Deep Networks and Edge Detection

- Deep Networks were first shown to be good for object classification. This is a task which they seem to be well suited for because it relies on non-local properties of image and the layers of the network extract features of the object that are increasingly invariant to its details. By contrast edge detection is a fine-scale task with attempts to detect local properties of the image (why doesn't the local details of the image get lost higher up the network?).
- As discussed in the previous lecture, the output of a deep network can be expressed in terms of compositions of elementary functions, where each elementary function depends on weights. This compositional structure can be exploited to calculate the derivatives of all the weights in the network, hence enabling an efficient learning algorithm. We illustrate this for a network with two layers of hidden variables (but the same procedure can be applied to networks with any numbers of layers).
- Formally, $\vec{O} = O(\vec{W}_3, \vec{H}_2)$, $\vec{H}_2 = H_2(\vec{W}_2, \vec{H}_1)$, $\vec{H}_1 = H_1(\vec{W}_1, \vec{I})$. A special case is $\vec{H}_2 = \mathrm{ReLu}(\vec{W}_2 \cdot \vec{H}_1)$ (etc). The input-output can be expressed as $\vec{O} = O(\vec{W}_3, \vec{W}_2, \vec{W}_1, \vec{I})$.
- The loss function takes the form $L(O(\vec{W}_3, \vec{W}_2, \vec{W}_1, \vec{I}), T)$, where T is the groundtruth.

Backpropagation: The Chain Rule of Differentiation

- The learning update rule – see previous lecture – requires computing the partial derivatives.
- For $\vec{W}_3$ the derivative is simple: $\frac{\partial L}{\partial \vec{W}_3} = \frac{\partial L}{\partial \vec{W}_2}, \frac{\partial L}{\partial \vec{W}_1}$. (because the weights $\vec{W}_3$ directly affect the output O so it easy to assign them credit).
- For $\vec{W}_2$, the derivative is given by $\frac{\partial L}{\partial \vec{W}_2} = \frac{\partial L}{\partial \vec{H}_2} \frac{\partial \vec{H}_2}{\partial W_2}$ with $\frac{\partial L}{\partial \vec{H}_2} = \frac{\partial L}{\partial O} \frac{\partial O}{\partial H_2}$. (This requires backpropagating information back to the earlier layers so to give credit to weights that do not directly affect the output).
- For $\vec{W}_1$, the derivative is given by $\frac{\partial L}{\partial \vec{W}_1} = \frac{\partial L}{\partial \vec{H}_1} \frac{\partial \vec{H}_1}{\partial W_1}$ with $\frac{\partial L}{\partial \vec{H}_1} = \frac{\partial L}{\partial \vec{H}_2} \frac{\partial O}{\partial H_1}$. (This requires backpropagating information back to the earlier layers so to give credit to weights that do not directly affect the output).
- These expressions yield the derivatives of the loss function with respect to the weights in terms of the derivatives of the compositional functions $\vec{O} = O(\vec{W}_3, \vec{H}_2)$, $\vec{H}_2 = H_2(\vec{W}_2, \vec{H}_1)$, $\vec{H}_1 = H_1(\vec{W}_1, \vec{I})$.

Edge Detection: HED

- The HED edge detector exploits the fact that edges occur at different scales in the image. Intuitively, we could detect edges at different scales by convolving the image with a Gaussian at different scales (scale corresponds to the standard deviation of the Gaussian) and taking the derivative. Small edges (i.e., small changes in intensity across the edge) will be blurred out by Gaussians with large variances and so will only give cues for edges at the smallest scales (e.g., texture edges, which HED does not want to detect). By contrast, large edges will remain as we blur the image with larger Gaussians. There will be information about edges at all scales (as exploited by statistical edge detectors).
- HED captures this intuition by having side layers which give evidence for edges at different levels of the hierarchy. These side layers have access to the ground truth and have a corresponding loss function. The side layers are combined together to yield the final decision which combines information from all scales. This enables the lower-layers, which have higher spatial resolution, to give accurate estimates for the local positions of edges while the higher-layers validate that there is an edge (HED wants to detect boundary edges which are large and which have evidence for them at most scales of the network).
- In summary, HED has a standard deep network architecture with side layers and a sum of loss functions for detecting edges at different scales.
- Note that, like statistical edge detection, the loss functions must stromgly penalize false negatives.

Edge Detection: Edge-Patches

- An alternative deep network design for edge detection requires first obtaining a dictionary of edge-patches (roughly one hundred). These capture the possible local structures of edges within an image region. One special patch contains no edges. These can be obtained from the groundtruth edge maps of images using clustering methods (similar to those for learning dictionaries of image patches).
- Edge detection can then be formulated as a classification task. The goal is to classify an image region by one of the edge-patches. This is similar to classification of objects (and was done before HED). It also must be biased to encourage edges to be detected.
- Note: edge-patches were used for an earlier non-neural-network edge detector which used alternative machine learning methods, like random forests, to make classification.

Symmetry-Axis Detection

- Symmetry axes have been studied as a way to break objects up into parts.They were originally studied for binary images. The intuitive idea of symmetry axes is to roll a ball inside the object so that it touches both sides of the object and its center sketches out the symmetry axis. When an object splits into parts, e.g., when two legs occur at the end of the torso, then the symmetry axis bifurcates (with one branch for each leg).
- Recently deep networks have been used to detect the symmetry axes of objects. Note that the scale of the symmetru axis corresponds to the distance between edges on either side of the object. Hence the symmetry axis of the leg has a smaller scale than the symmetry axis of the torso. Recall, as for edge detection, there is also evidence for edges at different scales in the image.
- Hence deep networks for symmetru axis detection have side structures, similar to HED, which estimate edges at different scales and also search for evidence for edges which are symmetric to each other. The symmetry axis and the scales/radii are output by the network by combining evidence for different layers of the network.