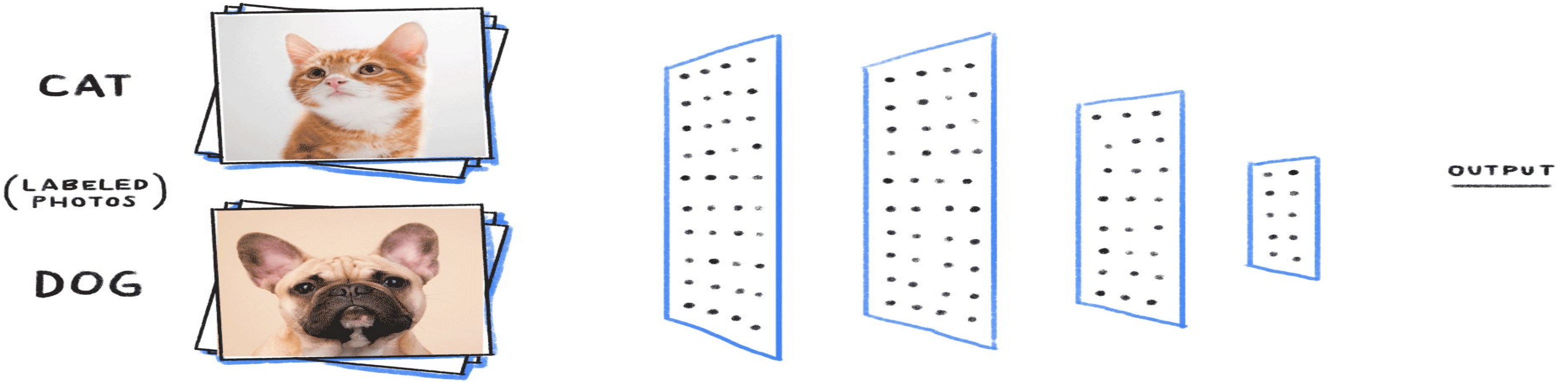
Michael(Mike) Erlihson

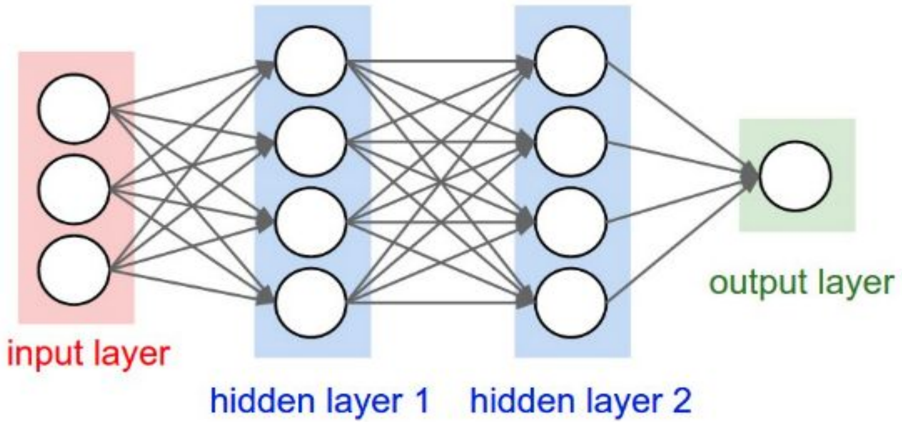


Convolutional Neural Nets: Intro

Agenda

- Neural Nets: Intro
- What is Convolutional Neural Network (CNN)
- CNN Architecture Overview
- Convolution Operation in CNN
- Typical CNN Layers
- CNN Advantages
- CNN Applications

Neural Network: Basics



Input example : one image

Output example : one class

airplane	dog
automobile	frog
bird	horse
cat	ship
deer	truck

What is CNN?

- A convolutional neural network (ConvNet, CNN) is a type of feed-forward artificial neural network

- The architecture of a ConvNet is designed to take advantage of the 2D structure of an input image.

- A CNN is comprised of one or more convolutional layers (plus pooling layer = subsampling) followed by fully connected layer/s

Motivation behind CNN: Why bother?

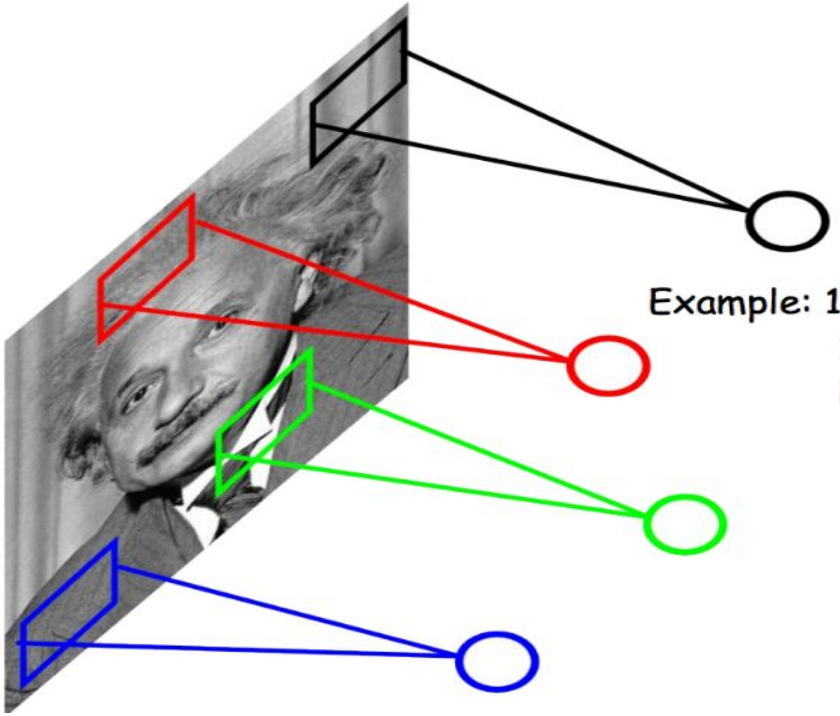
- For image of size 200x200x3 (3 color channels, 120K pixels in total)
  - Fully-connected(FN) NN: each neuron in a 1st hidden layer has **120K weights**
  - 1000 neurons on a first hidden layer: **120 Million(!!)** weights

Wasteful and leads to overfitting (due to a huge number of parameters)

- **CNN:** neurons connected only to a small region of the previous layer
- **Why:** Effectively exploits dependencies between nearby image pixels
- **Is translation-invariant**: "dog" looks the same whenever it is located in an image

CNN: General Idea

Reduce connection to local regions

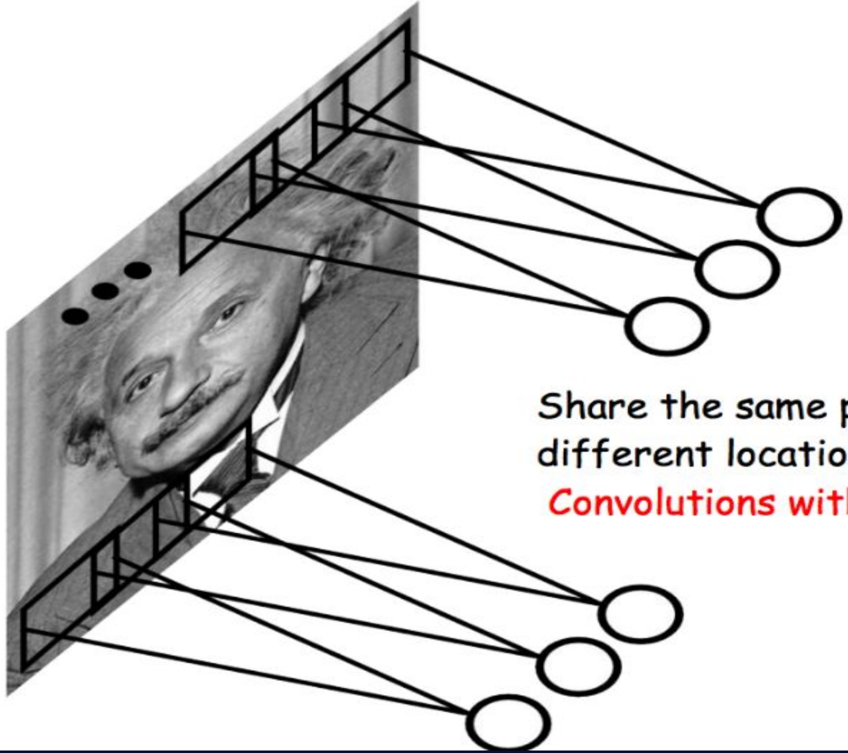


Example: 1000x1000 image
1M hidden units
Filter size: 10x10
10M parameters

67
Ranzato

CNN: General Idea

Reuse the same kernel everywhere



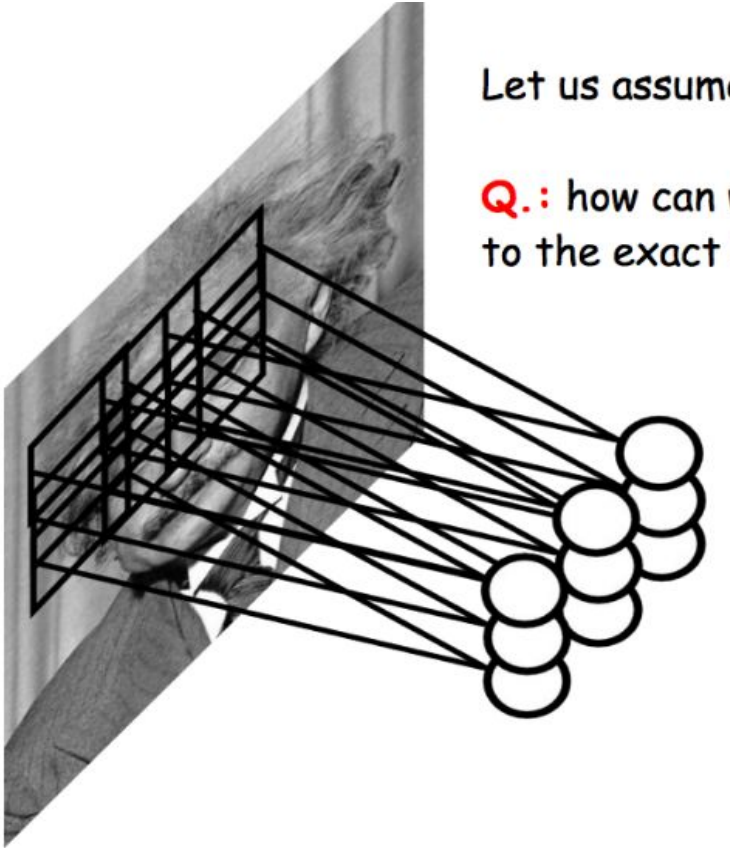
Because interesting features (edges) can happen at anywhere in the image.

Share the same parameters across different locations:
Convolutions with learned kernels

69

Ranzato

Translation invariance(location robustness)

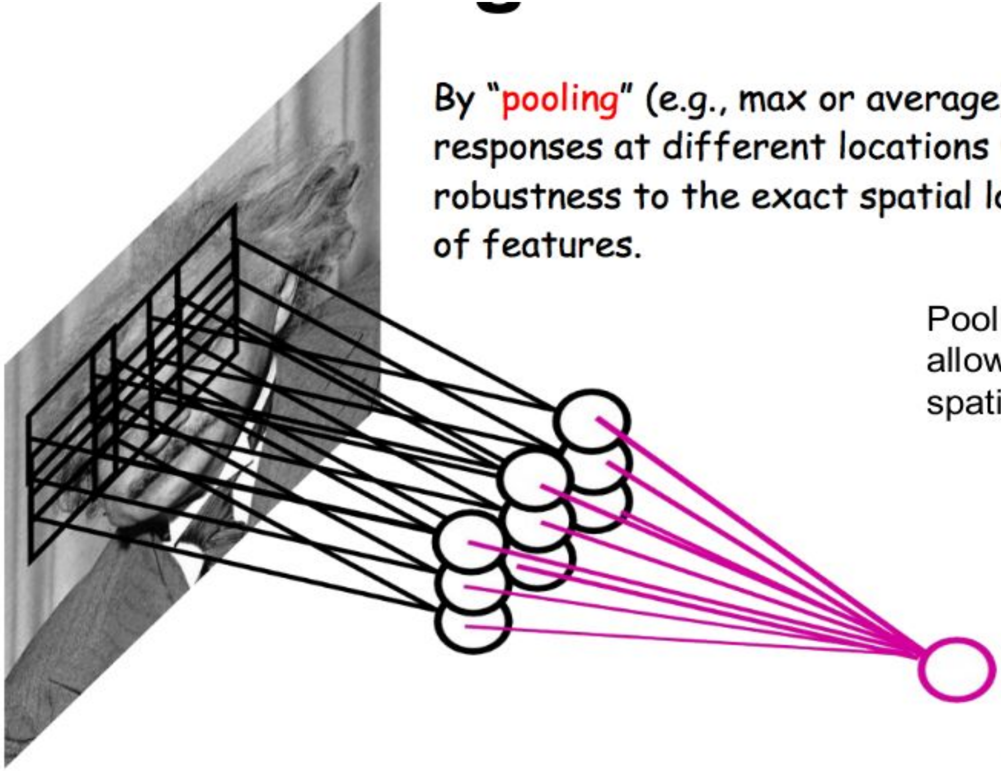


Let us assume filter is an "eye" detector.

Q.: how can we make the detection robust to the exact location of the eye?

Ranzato

Translation Invariance via Max Pooling



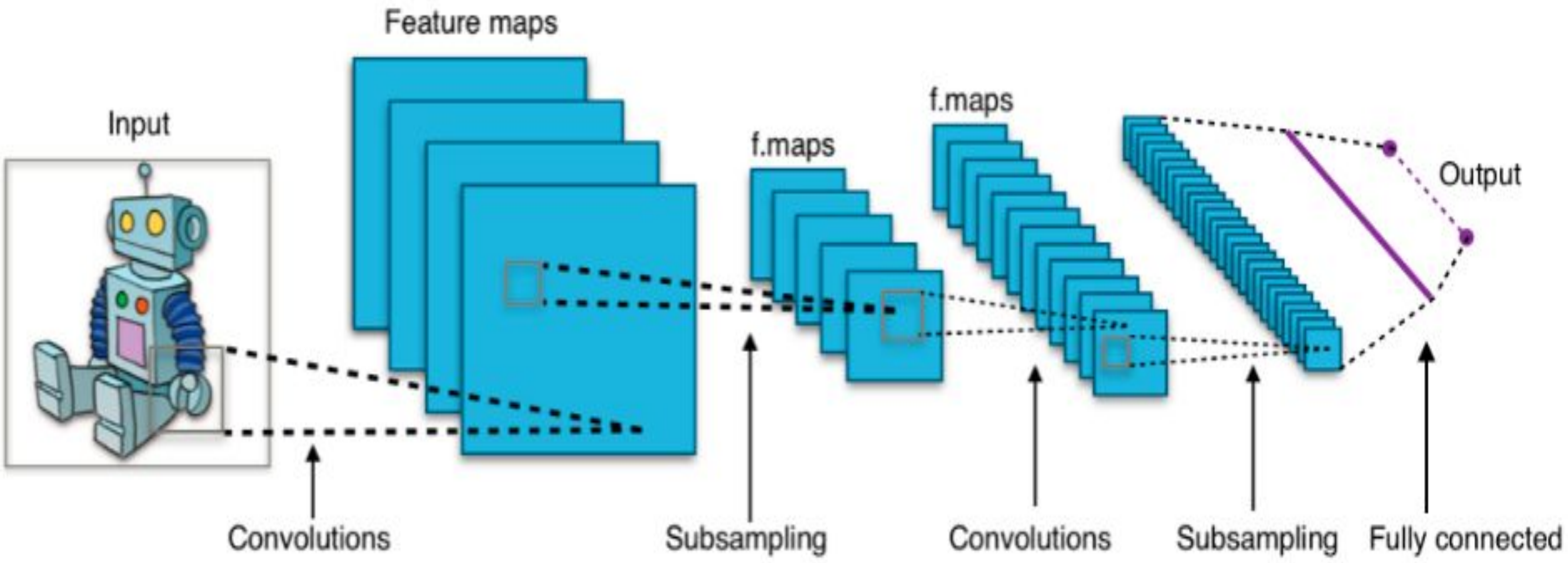
By "pooling" (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.

Pooling also subsamples the image, allowing the next layer to look at larger spatial regions.

73

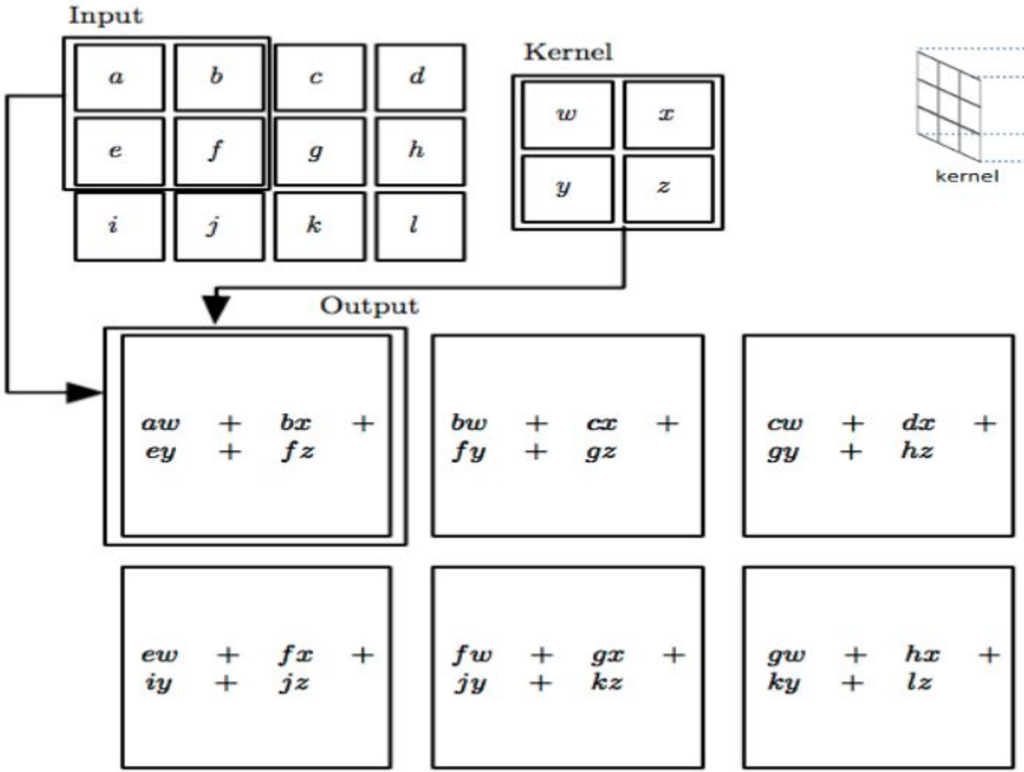
Ranzato

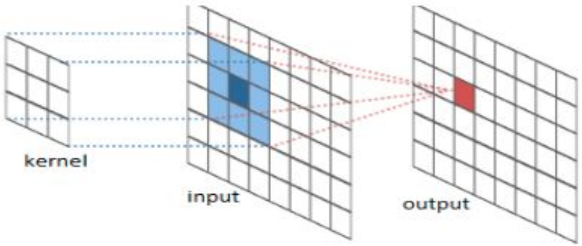
Typical CNN architecture



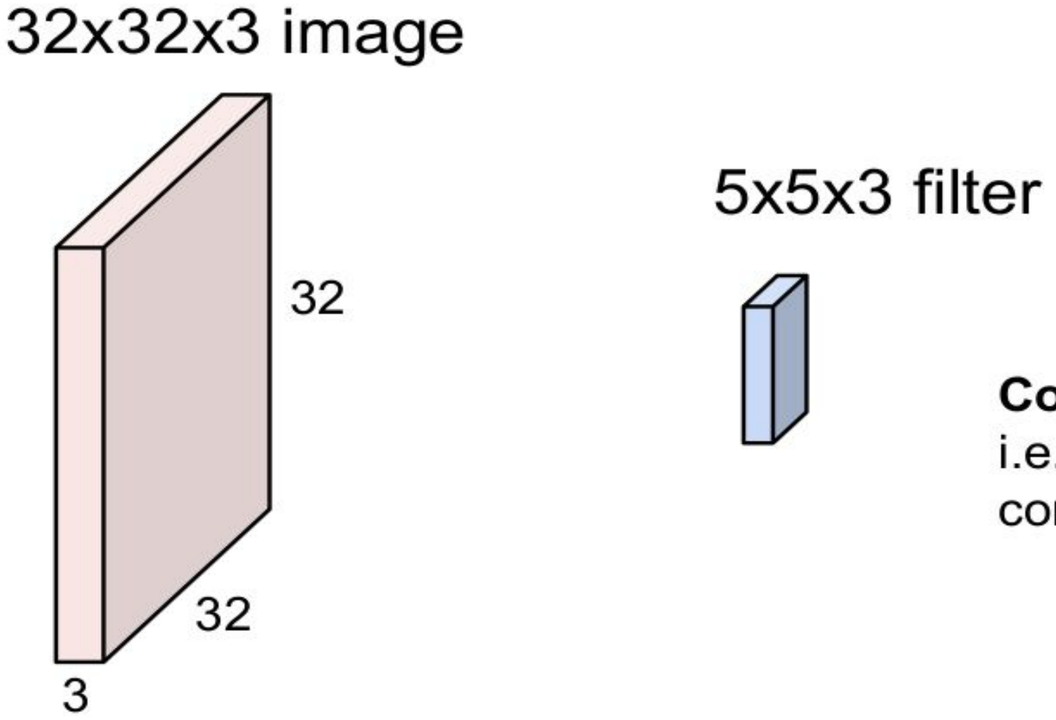
Don’t worry: You’ll learn all this stuff soon...

CNN Core: Convolution Operation





Convolution Layer

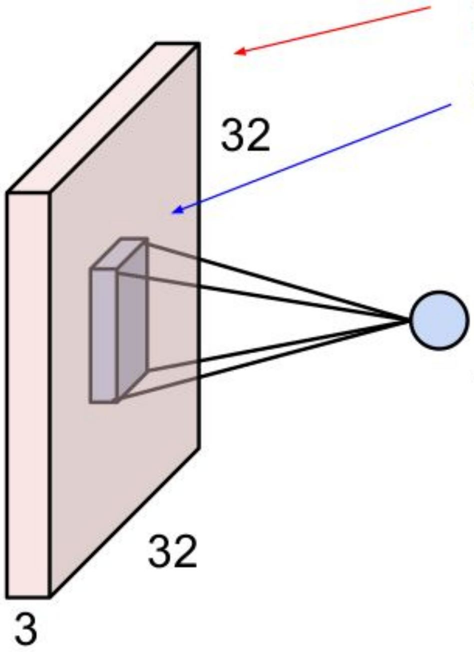


Convolve the filter with the image
i.e. "slide over the image spatially,
computing dot products"

Convolutional Layer: Cont’

32x32x3 image

5x5x3 filter w

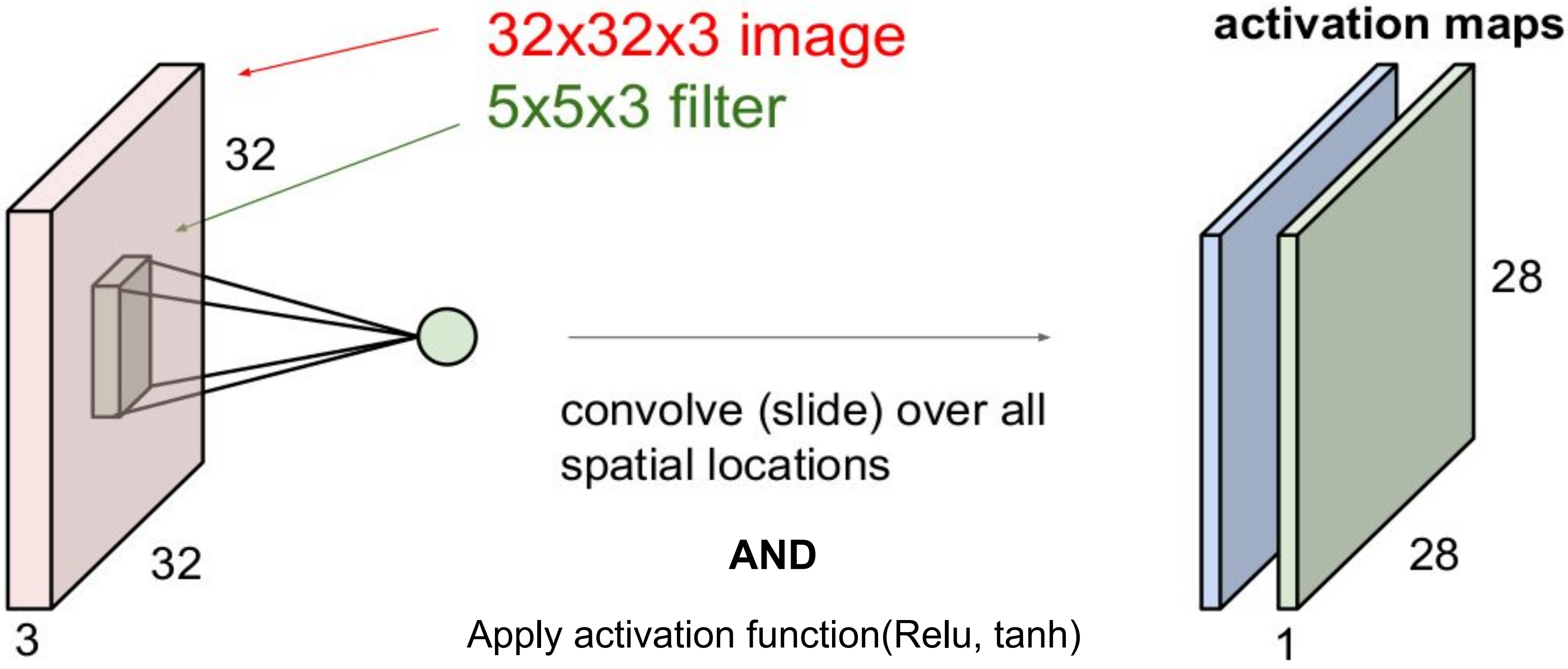


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

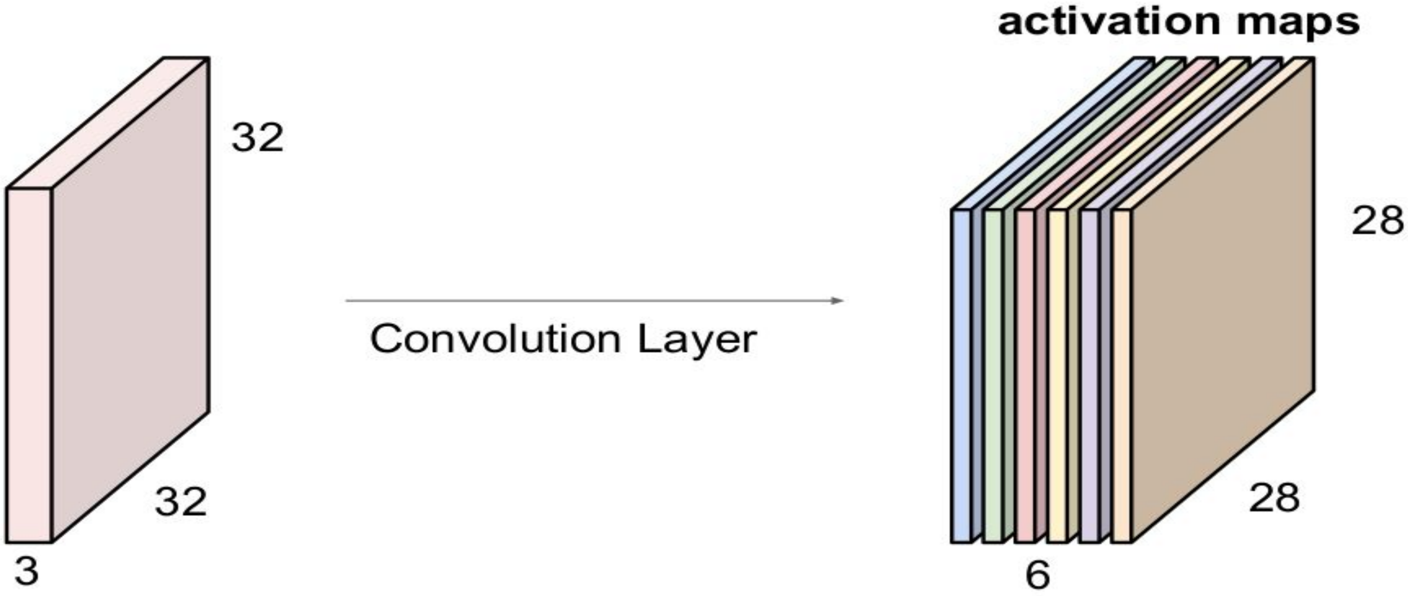
Convolutional Layer: Many filters(kernels)

Consider a second, green filter



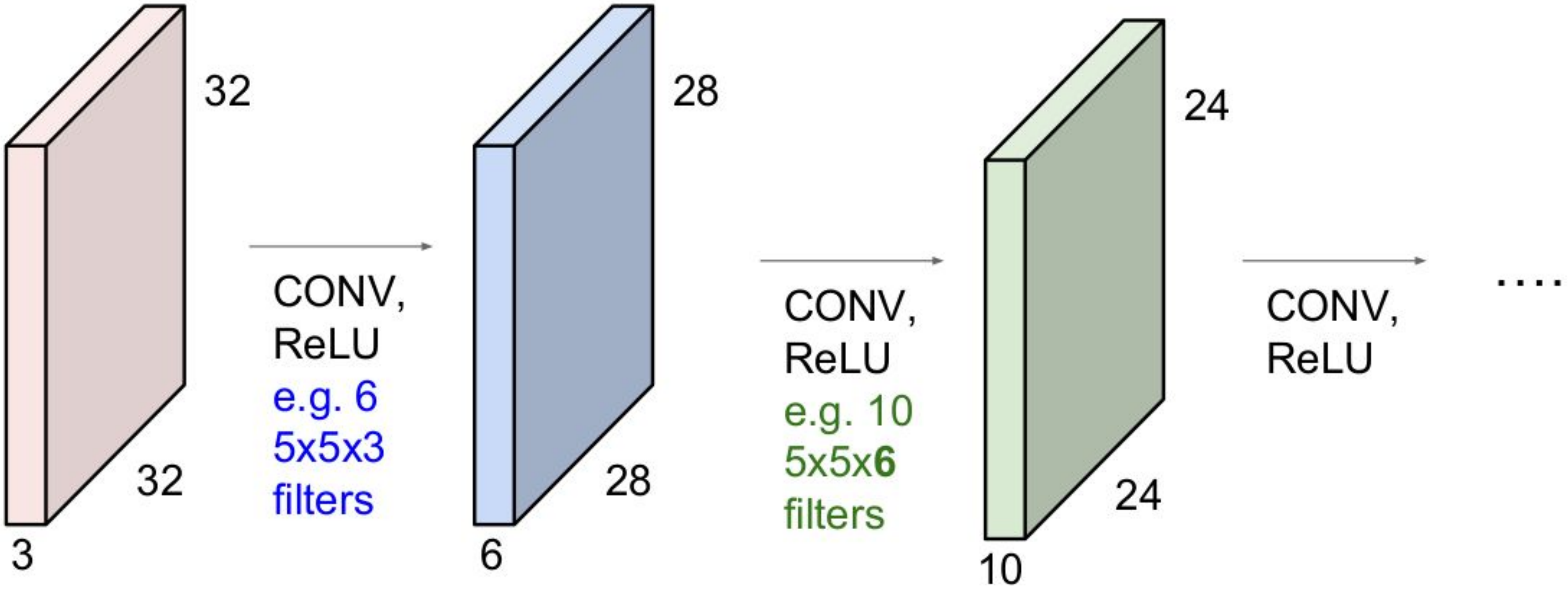
Convolutional Layer: Many filters, Cont’

If we had 6 5x5 filters, we’ll get 6 separate activation maps



We stack these up to get a “new image” of size 28x28x6!

ConvNet At Glance



Convolution operation on Image

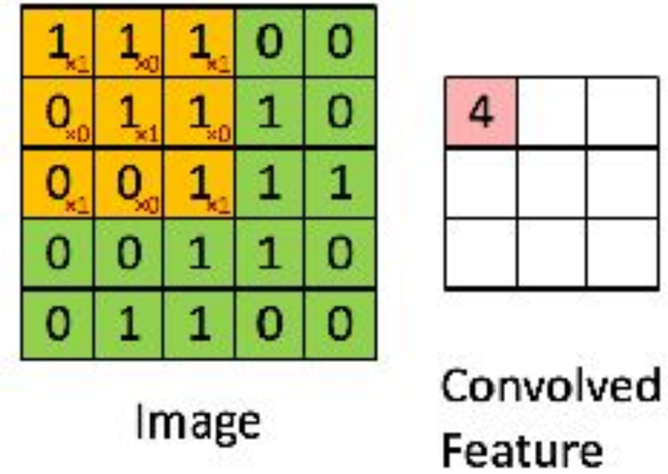
Image

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

Filter

1	0	1
0	1	0
1	0	1

Operation



- The Kernel shifts 9 times because of **Stride Length=1**(Non-Strided)
- Every time performs a **element-wise multiplication between K and the portion Pof the image** over which the kernel is hovering.

3D Convolution Filter

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$$308 \quad + \quad -498 \quad + \quad 164 \quad +1 = -25$$

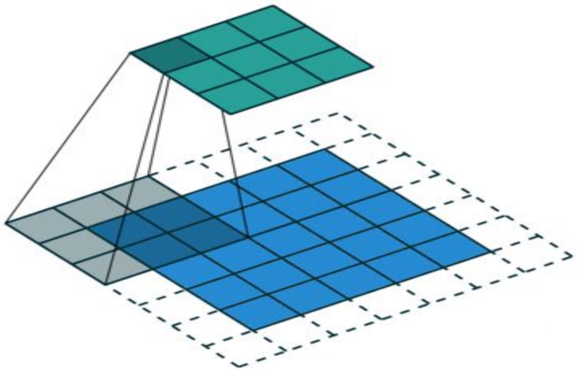
Bias = 1

Output

-25				...
				...
				...
				...
...	...	...	...	...

Conv Layers with Stride > 1

3 x 3 convolution with stride = 2



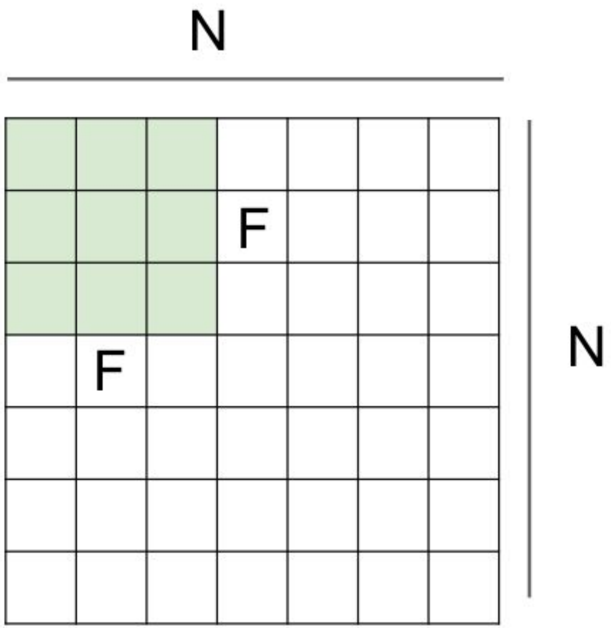
Remember:

Output size = (Input Size - Filter Size) / stride + 1

- Convolution with stride size > 1 reduces the convolved feature size by the stride size (compared to the input size)

- May serve as a replacement to pooling layers (e.g. GANs)

Padding: Why?



Output size:

(N - F) / stride + 1

e.g. N = 7, F = 3:

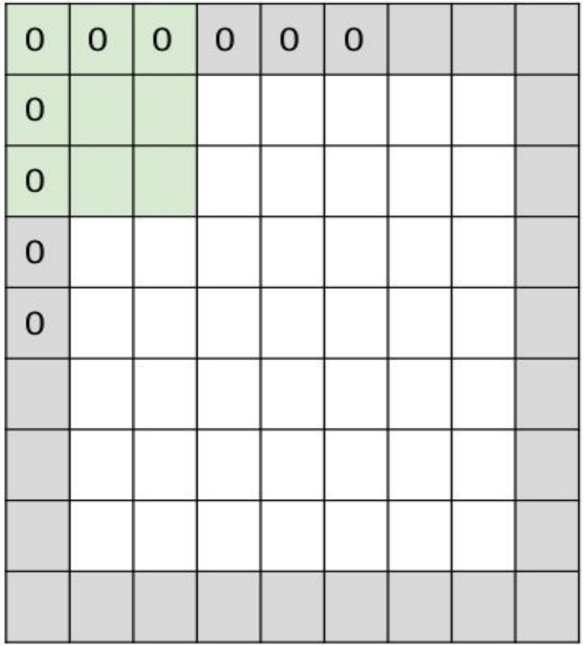
stride 1 => (7 - 3)/1 + 1 = 5

stride 2 => (7 - 3)/2 + 1 = 3

stride 3 => (7 - 3)/3 + 1 = 2.33

It seems we have a problem here

Zero padding



Input size: 7x7

Filter: 3x3 with stride **1**

Pad with 1 pixel border => what is the output?

Output: 7 x 7

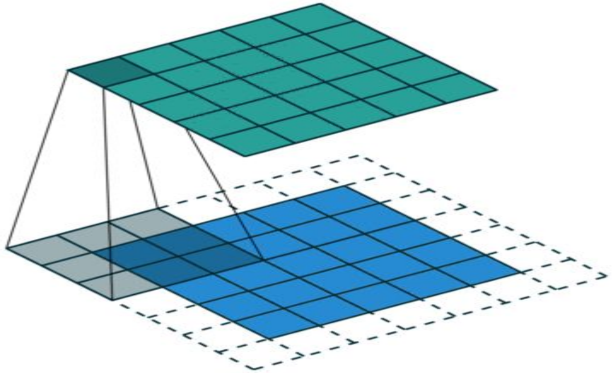
CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2 (will preserve size spatially)

F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Padding in ConvNets(Zero Padding)

- **Valid Padding:** convolved feature is reduced in dimensionality compared to input

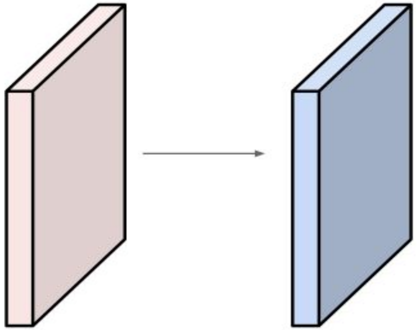
- **Same Padding**: convolved feature remains the same size(rarely increased)



SAME padding: 5x5x1 image is padded with 0s to create a 6x6x1 image

Output size computation: Example

Examples time:



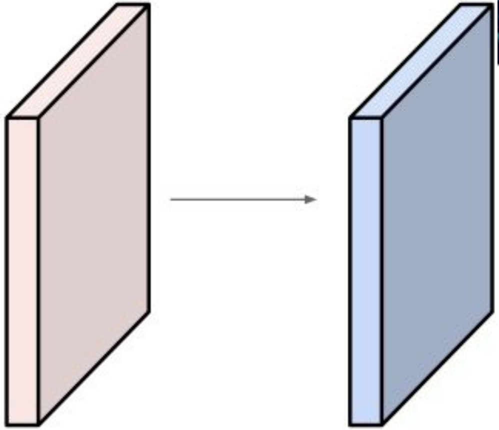
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:

(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Number of weights per layer: Example

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**

Conv Layer Weights: Cheat Sheet

Summary. To summarize, the 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.

Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)

- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

IMPORTANT SLIDE

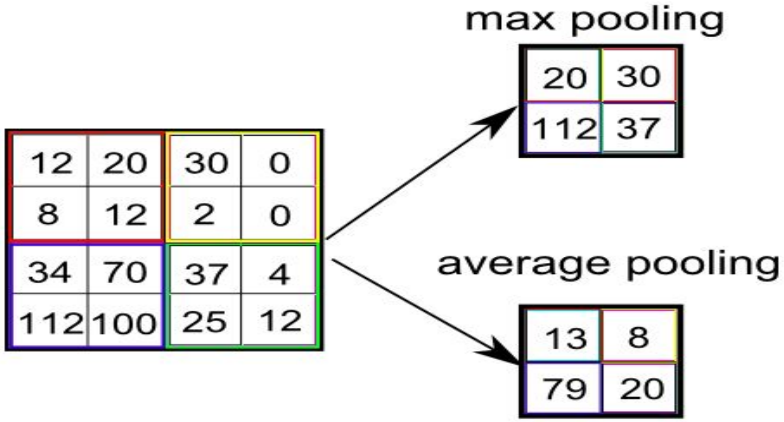
Introducing NonLinearity

- An additional operation called ReLU has been used after every Conv Operation
- ReLU mostly used in ConvNets (but GANs like Leaky ReLu)
- Tanh used for the last layer for the generation ConvNets
- Sigmoid(softmax) is used in classification/segmentation ConvNets (last layer)

Pooling Layers

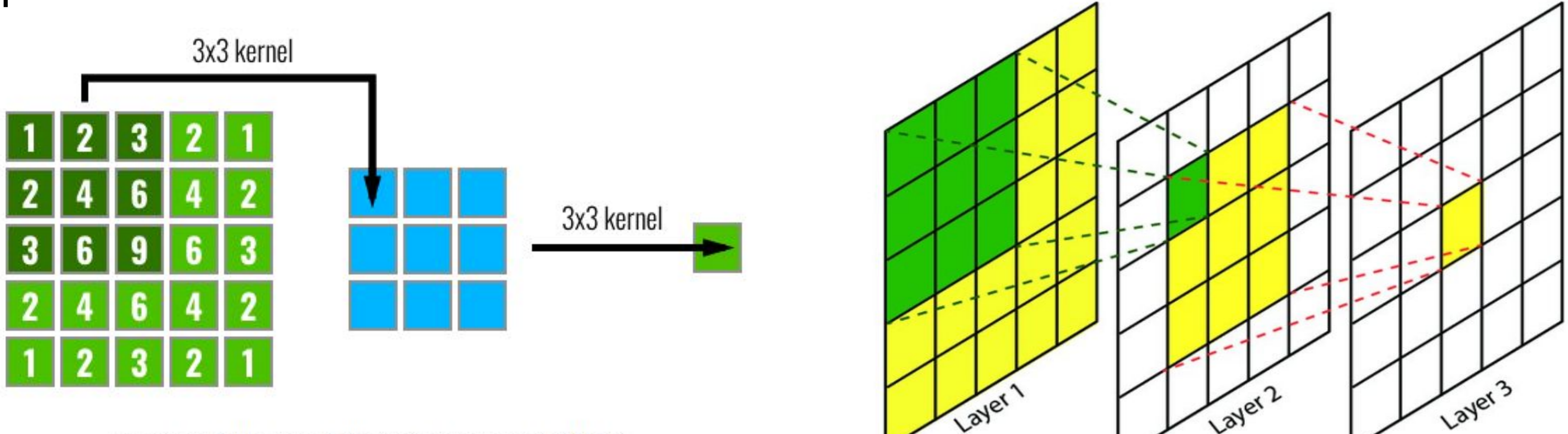
- Spatial Pooling (subsampling, downsampling) reduces the dimensionality of feature map but retains the most important information.

- Pooling can be of different types: Max, Average, Sum etc.



Receptive field:

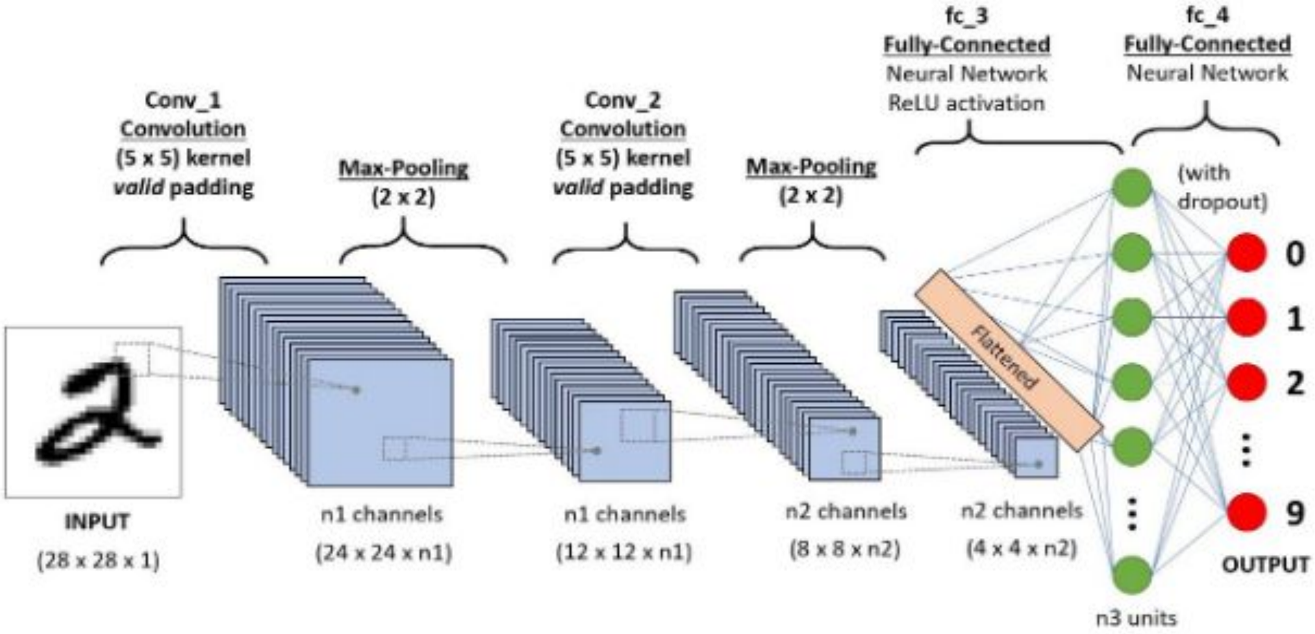
The receptive field in CNN the region of the input space that affects a particular unit of the network.



Receptive Field across 3 different layers using 3×3 filters.

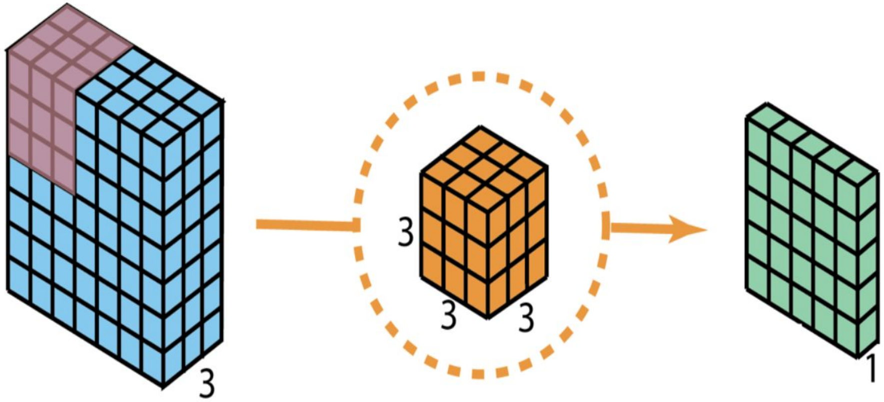
Important: This input region can be not only the input of the network but also output from other units in the network

Putting All Together: Simple ConvNet for MNIST

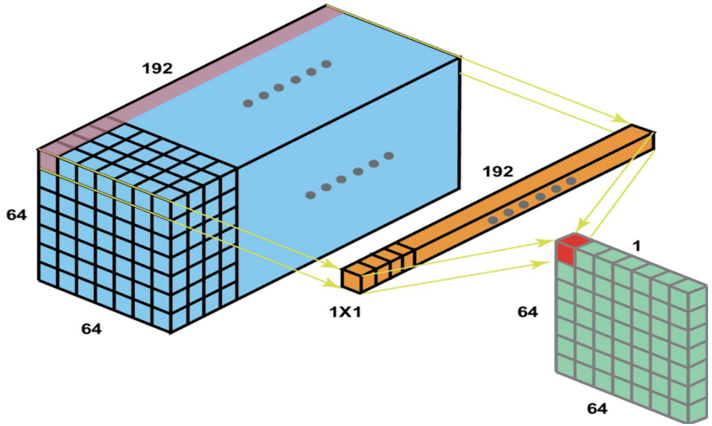


Popular Convolution Types: 1x1 conv

Convolve over “channel dimension”



“Standard” Convolution



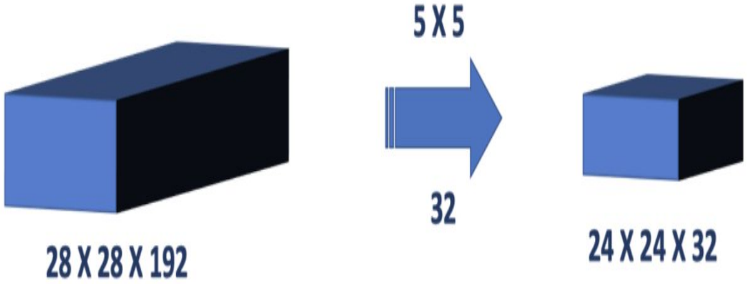
1x1 Convolution

Reduce number of channels while introducing nonlinearity

[Network in network, 2014]

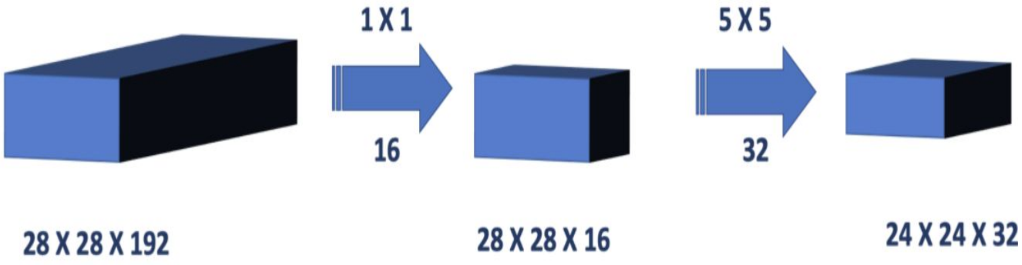
1x1 convolution: Dimensionality Reduction

Significant Computational load reduction



Number of Operations : (28X28X32) X (5X5X192) = 120.422 Million Ops

"Standard" Convolution

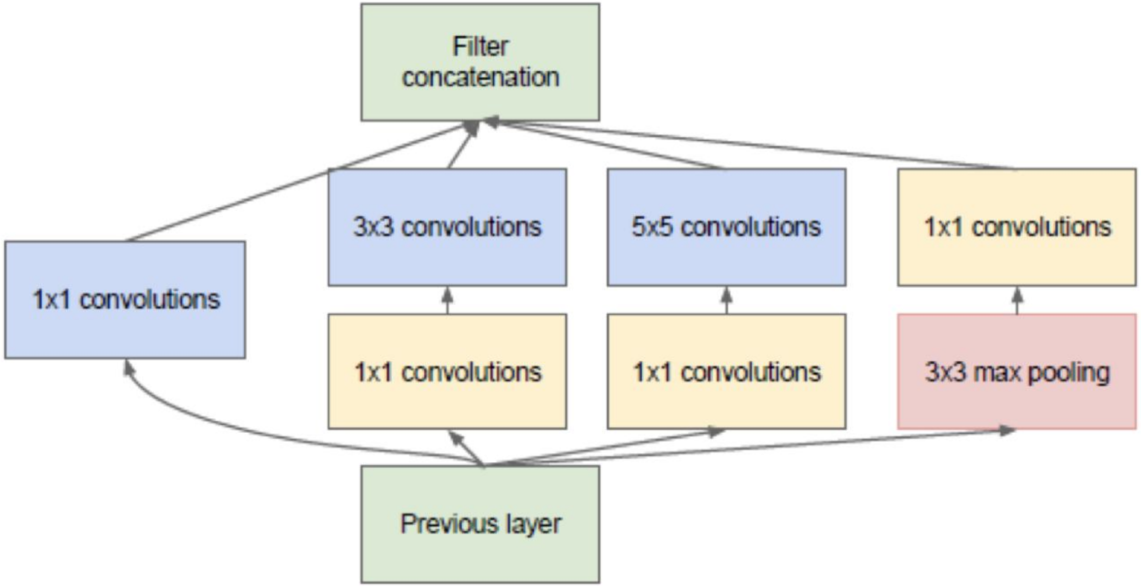


Number of Operations for 1 X 1 Conv Step : (28X28X16) X (1X1X192) = 2.4 Million Ops
Number of Operations for 5 X 5 Conv Step : (28X28X32) X (5X5X16) = 10 Million Ops
Total Number of Operations = 12.4 Million Ops

1x1 Convolution

Computation load is reduced by a factor of 10 (!!)

1x1 Convolution Usage: GoogleNet(2014)



(b) Inception module with dimensionality reduction

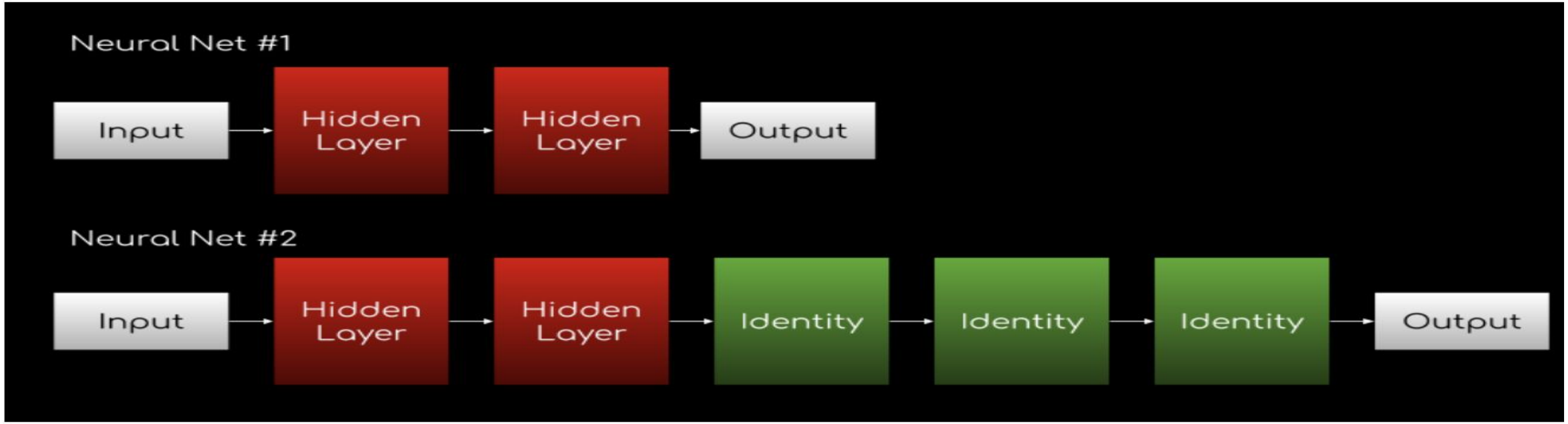
DiCaprio’s “We need to go DEEPER” in the movie Inception

1x1 conv, usage 2: Going DEEPER Network (ResNet)

What is the problem with very deep networks?

- **Theoretically**, more layers added to a neural net, the performance could either go UP or stay the SAME it should <u>never go down</u>

- Therefore, to make a neural network better, just add more layers!

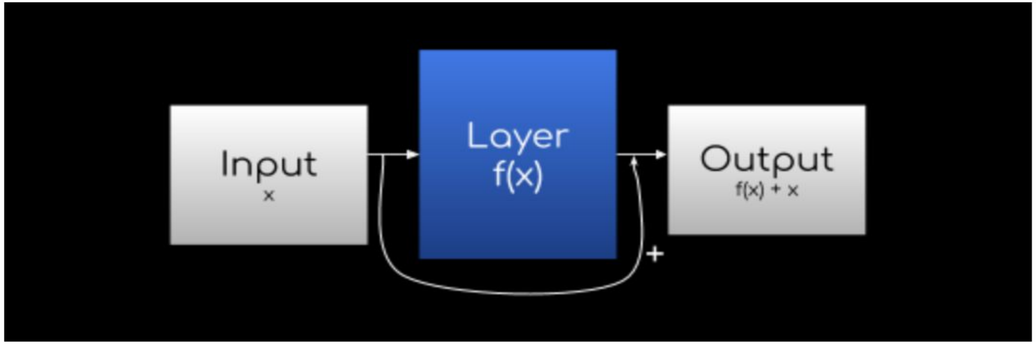


Very Deep Neural Nets Shortcoming

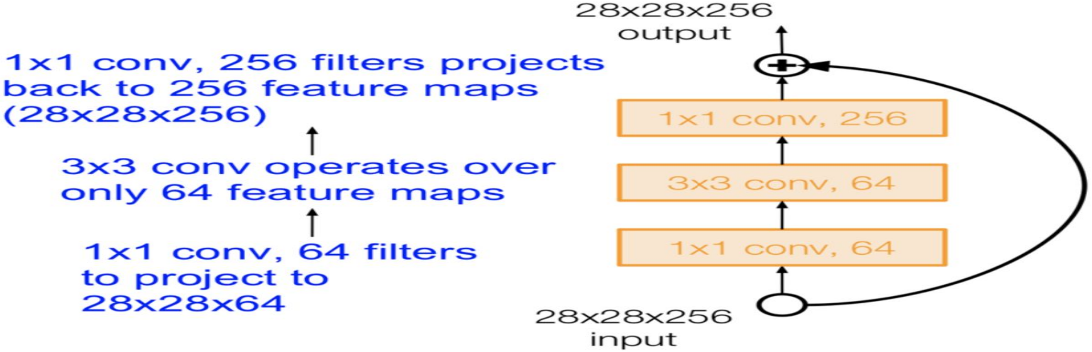
Suppose **Neural Net #1** has achieved 100% accuracy and its loss function is at the global minimum; in other words, the neural network is in its *best possible state*. Now as you add more hidden layers as seen in **Neural Net #2**, theoretically the new layers should learn the *identity function* (mapping the input directly to the output, e.g. $g(x) = x$) to preserve the current *best possible state* of the network — these 2 networks are essentially equivalent.

However, **experimentally**, learning the identity function is extremely difficult as the scope of all possible combination of weights and biases is enormous, thus the chance of learning the identity function is minuscule.

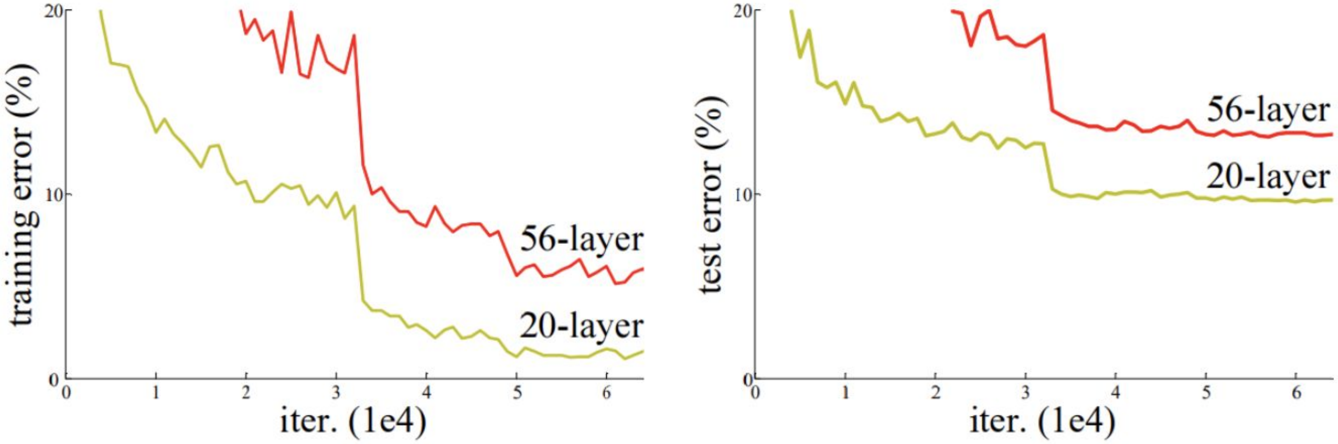
Very Deep Neural Nets: ResNet with 1x1



*f(x) does not have to be one hidden layer, could be any arbitrary number of hidden layers



Very Deep Neural Nets Shortcoming, Cont



Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

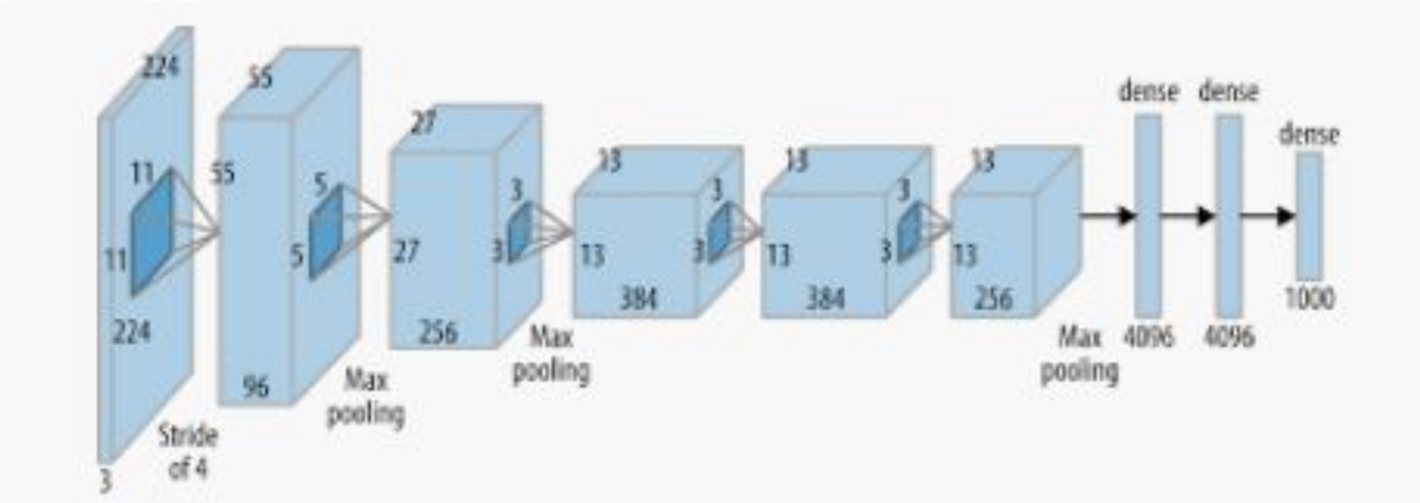
More about Conv Nets

- **Training:** back-propagation, all fancy optimizers(Adam, RMSProp etc)

- **Regularization**: various types of dropout, L2, L1

- **Loss**: Depends on the task
  - Cross-entropy for classification
  - Cross-entropy combined with L2 loss for object segmentation
  - L1 loss sometimes used for image inpainting
  - Adversarial loss (GANs)
  - And many others…

Making deeper ConvNets: AlexNet



Going Even Deeper

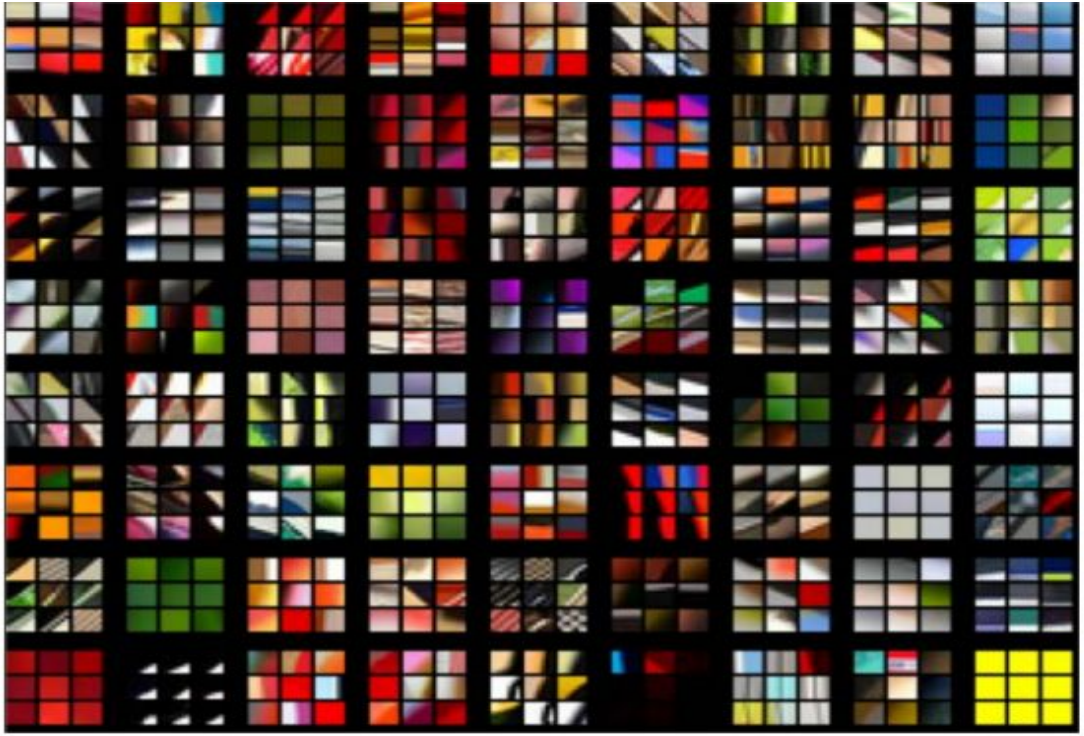
ConvNets in Keras: Simple example

```
model = Sequential([
    Conv2D(32, kernel_size=(3, 3),
                  activation='relu',
                  input_shape=input_shape),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
```

Now cool stuff about ConvNets is Coming

What all these Conv Layers are detecting?

Top 9 patches that activate each filter in layer 1:Alexnet



Each 3x3 block show the top 9 patches for one filter.

What all these Conv Layers are detecting?

What all these Conv Layers are detecting?

What all these Conv Layers are detecting?

What all these Conv Layers are detecting?

What all these Conv Layers are detecting?

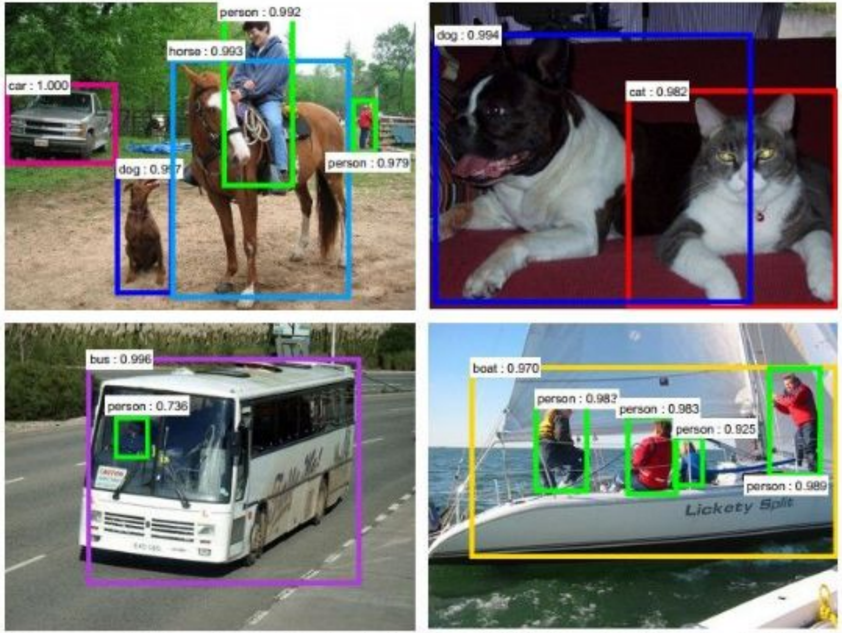
ConvNets are Everywhere

Classification

Retrieval

ConvNets are Everywhere 2

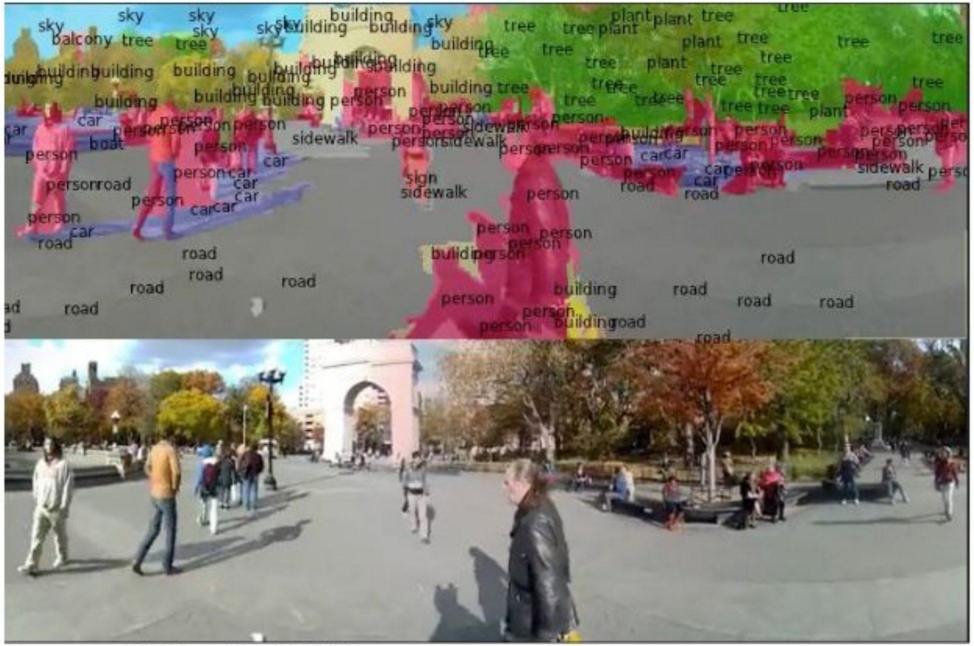
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



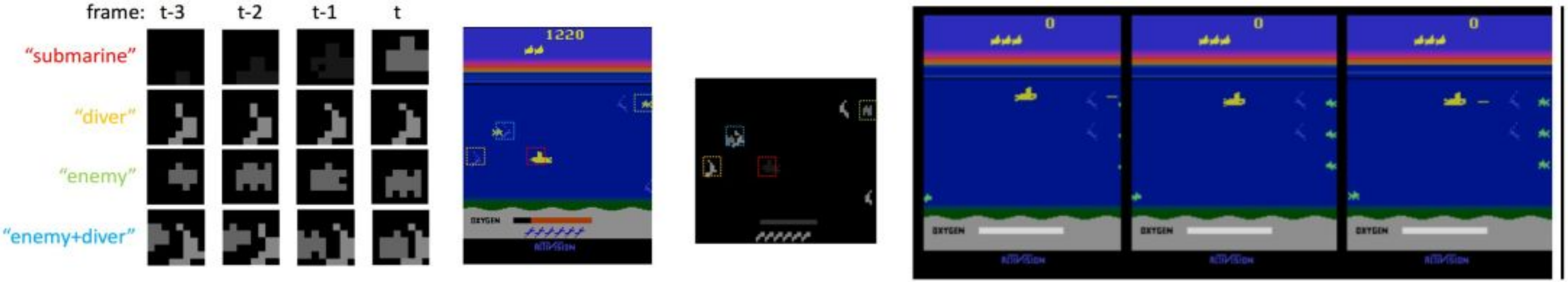
Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

ConvNets are Everywhere 3

Images are examples of pose estimation, not actually from Toshev & Szegedy 2014. Copyright Lane McIntosh.

[Toshev, Szegedy 2014]



[Guo et al. 2014]

Figures copyright Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard Lewis, and Xiaoshi Wang, 2014. Reproduced with permission.

ConvNets are Everywhere 4

Image Captioning

[Vinyals et al., 2015]
[Karpathy and Fei-Fei, 2015]

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

Minor errors

A man in a baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

Somewhat related

A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

Captions generated by Justin Johnson using Neuraltalk2

Topics Not Covered :(

- Other convolution filter types(dilated, atrous, non-rectangular, coordConv)
- How to use ConvNets to extract power image representation in a low-dimensional space (self-supervised learning)
- Interesting ConvNets architectures optimized for a wide range of tasks
- Use of ConvNets to video processing
- How ConvNets are used for performance assess… ative networks

This is only what came to my mind

Time for Q&A

INT

המרכז הבינלאומי
ללימודי הייטק וחדשנות

!Thanks

INT

המרכז הבינלאומי
ללימודי הייטק וחדשנות