Deep Learning for Computer Vision

Lecture 11: Building ConvNets in Keras, Pre-trained Neural Networks, Performing Network Surgery, Domain Transfer and Fine-Tuning

Peter Belhumeur

Computer Science
Columbia University
CIFAR-10

[Krizhevsky 2009]

10 categories x 6,000 instances = 60,000 images (32x32x3)
Small Convolutional Net

Conv layer:
- k=3x3
- stride=1
- pad=1
- no pooling
- depth=32

Conv layer:
- k=3x3
- stride=1
- pad=1
- no pooling
- max pool=2
- depth=64

Conv layer:
- k=3x3
- stride=1
- pad=1
- no pooling
- max pool=2
- depth=64

FC layer:
- dim=512

Output + Softmax

32x32x3 → 32x32x32 → 16x16x32 → 16x16x64 → 7x7x64 → 1x1x512 → 1x1x10
Cats vs. Dogs

Data from [Kaggle.com]

2 categories x 11,000 instances = 22,000 images
Small Convolutional Net

150x150x3 → Conv layer: k=3x3, stride=1, pad=0, max pool=2, depth=32 → 74x74x32

74x74x32 → Conv layer: k=3x3, stride=1, pad=0, max pool=2, depth=32 → 36x36x32

36x36x32 → Conv layer: k=3x3, stride=1, pad=0, max pool=2, depth=64 → 17x17x64

17x17x64 → FC layer: dim=64

Output + Softmax → 0.94
Open imagenet_vgg16_fine-tuning.html in browser
Why can’t we use one a much deeper ConvNet for this problem?
Open imagenet_vgg16_fine-tuning.html in browser
We usually don’t have enough data!
...but here is a FANTASTIC trick!
Domain Transfer

- Train a **deep** ConvNet using lots of data on a large image classification problem like ImageNet.

- Save the weights.

- Chop off the output layer (or final layers) of this ConvNet.

- Replace the output layer (or final layers) with one that fits your problem.

- Freeze the weights in the old layers and train on your data, allowing the weights in the new layer to settle.

- Unfreeze the whole network and train.
VGG16 [Simonyan and Zisserman, 2015]
ImageNet

[Deng et al. 2009]

20,000+ categories x ~1000 instances = 14,000,000+ images
Domain Transfer + Fine-Tuning
Network Surgery

224 × 224 × 3  224 × 224 × 64

112 × 112 × 128

56 × 56 × 256

28 × 28 × 512

14 × 14 × 512

7 × 7 × 512

1 × 1 × 4096  1 × 1 × 1000

convolution + ReLU
max pooling
fully connected + ReLU
softmax

Keep ← Toss →
Network Surgery

- Keep
- Add

- $224 \times 224 \times 3$
- $224 \times 224 \times 64$
- $112 \times 112 \times 128$
- $56 \times 56 \times 256$
- $28 \times 28 \times 512$
- $14 \times 14 \times 512$
- $7 \times 7 \times 512$

- $1 \times 1 \times 256$
- $1 \times 1 \times 1$

- convolution+ReLU
- max pooling
- fully connected+ReLU
- softmax
First, train the top layers
Then, train the whole network
Open imagenet_vgg16_fine-tuning.html in browser