CSC 411 Lecture 11: Neural Networks II

Ethan Fetaya, James Lucas and Emad Andrews

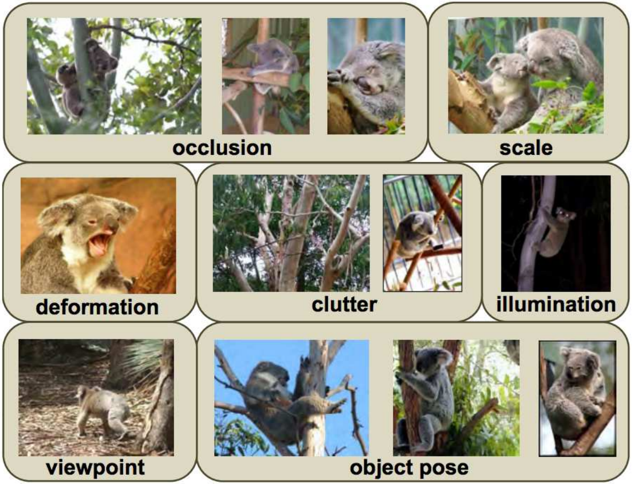
University of Toronto

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]

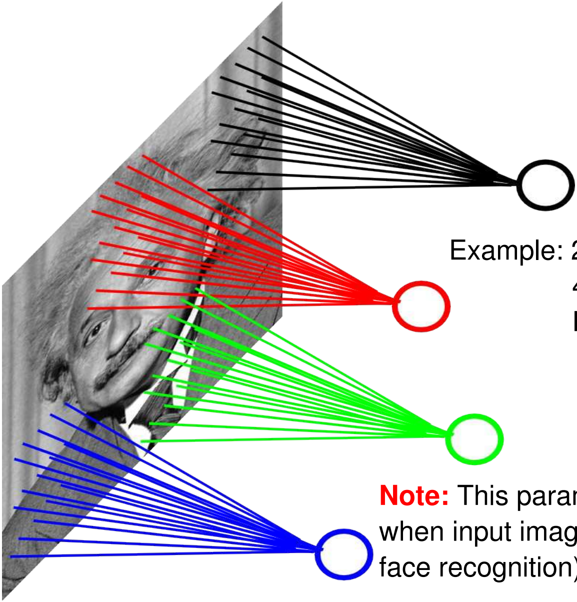
Neural Nets for Object Recognition

- 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]

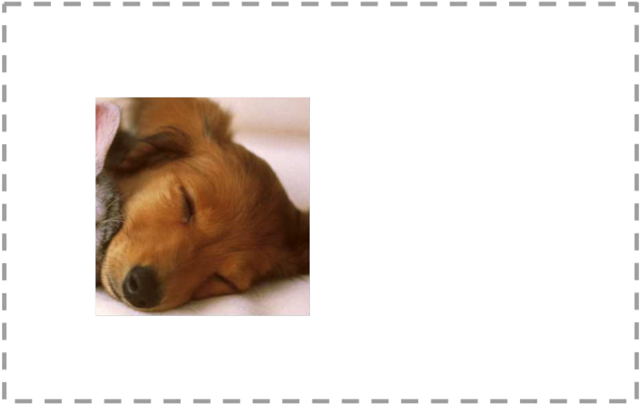
General Images

- The object can be anywhere

[Slide: Y. Zhu]

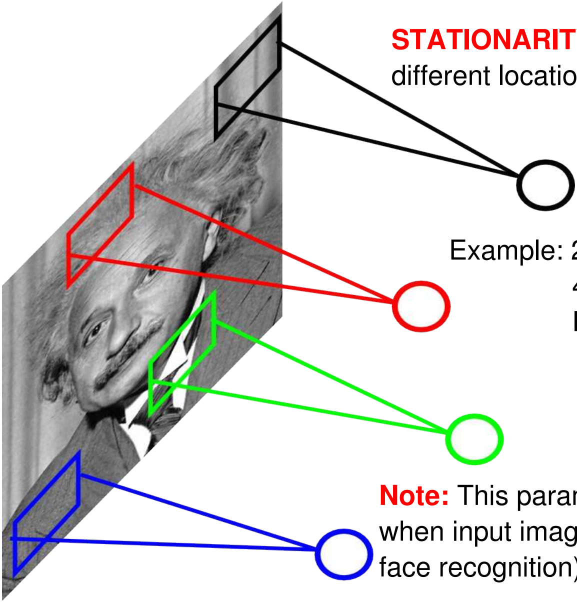
General Images

- The object can be anywhere



[Slide: Y. Zhu]

Locally Connected Layer



STATIONARITY? Statistics is similar at different locations

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

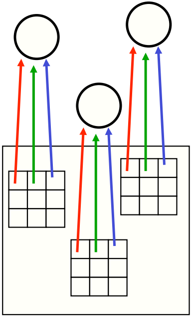
Note: This parameterization is good when input image is registered (e.g., face recognition).

35

Ranzato

The replicated feature approach

The red connections all have the same weight.

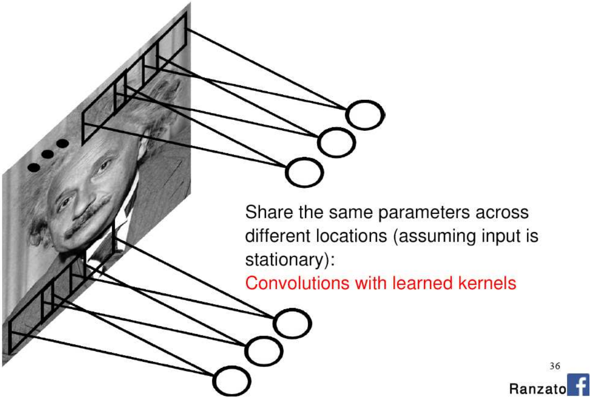


5

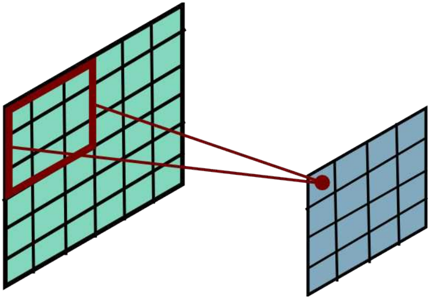
- Adopt approach apparently used in monkey visual systems
- Use many different copies of the same feature detector.
  - Copies have slightly different positions.
  - Could also replicate across scale and orientation.
    - Tricky and expensive
  - Replication reduces number of free parameters to be learned.
- 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



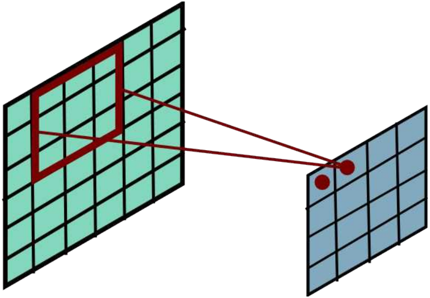
Convolutional Layer



Ranzato

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

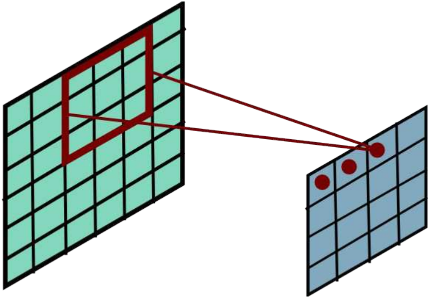
Convolutional Layer



Ranzato

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

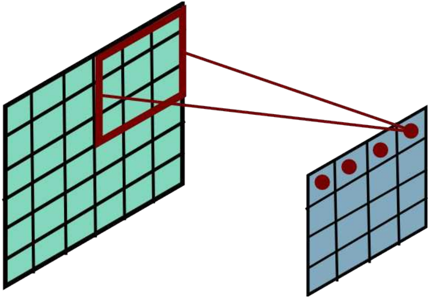
Convolutional Layer



Ranzato

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

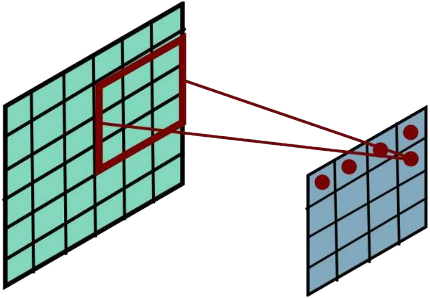
Convolutional Layer



Ranzato

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

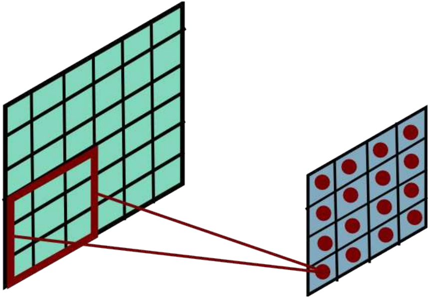
Convolutional Layer



Ranzato

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

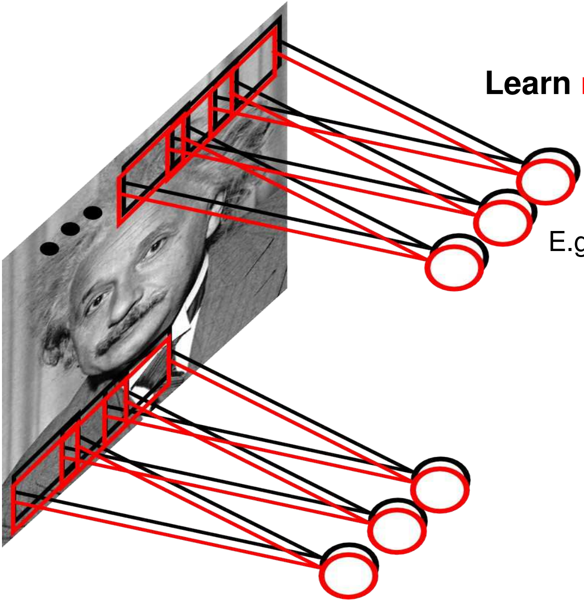
Convolutional Layer



Ranzato

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

Convolutional Layer



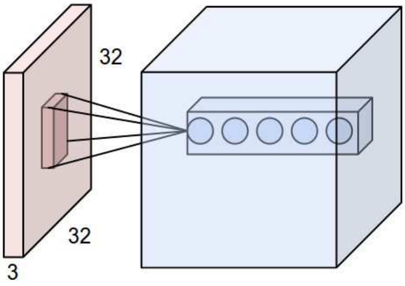
Learn multiple filters.

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

54

Ranzato

Convolutional Layer



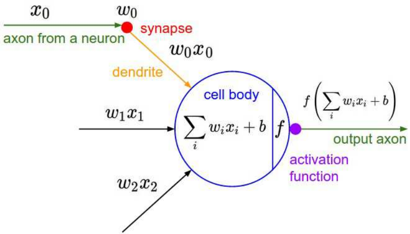


Figure: **Left:** CNN, **right:** Each neuron computes a linear and activation function

Hyperparameters of a convolutional layer:

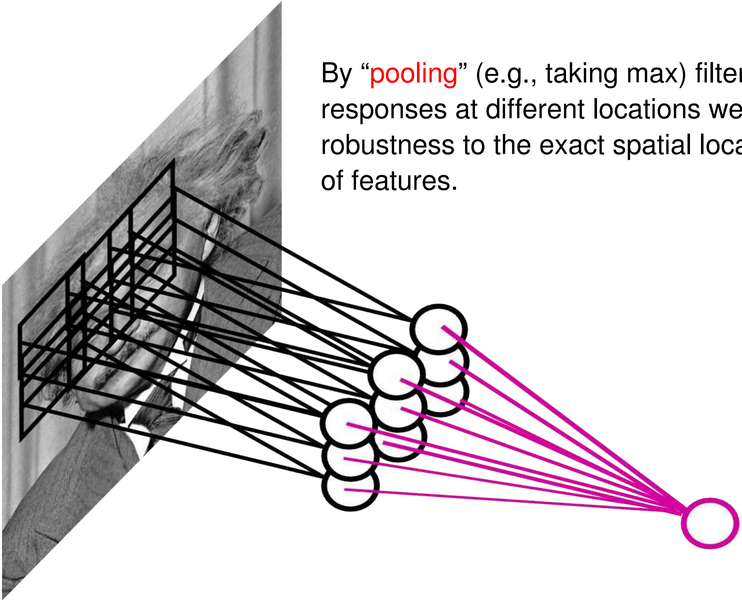
- The number of filters (controls the **depth** of the output volume)
- The **stride**: how many units apart do we apply a filter spatially (this controls the spatial size of the output volume)
- The size $w \times h$ of the filters

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

Output size

- If the input is $HxWxC_{in}$ and the kernel size is $k_1xk_2xC_{out}$ what is the output size?
  - $(H - k_1 + 1) \times (W - k_2 + 1) \times C_{out}$
- Input is $HxWxC_{in}$ and the kernel size is $k_1xk_2xC_{out}$ with stride s?
  - $H_{out} = \lfloor (H - k_1)/s + 1 \rfloor$
- Input is $HxWxC_{in}$ and the kernel size is $k_1xk_2xC_{out}$ with stride s with padding p?
  - $H_{out} = \lfloor (H + 2p - k_1)/s + 1 \rfloor$
- Without padding we can't have a very deep network (the size shrinks every convolution)

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

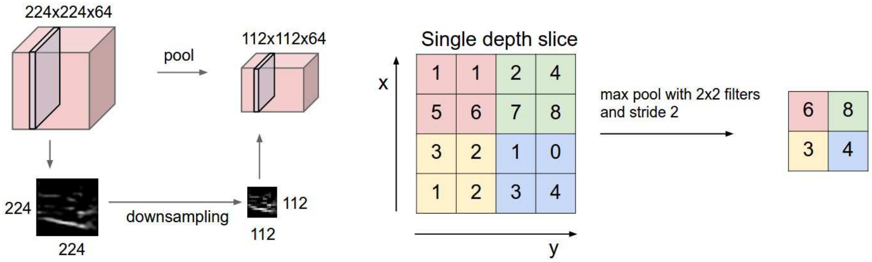


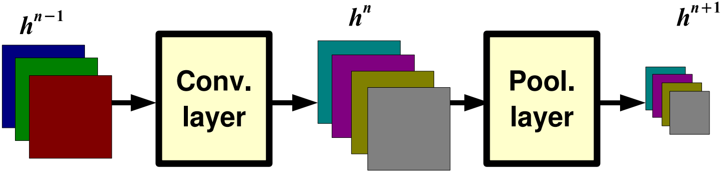
Figure: **Left:** Pooling, **right:** max pooling example

Hyperparameters of a pooling layer:

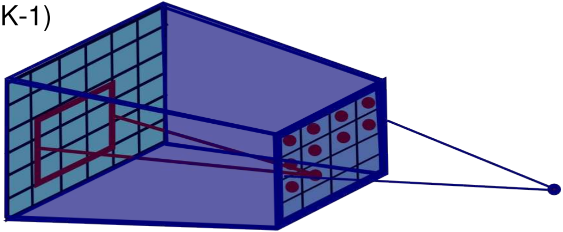
- The spatial extent F
- The stride

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

Pooling Layer: Receptive Field Size



If convolutional filters have size KxK and stride 1, and pooling layer has pools of size PxP, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: (P+K-1)x(P+K-1)



67

Ranzato

Backpropagation with Weight Constraints

- It is easy to modify the backpropagation algorithm to incorporate linear constraints between the weights

$$\begin{aligned} \text{To constrain:} \quad w_1 &= w_2 \\ \text{we need:} \quad \Delta w_1 &= \Delta w_2 \end{aligned}$$

- We compute the gradients as usual, and then modify the gradients so that they satisfy the constraints.

$$\begin{aligned} \text{compute:} \quad & \frac{\partial E}{\partial w_1} \text{ and } \frac{\partial E}{\partial w_2} \\ \text{use:} \quad & \frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial w_2} \text{ for } w_1 \text{ and } w_2 \end{aligned}$$

- So if the weights started off satisfying the constraints, they will continue to satisfy them.
- This is an intuition behind the backprop. In practice, write down the equations and compute derivatives (it's a nice exercise, do it at home)

Now let's make this very deep to get a real state-of-the-art object recognition system

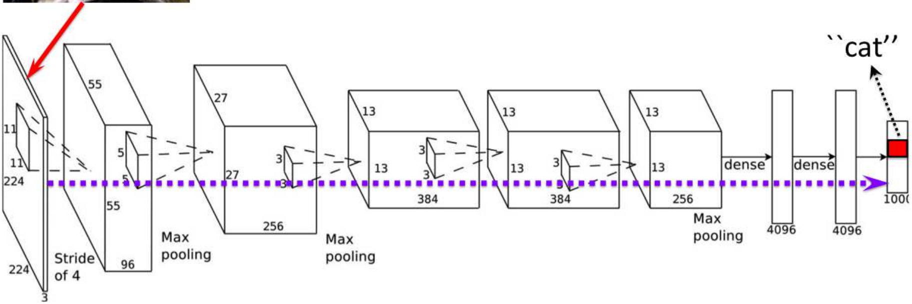
Convolutional Neural Networks (CNN)

- Basic filtering idea from computer vision/image processing
- If our filter is $[-1, 1]$, you get a vertical edge detector
- Now imagine we want to have many filters (e.g., vertical, horizontal, corners, one for dots). We will use a filterbank.
- So applying a filterbank to an image yields a cube-like output, a 3D matrix in which each slice is an output of convolution with one filter. We apply an activation function on each hidden unit (typically a ReLU).
- Do some additional tricks. A popular one is called max pooling. Any idea why you would do this?
- Do some additional tricks. A popular one is called max pooling. Any idea why you would do this? To get invariance to small shifts in position.
- Now add another "layer" of filters. For each filter again do convolution, but this time with the output cube of the previous layer.

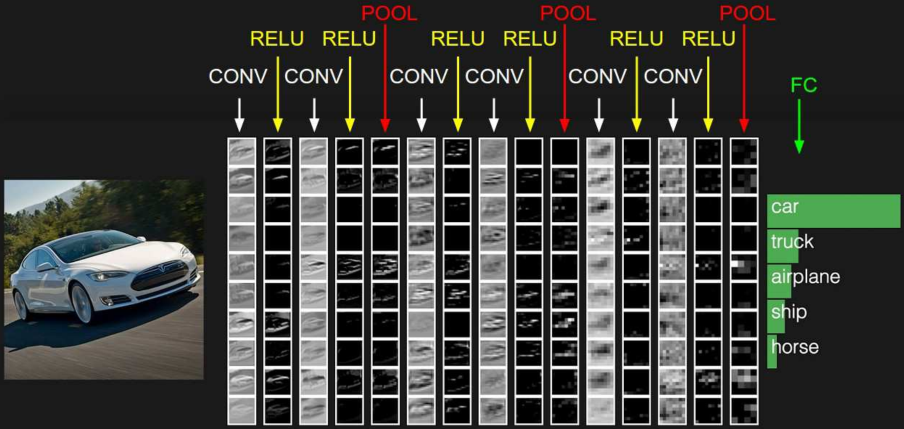
Classification

- Once trained we feed in an image or a crop, run through the network, and read out the class with the highest probability in the last (classif) layer.

What's the class of this object?

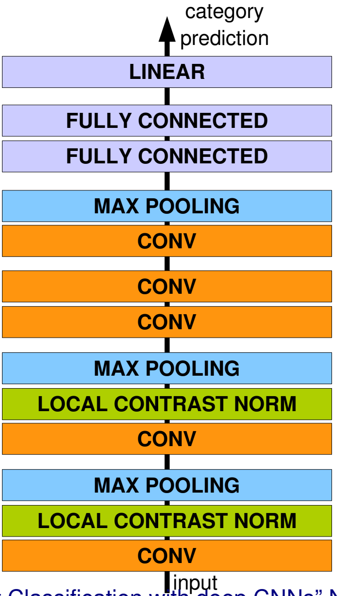


Example



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

Architecture for Classification



Krizhevsky et al. "ImageNet Classification with deep CNNs" NIPS 2012

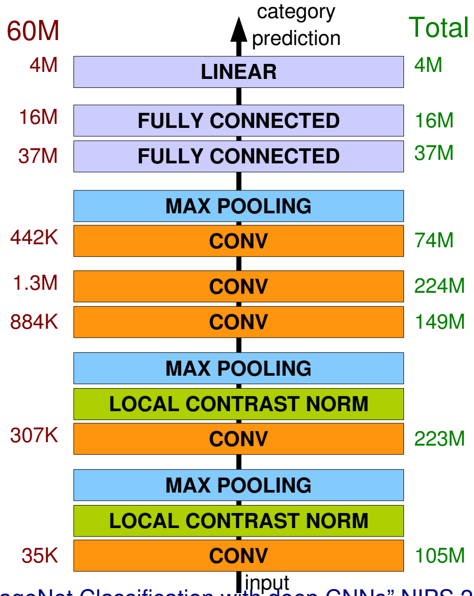
95

Ranzato

Architecture for Classification

Total nr. params: 60M

Total nr. flops: 832M



96

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012

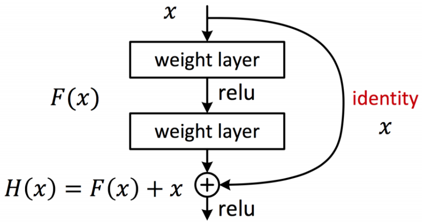
Ranzato

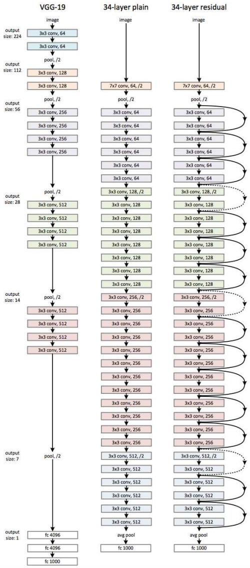
ImageNet

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

150 Layers!

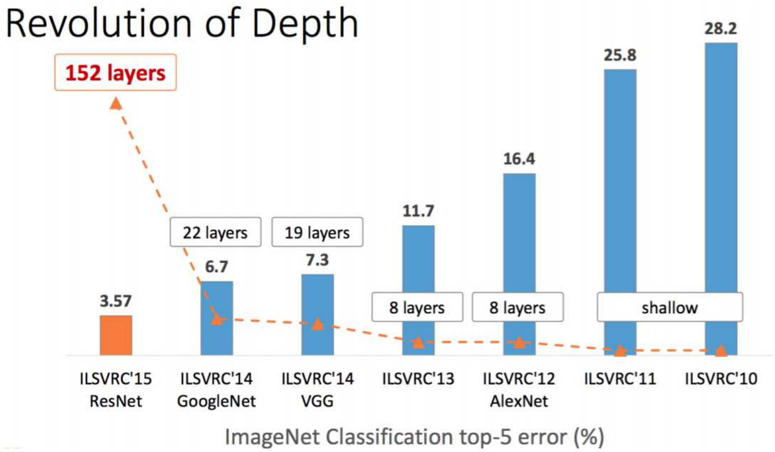
- Networks are now at 150 layers
- They use a skip connections with special form
- In fact, they don't fit on this screen
- Amazing performance!
- A lot of "mistakes" are due to wrong ground-truth





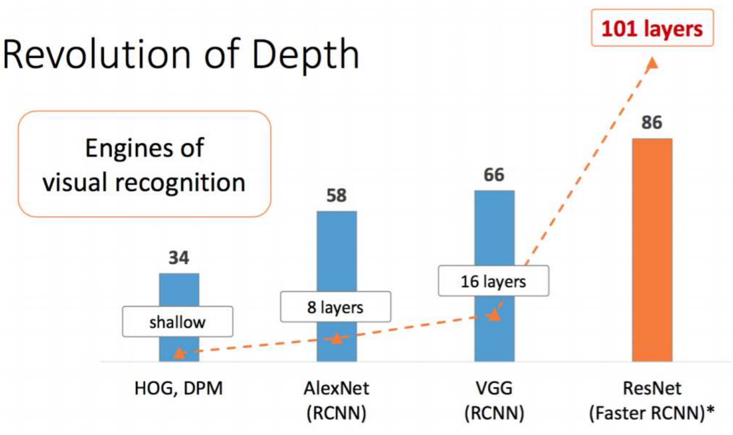
[He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Classification



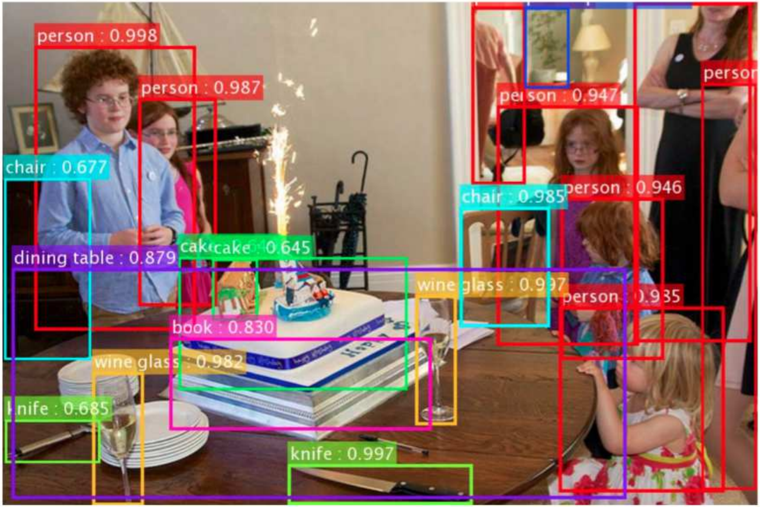
Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Detection



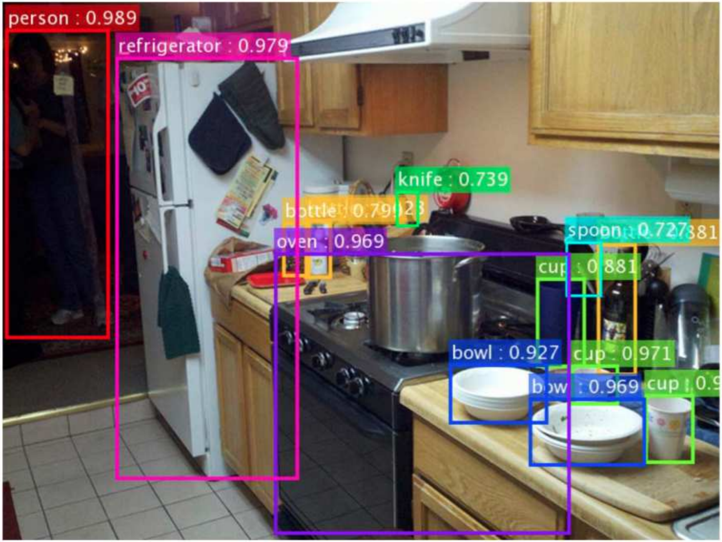
Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

Results: Object Detection



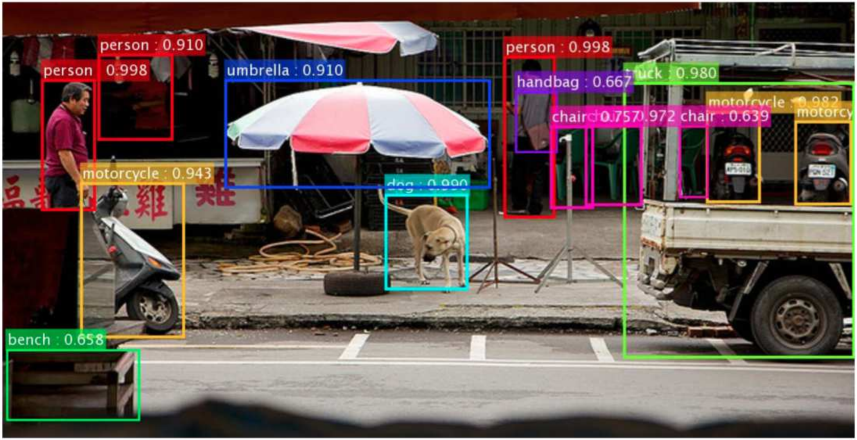
Slide: R. Liao, Paper: [He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2016]

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?

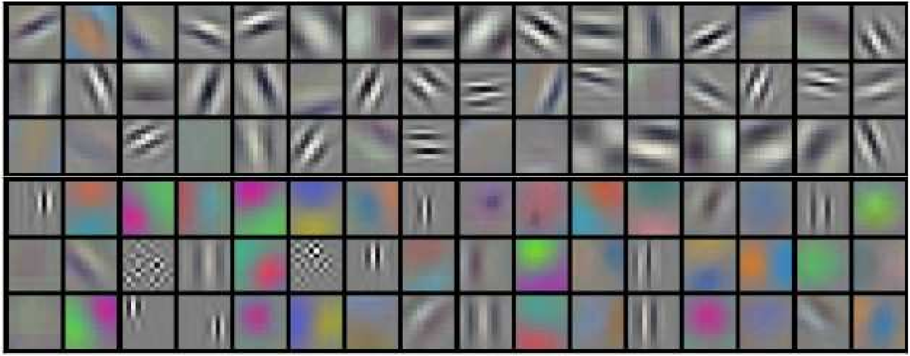


Figure: Filters in the first convolutional layer of Krizhevsky et al

What do CNNs Learn?

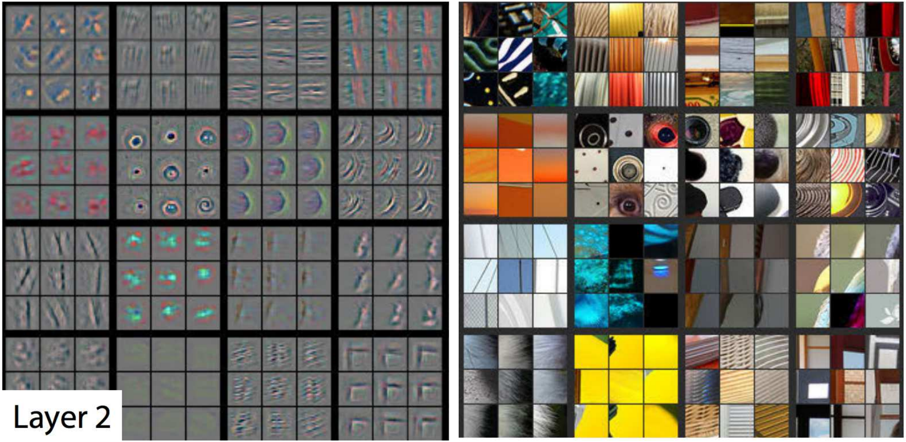


Figure: Filters in the second layer

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

What do CNNs Learn?

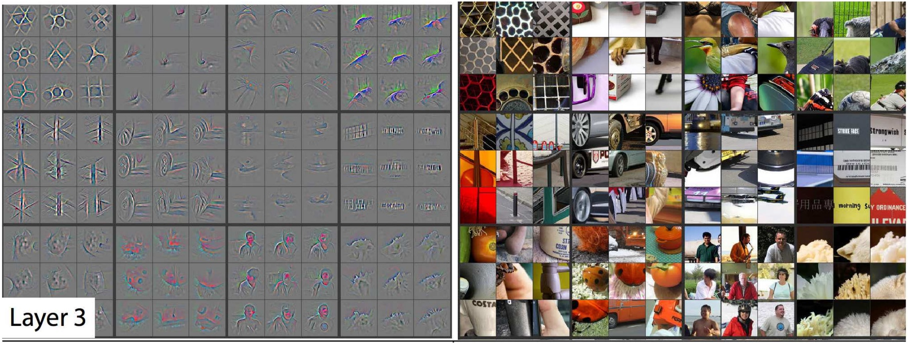
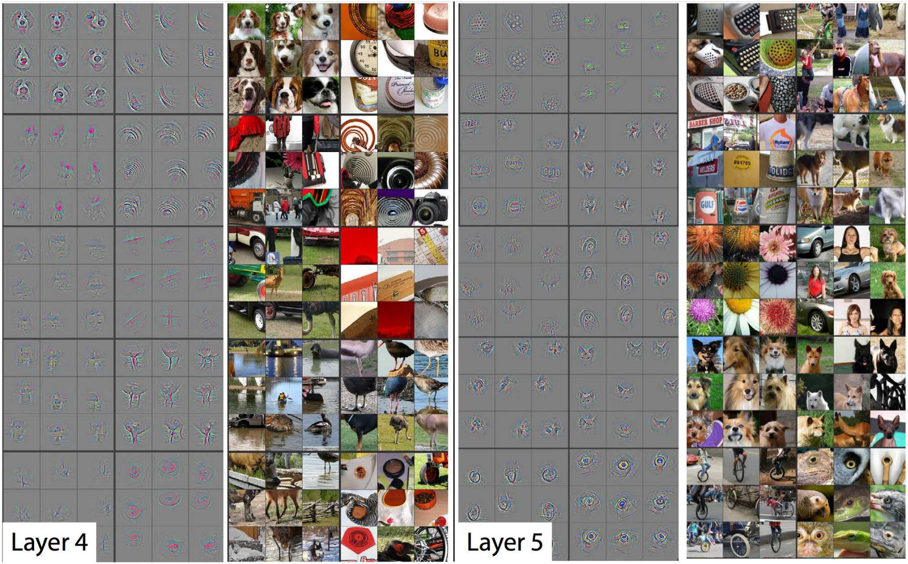


Figure: Filters in the third layer

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

What do CNNs Learn?



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

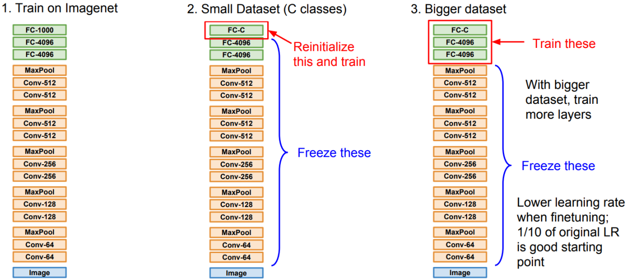
How to Train Good CNNs

- Normalize your data (standard trick: subtract mean, divide by standard deviation)
- Augment your data (add image flips, rotations, etc)
- Keep training data balanced
- Shuffle data before batching
- In training: Random initialization of weights with proper variance
- Monitor your loss function, and accuracy (performance) on validation
- If your labeled image dataset is small: pre-train your CNN on a large dataset (eg Imagenet), and fine-tune on your dataset

[Slide: Y. Zhu, check tutorial slides and code:
http://www.cs.utoronto.ca/~fidler/teaching/2015/CSC2523.html]

Transfer learning

- Main reason DL helps on (almost) any vision task, even when you don't have a huge dataset!



[From: http://cs231n.github.io/]

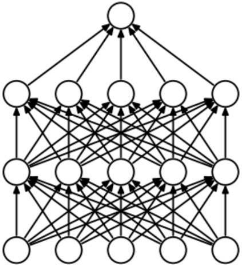
Overfitting

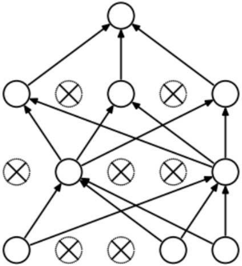
How to control overfitting?

- Early stopping
  - You don't have to take the last iteration!
  - Check validation during training (every few iterations/epoch) and take the best one.
- Weight decay
  - L_2 regularization, usually around $1e-4$
- Adding random noise
  - Dropout
  - Other ideas like Gaussian noise, batch normalization

Dropout

- At each iteration "kill" each neuron with probability p (usually 0.5).


(a) Standard Neural Net


(b) After applying dropout.

- The expected value decreased by p, fix by multiplying by $1/p$.
- At test time just use trained weights.

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