Here is some advice and clarifications about the semantic segmentation project

pierluigi.ferrari Oct '17

Here are a few things that I think are valuable to know if you're about to start this project (or if you are stuck).

Some advice for building the model:

1. If you read the paper, then you know that the original FCN-8s was trained in stages, not all at once. The authors later uploaded a version that was trained all at once to their GitHub repo, but that version has one important difference: The outputs of pooling layers 3 and 4 are scaled before they are fed into the 1x1 convolutions. This fact is completely omitted in the classroom and in the project walk-through. I trained my model with and without the scaling layers and found out that the model learns much better with the scaling layers included. It didn't converge that much faster, but I was able to reach much higher IoU and accuracy with the scaling layers. To include the scaling layers, simply add them to your model like so:

   pool3_out_scaled = tf.multiply(pool3_out, 0.0001, name='pool3_out_scaled')
   pool4_out_scaled = tf.multiply(pool4_out, 0.01, name='pool4_out_scaled')

   where `pool3_out` and `pool4_out` are the outputs of the VGG-16. You then feed the scaled outputs into your 1x1 convolutions and everything is as before from there. Note that the scaling factors are not the same for the two scaling layers. I took the scaling factors directly from the Prototxt of the original Caffe implementation.

2. When adding l2-regularization, realize that merely passing a regularizer in the arguments of the `tf.layers` is not enough. You also have to manually add all those regularization loss terms to your loss function, otherwise they are not doing anything. I didn't know this at first and thought I'm doing l2-regularization when I actually wasn't. See here for how to add the reg loss to your overall loss: https://stackoverflow.com/questions/46615623/do-we-need-to-add-the-regularization-loss-into-the-total-loss-in-tensorflow-mode

3. If TensorFlow's default variable initializer for `conv2d` and `conv2d_transpose` layers (which is Glorot uniform) doesn't work for you, try to change those initializers to truncated normal with a small standard deviation of 0.01 or even 0.001 instead. If you implemented l2-regularization correctly, you might not run into this problem though.

Furthermore, there are a few things Aaron says in the project walk-through video that are incorrect. They aren't big deals, but I think they should be clarified anyway.

Clarifications regarding the walk-through video:

1. Aaron mentions at one point in the video that the pretrained VGG-16 is frozen. That is not the case. The weights are not frozen. If you load the VGG and build layers on top of it and train the whole thing, all weights are trainable, not only the weights of the layers you added on top.
2. Aaron mentions that the model's output tensor needs to be reshaped into 2D because that is what TensorFlow's softmax function requires. That is incorrect. TensorFlow's softmax function (and tf.nn.softmax_cross_entropy_with_logits, for that matter) accepts tensors of any shape and will apply the softmax function on the last axis of the tensor. There is hence no need to reshape the output of the model, especially because you only end up reshaping it back into the image

dimensions after applying softmax anyway, so it’s just clock cycles wasted for your GPU. You should NOT reshape your model output. I don’t do it in my model and it works great 🙂

3. Aaron mentions that we need to add 1x1 convolutions on top of the VGG-16 network in order to preserve the spatial information of the data. That is incorrect, that is not the reason why we need to add 1x1 convs on top of the VGG-16. The pretrained VGG-16 model is already fully convolutionalized, i.e. it already contains the 1x1 convolutions that replace the fully connected layers. THOSE 1x1 convolutions are the ones that are used to preserve spatial information that would be lost if we kept the fully connected layers. The purpose of the 1x1 convolutions that we are adding on top of the VGG is merely to reduce the number of filters from 4096 to whatever the number of classes for our model is, that is all. This may sound pedantic, but I think it’s important to understand what is what.