CS 194-129 Deep Neural Networks
Spring 2018 Canny Lecture 8

1 Lecture

Recap

Batch normalization is an effective means of regularizing and keeping the training process sane. It involves taking the mean and standard deviation of activations for some layer (within a minibatch), then normalizing. We then un-normalize (scale and shift) according to learned parameters. Among other things, this prevents overly exaggerated or large activations from happening.

In practice, the statistics of activations or gradients in minibatches are not really Gaussian and typically have very large, long-tailed values. So we occasionally get monster gradients. Batch normalization (or gradient clipping) deals with this problem.

Ensembles: for models that are snapshots of the same model, we can average the parameters and end up with a single usable model. The only extra overhead is that we have to keep a moving average of the parameter values (as opposed to having to keep 25 models in memory or something).

CNN Applications

CNNs can do more than classification. For instance, they can perform localization and detection – and by extension instance segmentation.

Classification + localization refers to the act of classifying an object and then identifying a bounding box for that object. We can do this by first training a network to classify, then adding a regression head (of fully connected layers) which will output the box coordinates. At this point we will have both the fully connected classification head and the fully connected regression head branching out of the convolutional portion of the network. We can now train the regression head on top of this model which is already optimized for classification – insinuating that the necessary information (features) for localization should already be encoded somewhere within the convolutional feature map.

For **human pose estimation**, there was a paper by Toshev and Szegedy (DeepPose) that demonstrated the ability of deep networks to solve the visual inverse kinematic problem. Using fairly simple convolutional networks, they were able to predict the positions of objects (e.g. the wrists) and thus entire poses (all the links and joints in a human body). In some ways, this was a simpler problem in that bodies always have the same number of joints and links – so in terms of the output of the network, we can say "this is the shoulder output, this is the left wrist output..." and so on. In other words, the topology is uniquely mapped onto the outputs of the network.

More specifically, in a 2D case the network might regress (x, y) coordinates for each joint during the last fully connected layer of something like AlexNet. There are also some more ideas that went into this, such as normalized coordinates and iterative refinement.

Overfeat: we can slide a window around the image. Then, for each position of the window, we can give a classification score for each class. Presumably, the window will be centered over each object for some displacement and will give a high score to the correct class for that particular displacement. We can then greedily merge boxes; if we have a cluster of adjacent high-scoring outputs for some class, it's very likely that they correspond to a single bounding box.

Note: we do this with multiple scales. Also, we can perform the operation efficiently using convolution.

Detection is a more difficult problem in which we have multiple objects within an image and we'd like to produce a classification label and a bounding box for each one. Since our images can have a lot of clutter (and therefore a lot of objects), we can end up with many numbers. So the network has to decide not only what to output, but also how much to output.

Histogram of oriented gradients (HOG): a custom feature. We can score objects using these low-level features at multiple scales, then use non-maximal suppression to produce a single label (even when there's a cluster of strong responses for the same object).

For detection – we would like to prune down the set of possible object positions before we even run the classifier. This is the **region proposal** subproblem, in which we try to identify "blobby" regions that are likely to contain objects. It is a class-agnostic object detector, one that looks for homogeneous "blobs."

R-CNN (2014) takes the classical idea of regional proposals and puts it together with a sliding convolutional network. Such a combination produces a very effective object detection algorithm. The region proposal part of it was a separate black box which, given an image, would produce regions that were then fed into a CNN. The CNN's job would be to come up with a bounding box and a label.

An R-CNN basically maps a pre-trained classifier across the image, targeting the specific bounding boxes produced by the region proposal network and then deciding whether to keep the results or not.

Faster R-CNN inserts a region proposal network (RPN) after the last convolutional layer, eliminating the need for external region proposals.

YOLO (You Only Look Once) detection attempts to merge region proposals into analysis of the image at each pixel. This is faster than Faster R-CNN, but not as good.