Problem Set 4 Due Tonight!

Problem Set 5 will be posted shortly.
OBJECT DETECTION

Newer methods also use a neural network to generate "region proposals"

Efficient Implementations: bulk of the computation happens once on the entire image, and you crop a feature map for each region.

Even Faster Methods: Discretize image locations into grid, and directly output up to a fixed number of bounding boxes for each grid block.

TRANSFER LEARNING

Say you want to train a network to solve a problem.

- The task is complex, so you need a large network.
- But you don’t have enough training data to train such a network.
- Pick a related task for which you do have a lot of training data
  - ImageNet is a great database for this for a variety of semantic tasks
- Train a network (like VGG-16) to solve that task.
- Then, choose the output of some intermediate layer of that network
- Use it as a feature vector, and learn a smaller network for your problem which goes from those features to the desired output.

VGG-16 does well on Imagenet classification and gives you a feature representation that is surprisingly useful for a broad range of tasks.

Remember computing encoding $\tilde{x}$ from $x$: VGG-16’s pool5, fc1, fc2, features can be the $\tilde{x}$ for many tasks.

One can also "initialize" a network with the VGG-16 architecture to one trained with imagenet, and then "finetune" by replacing the final layer as classification for another task.

In general, empirical question to determine when training on Task A will provide good features for Task B.
CLASSIFICATION FOR OTHER TASKS

- Initialize Randomly
- Train on Imagenet
- Throw away final few layers
CLASSIFICATION FOR OTHER TASKS

- Initialize Randomly
- Train on ImageNet
- Throw away final few layers
- Keep Fixed

CLASSIFICATION FOR OTHER TASKS

- Initialize Randomly
- Train on ImageNet
- Throw away final few layers
- Keep Fixed / FineTune

CLASSIFICATION FOR OTHER TASKS

R-CNN: Regions with CNN features

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

Treat detection as "Classification" of a patch treated as an image

CLASSIFICATION FOR OTHER TASKS

Treat segmentation as "Classification" of the center pixel of a patch, with the patch given as input and treated as an image

Same approach: Pre-train on ImageNet, use activations or fine-tune.
The problem: Lots of patches (one for each pixel)

Let's think in 1D. And assume we had two conv layers, and then a fully connected layer.

No need to repeat computations.

Produces a Spatial Label Map
FULLY-CONVOLUTIONAL NETWORKS

But what about downsampling?

- Option 0: Just don’t use downsampling
  Bad, because down-sampling is a way to quickly increase the "receptive field" of your network.

- Option 1: Just produce a label map at lower-resolution.
- Option 2: If you downsample by \( N \) (typically \( N = 2^k \))
  Feed every \((N - 1) \times (N - 1)\) "shifted" version of your input through this FCN.
  Bad because if you down-sample multiple times, you’re still re-computing activations prior to the last-downsampling.

- Option 3: Dilated Convolutions

DILATED CONVOLUTION

In your original network, everytime you have downsampling, you should remove that stride (set it to 1), and multiply it with the dilation factor of all subsequent spatial layers (conv and pool).
- Deeper networks are able to express more complex functions. Larger hypothesis space.
- Empirically: deeper networks tend to overfit less than wider networks (i.e., more layers better than more channels)

- But, you need to back-propagate gradients of the loss computed at the final output back through a larger number of layers.
- With multiple non-linearities, gradients have a higher probability of vanishing.
- So, deeper networks are harder to train.
- Solution: provide an easier path for gradients back to initial layers.

Deeply-Supervised Nets

Chen-Yu Lee  
Dep. of EECS, UCSD  
edl93@ucsd.edu

Saining Xie  
Dep. of CS and Engg., UCSD  
sxie@cs.ucsd.edu

Patrick Gallagher  
Microsoft Research  
rigobertg@gmail.com

Zhenguo Zhang  
Microsoft Research  
zhengou@outlook.com

Qihang Yu  
Dep. of CSE, UCSD  
qihangy@gmail.com

David Ho  
Dep. of EECS, UCSD  
dho@ecs.ucsd.edu
Deep Residual Learning for Image Recognition

Kaiming He  Xiangyu Zhang  Shaoqing Ren  Jian Sun
Microsoft Research

{kahe, vx-zh, vx-ssh, jiansun}@microsoft.com

Figure 2. Residual learning: a building block.

Densely Connected Convolutional Networks

Gao Huang*  Zhuang Liu*  Laurens van der Maaten
Cornell University  Tongji University  Facebook AI Research

*Equal contribution

Abstract

Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient than other architectures. A new type of layer called a residual block is introduced. In such layers, the input is added to the output of a slowly trained function. This allows a network to be built where each layer has a small number of weights, and each layer is trained with a new and improved regularization scheme. The result is a network that is very deep, very accurate, and very scalable.
DEEP ARCHITECTURES

FractalNet: Ultra-Deep Neural Networks without Residuals

Gustav Larsson
University of Chicago
larsson@uchicago.edu

Michael Malek
TUM
malek@tum.de

Gregory Shakhnarovich
TUM
greg@tum.de

Layer Key
- Generator
- 2D
- Img
- Poly

Fractal Expansion Rule

f(x)

f(x)

FractalNet:

Iteration #1

Iteration #2

Iteration #3

Iteration #4