

Convolutional Neural Networks

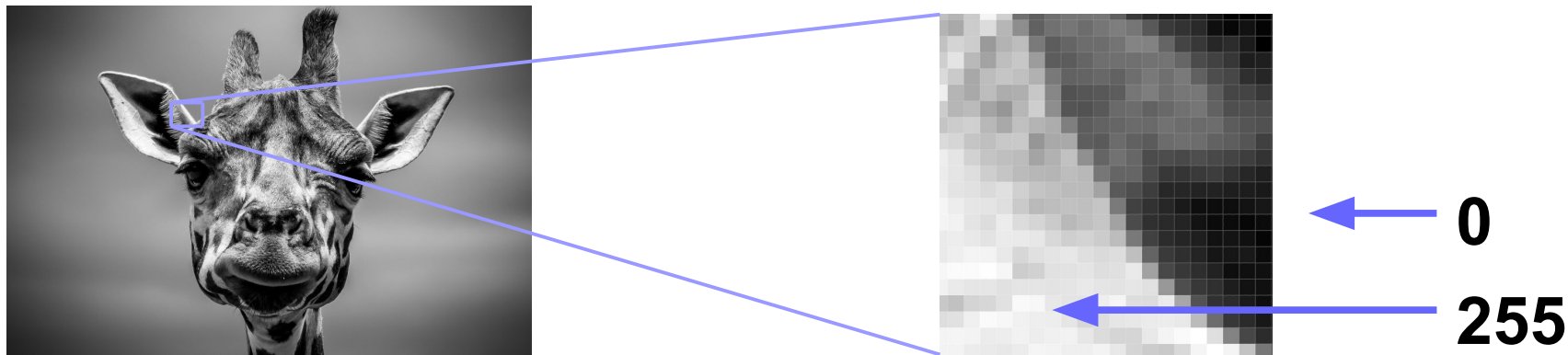
mlhep 2019

LAMBDA 



Images

- Grayscale image is a matrix of pixels [H x W]
 - Pixels = **p**icture **e**lements
- Each pixel stores number [0,255] for brightness



Images

- RGB image is a 3d array $[H \times W \times 3]$ or $[3 \times H \times W]$
 - Each pixel stores **R**ed, **G**reen & **B**lue color values $[0, 255]$

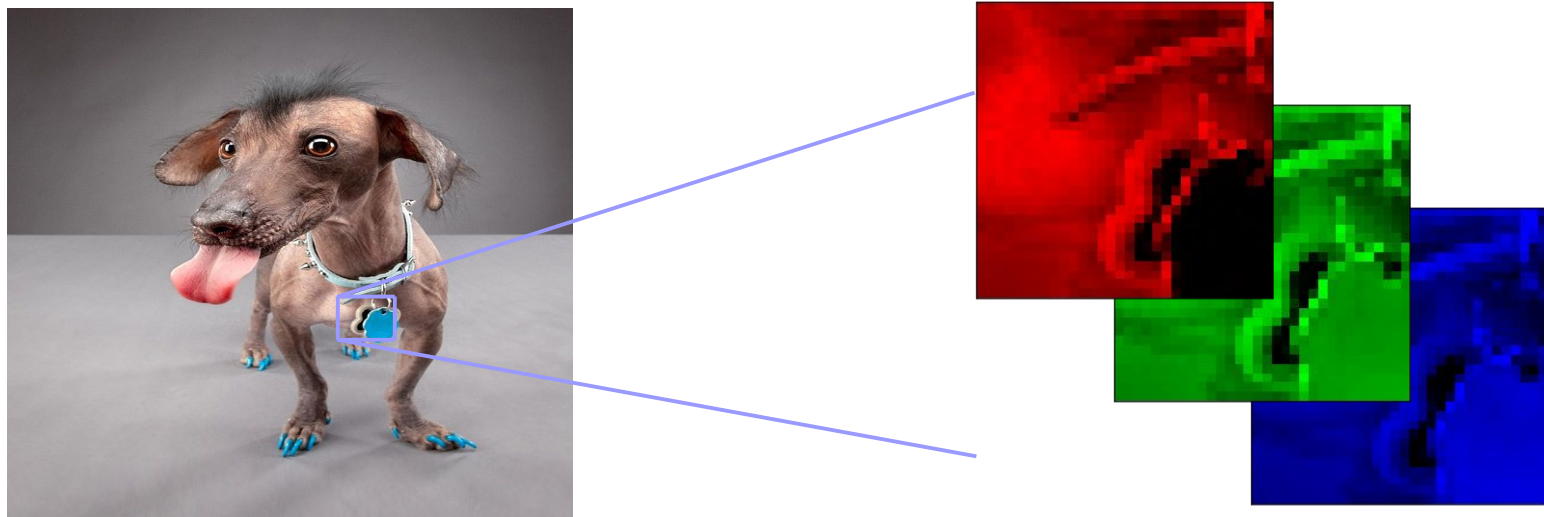
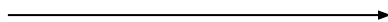


Image recognition



“Dog”

Image recognition



“Gray wall”

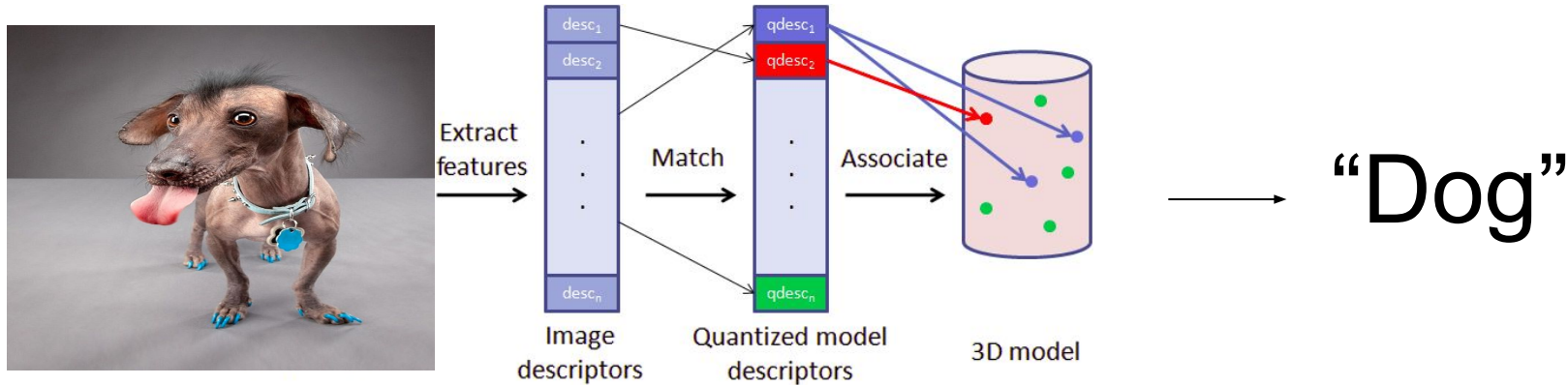
“Dog tongue”

“Dog”

<a particular kind of
dog>

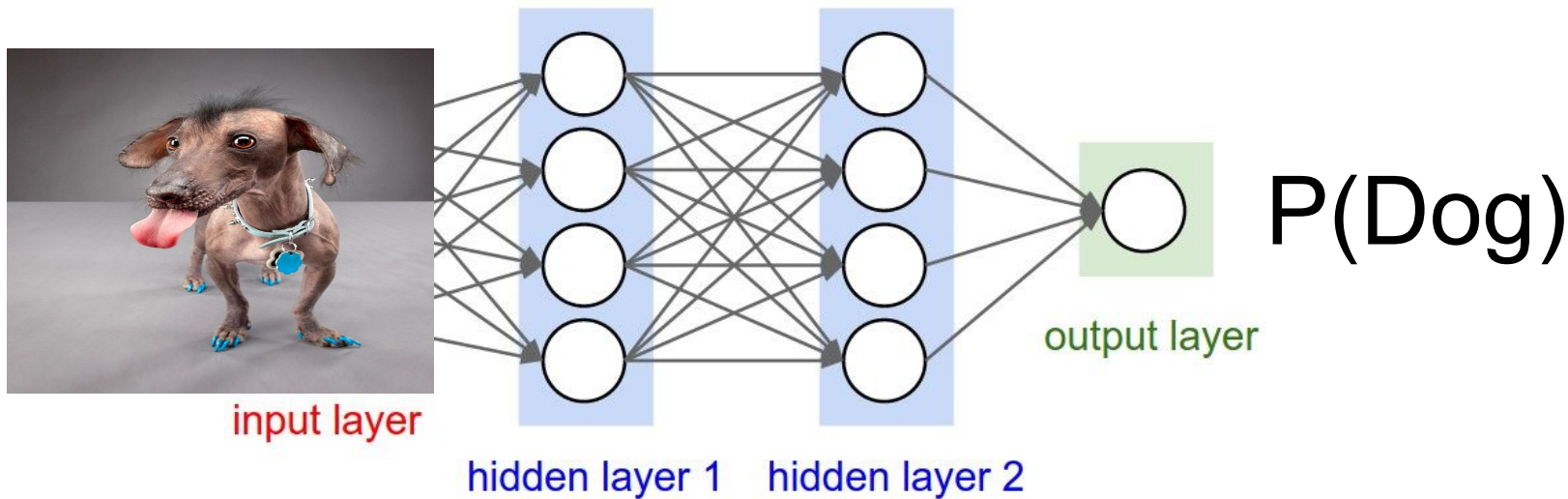
“Animal sadism”

Classical approach

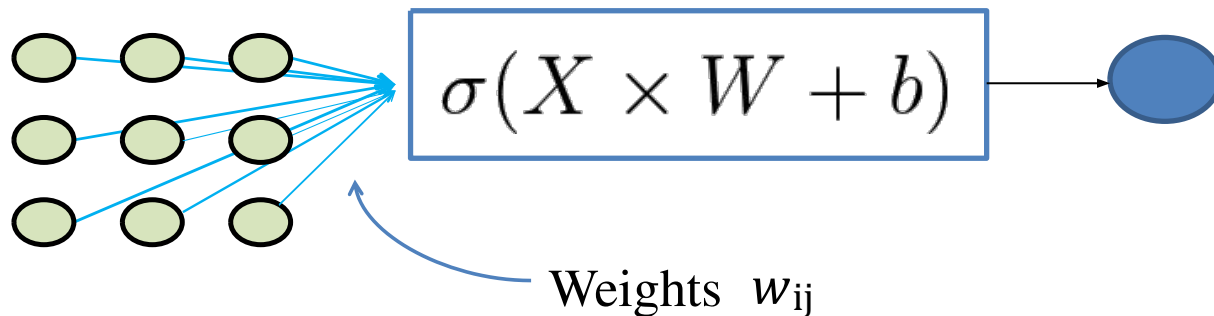


Guess what we're going to do now?

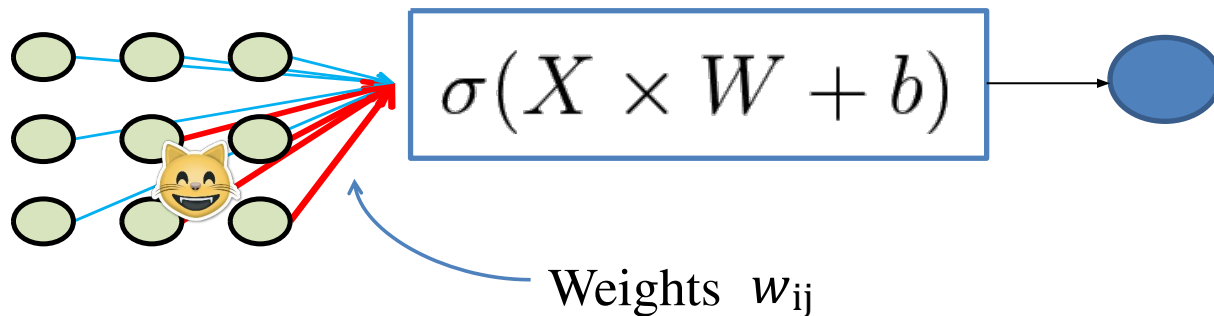
NN approach



Problem with images

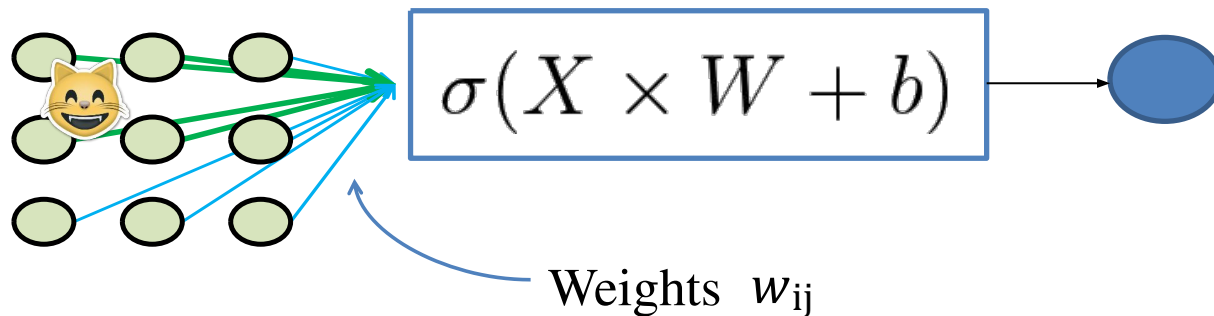


Problem with images



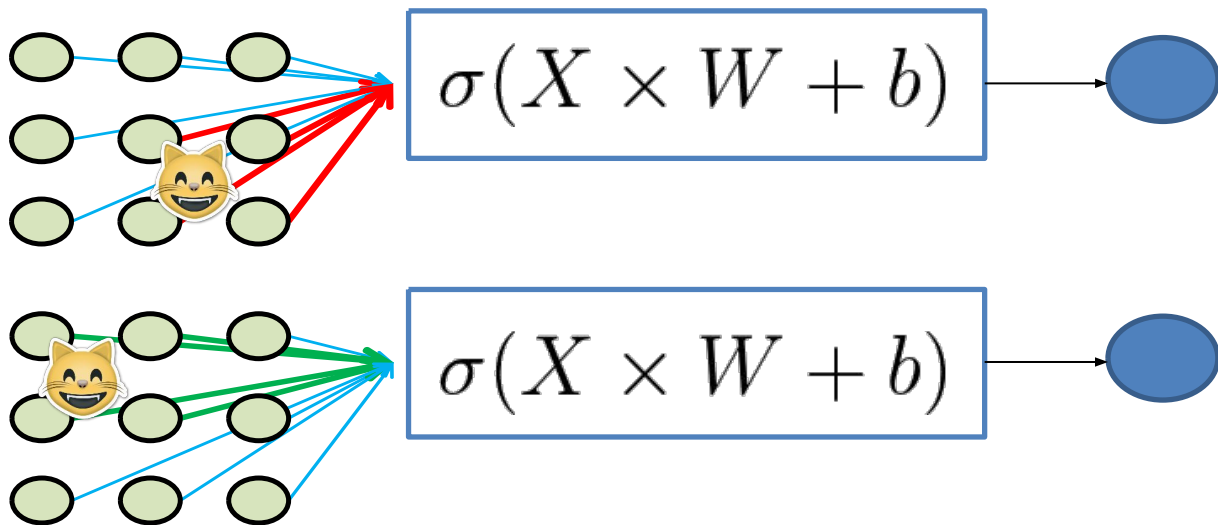
On this object, you will train **red**
weights to react on cat face

Problem with images



On this object, you will train **green** weights to react on cat face

Problem with images



You network will have to learn those two cases separately!

Worst case: one neuron per position.

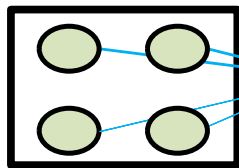
Slide from Andrey Zimovnov's "Deep learning for computer vision"

Problem

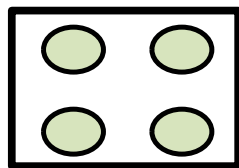
Idea: force all these “cat face” features to use **exactly the same weights**, shifting weight matrix each time.

Same features for each spot

Portable cat detector pro!



Weights w_{ij}

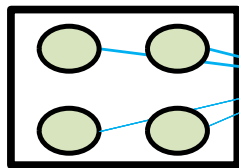


No cat

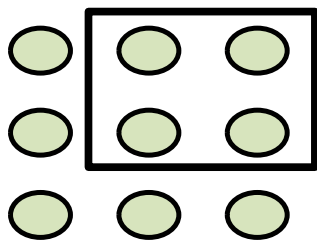


Same features for each spot

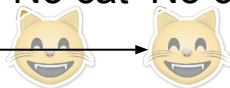
Portable cat detector pro!



Weights w_{ij}

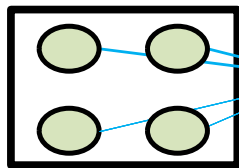


No cat No cat

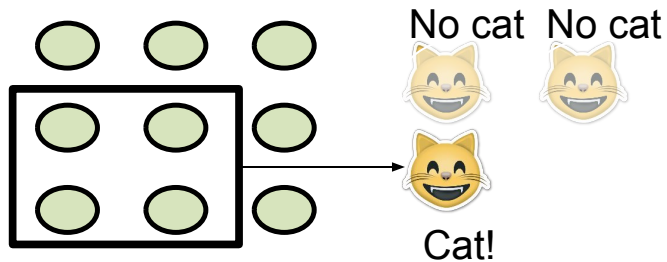


Same features for each spot

Portable cat detector pro!

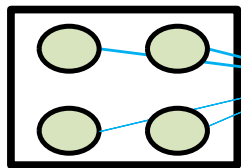


Weights w_{ij}



Same features for each spot

Portable cat detector pro!



Weights w_{ij}



No cat No cat

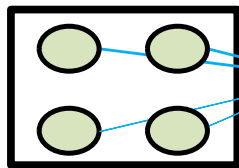


Cat!

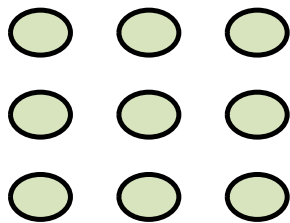
No cat

Same features for each spot

Portable cat detector pro!



Weights w_{ij}



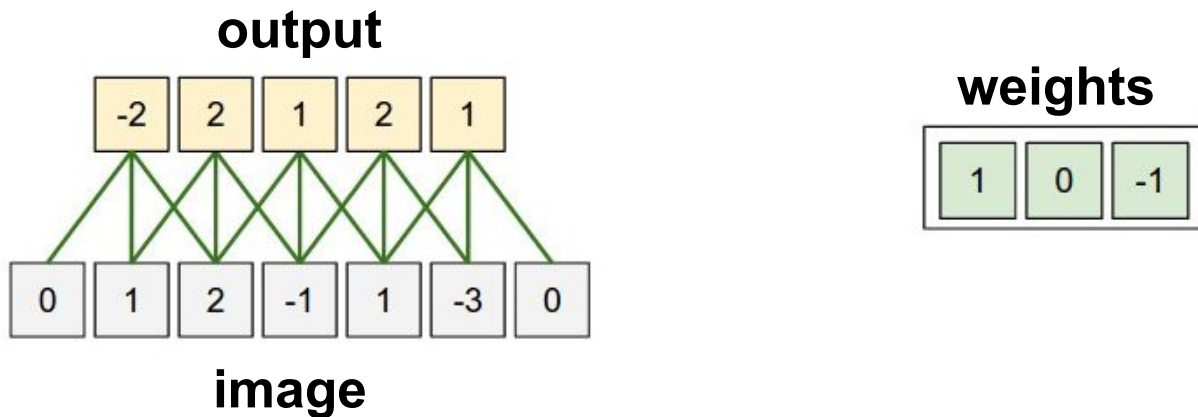
No cat No cat
 
 
Cat! No cat



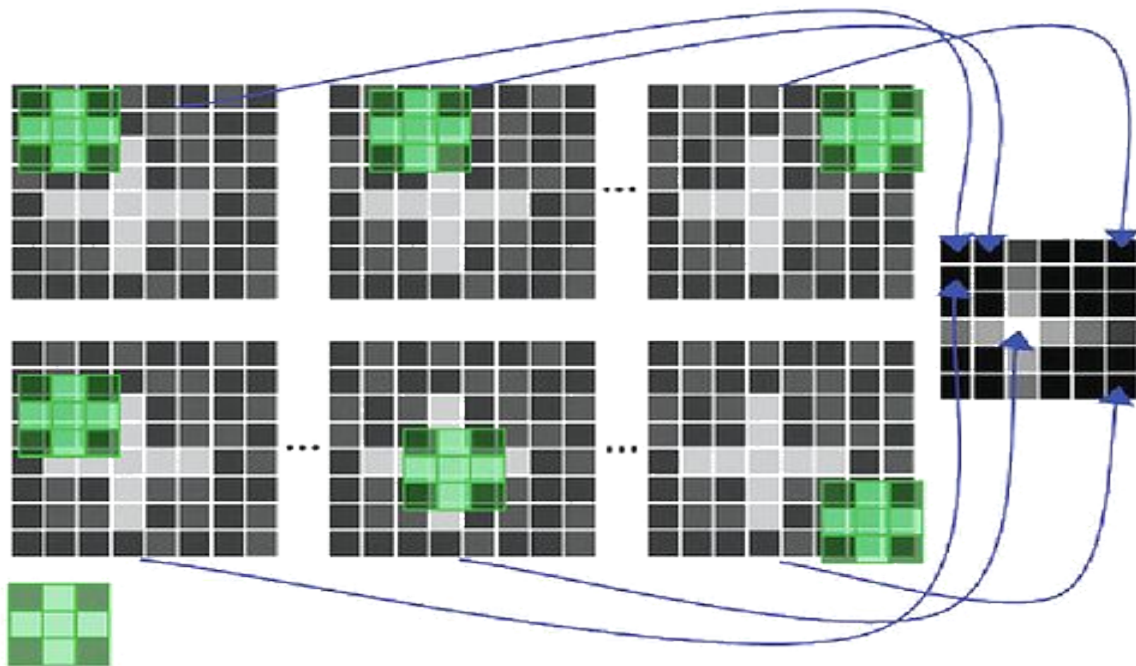
There's a cat here!

Convolution

- Apply same weights to all patches



Convolution



apply one “filter” to all patches

Convolution

5x5

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

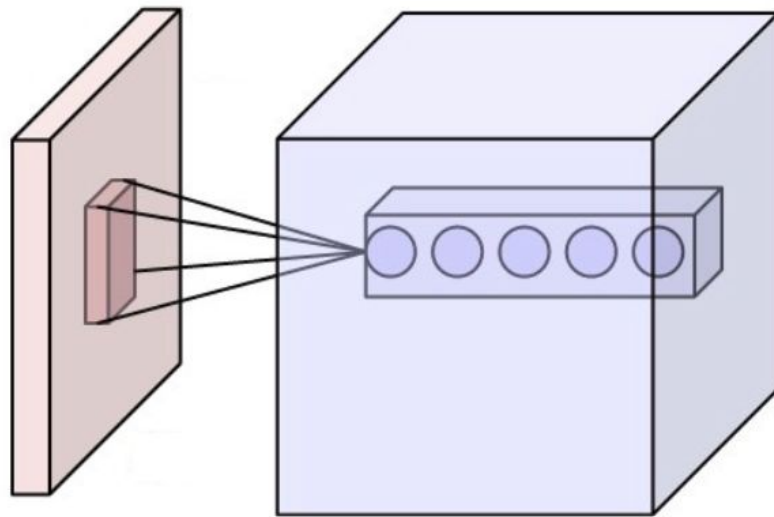
3x3 (5-3+1)

4		

Convolved
Feature

Intuition: how cat-like is this square?

Convolution



Intuition: how cat-like is this square?

Convolution

Input image



Convolution
Kernel

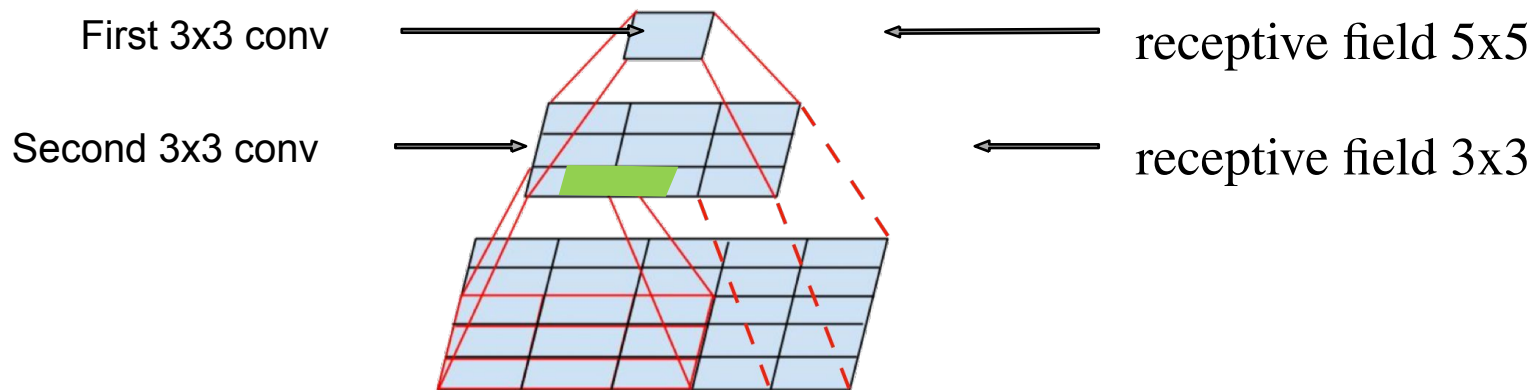
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map



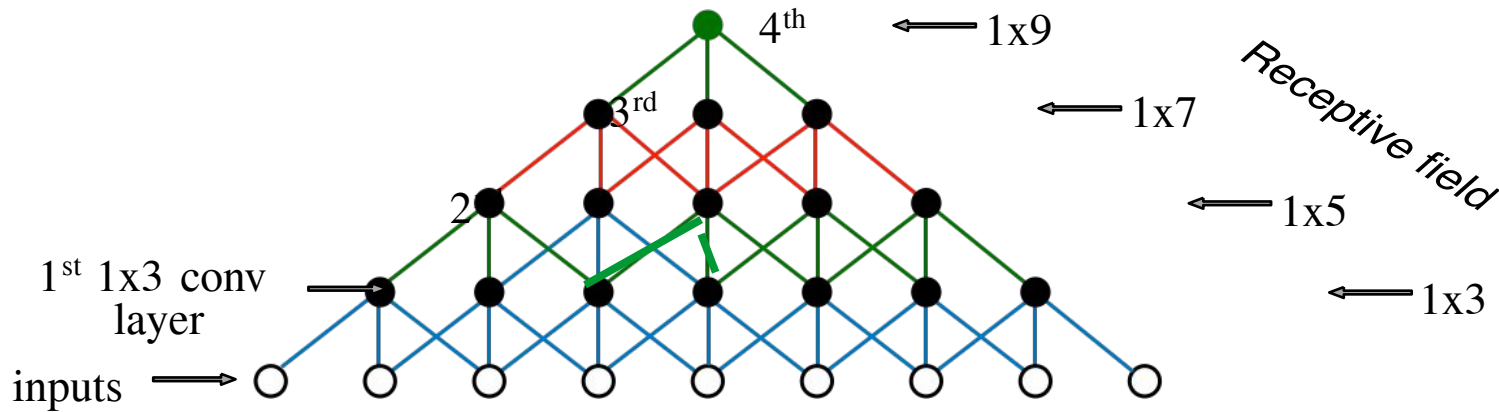
Intuition: how **edge-like** is this square?

Receptive field



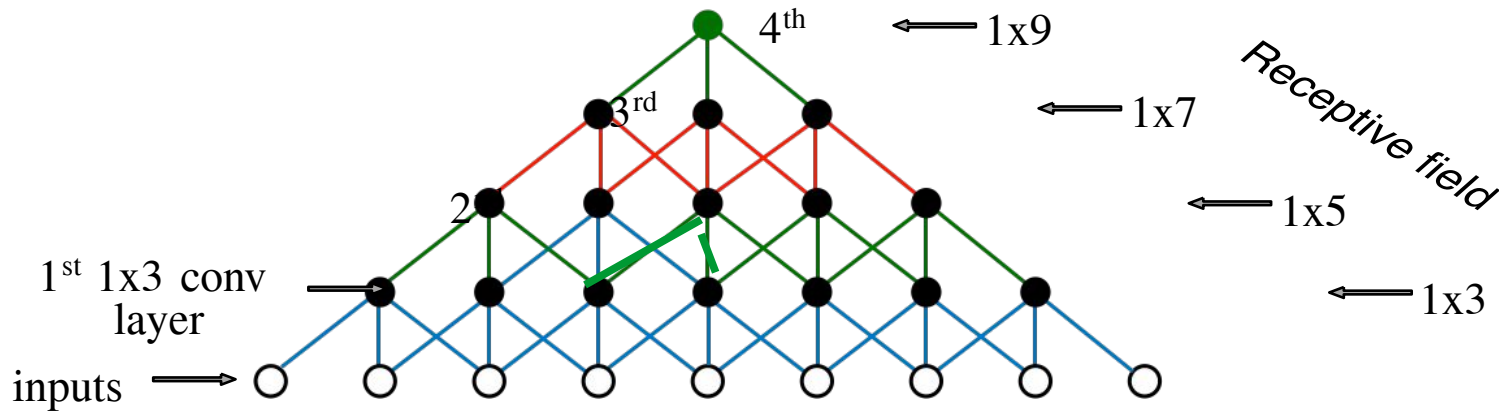
We can recognize larger objects by stacking several small convolutions!

Receptive field



Q: how many 3x3 convolutions we should use
to recognize a 100x100px cat

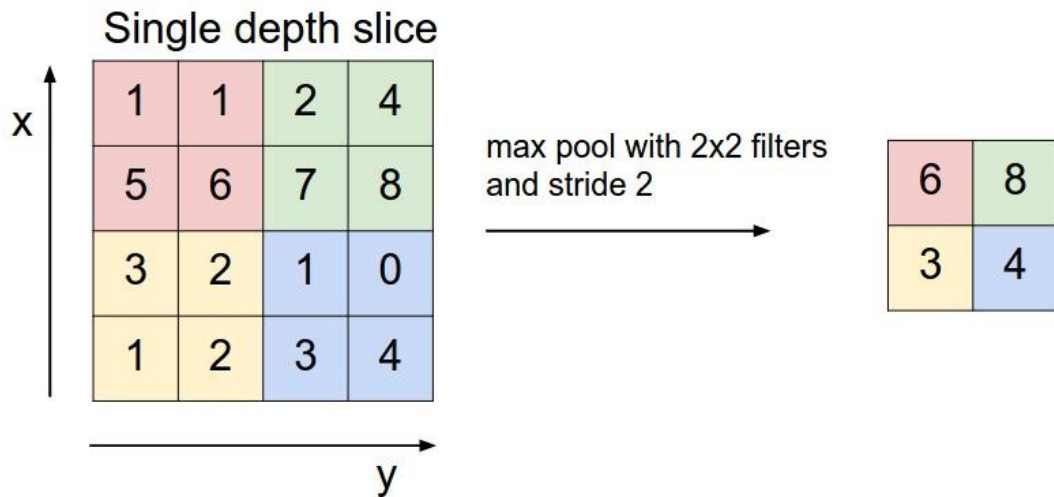
Receptive field



Q: how many 3x3 convolutions we should use
to recognize a 100x100px cat

A: around 50... we need to increase receptive field faster!

Pooling



Intuition: What is the highest catness over this area?

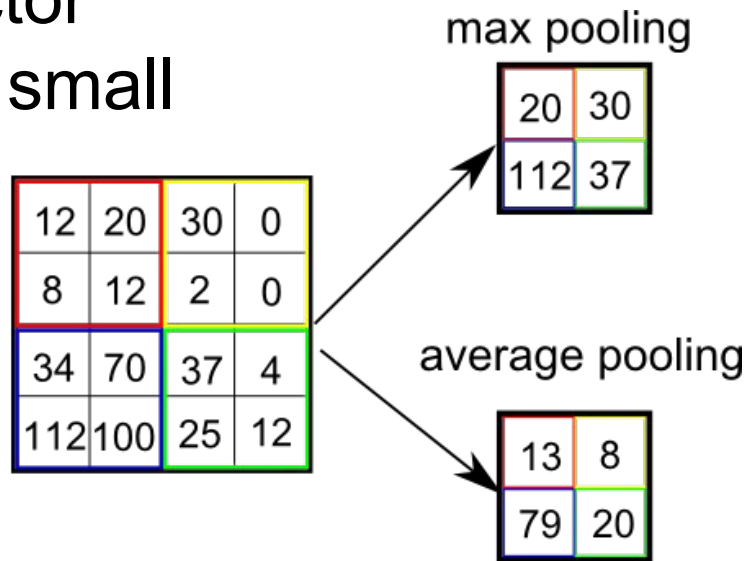
Pooling

Motivation:

- Reduce layer size by a factor
- Make NN less sensitive to small image shifts

Popular types:

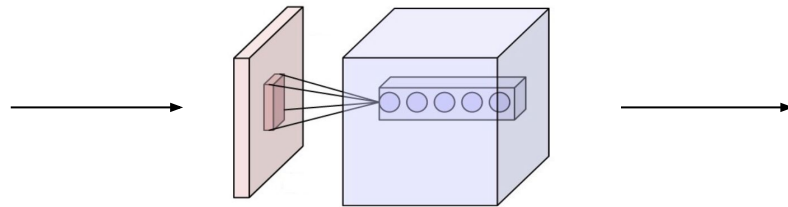
- Max
- Mean(average)



Convolution



Image : 3 (RGB) x 100 px x 100 px

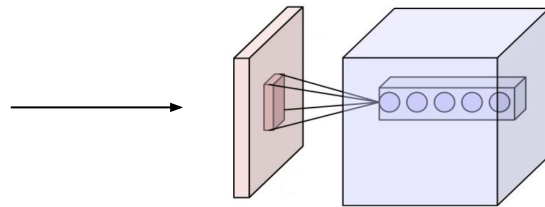


Filters: 100x(3x5x5)

Convolution



Image : 3 (RGB) x 100 px x 100 px



Filters: 100x(3x5x5)

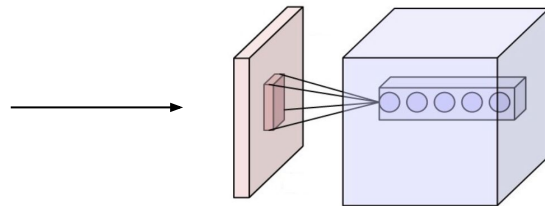
100x96x96
 $\sim 10^6$

Somewhat too many!

Convolution



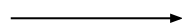
Image : 3 (RGB) x 100 px x 100 px



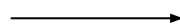
Filters: 100x(3x5x5)

→ 100x96x96
~10⁶

100x96x96



pool
3x4

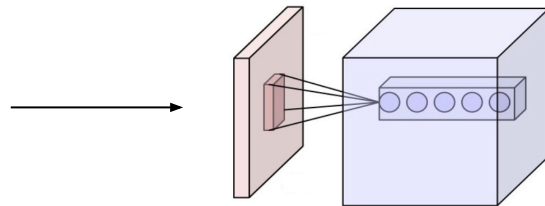


???

Convolution



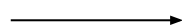
Image : 3 (RGB) x 100 px x 100 px



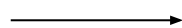
Filters: 100x(3x5x5)

100x96x96
~10⁶

100x96x96



pool
3x4



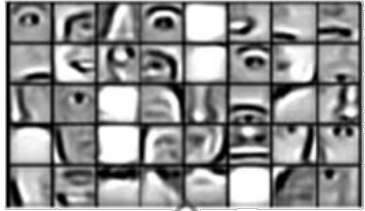
100x32x32

~10⁵

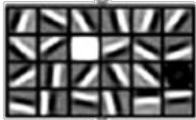


Discrete Choices

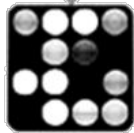
⋮



Layer 2 Features

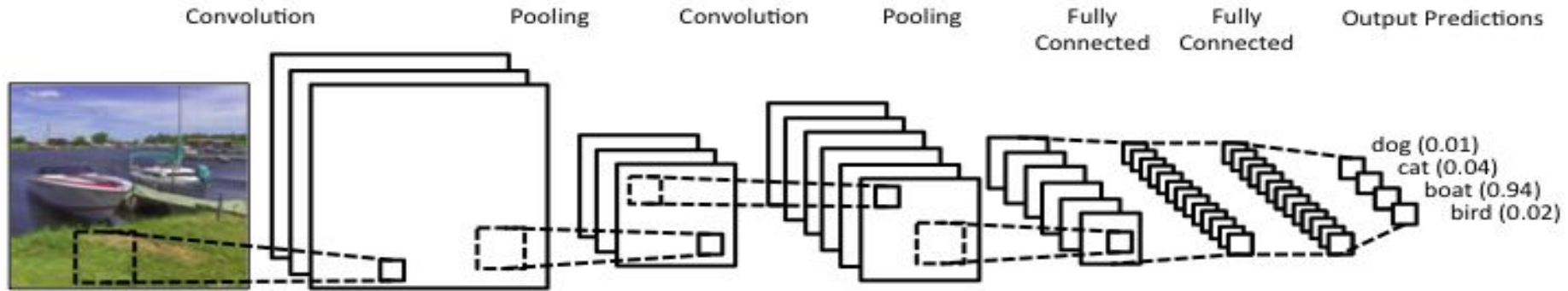


Layer 1 Features

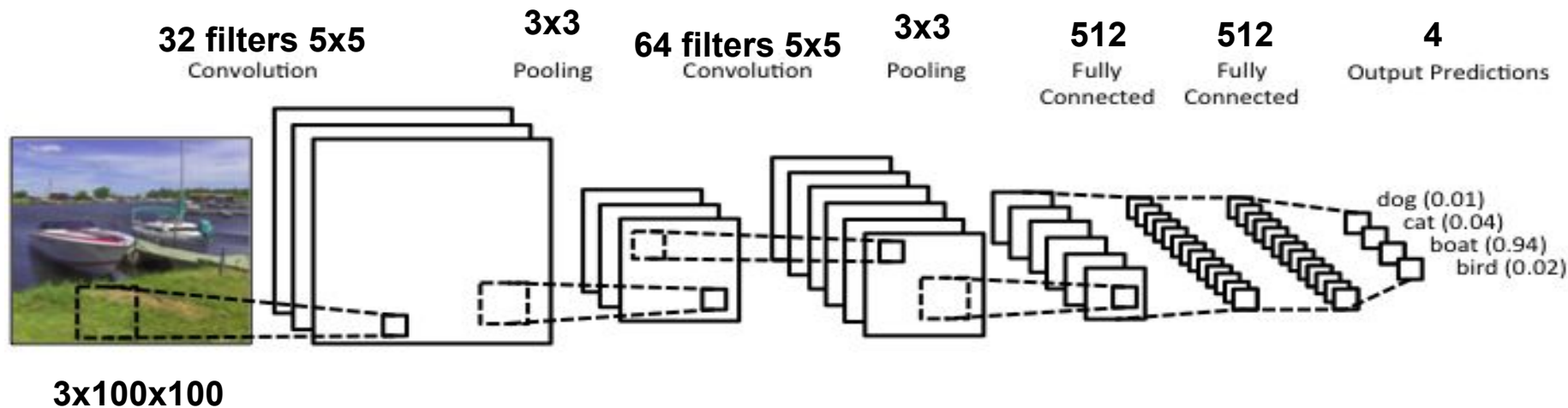


Original Data

Convolutional NNs



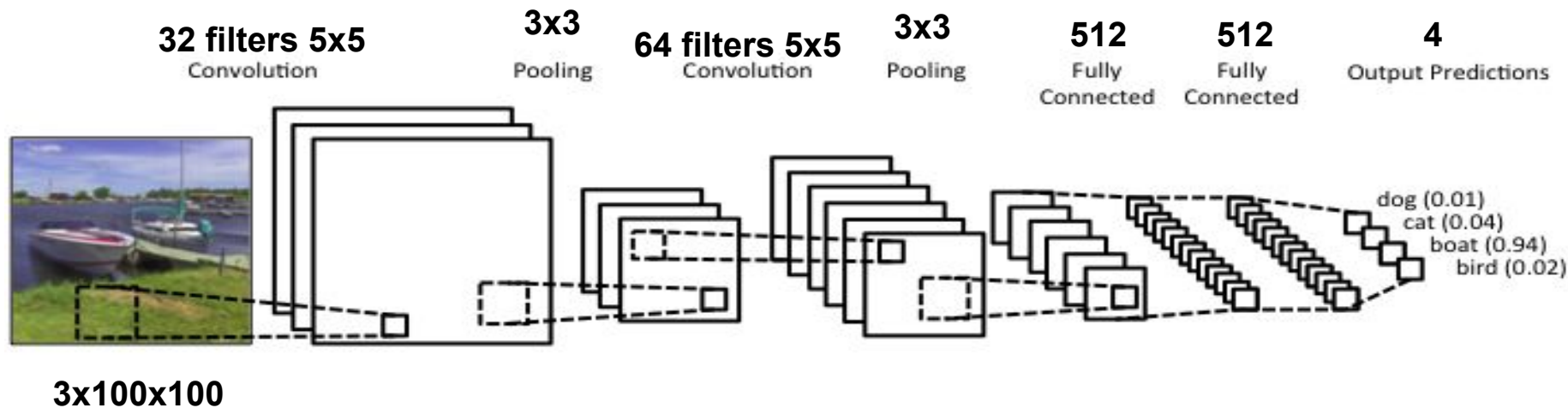
Convolutional NNs



Quiz:

1) What is the output shape **after second pooling**

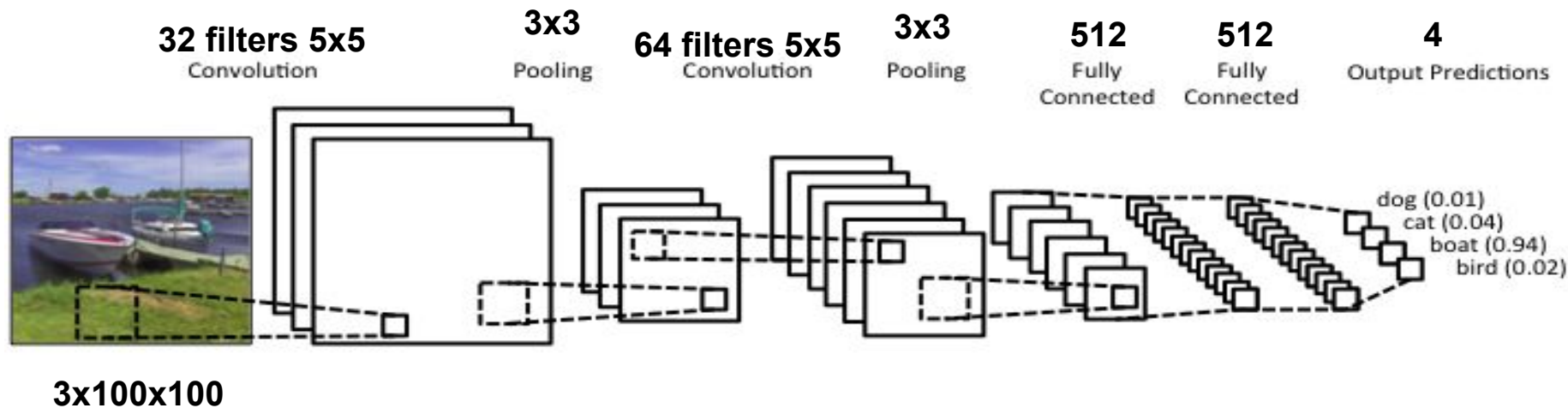
Convolutional NNs



Quiz:

2) How many image pixels does **one cell** after **second convolution** depend on?

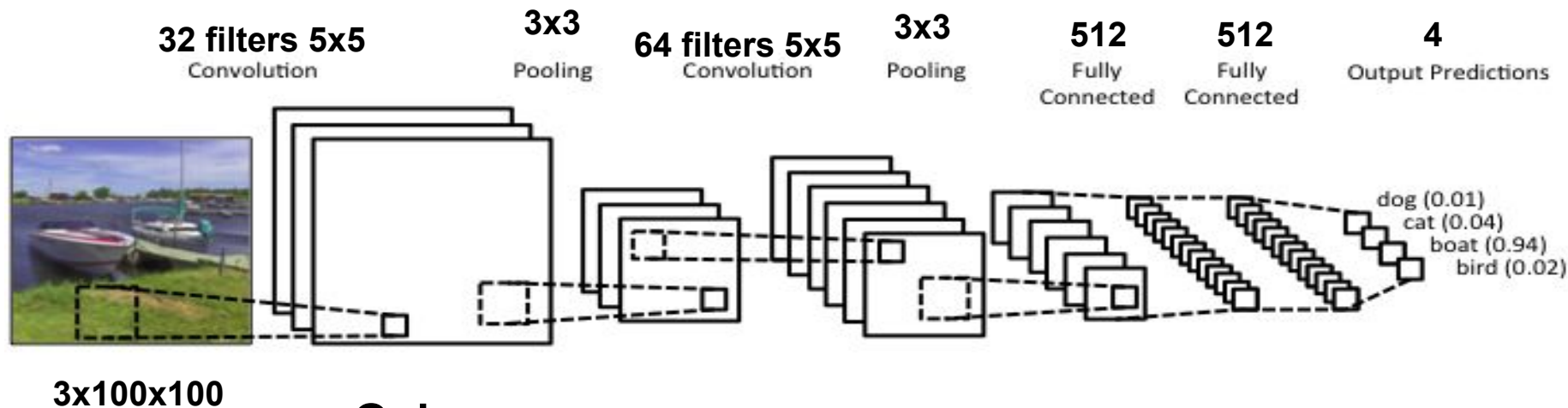
Convolutional NNs



Quiz:

- 3) Which layer is hardest to compute?
- 4) Which layer has most independent parameters?

Convolutional NNs



Quiz:

- 3) Which layer is hardest to compute?: **first conv**
- 4) Which layer has most independent parameters?

first dense

Problem with large networks

What you sign for if you stack 1000 layers:

- `MemoryError(0x...)`
- Gradients can vanish
- Gradients can explode
- Activations can vanish
- Activations can explode

Problem with large networks

What you sign for if you stack 1000 layers:

- `MemoryError(0x...)`
- Gradients can vanish
- Gradients can explode
- **Activations can vanish**
- **Activations can explode**

How do we fix these?

Problem with large networks

What you sign for if you stack 1000 layers:

- `MemoryError(0x...)`
- Gradients can vanish
- Gradients can explode
- **Activations can vanish**
- **Activations can explode**



Batch normalization
or similar

Problem with large networks

What you sign for if you stack 1000 layers:

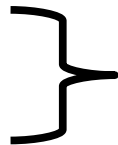
- `MemoryError(0x...)`

- **Gradients can vanish**

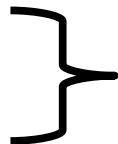
- **Gradients can explode**

- Activations can vanish

- Activations can explode



In a few days



Batch normalization
or similar

Data augmentation



- Idea: we can get N times more data by tweaking images.
- Rotate, crop, zoom, flip horizontally, add noise
- If you rotate cat image by 15° , it's still a cat
- Sound data: apply background noises

Other CV applications

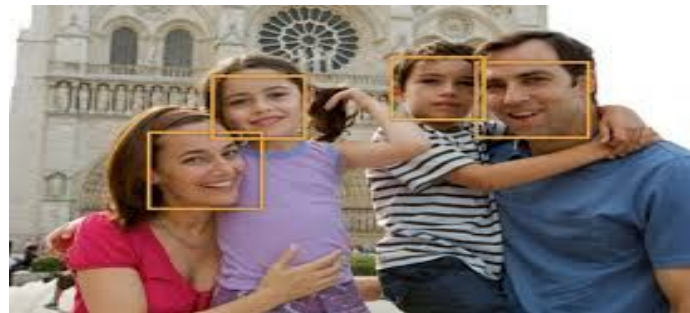
Real computer vision starts when
image classification is no longer enough.

Bounding box regression

Predict object bounding box

(x_0, y_0, w, h)

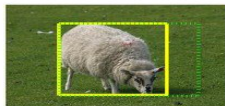
or several bounding boxes for multiple objects.



Applications examples:

- Face detection @ cameras
- Surveillance cameras
- Self-driving cars

IM:"005194" Conf=0.835223



IM:"003538" Conf=0.829488



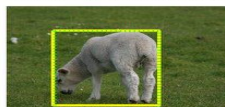
IM:"002810" Conf=0.801748



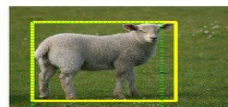
IM:"004522" Conf=0.799045



IM:"001064" Conf=0.797061



IM:"000819" Conf=0.794456



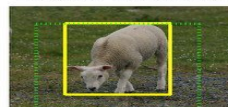
IM:"002306" Conf=0.789123



IM:"001956" Conf=0.788438



IM:"004285" Conf=0.782058



Segmentation

Predict class for each pixel

(fully-convolutional networks)

Applications examples:

- Moar surveillance
- Brain scan labeling
- Map labeling

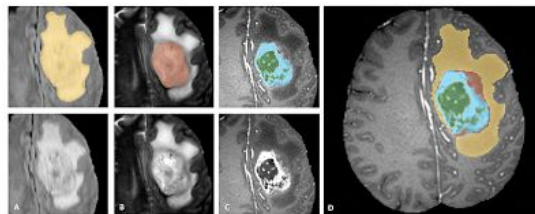
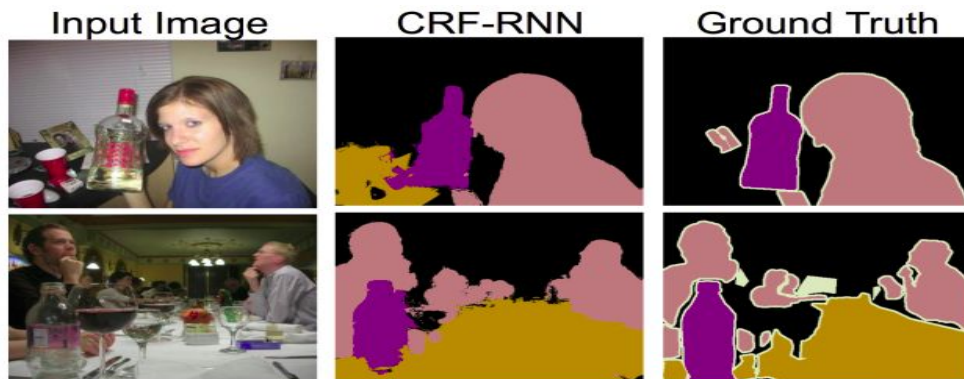
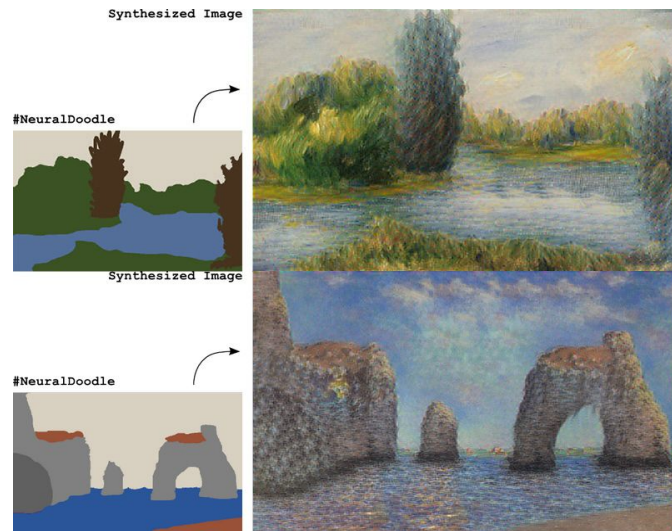


Image generation/transformation

- **Generation:** Given a set of reference images, learn to generate new images, resembling those you were given.
- **Transformation:** Given a set of reference images, learn to convert other images into ones resembling the reference set.



Neural Doodle
(D. Ulyanov et al.)

Image tagging
Image captioning
Image retrieval
Image encoding
Image morphing
Image encoding
Image upscaling
Object tracking on video
Video processing
Video interpolation
3D point clouds

Fine-tuning
Adversarial Networks
Variational Autoencoders
Knowledge transfer
Domain adaptation
Online learning
Explaining predictions
Soft targets
Scene reconstruction
3D object retrieval
Classifier optimization

Nuff

Let's train some CNNs!

