Problem Set 3

Julian Vanecek

October 2018

1 Flowers

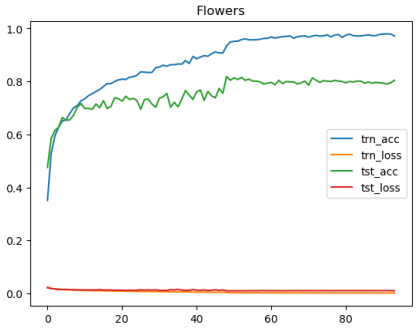

Figure 1: Flowers training

Flowers is using VGG architecture 64, 64, 'M', 128, 128, 'M', 256, 256, 'M', 512, 512, 'M' with batch normalization at every conv. layer no dropout, kernal sizes of 3, padding of 1, and no dropout. THe classification layer has 64 hidden nodes, dropout(0.5) and batch normalization. It reshuffles the train data at every epoch, uses an optimizer SGD with momentum 0.9, weight decay, and a learning rate decrease of gamma 0.1 at 50 and 75%. I'm using the class loader and all the class tricks. In the end, this model was a pain to train because I had a bunch of stupid bugs that should have thrown errors but didn't (a.k.a. one time a rounding error, and another my output softmax dimension off by one error, so straight off the bat accuracy+loss was screwed), that encouraged fruitless exploration of model space. I ran into memory out of bounds problems, especially for the larger VGG networks. I believe I solved these with wrapping my test_model() in a "with torch.no_grad()" clause. So after overcoming these, when the default class implementation worked I felt no reason to improve or innovate beyond my previous adventures. I'm not exactly sure on why the stepwise increase of feature dimensions in VGG would make intuitive sense, except that the step would refine the information for that many features. I guess one conv. layer would take a meta look at what features were extracted before expanding into a higher feature space, thereby improving the quality.

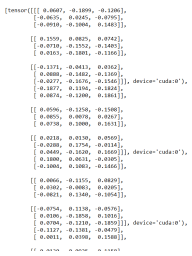

Figure 2: Some Flowers conv. weights (not zero)

1.1 Failures

To get a rep. of which flowers failed, I was having trouble with displaying the input flowers, because their RGB was scaled to a gaussian distribution (and rescaling was having problems), and showing the original was also difficult because the dataset was being shuffled at every epoch, so here's the best I could do: There is no obvious correlation of a single class being overly predicted or

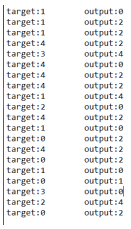

Figure 3: 20 failures of flower classification

misclassified.

2 Three Meter

I am not confident in my loss function being correct, or if the produced losses are off by some scalar. From Piazza L1 should work, if you divide it by the number of batches. I don't get good enough loss, but making the network larger or smaller in the number of layers and in the nodes in each layer, in order to expand or compress the data, doesn't improve the loss considerably, so I chose a smaller network. The optimizer is SGD which seems to work better than Adam. No weight decay, but Nesterov momentum. The learning rate decays at 50 and 75% again. You can see a bump in the training loss when the decay hits. The data is centered around a normal distribution. The model is [Relu(33), Relu(24), 16, Relu(16), Relu(24), 33]. I think a slow compression is standard. Looking at which inputs failed in the network, I couldn't find anything abnormal. Many have a few inputs which are 2 std dev away from mean, but that isn't a statistical abnormality with 33 input features.

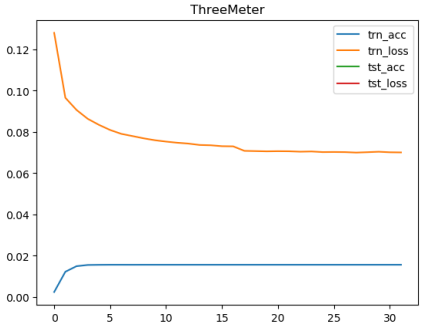

Figure 4: Three Meters training

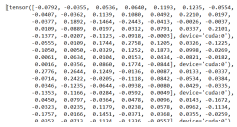

Figure 5: Some Three Meter weights (not zero)

3 Adult

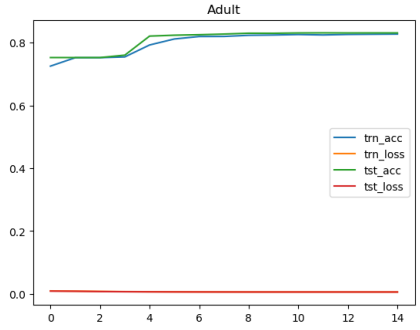

Figure 6: Adult Trainin

I use BCE because it works as a binary softmax if you will, and because it keeps the gradient high with a wrong classification, even though sigmoid gradient is usually zero, making it train faster. Anyways, I use a huge number of nodes, two hidden FC layers of 128 nodes, and a dropout of 0.2. I ended with a test accuracy of 0.83107 and a test loss of 0.00565 after 15 epochs.

3.1 Data

I mapped the features between zero and one. This worked well most of the time because most of the features were one hot encodings, which got mapped to 1, or something like age that exists as an int on a logical normal distribution, well I mapped that to [0,1] as well because, hey, close enough.

3.2 Weights

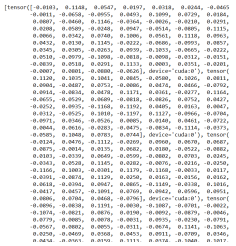

Figure 7: Some Adult weights (not zero)

3.3 Failures

The sum of the 20 failures makes

```
 8.3973,  12.7333,   0.7510,   1.2745,   9.2755,   1.0000,
 3.0000,  11.0000,   1.0000,   3.0000,   1.0000,   0.0000,
 0.0000,   0.0000,   0.0000,   0.0000,   0.0000,   1.0000,
 0.0000,   1.0000,   1.0000,   3.0000,   0.0000,   8.0000,
 2.0000,   0.0000,   1.0000,   3.0000,   2.0000,   0.0000,
14.0000,   0.0000,   4.0000,   0.0000,   0.0000,   1.0000,
 0.0000,   3.0000,   2.0000,   1.0000,   0.0000,   1.0000,
 1.0000,   0.0000,   3.0000,   0.0000,   5.0000,   0.0000,
 3.0000,  12.0000,   3.0000,   1.0000,   1.0000,   1.0000,
 2.0000,   0.0000,   0.0000,   3.0000,   0.0000,  17.0000,
 3.0000,  17.0000,  20.0000,   0.0000,   0.0000,   0.0000,
 0.0000
```

Figure 8: Adult misclassifications, summation of 20 vectors.

References