CSC 411 Lecture 11: Neural Networks II

Roger Grosse, Amir-massoud Farahmand, and Juan Carrasquilla

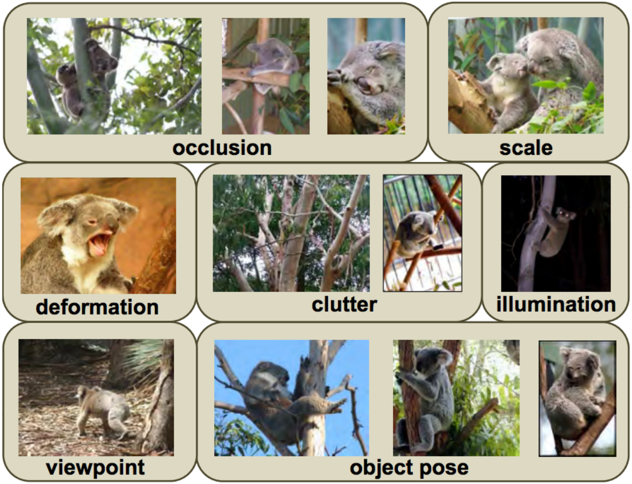
University of Toronto

Neural Nets for Visual Object Recognition

- People are very good at recognizing shapes
  - Intrinsically difficult, computers are bad at it
- Why is it difficult?

Why is it a Problem?

- Difficult scene conditions



[From: Grauman & Leibe]

Why is it a Problem?

- Huge within-class variations. Recognition is mainly about modeling variation.

[Pic from: S. Lazebnik]

Why is it a Problem?

- Tons of classes

[Biederman]

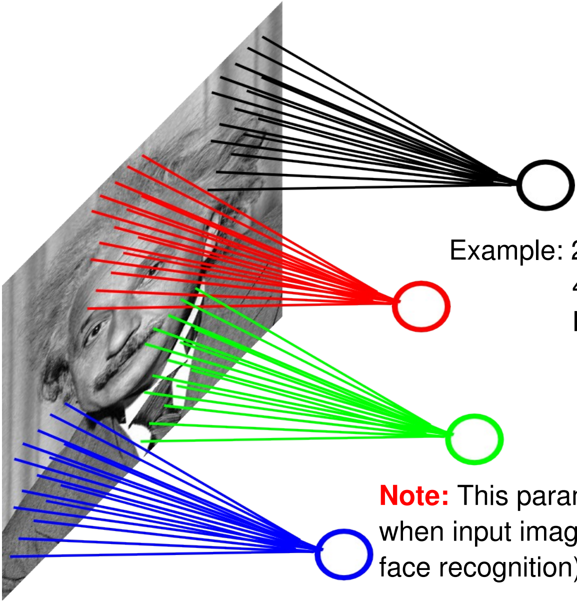
Neural Nets for Object Recognition

- People are very good at recognizing object
  - Intrinsically difficult, computers are bad at it
- Some reasons why it is difficult:
  - Segmentation: Real scenes are cluttered
  - Invariances: We are very good at ignoring all sorts of variations that do not affect class
  - Deformations: Natural object classes allow variations (faces, letters, chairs)
  - A huge amount of computation is required

How to Deal with Large Input Spaces

- How can we apply neural nets to images?
- Images can have millions of pixels, i.e., $\mathbf{x}$ is very high dimensional
- How many parameters do I have?
- Prohibitive to have fully-connected layers
- What can we do?
- We can use a locally connected layer

Locally Connected Layer



Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

34

Ranzato

When Will this Work?

When Will this Work?

- This is good when the input is (roughly) registered

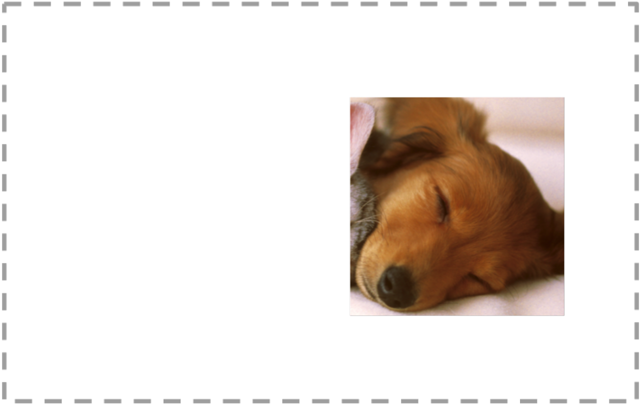
General Images

- The object can be anywhere

[Slide: Y. Zhu]

General Images

- The object can be anywhere



[Slide: Y. Zhu]

General Images

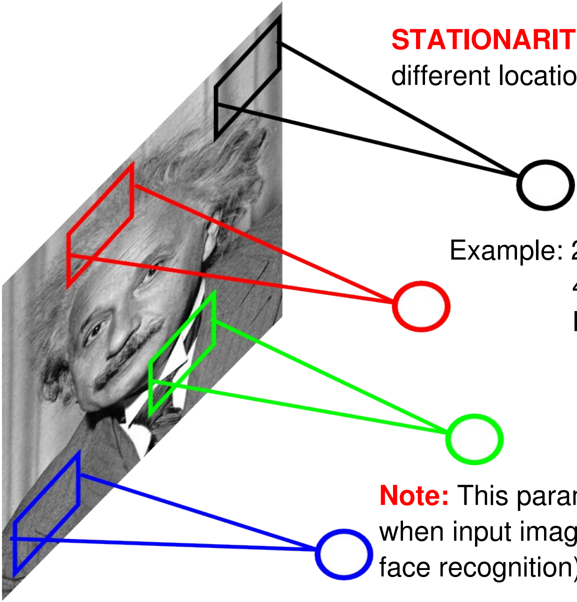
- The object can be anywhere

[Slide: Y. Zhu]

The Invariance Problem

- Our perceptual systems are very good at dealing with invariances
  - translation, rotation, scaling
  - deformation, contrast, lighting
- We are so good at this that its hard to appreciate how difficult it is
  - Its one of the main difficulties in making computers perceive
  - We still don't have generally accepted solutions

Locally Connected Layer



STATIONARITY? Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

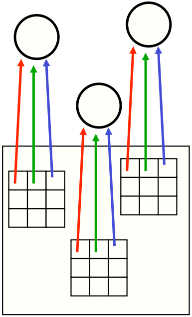
Note: This parameterization is good when input image is registered (e.g., face recognition).

35

Ranzato

The replicated feature approach

The red connections all have the same weight.

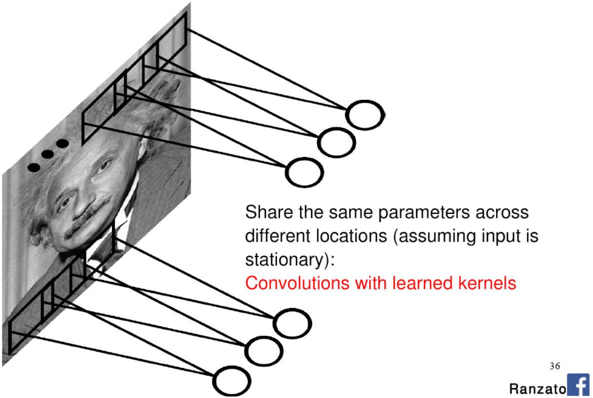


5

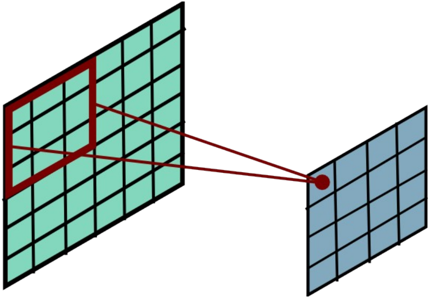
- Adopt approach apparently used in monkey visual systems
- Use many different copies of the same feature detector.
  - Copies have slightly different positions.
  - Could also replicate across scale and orientation.
    - Tricky and expensive
  - Replication **reduces the number of free parameters** to be learned.
- Use several **different feature types**, each with its own replicated pool of detectors.
  - Allows each patch of image to be represented in several ways.

Convolutional Neural Net

- Idea: statistics are similar at different locations (Lecun 1998)
- Connect each hidden unit to a small input patch and share the weight across space
- This is called a convolution layer and the network is a convolutional network



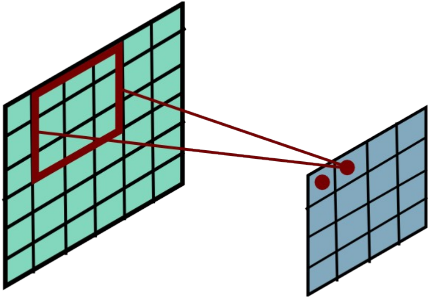
Convolutional Layer



Ranzato

$$h_j^n = \max(0, \sum_{k=1}^{K} h_k^{n-1} * w_{jk}^n)$$

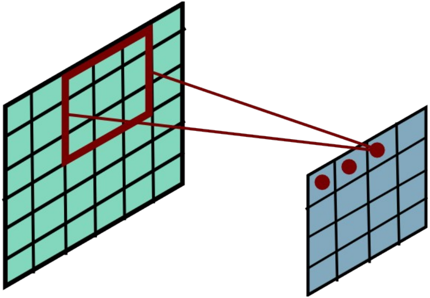
Convolutional Layer



Ranzato

$$h_j^n = \max(0, \sum_{k=1}^{K} h_k^{n-1} * w_{jk}^n)$$

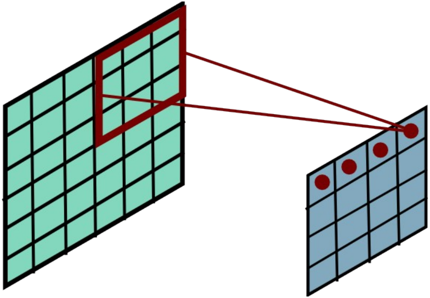
Convolutional Layer



Ranzato

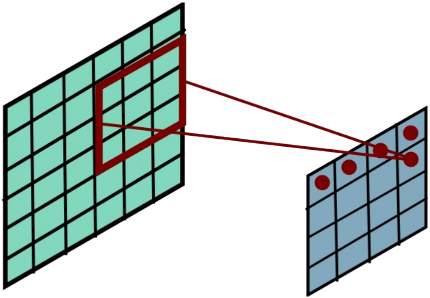
$$h_j^n = \max(0, \sum_{k=1}^{K} h_k^{n-1} * w_{jk}^n)$$

Convolutional Layer



$$h_j^n = \max(0, \sum_{k=1}^{K} h_k^{n-1} * w_{jk}^n)$$

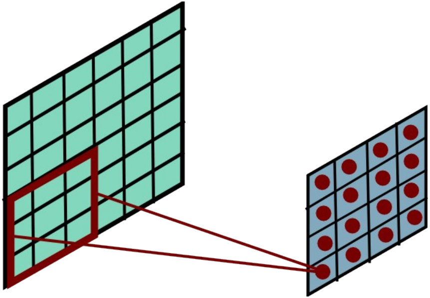
Convolutional Layer



Ranzato

$$h_j^n = \max(0, \sum_{k=1}^{K} h_k^{n-1} * w_{jk}^n)$$

Convolutional Layer



Ranzato

$$h_j^n = \max(0, \sum_{k=1}^{K} h_k^{n-1} * w_{jk}^n)$$

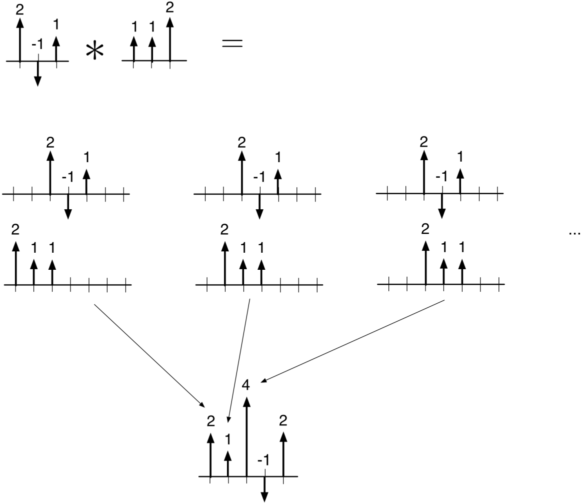
Convolution

- Convolution layers are named after the convolution operation.
- If a and b are two arrays,

$$(a * b)_t = \sum_{\tau} a_\tau b_{t-\tau}.$$

Convolution

"Flip and Filter" interpretation:

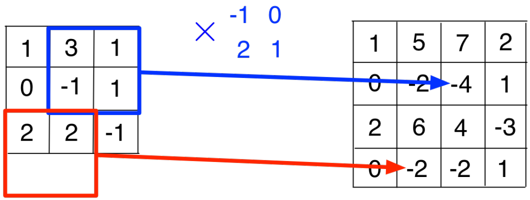


2-D Convolution

2-D convolution is analogous:

$$(A * B)_{ij} = \sum_s \sum_t A_{st} B_{i-s,j-t}.$$

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & -1 & 1 \\ 2 & 2 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix}$$



2-D Convolution

The thing we convolve by is called a kernel, or filter.

What does this convolution kernel do?

$*$

0	1	0
1	4	1
0	1	0

2-D Convolution

What does this convolution kernel do?

$*$

0	-1	0
-1	8	-1
0	-1	0

2-D Convolution

What does this convolution kernel do?

$*$

0	-1	0
-1	4	-1
0	-1	0

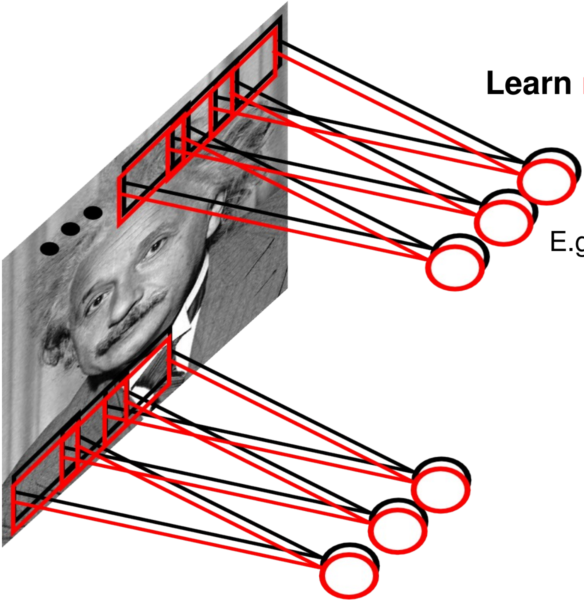
2-D Convolution

What does this convolution kernel do?

$*$

1	0	-1
2	0	-2
1	0	-1

Convolutional Layer



Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

54

Ranzato

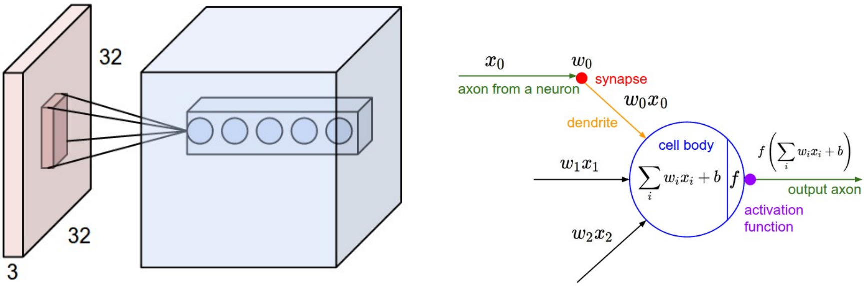


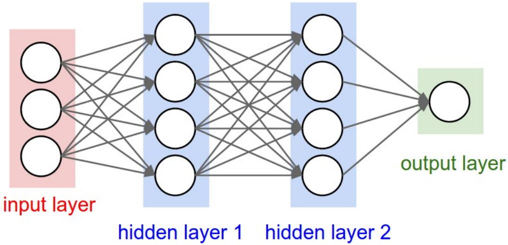
Figure: **Left:** CNN, **right:** Each neuron computes a linear and activation function

Hyperparameters of a convolutional layer:

- The number of filters (controls the **depth** of the output volume)
- The **stride**: how many units apart do we apply a filter spatially (this controls the spatial size of the output volume)
- The size $w \times h$ of the filters

[http://cs231n.github.io/convolutional-networks/]

MLP vs ConvNet



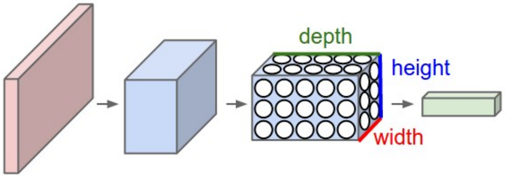
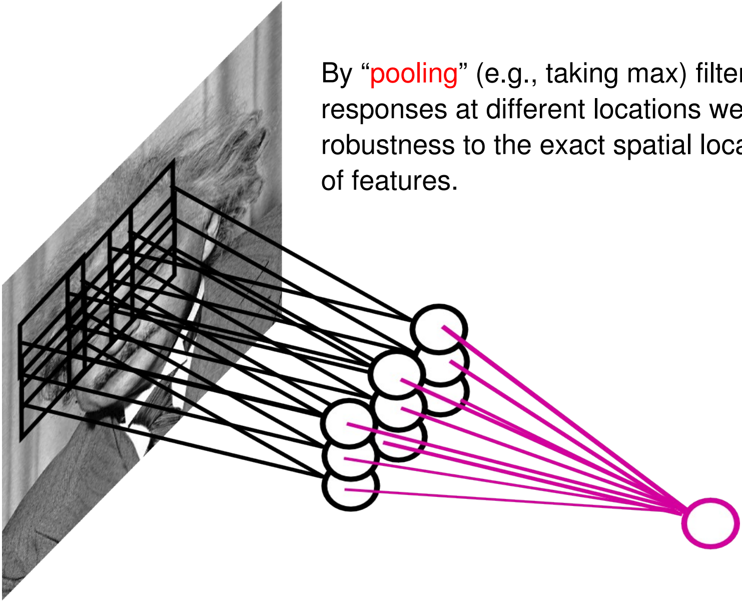


Figure : **Top:** MLP, **bottom:** Convolutional neural network

[http://cs231n.github.io/convolutional-networks/]

Pooling Layer



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

61

Ranzato

Pooling Options

- Max Pooling: return the maximal argument
- Average Pooling: return the average of the arguments
- Other types of pooling exist.

Pooling

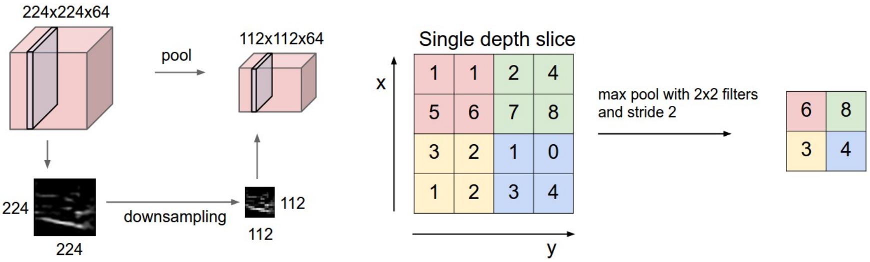


Figure: **Left:** Pooling, **right:** max pooling example

Hyperparameters of a pooling layer:

- The spatial extent F
- The stride

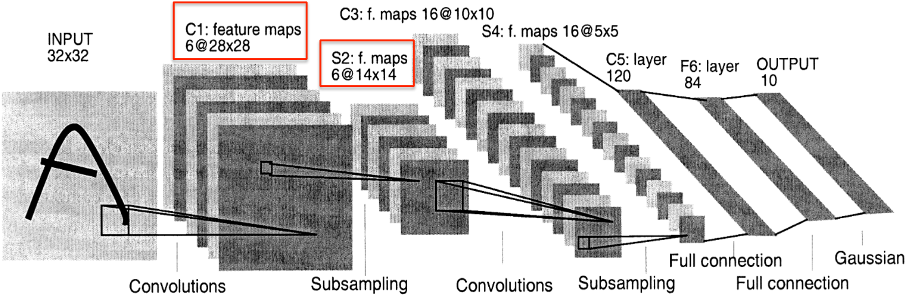
[http://cs231n.github.io/convolutional-networks/]

Backpropagation with Weight Constraints

- The backprop procedure from last lecture can be applied directly to conv nets.
- This is covered in csc421.
- As a user, you don't need to worry about the details, since they're handled by automatic differentiation packages.

LeNet

Here's the LeNet architecture, which was applied to handwritten digit recognition on MNIST in 1998:

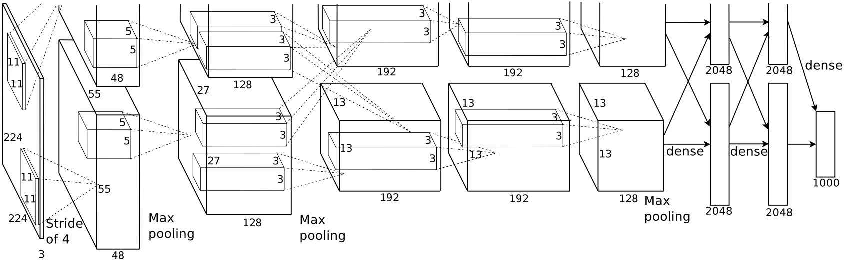


ImageNet

- Imagenet, biggest dataset for object classification: `http://image-net.org/`
- 1000 classes, 1.2M training images, 150K for test

AlexNet

- AlexNet, 2012. 8 weight layers. 16.4% top-5 error (i.e. the network gets 5 tries to guess the right category).

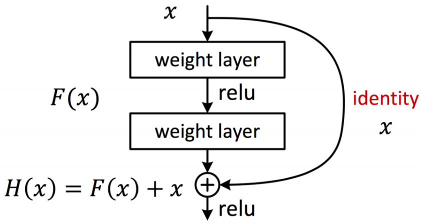


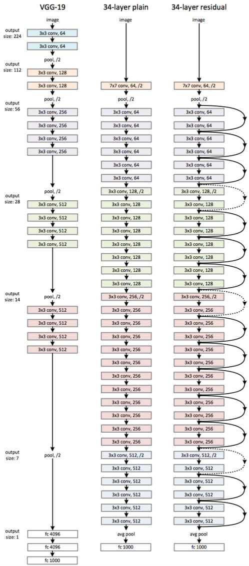
(Krizhevsky et al., 2012)

- The two processing pathways correspond to 2 GPUs. (At the time, the network couldn't fit on one GPU.)
- AlexNet's stunning performance on the ILSVRC is what set off the deep learning boom of the last 6 years.

150 Layers!

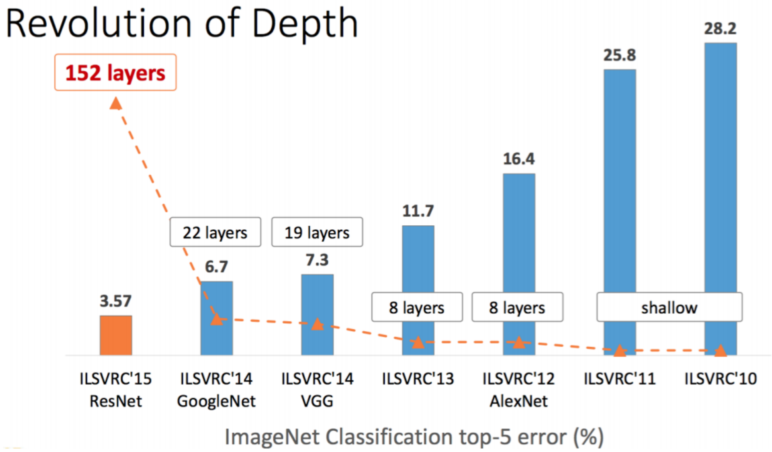
- Networks are now at 150 layers
- They use a skip connections with special form
- In fact, they don't fit on this screen
- Amazing performance!
- A lot of "mistakes" are due to wrong ground-truth





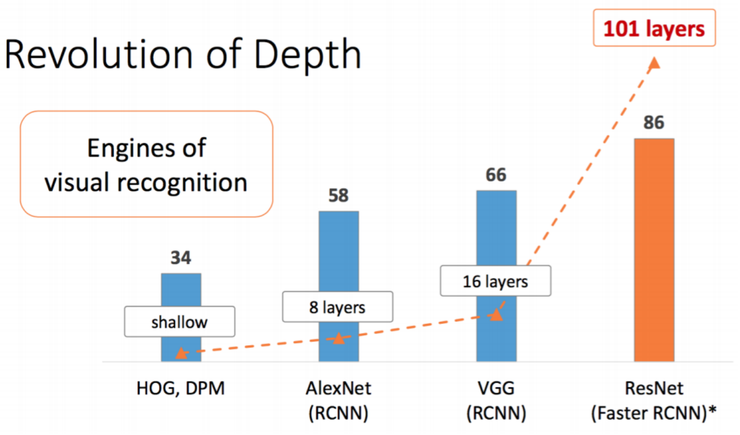
[He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Classification



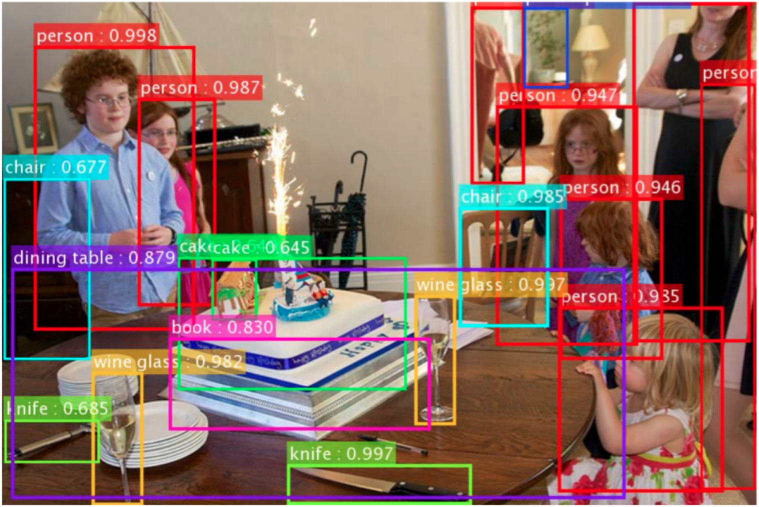
Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Detection



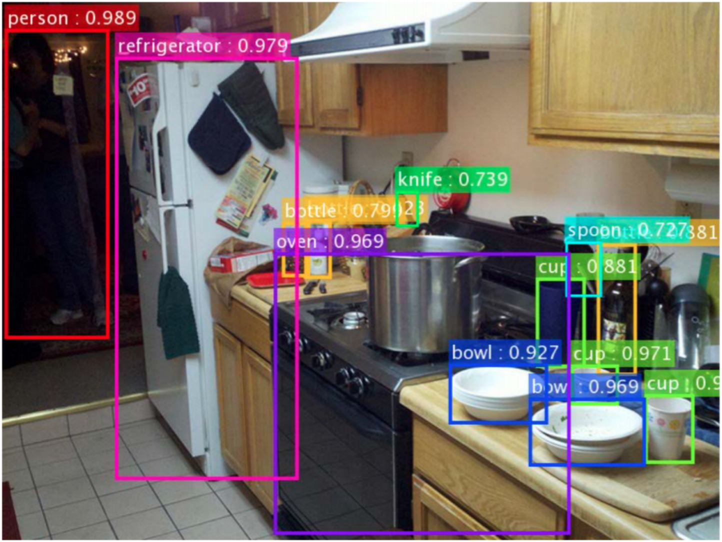
Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Detection



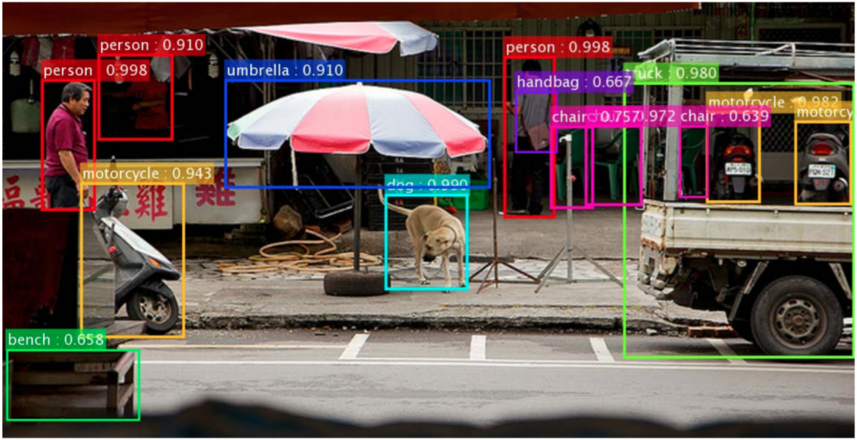
Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Detection



Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition

Results: Object Detection



Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

What do CNNs Learn?

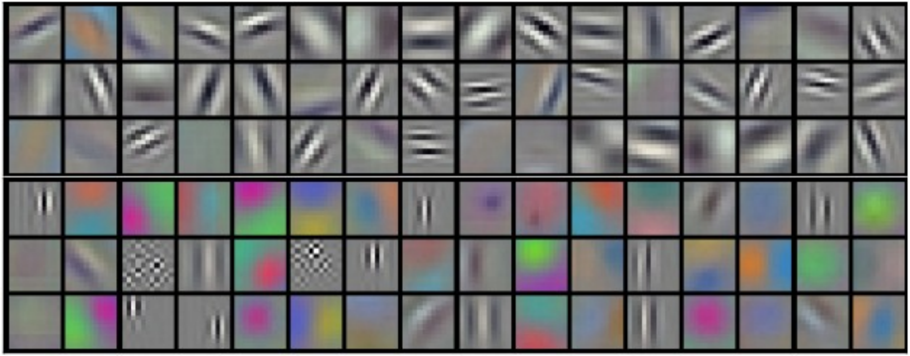


Figure: Filters in the first convolutional layer of Krizhevsky et al

What do CNNs Learn?

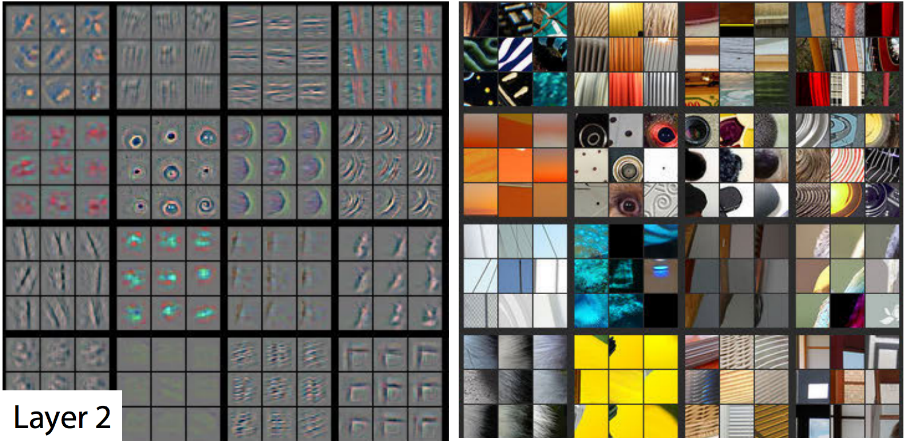


Figure: Filters in the second layer

[http://arxiv.org/pdf/1311.2901v3.pdf]

What do CNNs Learn?

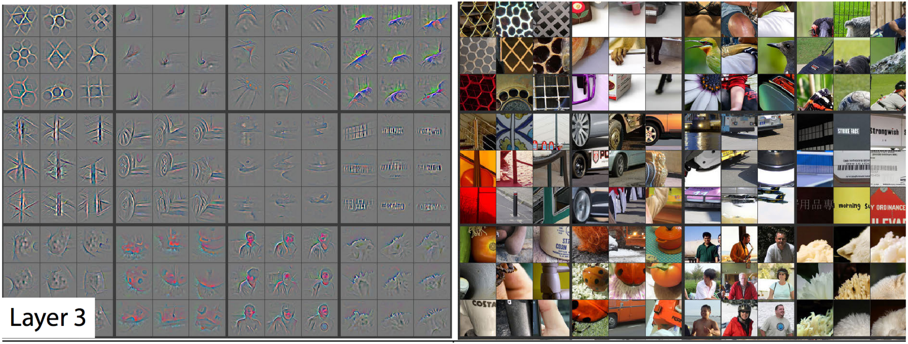
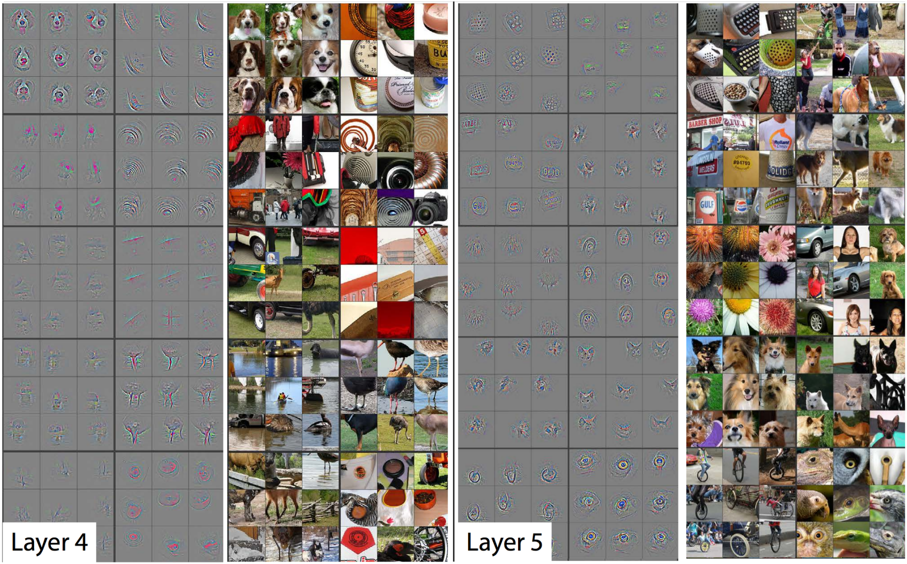


Figure: Filters in the third layer

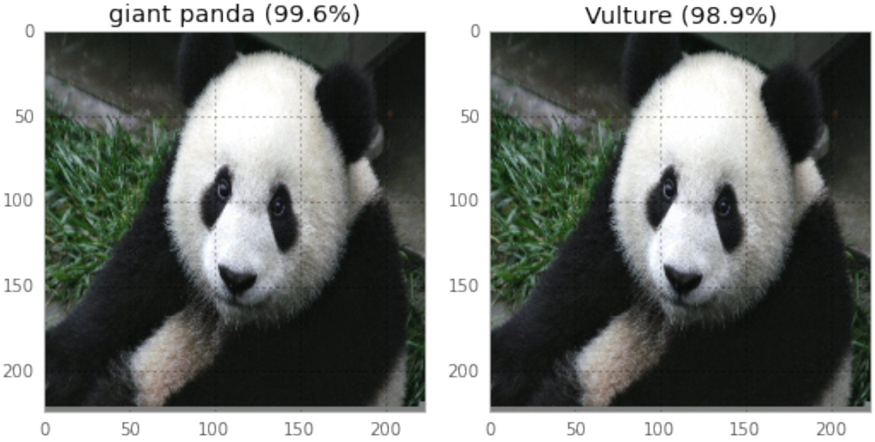
[http://arxiv.org/pdf/1311.2901v3.pdf]

What do CNNs Learn?



[http://arxiv.org/pdf/1311.2901v3.pdf]

Tricking a Neural Net



Read about it here (and try it!): https://codewords.recurse.com/issues/five/why-do-neural-networks-think-a-panda-is-a-vulture

Watch: https://www.youtube.com/watch?v=M2IebCN9Ht4

More on NNs

Figure : Generate images: http://arxiv.org/pdf/1511.06434v1.pdf

More on NNs

Generate text: https://vimeo.com/146492001, https://github.com/karpathy/neuraltalk2, https://github.com/ryankiros/visual-semantic-embedding

Figure : Compose music: https://www.youtube.com/watch?v=0VTI1BBLydE

Links

- Great course dedicated to NN: http://cs231n.stanford.edu
- Over source frameworks:
  - Pytorch http://pytorch.org/
  - Tensorflow https://www.tensorflow.org/
  - Caffe http://caffe.berkeleyvision.org/
- Most cited NN papers:
  https://github.com/terryum/awesome-deep-learning-papers