Image Segmentation
Image Segmentation

Dividing the pixels of an image into separate groups representing different objects

- Traditionally, accomplished via clustering in color/space, graph-cut, edge detection, etc.

- Can train networks to perform segmentations that are driven more so by human perception/semantics

- Training examples:
  - Input: an image (domain consists of all the pixels’ RGB values)
  - Output: labeling of the pixels as to what grouping they correspond to
Output Labels

- True/False values
- Binary Segmentation
- E.g. 1=dog, 0=not dog
Output Labels

- Integers between 1 to n for n categories
- Multiclass Segmentation
- e.g. 1=cat, 2=dog, 3=human, 4=everything else

Source: COCO dataset
Output Labels

- Real numbers between 0 and 1
- Probabilistic Segmentation
- e.g. 1=tree branch, 0.8=most likely a branch, 0.2=most likely not a branch)
Segmenting Trees
Segmenting trees

- Difficult problem!
- Trees are large scale structures
- The images have limited pixel resolution of individual branches
- Branches severely occlude each other
- Even humans have a hard time ascertaining the correct topological structure from a single image/view
- Train networks to help...
Training Data

- Begin with a dataset of labels created by people
- Draw lines and thicknesses on top of branches, then flatten this information into a **binary mask** of the image
Training Data

- Not enough masks & too high a resolution: 4K (3840x2160 pixels)
- Train with many 512x512 crop pairs from original image/mask pair
- But during evaluation, use the full resolution image
Training a Network

- A network is a complex nonlinear function with many parameters
- Find function parameters such that the network function gives minimal error on the training data (i.e., minimize network loss)
- Network should predict the binary mask from the input image

Input image

Target labels

Network outputs

https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/
Results

- After training, use the resulting network function to produce binary masks of new images that were not previously hand-labeled.
- Some errors are obvious to the human eye:
  - false positives in the grass and on trees in the background, disconnected regions along twigs, etc.
Local Approximations
Observation

- Input images seem to be of two different types:
  - branches against grass
  - clusters of branches
Local Approximation

- Divide the training data into these two disparate groups
- Train a separate network on each group of data: separate function, separate parameters, etc.

- Given an input image, evaluate it separately on each network, and combine the two sets of results
- Use the results that make the most sense locally in the image
Combining Outputs

- Combining results:

| Input          | Output 1 | Output 2 | Combine clusters | Final Result |
|----------------|----------|----------|-----------------|--------------|-------------|


Combining Outputs

For each pixel $p$, we take a small crop $I(p)$ of the image centered at $p$.

- When we split the training crops into 2 clusters, we measured the **hue and saturation** of the crops and did **k-means clustering** on those measurements. We now perform the same measurements on $I(p)$.

- We find the distances of $I(p)$'s measurements to the 2 cluster’s centers, and interpolate the contributions of the 2 networks’ output at pixel $p$ correspondingly. The closer we are to a cluster center, the more weight the network is given.
More Results & Some Branch Estimation