Announcements

• PS3 due 6/7 (Thurs), 11:59 pm

• Review session during Thurs lecture
  – Post questions on piazza

Convolutional Neural Networks (CNN)

• Neural network with specialized connectivity structure
• Stack multiple stages of feature extractors
• Higher stages compute more global, more invariant, more abstract features
• Classification layer at the end

Convolutional Neural Networks (CNN)

- Feed-forward feature extraction:
  1. Convolve input with learned filters
  2. Apply non-linearity
  3. Spatial pooling (downsample)

- Supervised training of convolutional filters by back-propagating classification error

Adapted from Liora Lazebnik

Convolutional Neural Networks (CNN) diagram

- Input Image
- Convolution (Learned)
- Non-linearity
- Spatial pooling
- Output (class probs)

Convolutions: More detail

32x32x3 image

width

height

depth

5x5x3 filter

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

Convolution Layer

- 32x32x3 image
- 5x5x3 filter

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)

Convolutions: More detail

Convolution Layer

- 32x32x3 image
- 5x5x3 filter

Convolve (slide) over all spatial locations

Convolutions: More detail

Convolution Layer

Consider a second, green filter

Convolve (slide) over all spatial locations
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!

Convolutions: More detail

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

\[ C(i,j) = \sum_{i'=-k}^{k} \sum_{j'=-k}^{k} H(i', j') * X(i-i', j-j') \]

Element-wise multiplication and sum of a filter and the signal (image)

Convolutions: More detail

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions
Convolutional Layers

ConvNet is a sequence of Convolutional Layers, interspersed with activation functions:

- **CONV, ReLU** e.g. 6 5x5x3 filters
- **CONV, ReLU** e.g. 10 5x5x6 filters

An example of a 32x32x3 image convolved with a 5x5x3 filter:

- 7x7 input (spatially)
- Assume 3x3 filter

A diagram illustrates the spatial convolutions and activation maps.
Convolutions: More detail
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
Convolutions: More detail
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
=> 5x5 output

Convolutions: More detail
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied with stride 2

Convolutions: More detail
A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied with stride 2
Convolutions: More detail

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied with stride 2
=> 3x3 output!

Convolutions: More detail

A closer look at spatial dimensions:

7x7 input (spatially)
assume 3x3 filter
applied with stride 3?

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.
Output size:
\[(N - F) / \text{stride} + 1\]

e.g. \(N = 7, F = 3;\)
\[
\text{stride 1} => (7 - 3)/1 + 1 = 5
\]
\[
\text{stride 2} => (7 - 3)/2 + 1 = 3
\]
\[
\text{stride 3} => (7 - 3)/3 + 1 = 2.33
\]

A Common Architecture: AlexNet
Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Only 3x3 CONV stride 1, pad 1 and 2x2 MAX POOL stride 2

best model

11.2% top 5 error in ILSVRC 2013

→

7.3% top 5 error

Andrej Karpathy

Case Study: GoogLeNet

[Szegedy et al., 2014]

Inception module

ILSVRC 2014 winner (6.7% top 5 error)

Andrej Karpathy

Case Study: ResNet

[He et al., 2015]

ILSVRC 2015 winner (3.6% top 5 error)

MSRA @ ILSVRC & COCO 2015 Competitions

- 1st places in all five main tasks
- Imagery Classification: "Unnatural" score = 152-layer res
- ImageNet Detection: 16% better than 2nd
- ImageNet Localization: 1.7% better than 2nd
- GDEB Detection: 1.1% better than 2nd
- GDEB Segmentation: 1.1% better than 2nd

Slide from Kaiming He’s recent presentation: https://www.youtube.com/watch?v=1PGLj-uKTW4
Case Study: ResNet

Andrej Karpathy

[He et al., 2015]
ILSVRC 2015 winner (3.6% top 5 error)

Revolution of Depth

ImageNet Classification top-5 error (%)

2-3 weeks of training on 8 GPU machine

Practical matters
Comments on training algorithm

- Not guaranteed to converge to zero training error, may converge to local optima or oscillate indefinitely.
- However, in practice, does converge to low error for many large networks on real data.
- Thousands of epochs (epoch = network sees all training data once) may be required, hours or days to train.
- To avoid local-minima problems, run several trials starting with different random weights (random restarts), and take results of trial with lowest training set error.
- May be hard to set learning rate and to select number of hidden units and layers.
- Neural networks had fallen out of fashion in 90s, early 2000s; back with a new name and significantly improved performance (deep networks trained with dropout and lots of data).

Over-training prevention

- Running too many epochs can result in over-fitting.
- Keep a hold-out validation set and test accuracy on it after every epoch. Stop training when additional epochs actually increase validation error.

Training: Best practices

- Use mini-batch
- Use regularization
- Use cross-validation for your parameters
- Use RELU or leaky RELU, don’t use sigmoid
- Center (subtract mean from) your data
- Learning rate: too high? too low?
Regularization: Dropout

- Randomly turn off some neurons
- Allows individual neurons to independently be responsible for performance

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Adapted from Jia-bin Huang

Data Augmentation (Jittering)

Create virtual training samples
- Horizontal flip
- Random crop
- Color casting
- Geometric distortion

Transfer Learning

“You need a lot of data if you want to train deep NNs”

Andrej Karpathy
Transfer Learning with CNNs

- The more weights you need to learn, the more data you need
- That’s why with a deeper network, you need more data for training than for a shallower network
- One possible solution:

Set these to the already learned weights from another network
Learn these on your own task

Transfer Learning with CNNs

Source: classification on ImageNet
Target: some other task/data

1. Train on ImageNet
2. Small dataset: Freeze these
3. Medium dataset: Freezing more data = retain more of the network (or all of it)
   Freeze these

Summary

- We use deep neural networks because of their strong performance in practice
- Convolutional neural networks (CNN)
  - Convolution, nonlinearity, max pooling
- Training deep neural nets
  - We need an objective function that measures and guides us towards good performance
  - We need a way to minimize the loss function: stochastic gradient descent
  - We need backpropagation to propagate error through all layers and change their weights
- Practices for preventing overfitting
  - Dropout; data augmentation; transfer learning