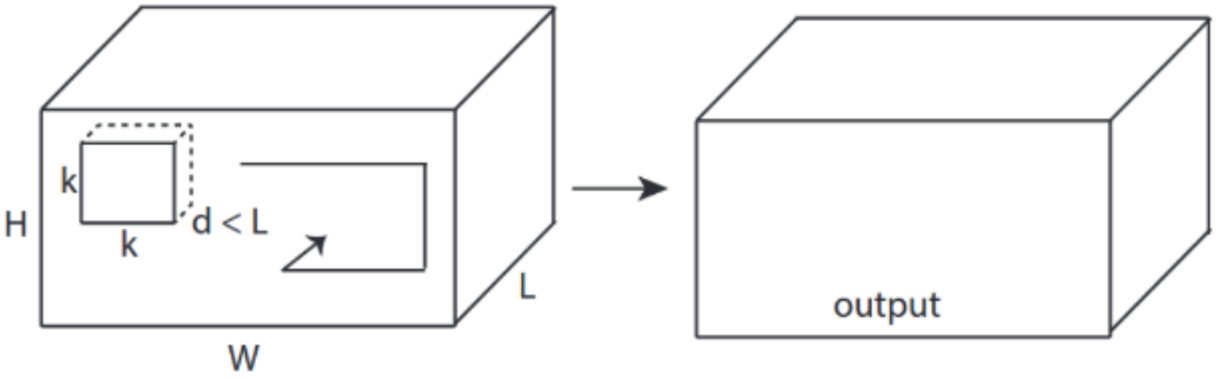
natural language processing

- 2d:



Computer Vision

- 3d:

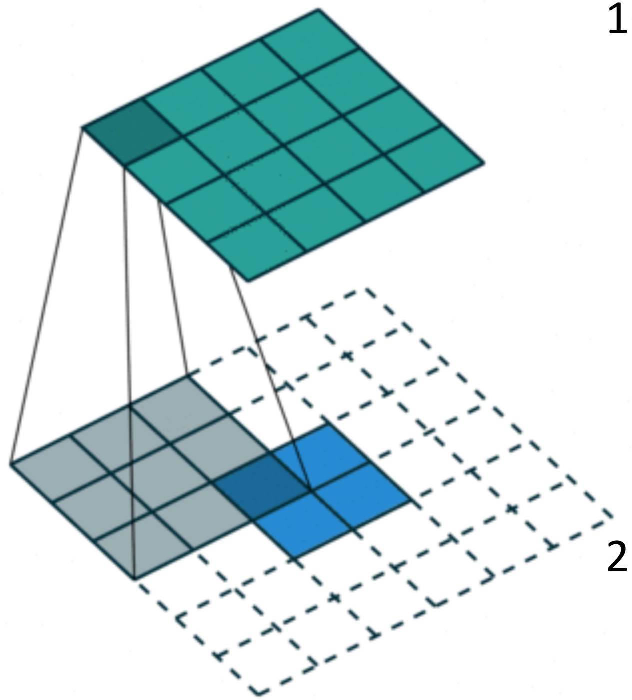


Videos

Paper: Learning Spatiotemporal Features with 3D Convolutional Networks

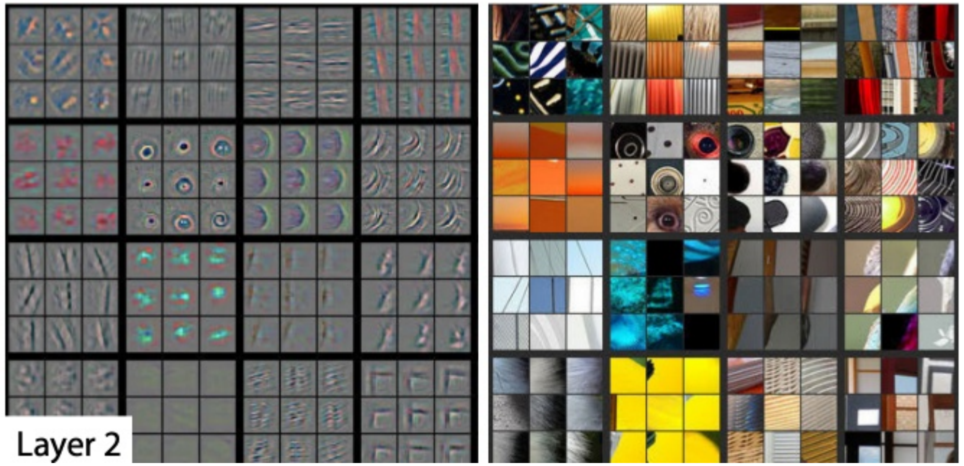
A general but not accurate description

Transposed Convolutional

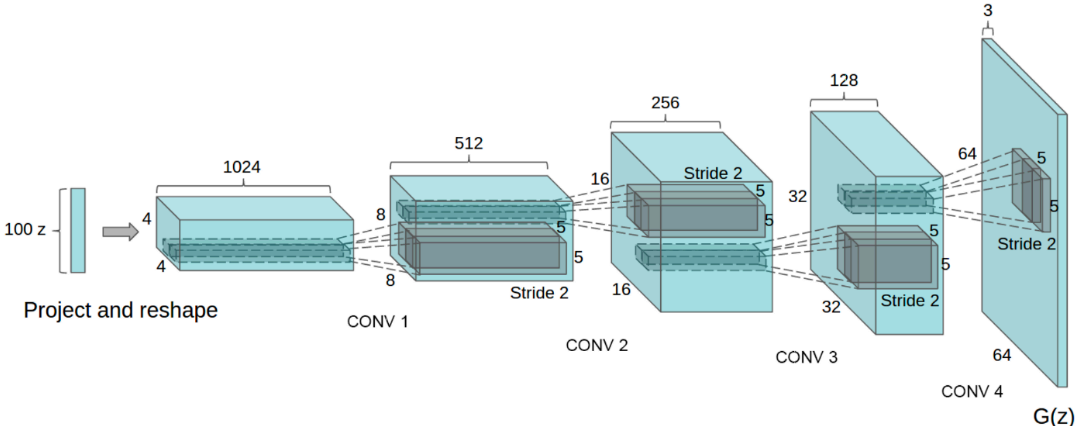


Applications:

1. Visualization with a Deconvnet: Visualizing and Understanding Convolutional Networks



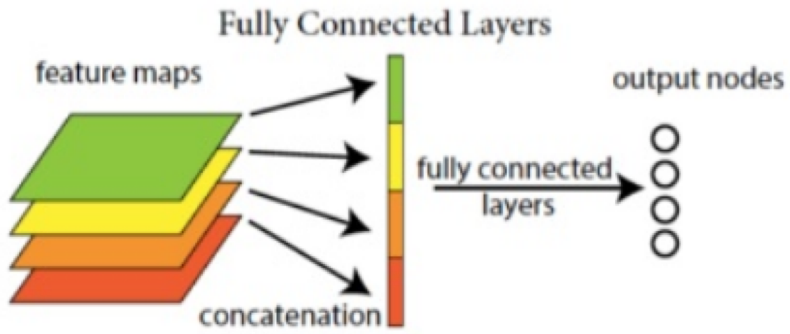
2. Deep convolutional generative adversarial networks:



Blue maps are **inputs**, and cyan maps are **outputs**.

Fully-connected layers (1x1 convolution)

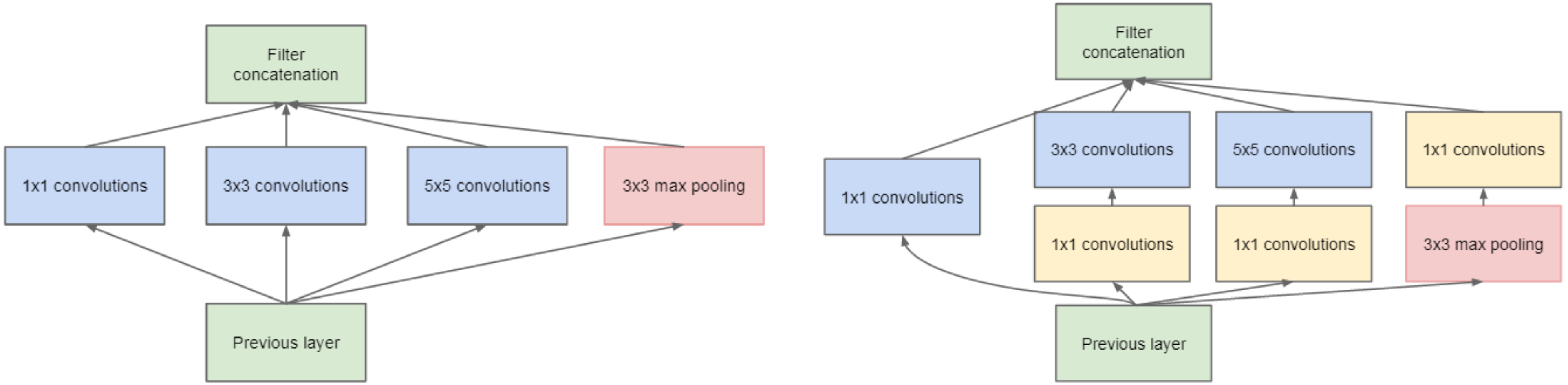
Simply treat the final feature map as a vector, and use a fully-connected neural network that outputs the desired number of dimensions.



This can only be done after all convolutions, because the output of the fully-connected layer does not store any location information. Performing a convolution on the output of a fully-connected layer will be meaningless!

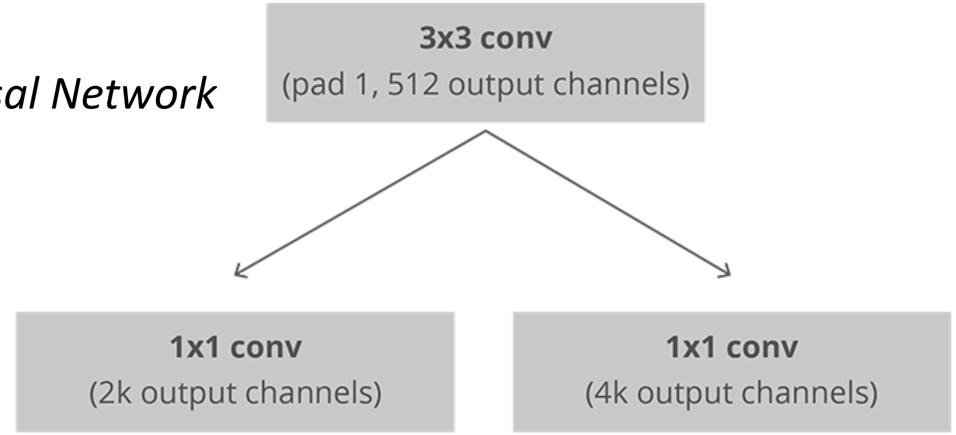
Credits to 11785

1x1 convolution (Network in Network)



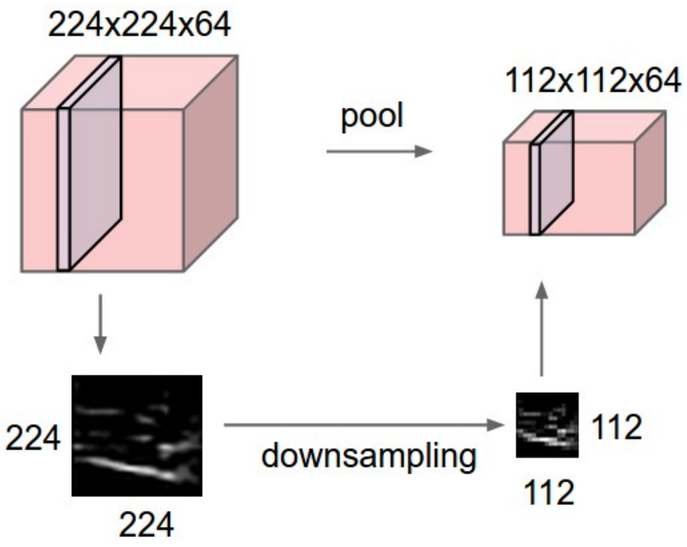
(a) Inception module, naïve version (b) Inception module with dimension reductions

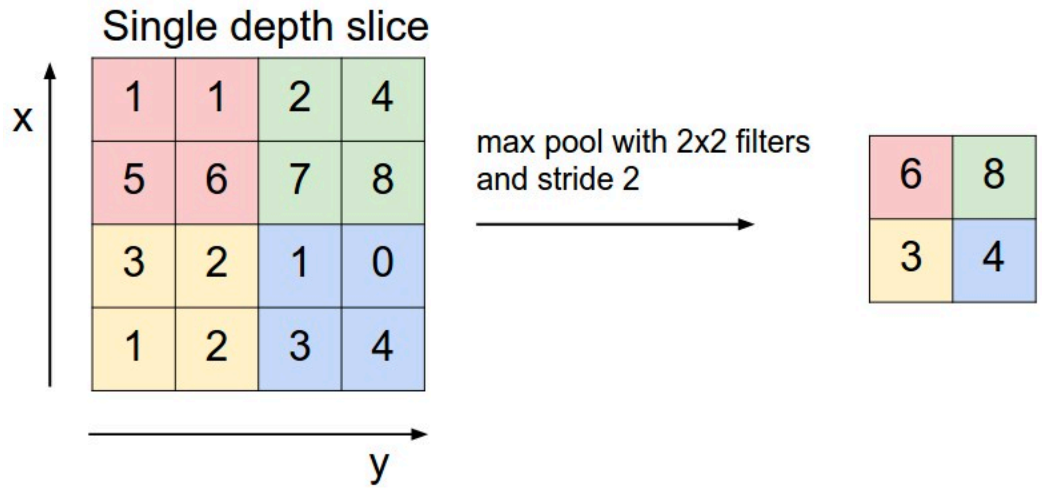
Convolutional implementation of an Region Proposal Network architecture, where k is the number of anchors:
(Faster R-CNN)



Try in hw2p2: fully connected layers can also be replaced by simple 1-by-1 convolutions

Pooling





Notice that the volume depth is preserved.

Its function is to progressively **reduce the spatial size of the representation** to reduce the amount of parameters and computation in the network, and hence to also control overfitting.

Fully Convolutional Network

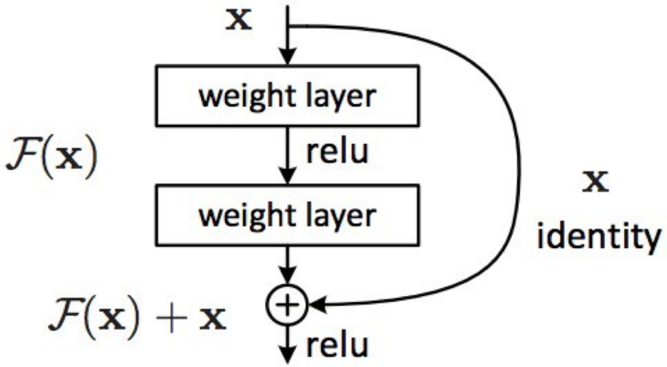
Getting rid of pooling. Many people dislike the pooling operation and think that we can get away without it. For example, [Striving for Simplicity: The All Convolutional Net] proposes to discard the pooling layer in favor of architecture that only consists of repeated CONV layers. To reduce the size of the representation they suggest using larger stride in CONV layer once in a while. Discarding pooling layers has also been found to be important in training good generative models, such as variational autoencoders (VAEs) or generative adversarial networks (GANs). It seems likely that future architectures will feature very few to no pooling layers

Model		
Strided-CNN-C	ConvPool-CNN-C	All-CNN-C
	Input 32×32 RGB image	
3×3 conv. 96 ReLU 3×3 conv. 96 ReLU with stride $r = 2$	3×3 conv. 96 ReLU 3×3 conv. 96 ReLU 3×3 conv. 96 ReLU	3×3 conv. 96 ReLU 3×3 conv. 96 ReLU
	3×3 max-pooling stride 2	3×3 conv. 96 ReLU with stride $r = 2$
3×3 conv. 192 ReLU 3×3 conv. 192 ReLU with stride $r = 2$	3×3 conv. 192 ReLU 3×3 conv. 192 ReLU 3×3 conv. 192 ReLU	3×3 conv. 192 ReLU 3×3 conv. 192 ReLU
	3×3 max-pooling stride 2	3×3 conv. 192 ReLU with stride $r = 2$

$\vdots$

Credits to cs231n

Skip Connection



Deep Residual Learning for Image Recognition

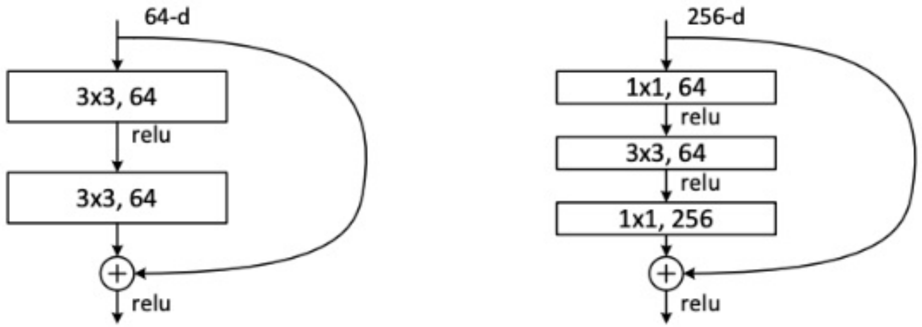
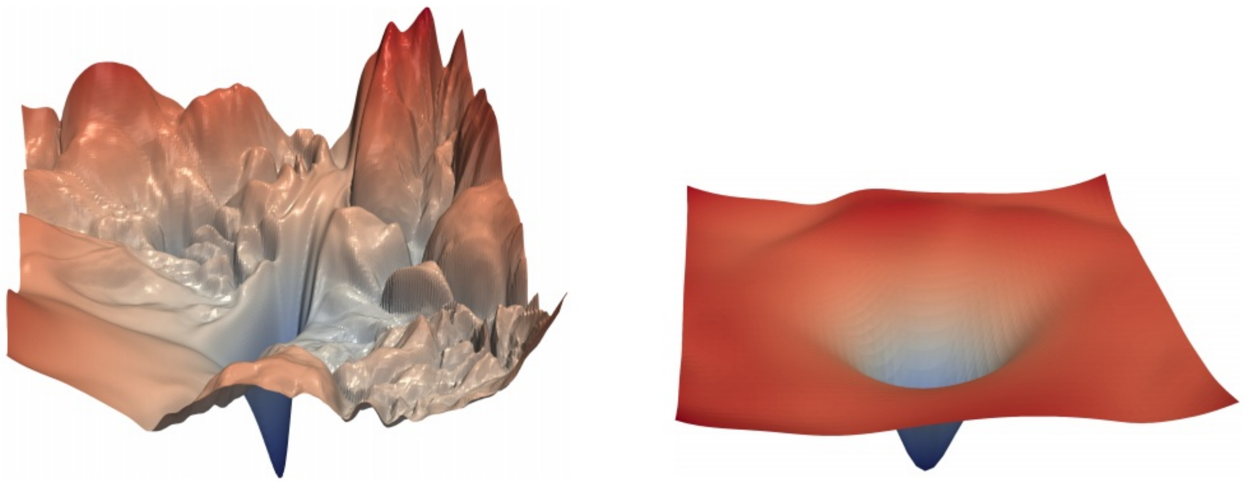


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

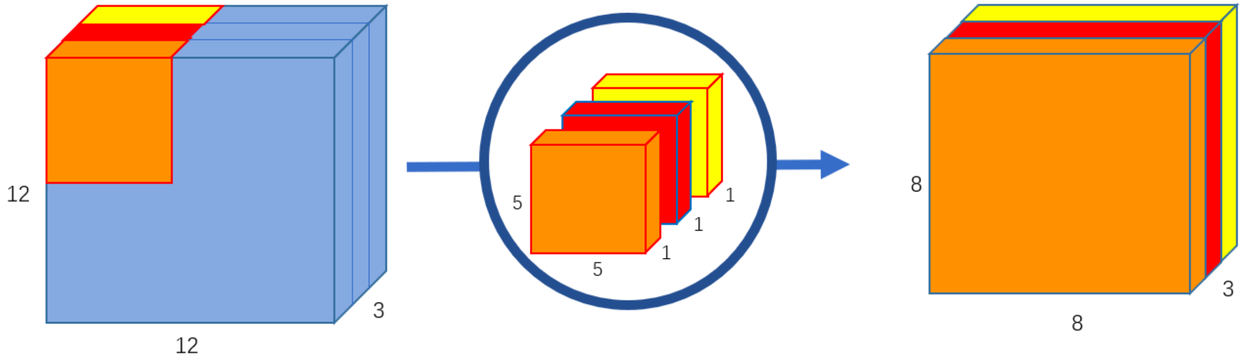
Paper: Visualizing the Loss Landscape of Neural Nets

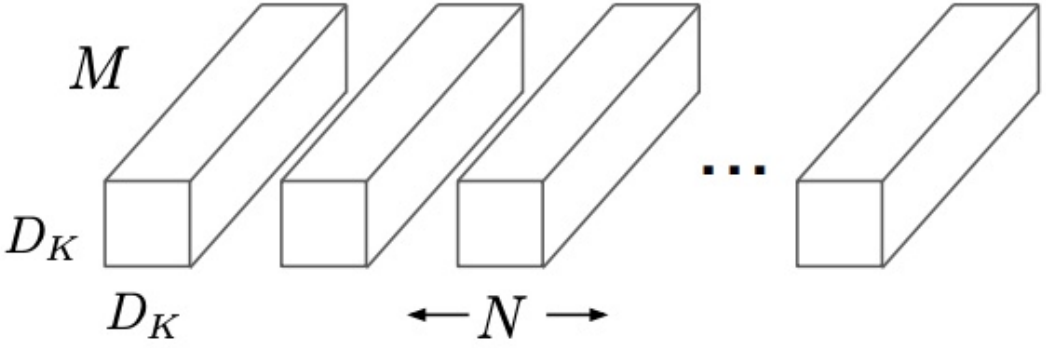


(a) without skip connections (b) with skip connections

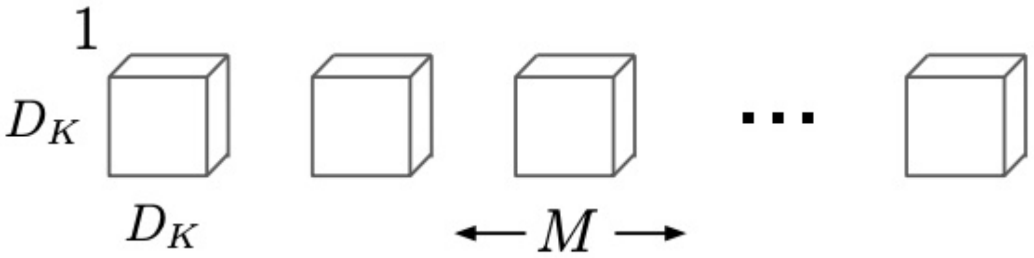
Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Depth-wise Separable Convolution





(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters

Each 5x5x1 kernel iterates 1 channel of the image (note: **1 channel**, not all channels), getting the scalar products of every 25 pixel group, giving out a 8x8x1 image. Stacking these images together creates a 8x8x3 image.

Paper: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

Inverted Residual Block

(a) Residual block (b) Inverted residual block

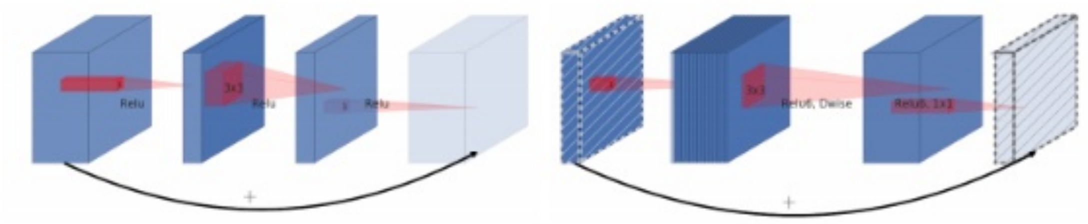
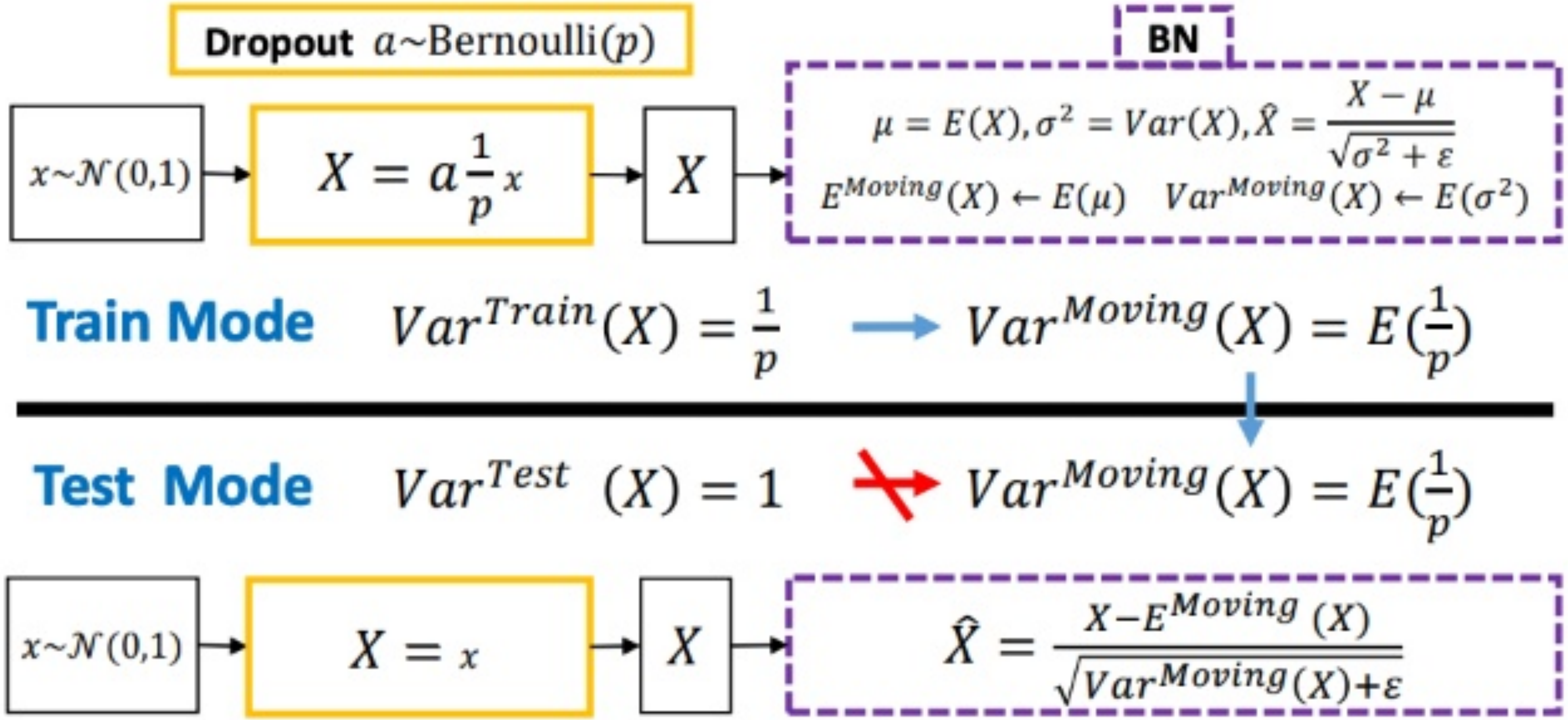


Figure 3: The difference between residual block [8, 30] and inverted residual. Diagonally hatched layers do not use non-linearities. We use thickness of each block to indicate its relative number of channels. Note how classical residuals connects the layers with high number of channels, whereas the inverted residuals connect the bottlenecks. Best viewed in color.

Dropout

why do the two most powerful techniques **Dropout** and **Batch Normalization (BN)** often lead to a worse performance when they are combined together?



Paper: Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift

Normalization

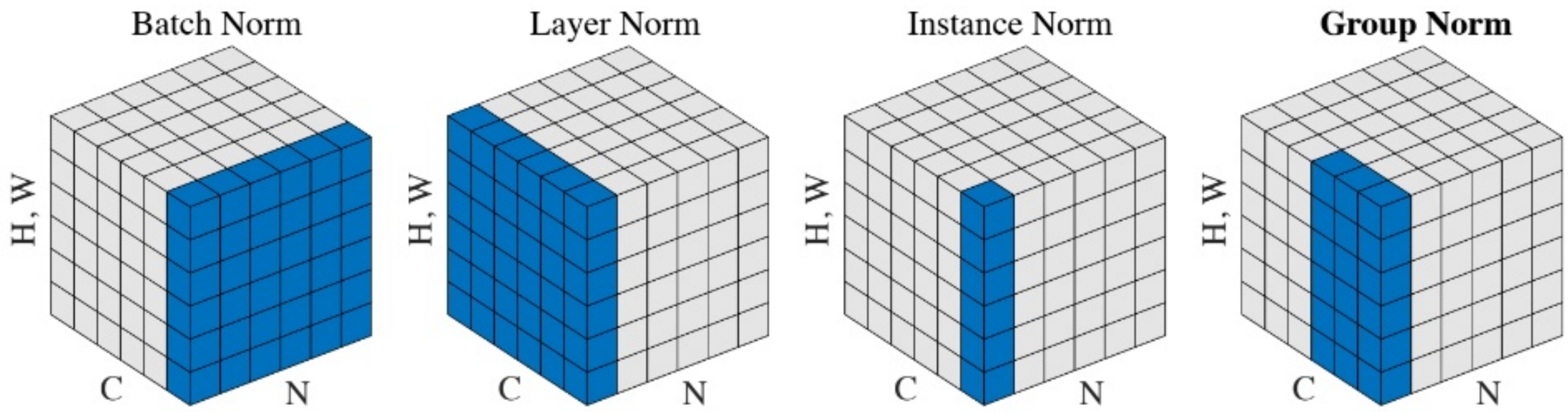
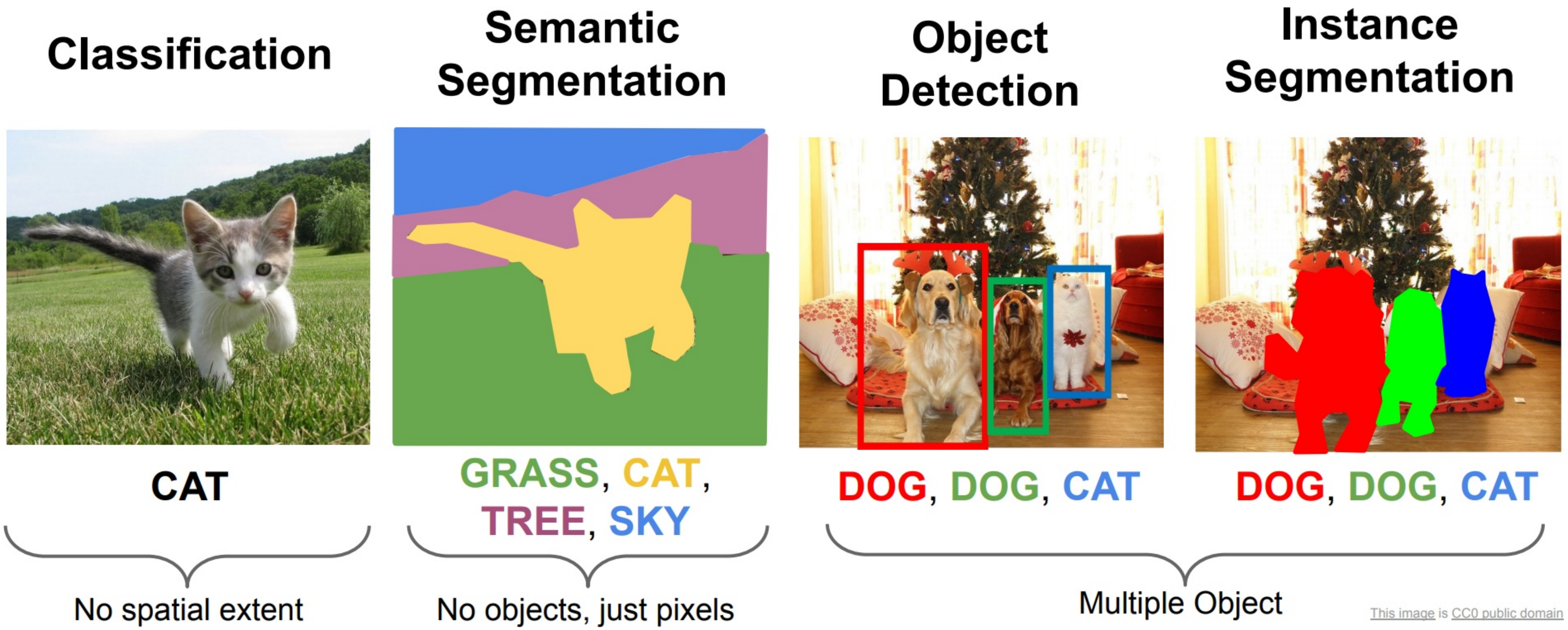


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Transformer

Image style transfer

Computer Vision Tasks



Credits to cs231n

References

- 3d conv: Learning Spatiotemporal Features with 3D Convolutional Networks
- Visualizing the Loss Landscape of Neural Nets
- Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift
- Deep Residual Learning for Image Recognition
- Deep convolutional generative adversarial networks
- Visualizing and Understanding Convolutional Networks
- MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications
- MobileNetV2: Inverted Residuals and Linear Bottlenecks
- Some Slides from CS231n and 11785