Last block:

Classify an image into one of two classes by
- Stack some pooling, FC layers on an encoder
- Adjust result into a vector
- Pass to linear classifier

Learn all parameters with SGD and various losses
- get training examples right

Idea:
- exploit this machinery to improve training
- exploit this machinery to segment image

Producing image representations with classification

Examples:

Two class:

edges

Multi-class:

interest points

semantic segmentation

Edges

Training is tricky

No ”true” ground truth!

Berkeley segmentation data set:

get numerous people to mark boundaries in images

(a) original image (b) ground truth

Zoom on ground truth – multiple ground truths!

Using the data

Now the loss is the cross-entropy loss of Section 20.2.3 (equivalently, the log-likelihood of the data under this model). Write $y_i(\mathbf{x};\mathcal{I})$ for the value at location $\mathbf{x}$ in the i'th annotation of $\mathcal{I}$ (there may be more than one). Use the convention that

$$y_i(\mathbf{x},\mathcal{I}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is an edge point} \\ -1 & \text{otherwise} \end{cases}$$

Edge points using classification ideas

At a high level, straightforward

construct decoder to produce a score at each pixel

network parameters

$$s(\mathbf{x}, \mathcal{I}; \theta)$$

score

location in image

image

interpret the score as

$$s(\mathbf{x}, \mathcal{I}; \theta) = \log \frac{P(\mathbf{x} \text{ is edge} | \mathcal{I}, \theta)}{P(\mathbf{x} \text{ is not edge} | \mathcal{I}, \theta)}$$

Edge points using classification ideas

At a high level, straightforward

Loss at a particular location for a particular example

$$\begin{aligned}\mathcal{C}(\theta;\mathcal{I},i,\mathbf{x}) &= -\left(\frac{1+y_i(\mathbf{x};\mathcal{I})}{2}\right)\left[s(\mathbf{x};\mathcal{I},\theta)+\log\left(1+\exp s(\mathbf{x};\mathcal{I},\theta)\right)\right] \\ &\quad -\left(\frac{1-y_i(\mathbf{x};\mathcal{I})}{2}\right)\left[\log\left(1+\exp s(\mathbf{x};\mathcal{I},\theta)\right)\right]\end{aligned}$$

Loss for network:

sum over all locations of all markups of all examples

$$\mathcal{L}(\theta) = \sum_{\mathcal{I}\in \text{images}} \left[\sum_{i \in \text{ground truth}} \left[\sum_{\mathbf{x}\in \text{locations}} \left[\mathcal{C}(\theta;\mathcal{I},i,\mathbf{x})\right]\right]\right]$$

Works badly

Edge points are rare

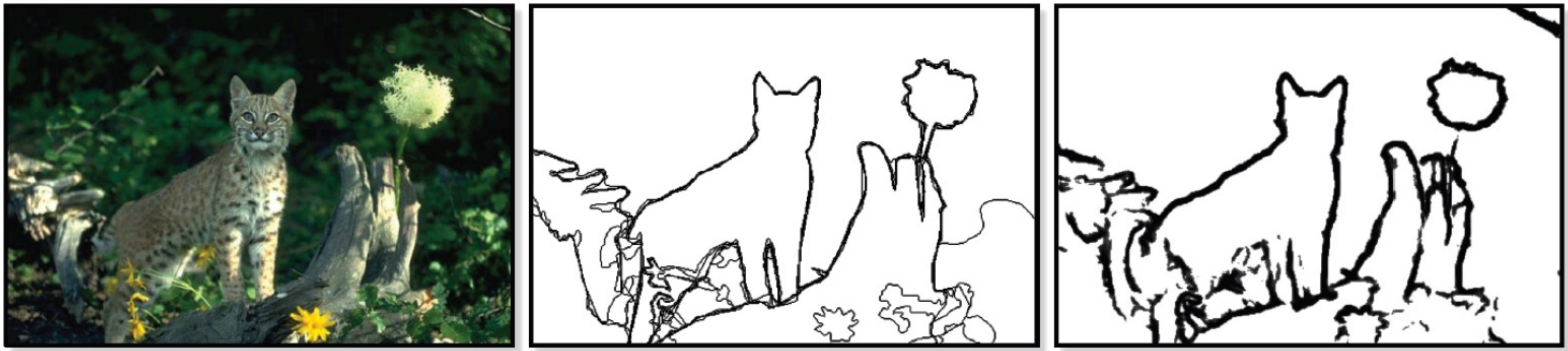
- it is hard to beat just saying there aren't any

Idea:

- reweight loss (high weight on edge points, low on non-edge)

effect is to introduce a weight to balance the classes. Write T for the total number of pixels in the annotated images (so if there are 3 annotations for an $M \times N$ training image, you count 3 M N pixels) and E for the total number of edge points in the annotated images. Then $\beta = E/T$ is the total fraction of edge points in the annotated training images. Then the reweighted loss is for the i'th value at location $\mathbf{x}$ in image $\mathcal{I}$ is

$$\begin{aligned}\mathcal{C}(\theta; \beta, \mathcal{I}, i, \mathbf{x}) &= -(1-\beta)\left(\frac{1+y_i(\mathbf{x};\mathcal{I})}{2}\right)\left[s(\mathbf{x};\mathcal{I},\theta) + \log\left(1+\exp s(\mathbf{x};\mathcal{I},\theta)\right)\right] \\ &\quad -\beta\left(\frac{1-y_i(\mathbf{x};\mathcal{I})}{2}\right)\left[\log\left(1+\exp s(\mathbf{x};\mathcal{I},\theta)\right)\right]\end{aligned}$$



(a) original image (b) ground truth (c) HED: output

Works fine, but

- evaluation is quite tricky (notes)
  - essentially, putting an edge point close to the right location should have some value
  - scoring appropriately requires care