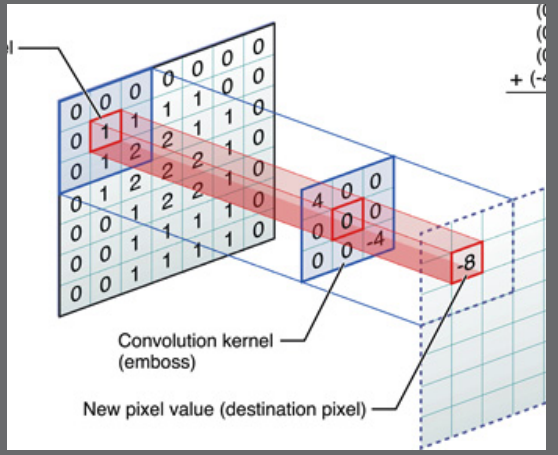
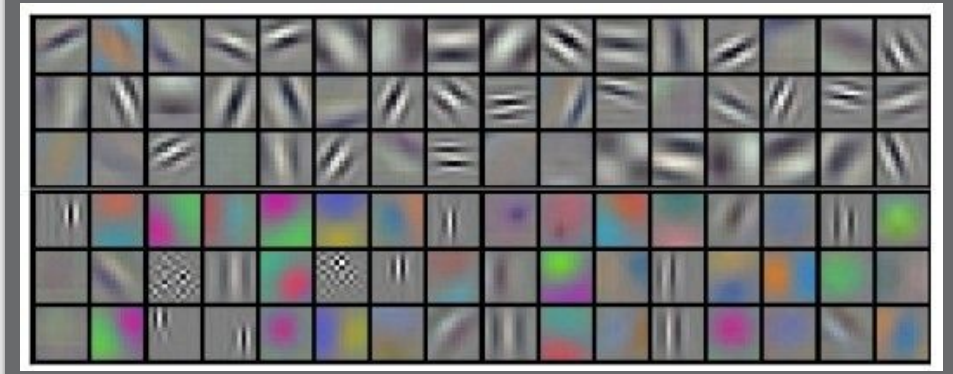
Convolutional Neural Networks

Taryn Heilman

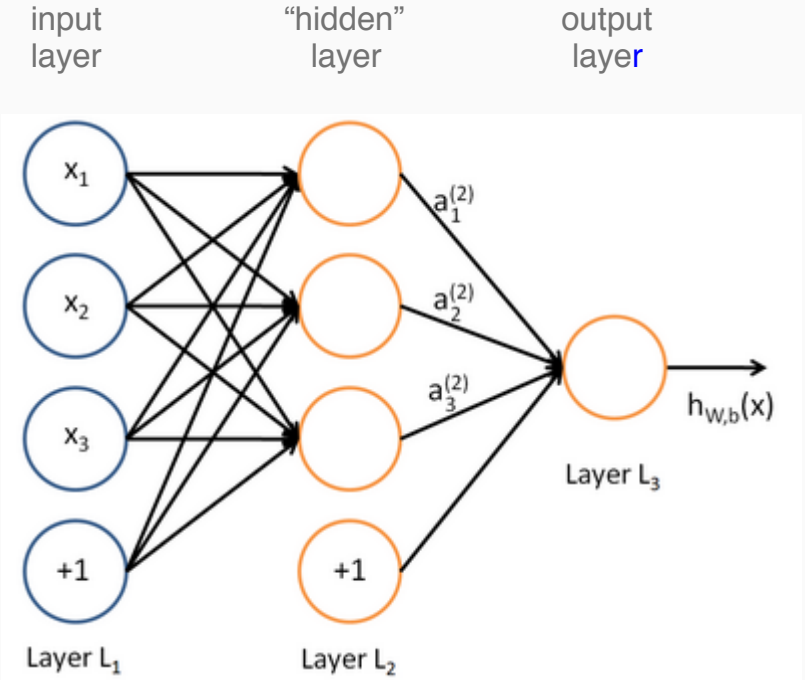
Substantial portions borrowed from:
- Andrej Karpathy, et al.
-Jon Courtney

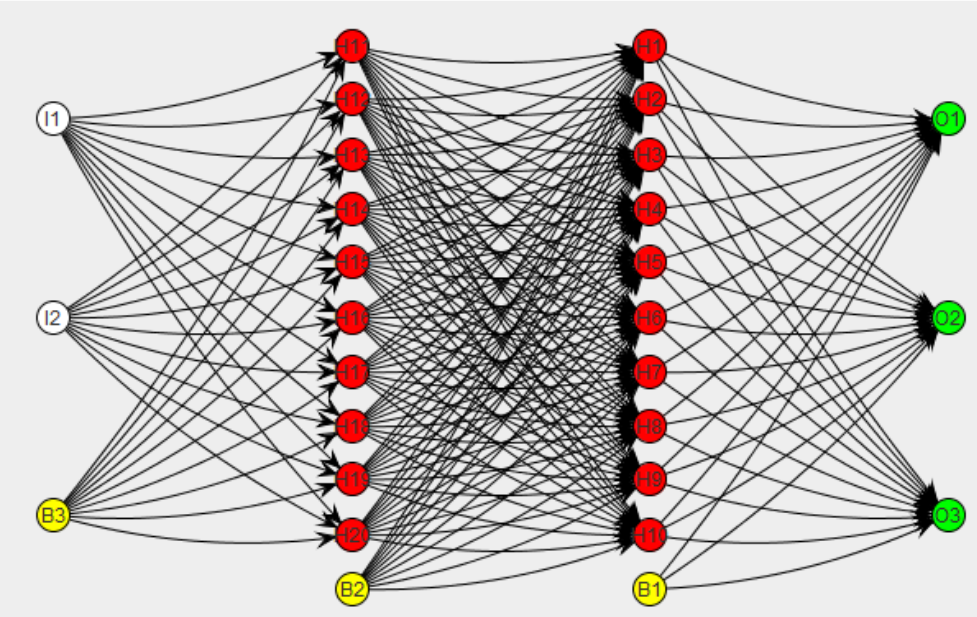
galvanize





Previously: Multilayer Perceptrons



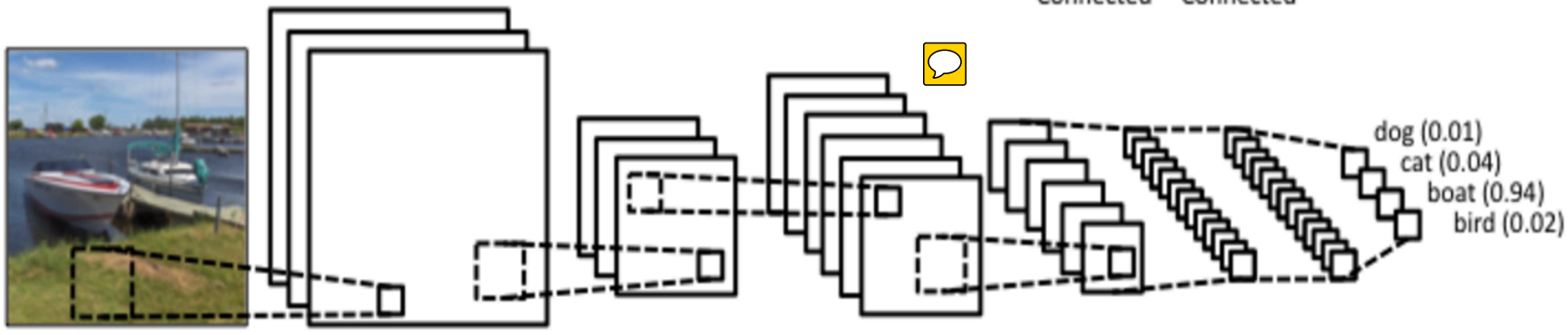


http://ufldl.stanford.edu/wiki/index.php/Neural_Networks

http://www.codeproject.com/KB/recipes/477689/jmsl-7.png

Today: Convolutional Neural Networks

galvanize



Recent Breakthroughs

galvanize

Acoustic Modeling using Deep Belief Networks
Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010
Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition
George Dahl, Dong Yu, Li Deng, Alex Acero, 2012

Imagenet classification with deep convolutional neural networks
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012

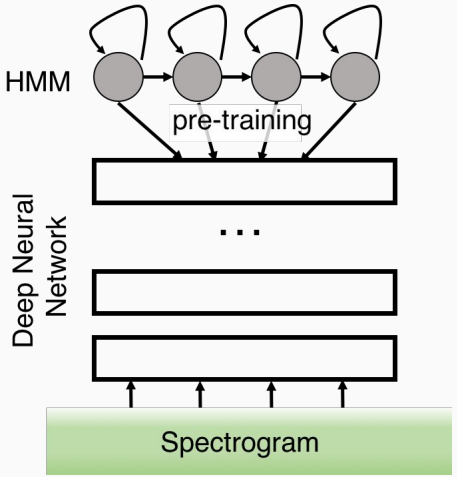
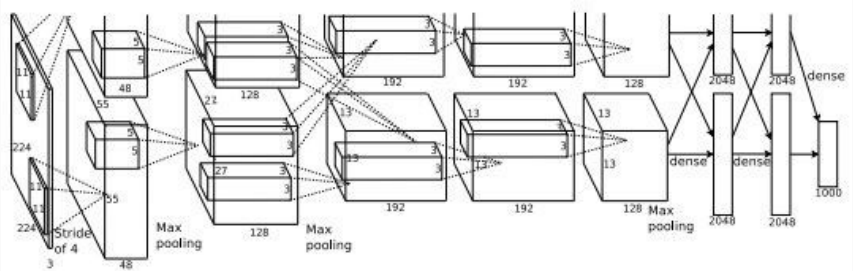
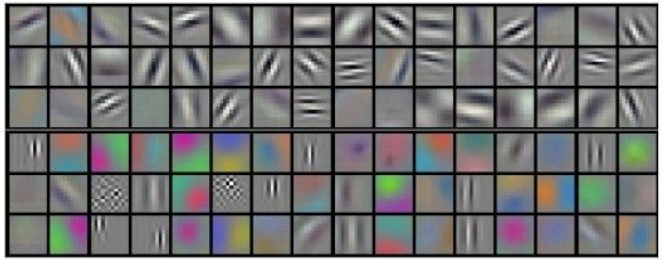


Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017





Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

galvanize

ImageNet Classification with Deep Convolutional Neural Networks *[Krizhevsky, Sutskever, Hinton, 2012]*

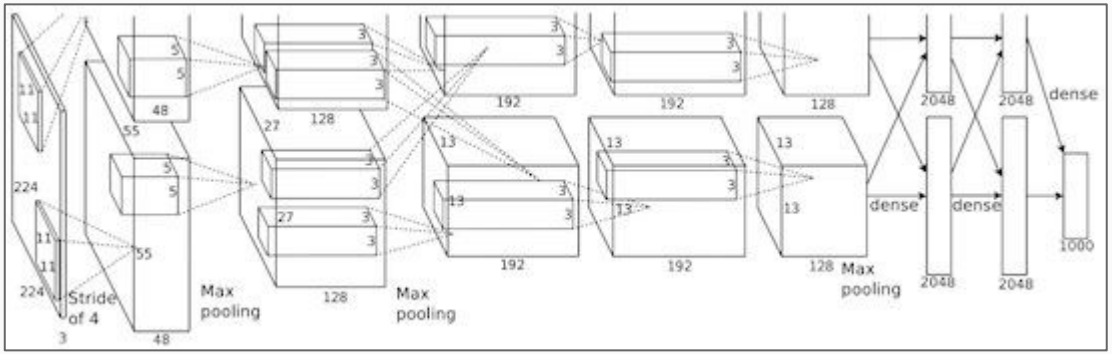


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

"AlexNet"

Fast-forward to today: CNNs are everywhere

galvanize

Classification

Retrieval

Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Fast-forward to today: CNNs are everywhere

galvanize

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]

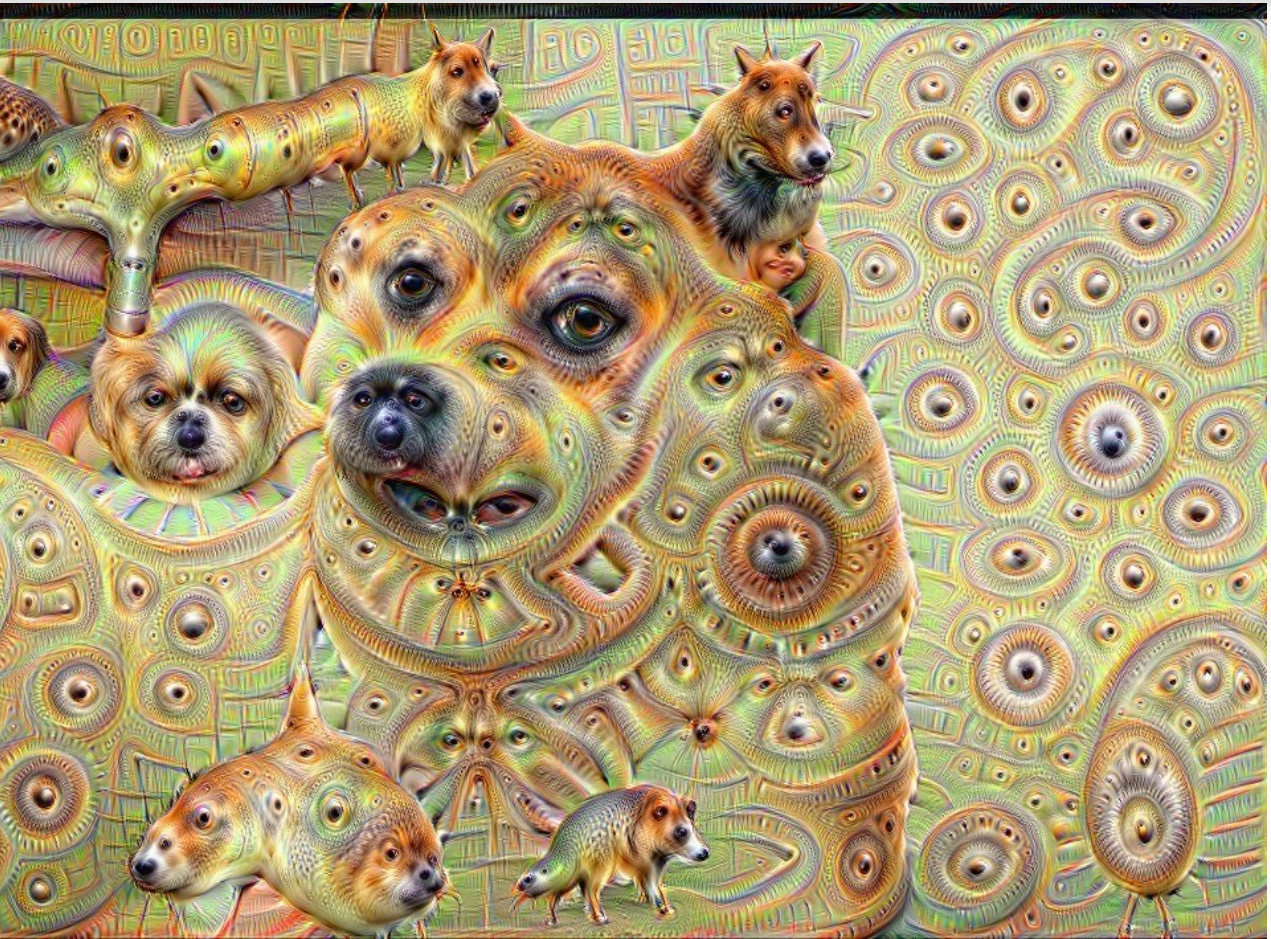
Applications

galvanize

- Self-driving cars
- Facial recognition software
- Post office (automated mail sorting)
- Academic image classification
- Image captioning (useful for AI…)
- Many more!

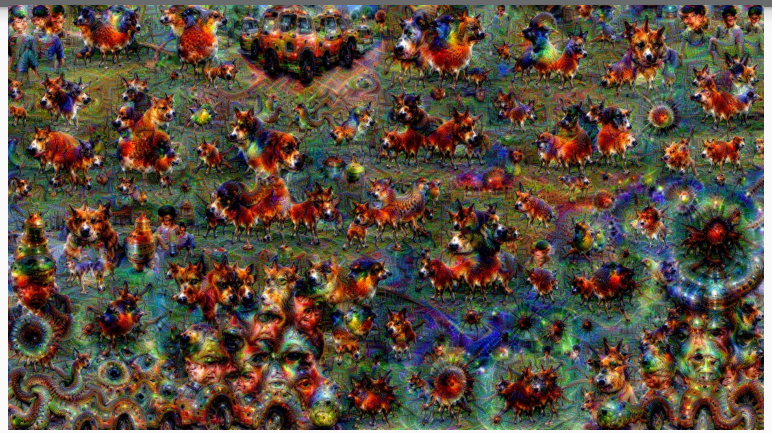
Google Deep Dream

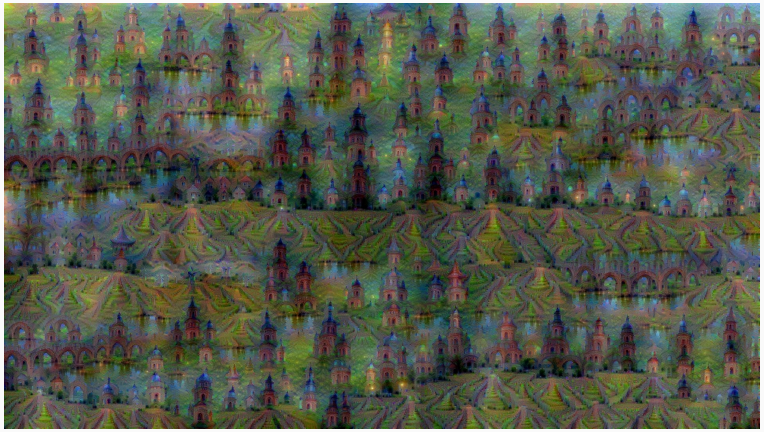
galvanize



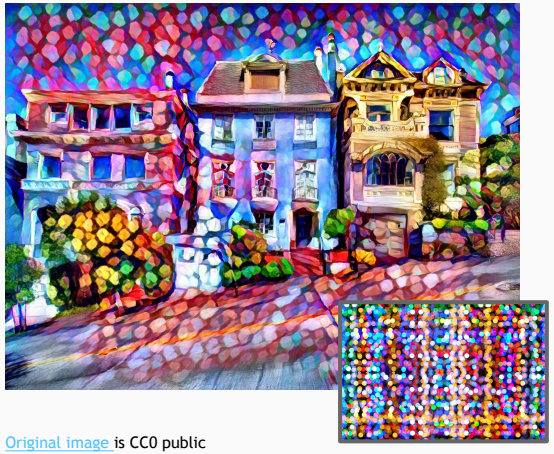
Art

galvanize





Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a blog post by Google Research.

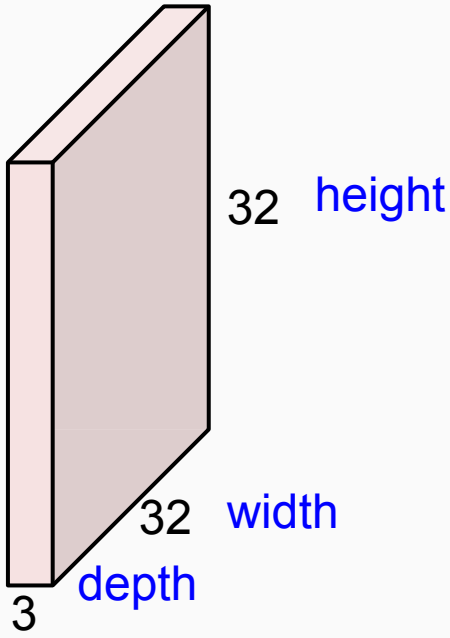


Original image is CC0 public domain Starry Night and Tree Roots by Van Gogh are in the public domain Bokeh image is in the public domain Stylized images copyright Justin Johnson, 2017; reproduced with permission

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016 Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

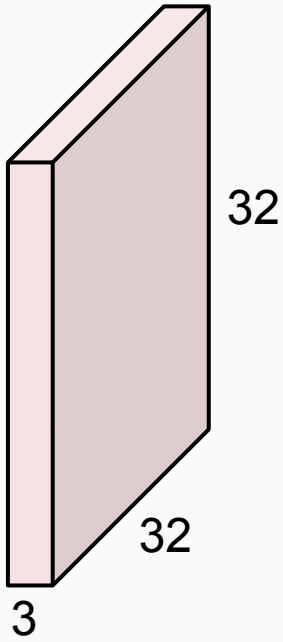
CNNs: Convolutional Layers: How they work

32x32x3 image -> preserve spatial structure

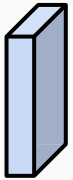


Convolution Layer

32x32x3 image



5x5x3 filter



Convolve the filter (aka “kernel”) with the image, i.e., “slide over the image spatially, computing dot products”

What is an image kernel?

galvanize

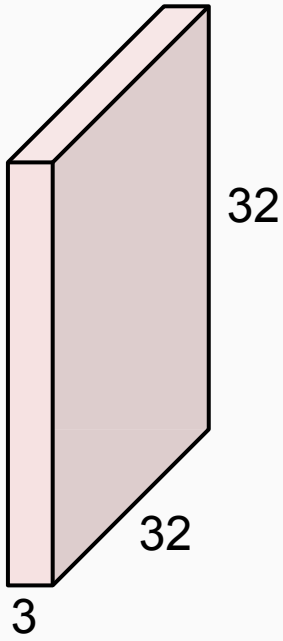
“An image kernel (filter, convolution) is a small matrix used to apply effects like you might find in Photoshop or Gimp, such as blurring, sharpening, etc. They're used in machine learning for 'feature extraction', a technique for determining the most important portions of an image. In this context the process is referred to more generally as *convolution.*”

Image kernels explained visually

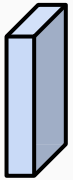
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

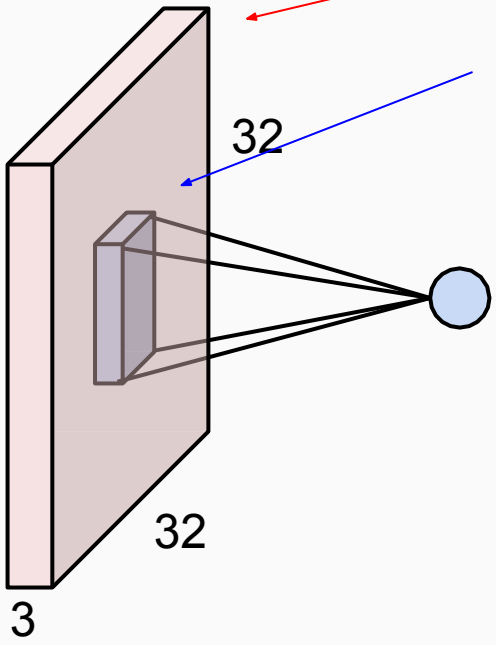


5x5x3 filter



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

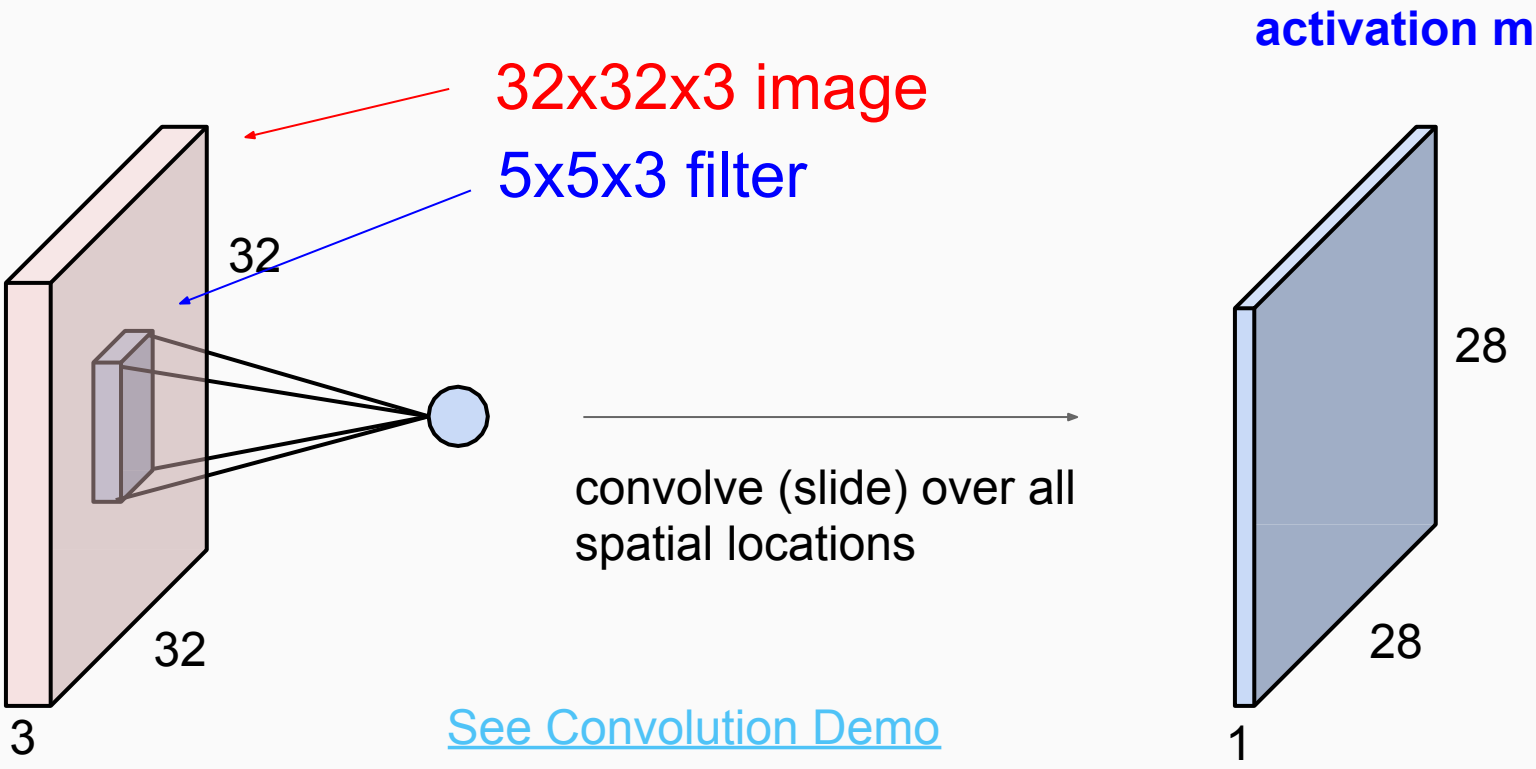
Convolution Layer



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)

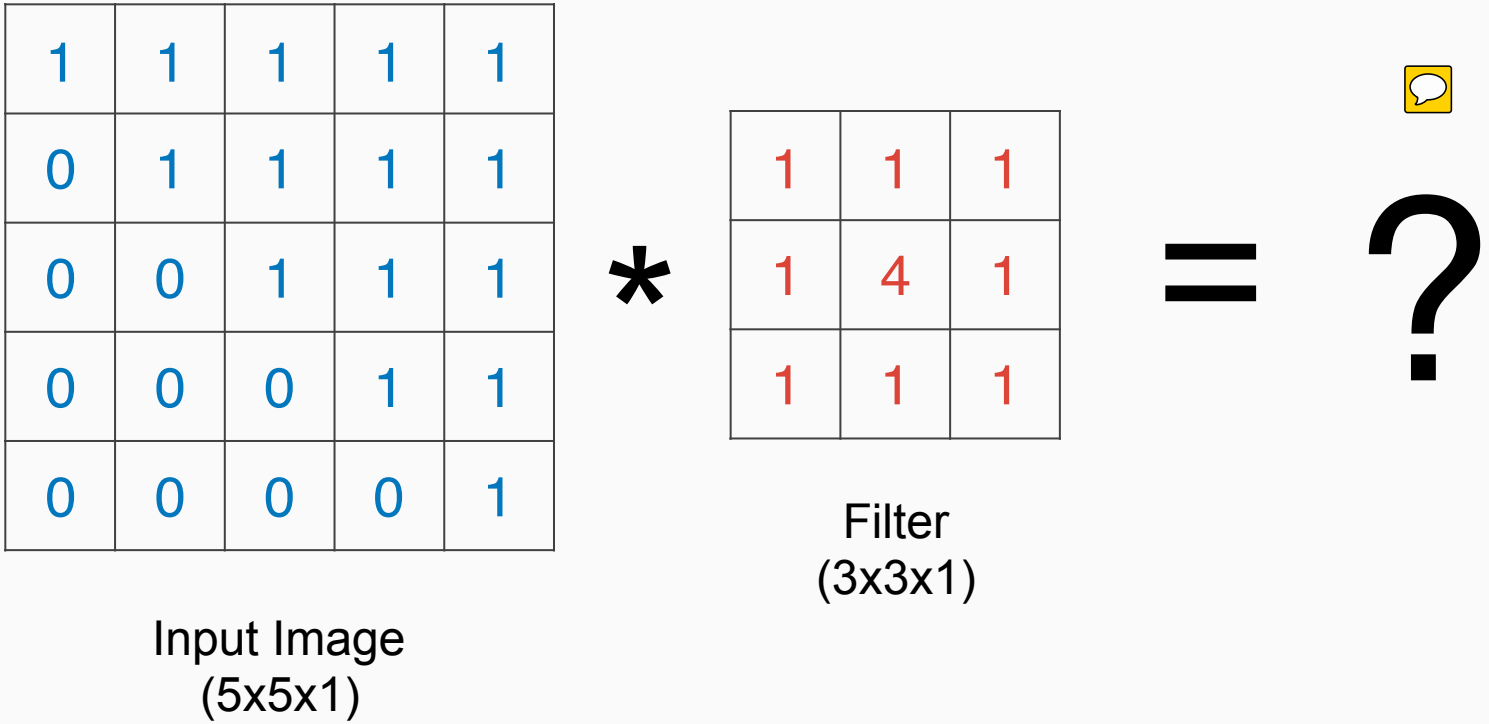
$$w^T x + b$$

Convolution Layer

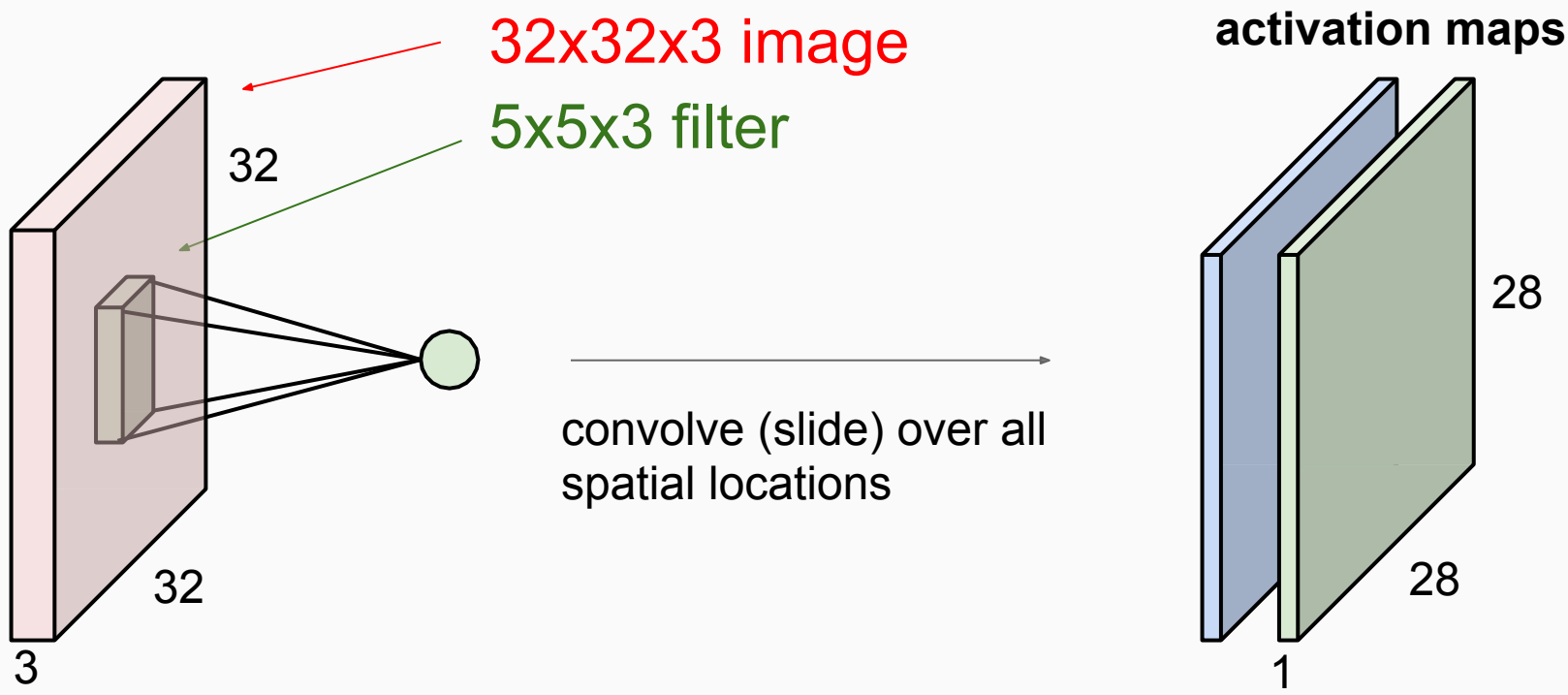


See Convolution Demo

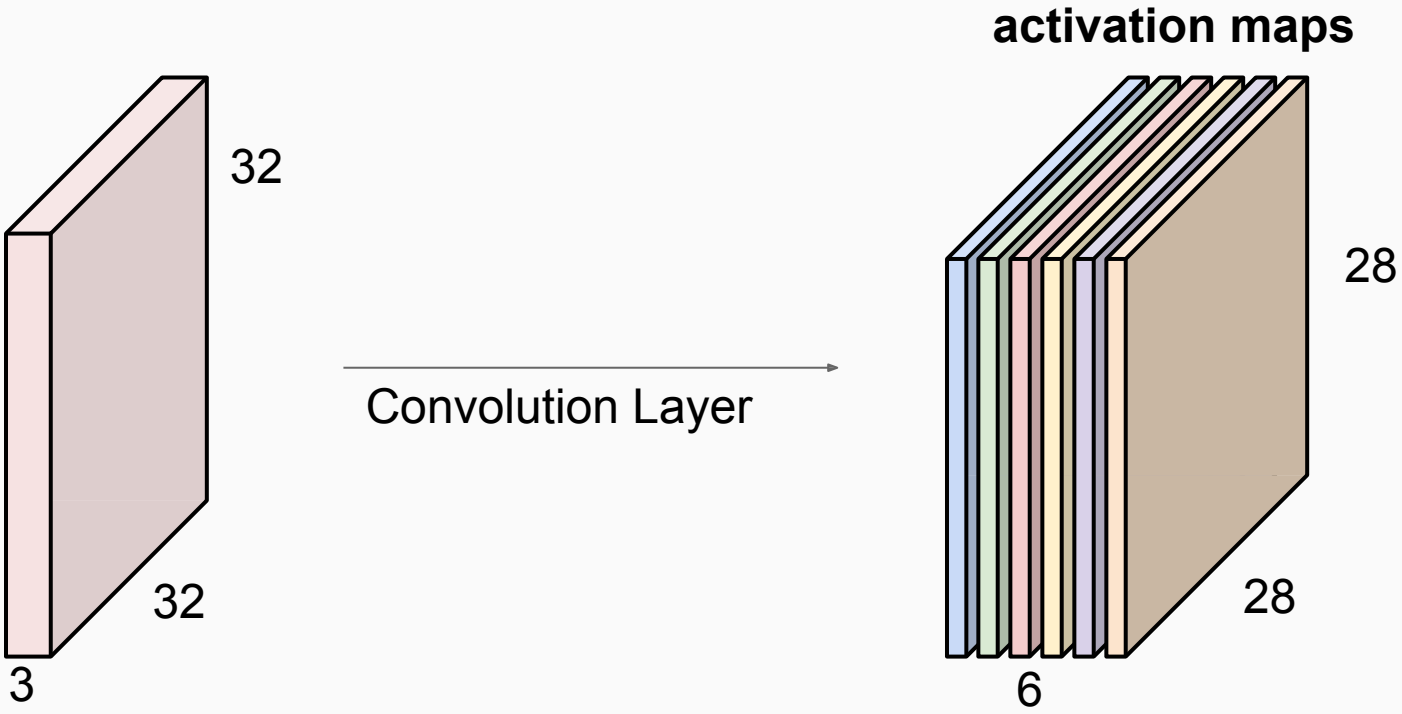
Breakout: Convolution Exercise



Consider a second, green filter



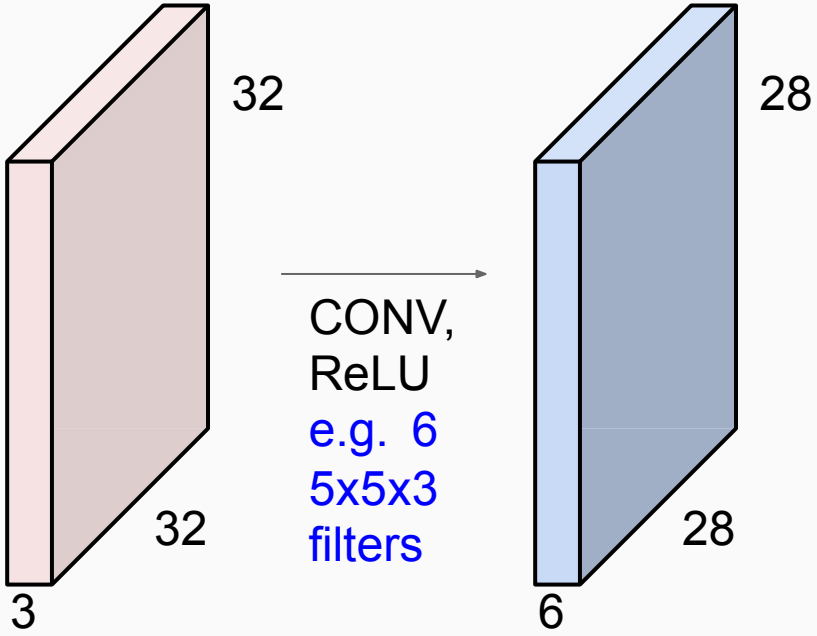
Multiple filters yield multiple activation maps



For example, if we had 6 5x5 filters, we'll get 6 separate activation maps

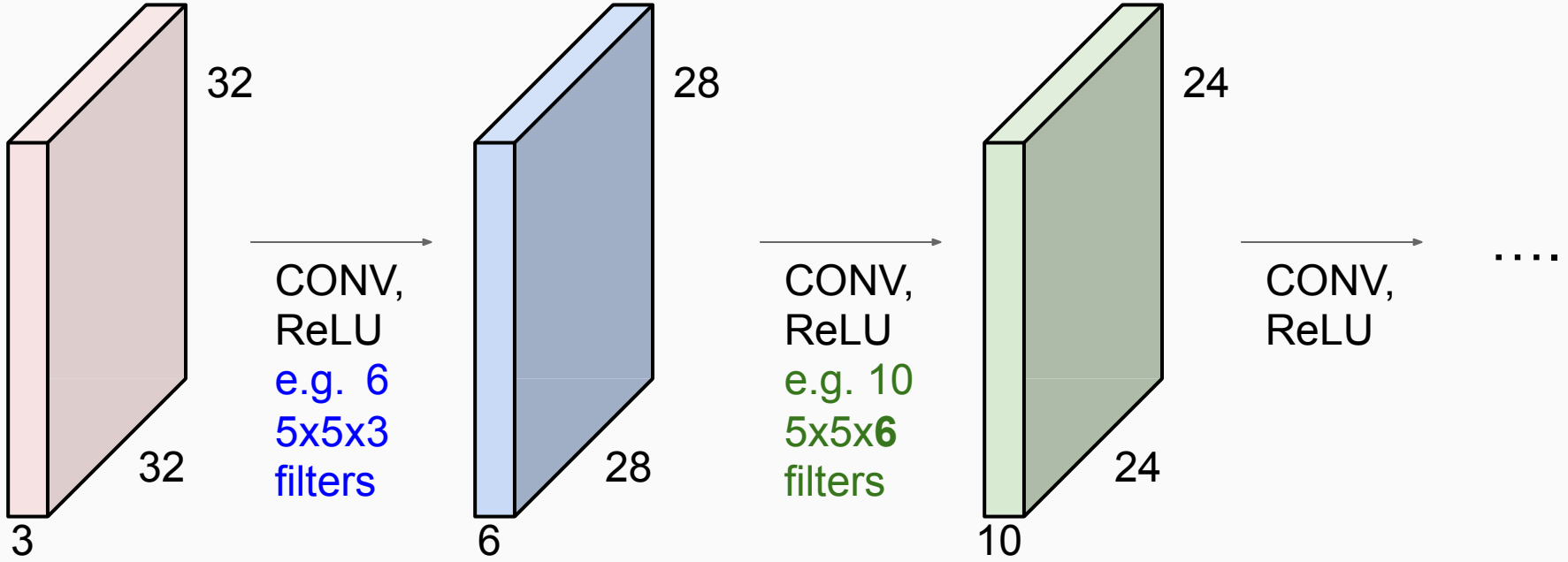
We stack these up to get a "new image" of size 28x28x6!

CNN Architecture



A CNN (or “ConvNet”) is a sequence of Convolution Layers, interspersed with activation functions.

CNN Architecture



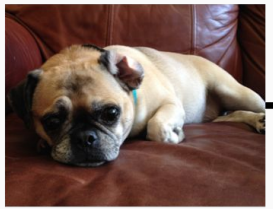
A CNN (or “ConvNet”) is a sequence of Convolution Layers, interspersed with activation functions.

Emergent Property: Hierarchical Feature Detection

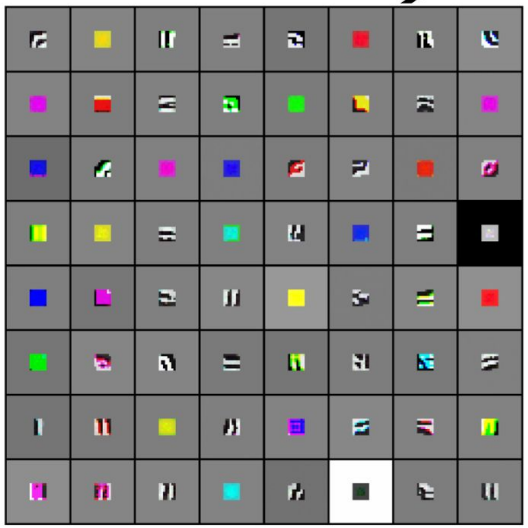
galvanize

[Zeiler and Fergus 2013]

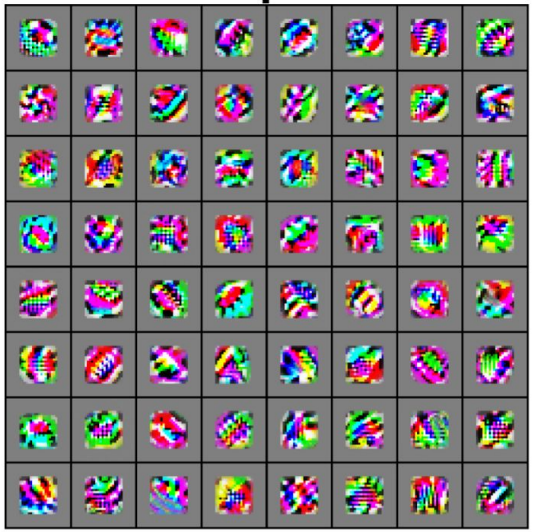
Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



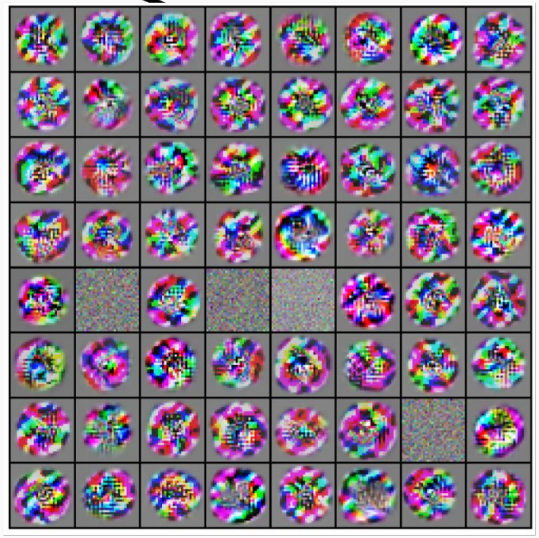
Low-level features → Mid-level features → High-level features → Linearly separable classifier



VGG-16 Conv1_1



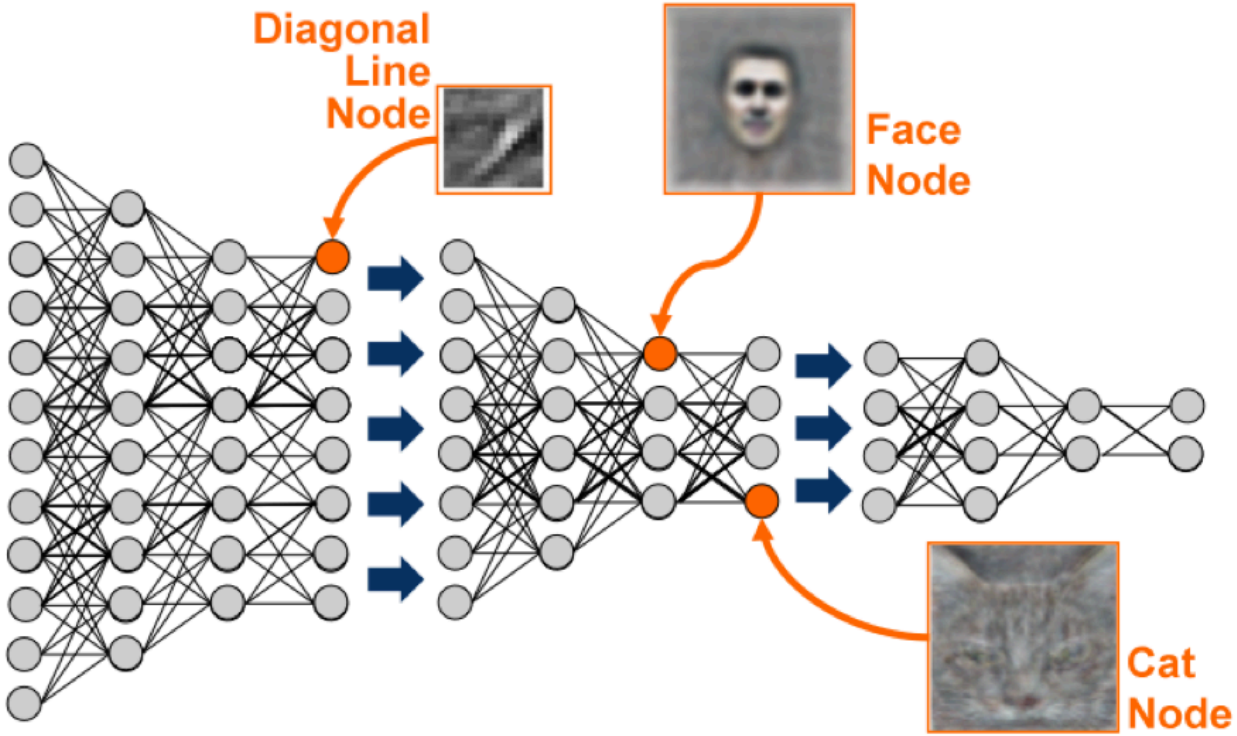
VGG-16 Conv3_2



VGG-16 Conv5_3

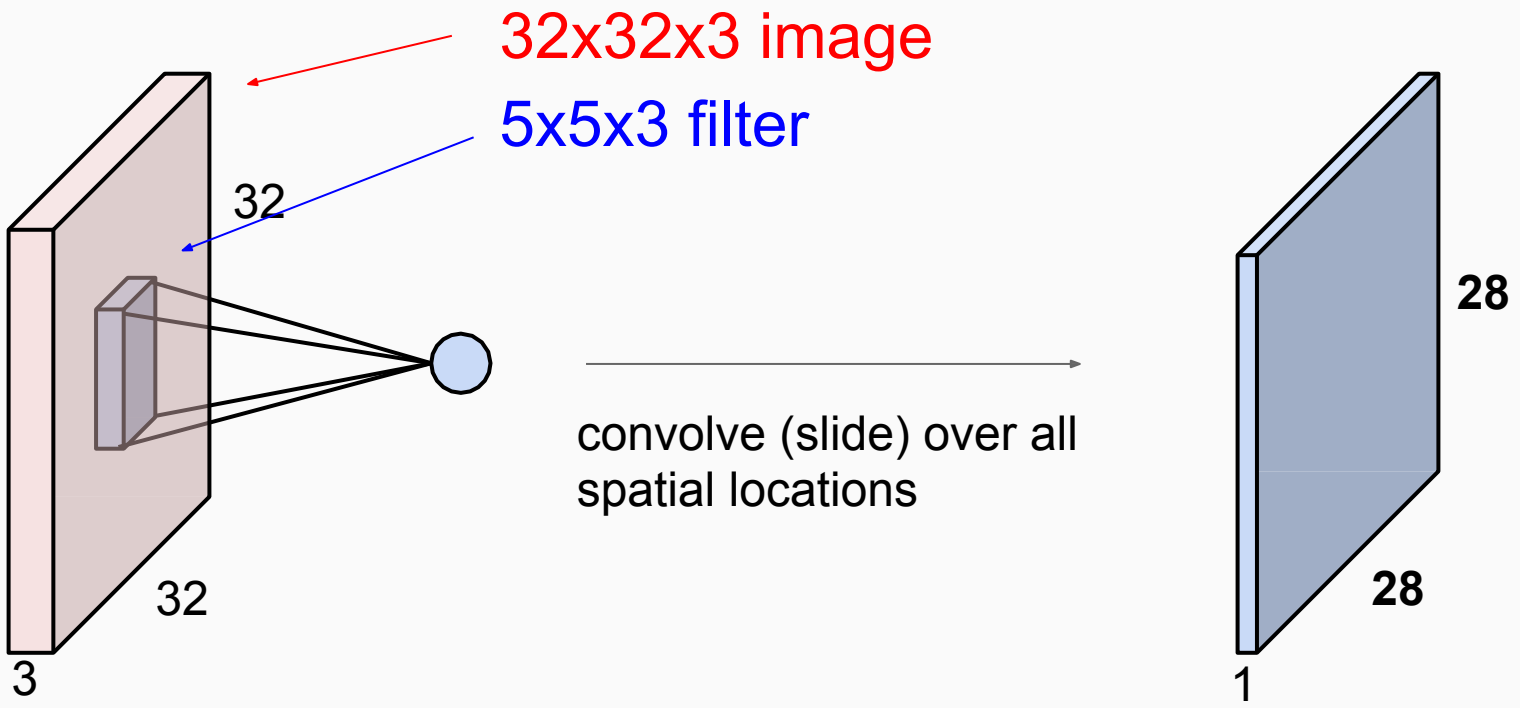
Deep Learning

Lots of Data + Neural Nets + Training = Hierarchical & Associational Feature Representation

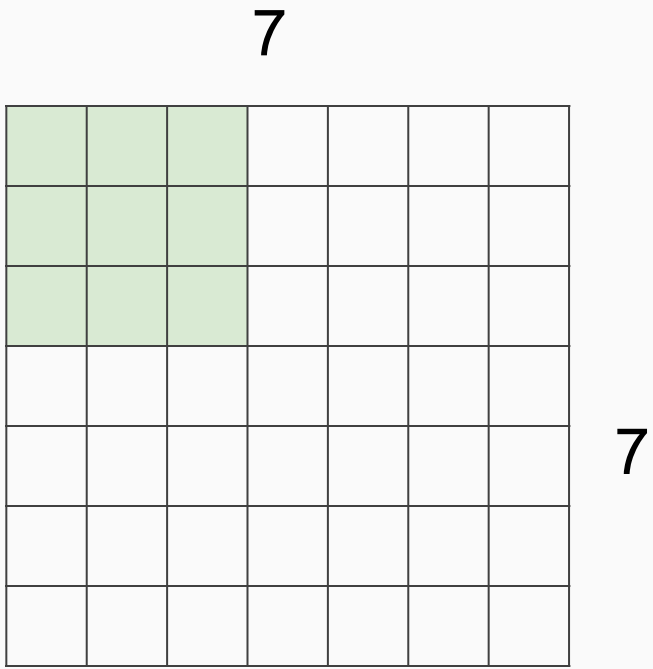


Convolutional Layers: Filters

A closer look at spatial dimensions:



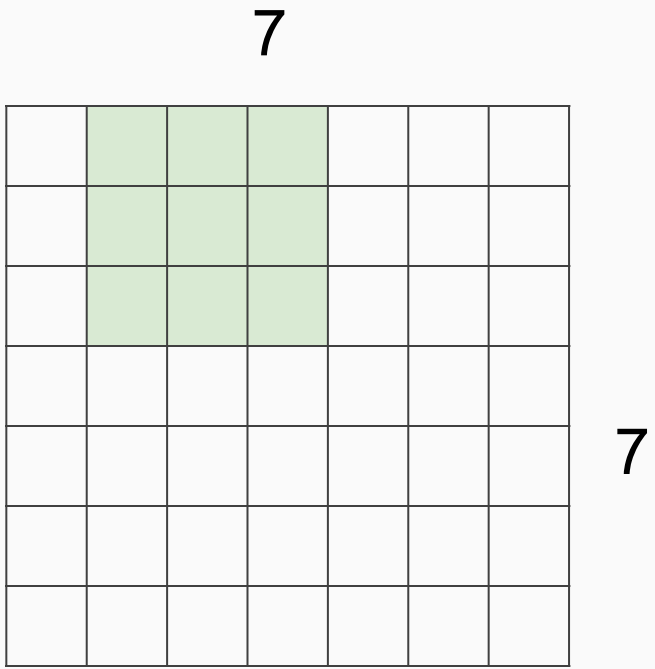
A closer look at spatial dimensions:



7x7 input (spatially)

assume 3x3 filter

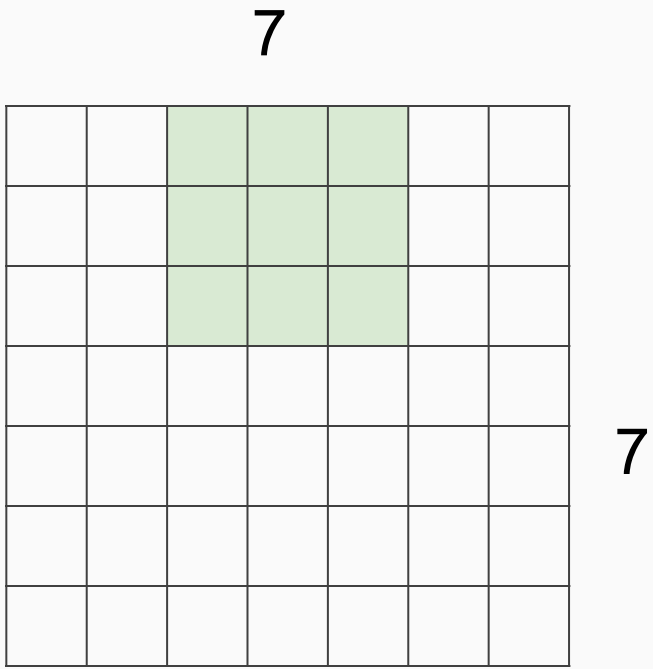
A closer look at spatial dimensions:



7x7 input (spatially)

assume 3x3 filter

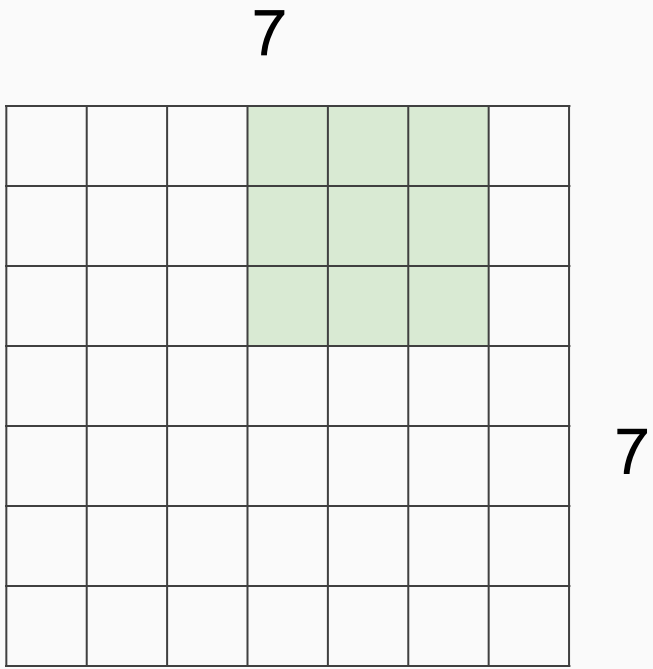
A closer look at spatial dimensions:



7x7 input (spatially)

assume 3x3 filter

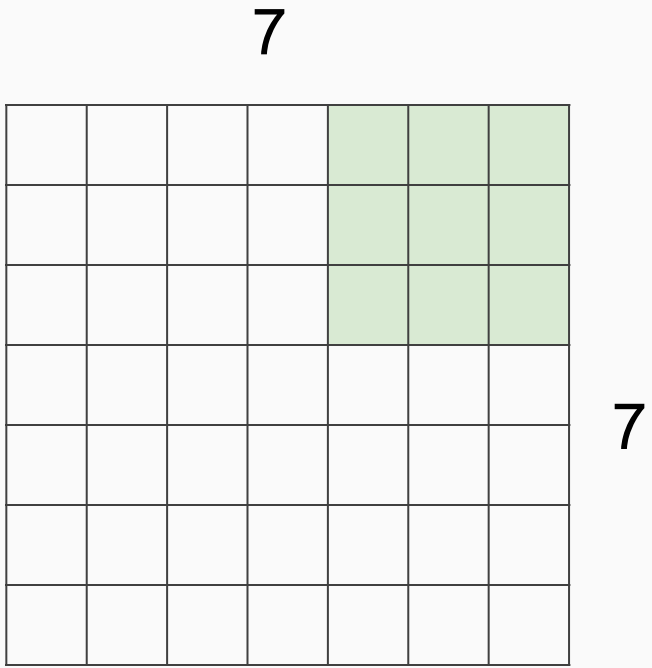
A closer look at spatial dimensions:



7x7 input (spatially)

assume 3x3 filter

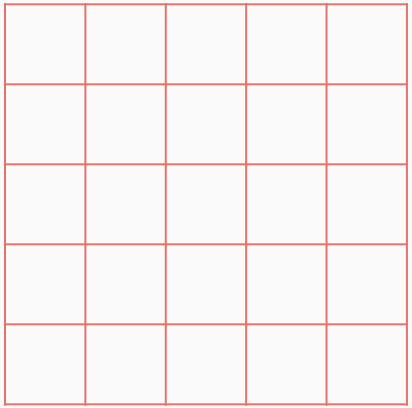
A closer look at spatial dimensions:



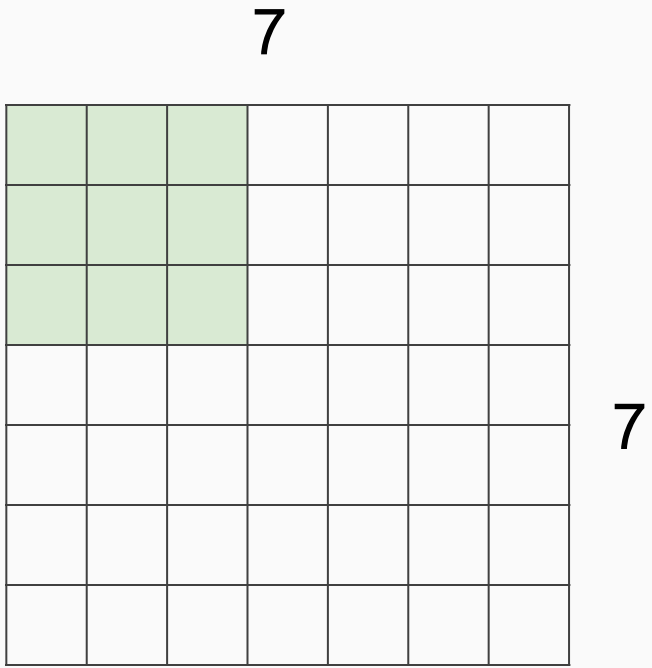
7x7 input (spatially)

assume 3x3 filter

=> 5x5 output



A closer look at spatial dimensions:

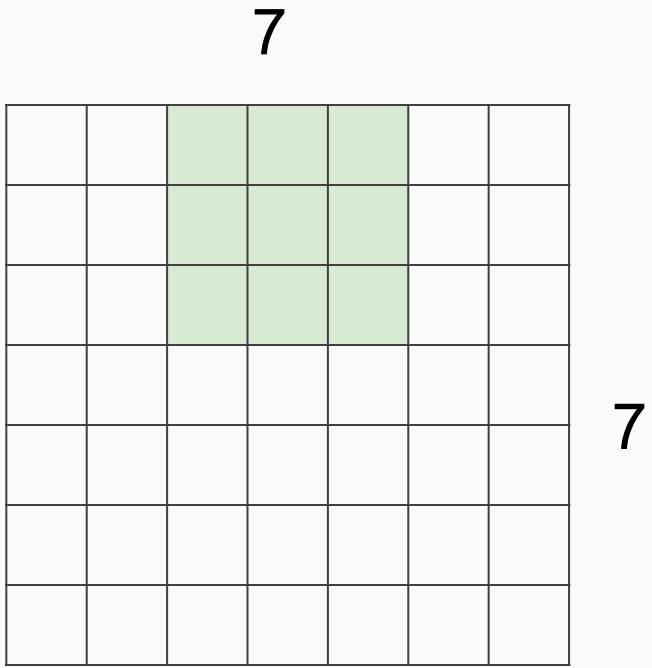


7x7 input (spatially)

assume 3x3 filter

applied **with stride 2**

A closer look at spatial dimensions:

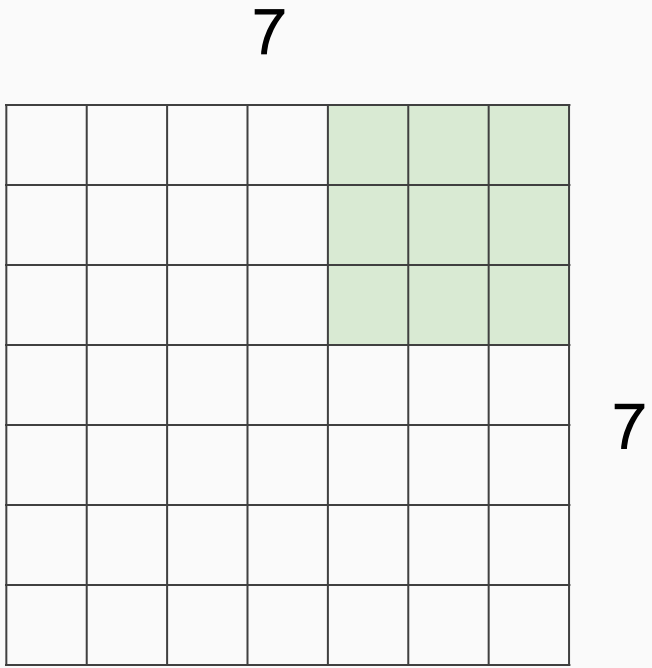


7x7 input (spatially)

assume 3x3 filter

applied **with stride 2**

A closer look at spatial dimensions:

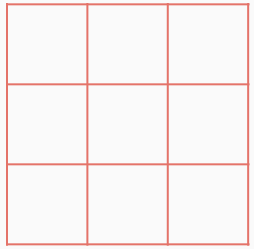


7x7 input (spatially)

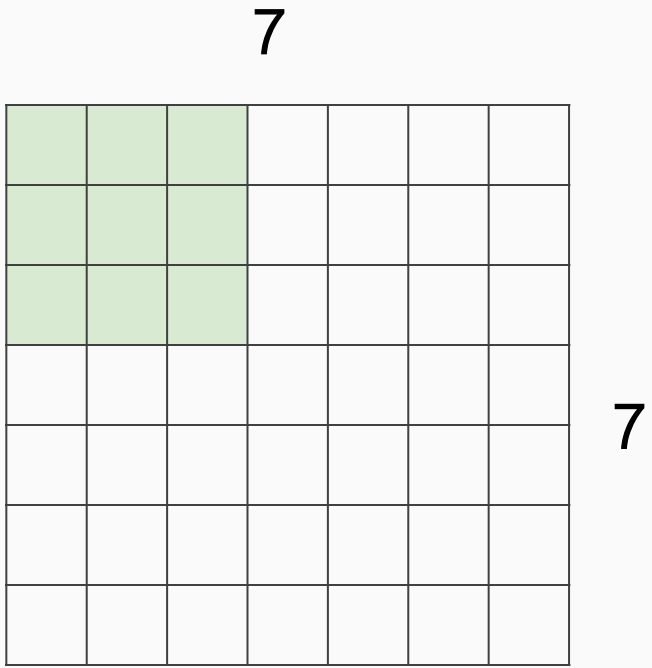
assume 3x3 filter

applied **with stride 2**

=> 3x3 output!



A closer look at spatial dimensions:



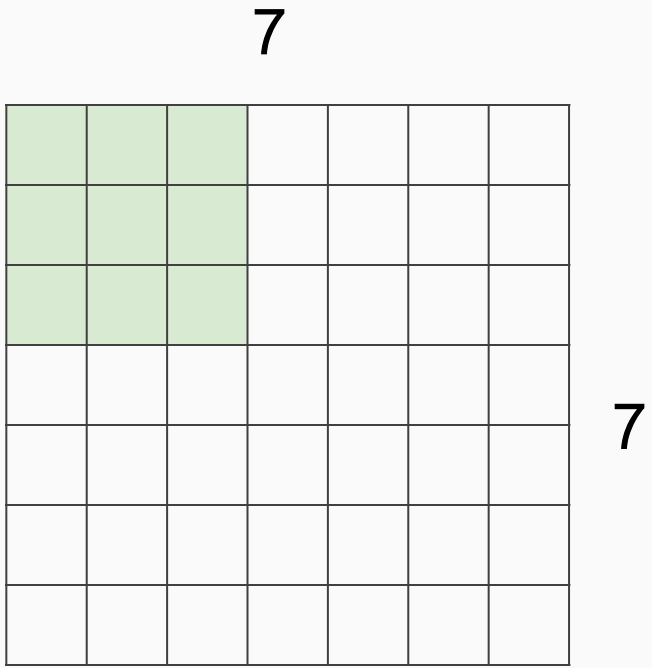
7x7 input (spatially)

What is the output of

3x3 filter applied

with stride 3?

A closer look at spatial dimensions:



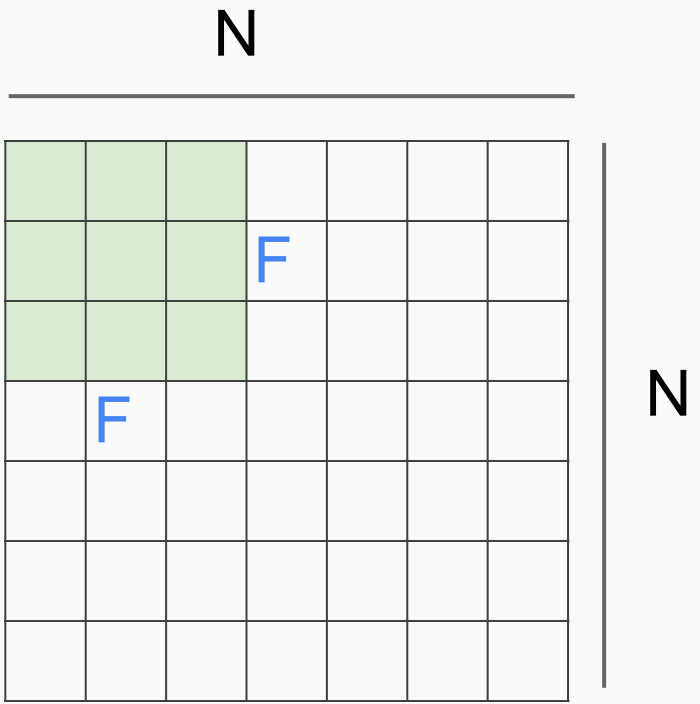
7x7 input (spatially)

What is the output of 3x3 filter applied **with stride 3?**

doesn't fit!

cannot apply 3x3 filter on 7x7 input with stride 3.

Calculating Output Size



Output size:

(N - F) / stride + 1

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

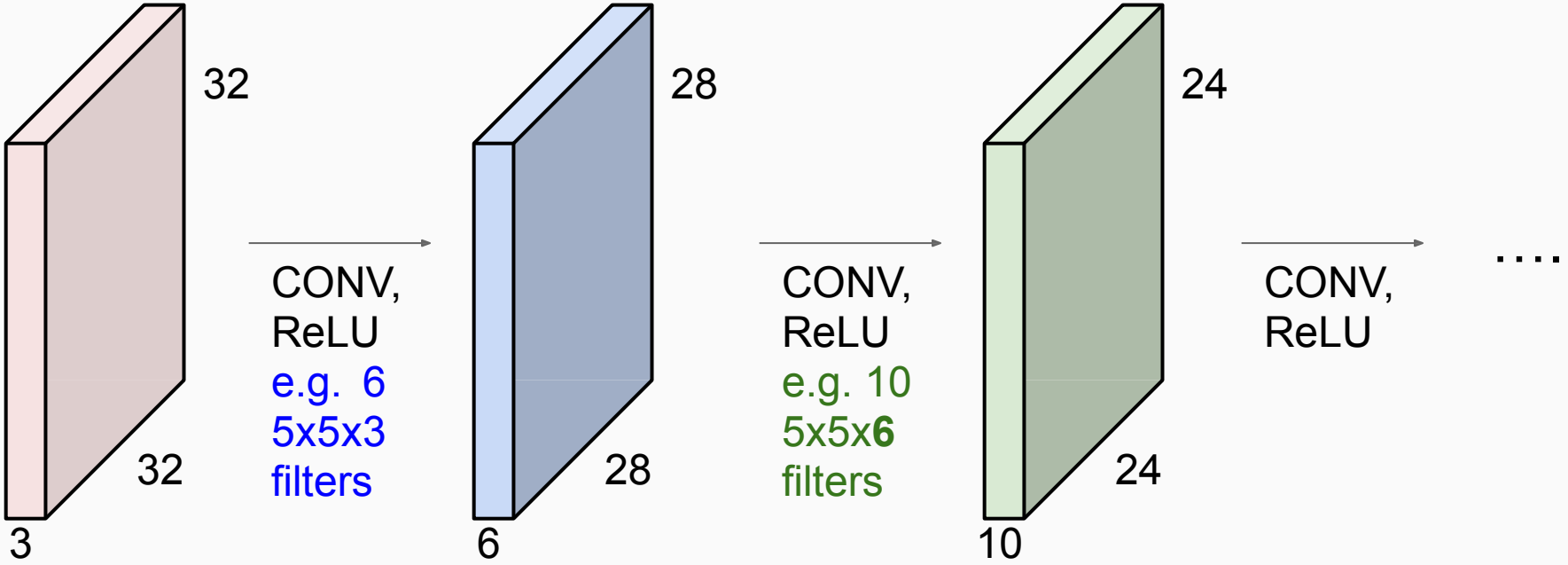
stride 1 => (7 - 3)/1 + 1 = 5

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

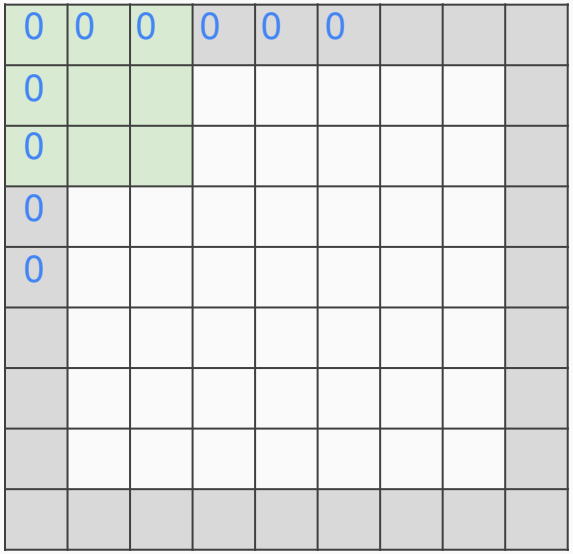
stride 3 => (7 - 3)/3 + 1 = 2.33 😕

Shrinking Output Volume

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



In practice: Common to zero pad the border

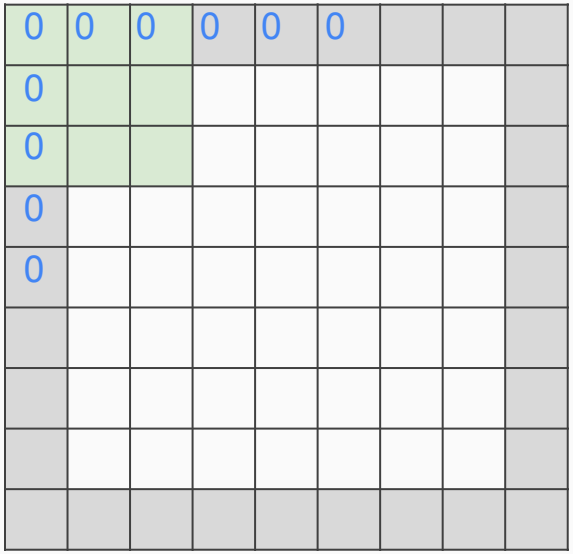


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)

(N - F) / stride + 1

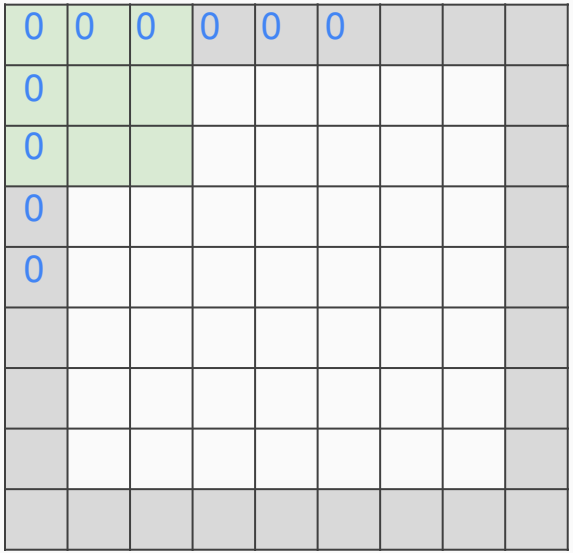
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)

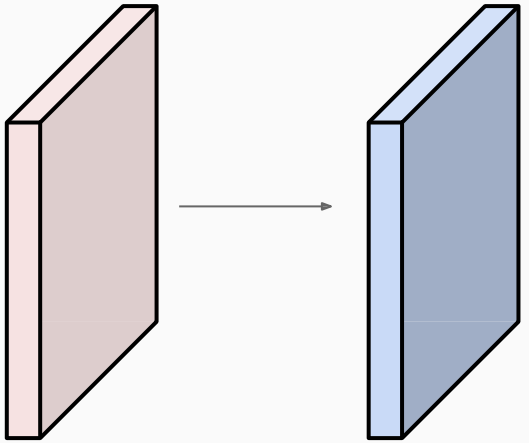
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Question #1

galvanize

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

Output volume size: ?

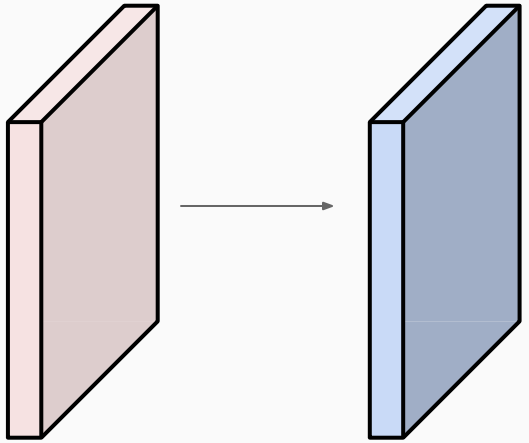


Question #1

galvanize

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

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

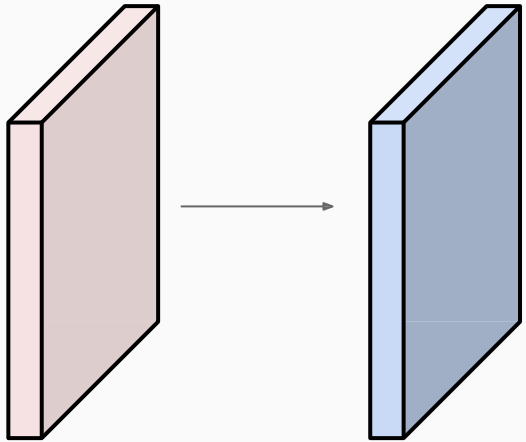


Question #2

galvanize

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

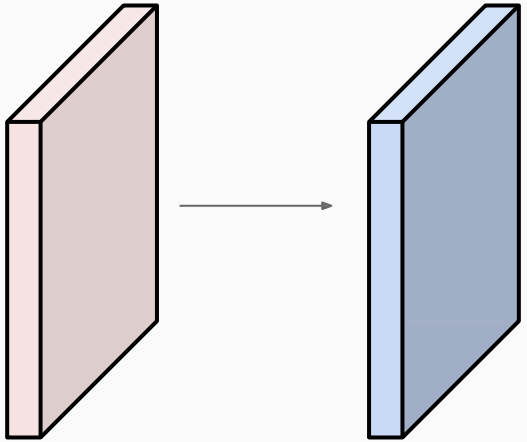
Number of parameters in this layer?



Question #2

galvanize

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

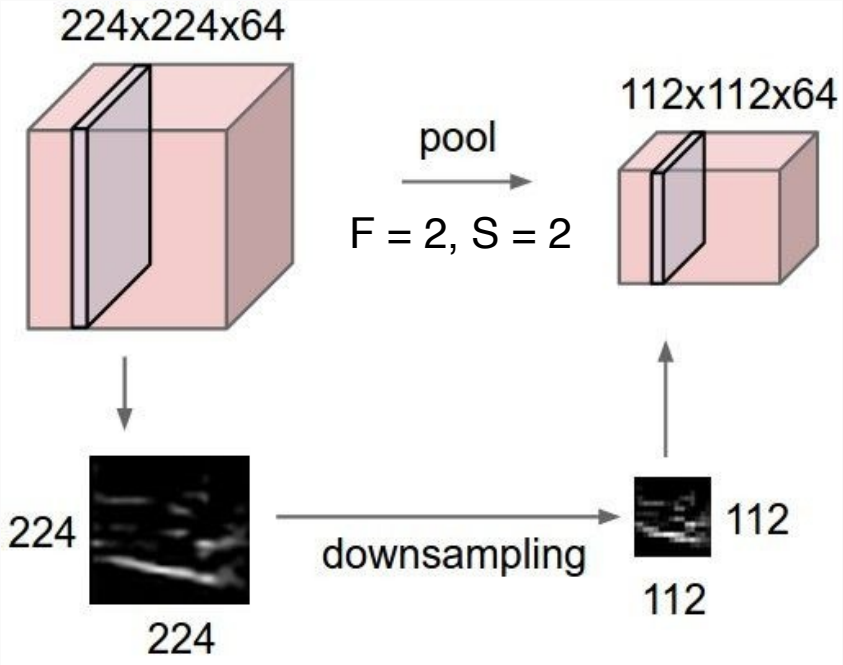


Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

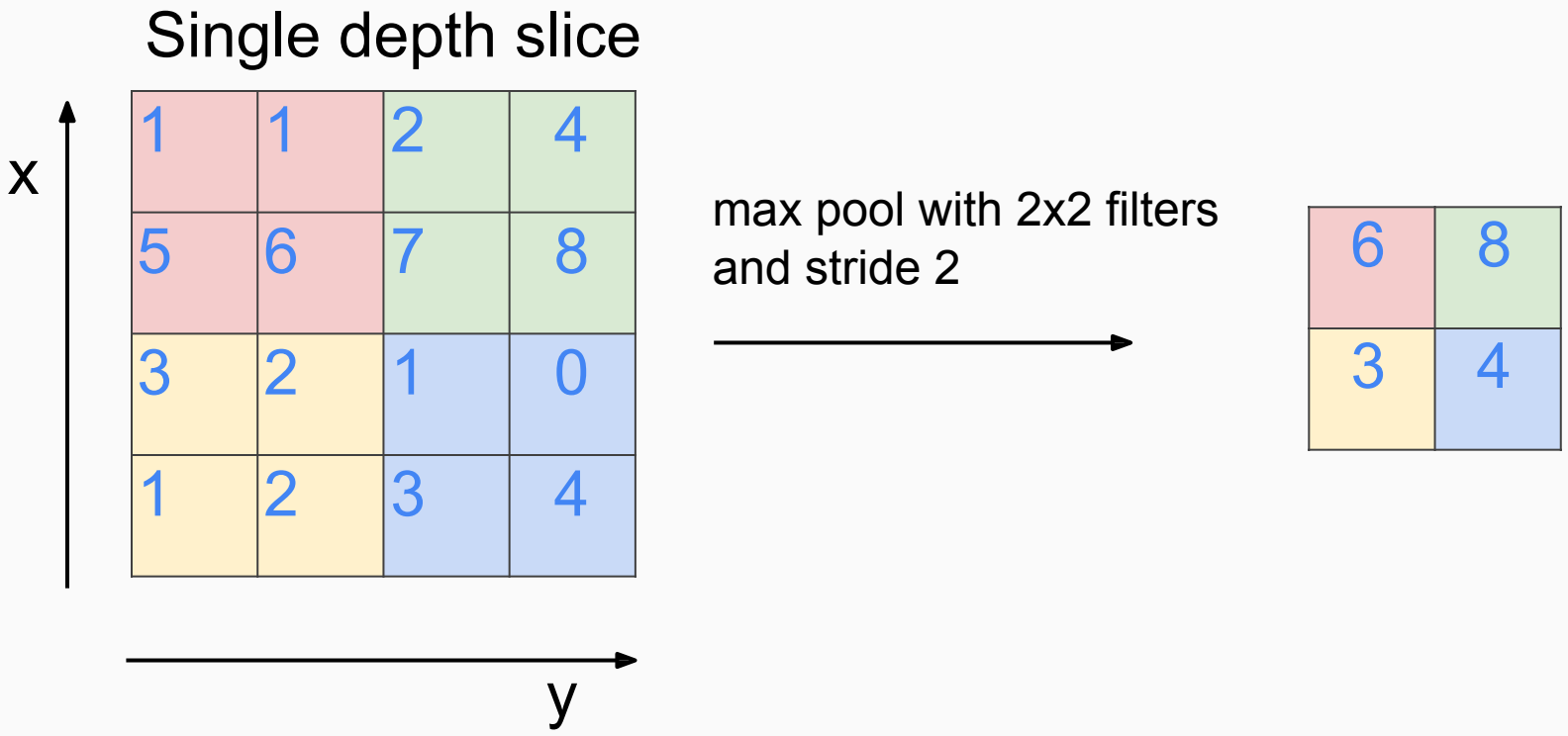
Pooling Layers

Pooling Layers: Spatial Downsampling

- Reduces model complexity, forces generalization
- Common settings: F = 2, S = 2; F = 3, S = 2
- Operates over each activation map independently:



Max Pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

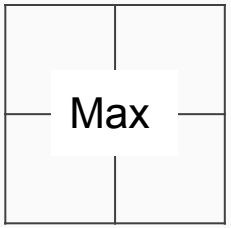
Breakout: Max Pooling Exercise

galvanize

2	7	5	5	3	3
6	3	0	3	4	0
6	3	9	0	2	3
0	1	6	2	6	6
6	3	0	5	1	0
7	3	6	4	0	5

Input Image
(6x6x1)

*



Max Pool
(2x2x1)
stride = 2

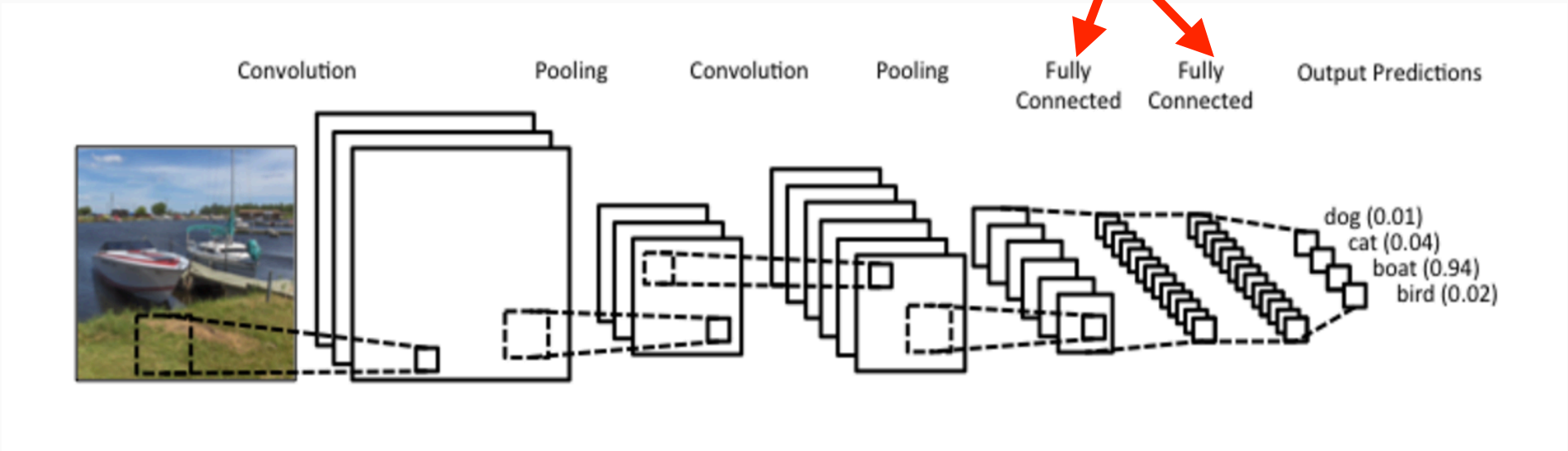
= ?

Fully connected layers

Fully Connected Layer (FC layer)

galvanize

- After “enough” spatial features are learned from convolutional layers, inputs are flattened out.
- Common CNN architecture includes several hidden, fully connected layers before the output layer
- Dimension of the fully connected layer will be x * y * n_filters (after the last convolutional or pooling layer)

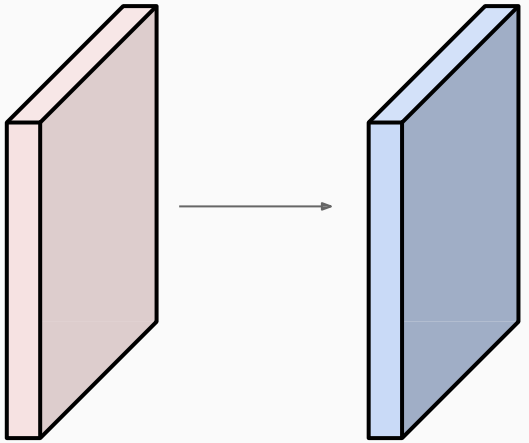


Question #1

galvanize

Input volume: **8x8x64**
Flatten for input to dense layer

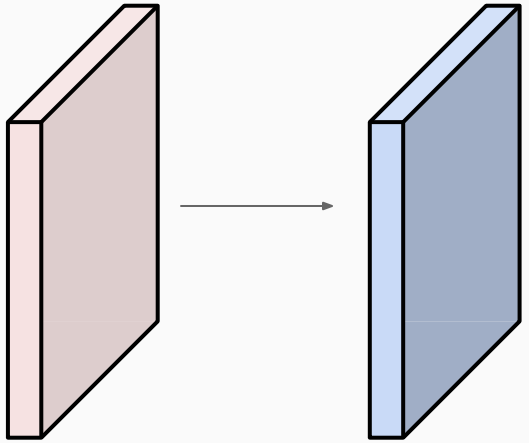
Output size: ?



Question #1

galvanize

Input volume: **8x8x64**
Flatten for input to dense layer

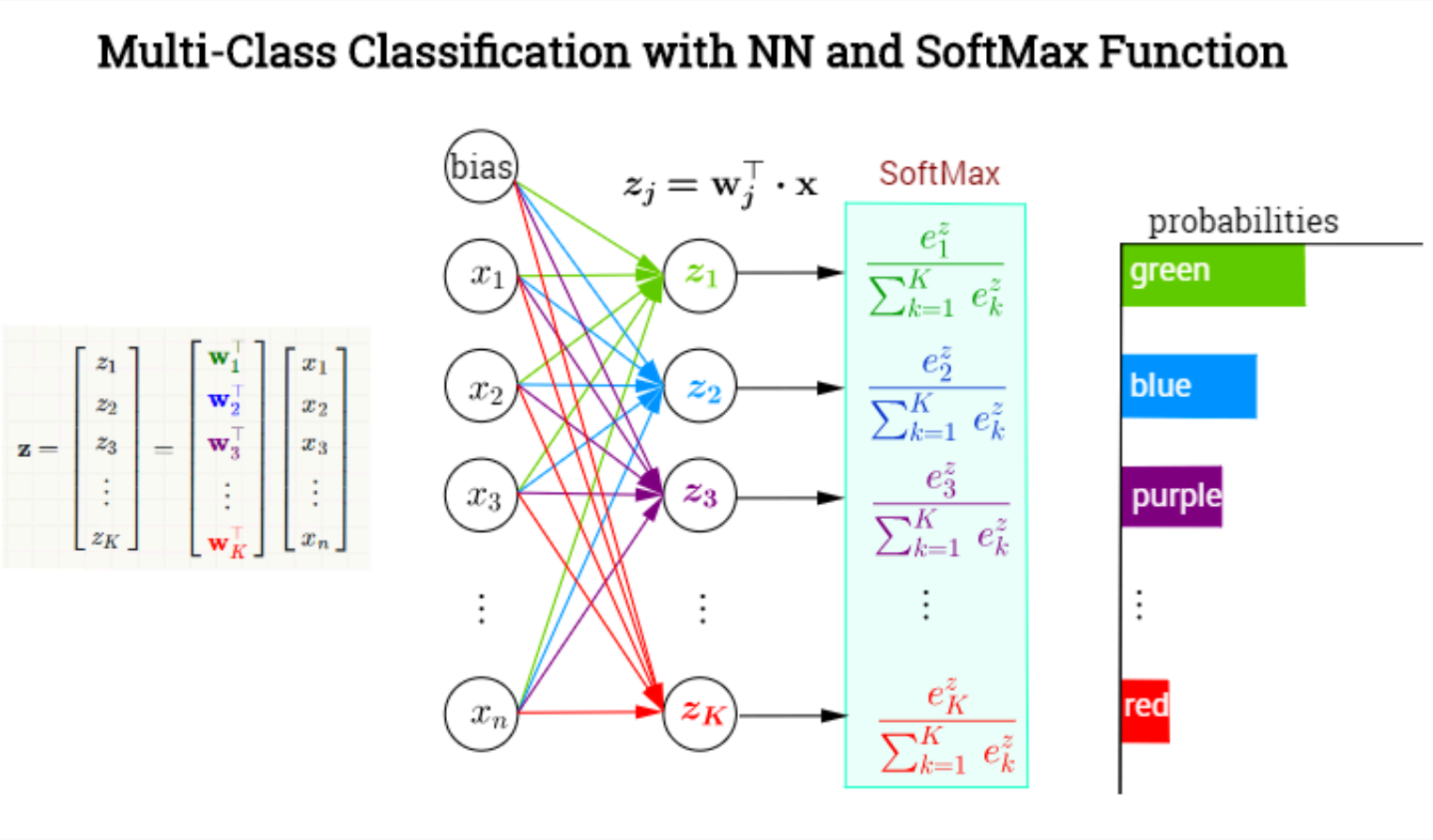


Output volume size:
8*8*64 = **4096** (it's that simple!)

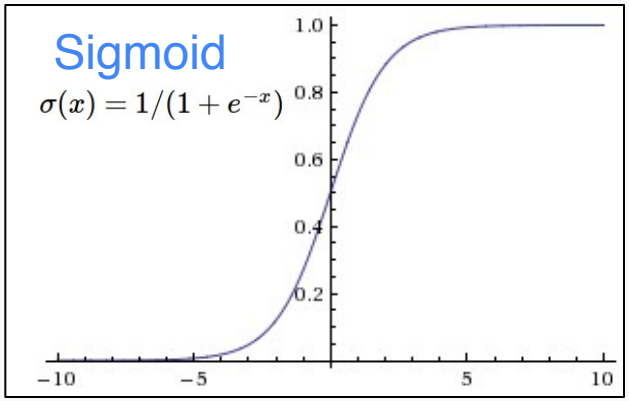
Output Activations

Output layer activations - Softmax for Classification

galvanize



Output layer activations - Multiple sigmoid for Labeling



https://github.com/Mikelew88/ChefNet

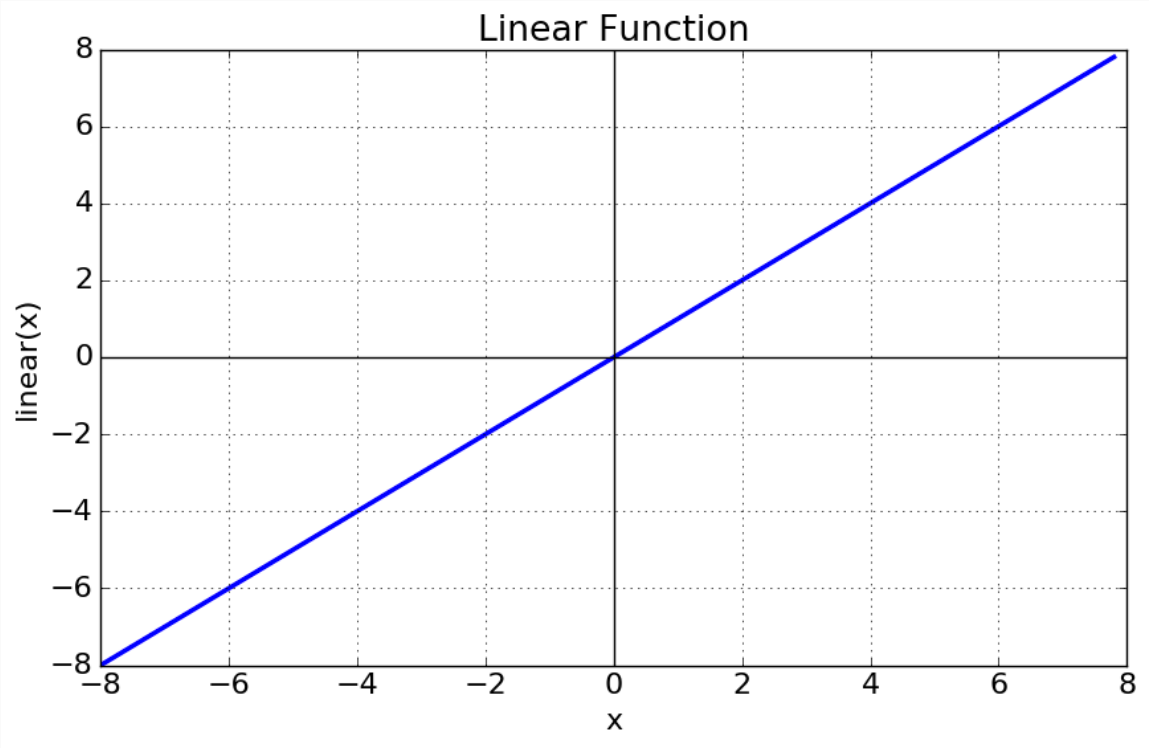
```
In [36]: predict_user_photo(model, vocab)
Please enter the image file name: tims_lunch.jpg
VGG net loaded
_____________________

ChefNet Predictions:

salt (62%), pepper (54%), garlic (40%), onion (40%), oil (37%), chicken (31%), butter (31%), egg (29%),
sauce (29%), sugar (28%)
```

Output layer activations - Regression

Generally, use linear activation - *pretty rare for image processing*



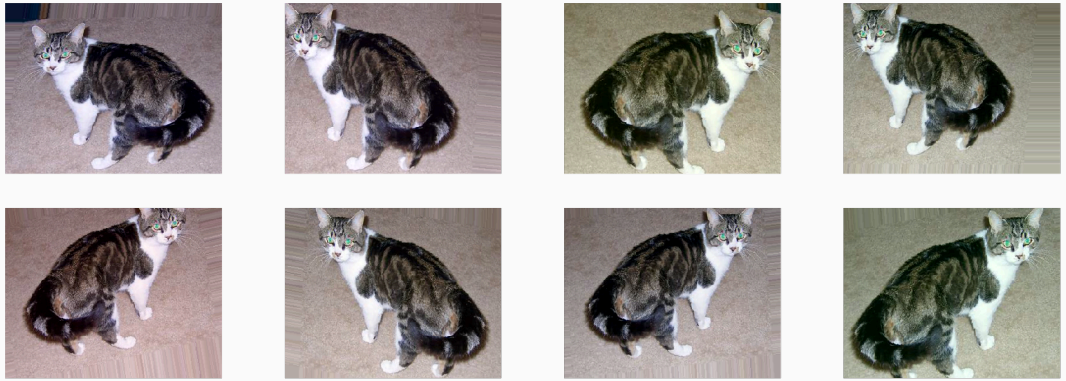
https://medium.com/towards-data-science/activation-functions-neural-networks-1cbd9f8d91d6

Data Augmentation

Data Augmentation

galvanize

- Neural networks work best with lots of data
- Dealing with small datasets and/or imbalanced classes is a difficult problem with tabular data
- Increasing number of samples can solve class imbalance issues and also make model more robust (less likely to overfit)
- Easy to upsample images : rotation, translations, flipping, skewing, cropping, adding noise, changing lighting, vignetting, perspective changes….



https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9

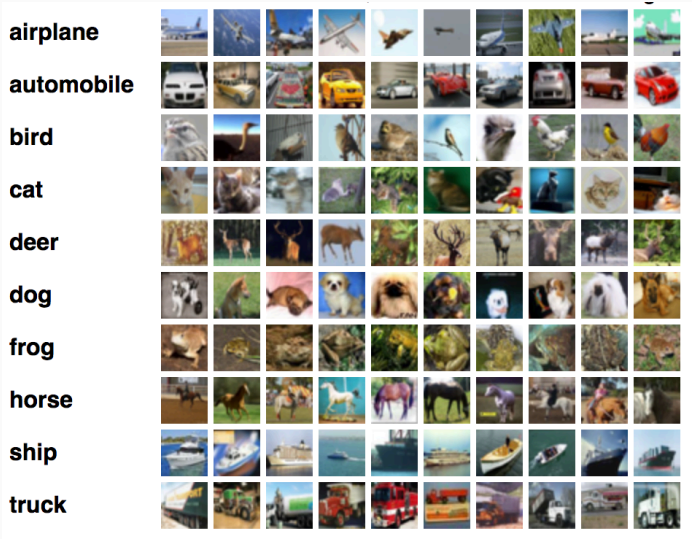
CIFAR-10 dataset

galvanize

Your exercise today : CIFAR 10

galvanize

- Labeled subset (60,000 images) of the 80 million tiny images dataset
- Like MNIST, has 10 classes
- UNLIKE MNIST, images are much more complex, training will be difficult and time consuming
- State of the art on this dataset is 90% (computer) and 94% (human) - with complex architecture, data augmentation, etc.



http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html