Convnet architectures
Deep Learning for Vision

Greg Shakhnarovich
TTI-Chicago

December 23, 2015
Neural network review

- General ("fully connected") layer:

\[ z_L = h(W_L z_{L-1} + b_L) \]

where \( z_L \in \mathbb{R}^k \) (layer with \( k \) units),
\( W_L \in \mathbb{R}^{k \times m} \) (previous layer with \( m \) units),
\( b_L \in \mathbb{R}^k \)

- Locally connected: \( W_{L,j} \) has zeros everywhere except for localized receptive field (unit \( j \))

- Convolutional: the non-zero part of \( W_{L,j} \) is same for all \( j \)
Network topology

- Usually should think in terms of *layer* topology
- Important types of layer connectivity:
  - fully connected
  - locally connected
  - convolutional

  Receptive field of a unit: set of values in preceding layer that feed into it
  Feature map: set of values computed by units in a layer
Convolutional layers

- **Input to layer** $L$ is output of layer $L - 1$: $n \times n \times k_{L-1}$ tensor ($k_{L-1}$ units in previous conv layer)
- Each of the $k_L$ units in layer $L$: filter of size $f \times f \times k_{L-1}$
- **Output of layer** $L$: $n - f + 1 \times n - f + 1 \times k_L$ tensor
- Common technique: padding output of $L - 1$ with $(f - 1)/2$ values on four “spatial” sides
  - then output of $L$ has same spatial size as $L - 1$
- **Careful:** what do we pad with?
  - zero – assuming we have normalized the input to the network (e.g., subtract mean pixel value)
Suppose we have two units: \[ + \] and \[ \text{and} \]
We want to capture constellation in which the first pattern is to the left of the second pattern.
Naively, can construct a filter with depth two
But it won’t allow for invariance
Convnets and pooling

cue Andrej Karpathy’s slides from Stanford course
Resolution reduction

- Two sources of resolution reduction:
  1. Convolution with filter size $f$, no padding: reduce by $f - 1$
  2. Stride $s$: reduce by factor $s$

- Pooling with receptive field size $f$ and stride $s$: reduces resolution by factor $s$ ($f$ does not matter assuming it fits the input size)
Neural network’s natural tasks: classification and regression

Detection with sliding window: classification applied at many locations

Modern pipeline: only consider a fraction of possible window placements, found with some proposal generation mechanism (e.g., Selective Search)

Straightforward application of NN: treat each proposal (bounding box) as an image to be classified

ImageNet pretraining (dominant object centered in the image) matches this setup so can initialize well
Region CNN

- R-CNN by Girshick et al., 2013

**R-CNN: Regions with CNN features**

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

**Figure 1: Object detection system overview.** Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010.** For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class ILSVRC2013 detection dataset, R-CNN’s mAP is **31.4%**, a large improvement over OverFeat [34], which had the previous best result at 24.3%.
Data augmentation

- Typical forms of data augmentation for classification:
  - horizontal flip (but not vertical)
  - multiple crops
- R-CNN training: no need to crop, only flip
Hypercolumns

- Higher layers trade off localization for abstraction
- We may want to retain some lower-level information about input
- Idea: use features from multiple layers

Figure 1. The hypercolumn representation. The bottom image is the input, and above it are the feature maps of different layers in the CNN. The hypercolumn at a pixel is the vector of activations of all units that lie above that pixel.
Convnets for segmentation

- Need to classify each pixel
- Applying the entire network at each pixel does not sound reasonable
- The hypercolumn approach (Hariharan et al.): use proposals; classify a coarse grid of locations using each level; combine scores; interpolate.
Skip-layer connections

- Connections can skip layers
- Example: output depends directly on features from different levels in the hierarchy
- Backprop works as usual
- Loss signal flows to L2 along two paths: through L1 and directly from output
Fully convolutional networks

cue Berkeley FCN slides
Very, very deep networks

cue slides by Kaiming He from ICCV ImageNet/MSCOCO workshop