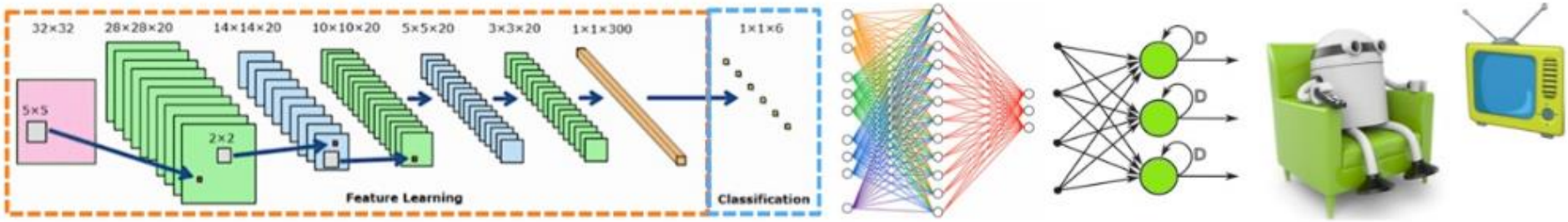
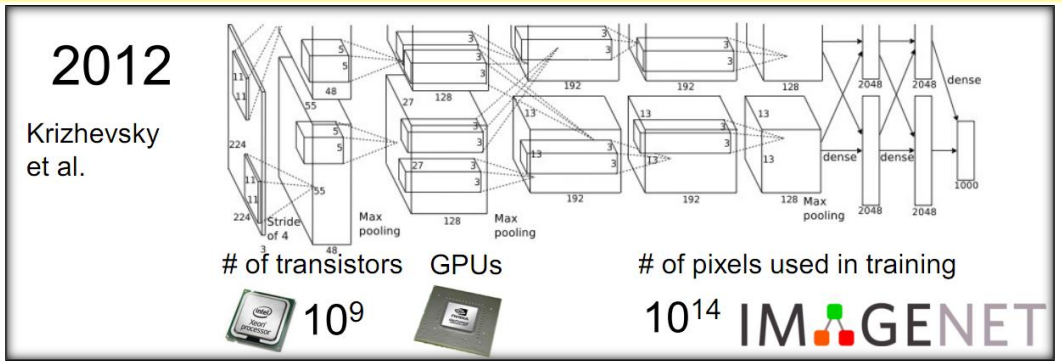


Lecture 7: Intro to CNNs

Dr. **Đinh Viết Sang**
Hanoi University of Science and Technology

Hanoi 2016

History of DL



Recurrent Neural Net

Convolutional Neural Net

Neural Net

Perceptron

SVM

2012

CNNs (data + GPU)

Deep (sparse/denoising) Autoencoder

Autoencoder Neural Net

Sparse Coding

GMM

ΣΠ

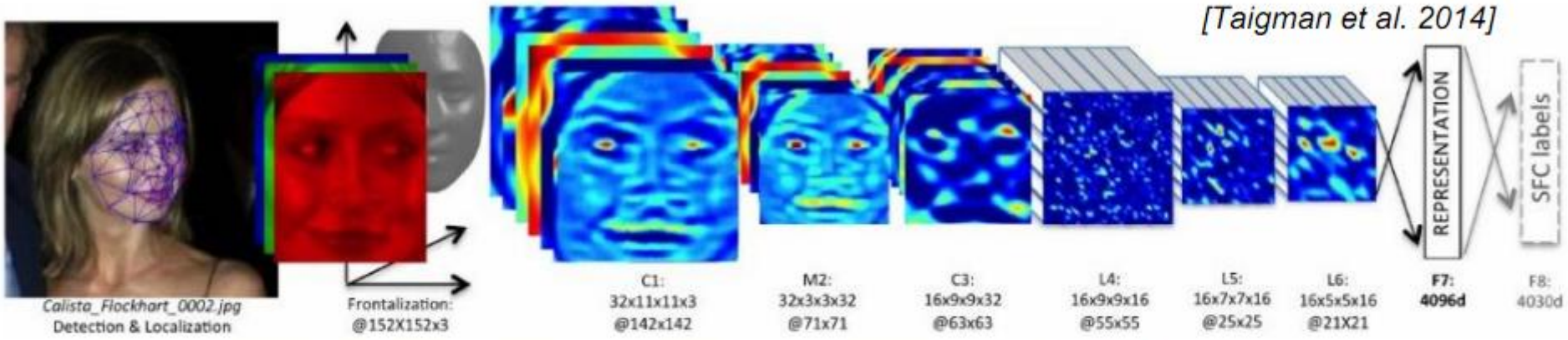
Deep Belief Net

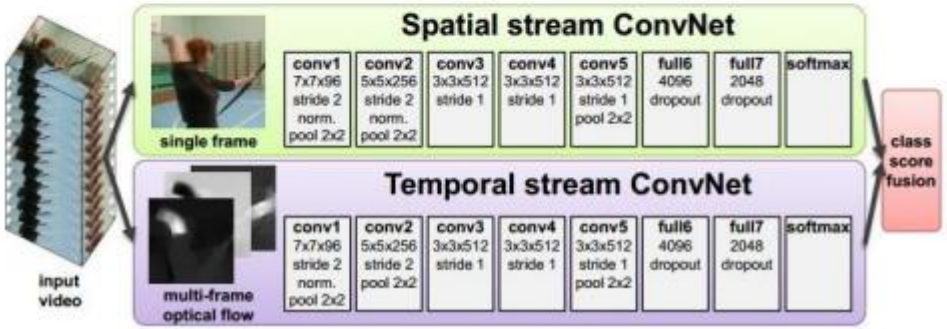
Restricted BM

BayesNP

ConvNets

Fast-forward to today: ConvNets are everywhere





[Simonyan et al. 2014]

[Goodfellow 2014]

ConvNets

Fast-forward to today: ConvNets are everywhere

Classification

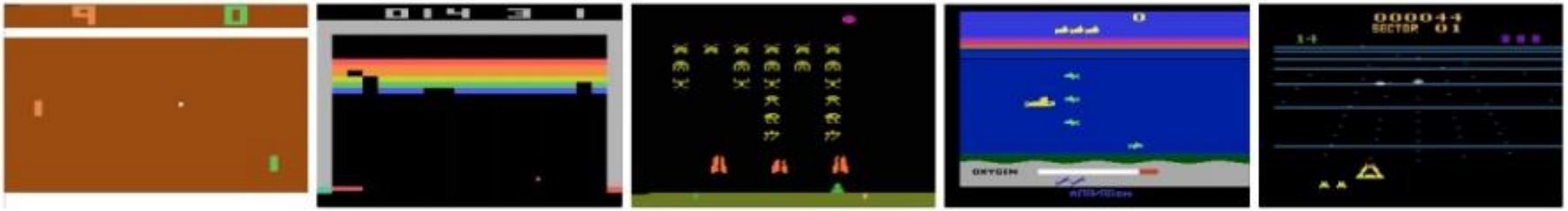
Retrieval

[Krizhevsky 2012]

ConvNets

Fast-forward to today: ConvNets are everywhere

[Toshev, Szegedy 2014]



[Mnih 2013]

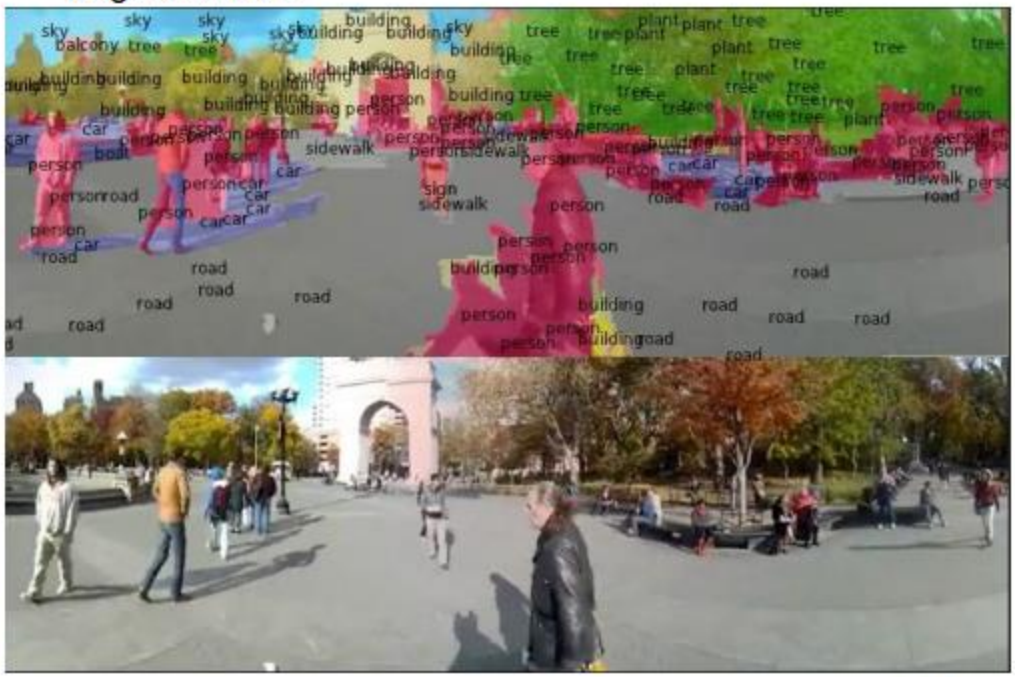
ConvNets

Fast-forward to today: ConvNets are everywhere

Detection

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

Segmentation



[Farabet et al., 2012]

ConvNets

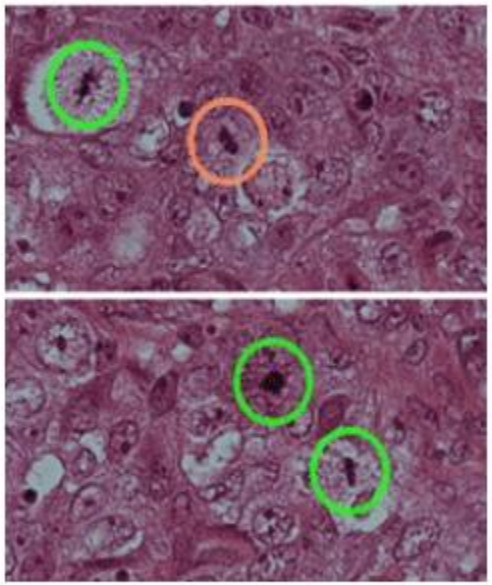
Fast-forward to today: ConvNets are everywhere

self-driving cars

NVIDIA Tegra X1

ConvNets

Fast-forward to today: ConvNets are everywhere

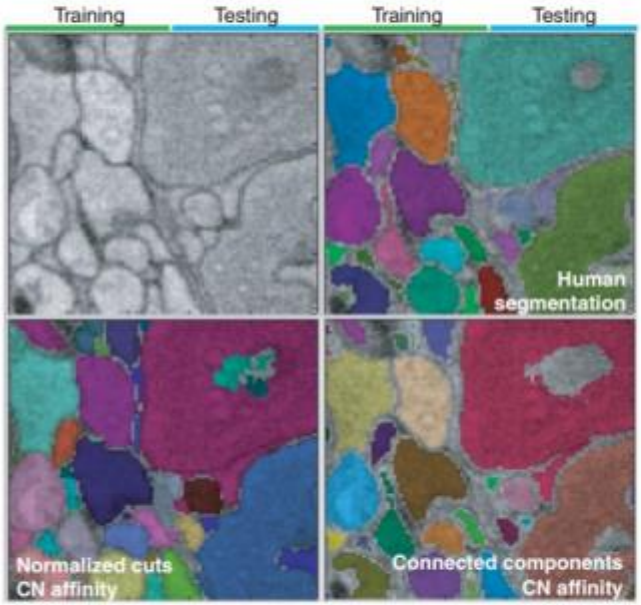


[Ciresan et al. 2013]

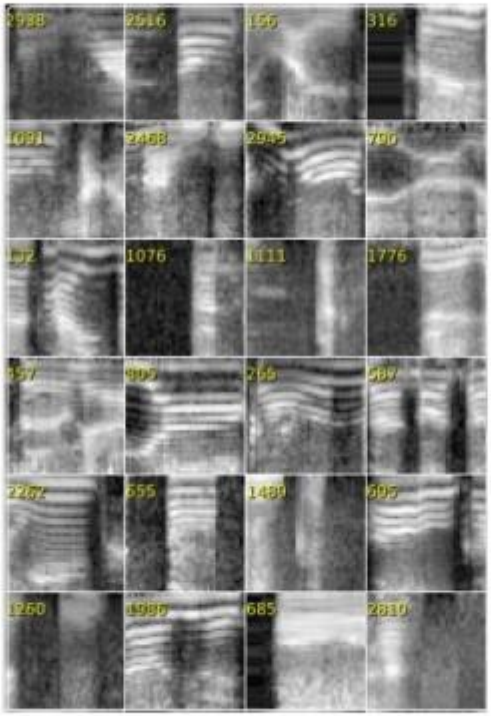
[Sermanet et al. 2011]
[Ciresan et al.]

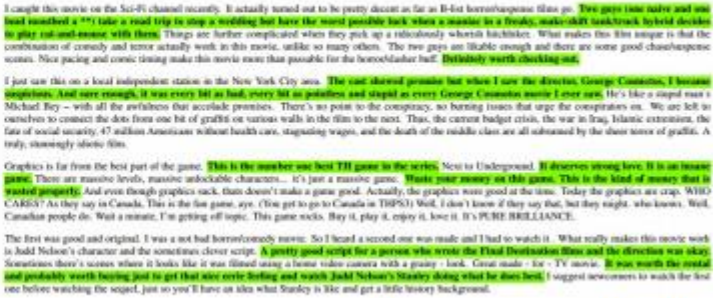
ConvNets

Fast-forward to today: ConvNets are everywhere

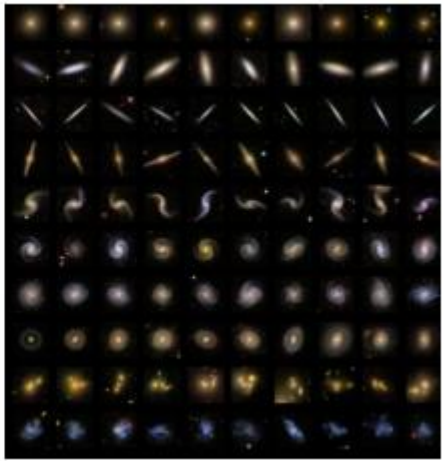


[Turaga et al., 2010]





[Denil et al. 2014]

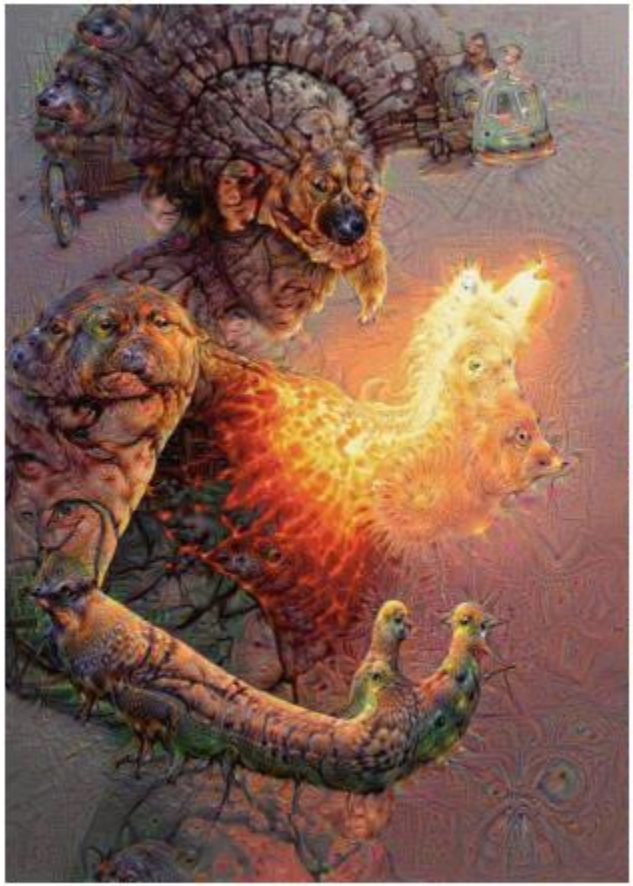


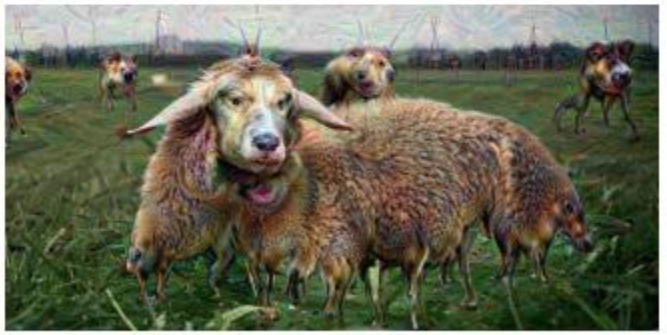
ConvNets

Image Captioning

[Vinyals et al., 2015]

ConvNets





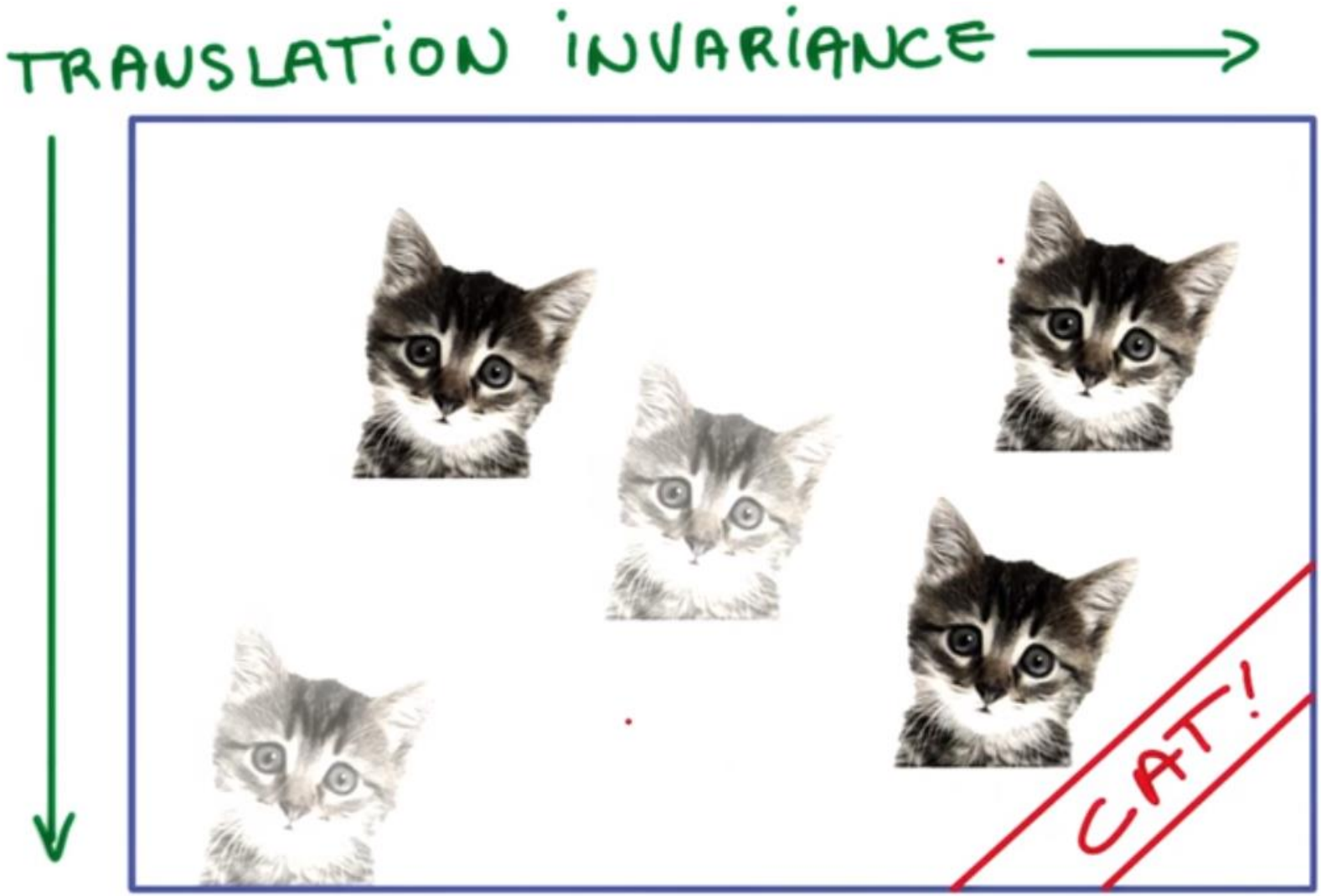
reddit.com/r/deepdream

Statistical Invariance

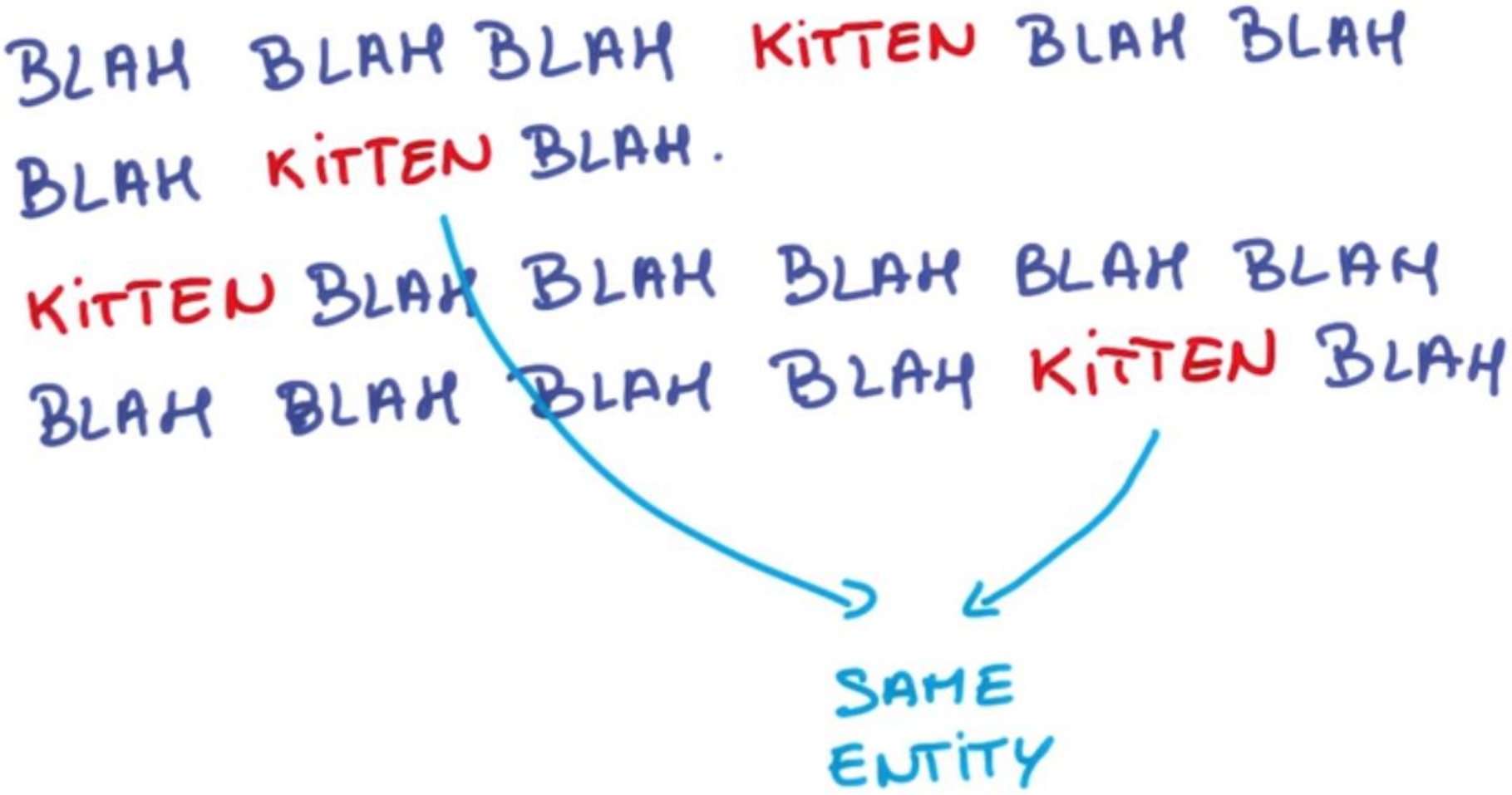
Statistical Invariance

Statistical Invariance

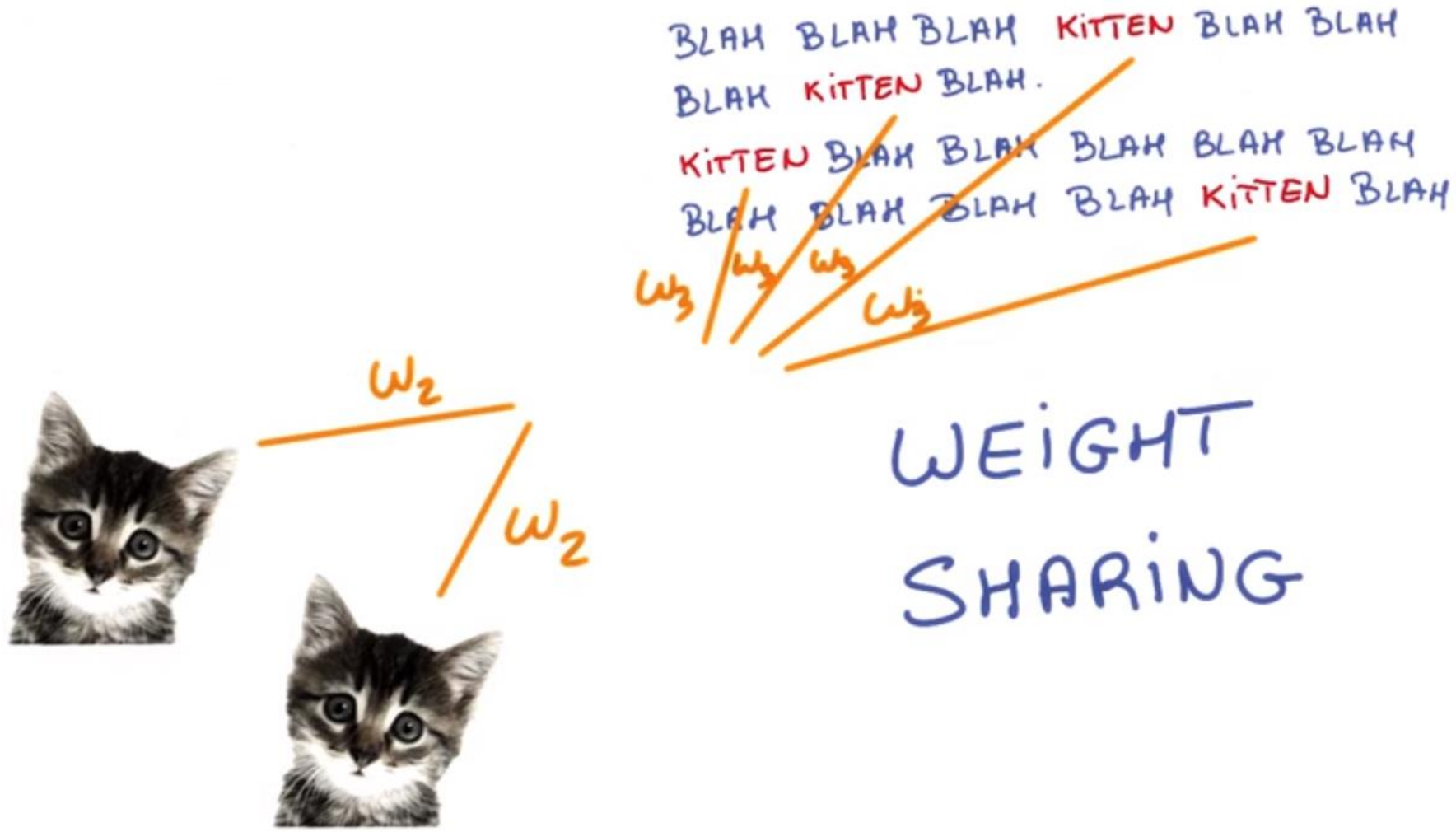
Statistical Invariance



Statistical Invariance

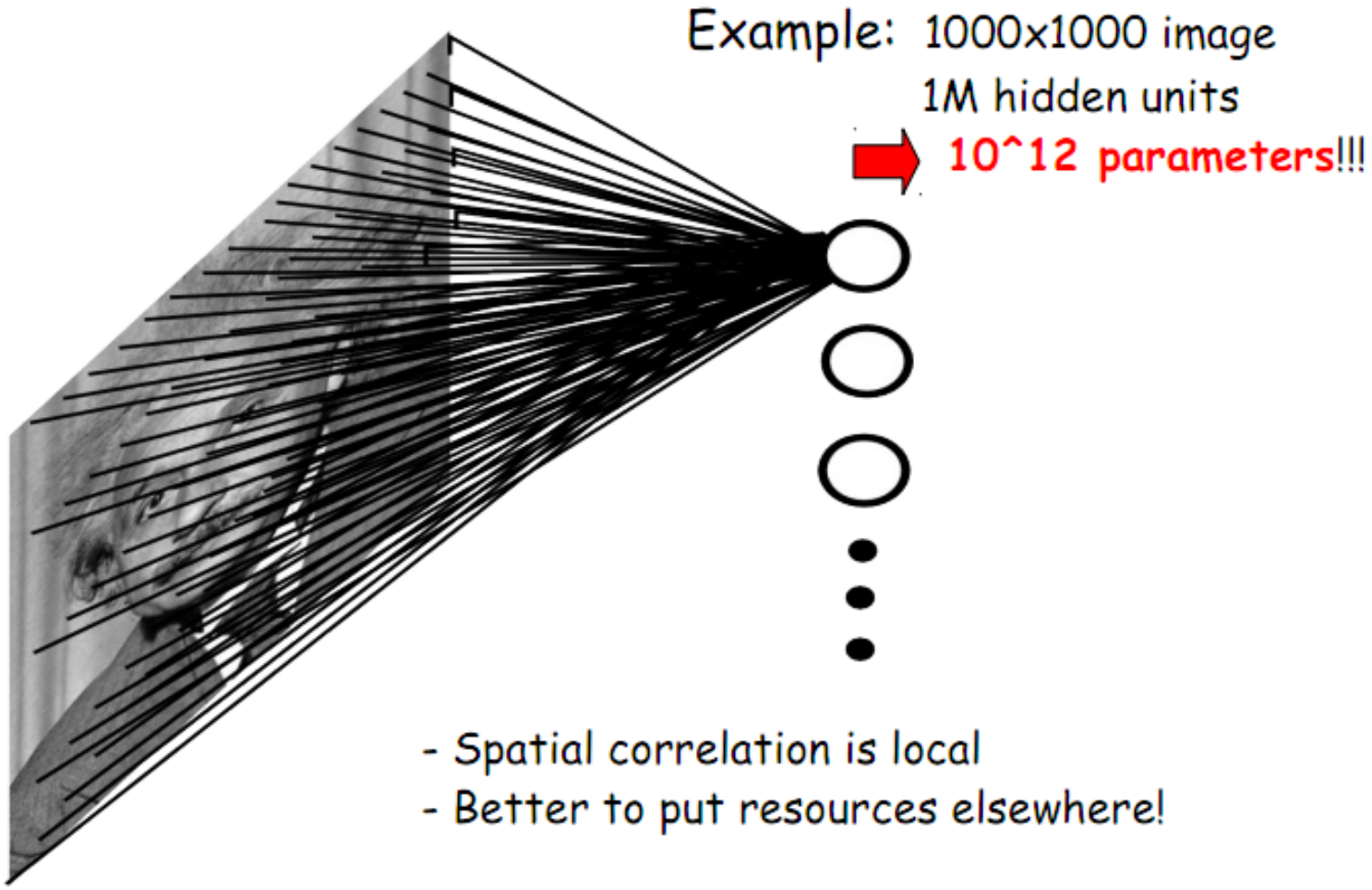


Statistical Invariance

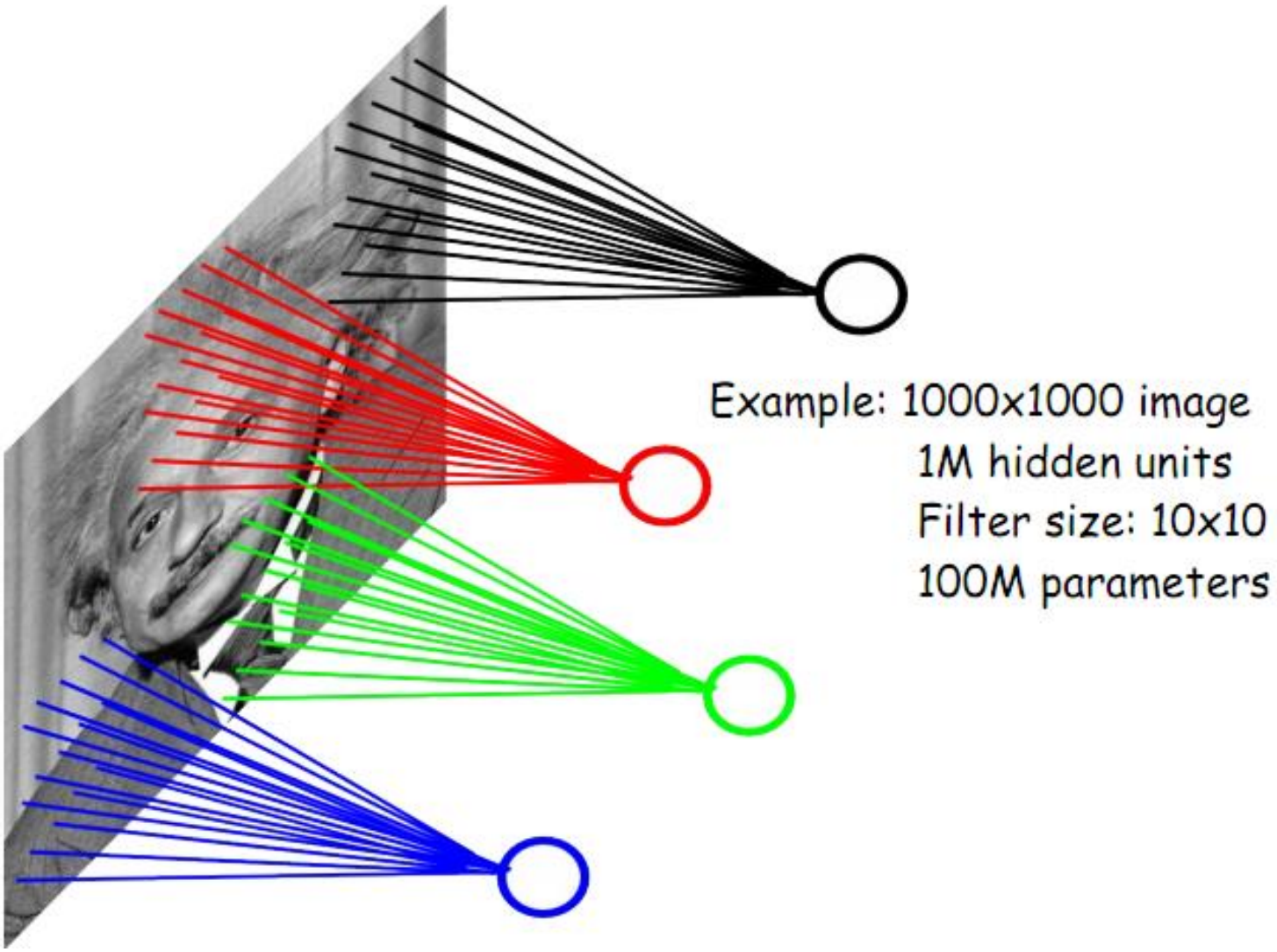


Neural Networks

FULLY CONNECTED NEURAL NET

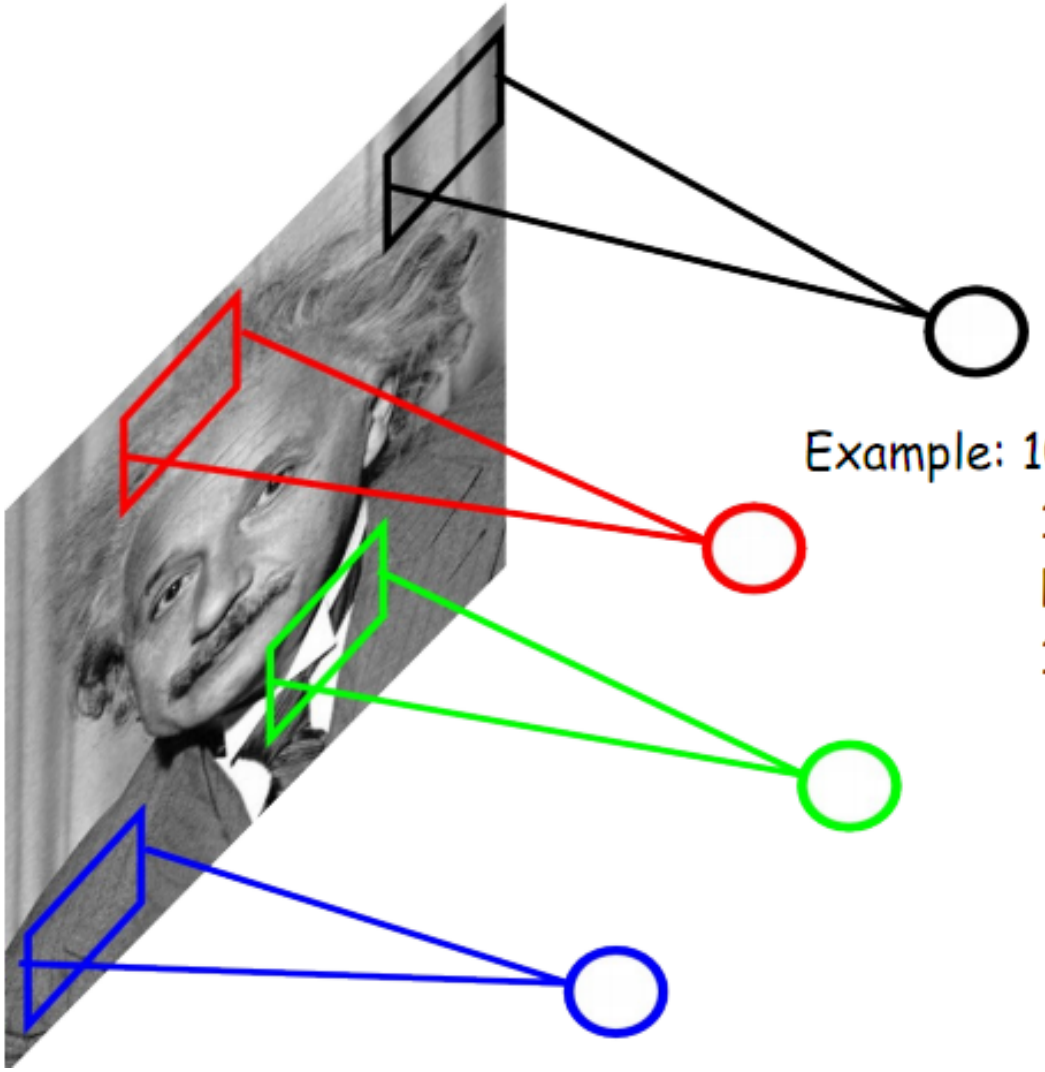


Local Connectivity



Local Connectivity

LOCALLY CONNECTED NEURAL NET

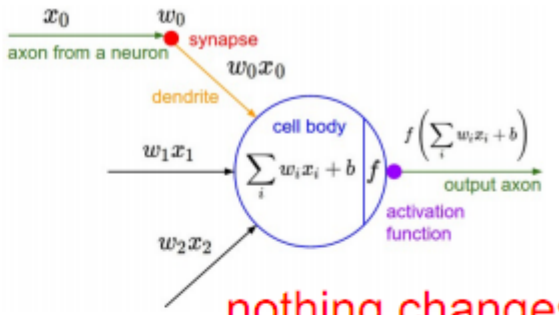


Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

ConvNets

Convolutional Neural Networks are just Neural Networks BUT:

1. **Local connectivity**



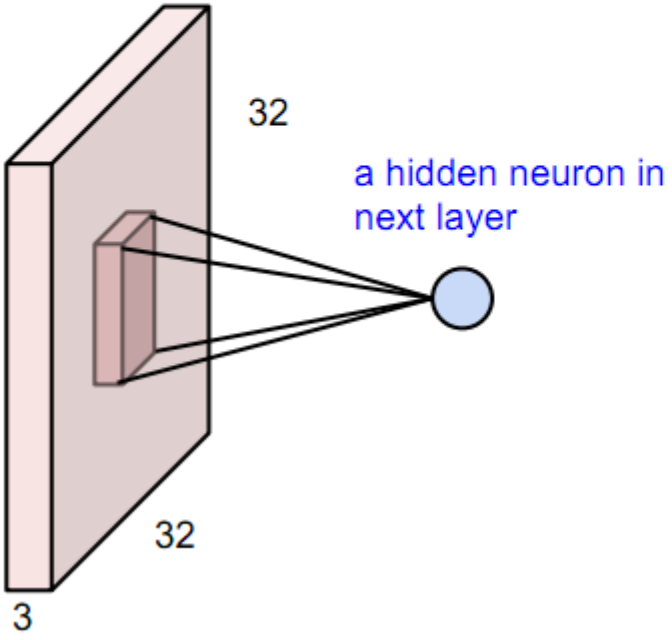


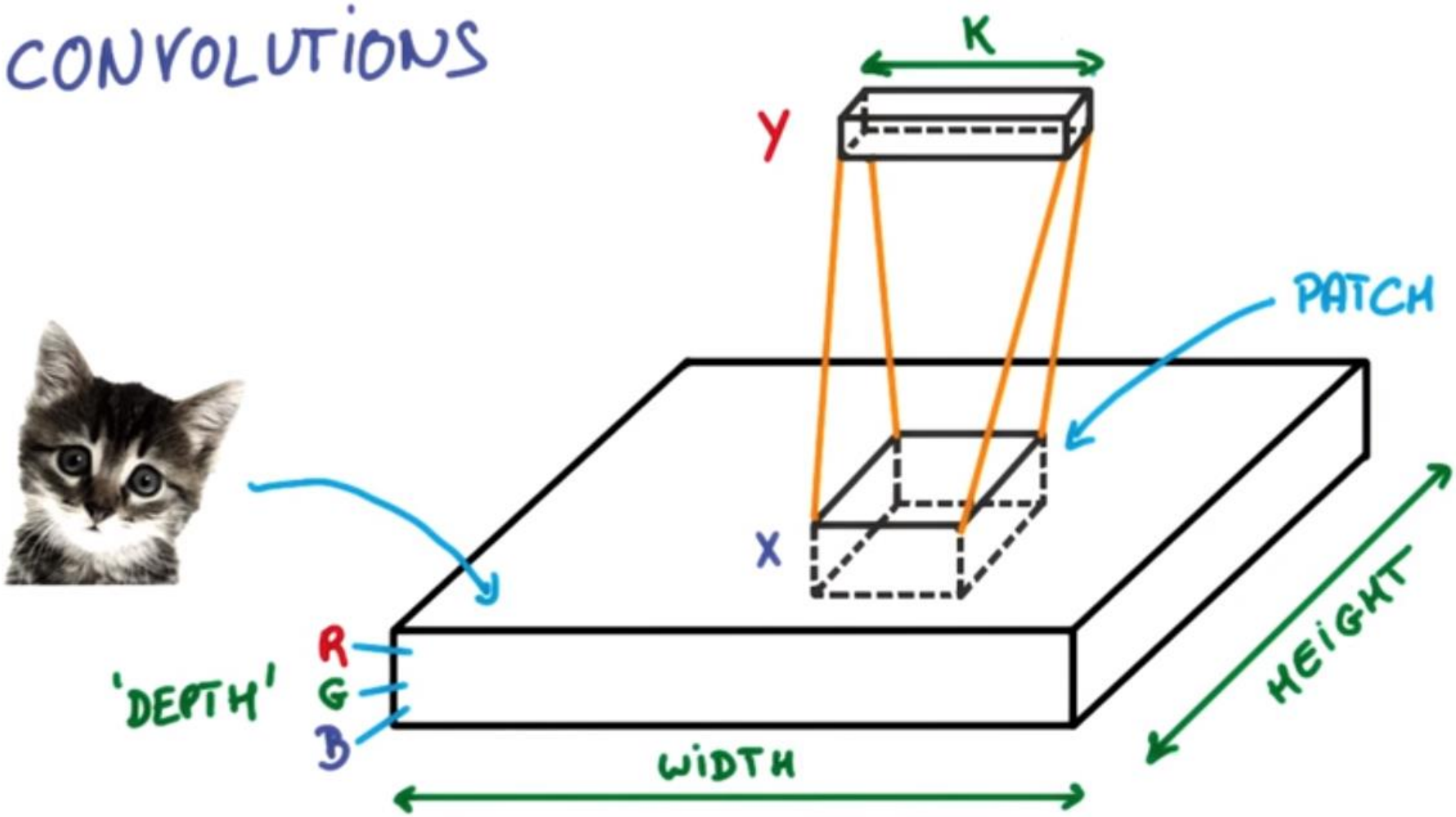
image: 32x32x3 volume
before: full connectivity: 32x32x3 weights

now: one neuron will connect to, e.g. 5x5x3 chunk and only have 5x5x3 weights.

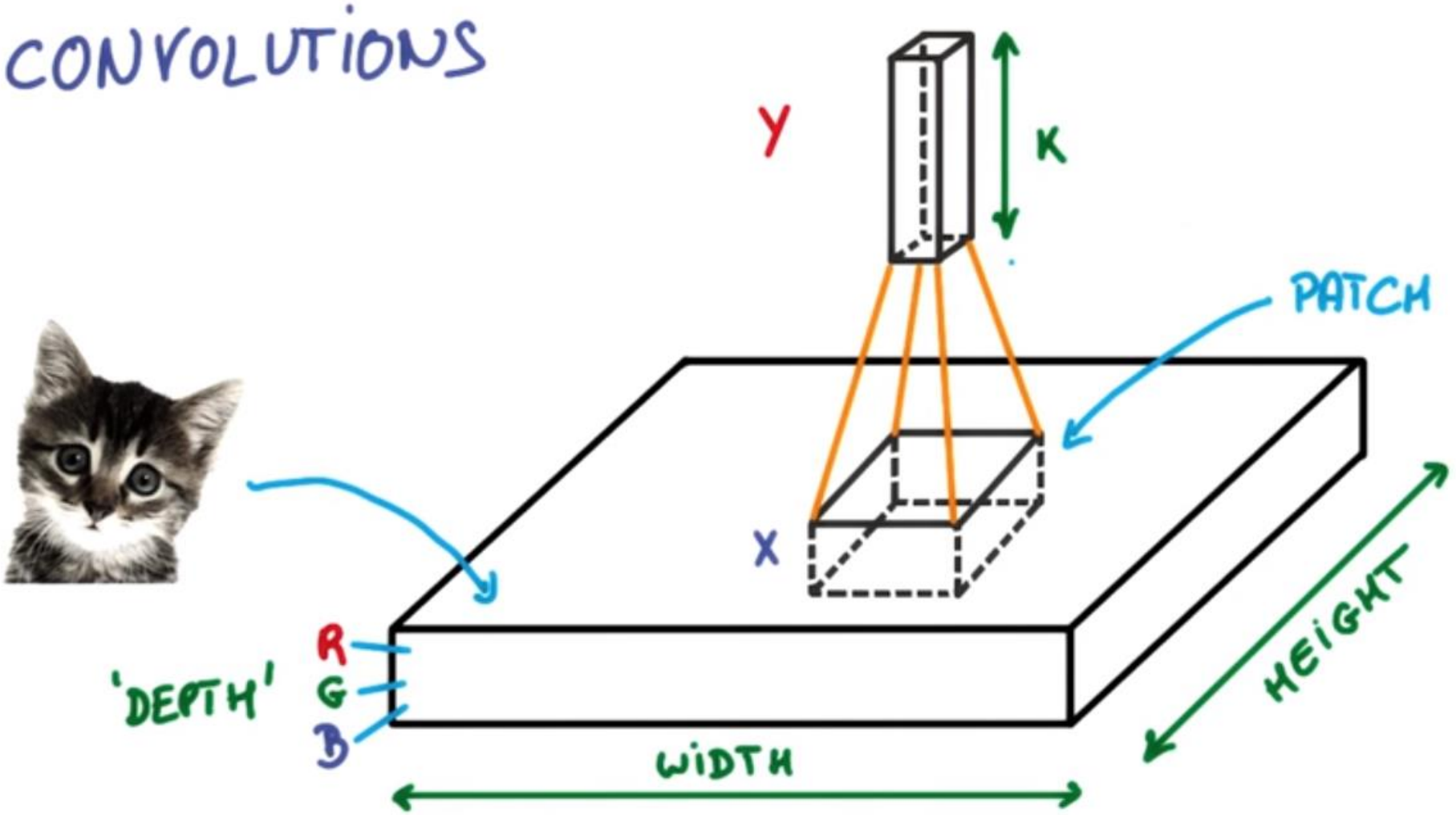
note that connectivity is:

- local in space (5x5 inside 32x32)
- but full in depth (all 3 depth channels)

ConvNets



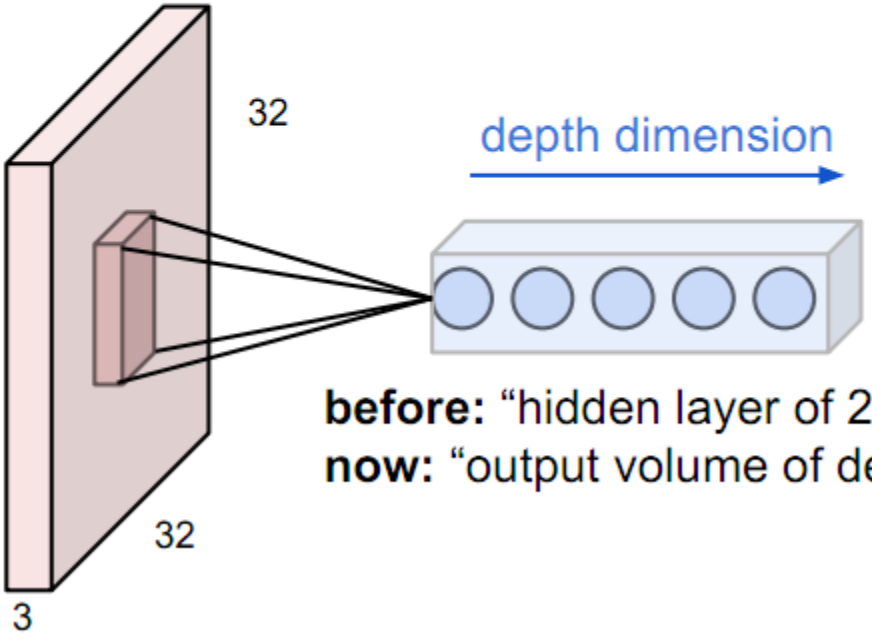
ConvNets



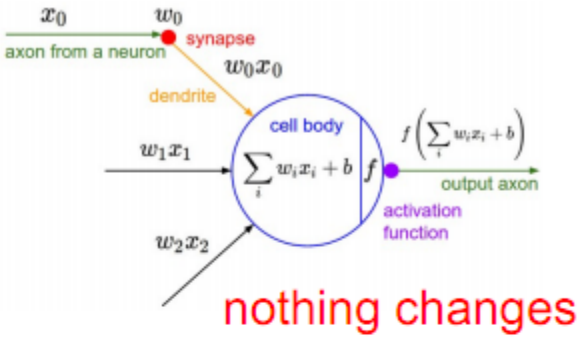
ConvNets

Convolutional Neural Networks are just Neural Networks BUT:

1. **Local connectivity**



before: “hidden layer of 200 neurons”
now: “output volume of depth 200”

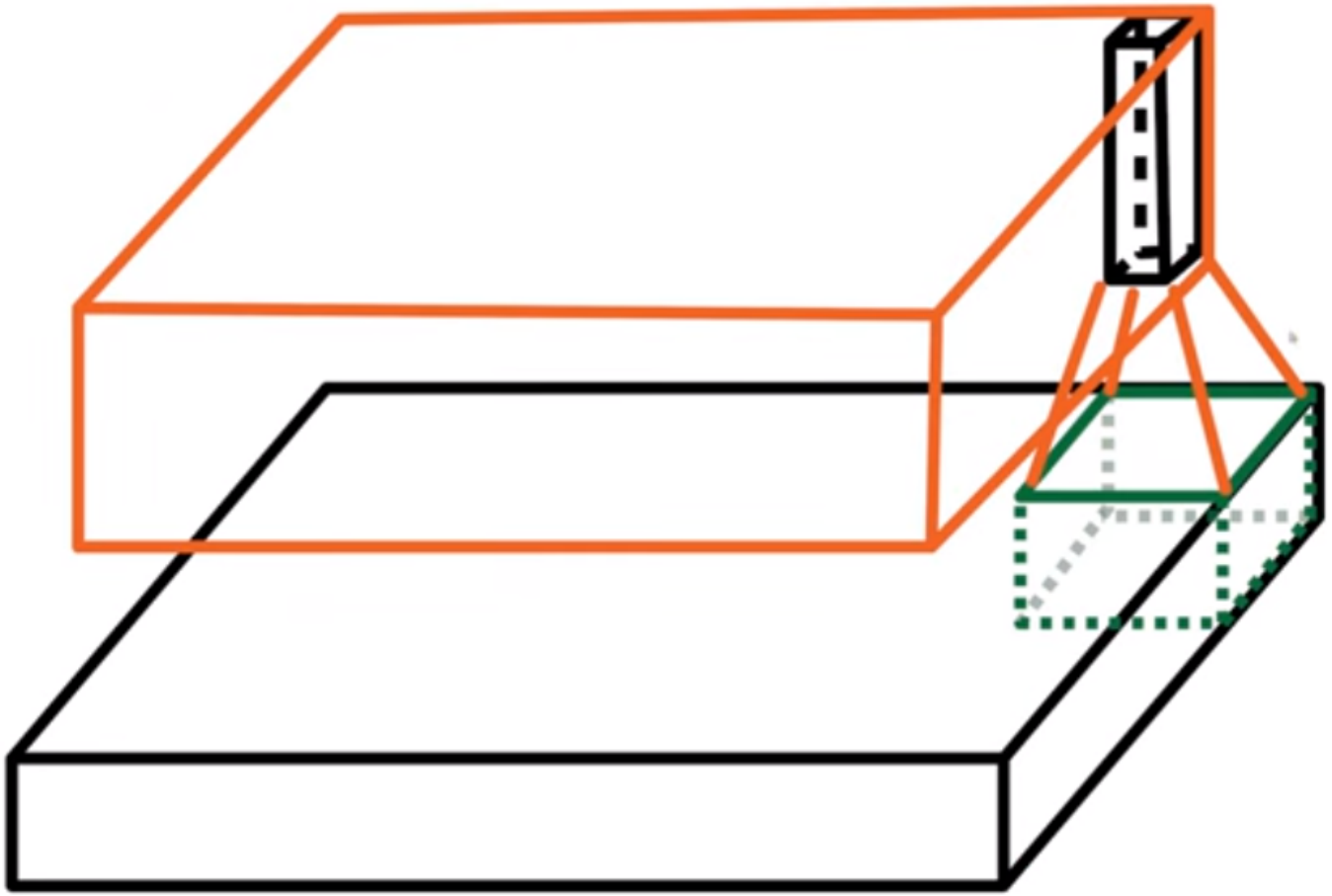


nothing changes

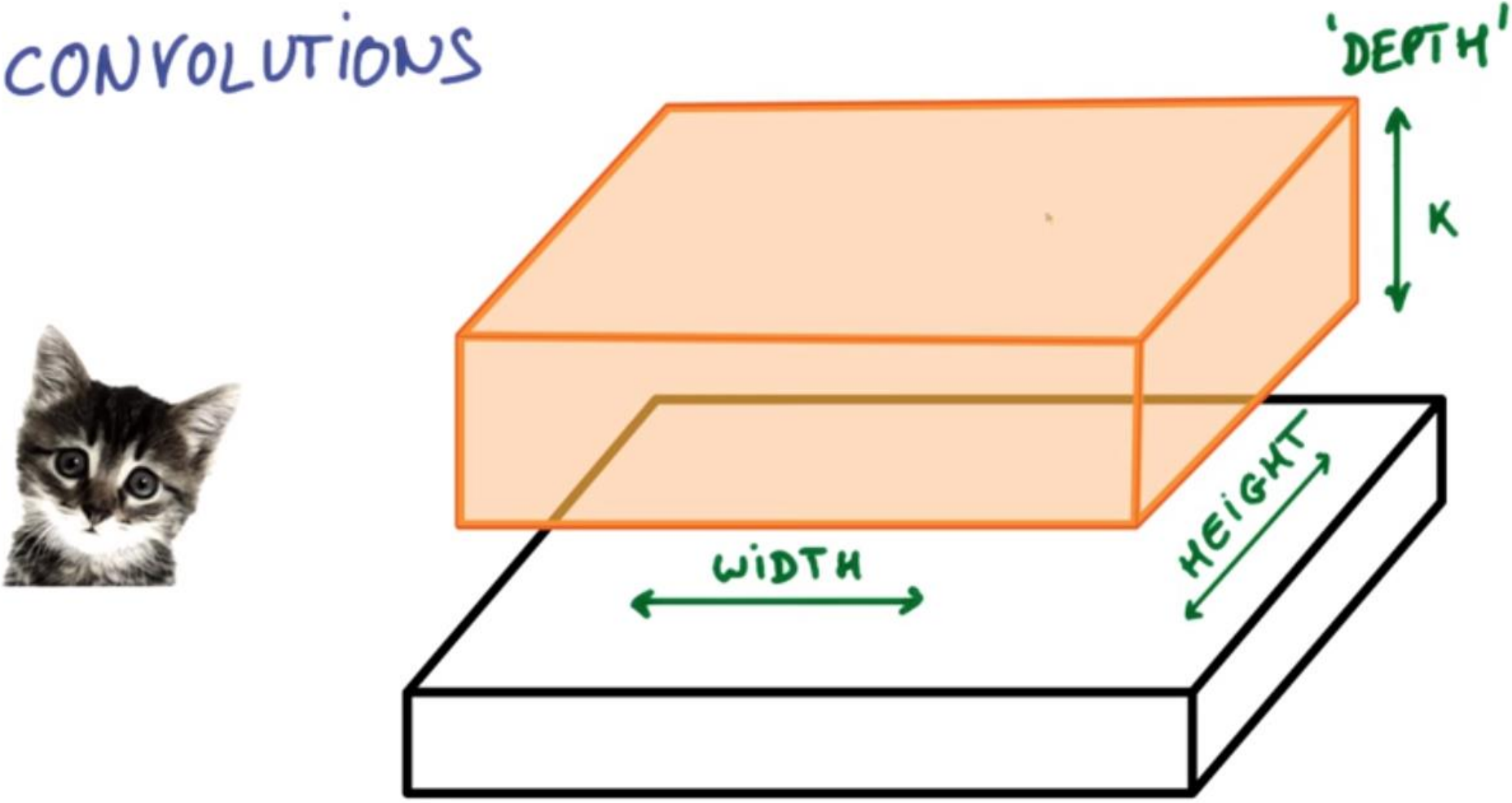
Multiple neurons all looking at the same region of the input volume, stacked along depth.

ConvNets

CONVOLUTIONS



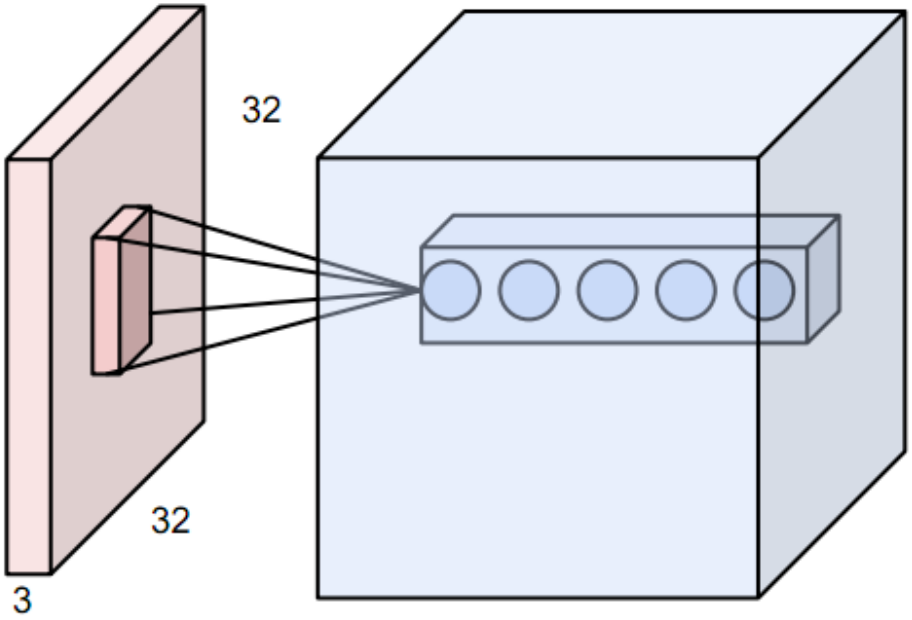
ConvNets

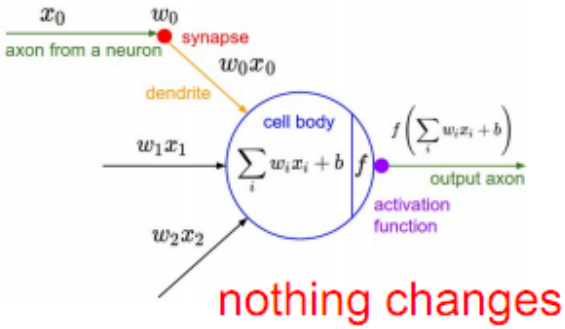


ConvNets

Convolutional Neural Networks are just Neural Networks BUT:

1. **Local connectivity**

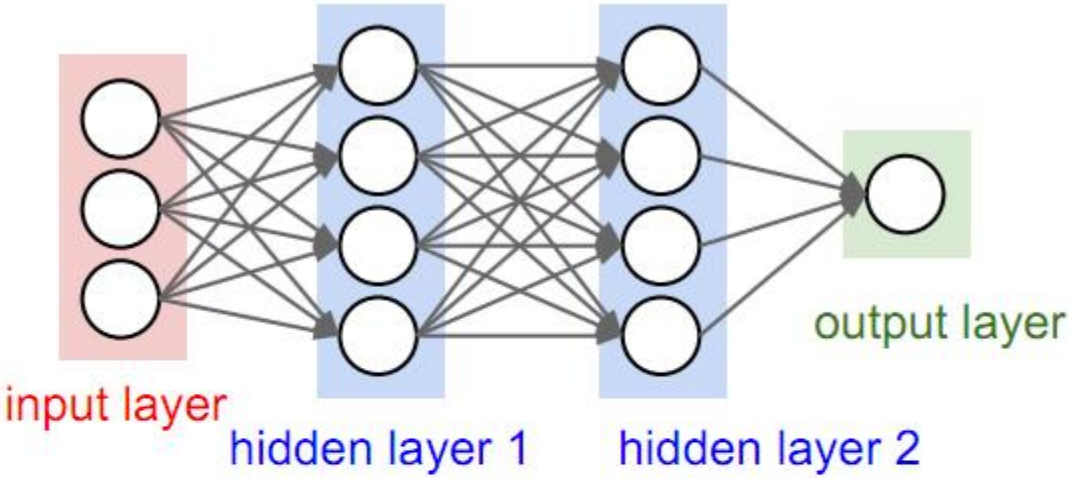




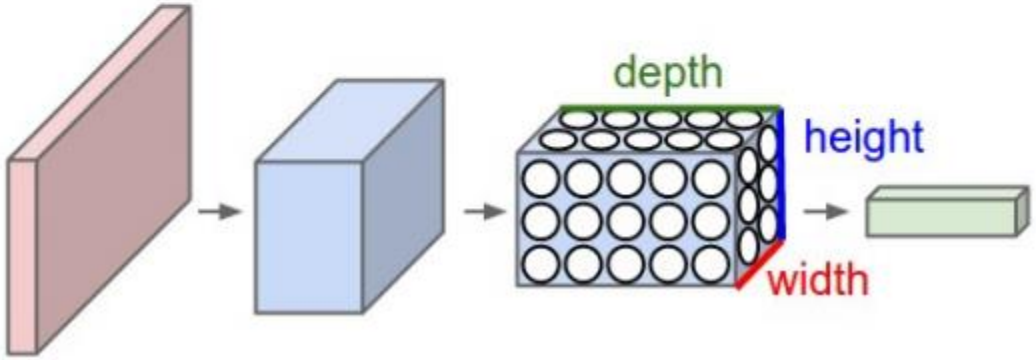
These form a single [1 x 1 x depth] "depth column" in the output volume

ConvNets

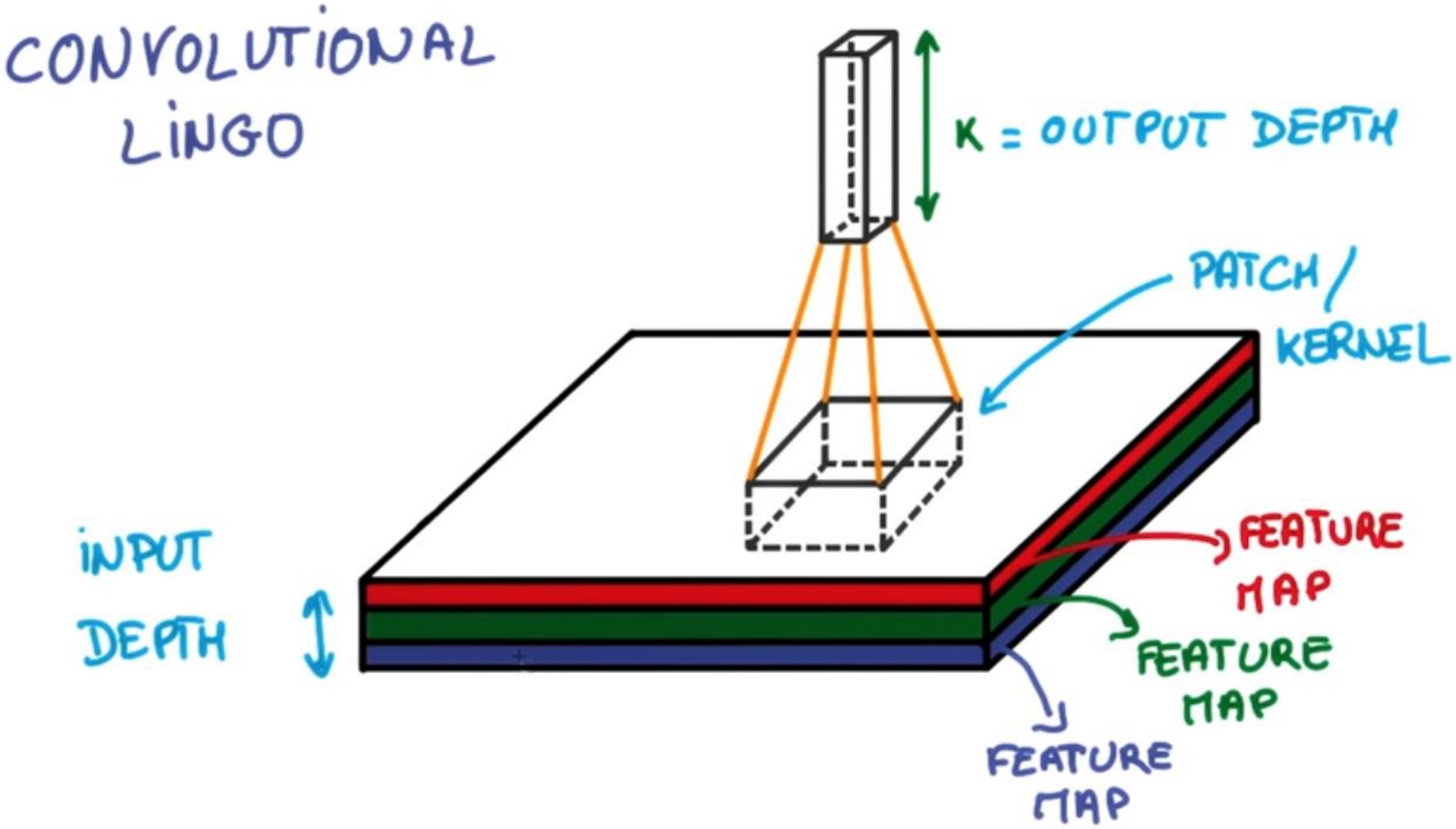
before:



now:

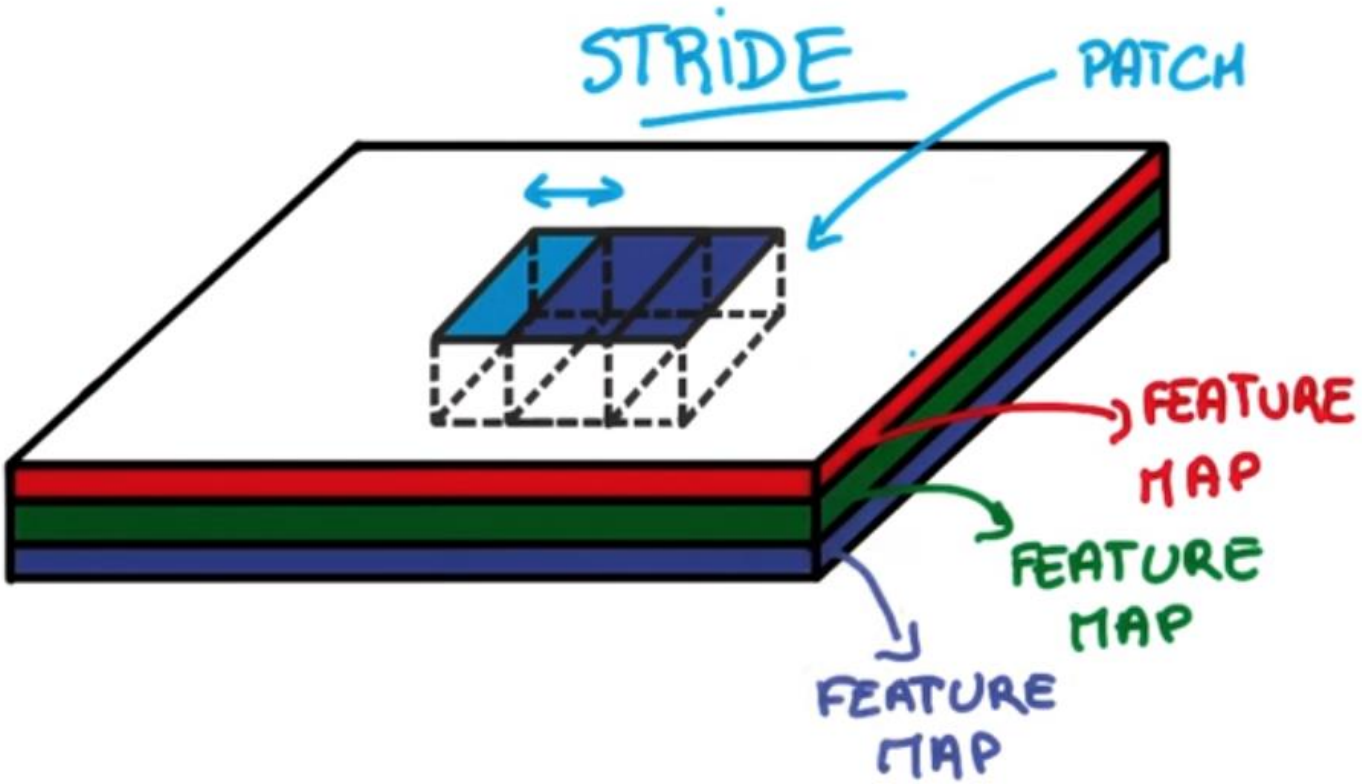


Feature Map


CONVOLUTIONAL
LINGO
K = OUTPUT DEPTH
PATCH /
KERNEL
INPUT
DEPTH
FEATURE
MAP
FEATURE
MAP
FEATURE
MAP

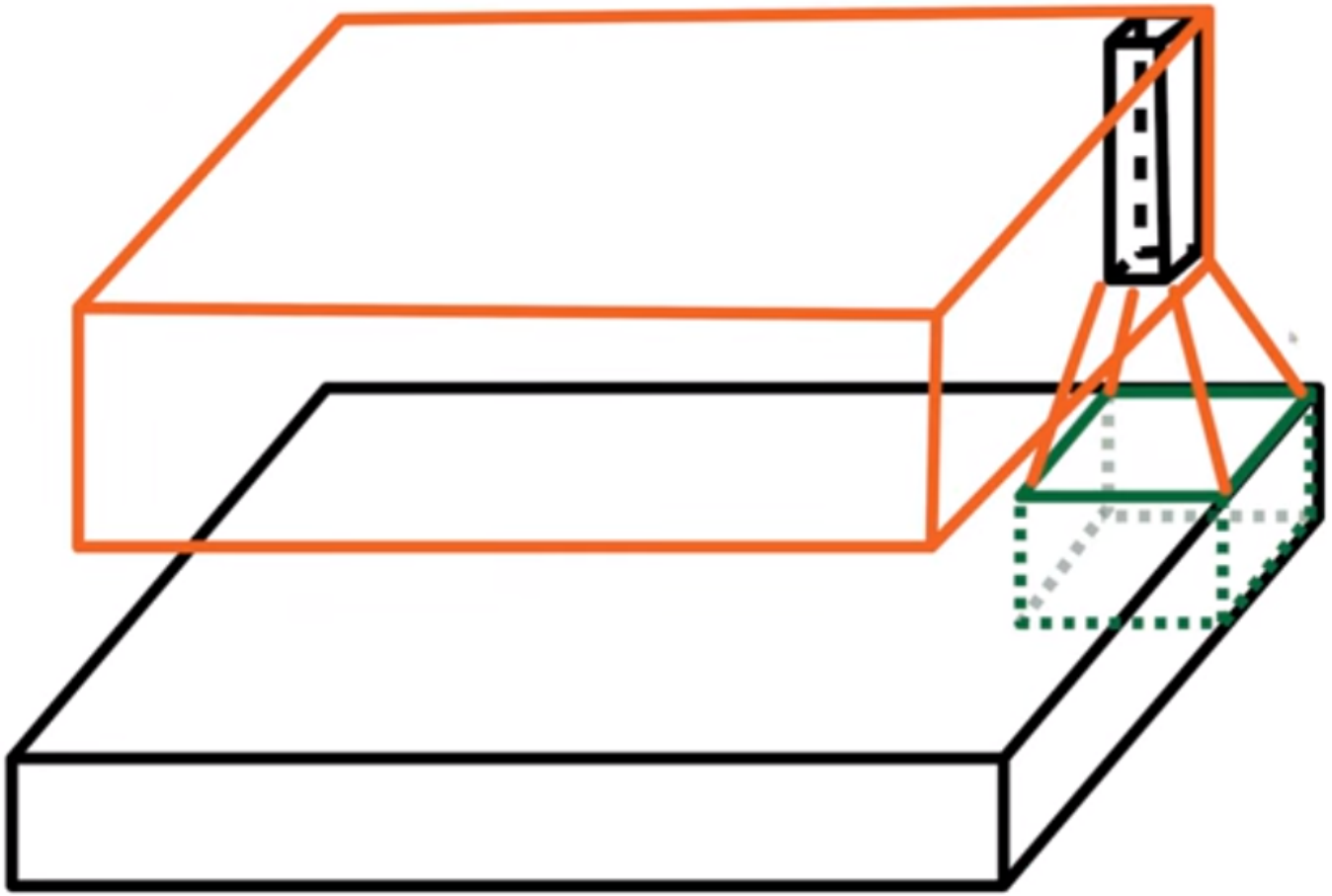
Strike

CONVOLUTIONAL LINGO

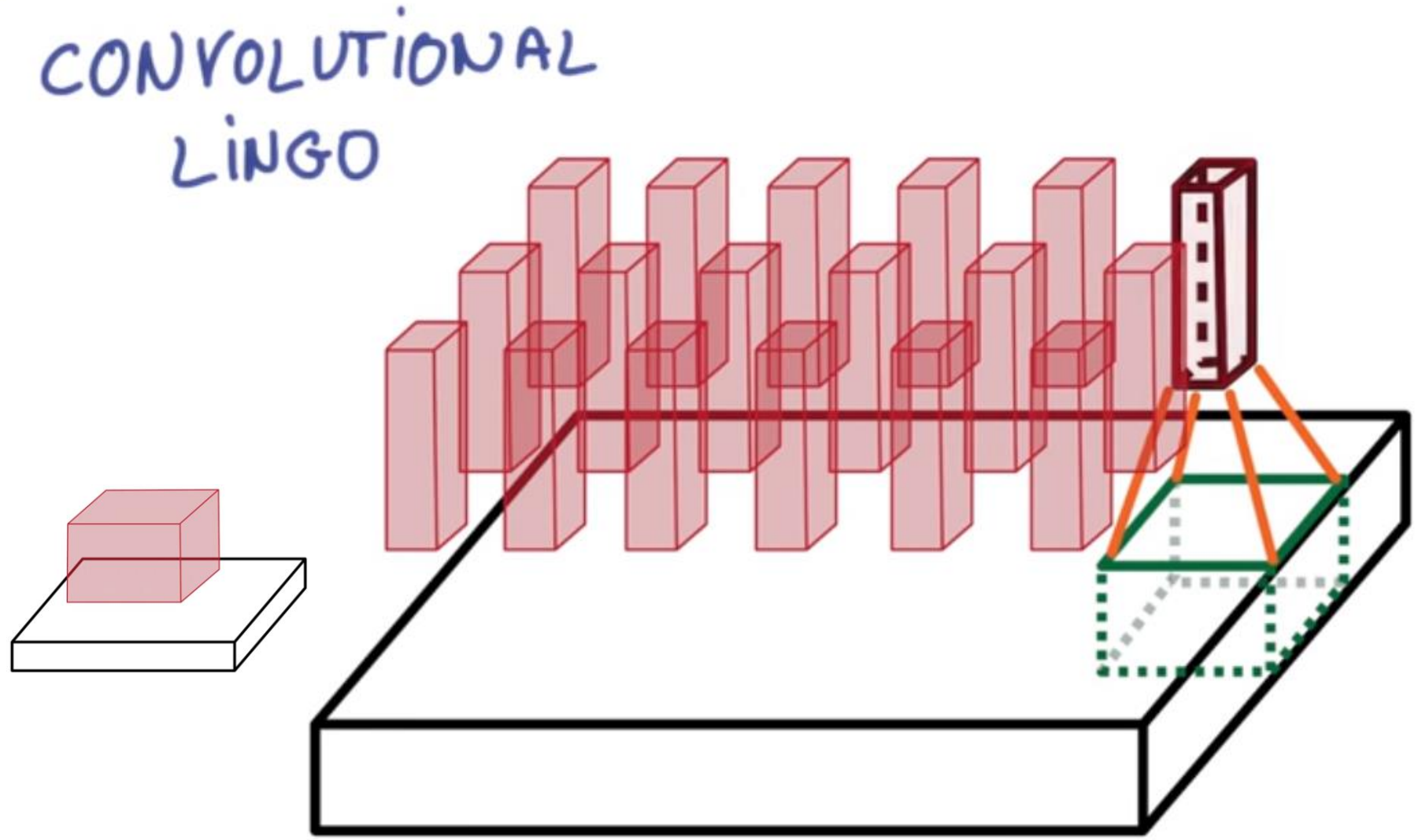


Strike

CONVOLUTIONS

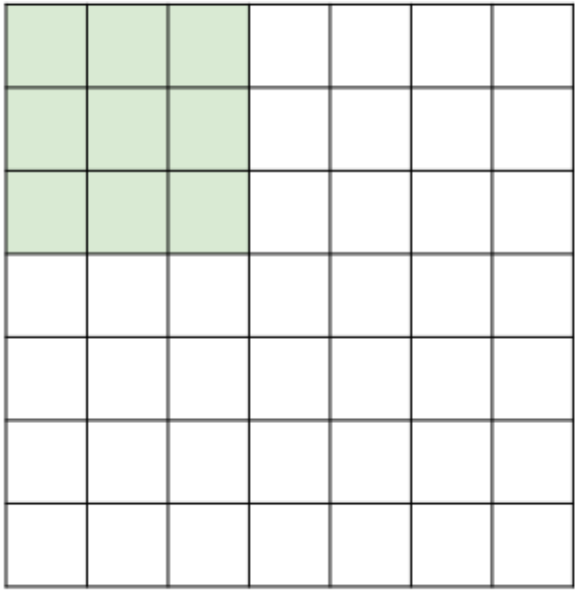


Strike



Strike

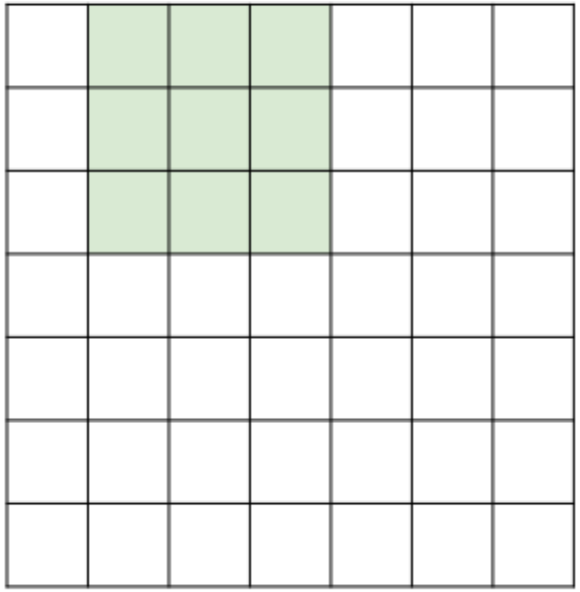
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

Strike

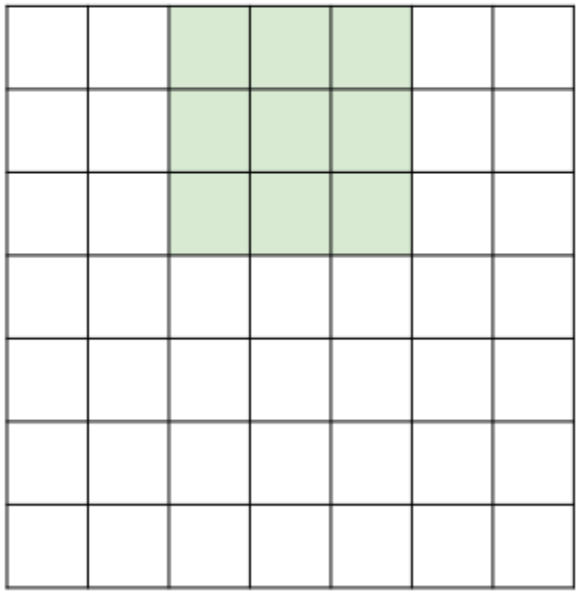
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

Strike

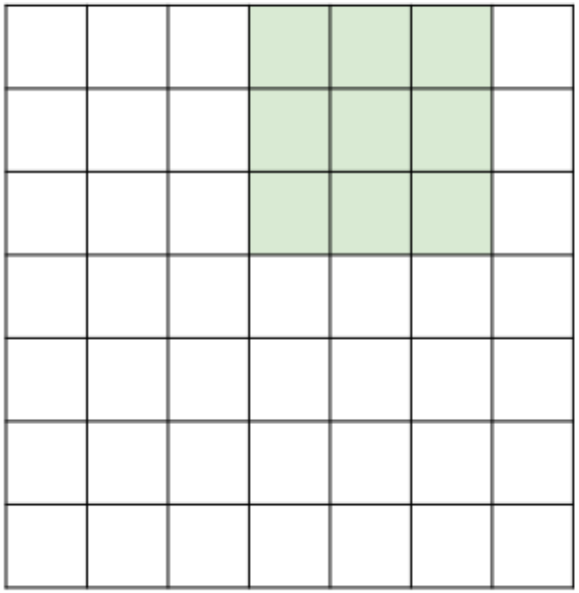
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

Strike

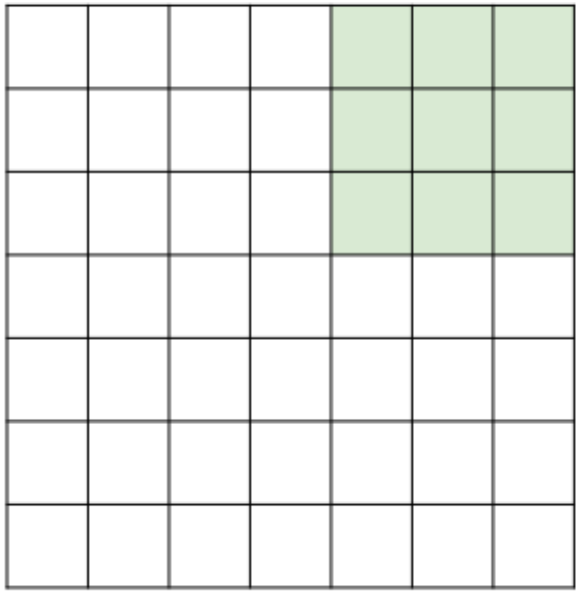
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1

Strike

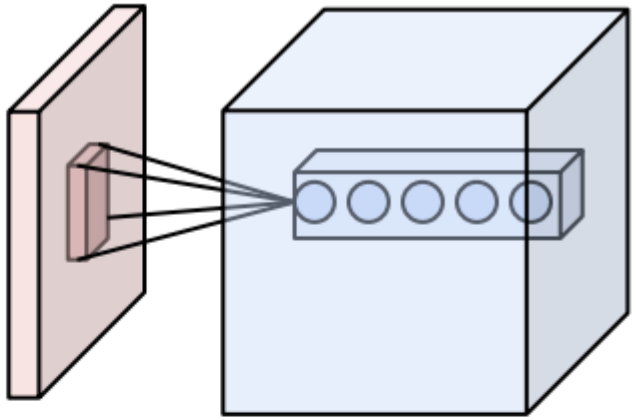
Replicate this column of hidden neurons across space, with some **stride.**



7x7 input
assume 3x3 connectivity, stride 1
=> **5x5 output**

Strike

Examples time:

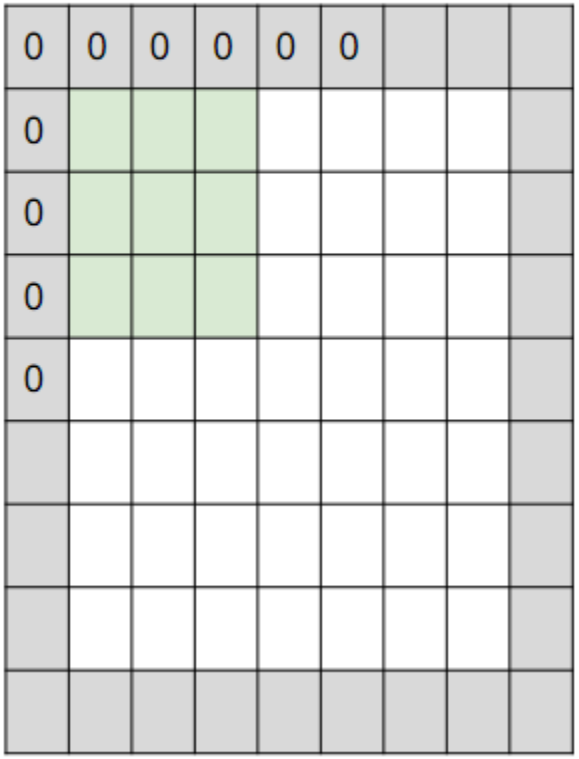


Input volume: **32x32x3**
Receptive fields: **5x5**, **stride 1**
Number of neurons: **5**

Output volume: (32 - 5) / 1 + 1 = 28, so: **28x28x5**
How many weights for each of the 28x28x5 neurons?

Strike

In practice: Common to zero pad the border



(in each channel)

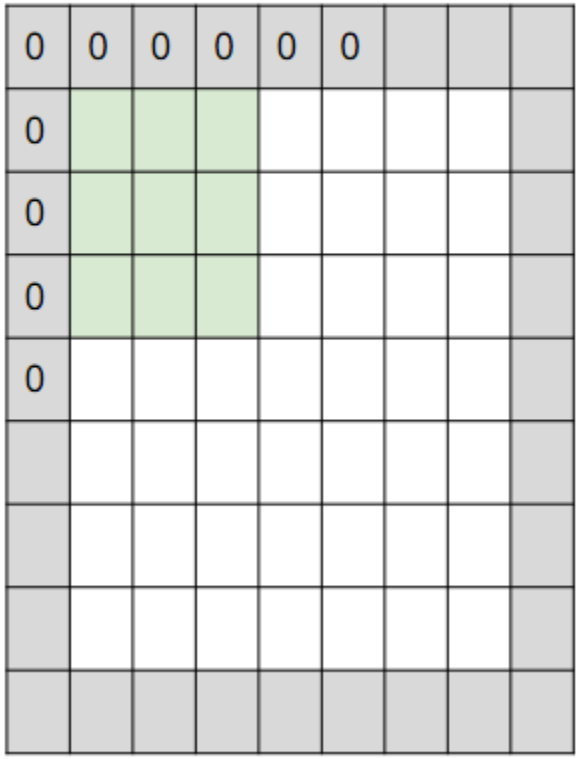
e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border => what is the output?

7x7 => preserved size!

in general, common to see stride 1, size F, and zero-padding with (F-1)/2.
(Will preserve input size spatially)

Strike

In practice: Common to zero pad the border



(in each channel)

e.g. input 7x7
neuron with receptive field 3x3, stride 1
pad with 1 pixel border => what is the output?

7x7 => preserved size!

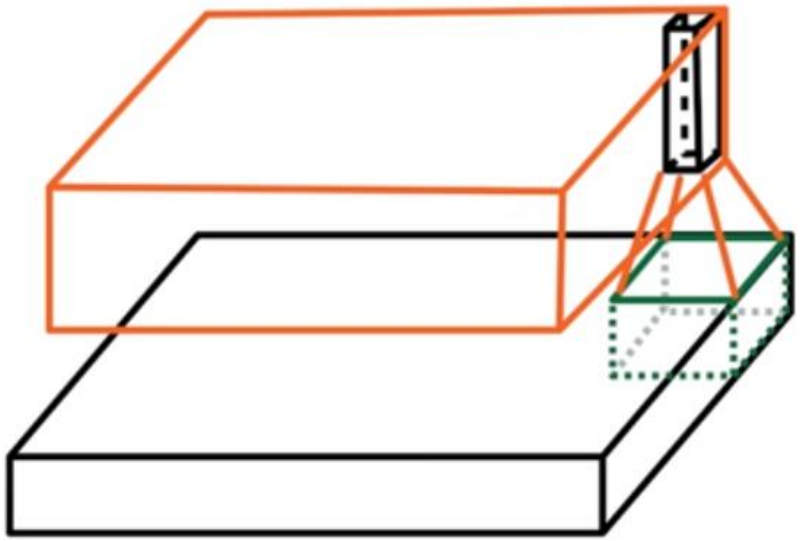
in general, common to see stride 1, size F, and zero-padding with (F-1)/2.
(Will preserve input size spatially)

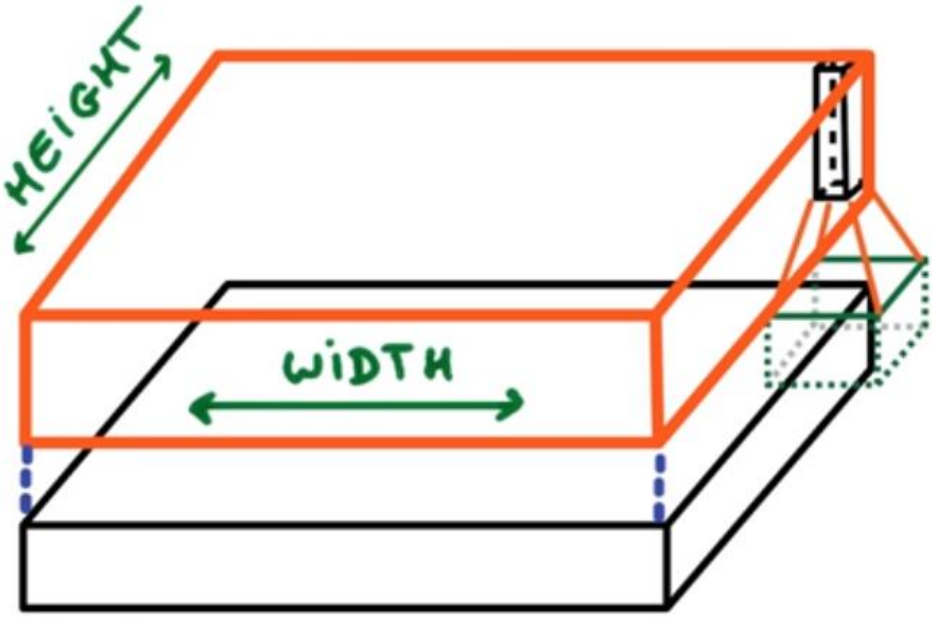
Padding

CONVOLUTIONAL LINGO

'VALID' PADDING

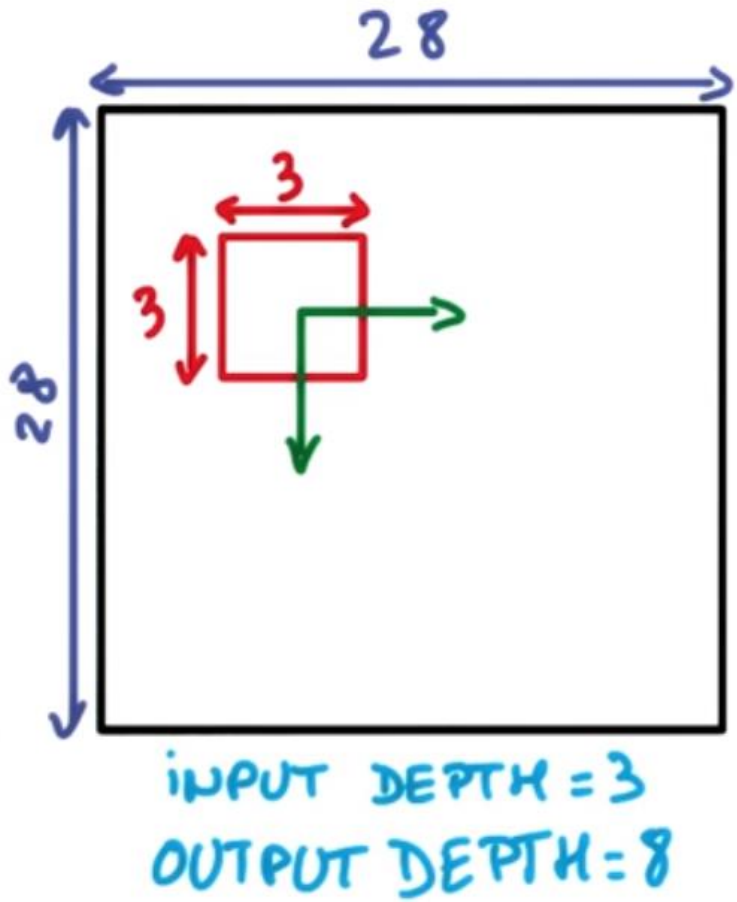
'SAME' PADDING





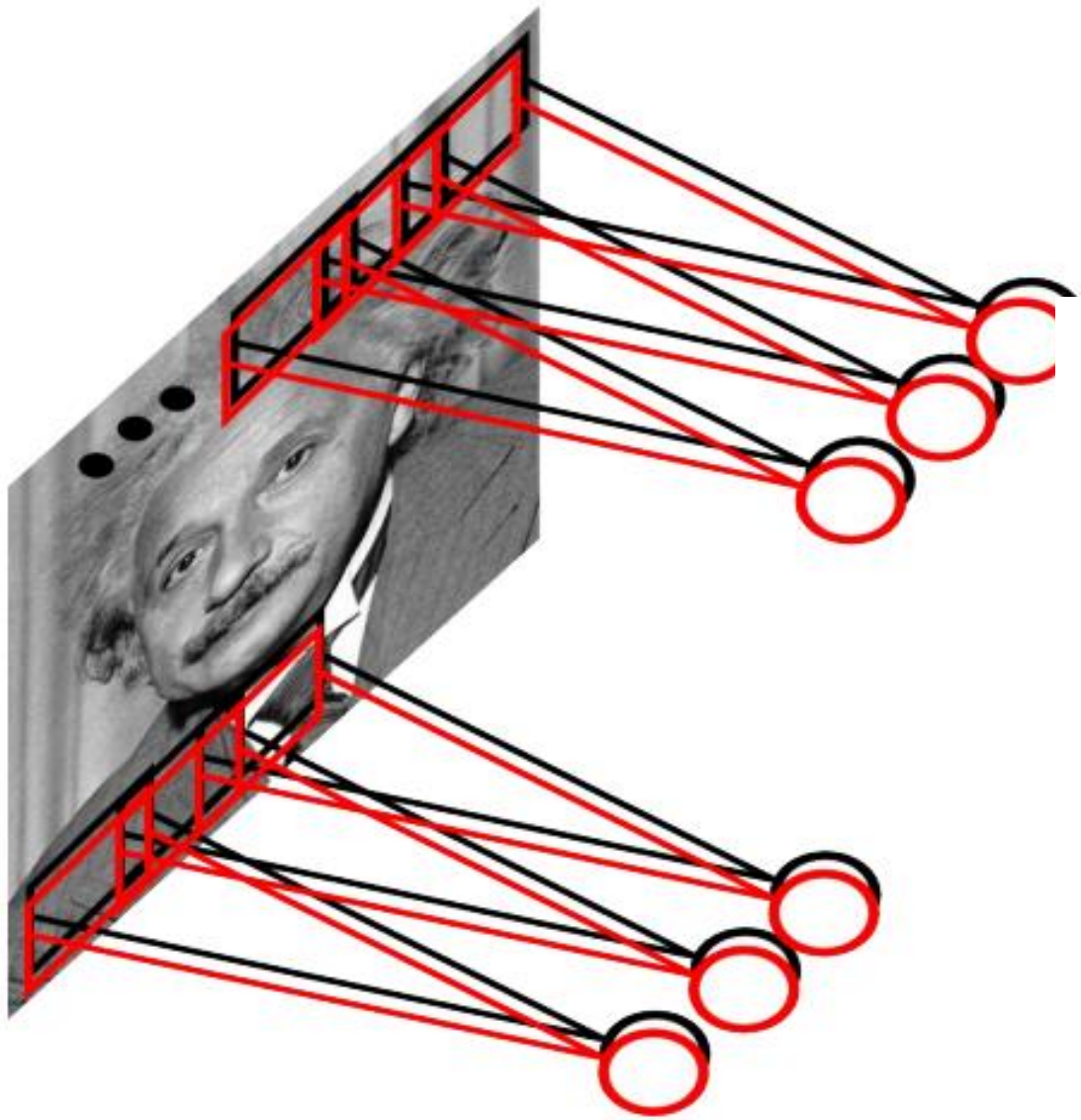
Quiz

STRIDES, DEPTH & PADDING



		OUTPUT		
PADDING	STRIDE	WIDTH	HEIGHT	DEPTH
'SAME'	1			
'VALID'	1			
'VALID'	2			

ConvNets

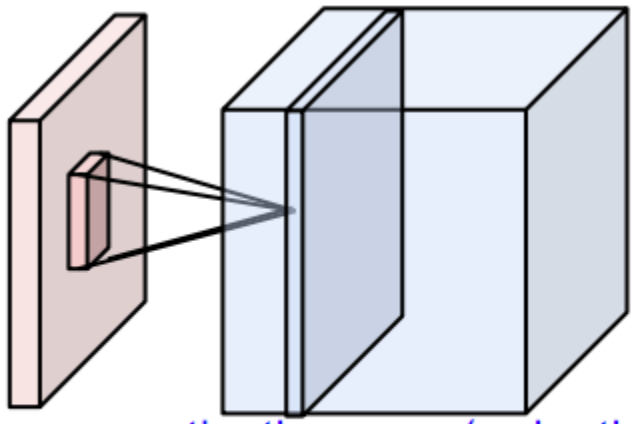


Share weights

ConvNets

These layers are called **Convolutional Layers**

1. Connect neurons only to local receptive fields
2. Use the same neuron weight parameters for neurons in each “depth slice” (i.e. across spatial positions)



one activation map (a depth slice), computed with one set of weights

Visualize CNNs

5x5 filters

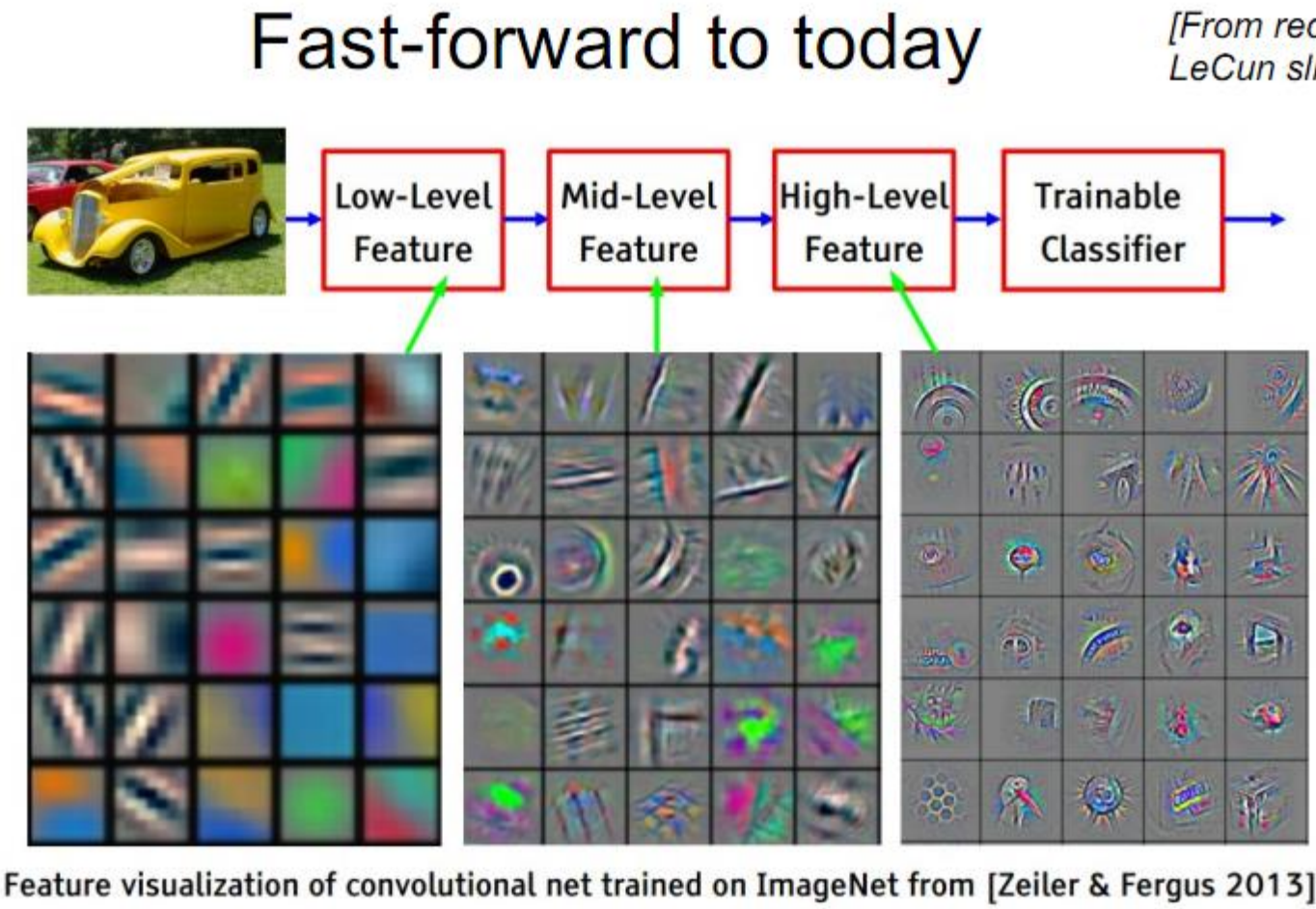
Can call the neurons “filters”

We call the layer convolutional because it is related to convolution of two signals (kind of):

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum a filter and the signal (image)
= np.dot(w,x) + b

Visualize CNNs

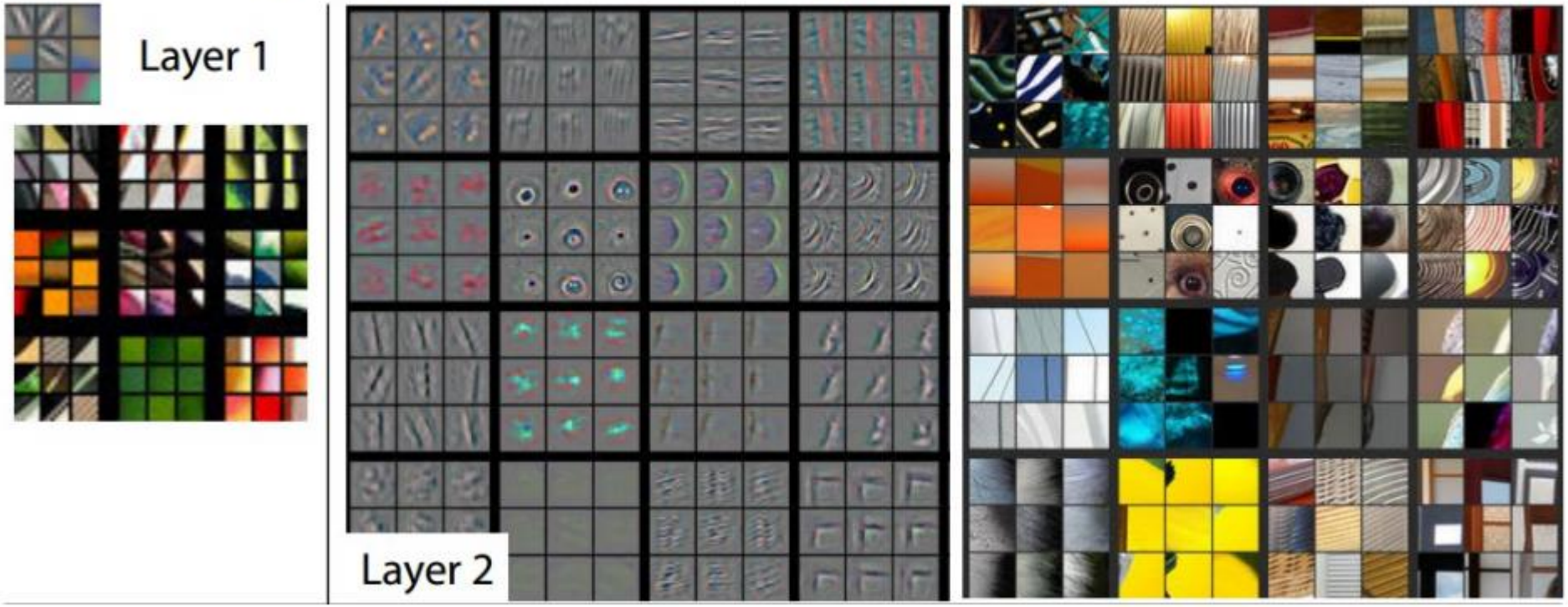


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Visualize CNNs

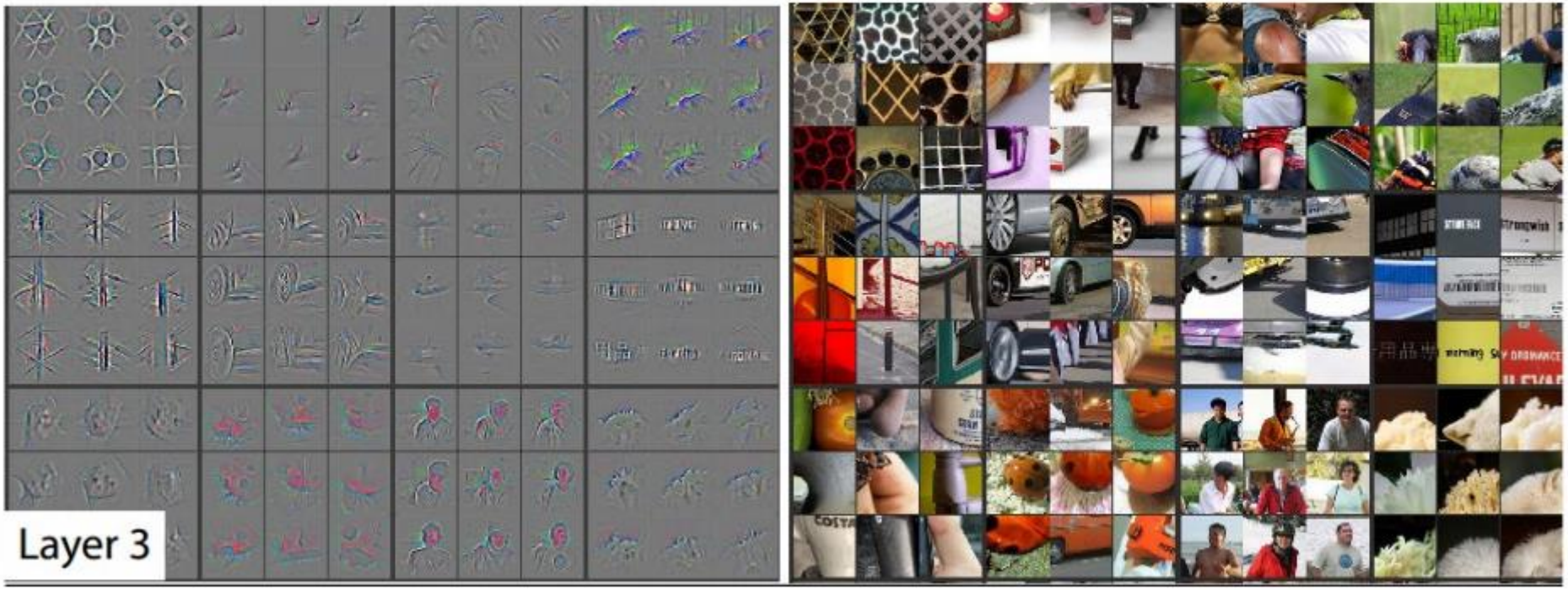
Visualizing and Understanding Convolutional Networks
Zeiler & Fergus, 2013

Visualizing arbitrary neurons along the way to the top...



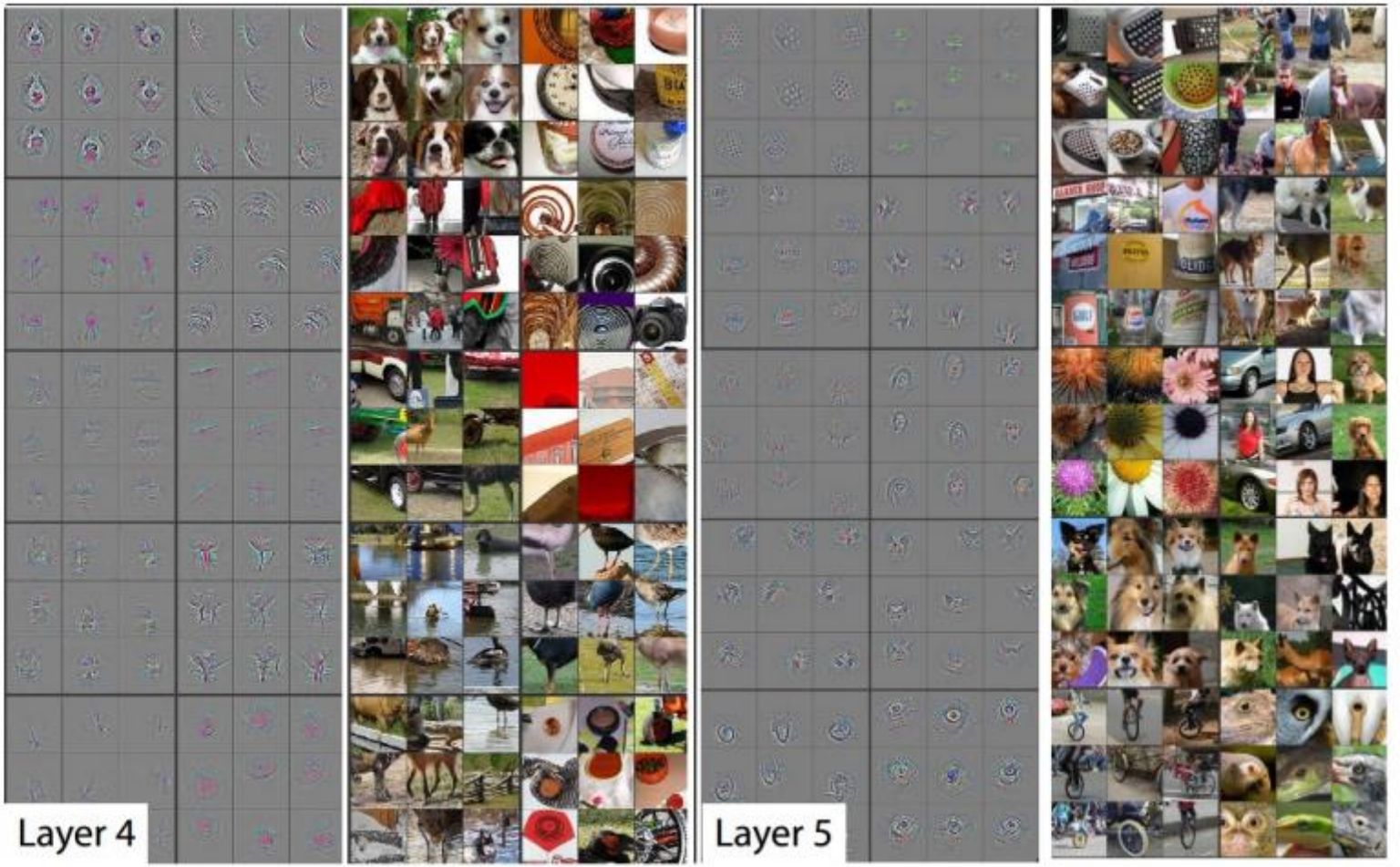
Visualize CNNs

Visualizing arbitrary neurons along the way to the top...

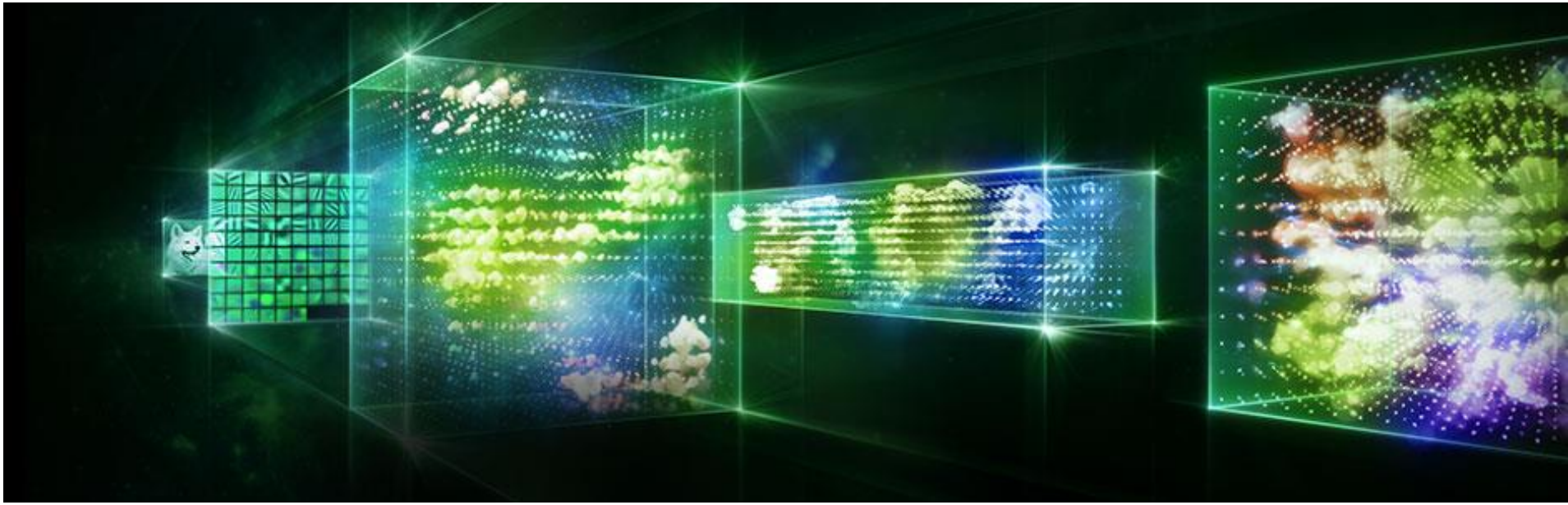


Visualize CNNs

Visualizing arbitrary neurons along the way to the top...



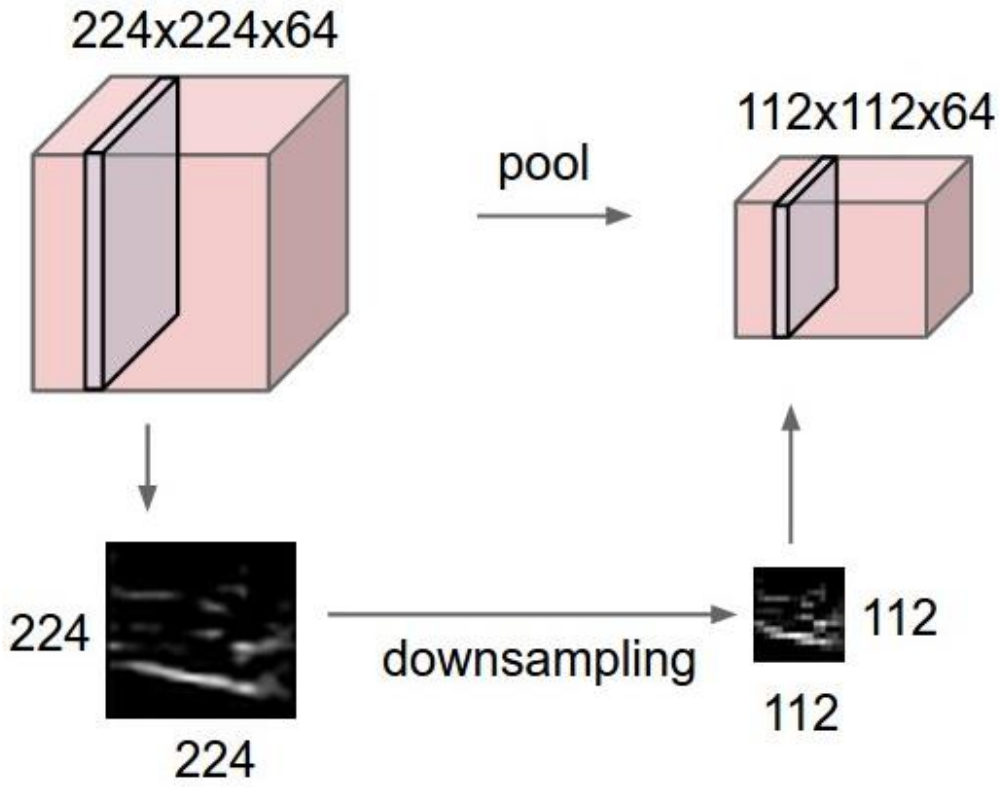
Visualize CNNs



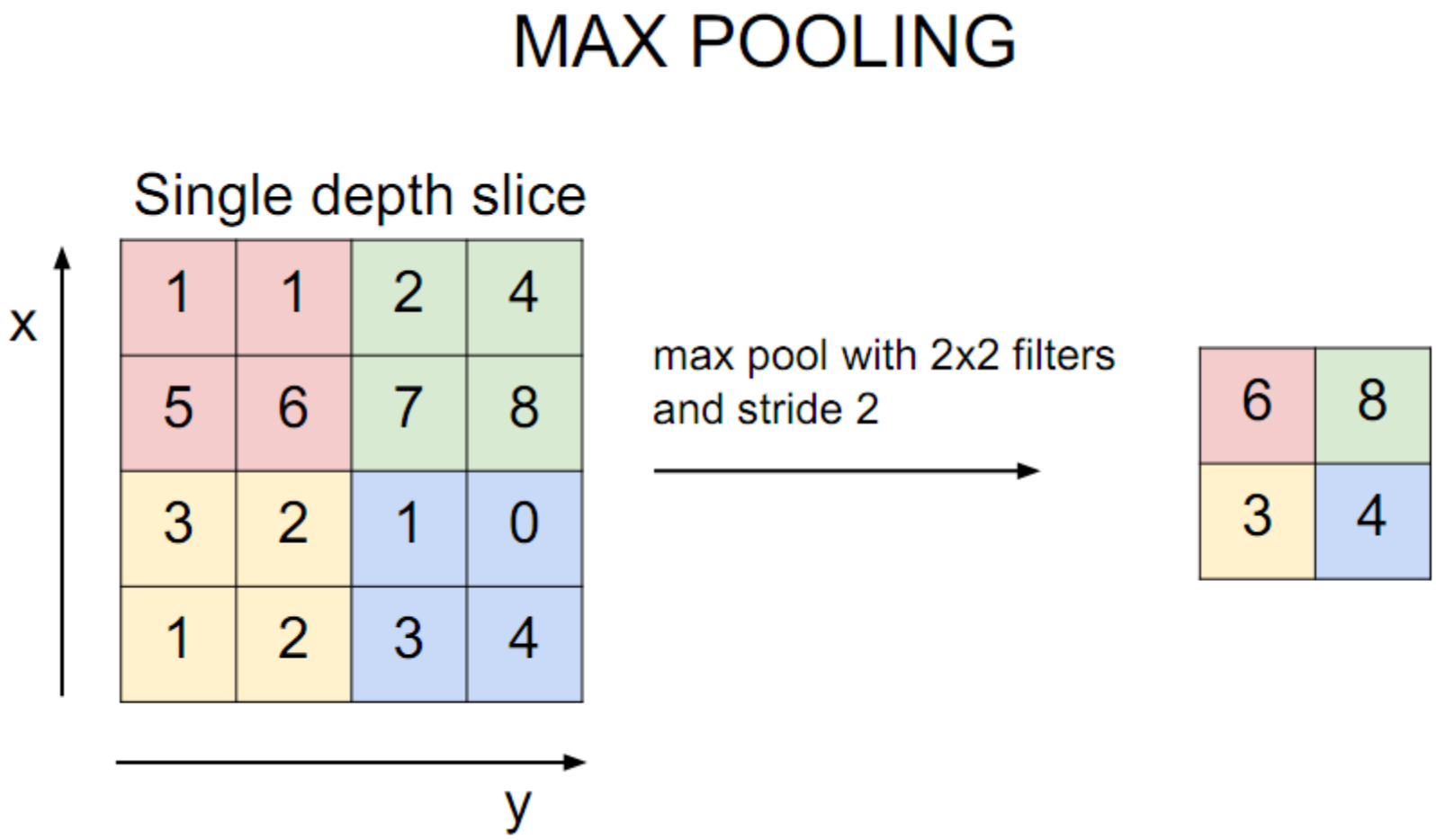
Pooling

In ConvNet architectures, **Conv** layers are often followed by **Pool** layers

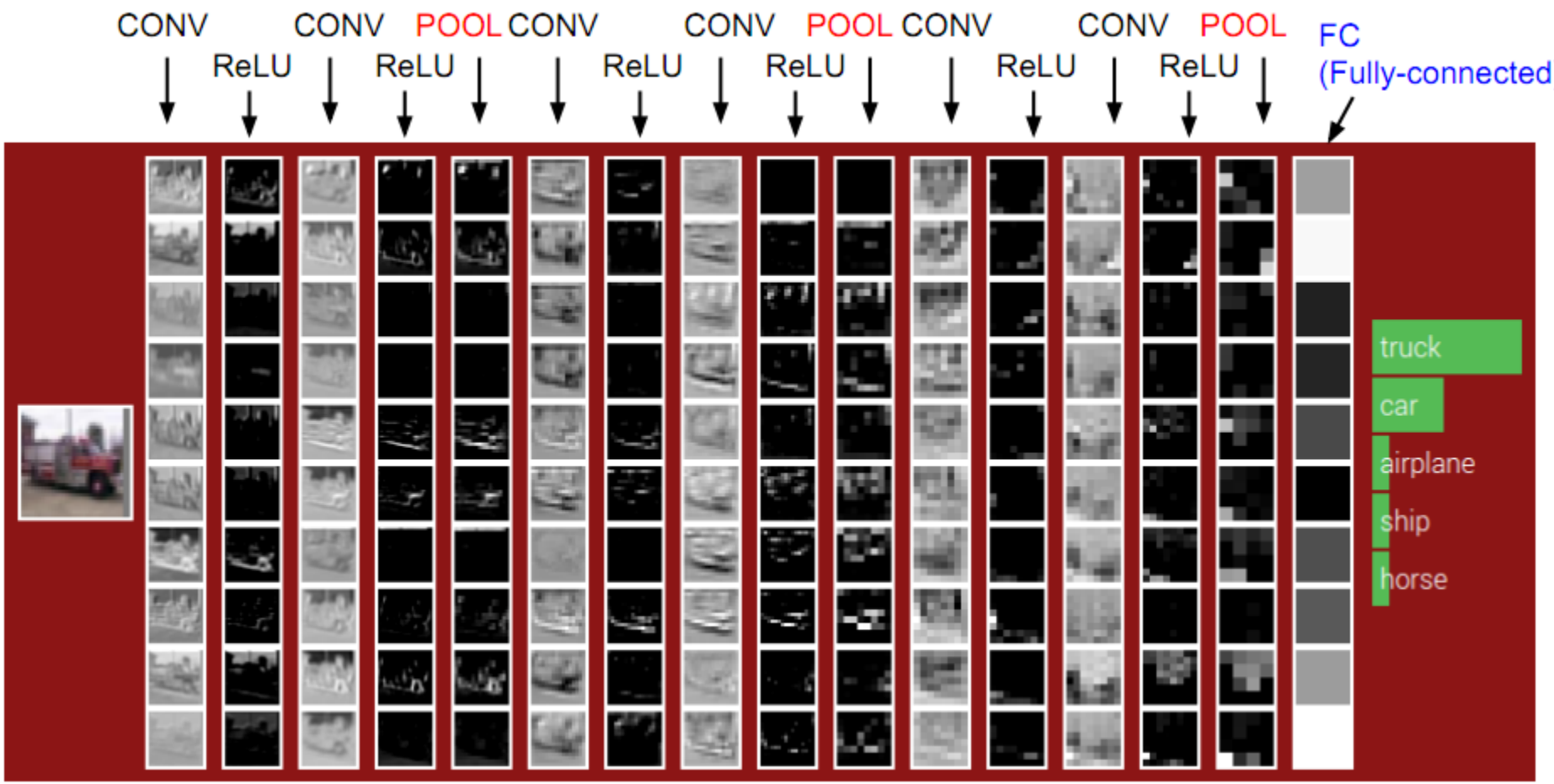
- convenience layer: makes the representations smaller and more manageable without losing too much information. Computes MAX operation (most common)



Pooling


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

ConvNets


CONV
ReLU
CONV
ReLU
POOL
CONV
ReLU
CONV
ReLU
POOL
CONV
ReLU
CONV
ReLU
POOL
FC
(Fully-connected)
truck
car
airplane
ship
horse

ConvNets

Modern CNNs:

- use **filter sizes of 3x3** (maybe even 2x2 or 1x1!)
- use **pooling sizes of 2x2** (maybe even less - e.g. fractional pooling!)
- **stride 1**
- **very deep**

INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC
where the * indicates repetition, and the POOL? indicates an optional pooling layer.
N >= 0 (and usually N <= 3), M >= 0, K >= 0 (and usually K < 3).

ConvNets

Case study: VGGNet / OxfordNet
(runner-up winner of ILSVRC 2014)
[Simonyan and Zisserman]

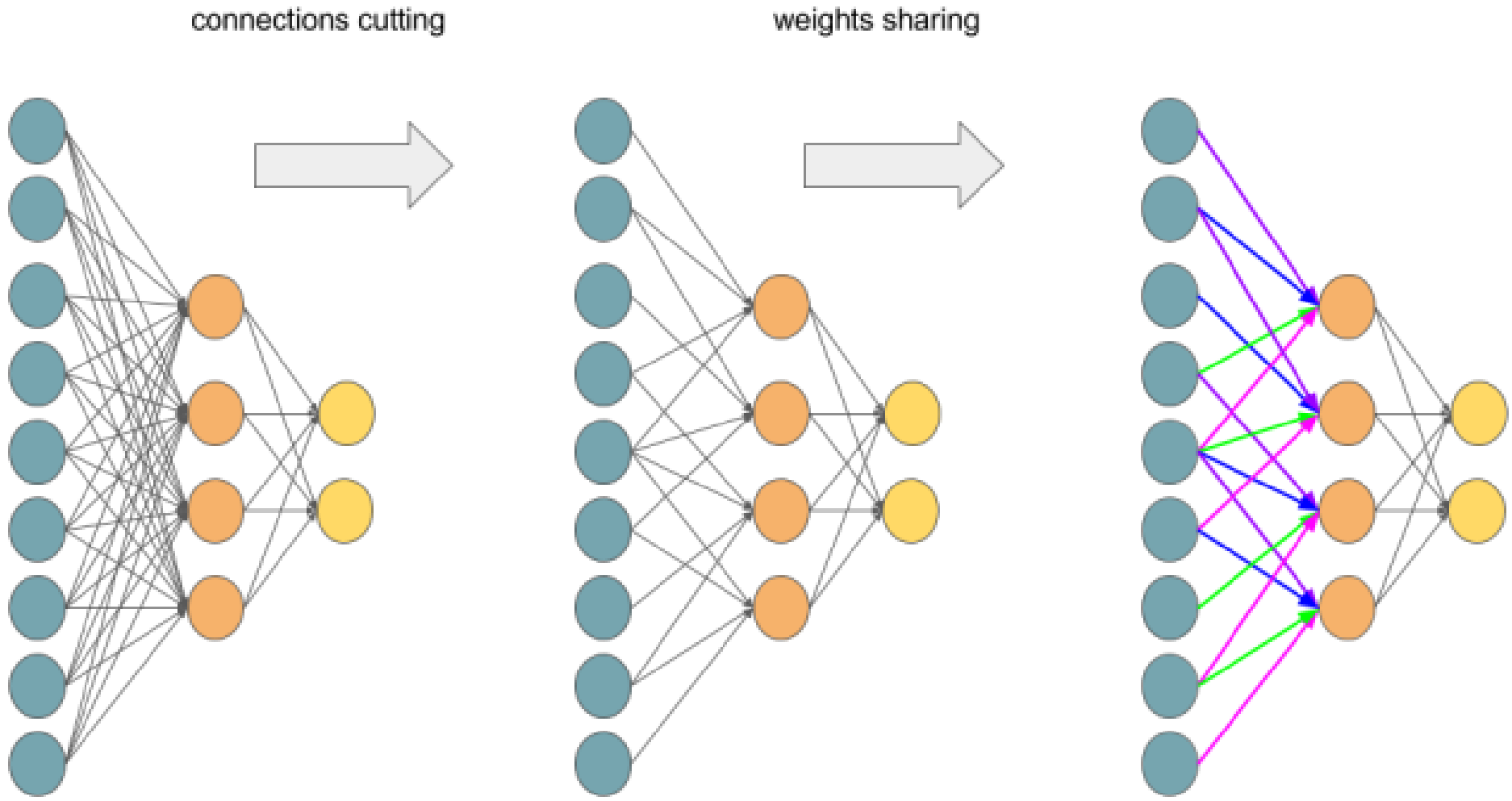
best model

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

BackPropagation in CNNs



BackPropagation in CNNs

Error signals for each example are computed by upsampling. Upsampling is an operation which backpropagates (distributes) the error signals over the aggregate function g using its derivatives $g'_n = \partial g / \partial x_{(n-1)m+1:nm}$. g'_n can change depending on pooling region n.

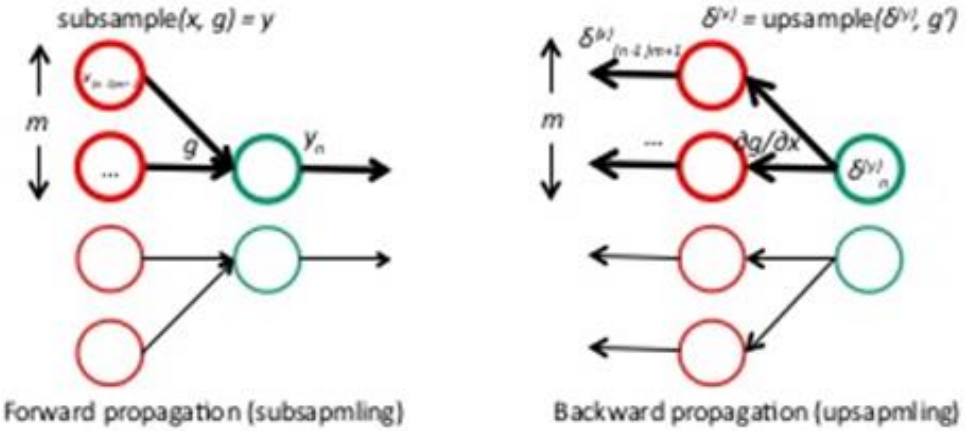
- In max pooling, the unit which was the max at forward propagation receives all the error at backward propagation and the unit is different depending on the region n.

- Definition: *upsample*

upsample(f, g)[n] denotes the n-th element of upsample(f, g).

$$\delta^{(x)}_{(n-1)m+1:nm} = \text{upsample}\left(\delta^{(y)}, g'\right)[n] = \delta^{(y)}_n g'_n = \delta^{(y)}_n \frac{\partial g}{\partial x_{(n-1)m+1:nm}} = \frac{\partial J}{\partial y_n} \frac{\partial y_n}{\partial x_{(n-1)m+1:nm}} = \frac{\partial J}{\partial x_{(n-1)m+1:nm}}$$

$$\delta^{(x)} = \text{upsample}\left(\delta^{(y)}, g'\right) = \left[\delta^{(x)}_{(n-1)m+1:nm}\right]$$



References

1. CNNs for Visual Recognition by Stanford
http://cs231n.stanford.edu/slides/winter1516_lecture7.pdf

2. Deep Learning Course by Udacity
https://classroom.udacity.com/courses/ud730/lessons/6377263405/concepts/63741833610923#

Thank you!