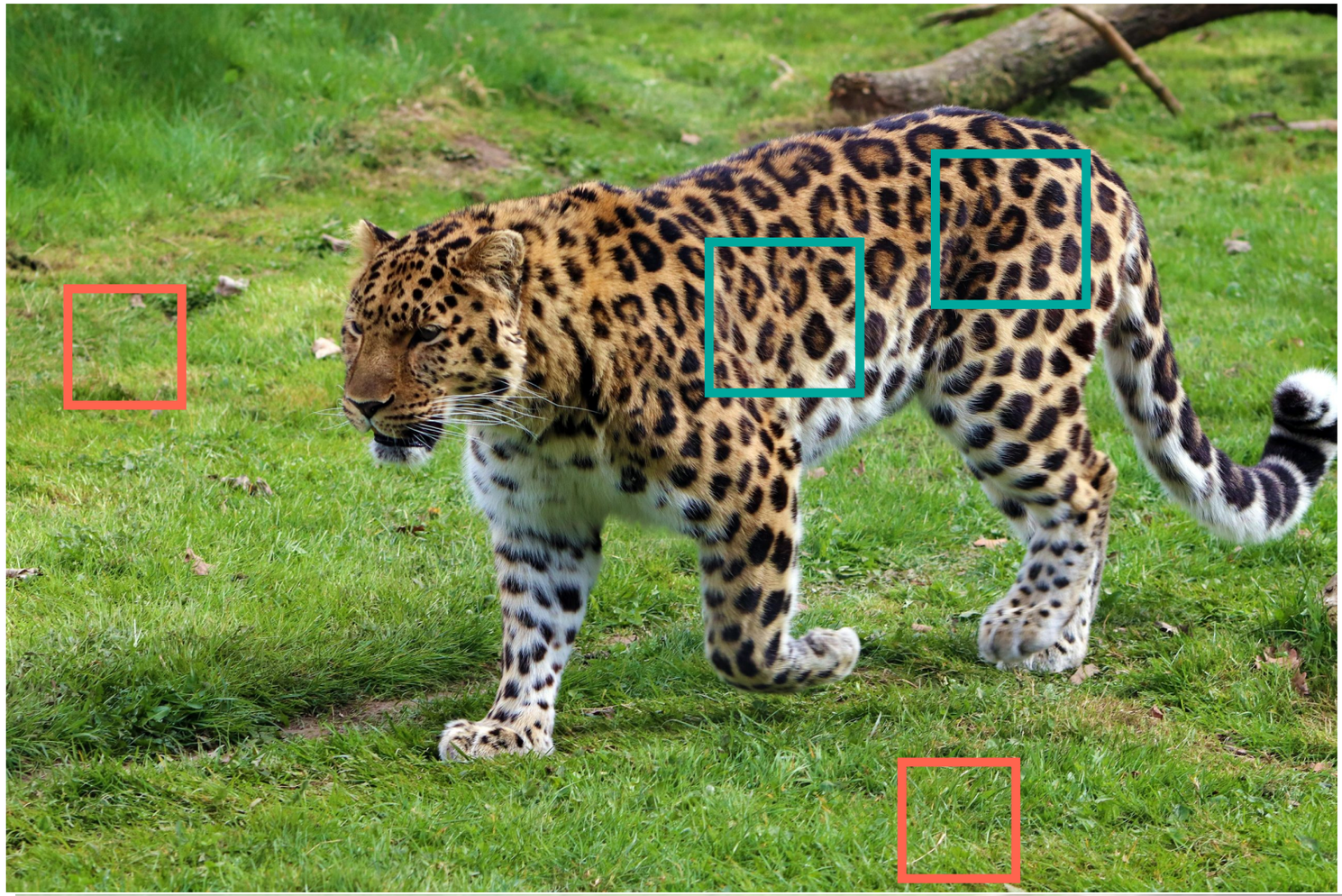
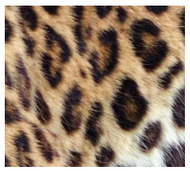
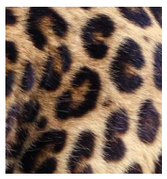
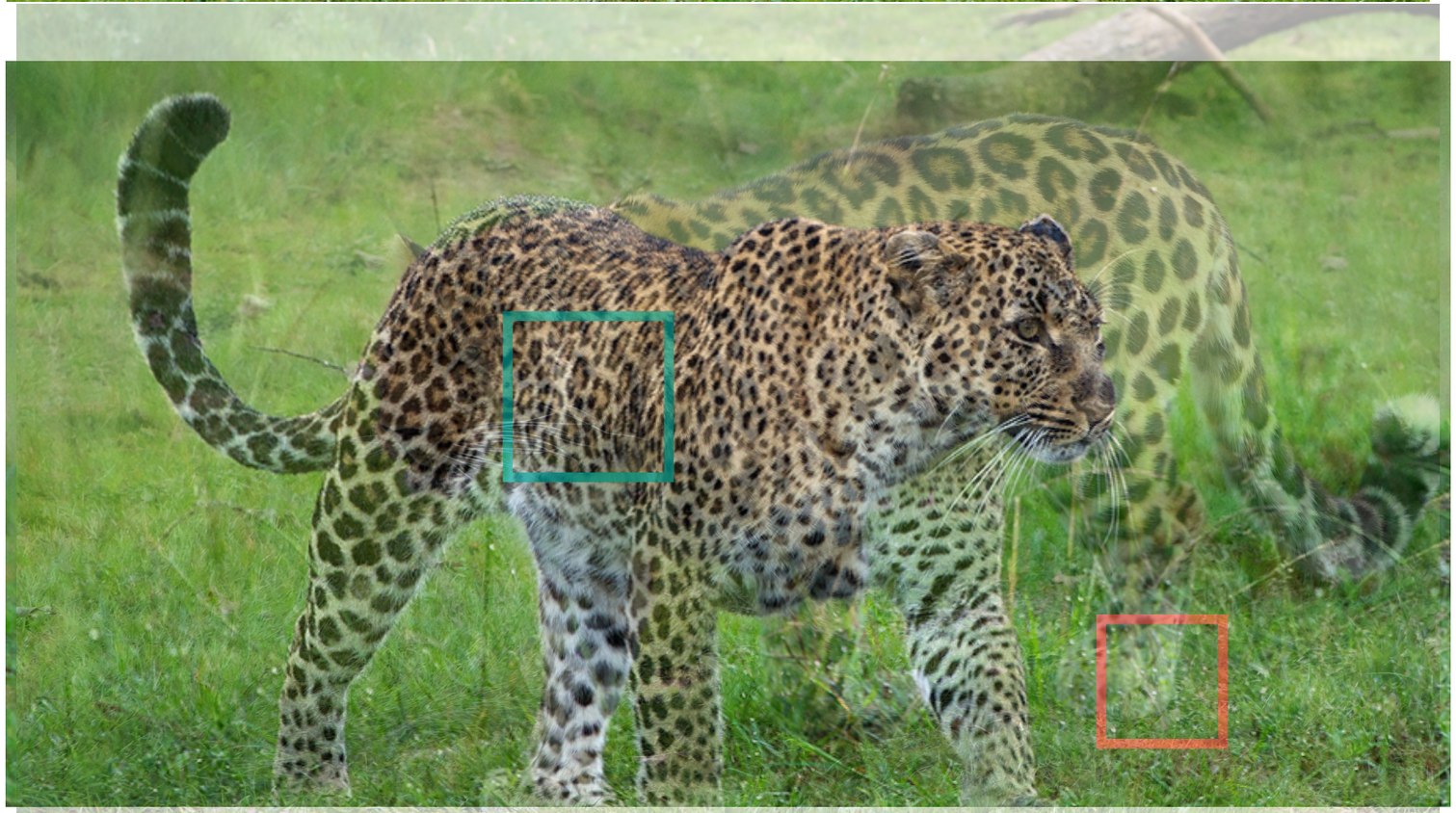
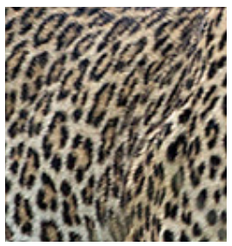
Lecture 5: Convolutional Neural Networks

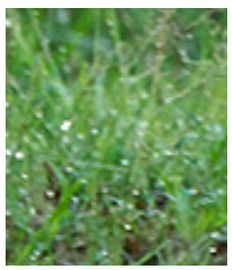








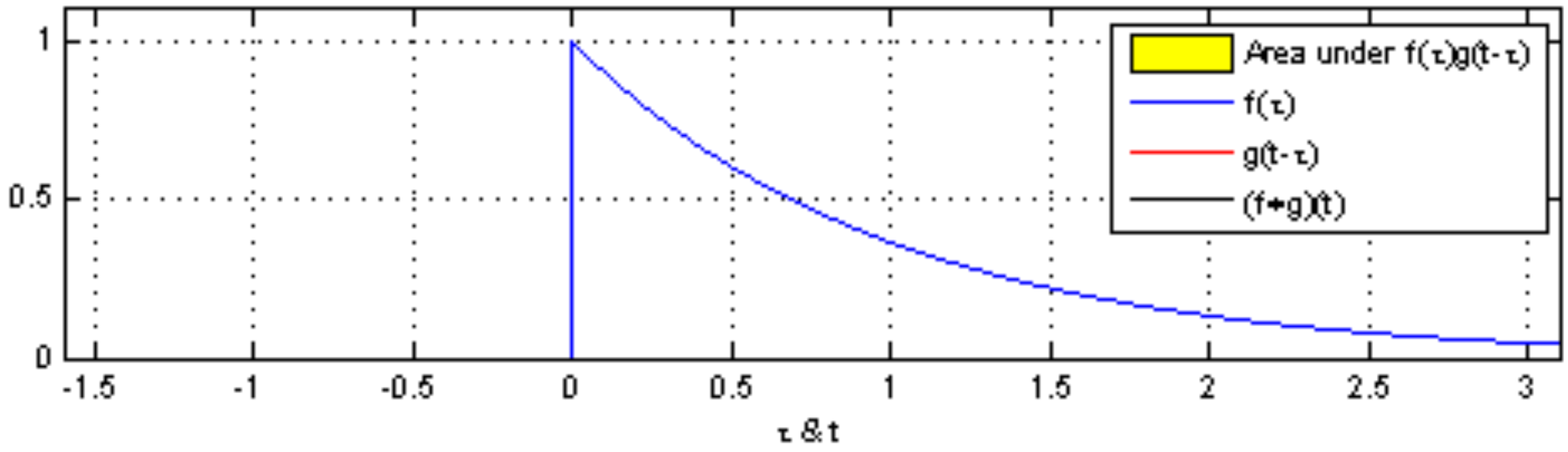




Convolution Operation

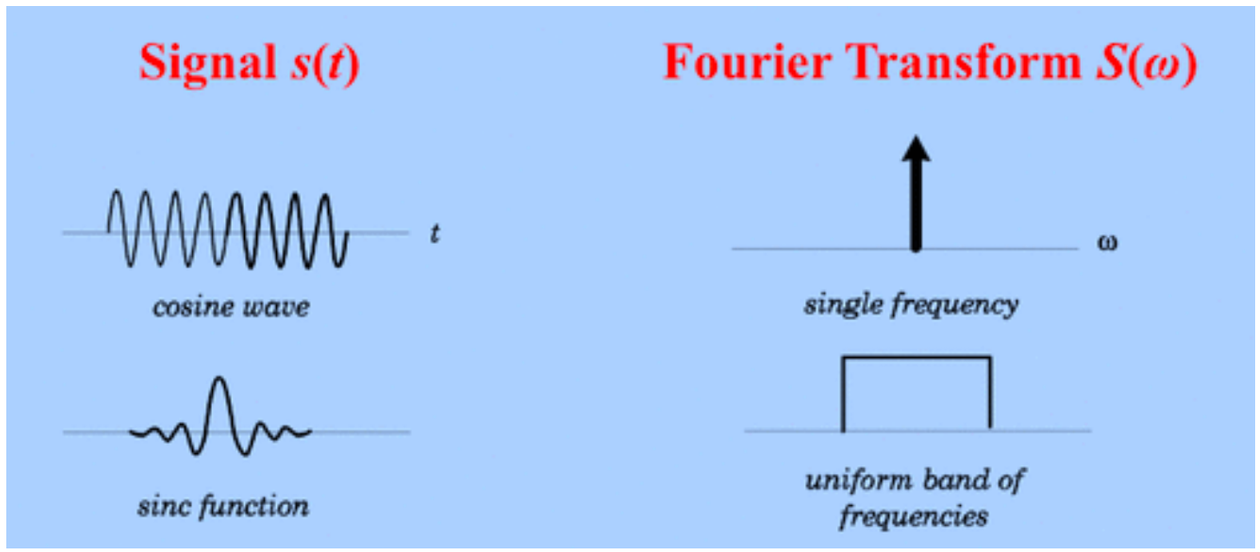
$$s(t) = \int x(a)w(t-a)da.$$

$$s(t) = (x * w)(t).$$

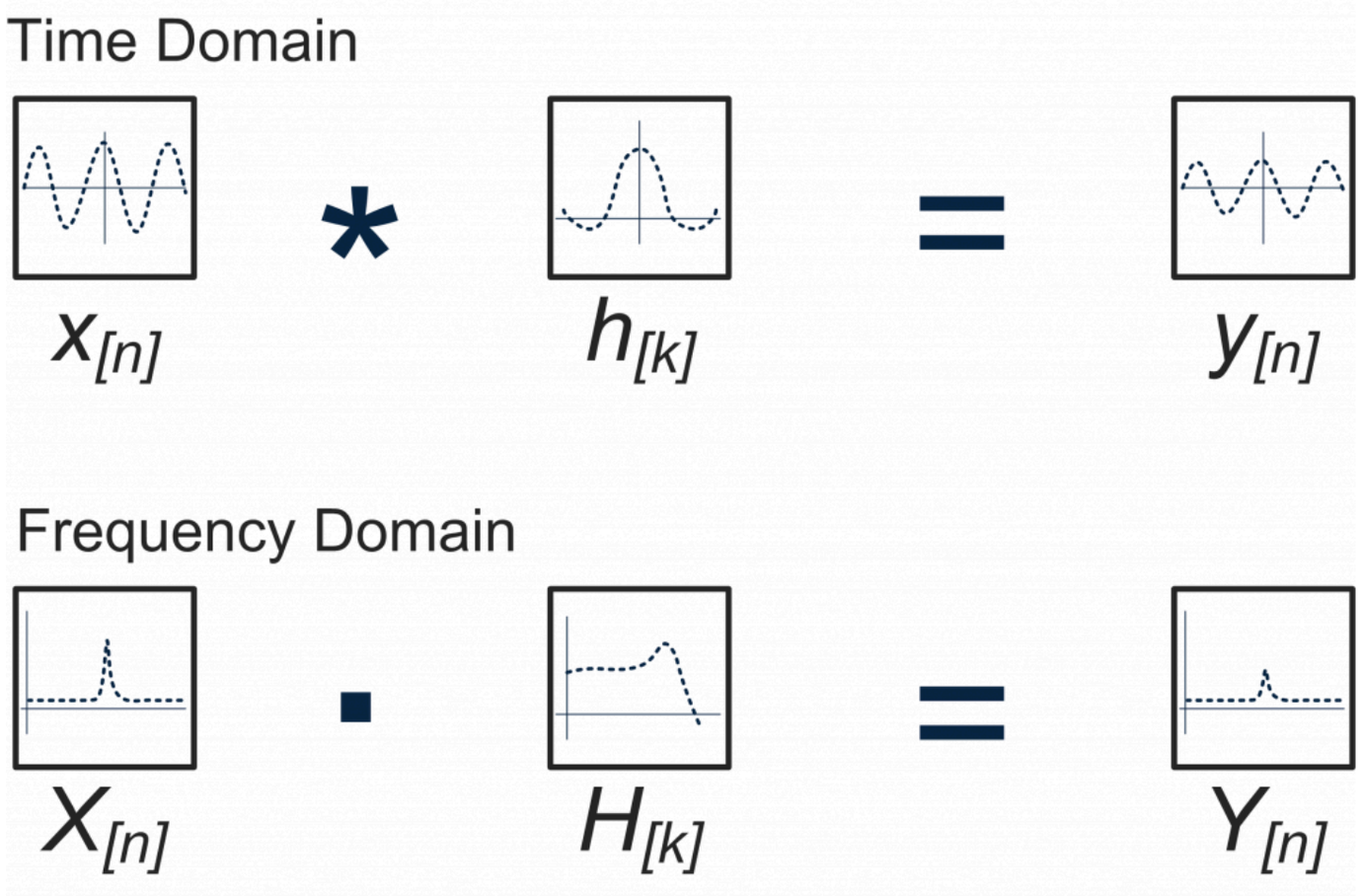


From Wikimedia

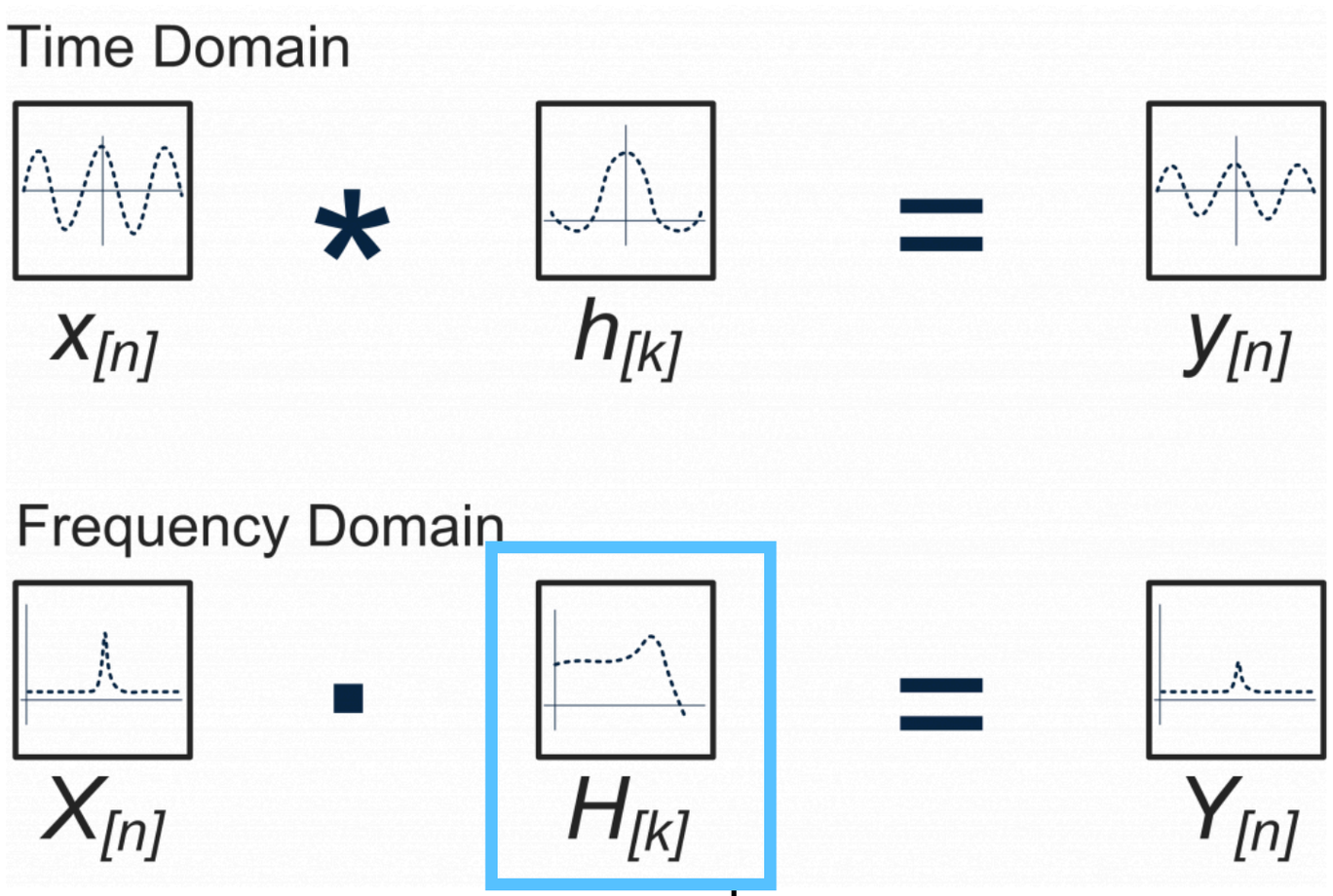
Convolution Operation

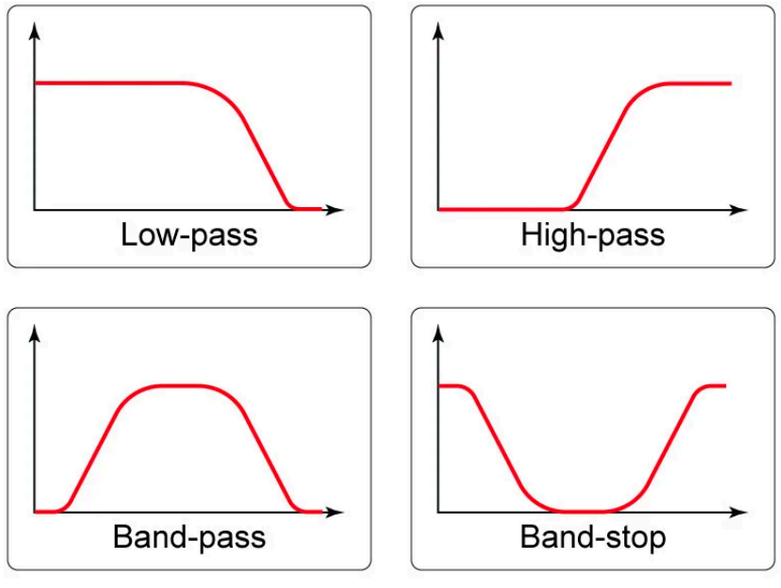


Convolution -> Multiplication in Frequency Domain



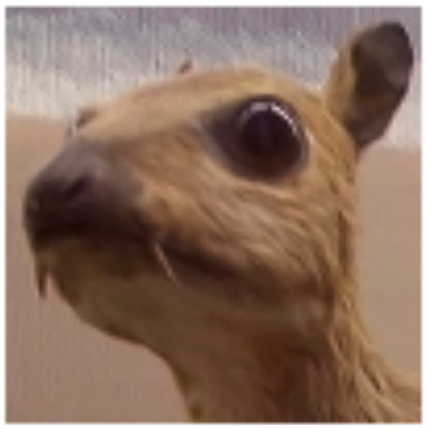
Convolution Operation

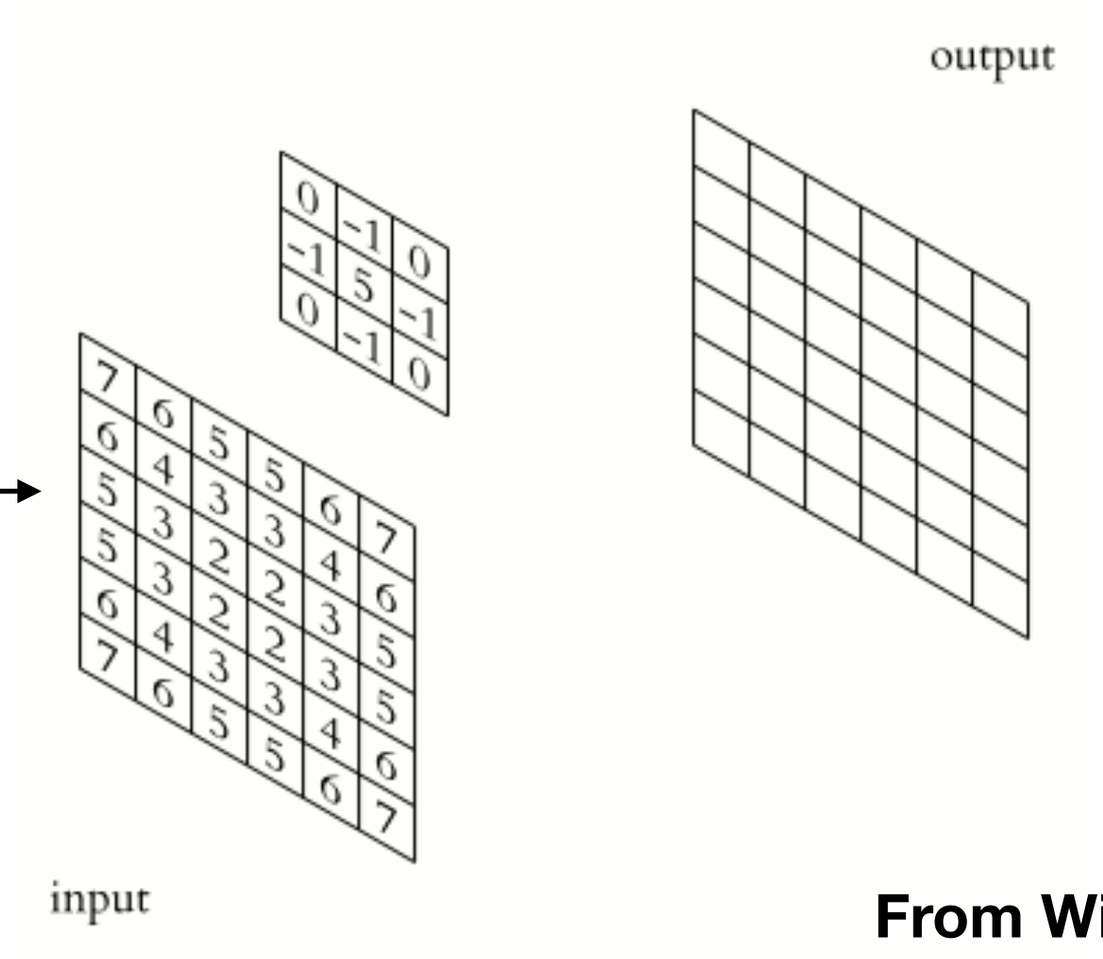




Frequency domain view

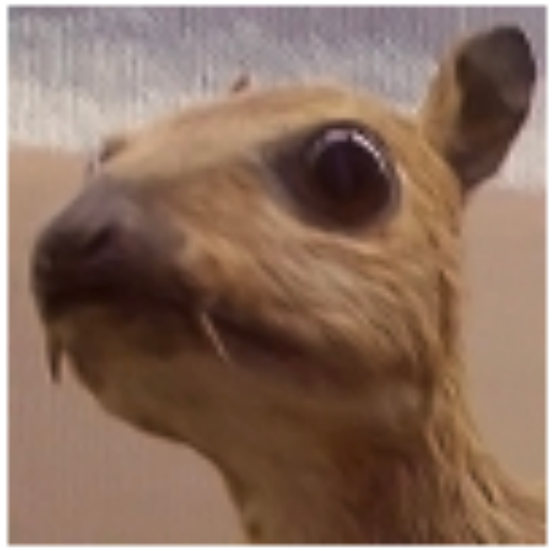
Discrete Convolution in 2D





From Wikimedia

Intuition for Convolutional Filters



Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

From Tim Detmers

Is this a high pass or low pass filter?

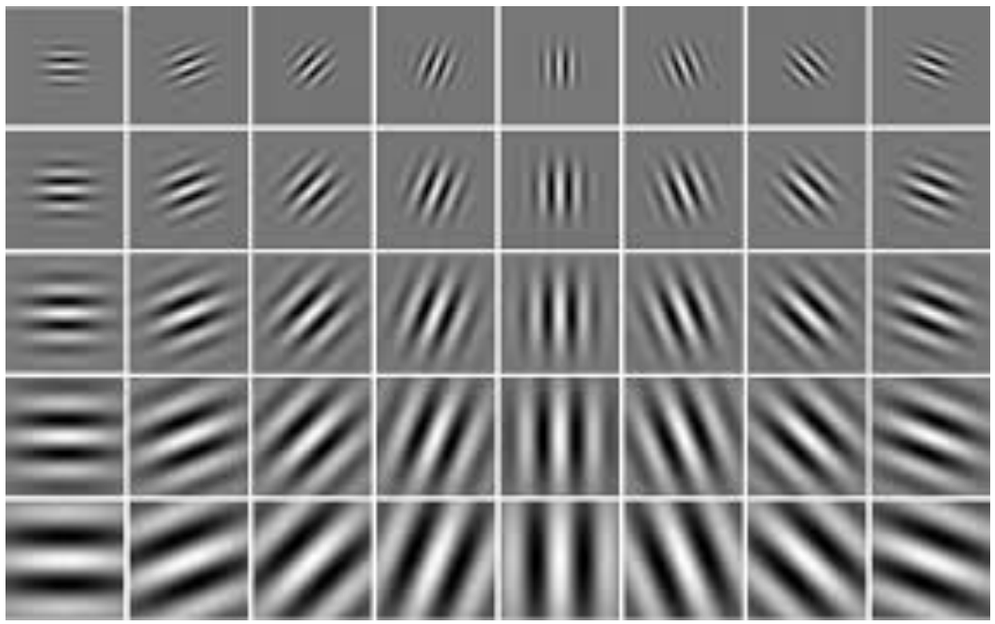
Convolution in Signal Processing

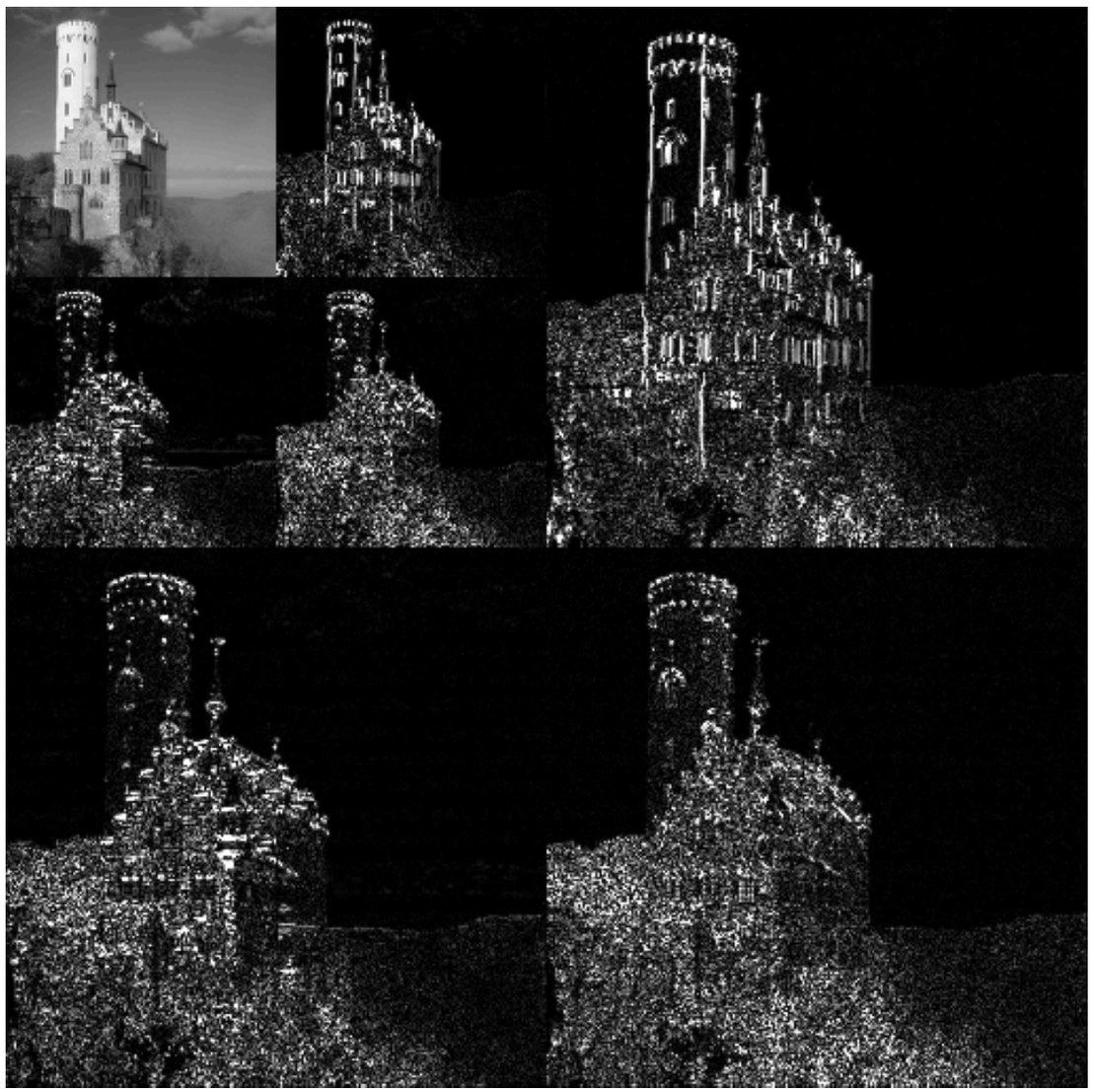
- Heavily used in Digital Signal Processing
  - Image processing such as denoising and compression
  - Audio Processing

Convolution in Image Processing and Vision

- Many classic hand-crafted computer vision methods rely on convolutions or similar operations

- Wavelet fitlerbanks

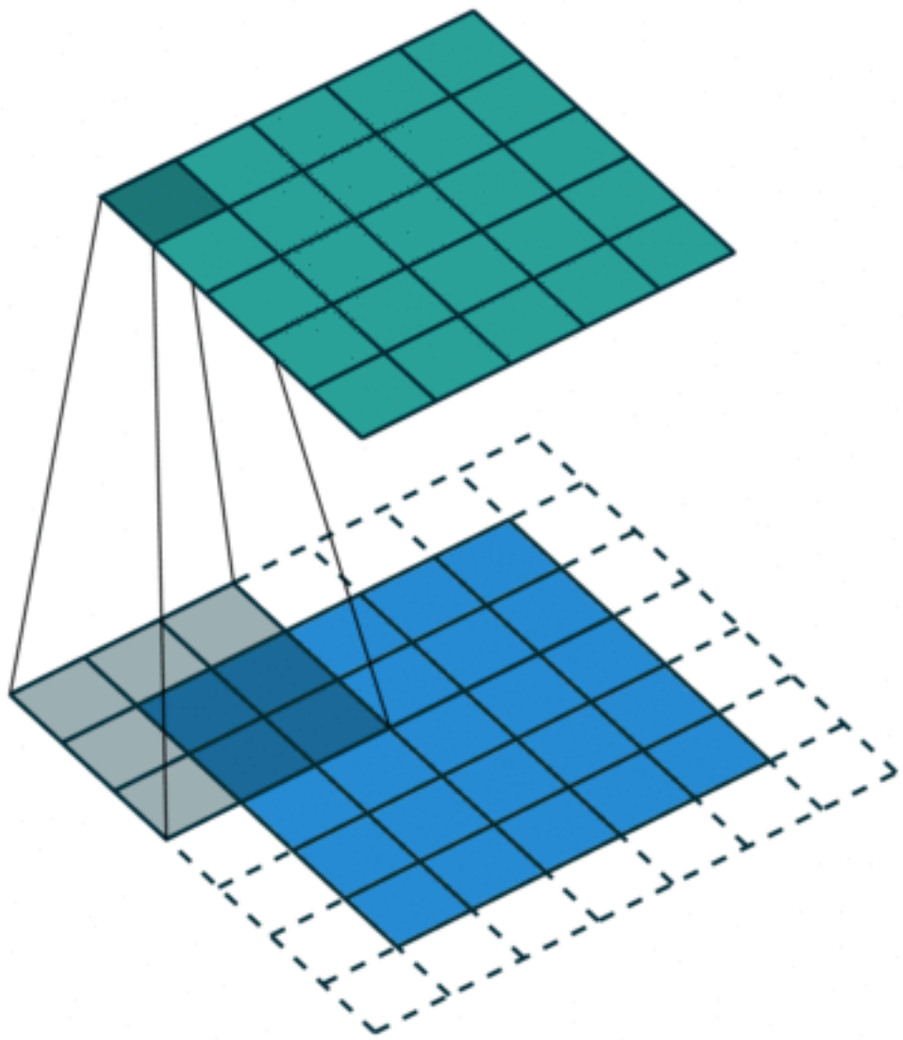




Convolutional Neural Network layers can be seen as creating more flexible (data derived) versions of these classic methods

Padding

Padding can use zeros or copy the last pixel



Output

Input

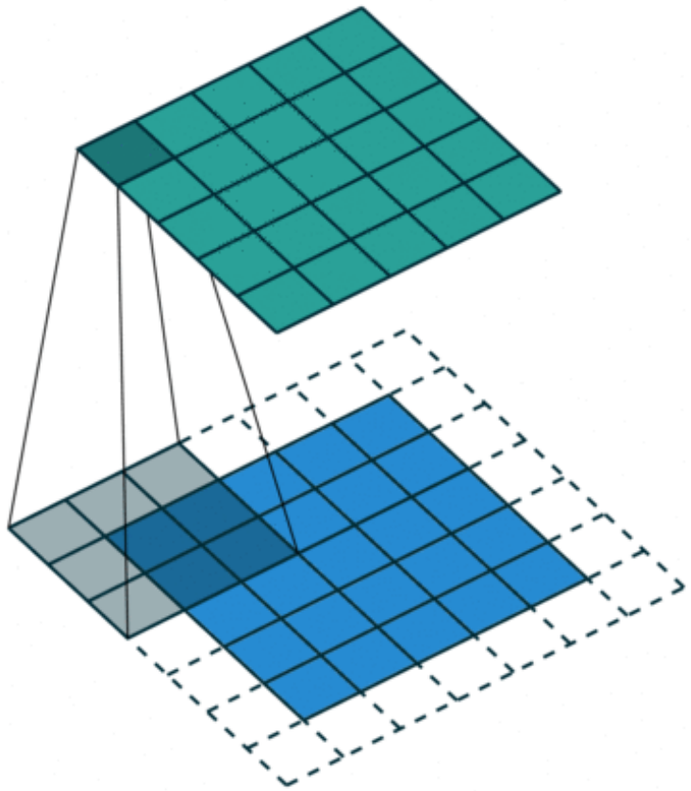
A guide to convolution arithmetic for deep learning

Vincent Dumoulin, Francesco Visin

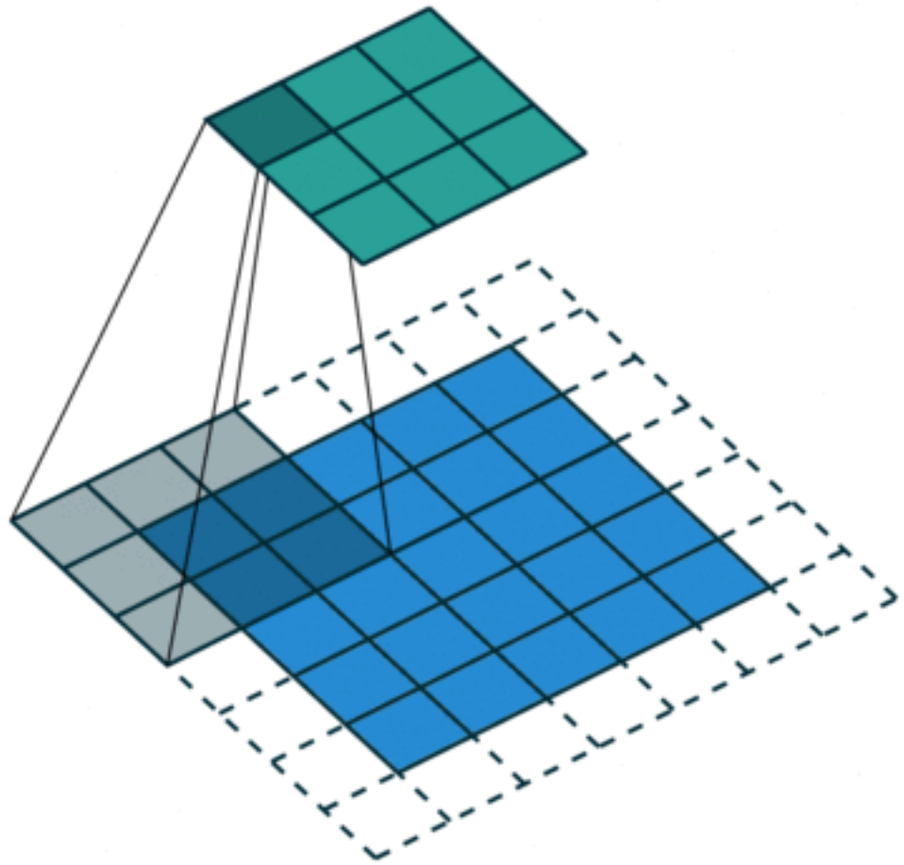
https://github.com/vdumoulin/conv_arithmetic

Strides

Padding = 1



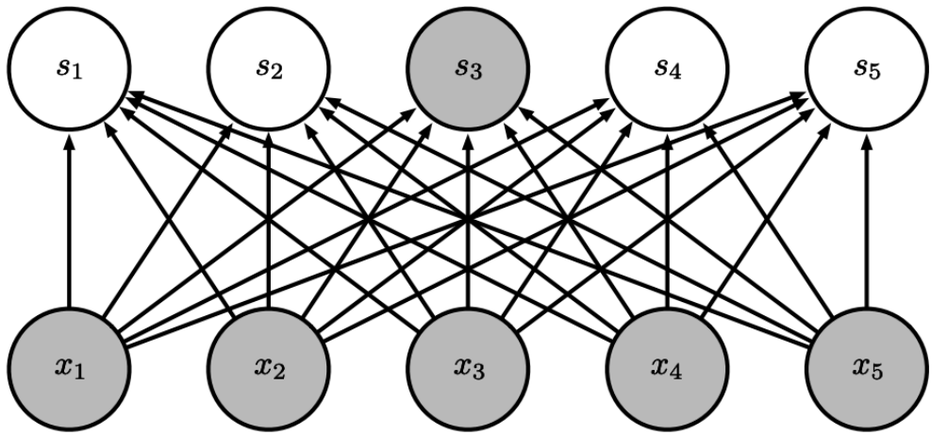
Stride = 2



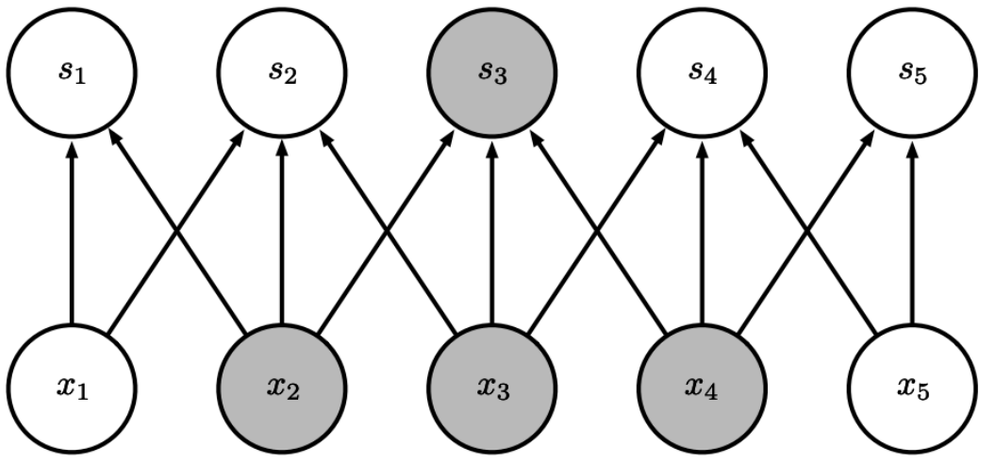
Stride - how many pixels we move the kernel each step

Convolutional Layer

Fully connected



Locally connected



Conv Layer

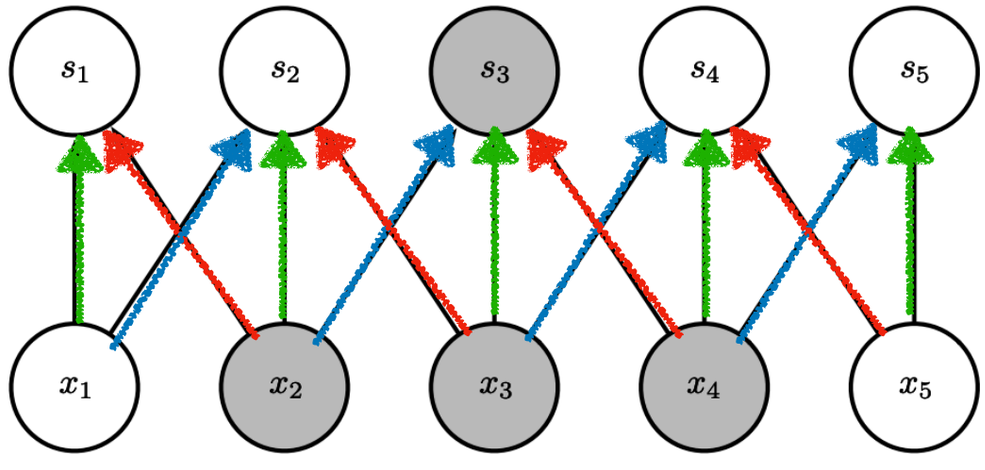
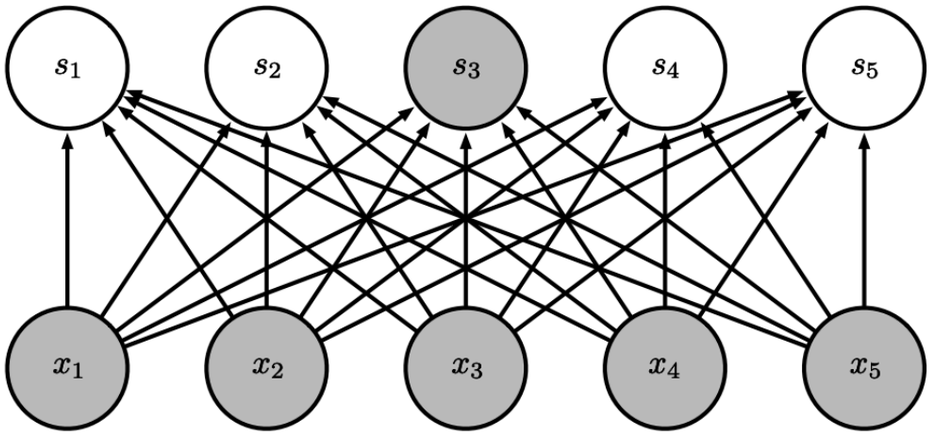


Image Credit: Goodfellow

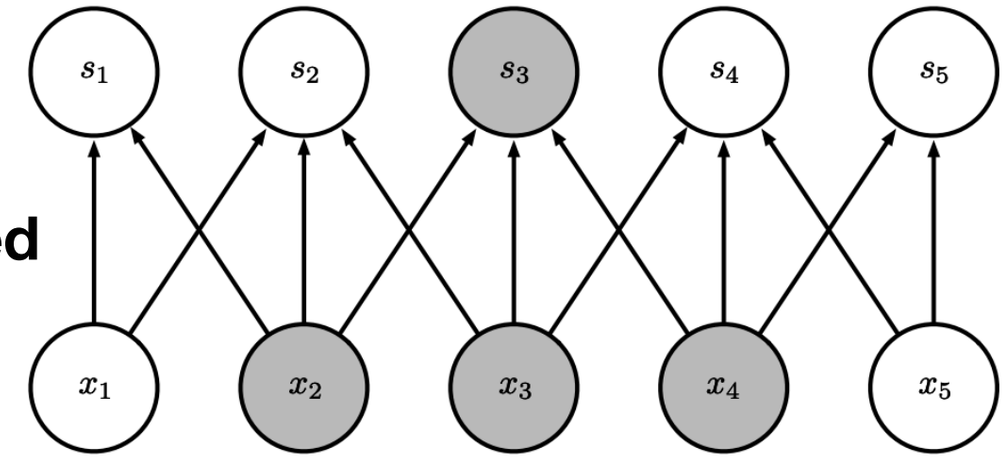
Convolutional Layer

Fully connected



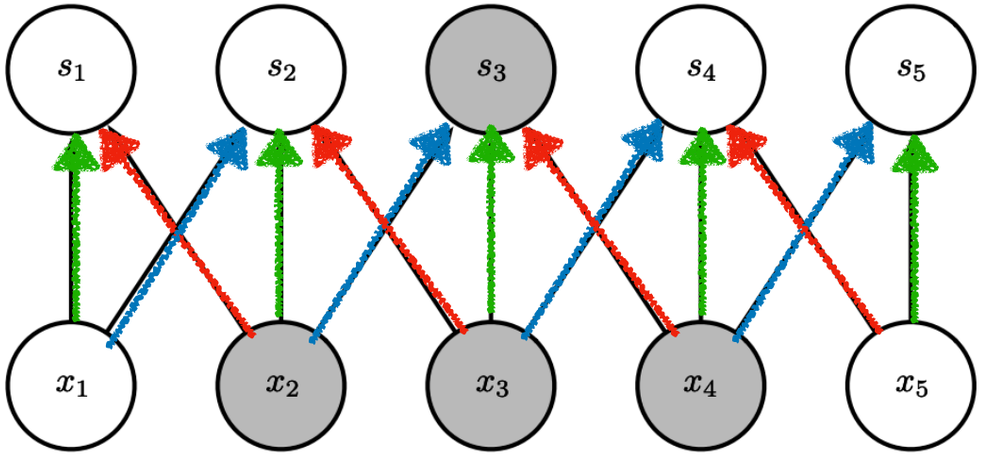
$$f_{W_1}(x) = W_1 x$$

Locally connected



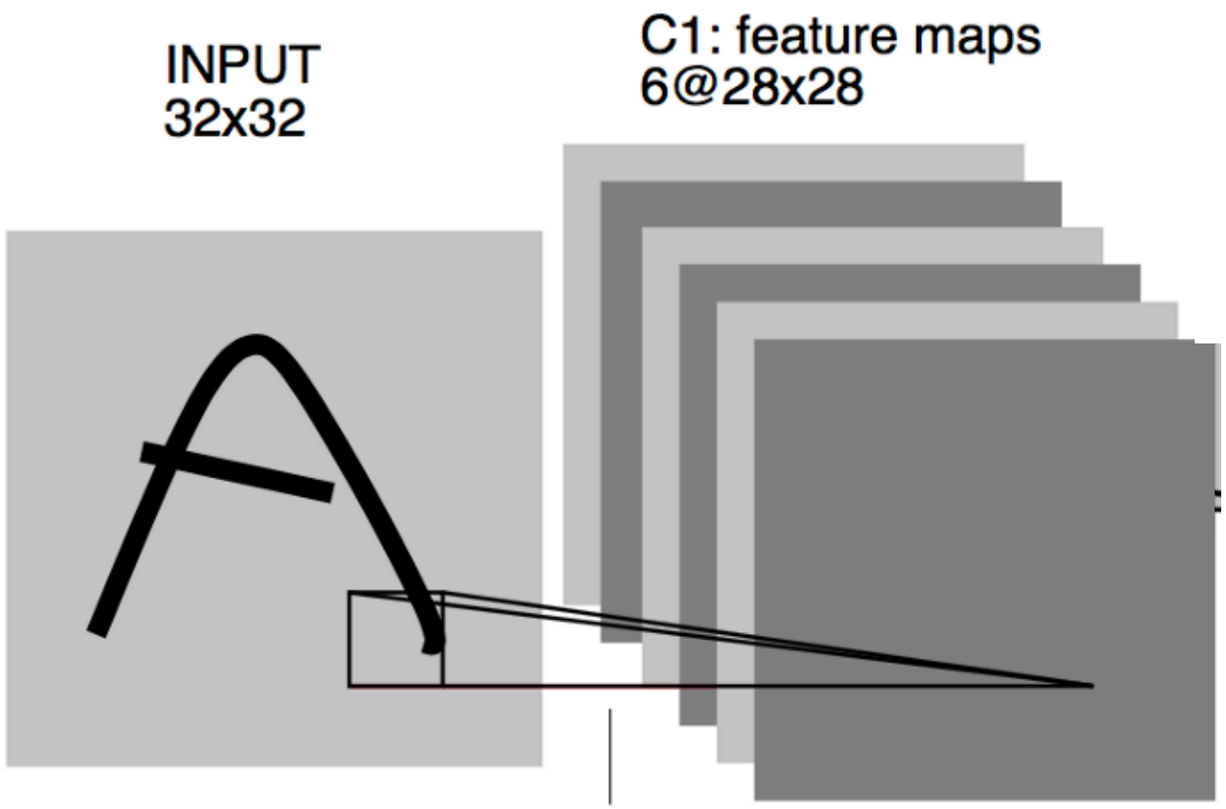
$$W_1 = ?$$

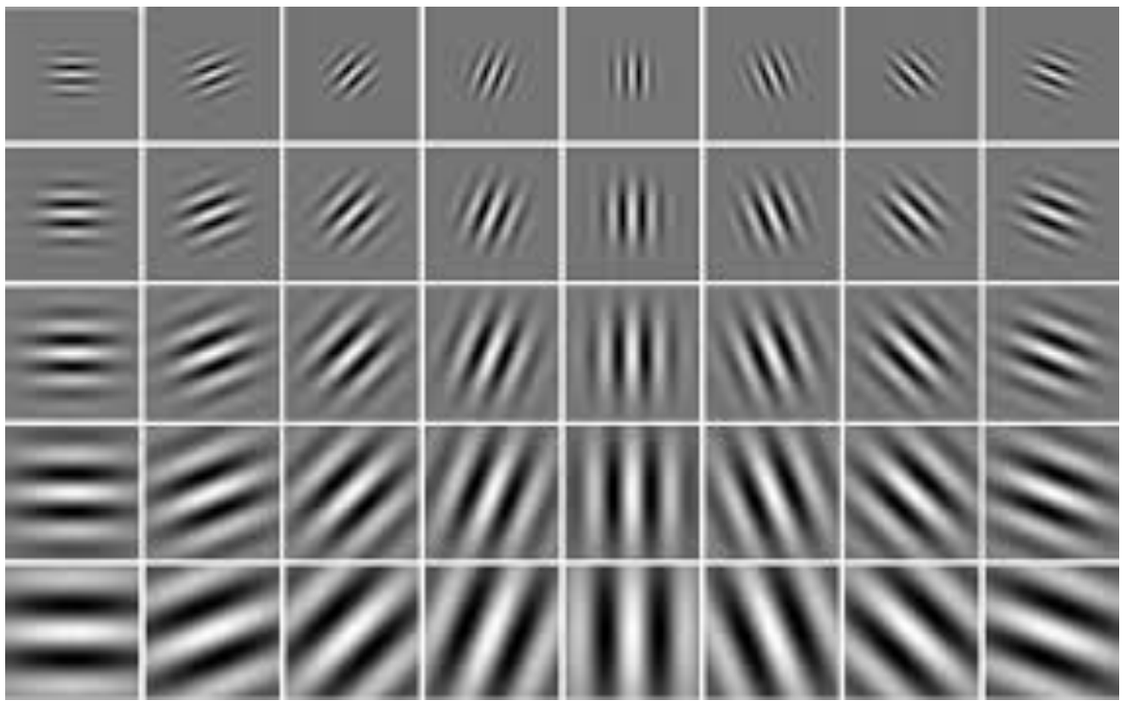
Conv Layer



$$W_1 = ?$$

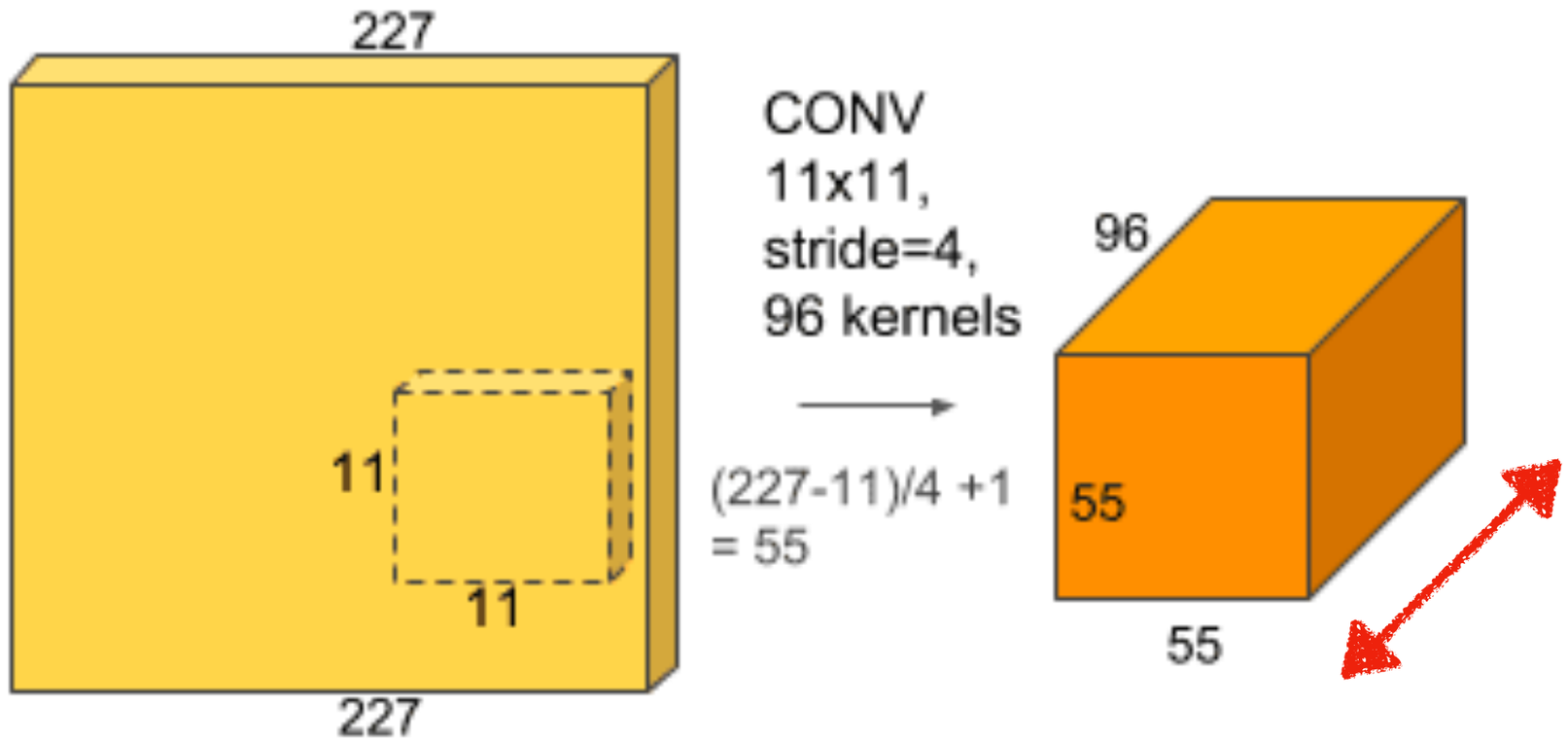
Feature Maps





Convolution Layer

Learnable weights
96 Filters x 3 x 11 x 11

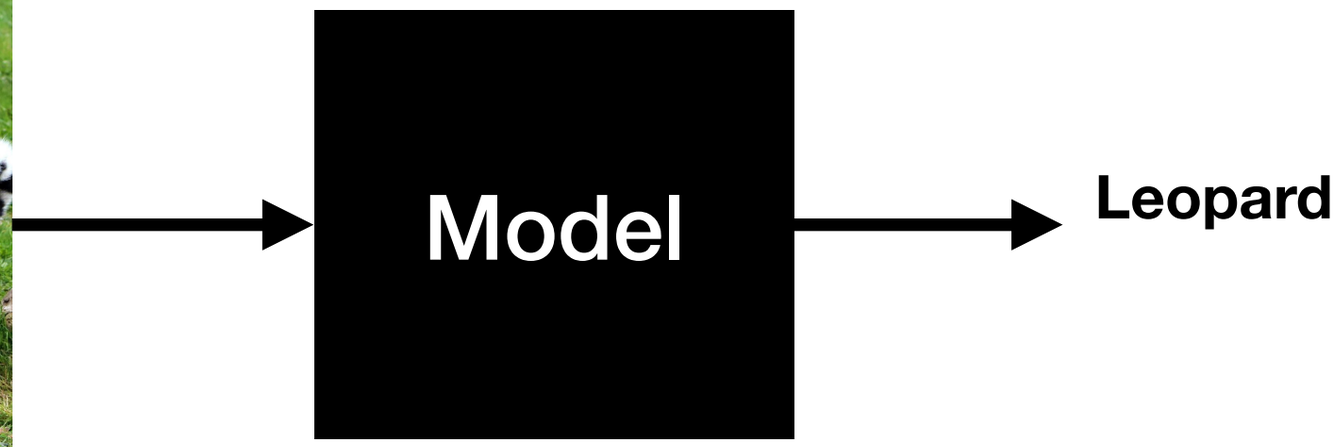


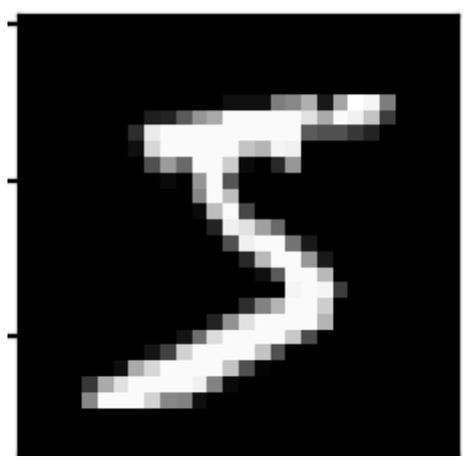
Initial H x Initial W x 3

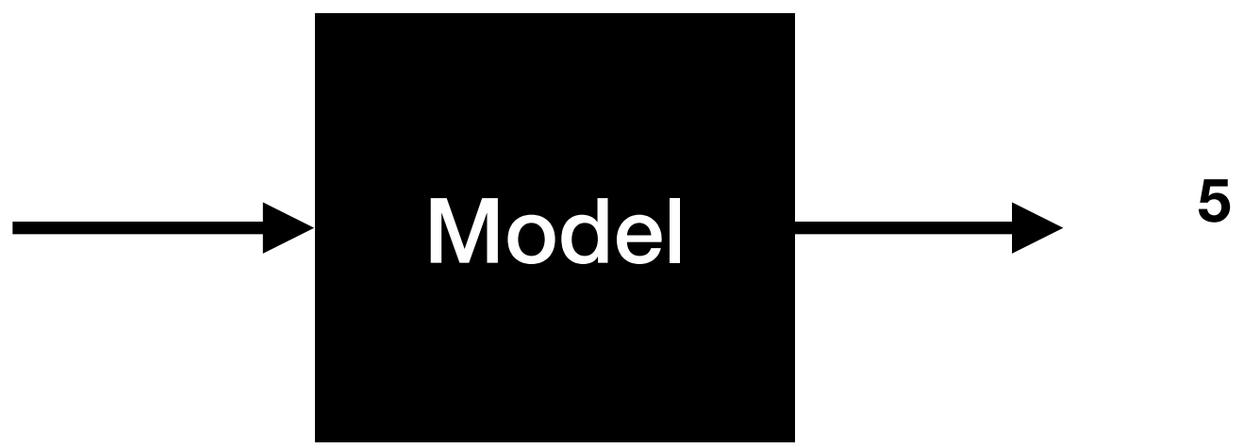
Feature Maps

H1 x W1 x Feature Maps

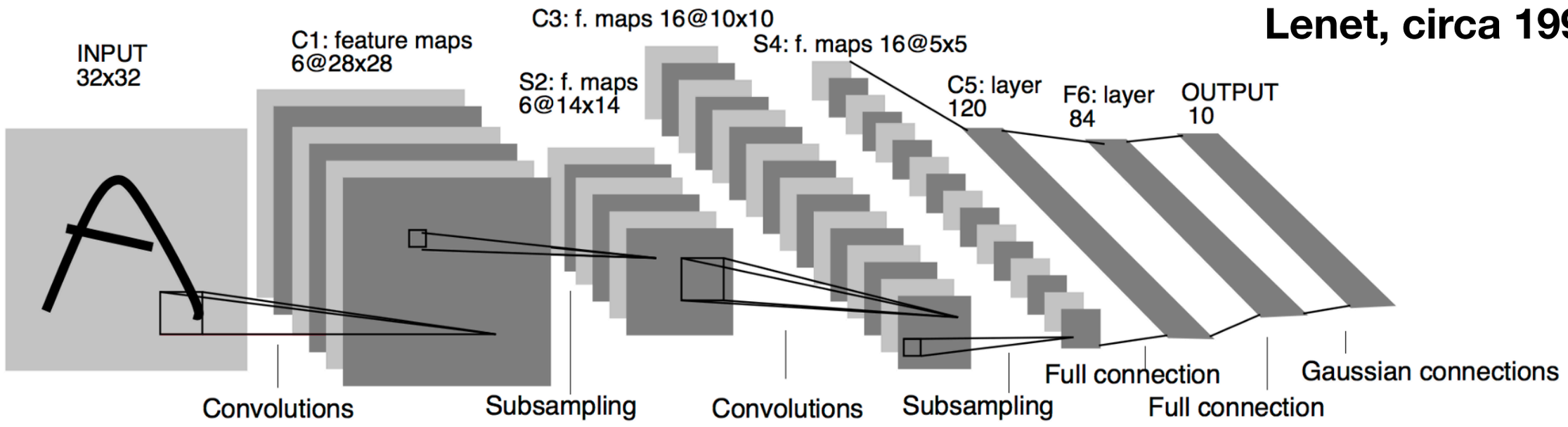
Image Classification







Convolutional Networks



- Key components: Convolution layers, non-linearity, pooling
- Conv + non-linearity is sometimes seen as detecting features
- Pooling
  - Further creates spatial invariance
  - Used to control layer output size (critical for efficiency)
  - Max and avg pooling is often used

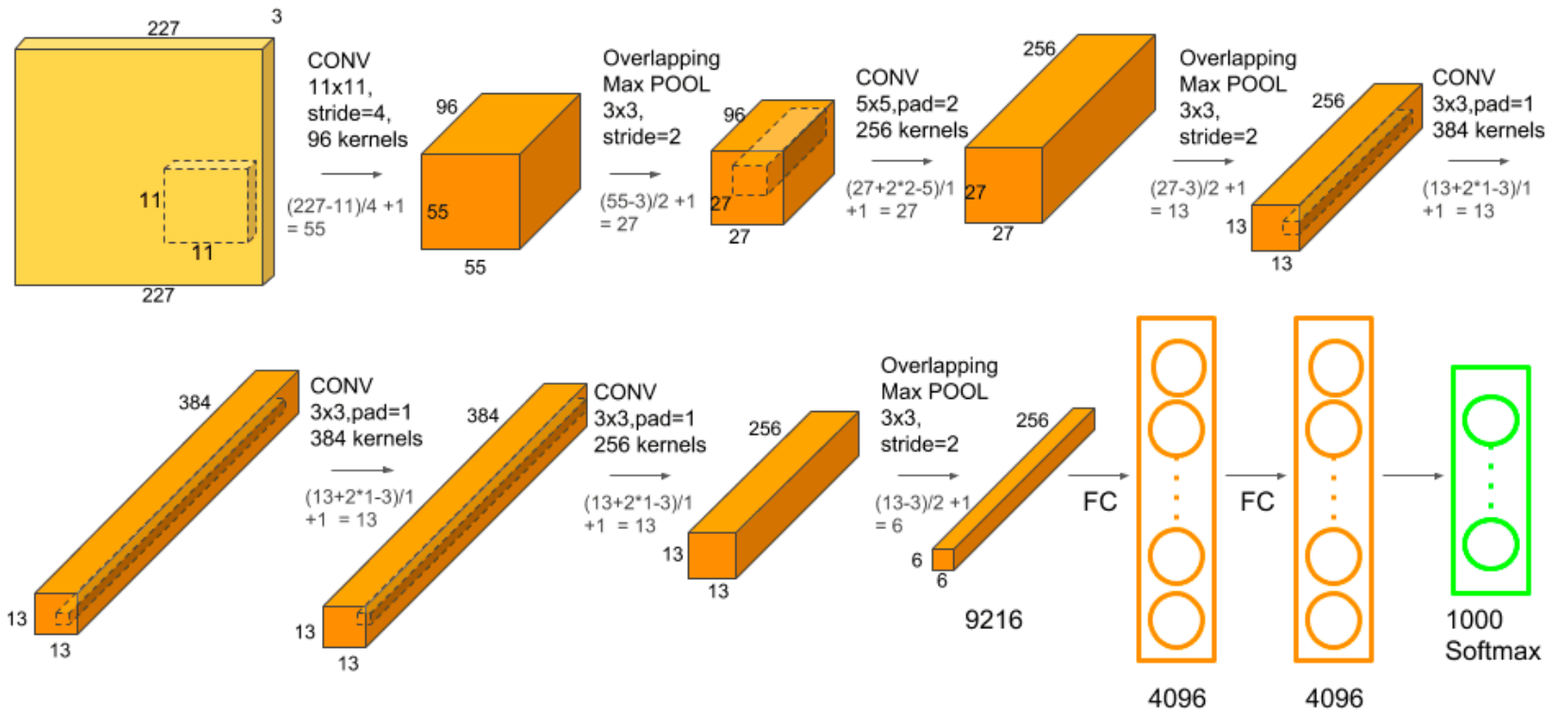
Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Convolutional Networks for Object Classification

ImageNet Classification with Deep Convolutional Neural ...

by A Krizhevsky · Cited by 76402 · Related articles



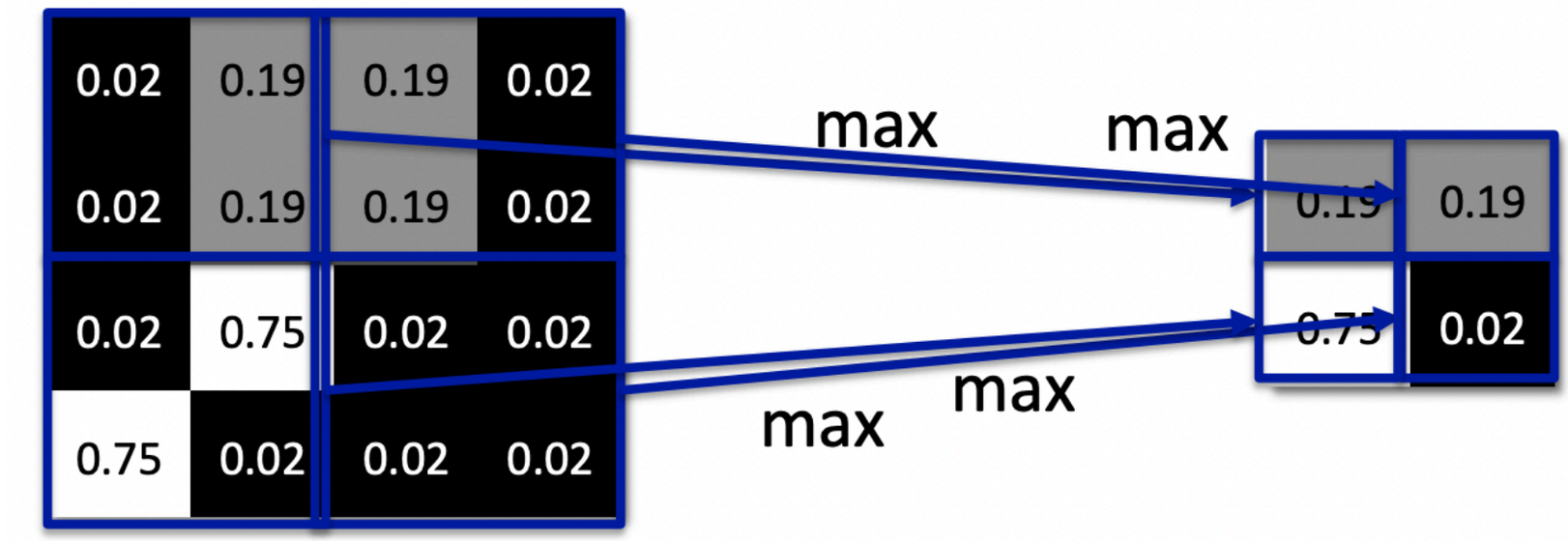
Team name	Filename	Error (5 guesses)
SuperVision	test-preds-141-146.2009-131-137-145-146.2011-145f.	0.15315
SuperVision	test-preds-131-137-145-135-145f.txt	0.16422
ISI	pred_FVs_wLACs_weighted.txt	0.26172
ISI	pred_FVs_weighted.txt	0.26602

- Key result that popularized deep learning in 2010s
- Imagenet has 1.2 million images (before data augmentation) and 1000 object classes - first model trained on such scale
- LeNet model with a few tweaks and trained on GPU
- CUDA optimized convolutions allowed scalable learning
- Emergent properties of trained model lead to new insights

Pooling Layers

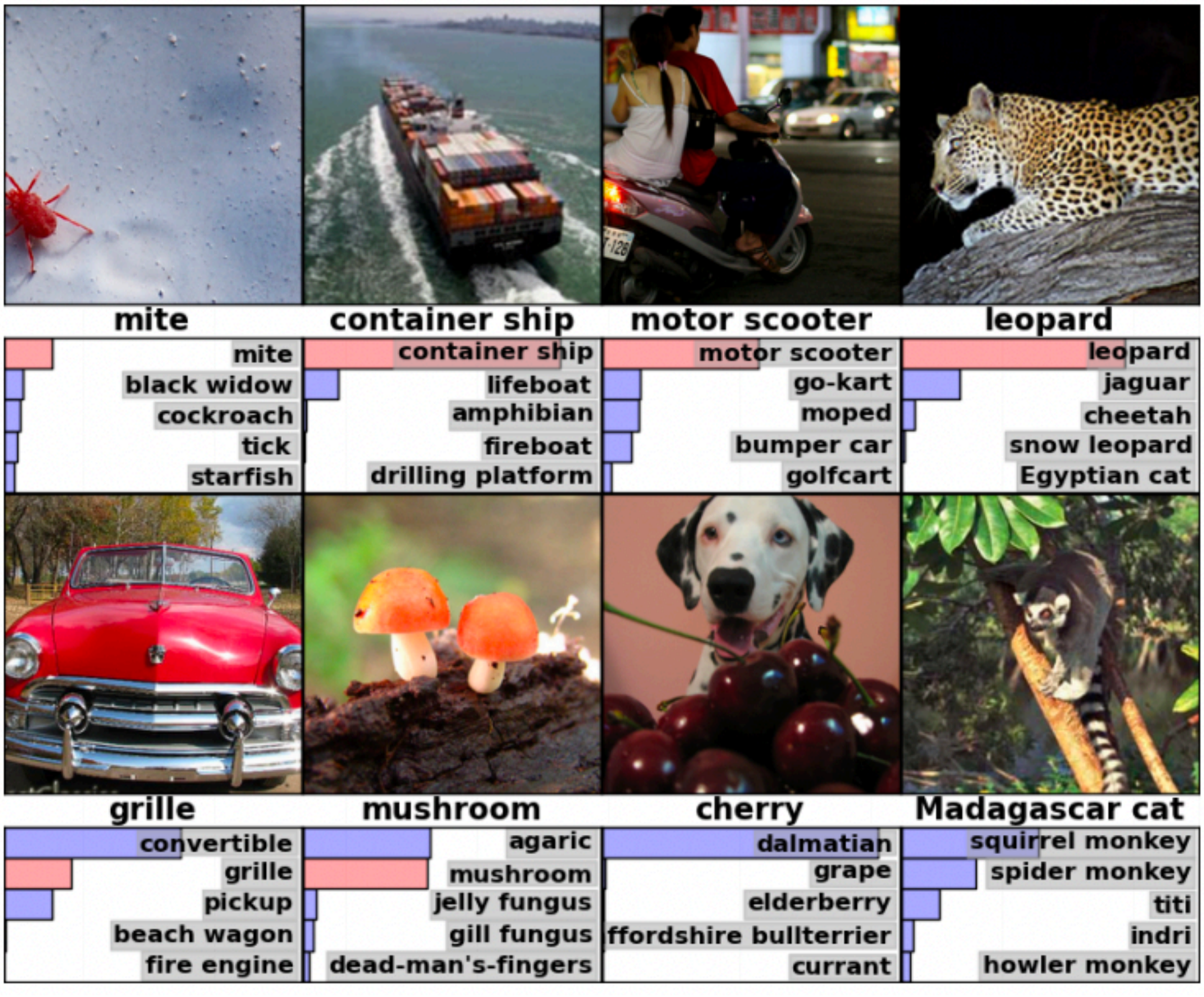
Increase invariance

Image Credit: Hugo Larochelle



Max or Avg Pooling is commonly used

Convolutional Networks



ImageNet Classification with Deep Convolutional Neural ...

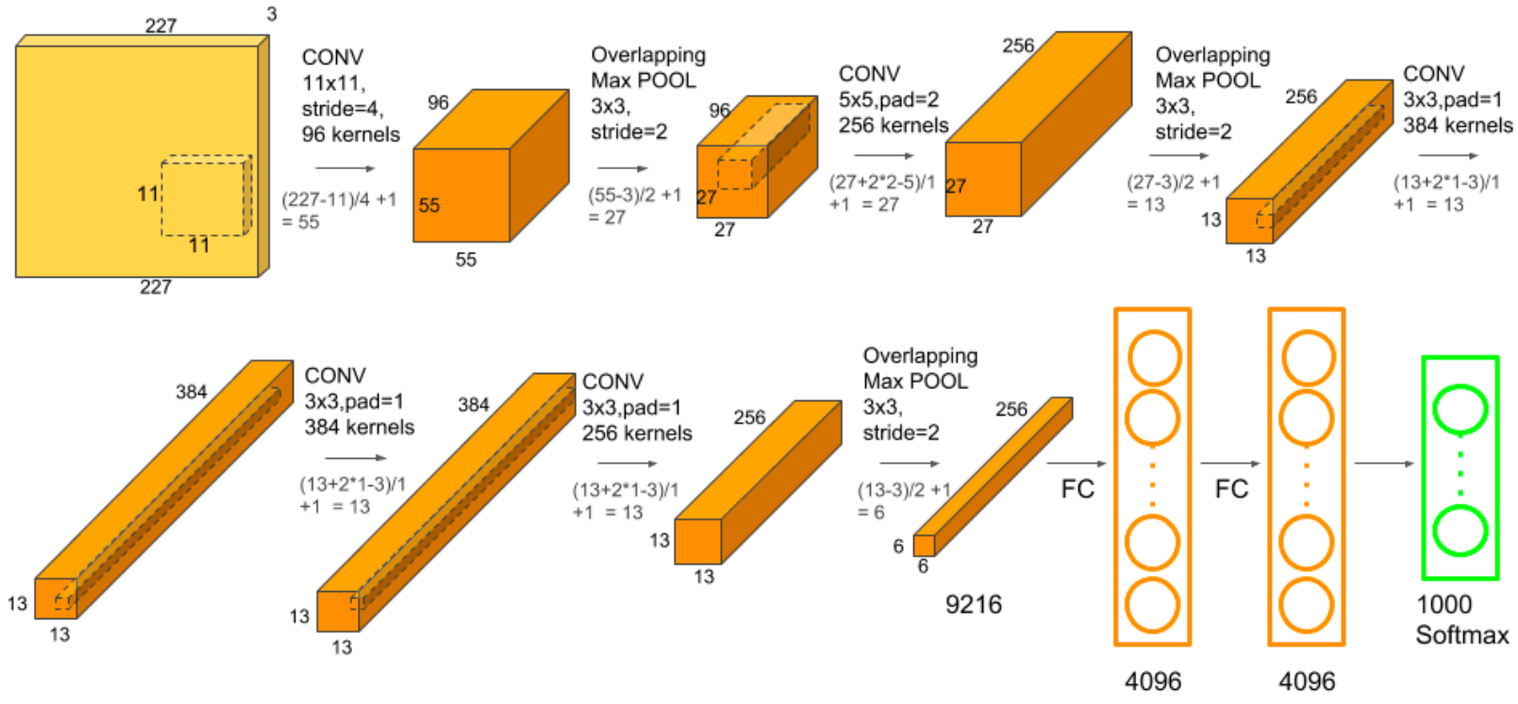
by A Krizhevsky · Cited by 76402 · Related articles

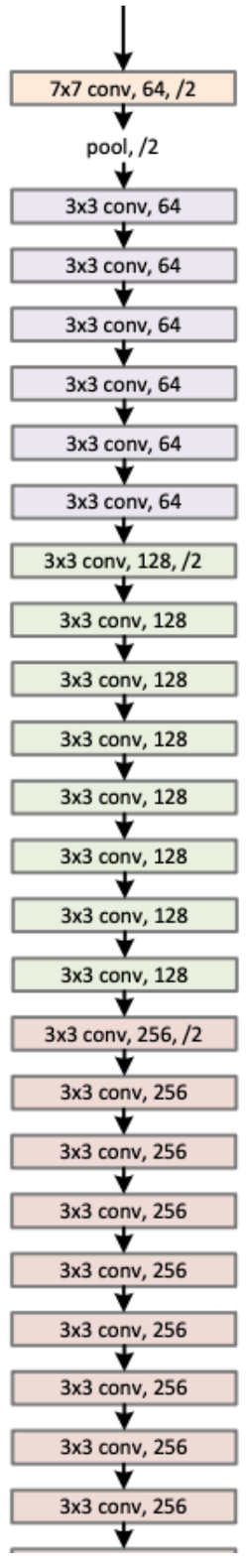
Convolutional Networks Advantages

- Exploits local structure of problems
  - Builds Equivariant representations
- Parameter sharing — parameter efficient
  - Sparse connectivity
- Convolution operation can be heavily optimized

Pooling Layers

Keep computation from increasing while increasing number of features





Efficiency of Convolution

Input size: 320 by 280

Kernel size: 2 by 1

Output size: 319 by 280

	Convolution	Dense matrix	Sparse matrix
Stored floats	2	319*280*320*280 > 8e9	2*319*280 = 178,640
Float muls or adds	319*280*3 = 267,960	> 16e9	Same as convolution (267,960)

(Goodfellow 2016)

CNN models are typically much smaller in terms of parameters

Compared to FC layers they can be seen as analogous to efficiency of sparse matrix vs dense

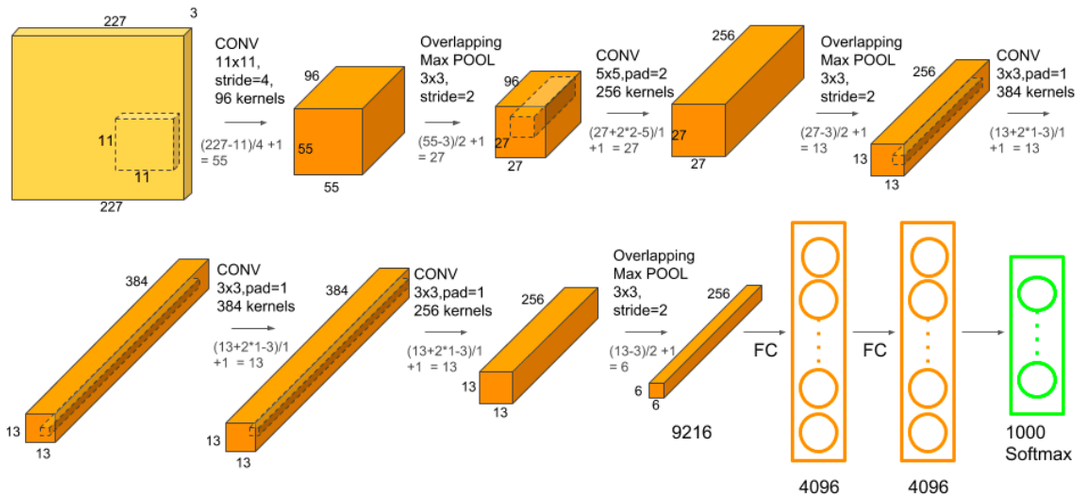
CNN in torch

```python
class AlexNet(nn.Module):

    def __init__(self, num_classes: int = 1000) -> None:
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, num_classes),
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.features(x)
        x = self.avgpool(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)
        return x
```

```python
nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2)
```



CNN in torch

```
class AlexNet(nn.Module):

    def __init__(self, num_classes: int = 1000) -> None:
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1)
```

Import well known models either randomly initialized or with existing weights

```
import torch
mini_batch=32
input_features=3
height=224
width=224
model = torch.hub.load('pytorch/vision:v0.6.0', 'alexnet', pretrained=True)
toy_train_images = torch.rand(mini_batch, input_features, height, width)
model(toy_train_images).shape
```

```
Using cache found in /root/.cache/torch/hub/pytorch_vision_v0.6.0
torch.Size([32, 1000])
```

Convolution Math

```
nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2)
```

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

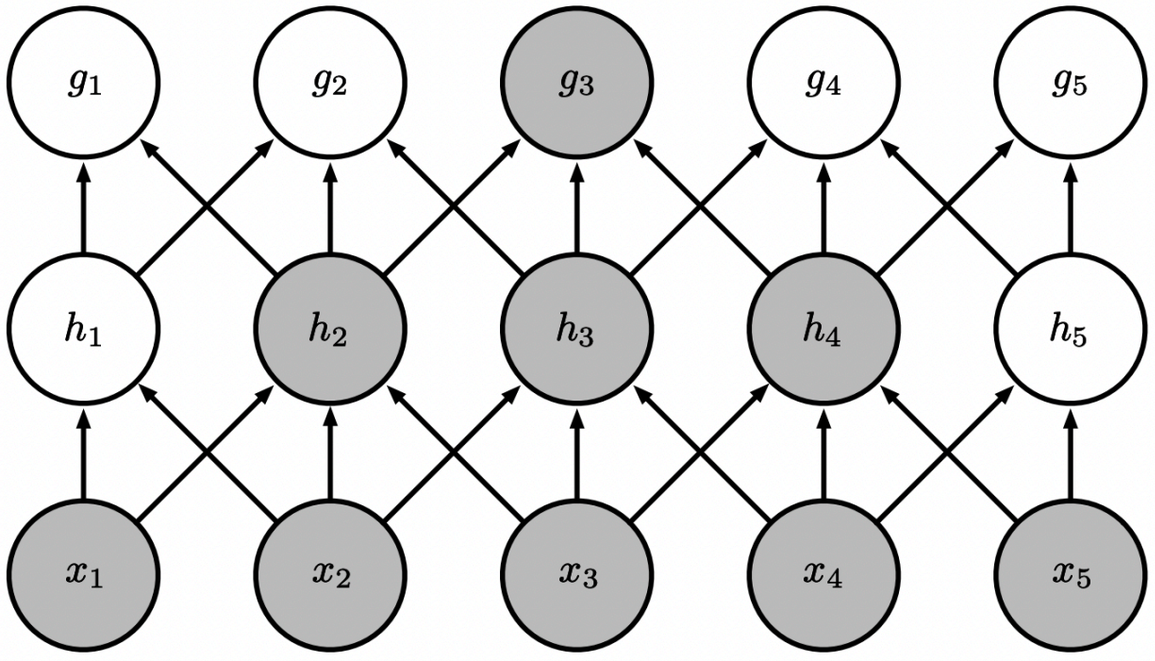
Implementation in GPU

- Highly optimized CUDA kernels have allowed convolutional networks to scale to big datasets

- Alexnet - cuda kernels

- Typical kernels are small thus more effectively implemented in time domain

Receptive field



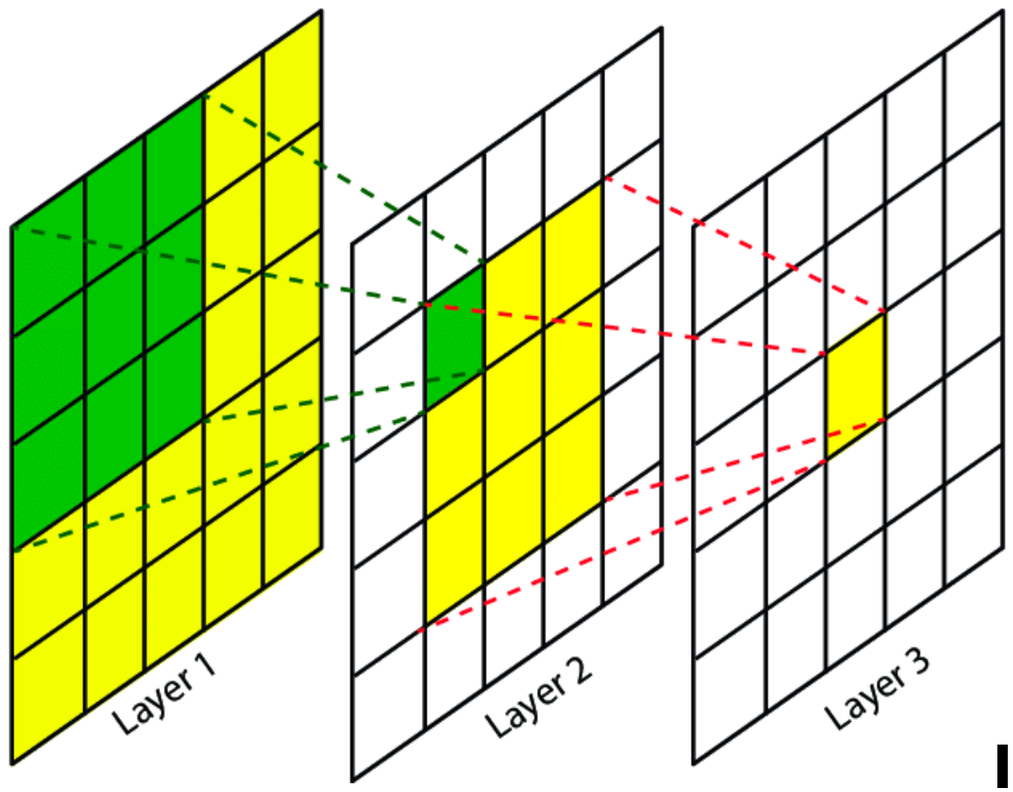


Image from Haoining Lin

CNN as Building Invariants

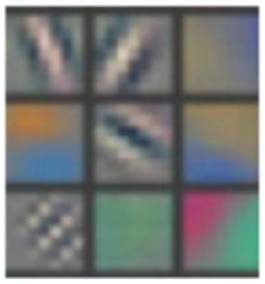
Optional

- Data has symmetries which can be exploited to tackle high dimensional data
  - Simple - translation, rotation, scale
  - More complex symmetries exist but hard to characterize
- One view of CNN is progressively building invariants to complex symmetries

Understanding deep convolutional networks

Stéphane Mallat

What Do CNNs learn

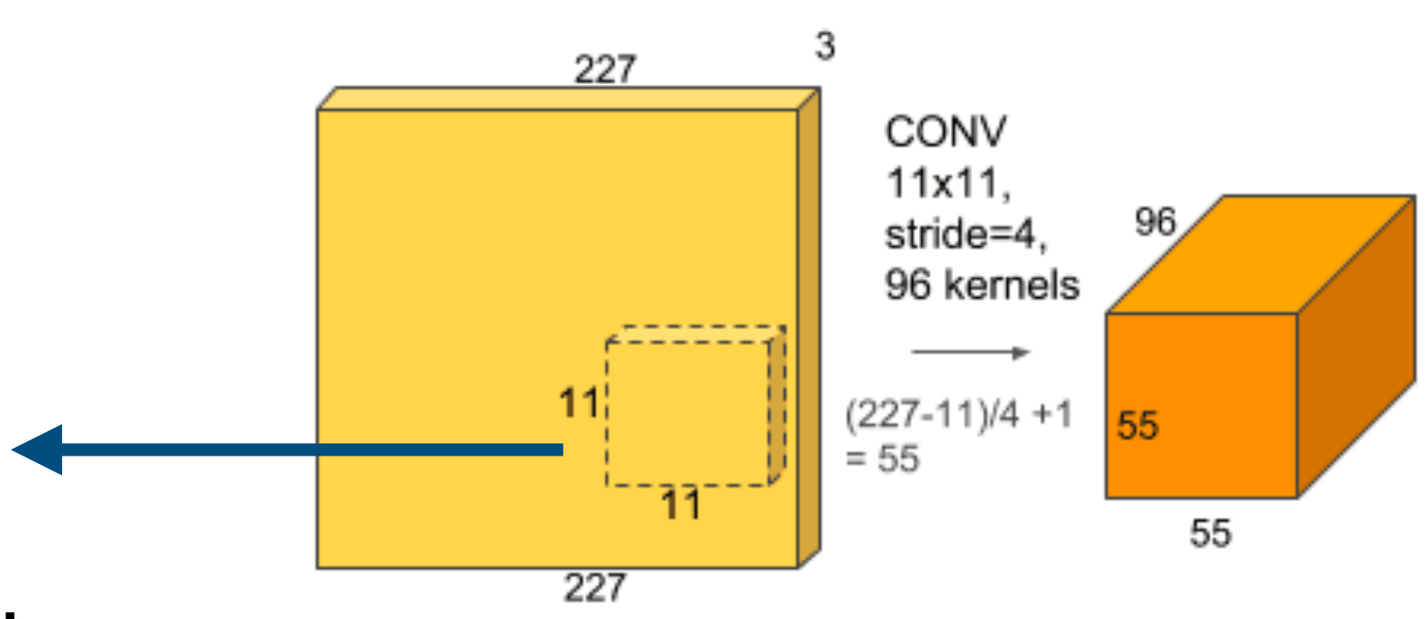


Learnable weights
- 96 Filters

Layer 1

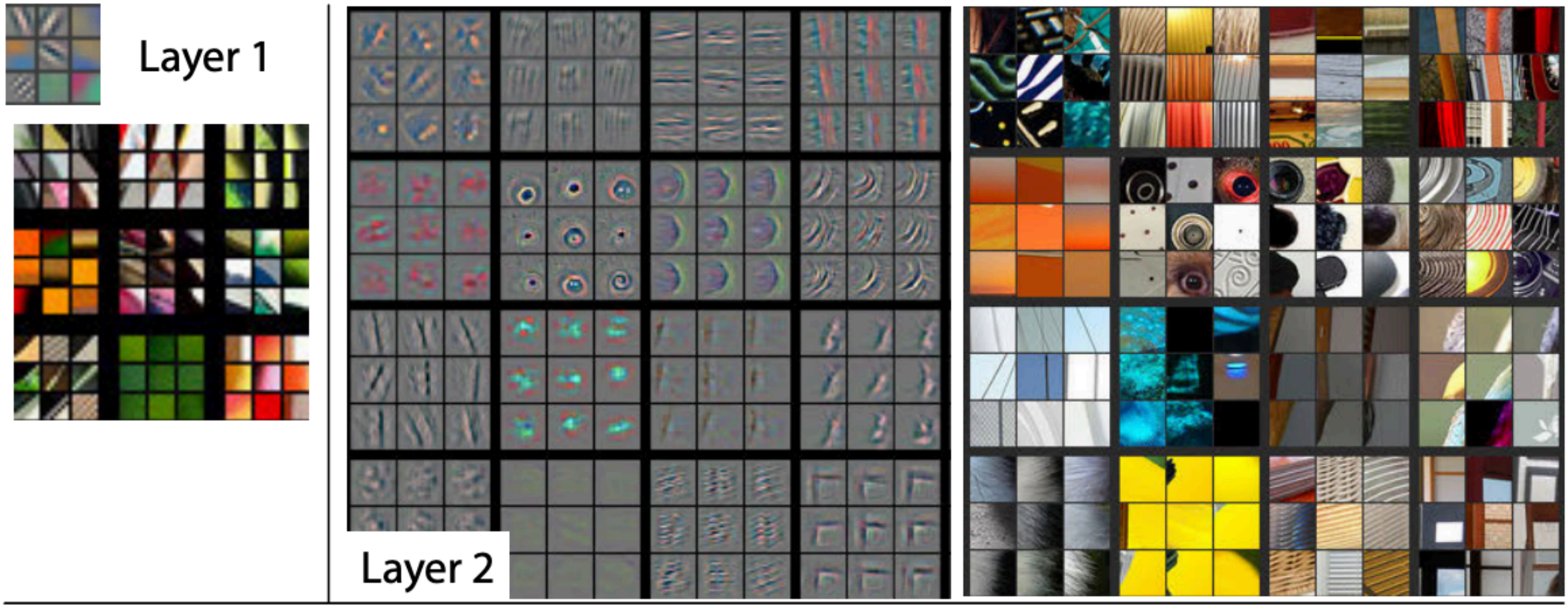
Handcrafted filters

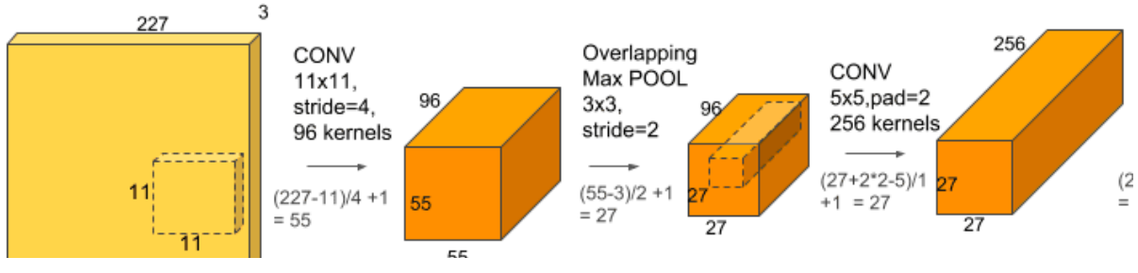
Image Patches Most Activated



Zeiler and Fergus

What Do CNNs learn



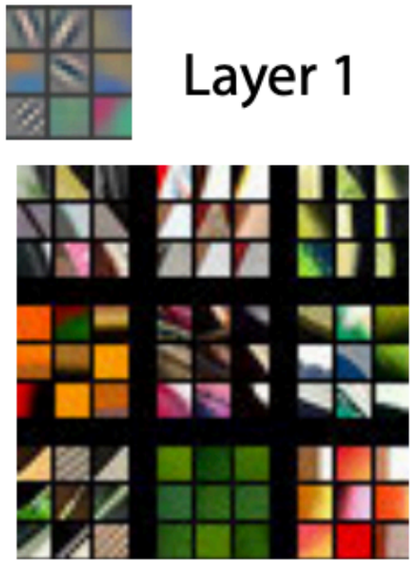


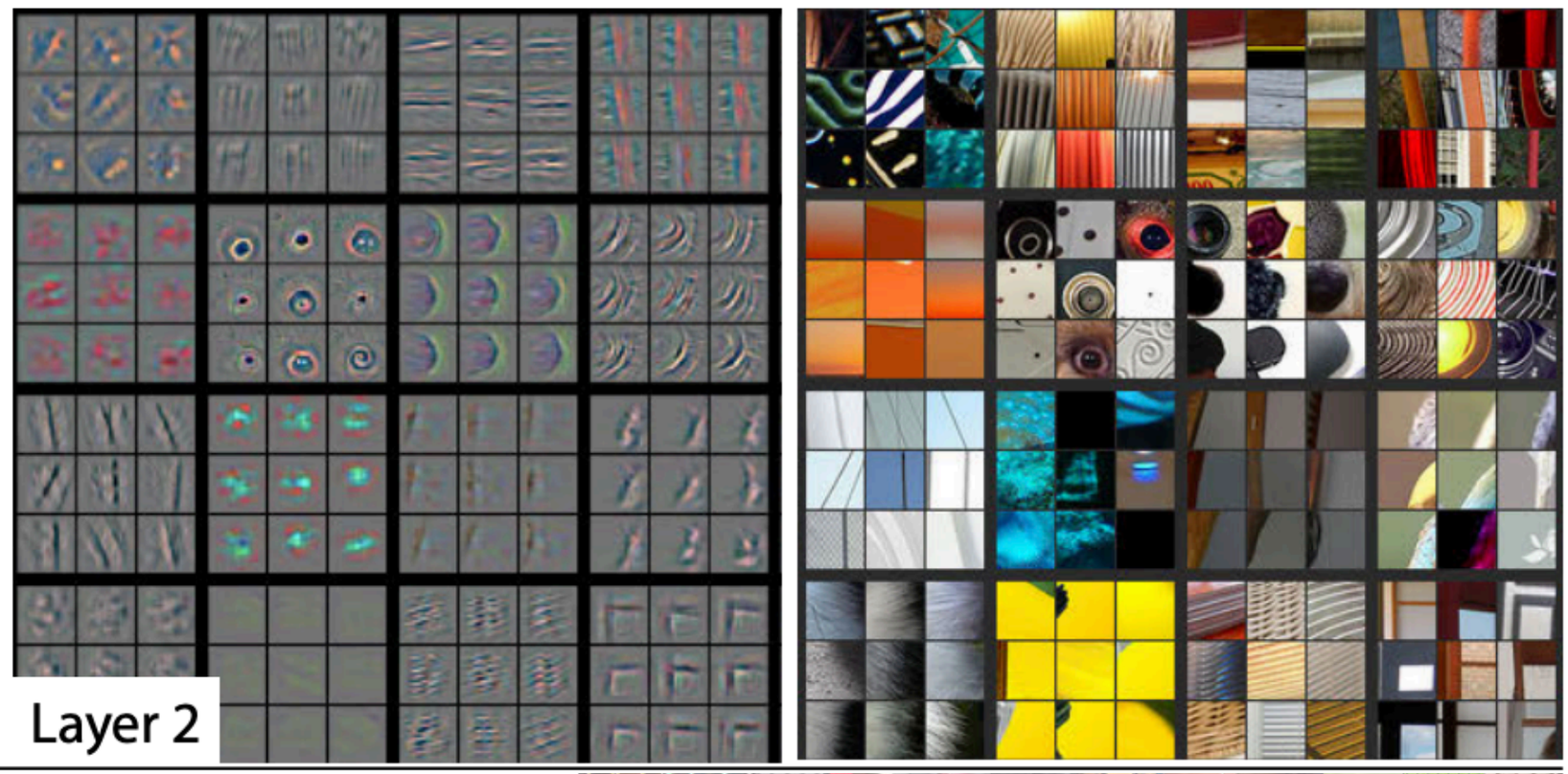
Visualizing and Understanding Convolutional Networks

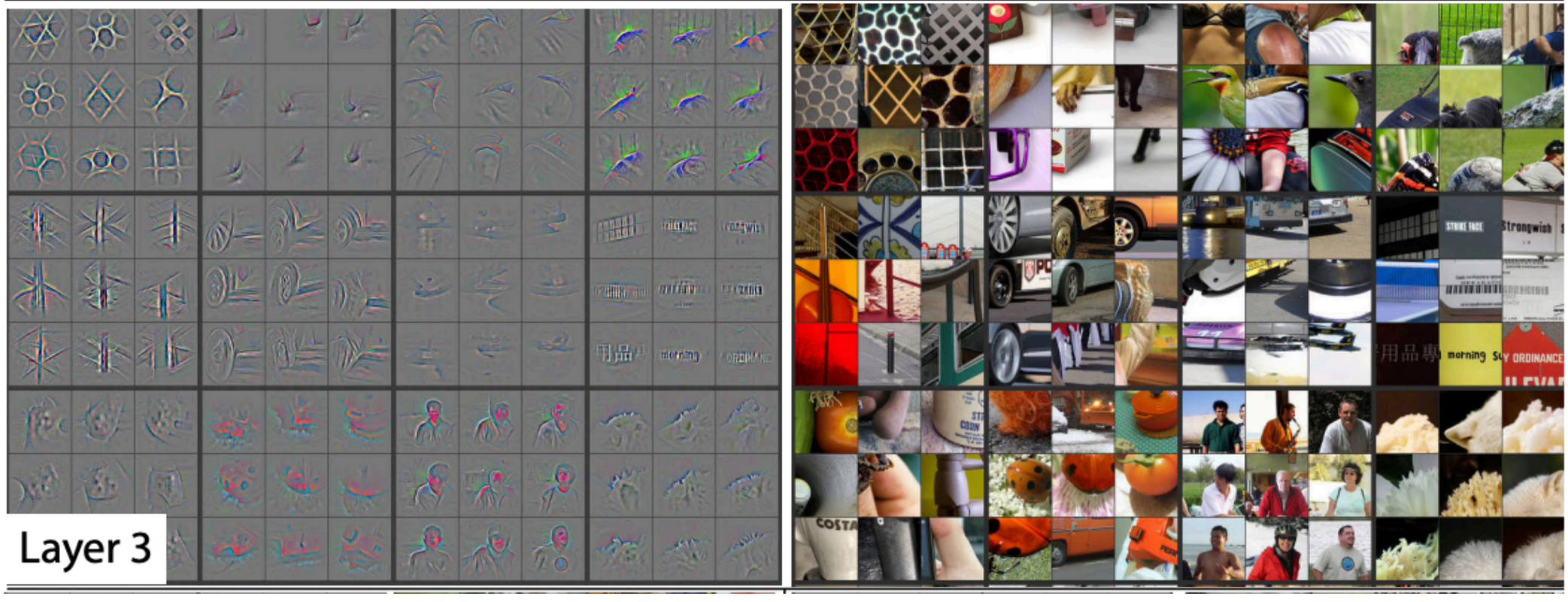
Matthew D. Zeiler ZEILER@CS.NYU.EDU
Dept. of Computer Science, Courant Institute, New York University

Rob Fergus FERGUS@CS.NYU.EDU
Dept. of Computer Science, Courant Institute, New York University

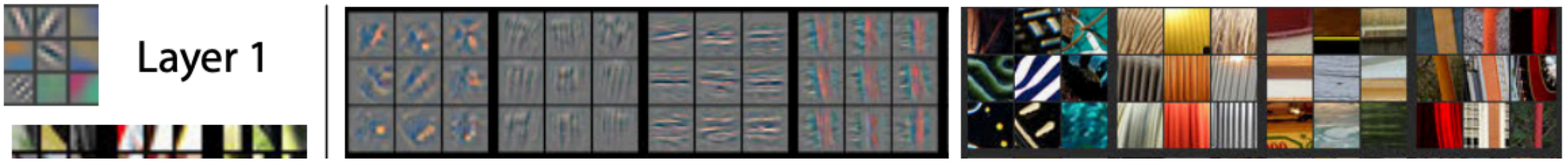
What Do CNNs learn

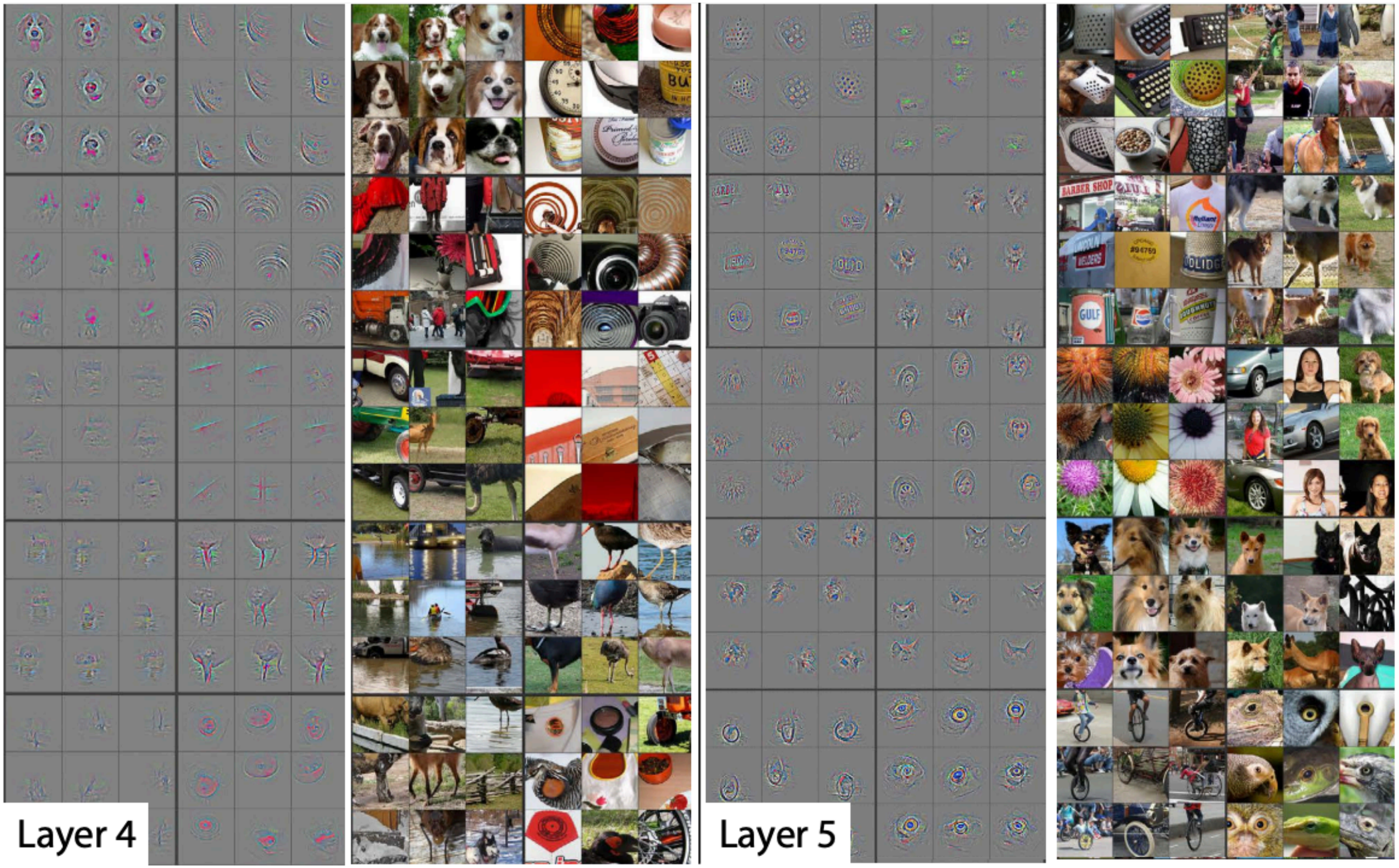






What Do CNNs learn

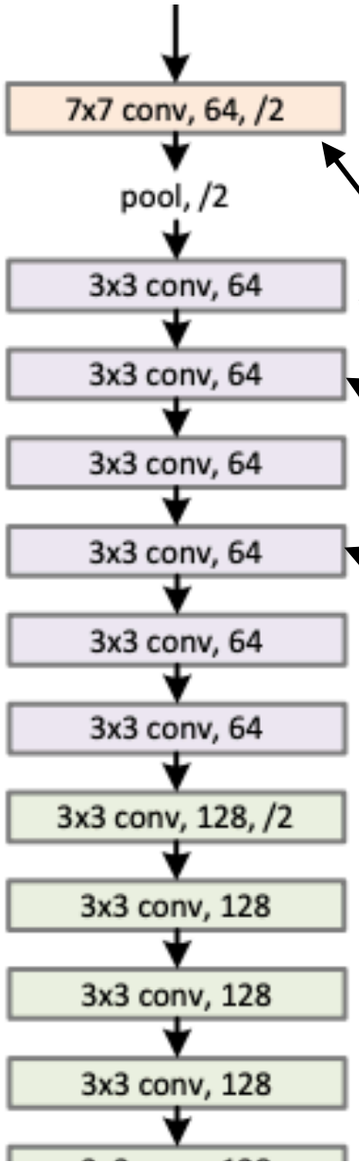




CNN Progressive Linear Separability

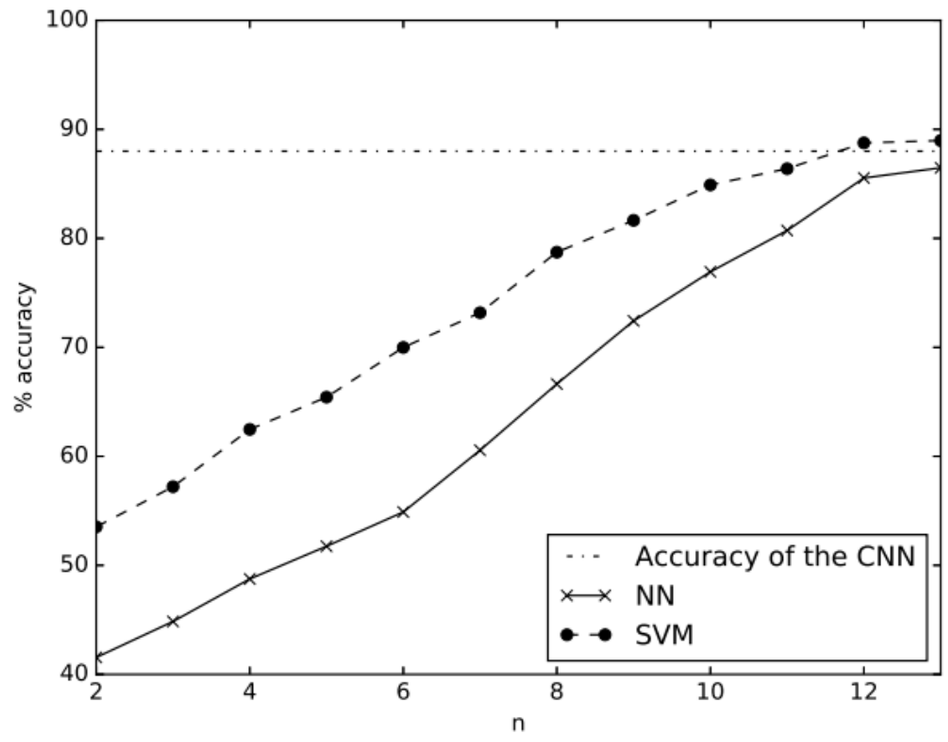
- Supervised Trained CNNs show an emergent property of progressively increasing linear separability of data in target classes

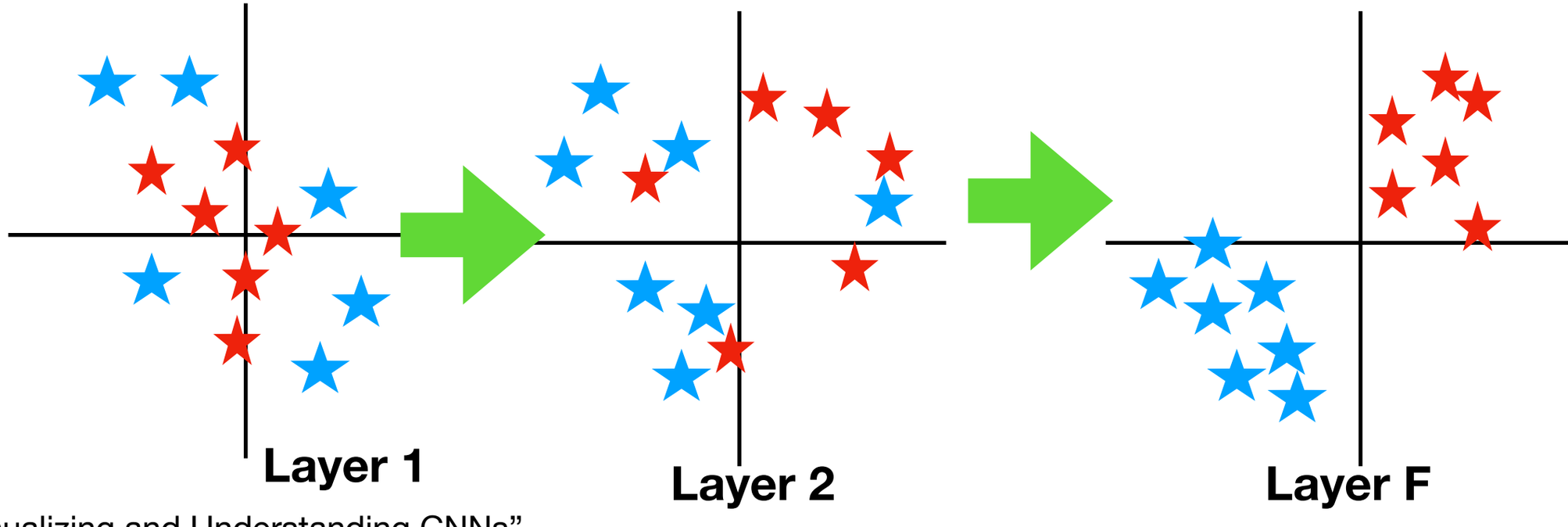
Image



Linear Probe

Nearest Neighbour

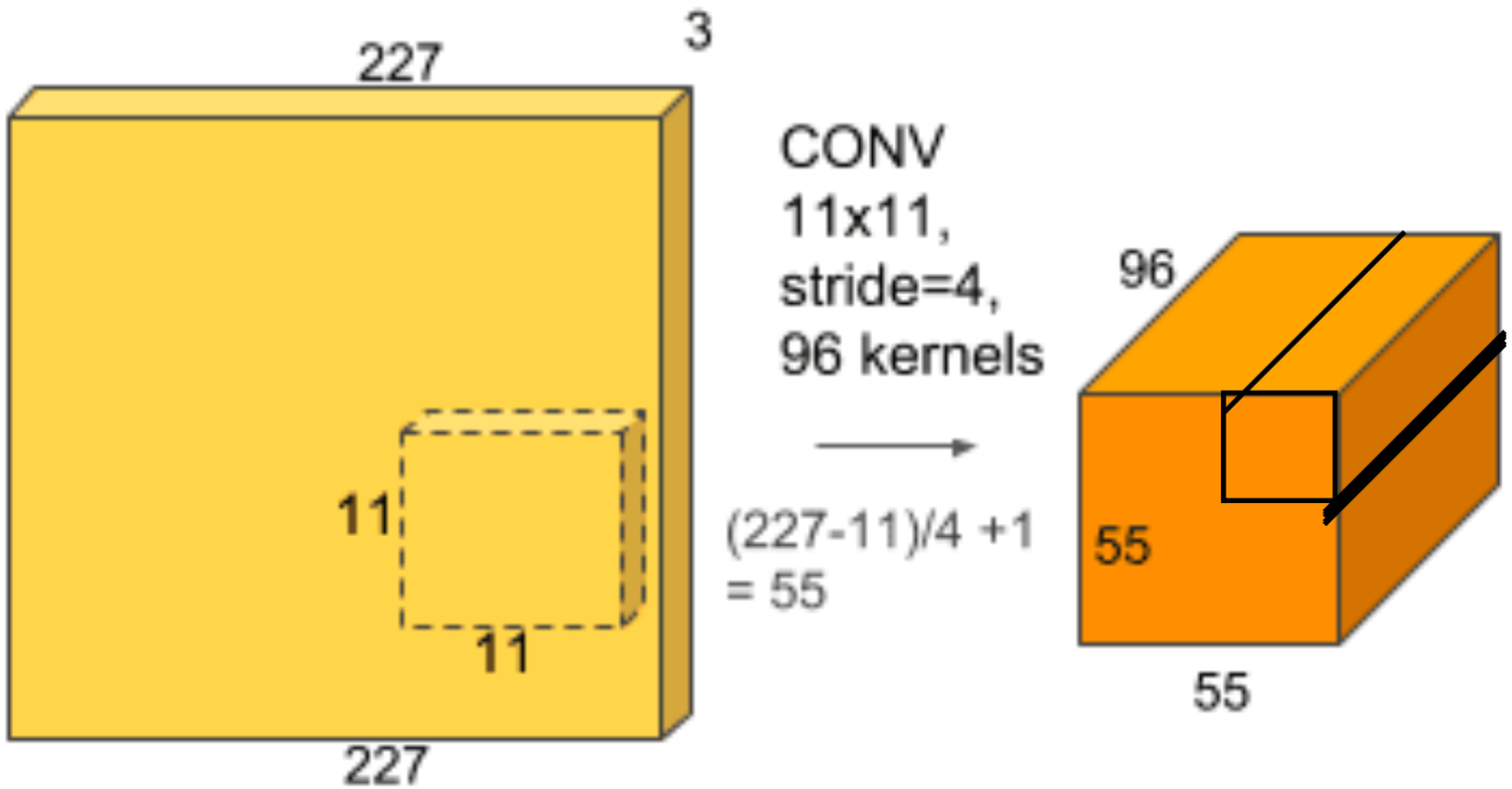




Zeiler et al “Visualizing and Understanding CNNs”
Oyallon “Building a Regular Decision Boundary with Deep Networks”

2D Batchnorm

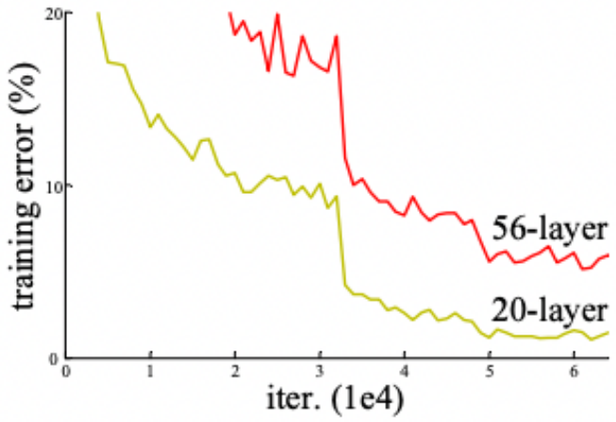
$$\frac{\gamma_i}{\hat{\sigma}_i}(x_i - \hat{\mu}_i) + \beta_i$$



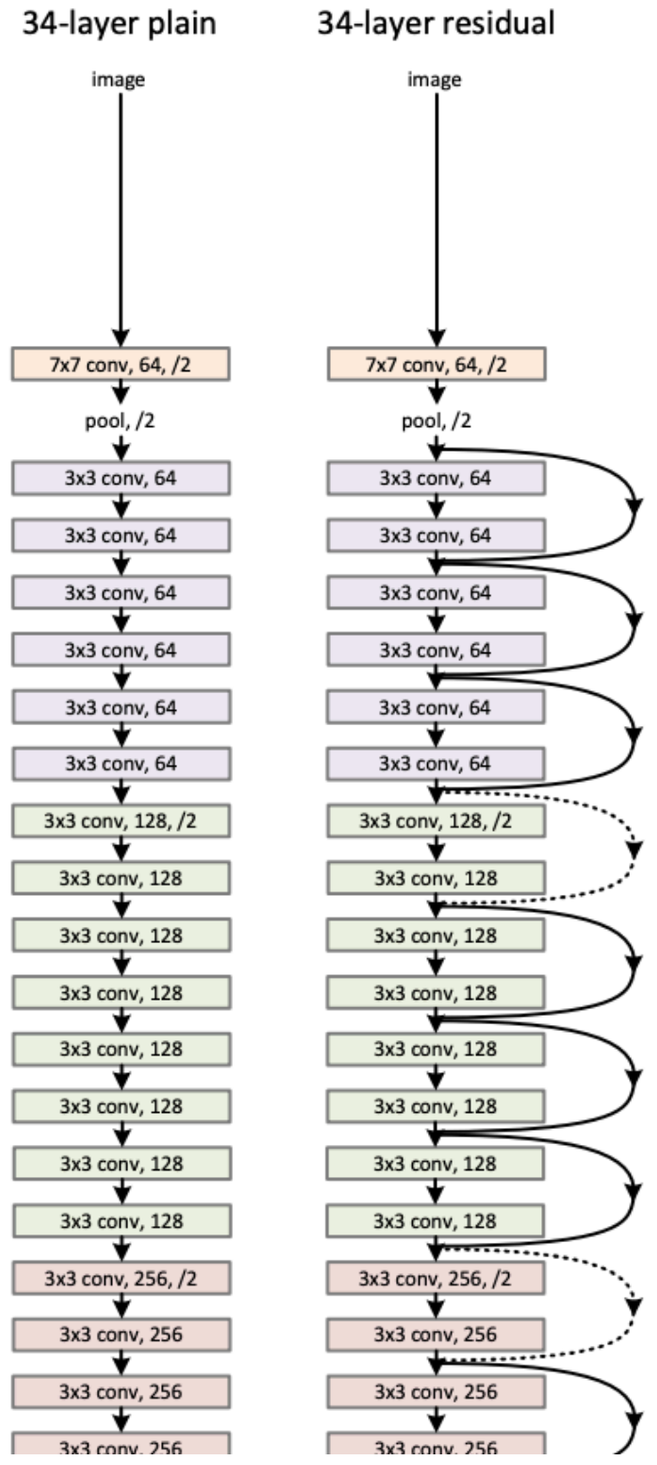
In this case we would keep 96 γ_i, β_i

Compute for minibatch 96 μ_i, σ_i

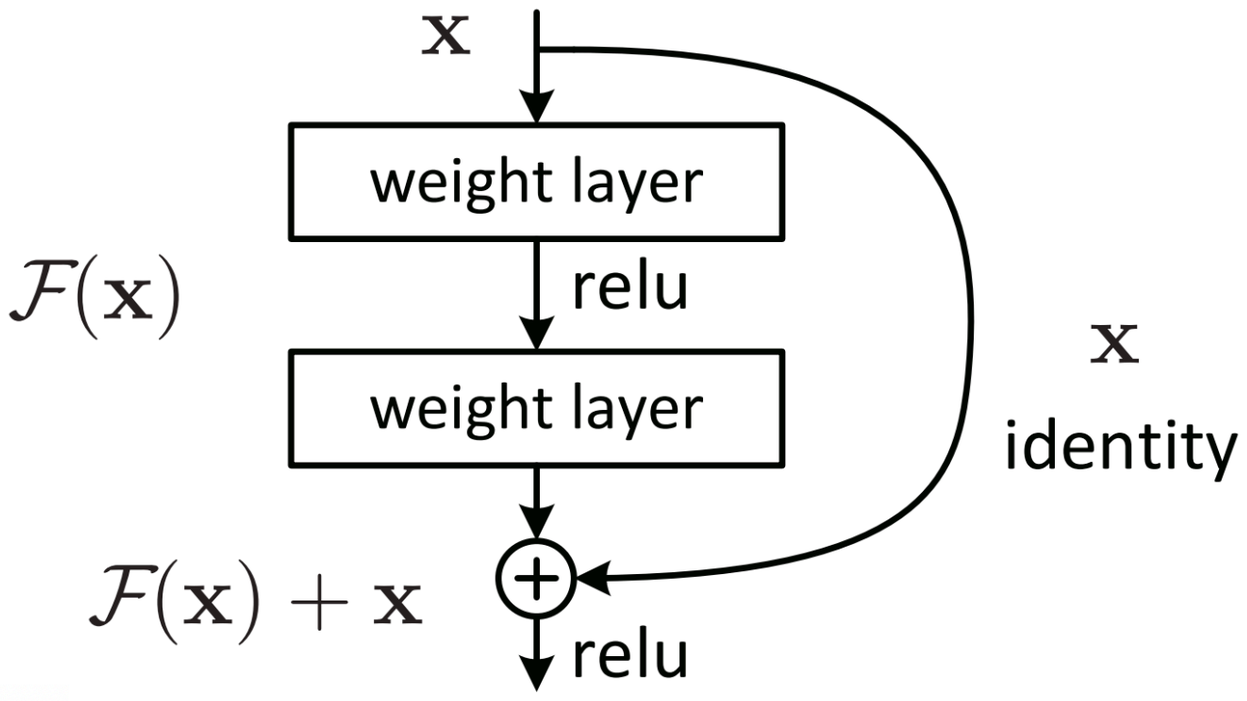
Residual Networks



- Originally introduce to fight the vanishing gradient problem and train deeper



	Err
plain-34	28.54
ResNet-34 A	25.03
ResNet-34 B	24.52
ResNet-34 C	24.19
ResNet-50	22.85
ResNet-101	21.75



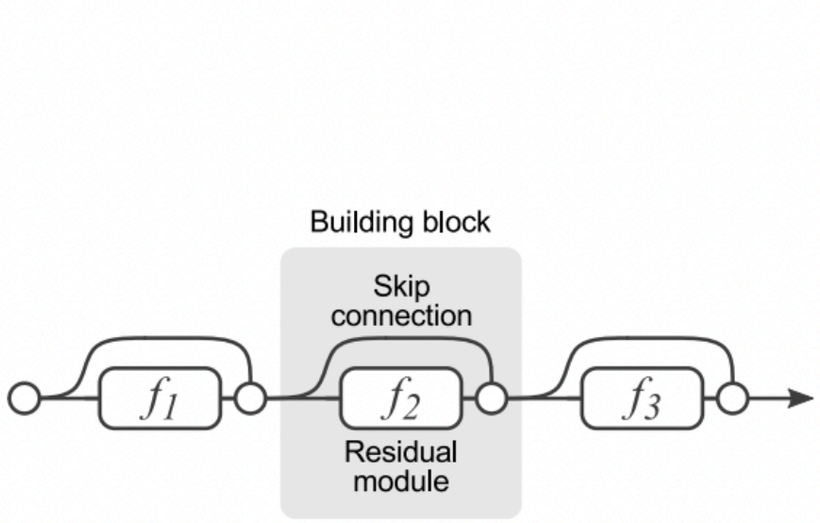
Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

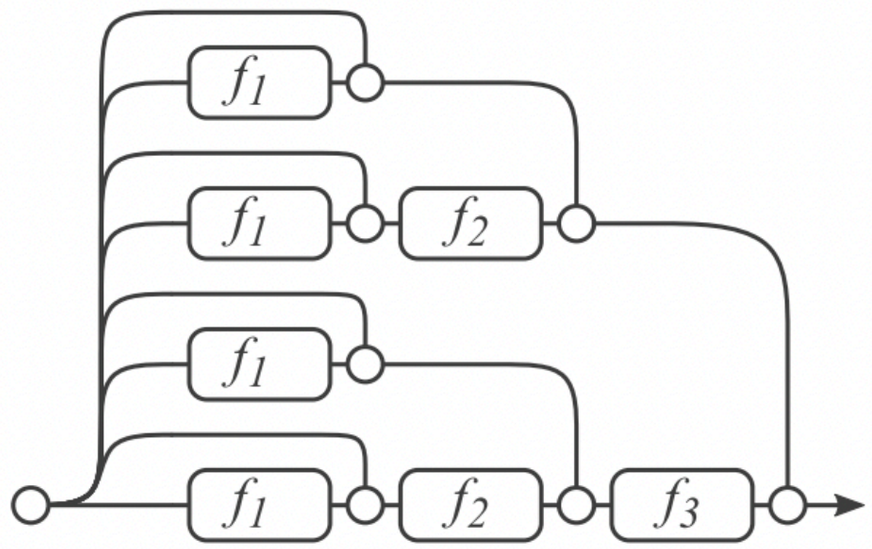
Other Interpretations of ResNets

- Besides allowing gradient flow ResNets are hypothesized to have inductive biases that help generalization

- One view: exponential ensembles



(a) Conventional 3-block residual network



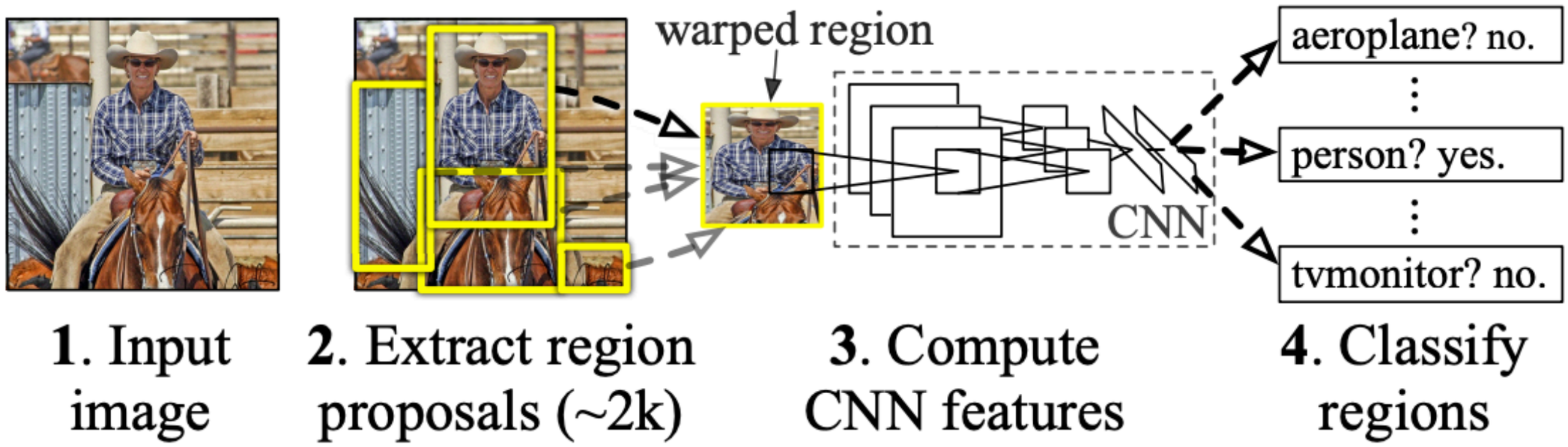
(b) Unraveled view of (a)

Residual Networks Behave Like Ensembles of Relatively Shallow Networks

Andreas Veit **Michael Wilber** **Serge Belongie**
Department of Computer Science & Cornell Tech
Cornell University

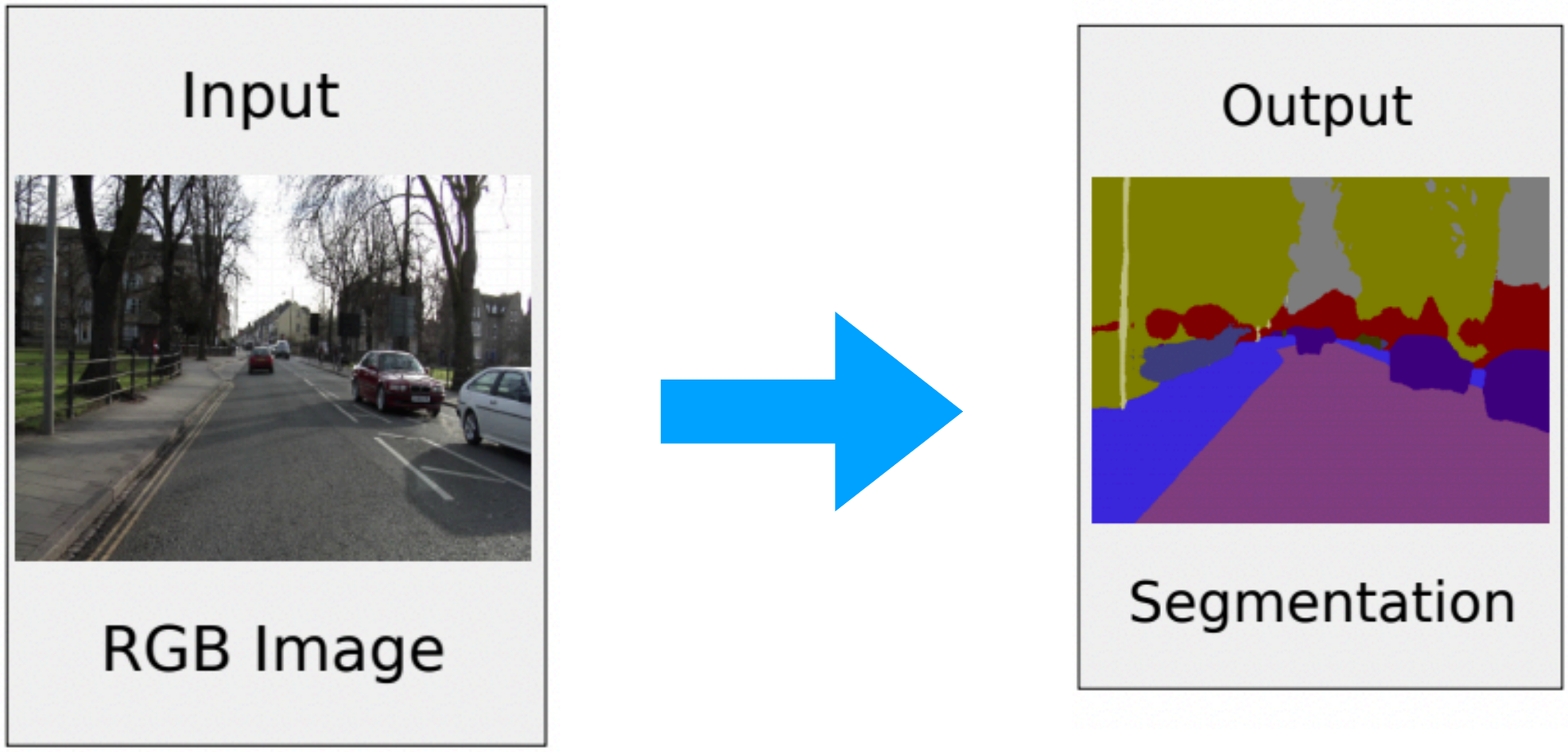
Other Applications of CNNs in Vision

Object Detection



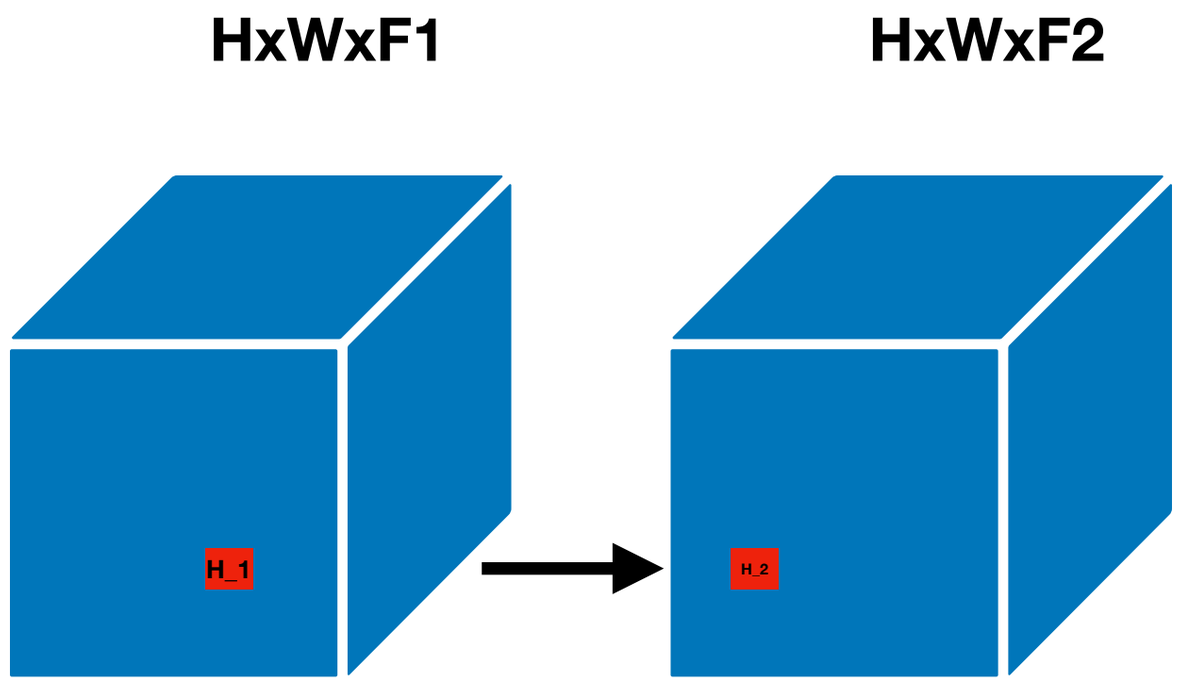
Girschik et al 2013

Semantic Segmentation



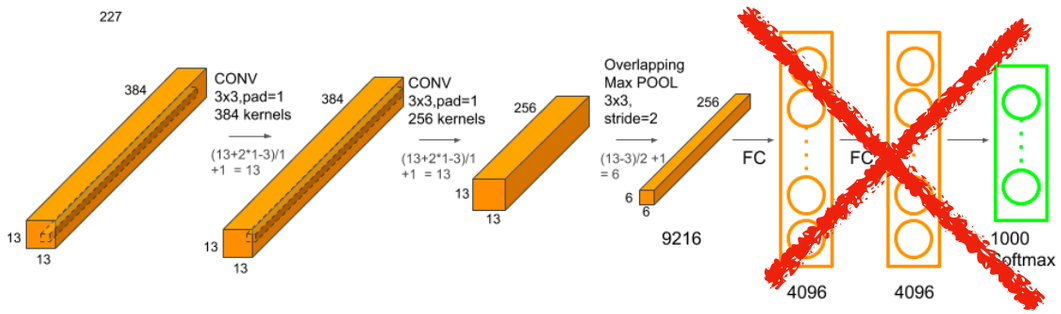
- Multiple output variables
- Each pixel predicts a class

1x1 Convolution

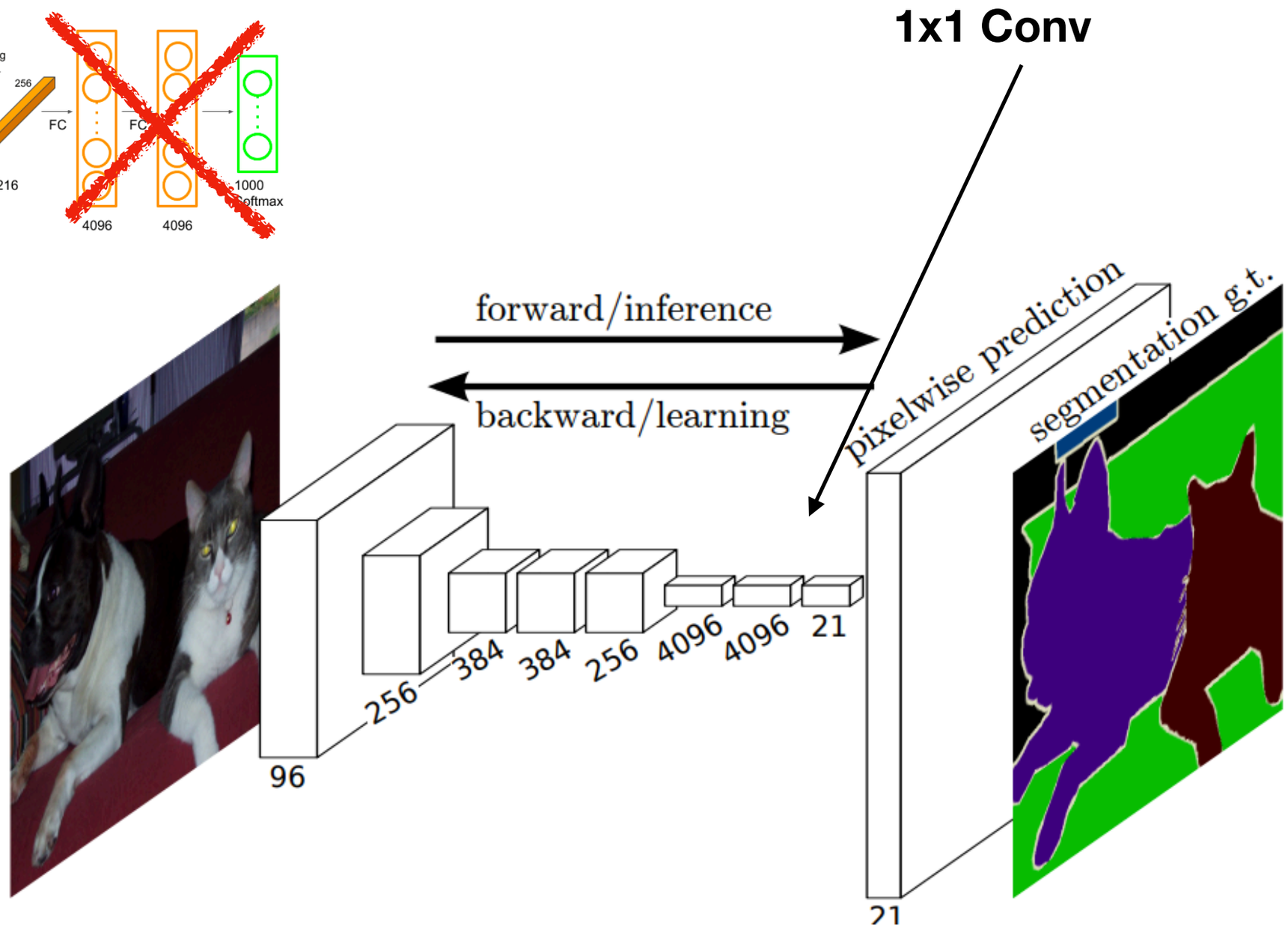


- Each spatial position is applied the same linear operator

Fully Convolutional Networks



AlexNet



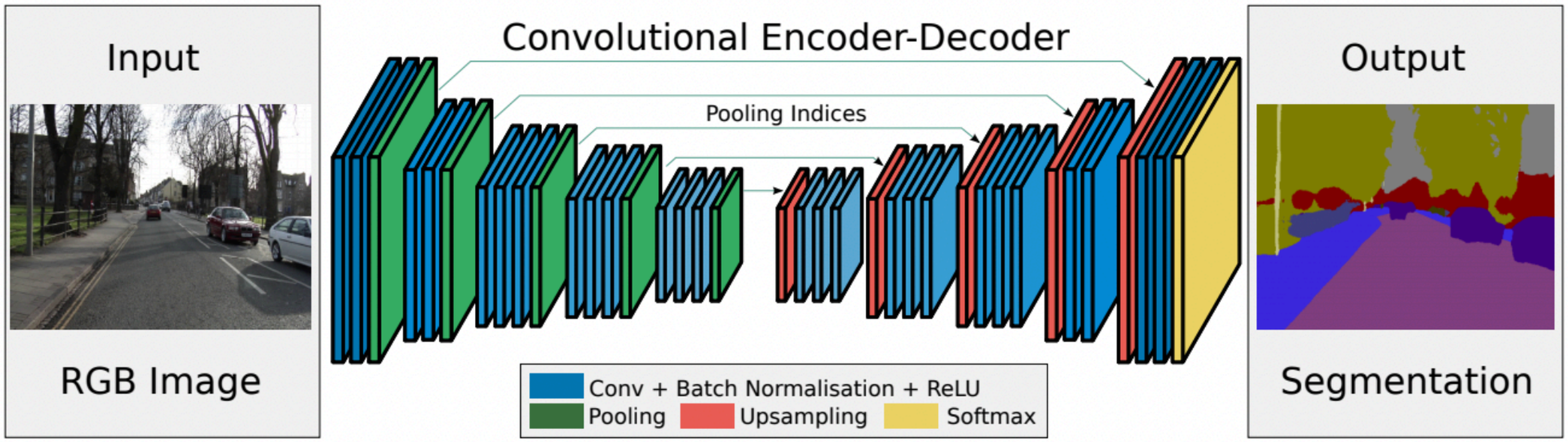
- A 1x1 cone transforms the final output to HxWxO which feeds into a softmax and cross entropy which we sum over all pixels

- We will often need to upsample this to get the full mask

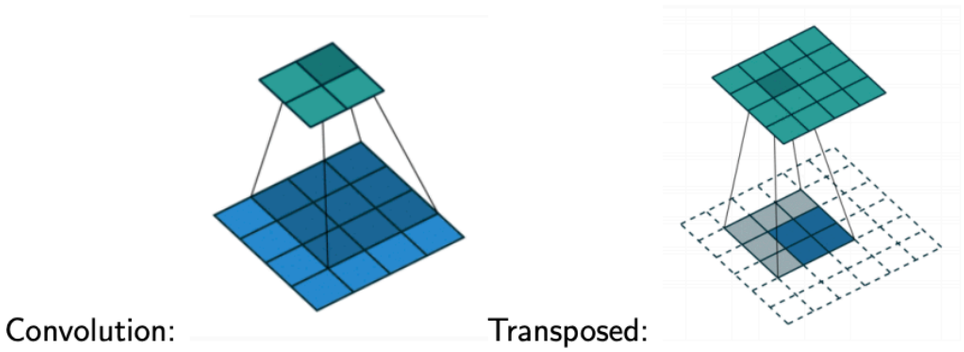
Fully Convolutional Networks for Semantic Segmentation

Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley

Fully Convolutional + Decoder



- Encoder- decoder architecture
- At each output we have a softmax and cross entropy which we sum
- We may also use more sophisticated lo functions specialized for segmentation



https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/image_segmentation.html

The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks

Transposed Convolution

1	2	3
6	5	3
1	4	1

3x3 Input

1	2
2	1

2x2 Kernel

Conv Mat

1	2	0	2	1	0	0	0	0
0	1	2	0	2	1	0	0	0
0	0	0	1	2	0	2	1	0
0	0	0	0	1	2	0	2	1

x

1
2
3
6
5
3
1
4
1

=

22
21
22
20

Transpose

1	0	0	0
2	1	0	0
0	2	0	0
2	0	1	0
1	2	2	1
0	1	0	2
0	0	2	0
0	0	1	2
0	0	0	1

x

1
2
2
4

=

1
4
4
4
13
10
4
10
4

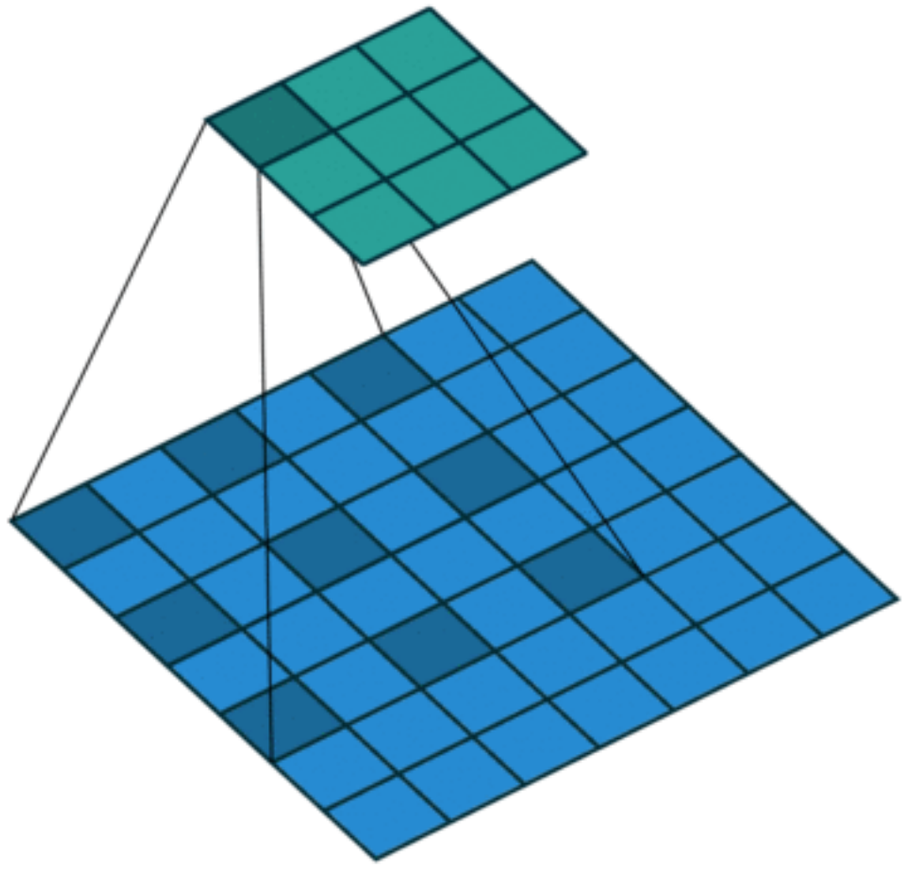
=

1	4	4
4	13	10
4	10	4

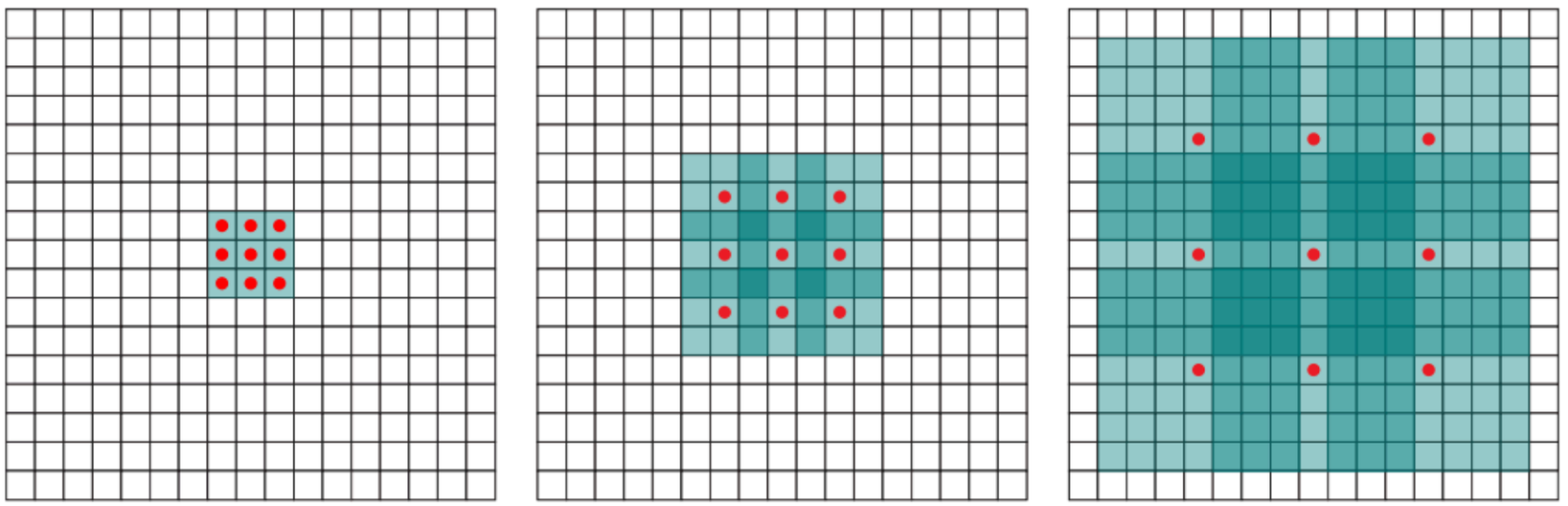
Image credits: www.machinecurve.com

Dilations

- Dilation = 1
- Normal convolutions dilation = 0
- Dilation is also sometimes called “a trous”



Dilation in Segmentation



(a) (b) (c)

MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

Fisher Yu
Princeton University

Vladlen Koltun
Intel Labs

Receptive field to grow exponentially without losing resolution

Layer	1	2	3	4	5	6	7	8
Convolution	3×3	3×3	3×3	3×3	3×3	3×3	3×3	1×1
Dilation	1	1	2	4	8	16	1	1
Truncation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Receptive field	3×3	5×5	9×9	17×17	33×33	65×65	67×67	67×67

- Dilation is also often used to implement transpose of strides convolutions

1D and 3D Conv Nets

- The same ideas have been applied in 1-d and 3-D data
- 1-d convolution has been used in some audio processing
- 3-d convolution is popular
  - Representing and Classifying shape
  - Medical Image volumes

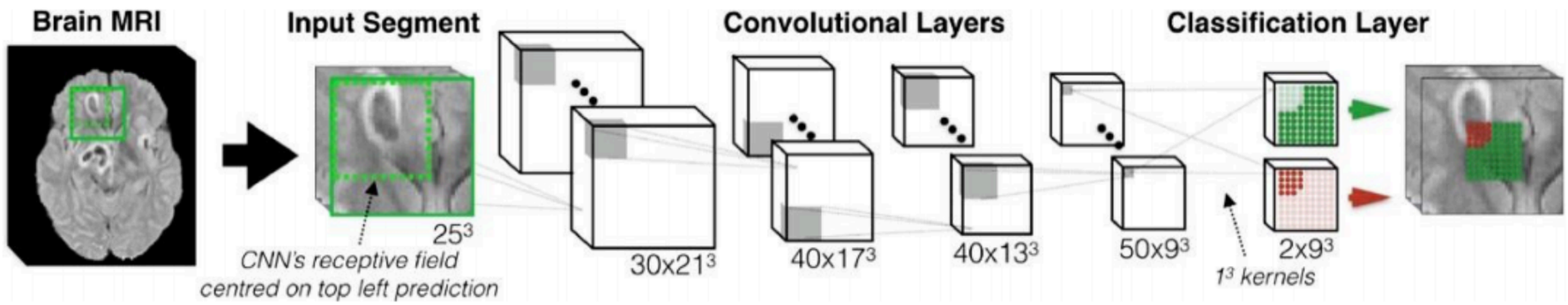


Image credit:

Review

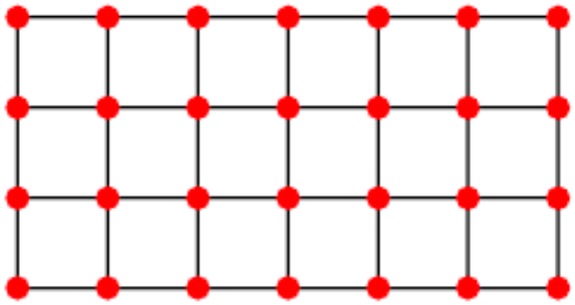
3D Deep Learning on Medical Images: A Review

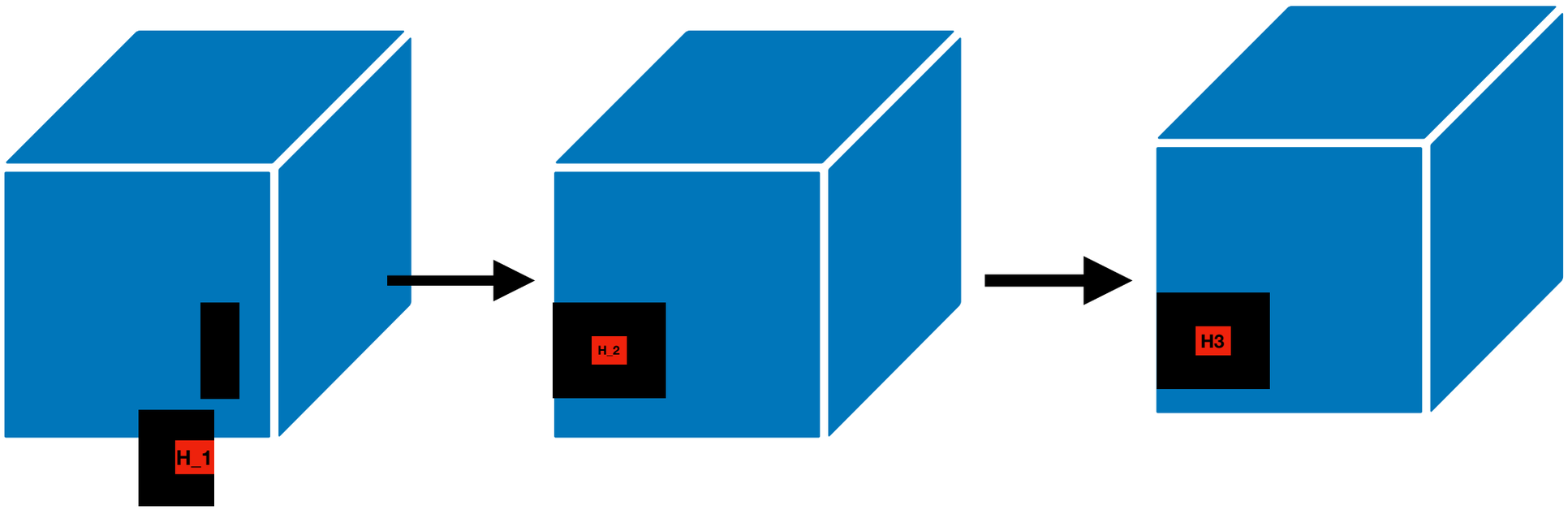
Satya P. Singh [1,2], Lipo Wang [3], Sukrit Gupta [4], Haveesh Goli [4], Parasuraman Padmanabhan and Balázs Gulyás [1,2,5]

Parallels between ConvNets and MP

- Another view on ConvNets

- Consider a ConvNet with no pooling, stride of 1, and tied weights(same weights at each layer)





H2 = f(H1, Neighbors H1, W)

H3 = f(H2, Neighbors H2, W)

Can mimic various dynamic programming algorithms

Some examples of these ideas:

Optional

Tamar, Aviv, et al. "Value iteration networks." *arXiv preprint arXiv:1602.02867* (2016).

Conditional Random Fields as Recurrent Neural Networks