Ex 07 & Ex 08: Guided Back Propogation

We'll look at implementing guided back propagation in tensorflow. Summary - it doesn't look as good as what we did before (i.e: guidedbackprop). Some differences:

Original work:

- Keras using theano
- pre-procecess images with log transofrm and grand mean subtraction
- classifying the two color shot - 4 outputs, just the lasing runs

Present work:

- Tensorflow
- Our own batch normalization
- classifying two outputs, from separate runs
- We may be learning some systematic run differences
- Not using a log transform
    - outlier speckle gets picked up in guided backprop

Issue

One uses the tf.gradients function. For some reason, it is producing nan's when it goes through my BatchNormalization ops, on Dense layers. It is fine on the convnet. Workaround, skip over those layers in backprop.

Code:

- Move data reading, one hot, confusion matrix into MLUtil.py, one data reader for lasing, and another for 2 color labels with 4 outputs.
    - Reading 2 color data is more complicated
        - convention is a label of -1 means don't use this sample for training
- Build models here, TFModel.py, notes:
    - SequentialModel now has a methods to keep track of regularization
    - build up sum of terms
    - Total loss: mean(batch model) + sum(variable - L2 L1)
        - should it be mean of variables?
        - I don't think so, this way adding more variables mean you work harder to minimize that term
        - move making the optimizer and final loss into SequentialModel
            - ex07 won't use this, still make loss directly
            - ex08, which will regularize over 2 color 4 outputs, will use it

Guided Back Prop

- start with logit that scored high for predicted label
    - don't use softmax, nonlinearity mixes in tyring to make the other logits small
- compute derivative of logit with respect to image
- Could do this with one call to tf.gradients, but for guided, and to work around batchnorm nan issue, go layer by layer
- Per paper: Striving for Simplicity: The All Convolutional Net, do the following
    - When doing backprop through a relu:
        - zero out negative values of gradient (don't propogate errors from above)
        - zero out gradient values where relu is 0 (don't propogate errors from below)
    - Still a little confused on these steps, why it works, another reference: github saliency maps Backpropagation.ipynb
- for plotting, don't import matplotlib unless gbprop command is given, best not to import matplotlib if running in batch

Running ex07 code

python ex07_tf_guided_backprop.py train

python ex07_tf_guided_backprop.py predict

python ex07_tf_guided_backprop.py gbprop

Running ex08 code

Getting an accurate model

- Need to use more data
- Bigger batch size, meaning need to use more memory
- Best to train on batch

Code: ex08_tf_4way_class.py

- The train function can be modified to pass 'test' to the getTrainData function. Otherwise reads all data (takes about a minute)
- The separate getTrainData function can be used from interactive ipython
- large batch size of 64, will use too much memory on interactive nodes, reduce for testing
- new function TFModel.build_2color_model
- We will print the total loss, which includes the regularization term, as well as the cross entropy
- For guided backprop, just print class 3 - both colors lased

Exercises

- compare guided backprop to unguided - just a straight derivative of image