CSE 559A: Computer Vision

Fall 2018: T-R: 11:30-1pm @ Lopata 101

Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Course Staff: Zhihao Xia, Charlie Wu, Han Liu

http://www.cse.wustl.edu/~ayan/courses/cse559a/

November 12, 2018
• Problem Set 4 Due Tonight!

• Problem Set 5 will be posted shortly.
Rich feature hierarchies for accurate object detection and semantic segmentation
Tech report (v5)

Ross Girshick  Jeff Donahue  Trevor Darrell  Jitendra Malik
UC Berkeley

R-CNN: Regions with CNN features

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions
Carreira et al.: Constrained Parametric Min-Cuts, CVPR 2010

Figure 1. Our object segmentation framework. Segments are extracted around regularly placed foreground seeds, with various background seeds corresponding to image boundary edges, for all levels of foreground bias, which has the effect of producing segments with different scales. The resulting set of segments is filtered and ranked according to their plausibility of being good object hypotheses, based on mid-level properties.
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 upto 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.
TRANSFER LEARNING

- 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

Distribution over 1000 Imagenet classes
- Initialize Randomly
- Train on Imagenet
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
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
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 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 finetune.
The problem: Lots of patches (one for each pixel)
FULLY-CONVOLUTIONAL NETWORKS

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

No need to repeat computations.
Let's think in 1D. And assume we had two conv layers, and then a fully connected layer.

No need to repeat computations.

Crop

Full Image
FULLY-CONVOLUTIONAL NETWORKS

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

3x3 Conv

3x3 Conv
Stride 2
DILATED CONVOLUTION

3x3 Conv

3x3 Conv
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).

3x3 Dilated Conv
= 5x5 Conv with alternate weights fixed to 0

3x3 Conv

c3  c3s2  c3  c3s2  c3  

c3  c3  c3d2  c3d2  c3d4
DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs

Liang-Chieh Chen, George Papandreou, Senior Member, IEEE, Iasonas Kokkinos, Member, IEEE, Kevin Murphy, and Alan L. Yuille, Fellow, IEEE
SEMANTIC SEGMENTATION

(a) Image
(b) Before CRF
(c) After CRF
- 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 *
Dept. of EECS, UCSD
chl260@ucsd.edu

Saining Xie *
Dept. of CSE and CogSci, UCSD
s9xie@ucsd.edu

Patrick Gallagher
Dept. of CogSci, UCSD
rexaran@gmail.com

Zhengyou Zhang
Microsoft Research
zhang@microsoft.com

Zhuowen Tu †
Dept. of CogSci, UCSD
ztu@ucsd.edu
Deeply-Supervised Nets

- $z^{(m)}$: Feature maps
- $Q^{(m)}$: Responses
- $W^{(m)}$: Filters
- $Q^{(m)}$: Responses
- $z^{(m)}$: Feature maps
- $W^{(m)}$: Filters

Output
Deep Residual Learning for Image Recognition

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

Figure 2. Residual learning: a building block.
Deep Residual Learning for Image Rec

Kaiming He  Xiangyu Zhang  Shaoqing Ren
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Figure 2. Residual learning: a building block.
Densely Connected Convolutional Networks

Gao Huang*
Cornell University
gh349@cornell.edu

Zhuang Liu*
Tsinghua University
liuzhuang13@mails.tsinghua.edu.cn

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

Kilian Q. Weinberger
Cornell University
kpow@cornell.edu

Abstract

Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. In this paper, we embrace this observation and introduce the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with \( L \) layers have \( L \) connections—one between each layer and its subsequent layer—our network has \( \frac{L(L+1)}{2} \) direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of para-

Figure 1: A 5-layer dense block with a growth rate of \( k = 4 \). Each layer takes all preceding feature-maps as input.
FRACTALNET: Ultra-Deep Neural Networks without Residuals

Gustav Larsson
University of Chicago
larsson@cs.uchicago.edu

Michael Maire
TTI Chicago
mmaire@ttic.edu

Gregory Shakhnarovich
TTI Chicago
greg@ttic.edu

Fractal Expansion Rule

Layer Key
- Convolution
- Join
- Pool
- Prediction

$fc(z) \rightarrow f_{c+1}(z)$

$fc(z)$

$y$