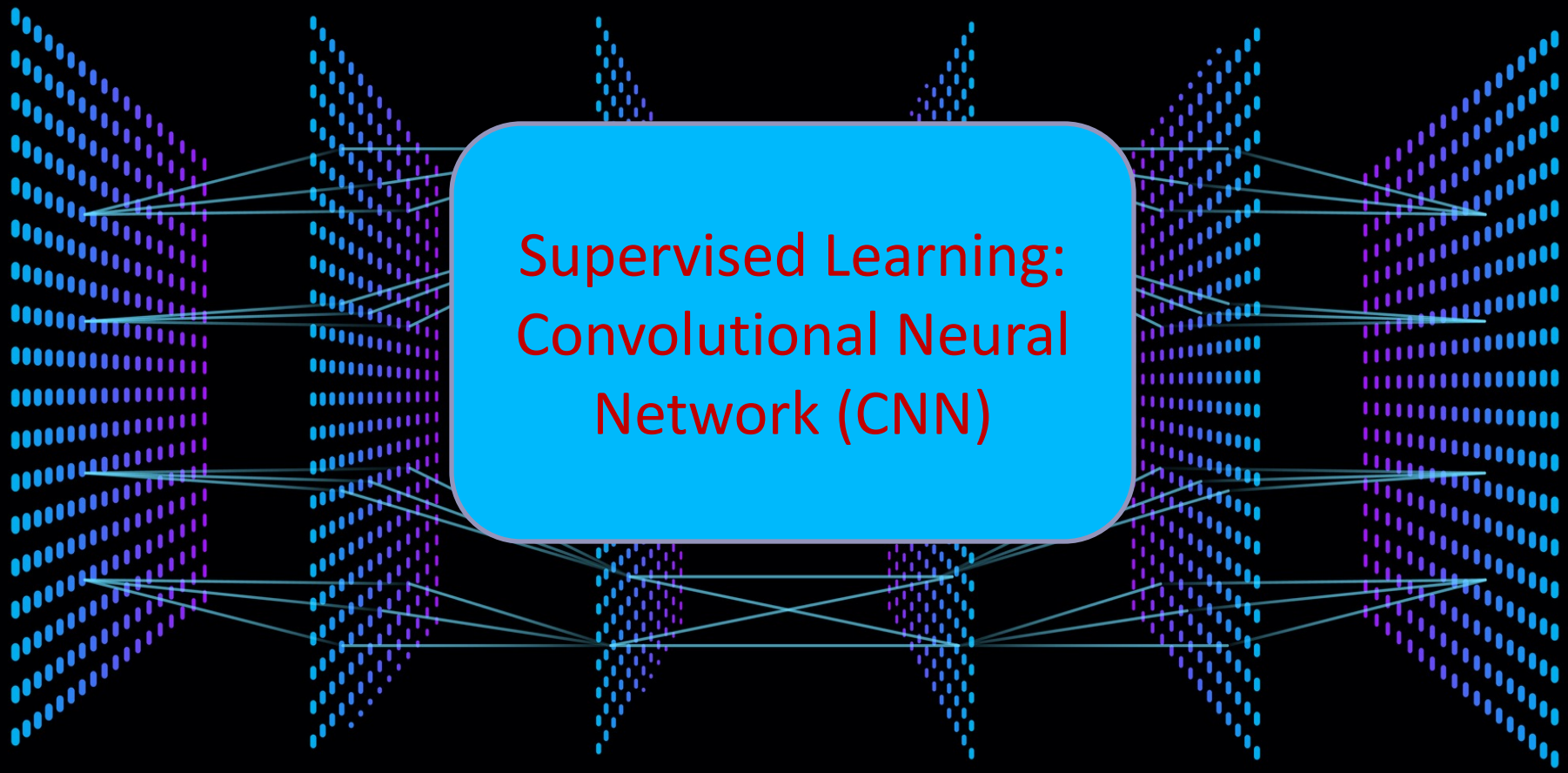
Deep Learning

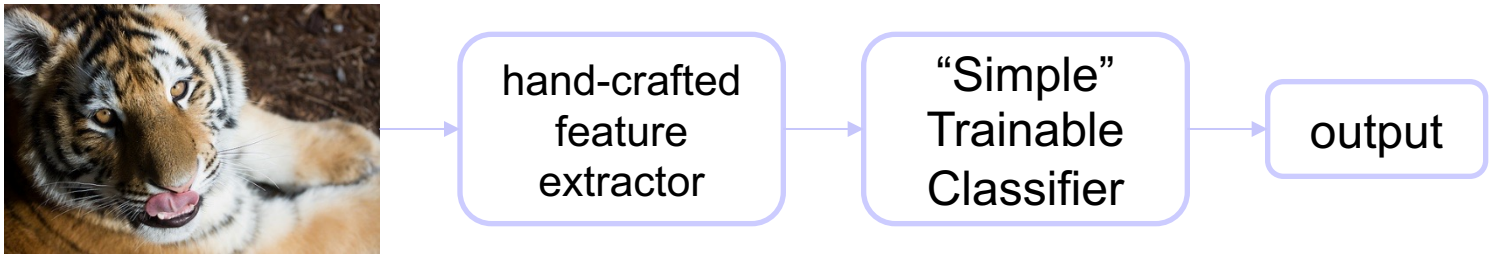


Roadmap

- Concept of deep learning
- Convolutional Neural Network
  - Concept of convolution
  - Architecture: VGGNet
  - Architecture Explained
- Interesting Applications
- Remarks and Take-home messages

Introduction

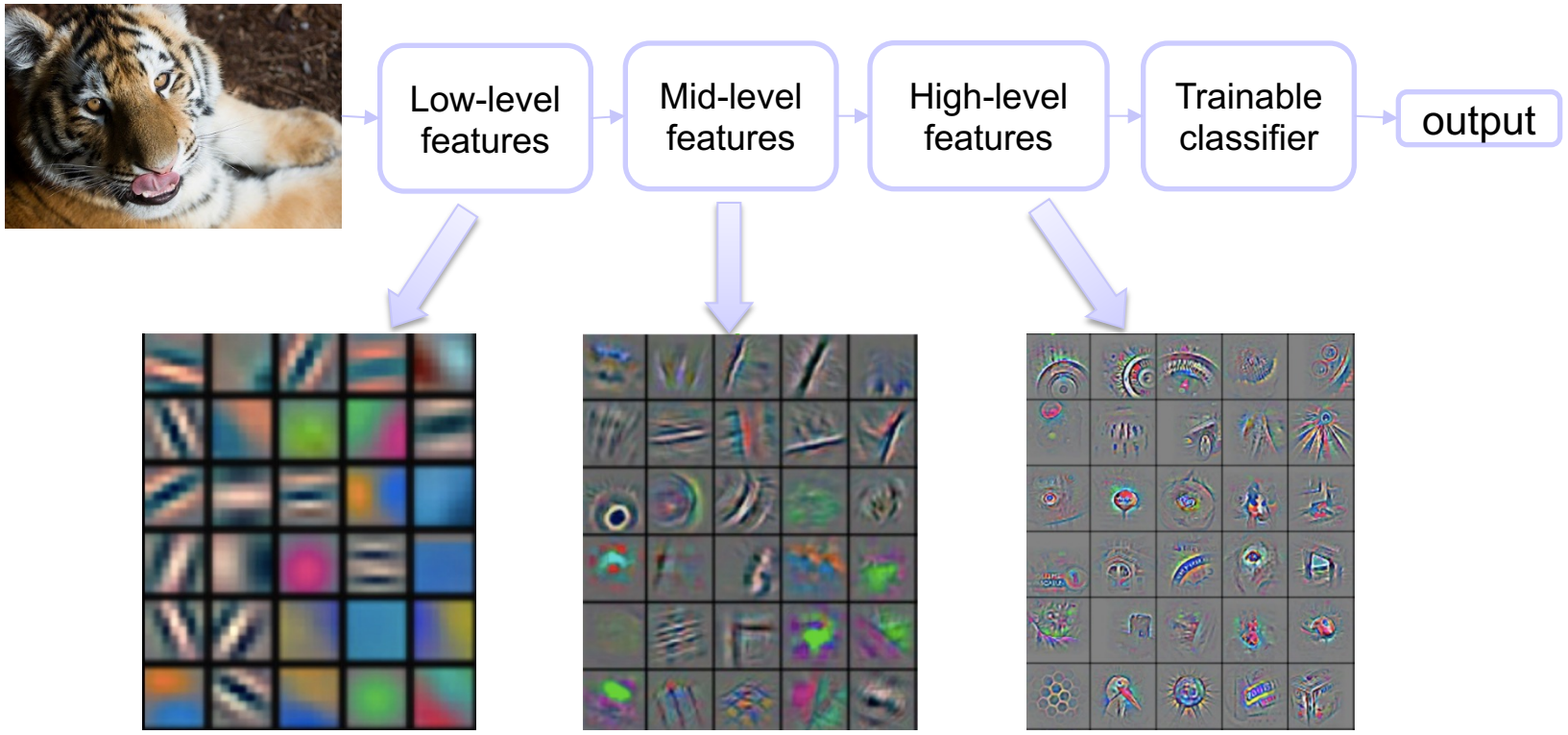
- Traditional computer vision and pattern recognition (CVPR) models use hand-crafted features (feature engineering) and relatively simple trainable classifier.



- This approach has the following limitations:
  - It is very tedious and costly to develop hand-crafted features
  - The hand-crafted features are usually highly dependent on one application, and cannot be transferred easily to other applications

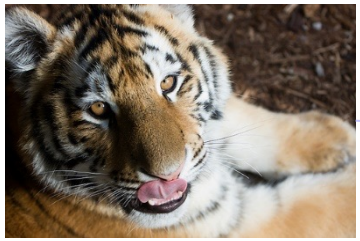
Deep Learning

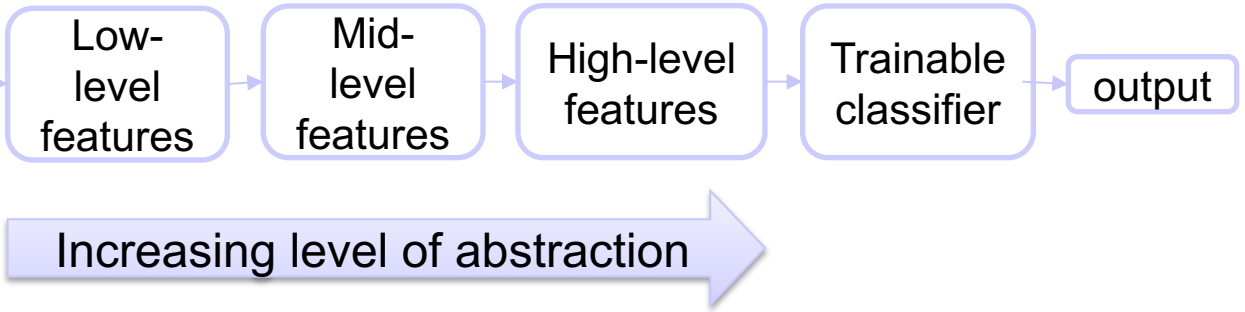
- Deep learning (a.k.a. representation learning) seeks to learn rich hierarchical representations (i.e. features) automatically through multiple stage of feature learning process.



Feature visualization of convolutional net trained on ImageNet (Zeiler and Fergus, 2013)

Learning Hierarchical Representations

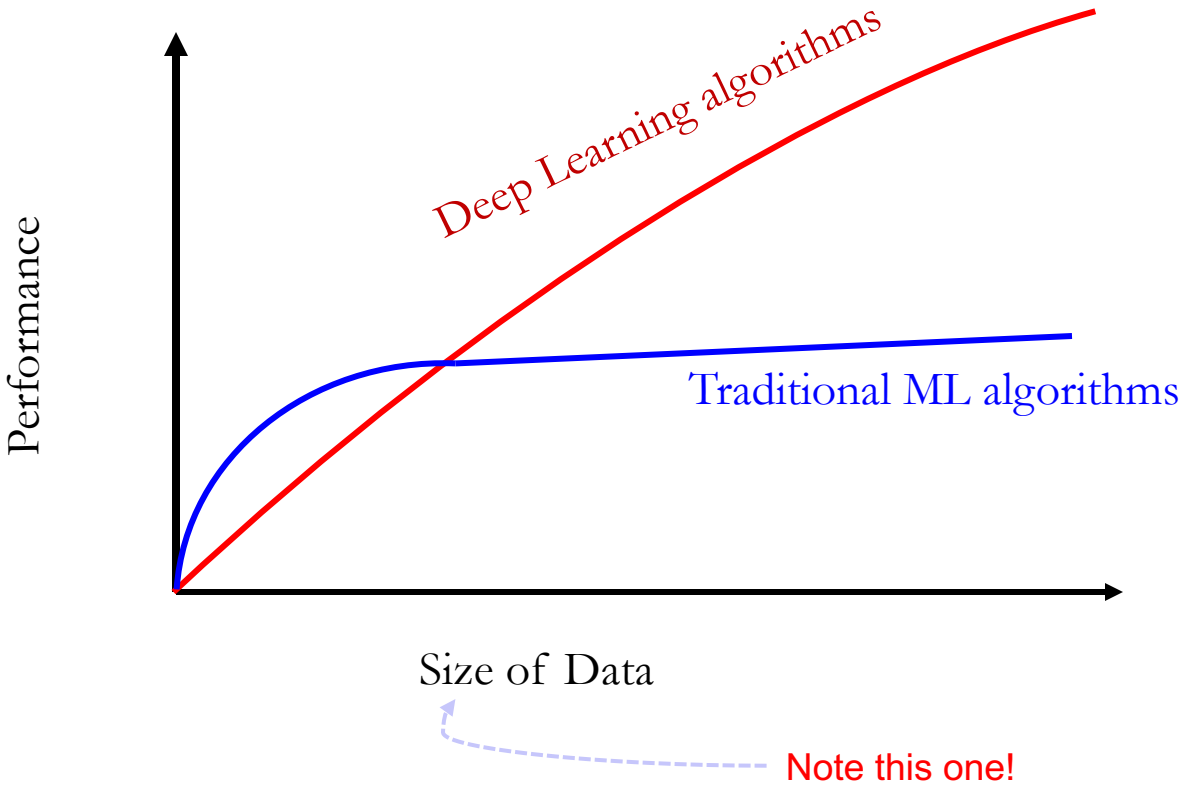




- Hierarchy of representations with increasing level of abstraction. Each stage is a kind of trainable nonlinear feature transform
- Image recognition
  - Pixel → edge → texton → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story

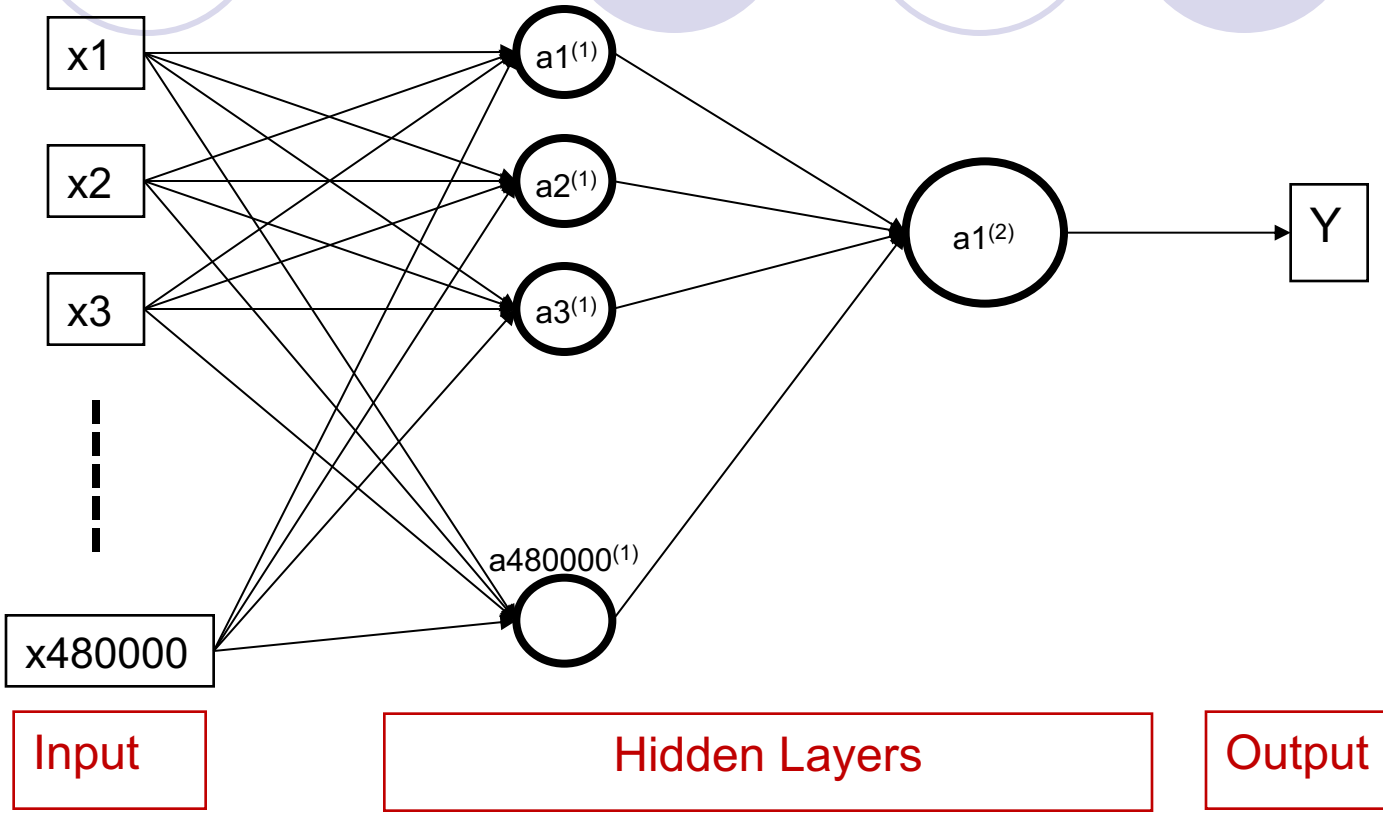
“Deep Learning doesn’t do different things,
it does things differently”

Performance vs Sample Size



If the input is an Image?

(H)400 X (W)400 X (D)3
= 480000



Number of Parameters

480000*480000 + 480000 +1 = **approximately 230 Billion !!!**

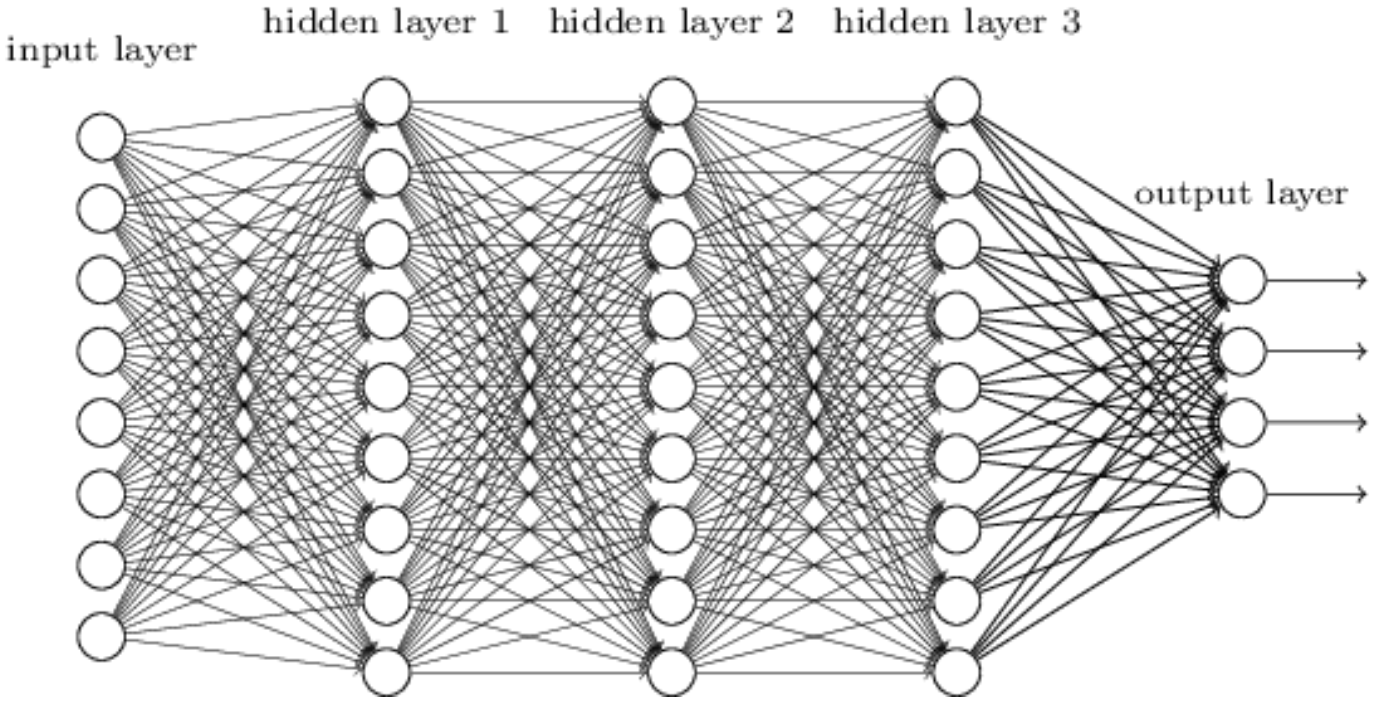
480000*1000 + 1000 +1 = approximately **480 million !!!**

Number of hidden nodes: 480000 vs 1000

Let us see how convolutional layers help.

Convolutional Neural Network (CNN)

- We know it is good to learn a small model (at least smaller number of learnable parameters, for SGD).
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?



So, we need to

- have sparse connections
- have parameter sharing
- automatically generalize across spatial translations of inputs (from an image processing point of view)

Furthermore, we need to

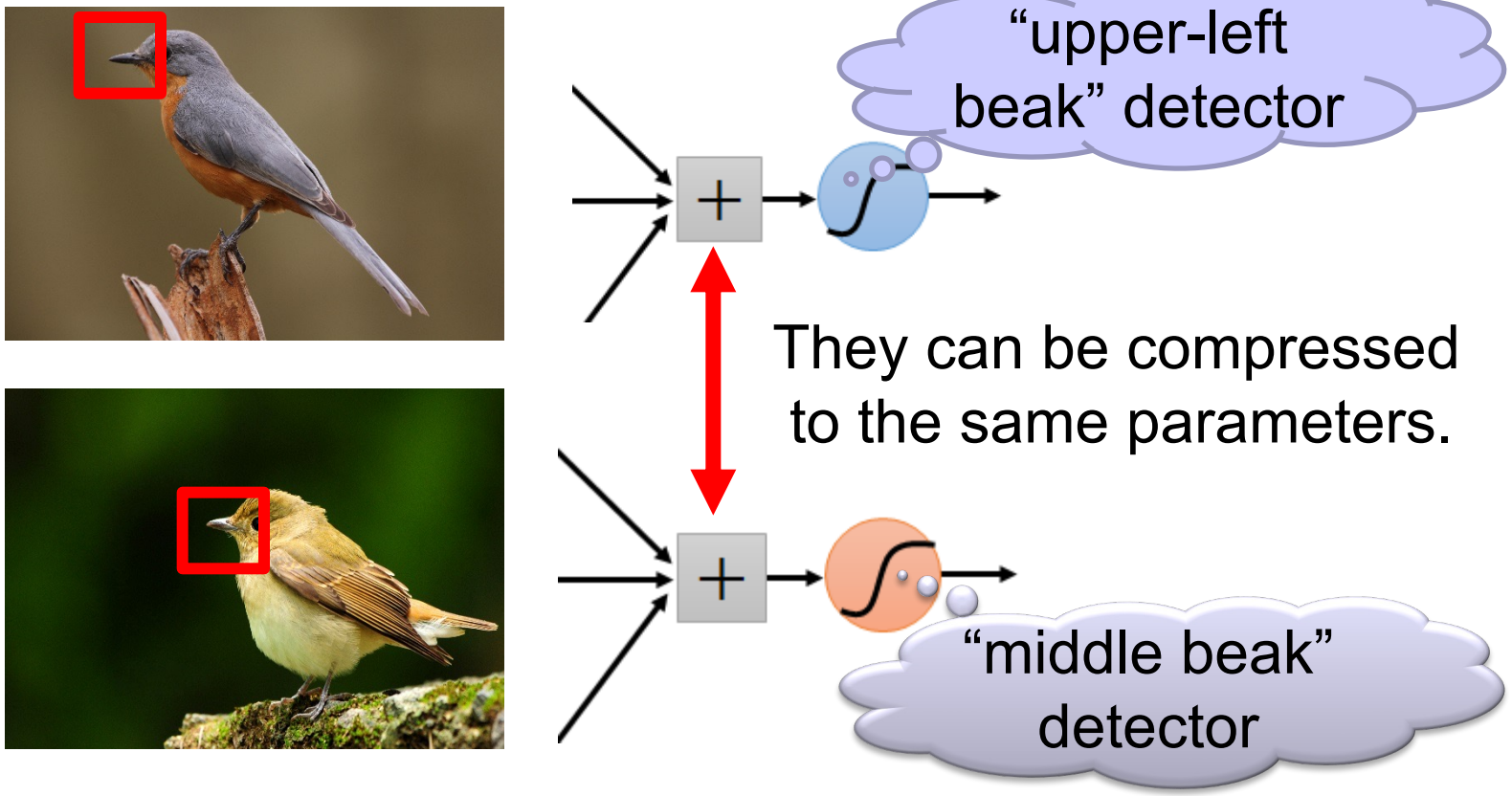
- scale up neural networks to process very large images/video sequences

Consider learning an image:

- Some patterns are much smaller than the whole image

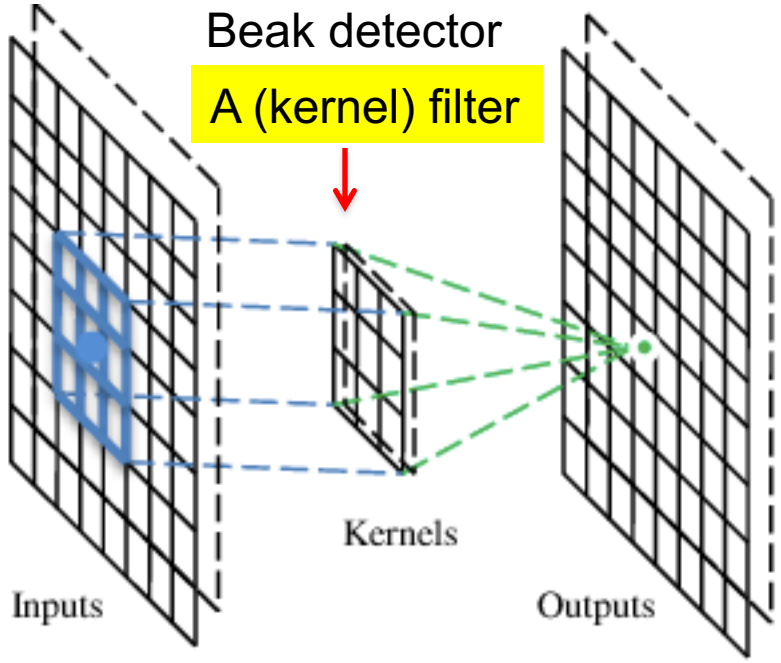
Can represent a small region with fewer parameters

Same pattern appears in different places:
They can be compressed!
What about training a lot of such “small” detectors and each detector can “move around”.



A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



2D Convolution

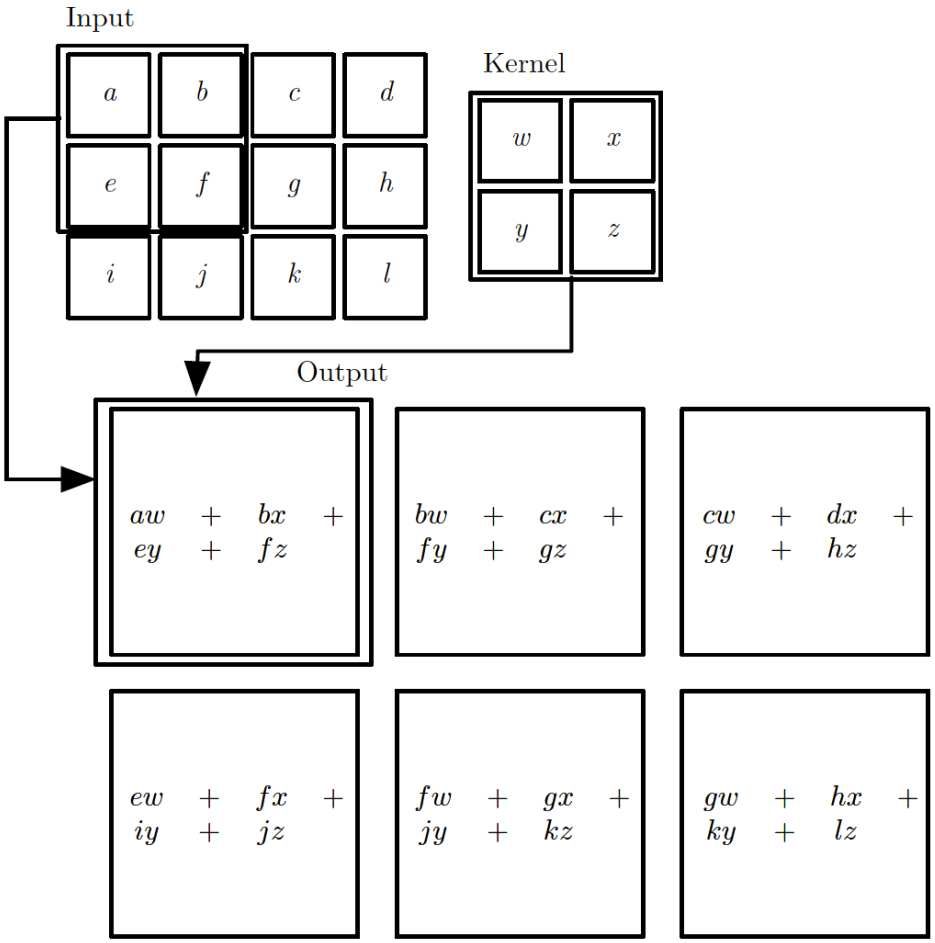


Figure 9.1

Convolution

These are the network parameters to be learned.
Only 2 filters here!

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

Defines how far the filter moves

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Dot product →

3 -1

6 x 6 image

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

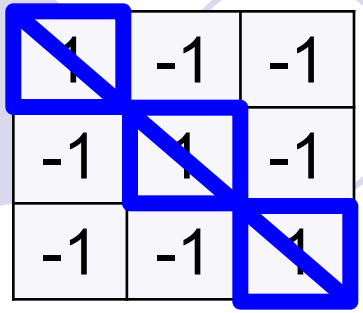
If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

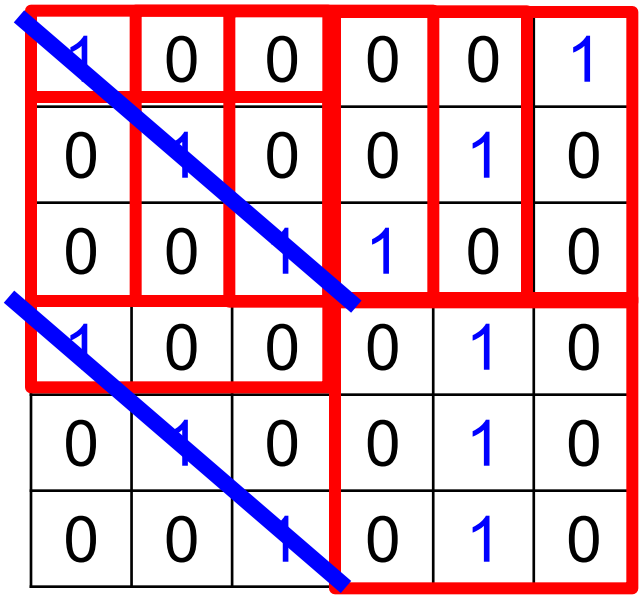
3 -3

Convolution

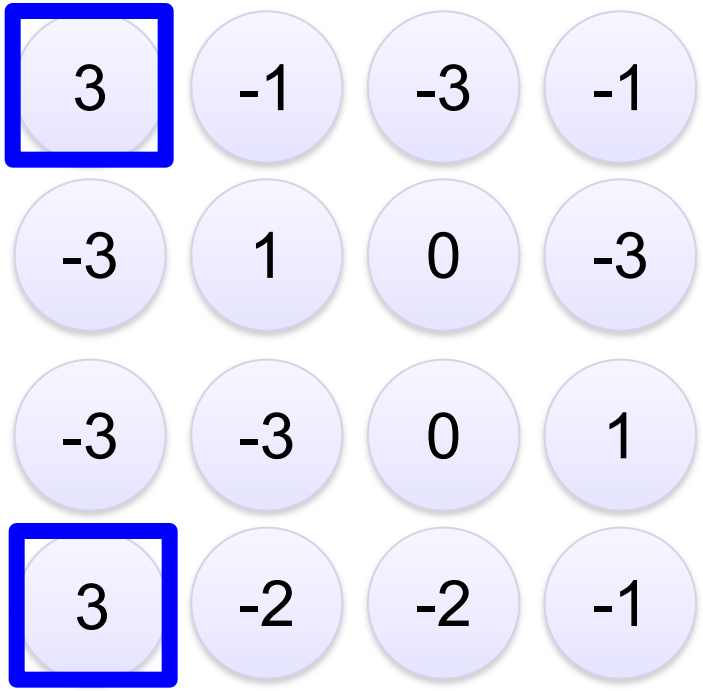


Filter 1

stride=1



6 x 6 image



Convolution

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

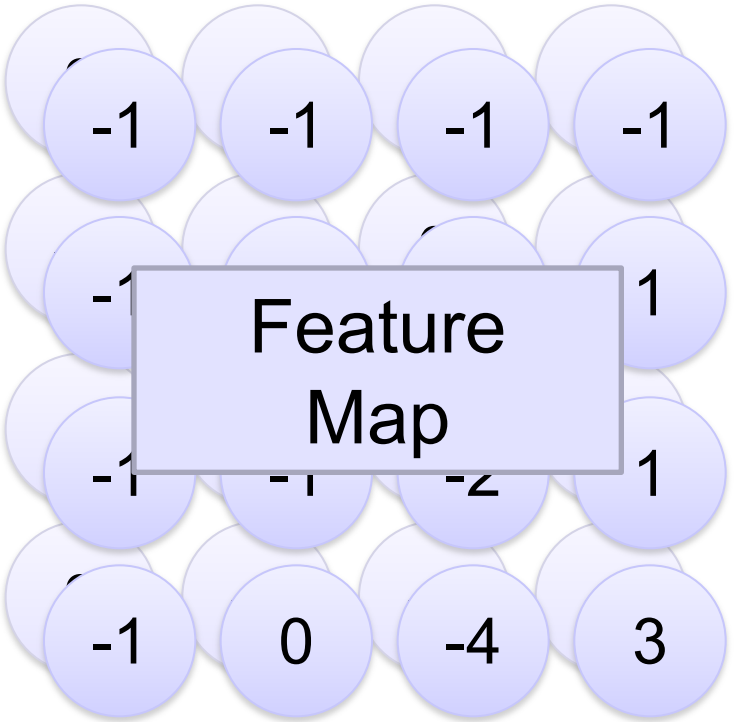
stride=1

define how far the filter moves

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

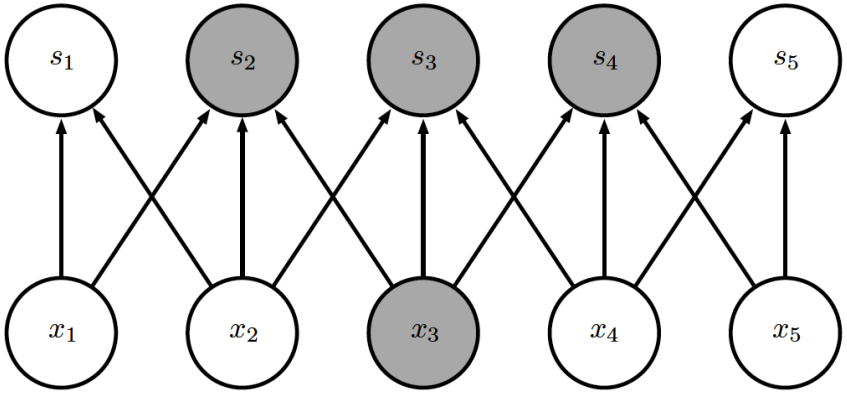
Repeat this for each filter



Two 4 x 4 images
Forming 4 x 4 x 2 matrix

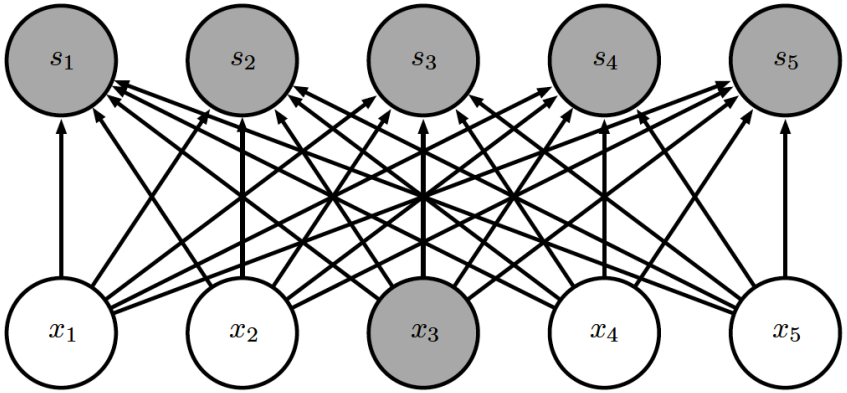
Concepts Behind:
Sparse Connectivity

Sparse connections due to small convolution kernel



1x3 connections (weights)/x_i

Dense connections



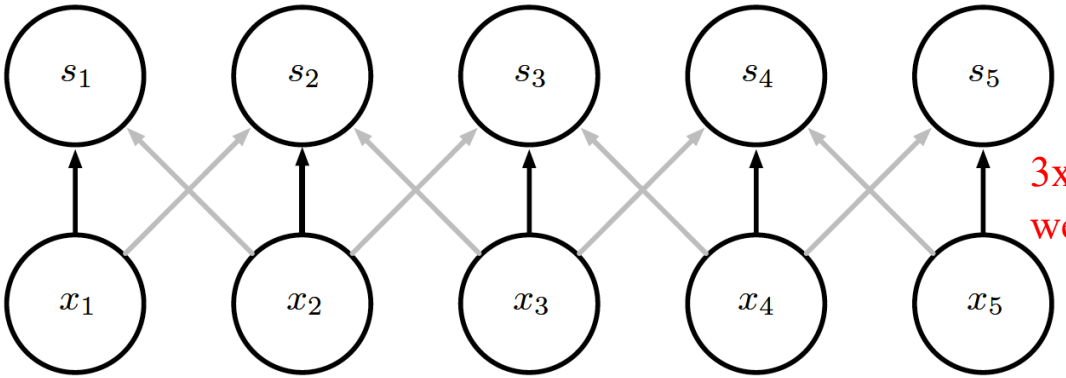
1x5 connections (weights)/x_i

Figure 9.2

(Goodfellow 2016)

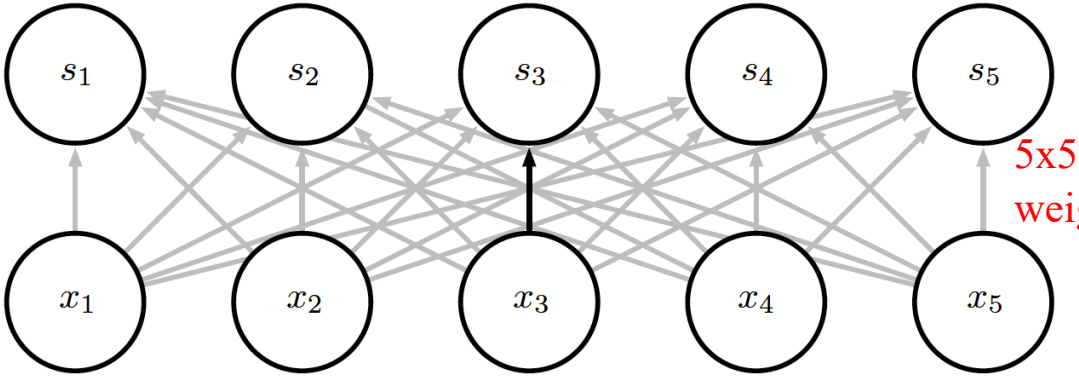
Concepts Behind:
Parameter Sharing

Convolution shares the same parameters across all spatial locations



3x1 connection weights only

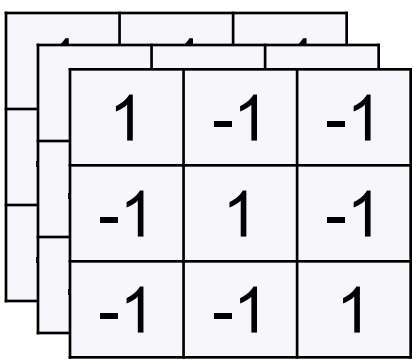
Traditional matrix multiplication does not share any parameters



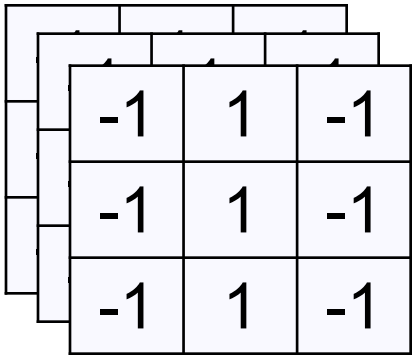
5x5 connection weights

Figure 9.5

Color image: RGB 3 channels

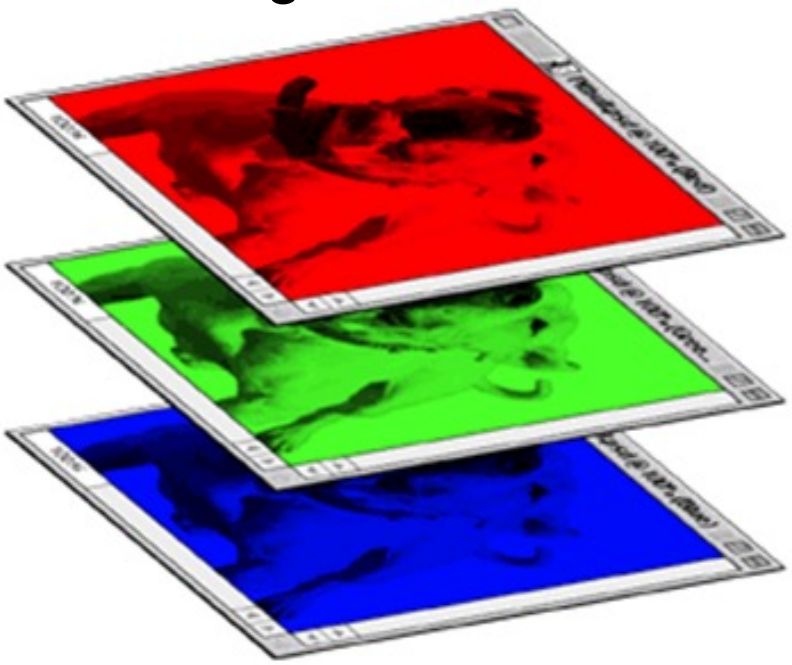


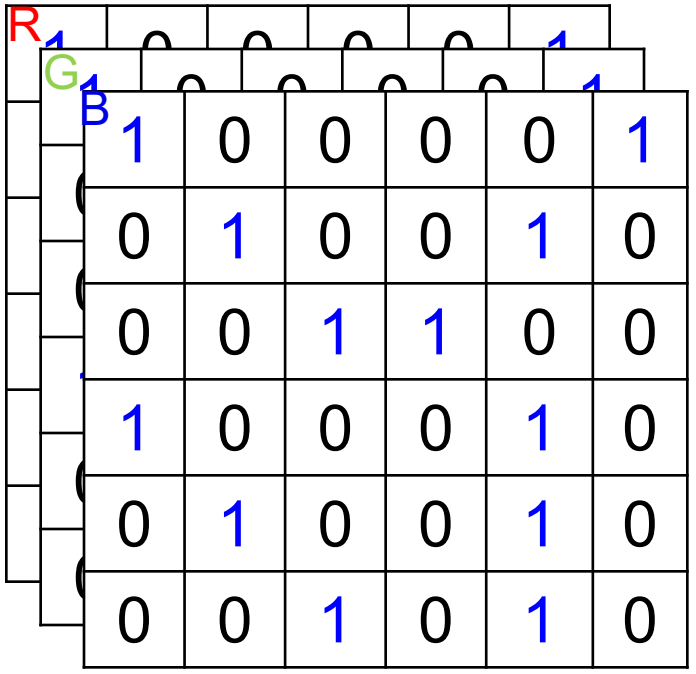
Filter 1



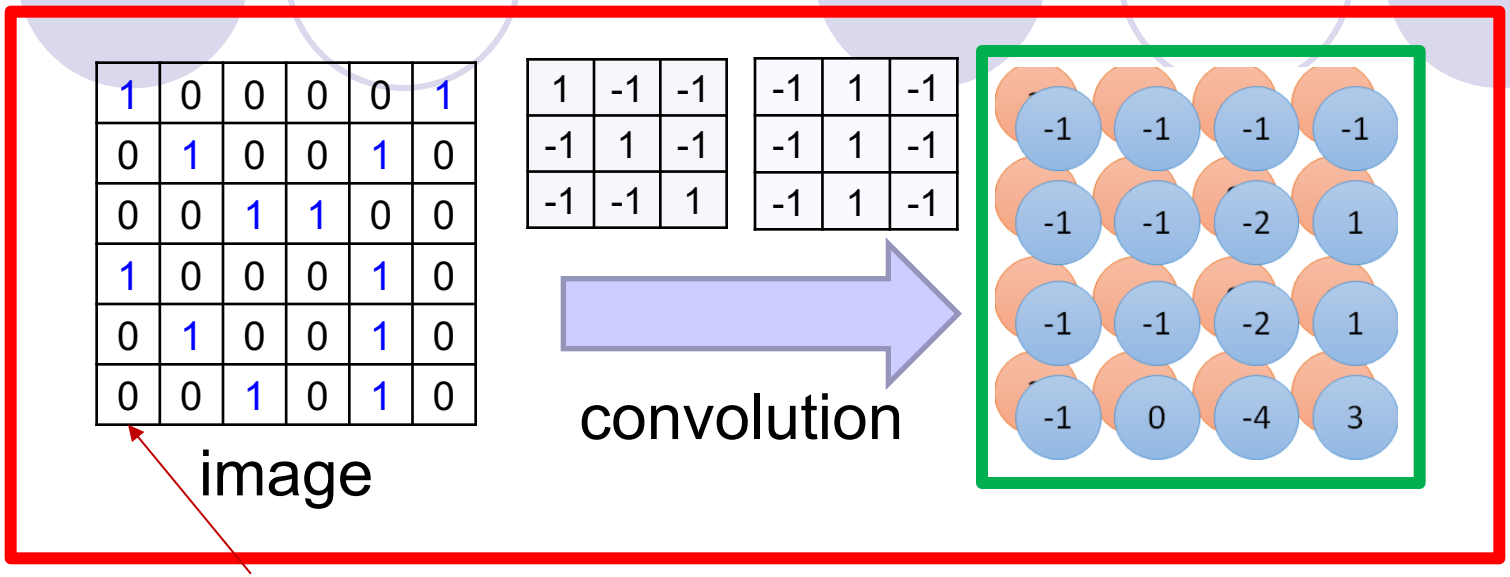
Filter 2

Color image





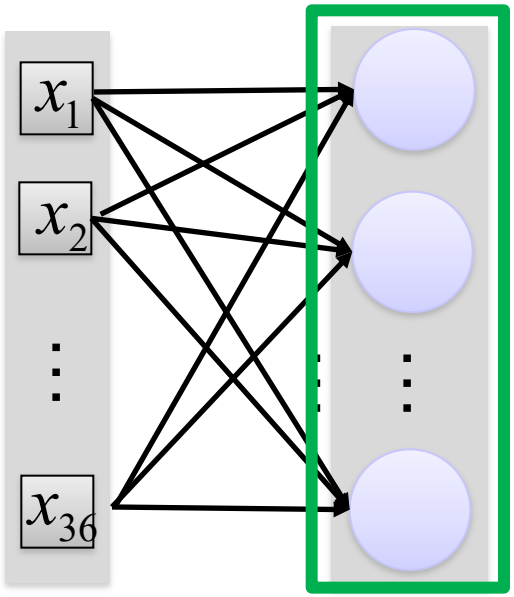
Convolution v.s. Fully Connected

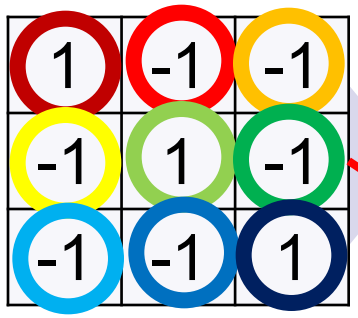


Only 1 channel shown here!

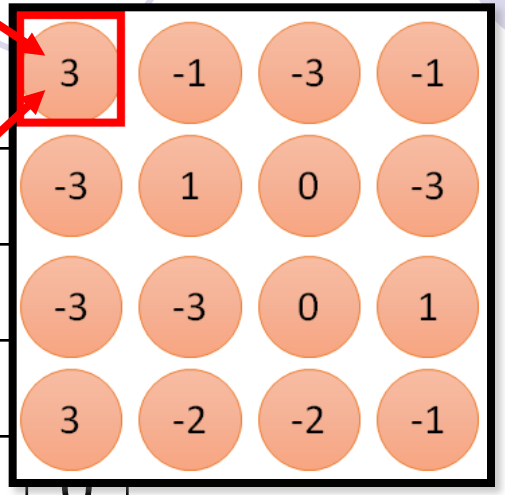
Fully-connected

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0





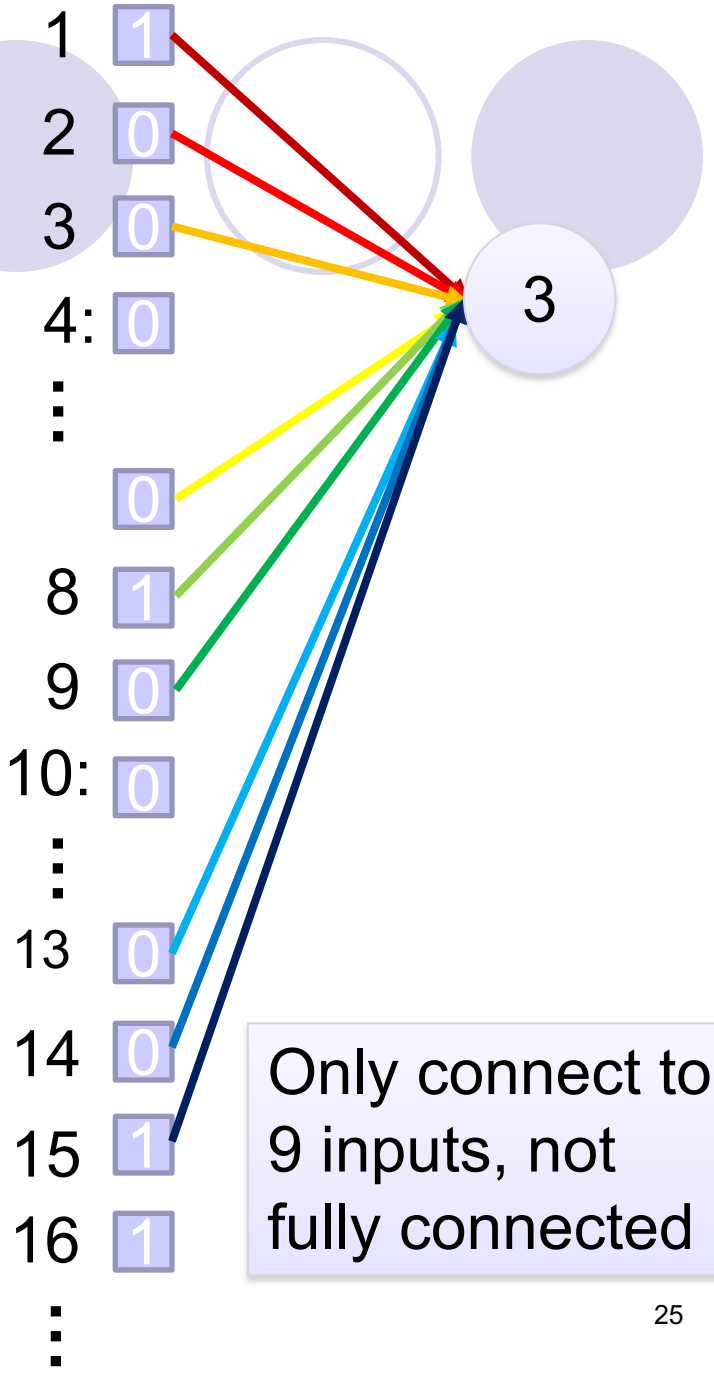
Filter 1



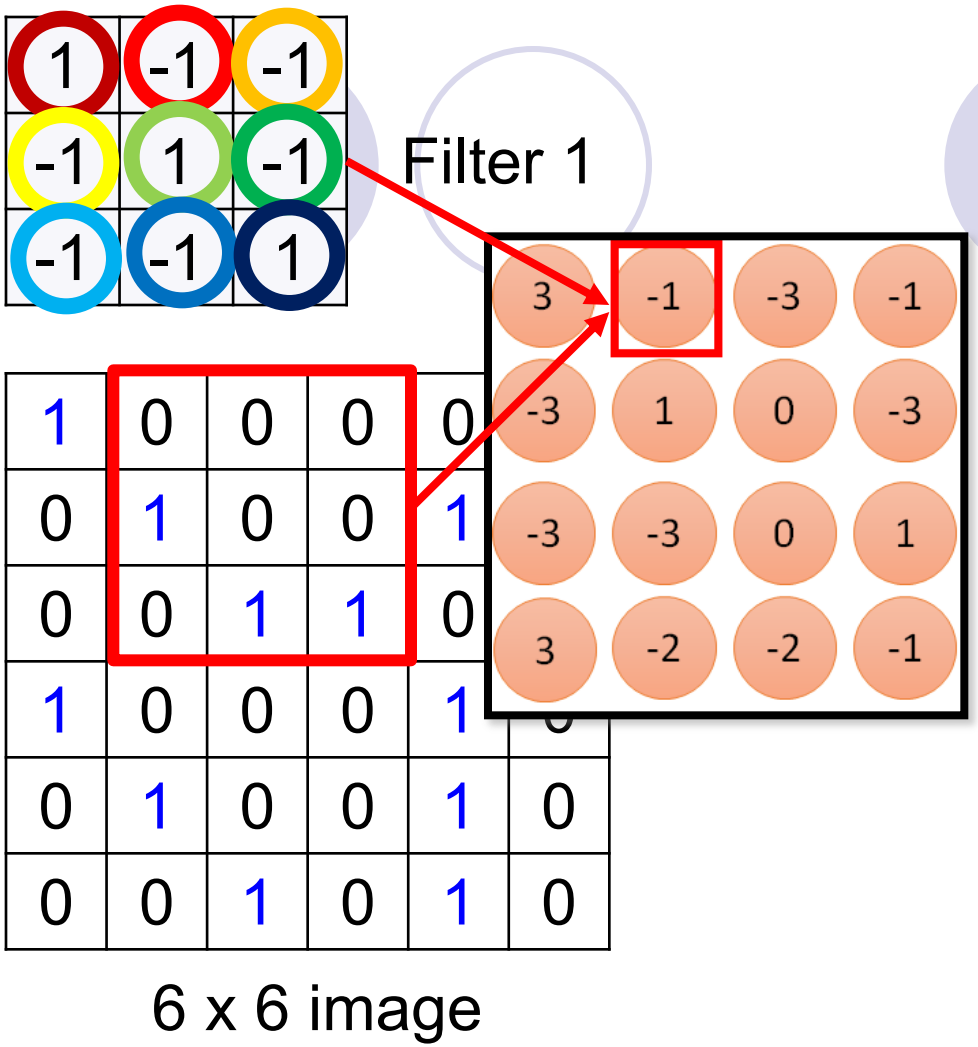
1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

fewer parameters!

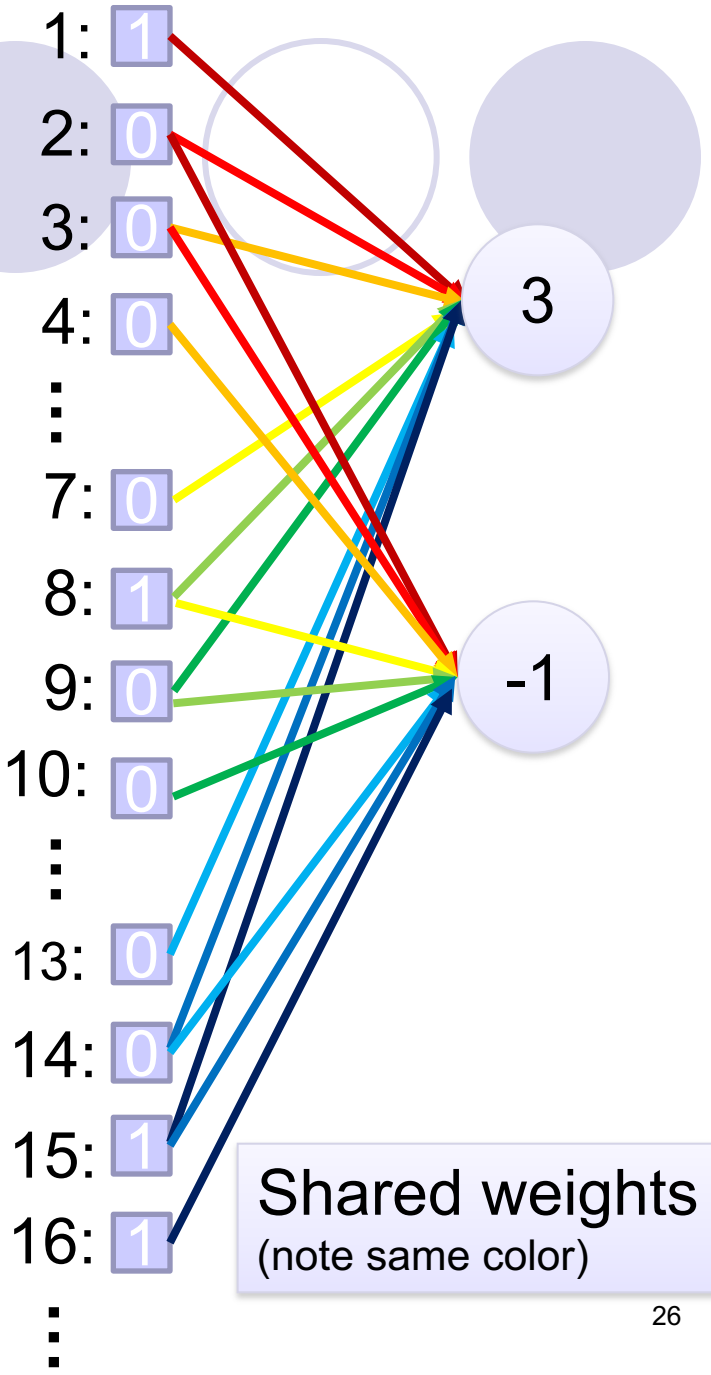


Only connect to 9 inputs, not fully connected



Fewer parameters

Even fewer parameters



Further example of convolution

Edge detection by convolution

Demo link 1 (click here!)
Demo link 2 (click here!)

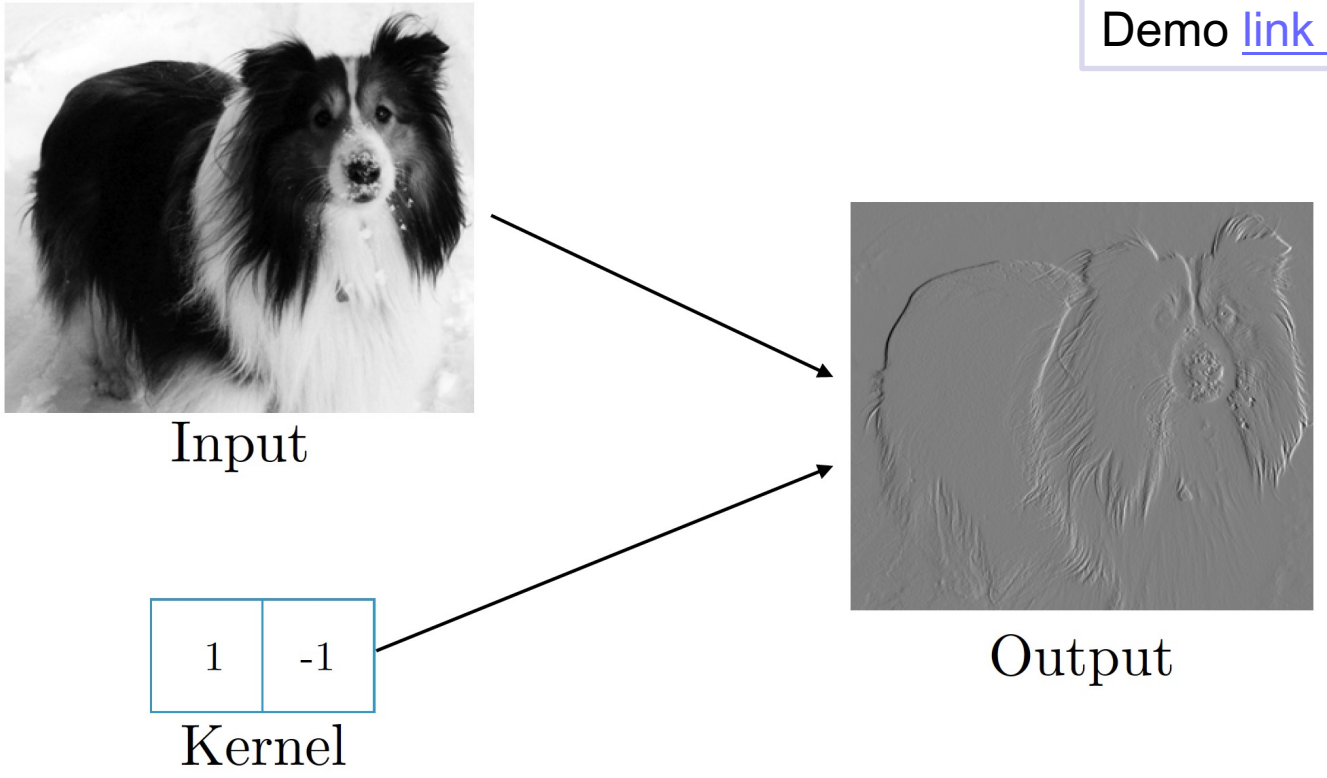


Figure 9.6

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)

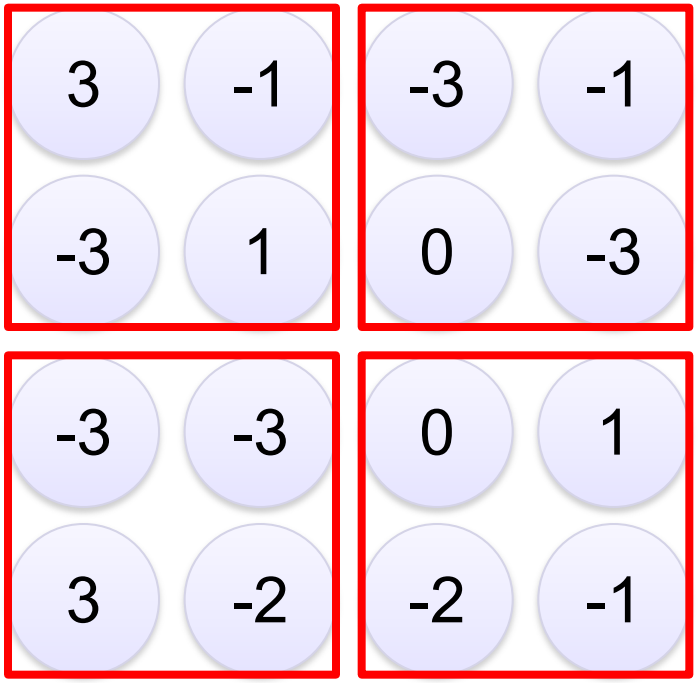
Max Pooling

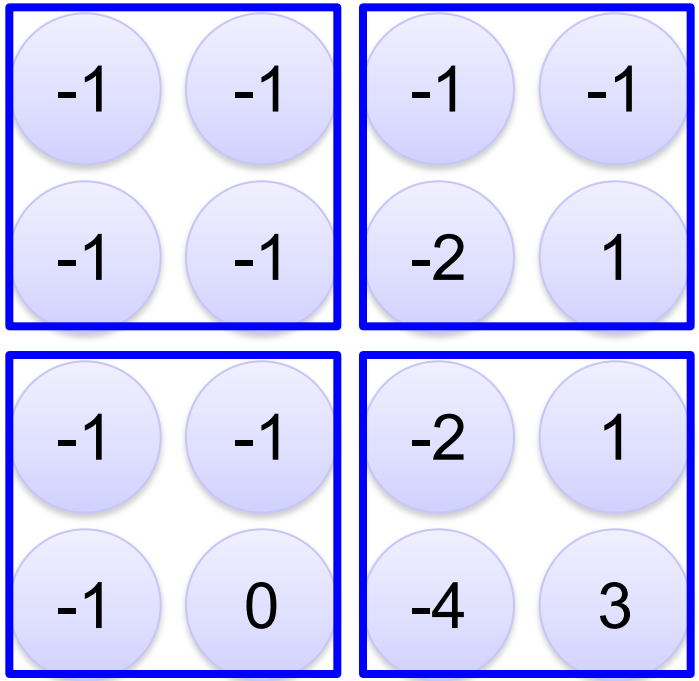
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2





Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

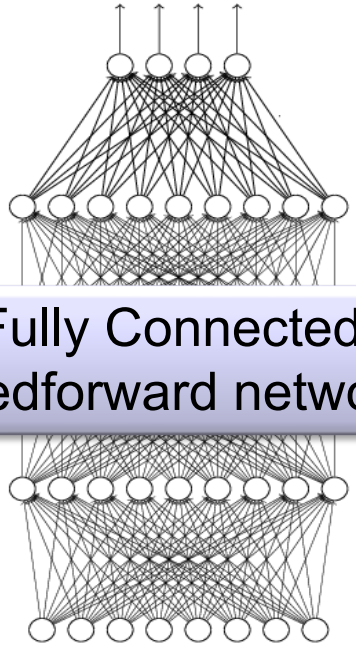
bird

We can subsample the pixels to make image smaller

→ fewer parameters to characterize the image

The whole CNN

cat dog ……



Fully Connected Feedforward network

Convolution

Max Pooling

Convolution

Max Pooling

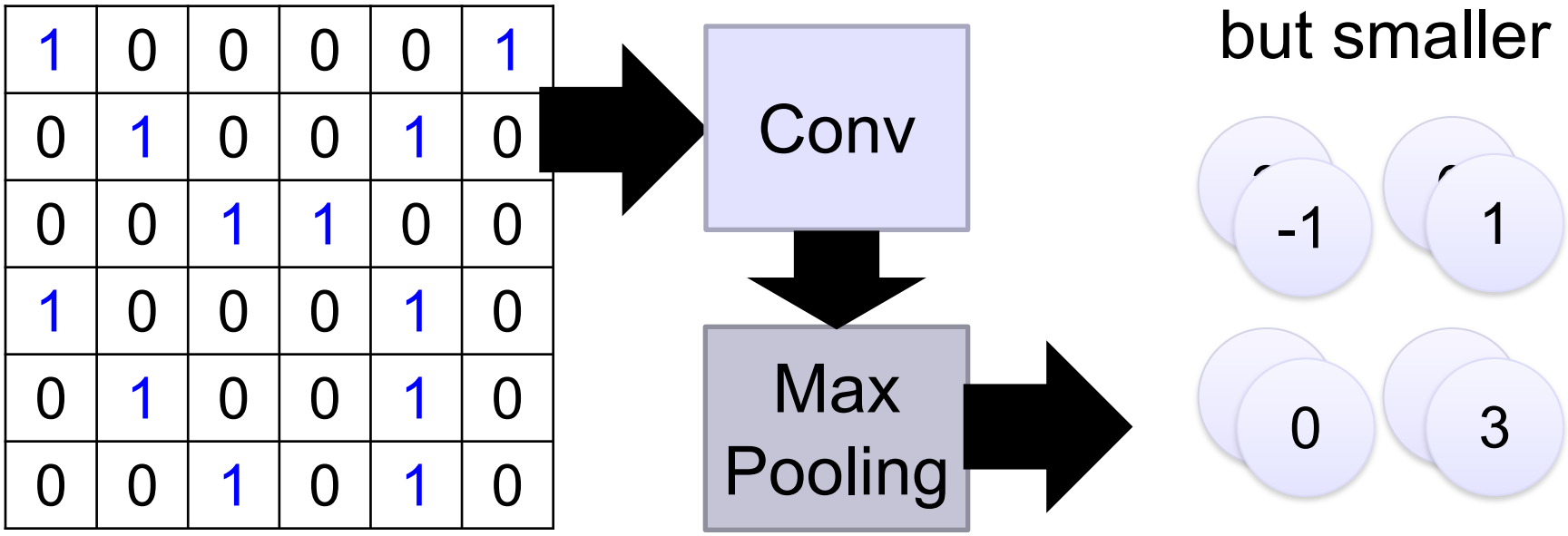
Can repeat many times

Flattened

A CNN compresses a fully connected network in two ways:

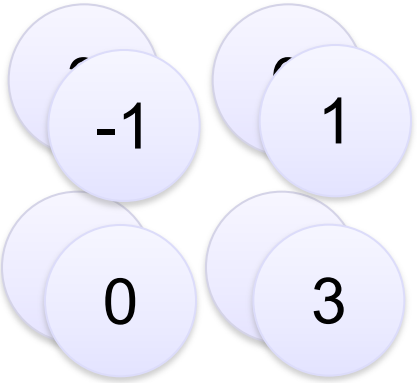
- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

Max Pooling

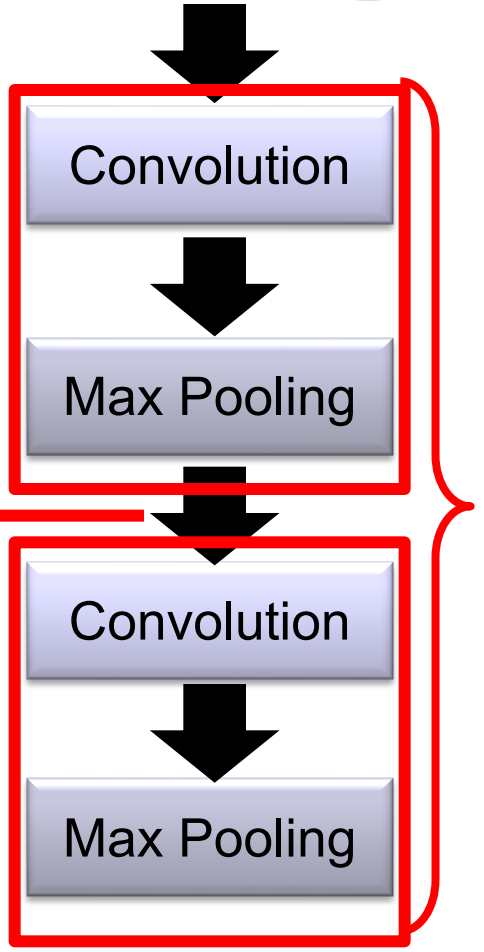


6 x 6 image

The whole CNN



A new image



Can repeat many times

Smaller than the original image

The number of channels is the number of filters

The whole CNN

cat dog ……

Convolution

Max Pooling

A new image

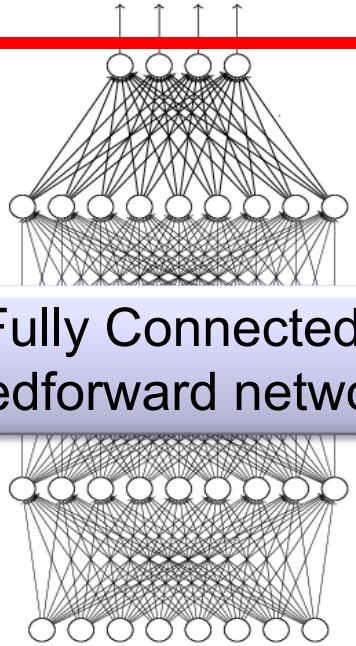
Convolution

Max Pooling

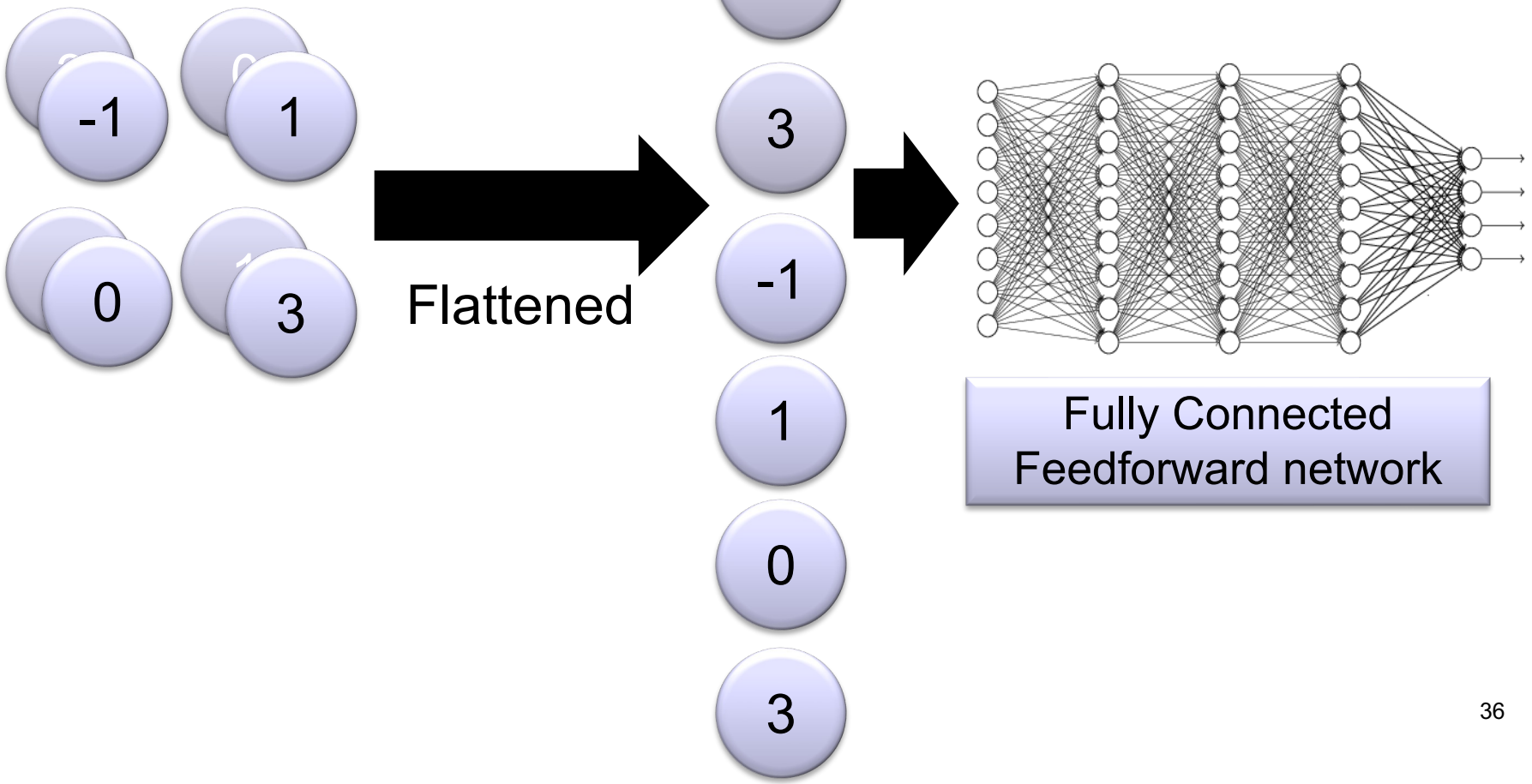
A new image

Flattened

Fully Connected Feedforward network

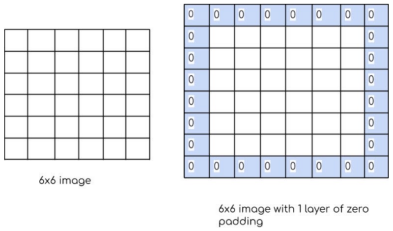


Flattening


-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

Convolutional Neural Networks

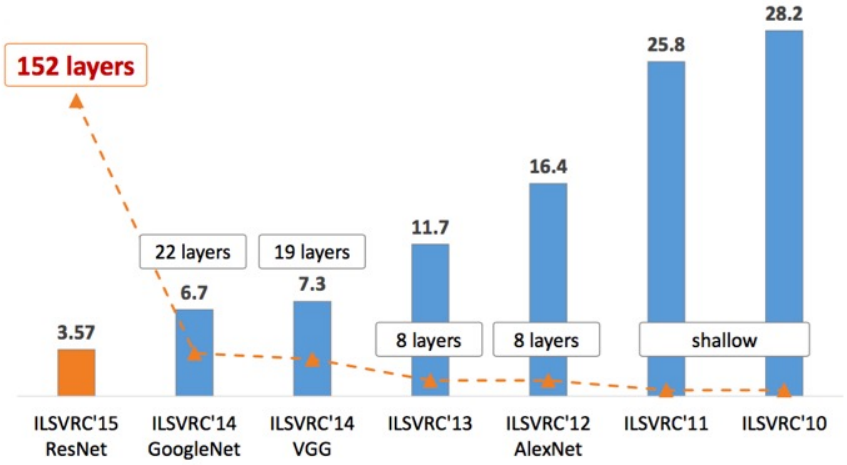
- **Output:** Binary, Multinomial, Continuous, Count
- **Input:** fixed size, can use padding (e.g. zero padding) to make all images same size.



- **Architecture:** Choice is ad hoc
  - requires experimentation, see coming slides.
- **Optimization:** Backward propagation
  - hyper parameters (number of filters, layer sizes, etc.) for very deep model can be estimated properly only if you have billions of images.
    - Use an architecture and trained hyper parameters from other papers (Imagenet or Microsoft/Google APIs etc)
- **Computing Power:** Buy a GPU!!

CNN Architectures for Visual Applications:
See this link! And this update!

- LeNet, AlexNet, VGG, GoogLeNet, ResNet designed for the ImageNet project (large visual database designed for use in visual object recognition software research)

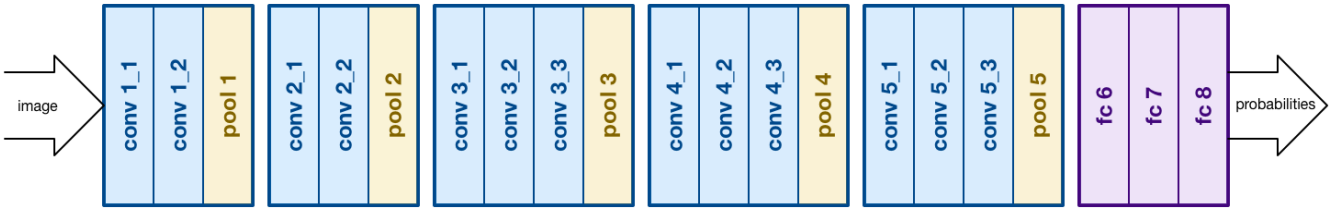


Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	

Take a further look at this important link!

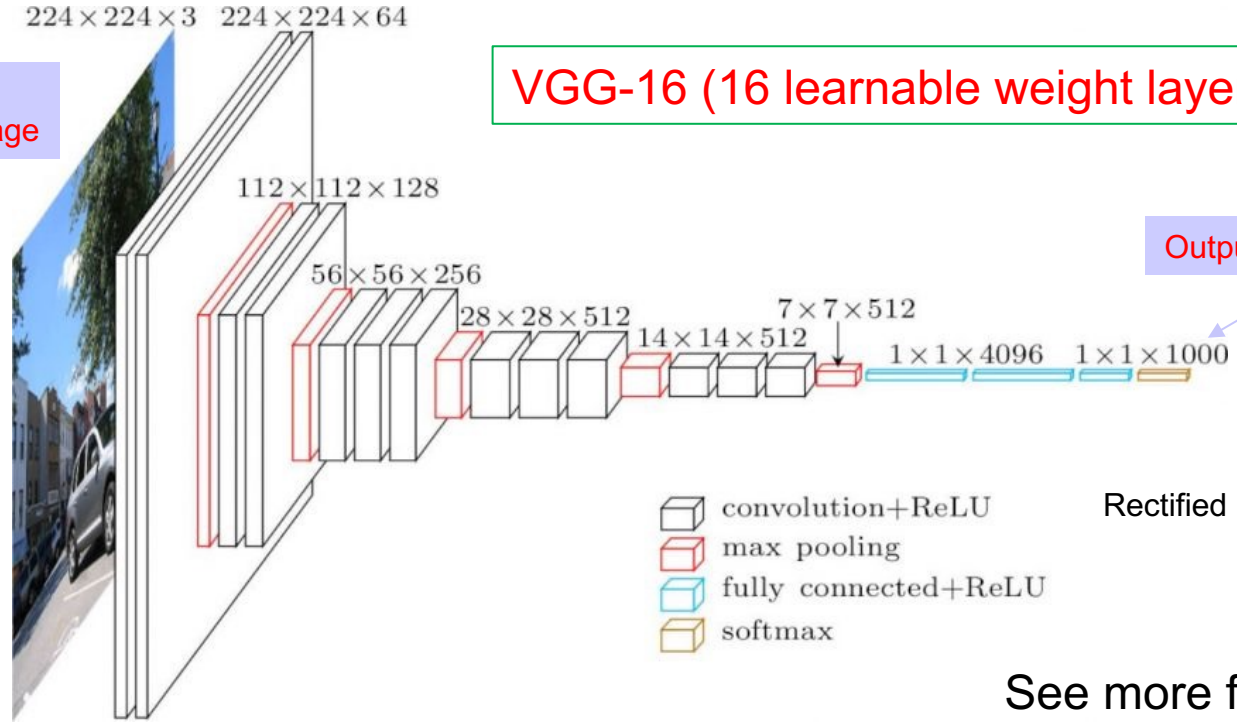
Popular CNN Architecture: VGGNet

- VGGNet16 (2014) with 3x3 convolutions and lots of filters
- Quoted as “trained on 4GPUs for 2-3 weeks” for 14 million images from 1000 classes

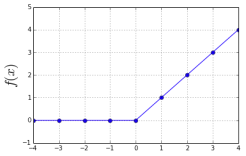


Iutput:
2D color image

VGG-16 (16 learnable weight layers)



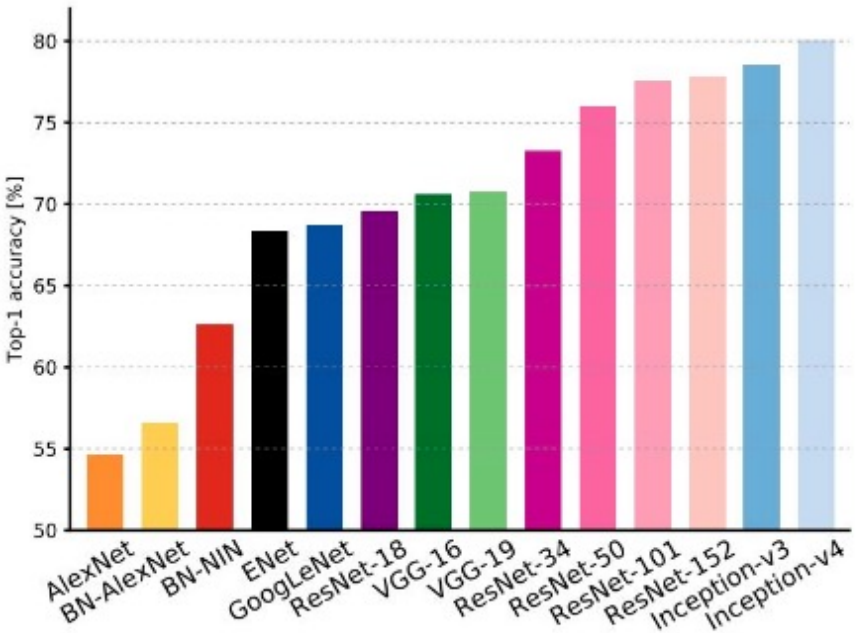
Output: 1000 classes

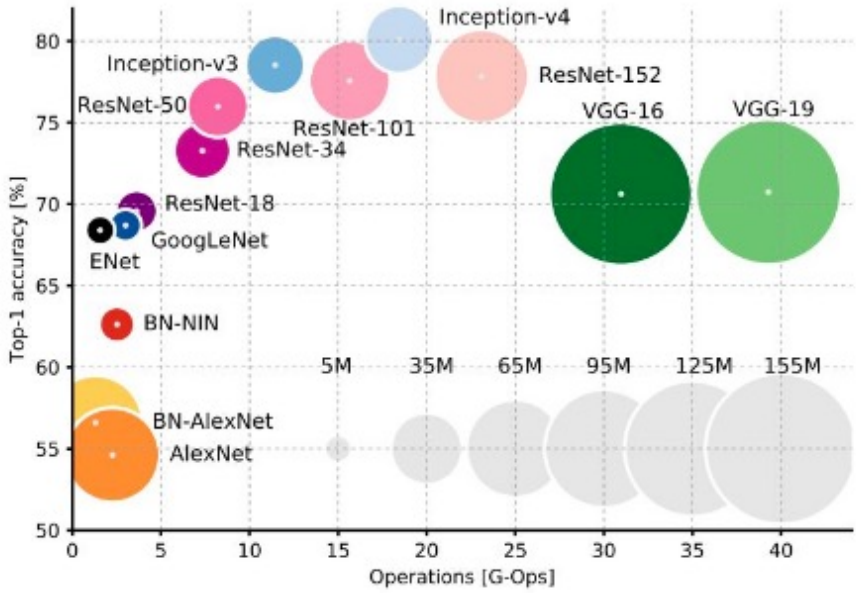


Rectified linear unit: f(x)=max(0,x)

See more from this link.

Yet another analysis



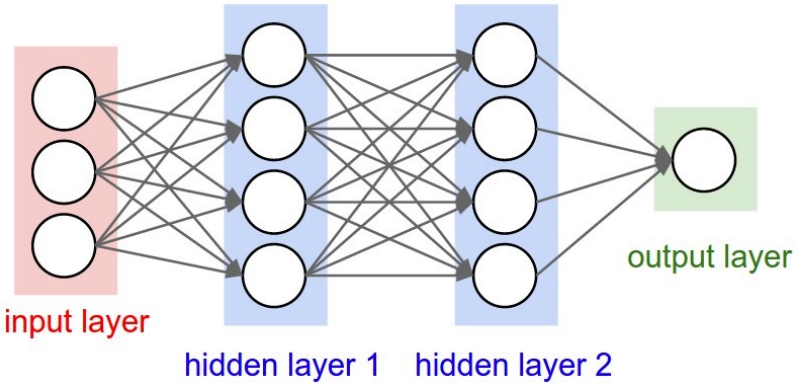


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

CNN Architecture Explained

(based on https://cs231n.github.io/convolutional-networks/)

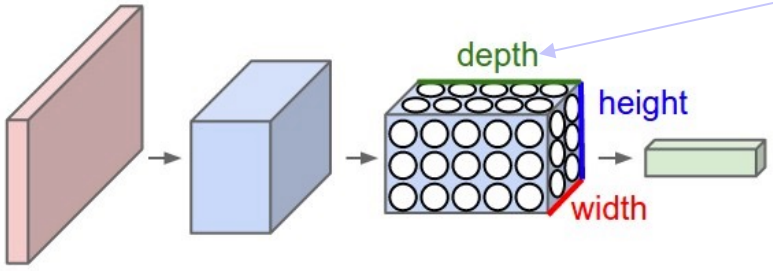
- NN Architecture vs CNN Architecture



1D-to-1D 1D-to-1D 1D-to-1D

For a color image, the height and width correspond to the spatial size while the depth corresponds to the color depth, i.e., 3 channels.

The depth here corresponds to the number of filters used.

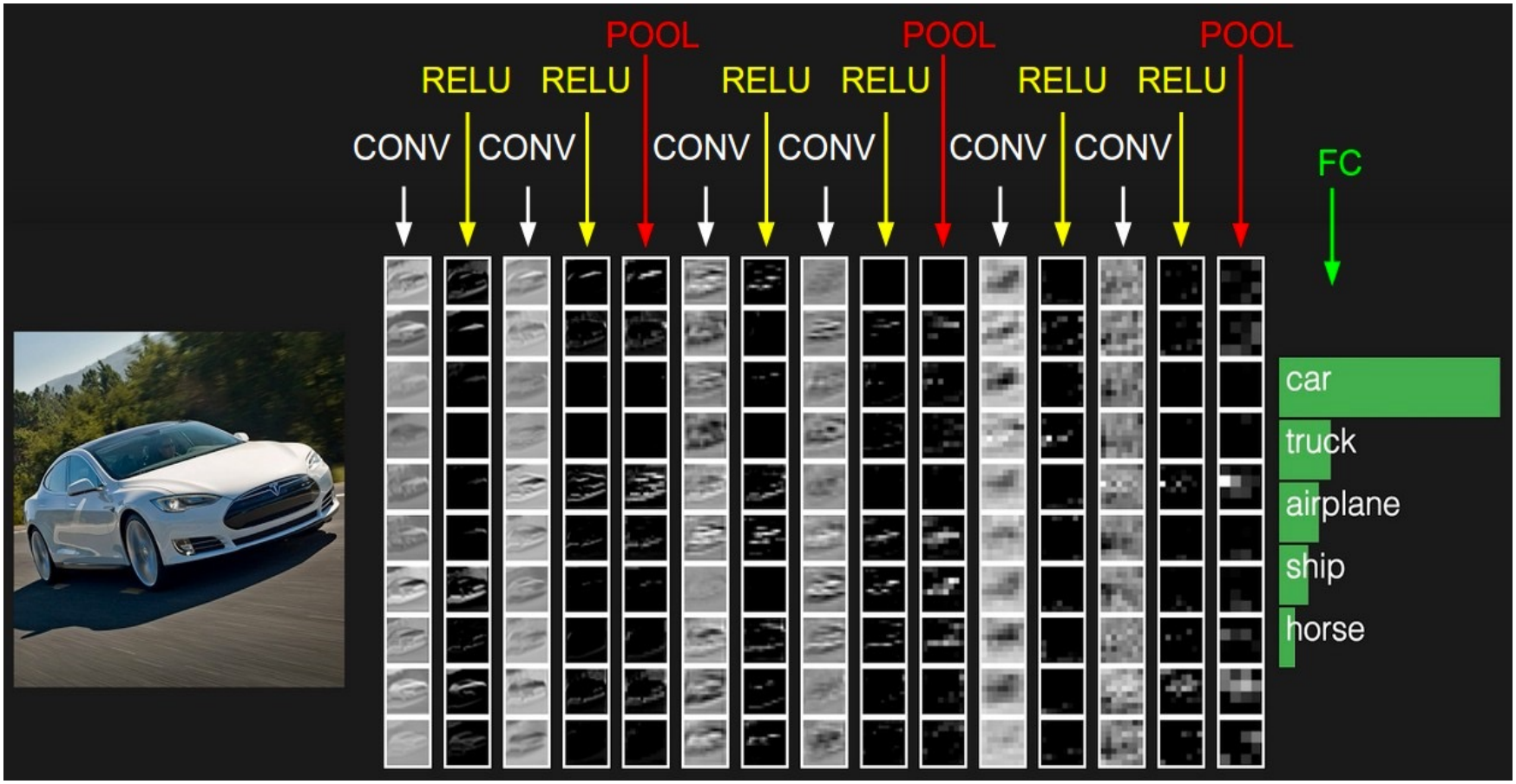


3D-to-3D 3D-to-3D 3D-to-3D

Layers in CNN

- Convolutional layer
  - Size (depth) adjusts mainly according to the number of filters used
  - Size (height x width) adjusts according to stride value
- ReLU (Rectified Linear Unit): e.g. f(x)=max(0,x)
  - Size will be kept the same
- Pooling layer
  - Size (height & width) will be reduced depending on the pooling ratio
- Fully-connected layer
  - Size depends on the architecture
- Input and Output
  - Size of input layer depends on the application (zero padding may apply)
  - Size of output layer depends on the application

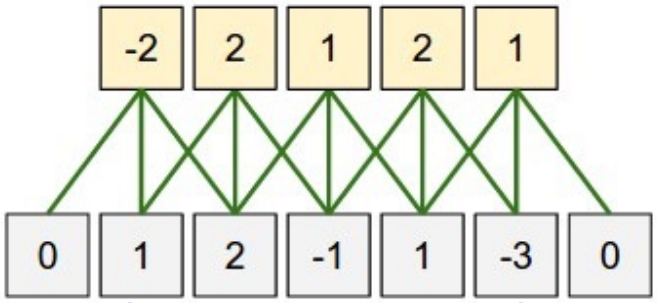
A scenario


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Zero padding and Striding

(1D scenario)

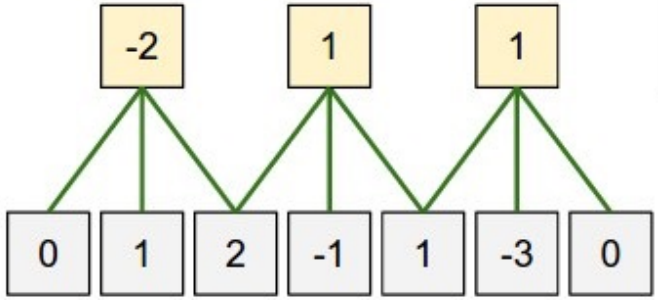
Stride=1



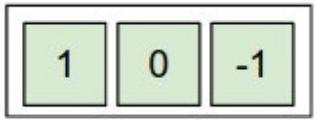
Input: size=1x5

Padding: size=1

Stride=2



1D Filter: 1x3



- Input size: 5
- Padding: 1

A summary

- Conv Layer:
  - Accepts a volume of size $W_1 \times H_1 \times D_1$
  - Requires four hyperparameters:
    - Number of filters K,
    - their spatial extent F,
    - the stride S,
    - the amount of zero padding P.
  - Produces a volume of size $W_2 \times H_2 \times D_2$ where:
    - $W_2 = (W_1 - F + 2P)/S + 1$
    - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
    - $D_2 = K$
  - With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
  - In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Typical hyperparameter values: *F*=3, *S*=1, *P*=1

A summary

- Pooling Layer
  - Accepts a volume of size $W_1 \times H_1 \times D_1$
  - Requires two hyperparameters:
    - their spatial extent F,
    - the stride S,
  - Produces a volume of size $W_2 \times H_2 \times D_2$ where:
    - $W_2 = (W_1 - F)/S + 1$
    - $H_2 = (H_1 - F)/S + 1$
    - $D_2 = D_1$
  - Introduces zero parameters since it computes a fixed function of the input
  - For Pooling layers, it is not common to pad the input using zero-padding.

Typical hyperparameter values: *F*=2, *S*=2 -> size doesn't change!
For *F*=3, *S*=2 -> overlapping pooling!

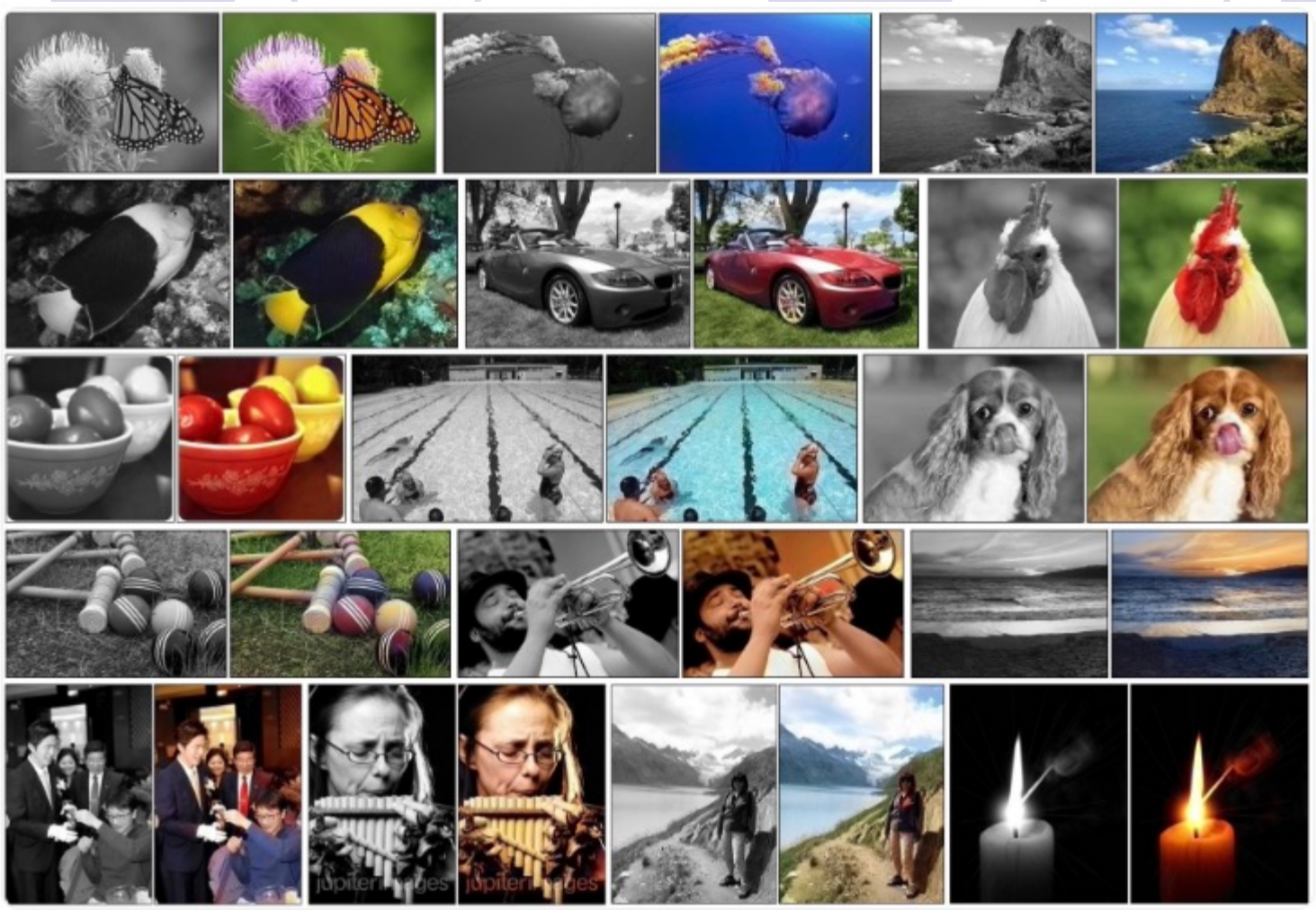
VGGNet Case Study

```
INPUT: [224x224x3]        memory:  224*224*3=150K   weights: 0
CONV3-64: [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*64)*64 = 36,864
POOL2: [112x112x64]  memory:  112*112*64=800K   weights: 0
CONV3-128: [112x112x128]  memory:  112*112*128=1.6M   weights: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]  memory:  112*112*128=1.6M   weights: (3*3*128)*128 = 147,456
POOL2: [56x56x128]  memory:  56*56*128=400K   weights: 0
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*256)*256 = 589,824
POOL2: [28x28x256]  memory:  28*28*256=200K   weights: 0
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]  memory:  14*14*512=100K   weights: 0
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]  memory:  7*7*512=25K  weights: 0
FC: [1x1x4096]  memory:  4096  weights: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]  memory:  4096  weights: 4096*4096 = 16,777,216
FC: [1x1x1000]  memory:  1000 weights: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters
```

Not limited to object classification!

Automatic Colorization of Black and White Images



See this link!

Automatic Colorization of Black and White Images: Some more results

Automatic Colorization of Black and White Images: Problem Formulation

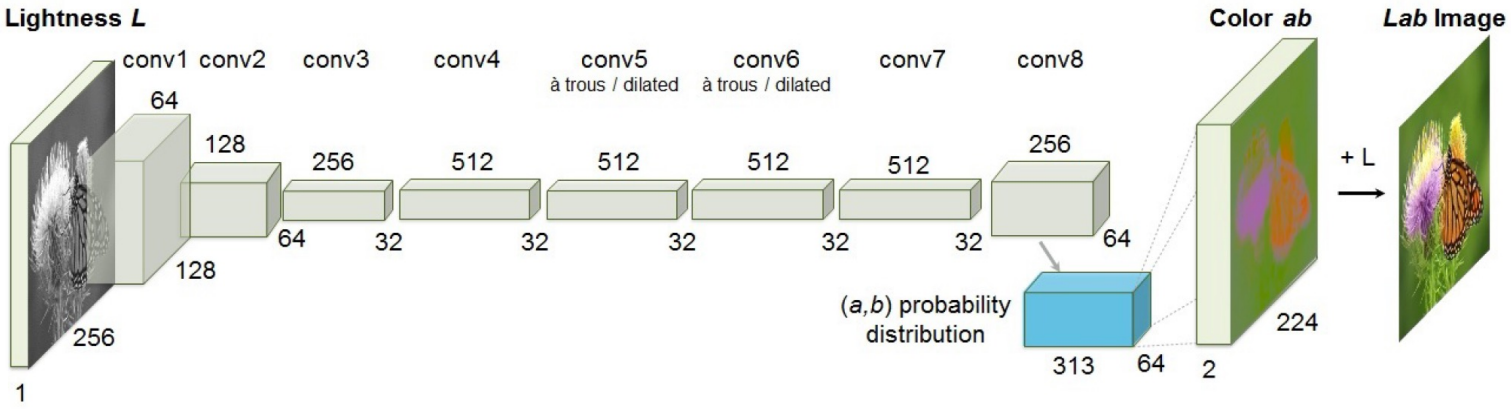
- So, what are the input-output $\{x^t, r^t\}$?

2.1 Objective Function

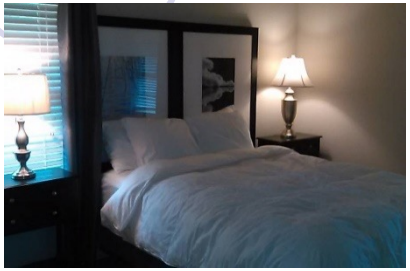
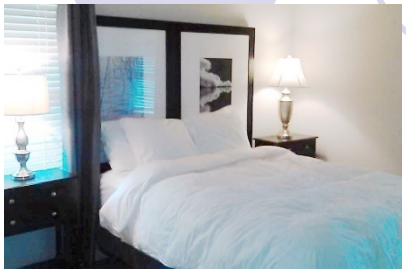
Given an input lightness channel $\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$, our objective is to learn a mapping $\widehat{\mathbf{Y}} = \mathcal{F}(\mathbf{X})$ to the two associated color channels $\mathbf{Y} \in \mathbb{R}^{H \times W \times 2}$, where H, W are image dimensions.

$$\mathrm{L}_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2} \sum_{h,w} \|\mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w}\|_2^2$$

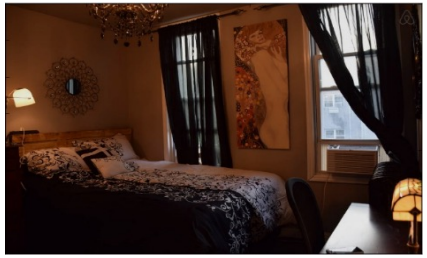
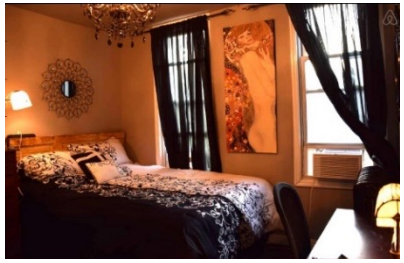
- Architecturally, it has been proposed as



Optimizing Images

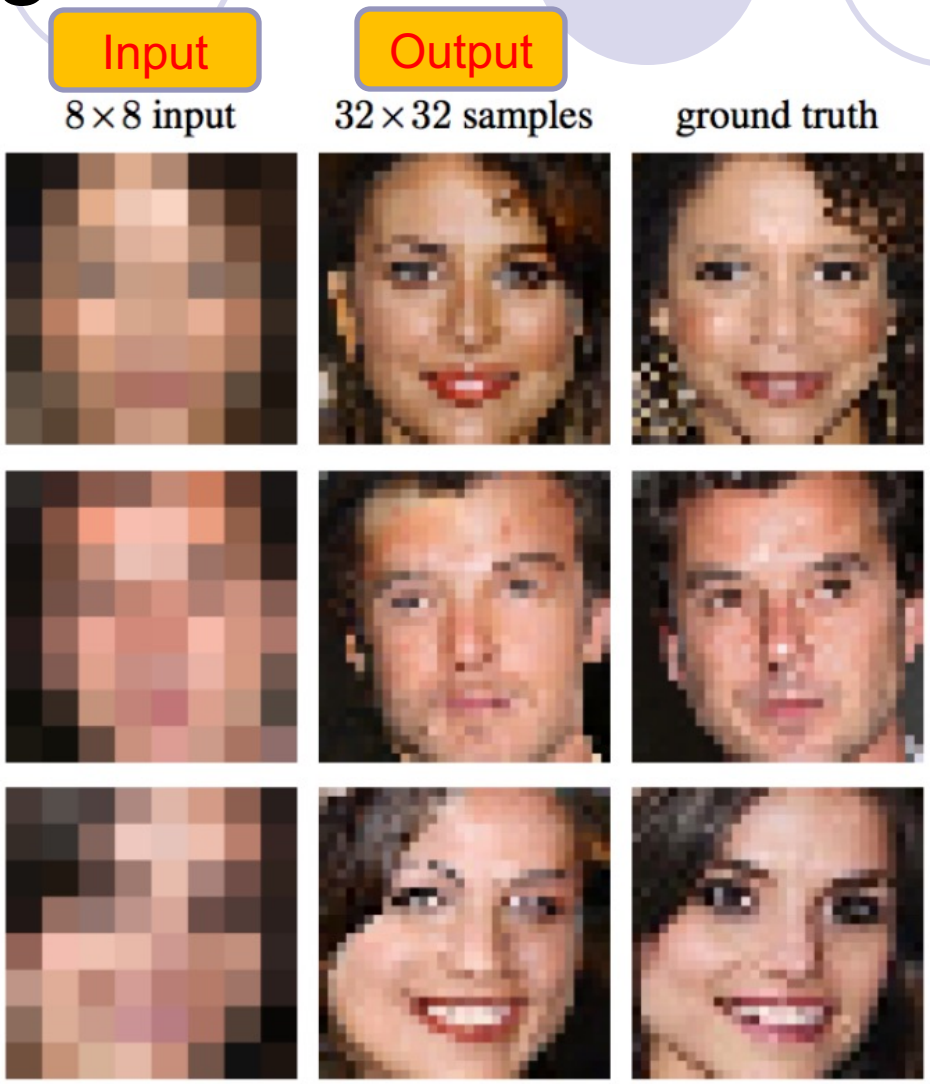
Post Processing Feature Optimization
(Color Curves and Details)

Post Processing Feature Optimization
(Illumination)

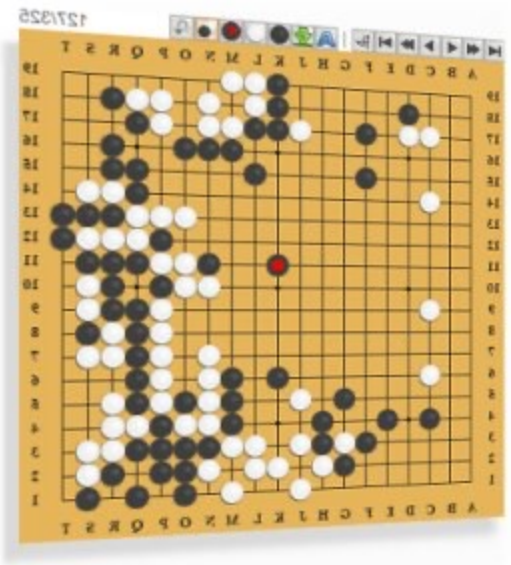

Post Processing Feature Optimization
(Color Tone: Warmness)

Guessing what?



8x8 pixel photos were inputted into a Deep Learning network which tried to guess what the original face looked like. As you can see it was fairly close (the correct answer is under "ground truth").

AlphaGo



19 x 19 matrix

Black: 1

white: -1

none: 0

Neural Network

Next move

(19 x 19 positions)

Fully-connected feedforward network can be used

But CNN performs much better

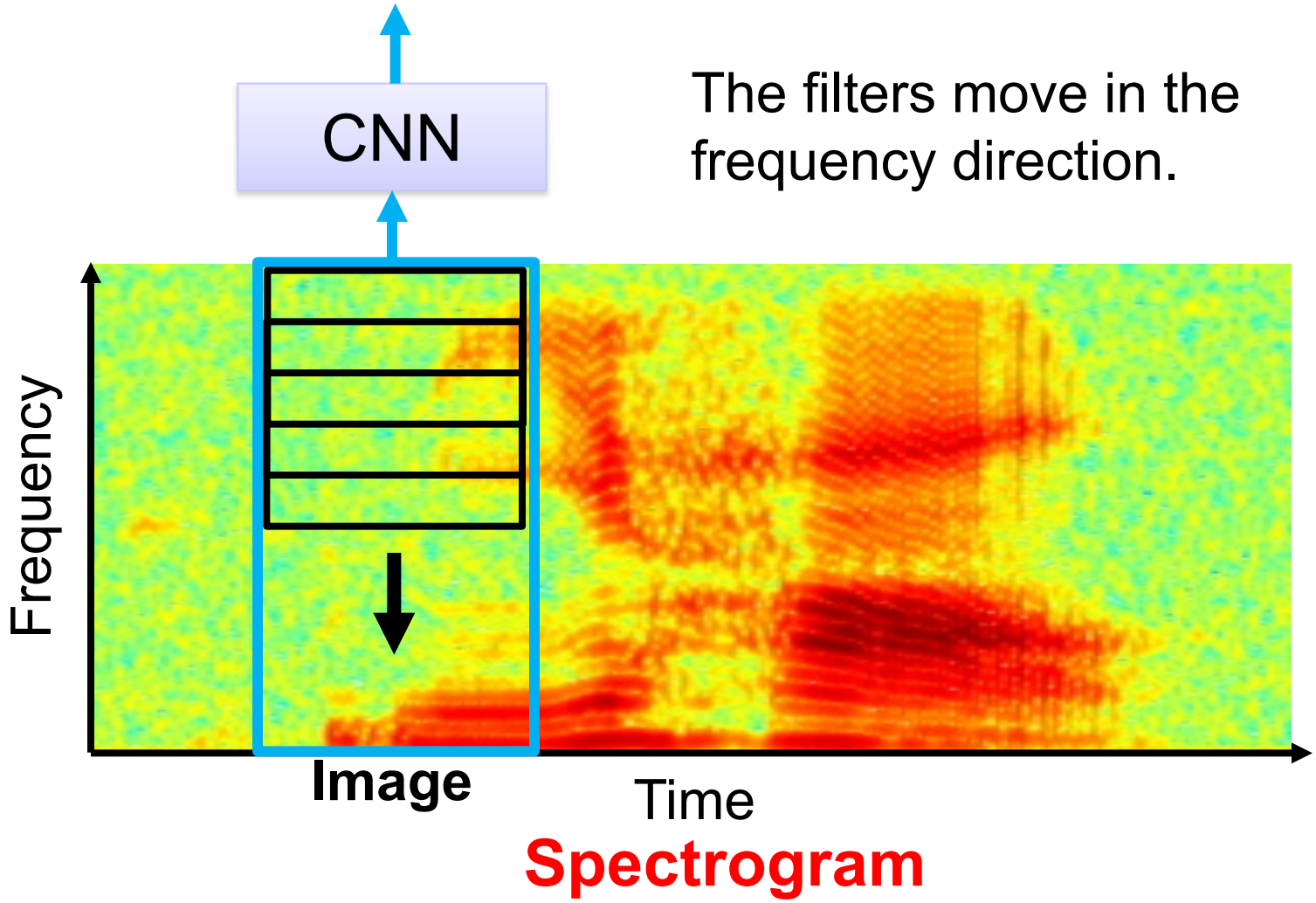
AlphaGo's policy network

The following is quotation from their Nature article:

Note: AlphaGo does not use Max Pooling.

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

CNN in speech recognition



Remarks and Take-home Messages

- Deep learning is deep!
- CNN is only one popular supervised learning DL model.
- Training of a brand new CNN is tough because it involves so many weight parameters (e.g. 138M in VGGNet).
- We adopt, modify and update an existing CNN rather than start with a new one!
- It is really powerful, revolutionizing the traditional ML concepts and opening up tons of impressive applications.
- DL models have been widely supported in different platforms!

Acknowledgement

- Slides from following sources:
  - Ming Li, U of Waterloo
  - Param Vir Singh, Shunyuan Zhang, Nikhil Malik, CMU
  - Qiang Yang, HKUST
  - Goodfellow's slides of the reference book "Deep Learning"
  - https://cs231n.github.io/convolutional-networks
- Images from Google search of Internet