CSE 559A: Computer Vision

Fall 2017: T-R: 11:30-1pm @ Lopata 101
Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Staff: Abby Stylianou (abby@wustl.edu), Jarett Gross (jarett@wustl.edu)

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

Nov 22, 2017

GENERAL

- PSET 5 due Dec 1.
- Project Presentation Slides will be night of Dec 3rd at 11:59pm.
  - Slides must be exported to PDF. No embedded videos (email me if they're essential).
  - Each talk will be for 3 minutes + 1 minute for questions.
- Schedule to be posted later. Assume you'll be giving talk on first day (Dec 5).

CORE SEMANTIC TASKS IN VISION

- Image "Classification"
- Object Detection
- Semantic Segmentation

Image Classification: Given an image, classify it as being one of N categories

Scene Type

Aquarium
**CORE SEMANTIC TASKS IN VISION**

**Image Classification:** Given an image, classify it as being one of N categories

- Object Types:
  - Tiger
  - Mushroom
  - Cherry
  - Madagascar cat

But there might be more than one 'obvious' answer for an image

**Detection:** Check if an object exists in an image, and if so, draw a box around it.

But a box is too coarse...
**CORE SEMANTIC TASKS IN VISION**

**Semantic Segmentation:** Label the class of each pixel.

![Image of motorcycle](image1.png)
![Semantic Segmentation of motorcycle](image2.png)

**Variant:** Separate pixels of the same class into instances

**New Semantic Tasks: Many at the Intersection of Vision and Language**

![Image Captioning](image3.png)
![Visual Question Answering](image4.png)

Image Captioning (from COCO challenge)
Visual Question Answering (visualqa.org)
**CORE SEMANTIC TASKS IN VISION**

Semantic Tasks: Need to be evaluated (and trained) on data

Progress: Move from anecdotal to data

---

The PASCAL Visual Object Classes Challenge 2007

---

**CORE SEMANTIC TASKS IN VISION**

Semantic Tasks: Need to be evaluated (and trained) on data

- Huge effort to label a large number of images (using Amazon MTurk)
- Labels were from "Word-net" words
- It was thought that having a taxonomy of words would be helpful for visual semantics (turns out not).
- "Imagenet" Classification Challenge
  1000 classes x 1300 images per class = 1.3 Million Images

Changed the nature of ML methods.

---

**CORE SEMANTIC TASKS IN VISION**

Top-5 error rate on ImageNet

---

**CORE SEMANTIC TASKS IN VISION**

Top-5 error rate on ImageNet

---

All based on convolutional neural networks
**Architectures**

**Broad Design Principles**
- Think of a network that can "express" the operations that you think are needed to solve the problem
  - What kind of a "receptive" field should it have.
  - How non-linear does it need to be.
  - What should be the nature of the flow of information across the image.
- Make sure its a function you can actually learn.
  - Think of the flow of gradients.
  - Try to make other architectures that you know can be successfully trained as a starting point.
- One approach:
  - First find the biggest deepest network that will overfit the data
    (Given enough capacity, CNNs will often be able to just memorize the dataset)
  - Then scale it down so that it generalizes.

**Architectures**

Let's consider image classification.
- We will fix our input image to be a specific size.
  - Typically choose square images of size $S \times S$
  - Given image, resize proportionally so that smaller side (height or width) is $S$
  - Then take an $S$ crop along the other direction
  - (Sometime take multiple crops and average)
- The final output will be a $C$ dimensional vector for $C$ classes.
  - Train using soft-max cross entropy.
  - Classify using arg-max
- Often, you’ll hear about Top-K error.
  - How often is the true class in the top K highest values of predicted $C$ vector.

**Architectures**


**Reference**

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets"

Karen Simonyan & Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition."

Four kinds of layers:
- Convolutional
- Max-pooling
- Fully Connected
- Soft-max

**Architectures**

Convolutional Layers
Take a spatial input, produce a spatial output.

$$B \times H \times W \times C \Rightarrow B \times H' \times W' \times C'$$

Can also combine with down-sampling.

$$g[b, y, x, c_2] = \sum_{k_y} \sum_{k_x} \sum_{c_1} f[b, y + s, x + s, k_y, k_x, c_1] k[k_y, k_x, c_1, c_2]$$

Here, $s$ is stride.
- PSET 5 asks you to implement ‘valid convolution’
- But often combined with padding (just like in regular convolution)
**Question**: Input activation is $B \times H \times W \times C_1$, and I convolve it with a kernel of size $K \times K \times C_1 \times C_2$. What is the size of my output? Assume 'valid' convolution.

$B \times (H - K + 1) \times (W - K + 1) \times C_2$

**Question**: What if I do this with a stride of 2?

Downsample above by 2. Think of what happens when sizes are even or odd.

$B \times [(H - KY2) + 1] \times [(W - KY2) + 1] \times C_2$

In general, you want to pad such that $H - K$ and $W - K$ are even, so that you keep the right and bottom edge of your images.

---

**ARCHITECTURES**

**VGG-16**

Input is a 224x224x3 Image

- **Block 1**
  - 3x3 Conv (Pad 1): 3->64 + RELU (*pad 1 means on all sides, all conv layers have a "bias")
  - 3x3 Conv (Pad 1): 64->64 + RELU
  - 2x2 Max-Pool (Pad 0, Stride 2): 64->64

Input to Block 2 is 112x112x64 (called pool1)

- **Block 2**
  - 3x3 Conv (Pad 1): 64->128 + RELU
  - 3x3 Conv (Pad 1): 128->128 + RELU
  - 2x2 Max-Pool (Pad 0, Stride 2): 128->128

Input to Block 3 is 56x56x128 (called pool2)

- **Block 3**
  - 3x3 Conv (Pad 1): 128->256 + RELU
  - 3x3 Conv (Pad 1): 256->256 + RELU
  - 3x3 Conv (Pad 1): 256->256 + RELU
  - 2x2 Max-Pool (Pad 0, Stride 2): 256->256

Input to Block 4 is 28x28x256 (called pool3)

- **Block 4**
  - 3x3 Conv (Pad 1): 256->512 + RELU
  - 3x3 Conv (Pad 1): 512->512 + RELU
  - 3x3 Conv (Pad 1): 512->512 + RELU
  - 2x2 Max-Pool (Pad 0, Stride 2): 512->512

Input to Block 5 is 14x14x512 (called pool4)

- **Block 5**
  - 3x3 Conv (Pad 1): 512->512 + RELU
  - 3x3 Conv (Pad 1): 512->512 + RELU
  - 3x3 Conv (Pad 1): 512->512 + RELU
  - 2x2 Max-Pool (Pad 0, Stride 2): 512->512

Output of Block 5 is 7x7x512 (called pool5)

---

**Max-Pooling Layer**

$B \times H \times W \times C \Rightarrow B \times H' \times W' \times C$

$$g[b, y, x, c] = \max_{k,k'} f[b, y+s+k, x+s+k, c_1]$$

For each channel, choose the maximum value in a spatial neighborhood.

- What will the gradients of this look like?
- Motivated by intuition from traditional object recognition (deformable part models). Allows for some 'slack' in exact spatial location.
ARCHITECTURES

VGG-16

Output of Block 5 is 7x7x512 (called pool5)
- Reshape to a (49*512=25088) dimensional vector (or $8\times 8\times 512$)
- Fully connected (matmul + bias) 25088 -> 4096 + RELU
- Fully connected (matmul + bias) 4096 -> 4096 + RELU
- Fully connected (matmul + bias) 4096 -> 1000

This is the final output that is trained with a softmax + cross entropy.

- Lots of layers: 138 Million Parameters
- Compared to previous architectures, used really small conv filters.
  - This has now become standard.
  - Two 3x3 layers is "better" than a single 5x5 layer.
    - More non-linear
    - Fewer independent weights
- Train this with backprop!
- Back in the day, would take a week or more.

ARCHITECTURES

Regularization: Dropout

- Key Idea: Prevent a network from "depending" too much on the presence of a specific activation.
- So, randomly drop these values during training.
- $g = \text{Dropout}(f, p)$; $f$ and $g$ will have the same shape.

Different behavior during training and testing.

- Training
  - For each element $f_i$ of $f$, 
    - $g_i = 0$ with probability $p$
    - $g_i = \frac{f_i}{1-p}$ with probability $(1-p)$
- Testing: $g_i = f_i$

Dropout is a layer. You will backpropagate through it!

- Like RELU, but kills gradients based on what you selected in the forward pass.

ARCHITECTURES

Training VGG-16 by itself will work well, but soon begin to overfit
- Keep track of training and dev-set errors
  - Training errors will keep going down, but dev will saturate.
- Solution: Regularization

Weight Decay

- Add a squared or absolute value penalty on all weight values (for example, on each element of every convolutional kernel, matmul matrix) except biases. $\sum_i w_i^2$ or $\sum_i |w_i|$  
- Add this to your training loss with some factor. What is the effect?

ARCHITECTURES

VGG-16 trained with dropout at $p = 0.5$ on the first two fully connected layer outputs.
- Also use data augmentation
  - While training, randomly scale, horizontally flip, etc.
  - This gives us a network that does well on Imagenet classification
    - and ...

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.
Rich feature hierarchies for accurate object detection and semantic segmentation

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

Object Plausibility

 Parametric Min-Cuts

Degree of foreground bias

Ranking

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.
SEMANTIC SEGMENTATION

- Do the same? For each pixel, provide a window around it as input?
- Sounds expensive.

Next Time: "Fully Convolutional Networks"