GENERAL

- Problem Set 4 Due Tonight!

- Problem Set 5 has been posted and is ready to clone.
  ■ Due after thanksgiving, on Dec 3rd!

- We are pushing back the project report deadline by two days: Dec 10th.
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.

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 $\hat{x}$ from $x$. VGG-16’s pool5, fc1, fc2, features can be the $\hat{x}$ 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

**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
Treat *segmentation* as "Classification" of a patch treated as an image.

The problem: Lots of patches (one for each pixel).

Same approach: Pre-train on imagenet, use activations or finetune.

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

No need to repeat computations.
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, every time 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).

c3 c3s2 c3 c3s2 c3 c3 c3d2 c3d2 c3d4
SEMANTIC SEGMENTATION

DEEP ARCHITECTURES

- 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

Deeply-Supervised Nets
Deep Residual Learning for Image Recognition

Kaiming He  Xiangyu Zhang  Shaoqing Ren  Jian Sun
Microsoft Research
{khe, x-zhang, v-shren, jiansun}@microsoft.com

\[
\mathcal{F}(x) + x
\]

Figure 2. Residual learning: a building block.

Densely Connected Convolutional Networks

Gao Huang*  Zhang Liu*  Kilian Q. Weinberger
Cornell University  Tsinghua University  Cornell University
yuhuangg@cornell.edu  jliu@tsinghua.edu.cn  kqweinber@cornell.edu

Laurens van der Maaten
Facebook AI Research
laurens@fb.com

Abstract

Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train than previously considered models. This is mainly due to the use of so-called “residual connections” that allow the model to learn residual functions with reference to the layers closer to the input and those close to the output. In this paper we introduce the DenseNet architecture, which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with \( k \) layers have \( \mathcal{O}(k) \) parameterizations, our DenseNet has \( \mathcal{O}(k^2) \) such parameters. This comes at essentially no added cost, and results in networks that are more robust to degradations.

\[
\text{Figure B}: A 5-layer dense block with a growth rate of \( k = 4 \). Each layer takes all preceding feature maps as input.

FRAC-\text{NET}: ULTRA-DEEP NEURAL NETWORKS WITHOUT RESIDUALS

Gustavo Larocco  Michael Maim  Gregory Shakhnarovich
University of Chicago  TTI Chicago  TTI Chicago
larocco@cs.uchicago.edu  maim@ttic.edu  greg@ttic.edu

Layer Key
- Convolution
- Pool
- Add
- Residual

Figure B: A 5-layer dense block with a growth rate of \( k = 4 \). Each layer takes all preceding feature maps as input.