Edge detection by UNet

D.A. Forsyth,

University of Illinois at Urbana Champaign

Ideas:

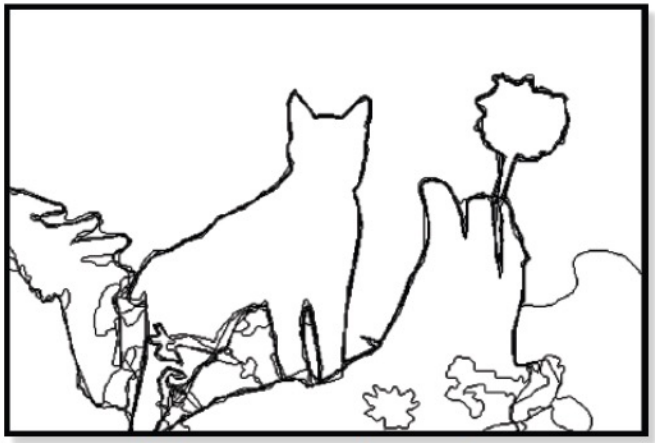
- Simple Image Classification:
  - Classify an image into one of two classes by
    - Stack some pooling, FC layers on an encoder
    - Adjust result into a vector
    - Pass to linear classifier
  - Learn all parameters with SGD and various losses
    - get training examples right

- Simple image regression:
  - Build U-Net
    - input is image
    - output is image-like thing (depth,normal, etc)
    - train to produce the right answer
- Merge these to produce class scores at each pixel
  - rather than a depth…

Edges

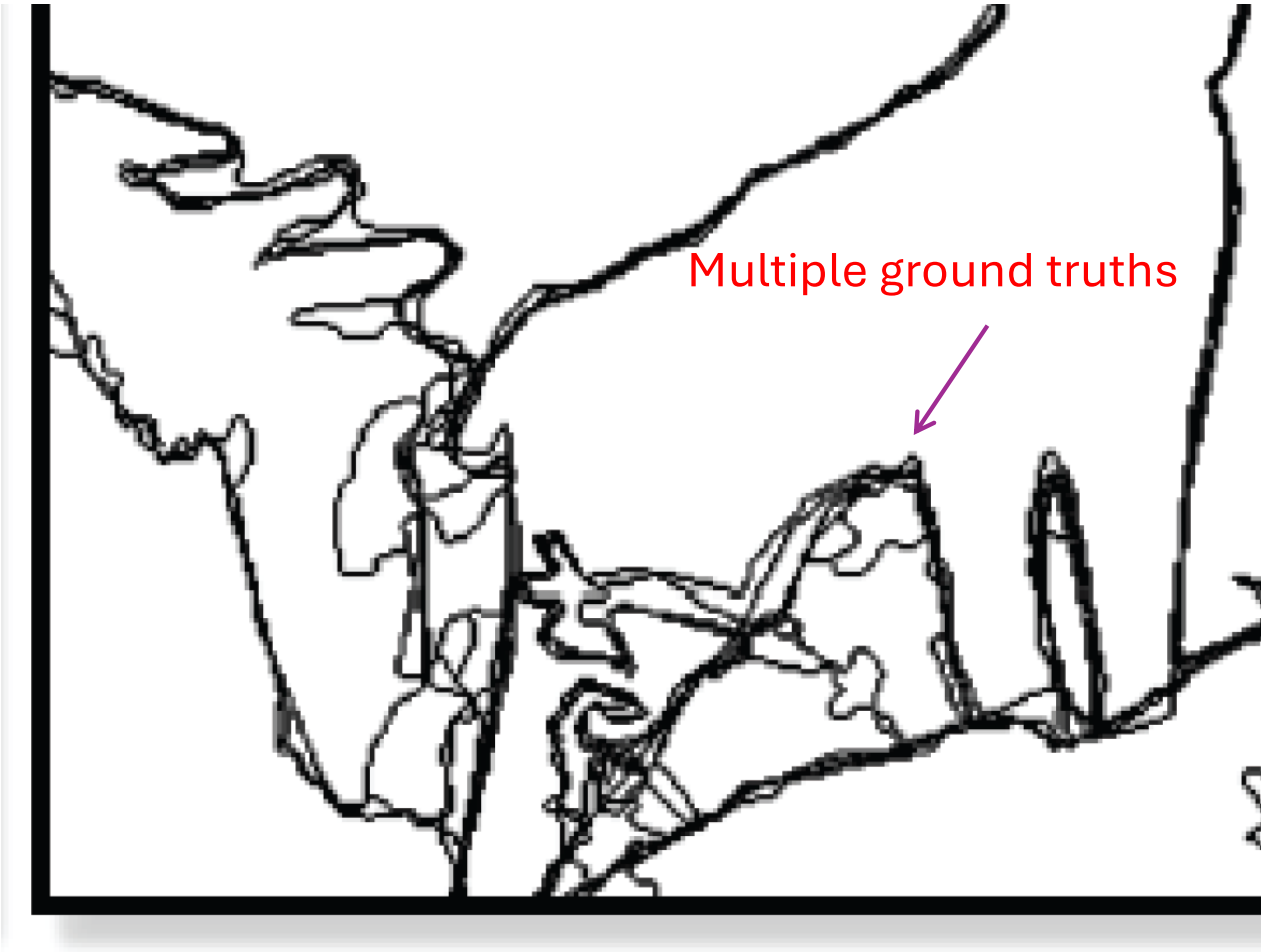
- Training is tricky
  - No ”true” ground truth!
- Berkeley segmentation data set:
  - get numerous people to mark boundaries in images

(a) original image



(b) ground truth

Zoom on ground truth



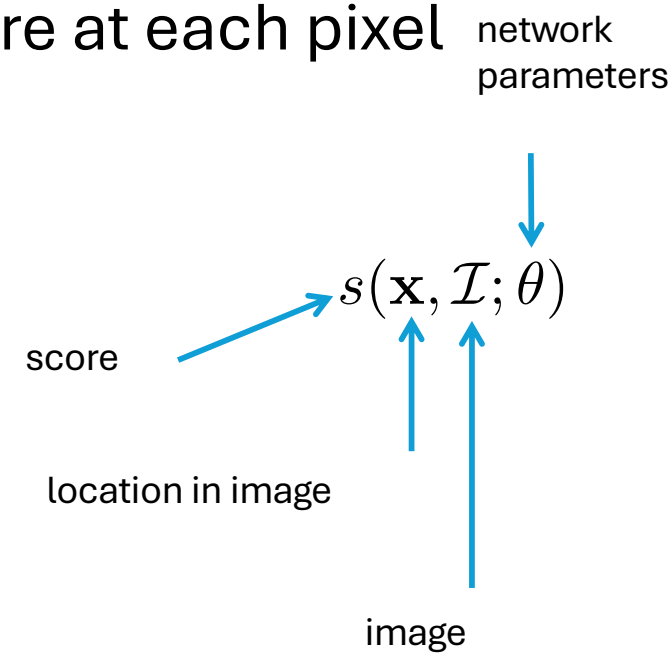
Using the data

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

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

Learning to classify edge points

- At a high level, straightforward
  - construct decoder to produce a score at each pixel



- interpret the score as

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

Learning to classify edge points

- At a high level, straightforward
    - Use cross entropy loss at each location for each example
- Careful:
    - examples are (I_i, y_i(x))
        - i.e. an image and an annotation
        - different examples might have the same image
            - cause different annotators mark up the same image
- Loss for network:
    - sum cross-entropy over all locations of all examples

This works very badly – predicts no-edge everywhere

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)

One way to deal with this 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

Weighted loss

is for the value at location $\mathbf{x}$ in image $\mathcal{I}_i$ is

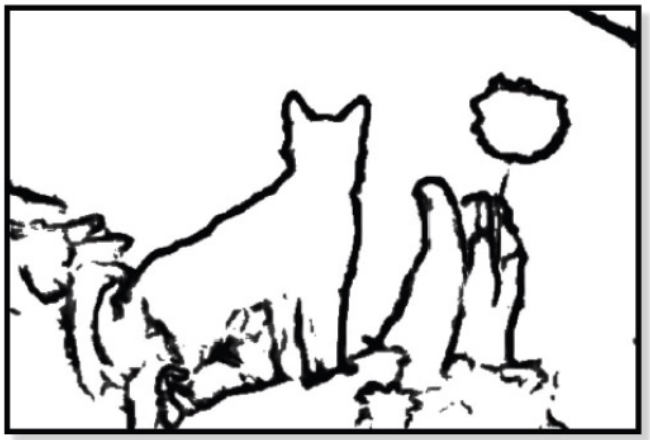
$$\begin{aligned}\mathcal{C}(\theta;\beta,\mathcal{I}_i,y_i(\mathbf{x}),\mathbf{x}) &= -(1-\beta)\left(\frac{1+y_i(\mathbf{x};\mathcal{I}_i)}{2}\right)[s(\mathbf{x};\mathcal{I}_i,\theta)+\log(1+\exp s(\mathbf{x};\mathcal{I}_i,\theta))] \\ &\quad -\beta\left(\frac{1-y_i(\mathbf{x};\mathcal{I}_i)}{2}\right)[\log(1+\exp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

The overall loss is then

$$\mathcal{L}(\theta)=\frac{1}{N}\sum_{i\in\text{training set}}\left[\frac{1}{N_p(i)}\sum_{\mathbf{x}\in\text{locations}}[\mathcal{C}(\theta;\beta,\mathcal{I}_i,y_i(\mathbf{x}),\mathbf{x})]\right].$$

(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

Think about this...

22.7. Estimate how big the class imbalance is in edge prediction. Equivalently, what fraction of pixels in a representative image are edge pixels.

22.8. Section 22.3.2 has: "You can construct the edge map of an image at any scale, which offers a training opportunity. The decoder produces a sequence of data blocks with different spatial dimensions. Each of these blocks could be decoded using a simple linear map to a predicted edge map at those spatial dimensions. " Explain.