DLCV Kaggle Competition - Manikanda Krishnan V (Kaggle user name : QuantumMole, accuracy : 67.25%, rank 3)

21 February 2018 09:26

Data Augmentation

There was a class imbalance for classes 5,10 which was fixed by oversampling from these two classes.

The data was then augmented by choosing a random combination(0 or more) of the following :

- Image rotation by a multiple of 22.5 degrees
- Left right flipping
- Randomly cropping a 8x8 neighbourhood
- Adding gaussian/poisson noise

Network Design

I had used VGG 16 as base. The final convolutional layer of VGG 16 was then connected to naïve inception module containing 1x1,3x3,5x5 and 7x7 filters which was then connected to another 5x5 filter followed by max pooling which is then connected to a fully connected layer(I will refer to this as fc1 from hereafter) with 1024 neurons.

The primary classification unit is contains a fully connected layer with 2048 neurons which takes input from the fc1 layer that is then connected to the classification layer of 20 neurons corresponding to the 20 output classes.

```
final_layer1 = tf.stop_gradient(vgg_model.get_layer_tensors([12])[0])
params = [(256,(1,1)),(512,(3,3)),(512,(5,5)),(512,(7,7))]
final_layer = tf.nn.relu(batchnorm_layer(inception_module(final_layer1,params)))
classifier_1 = max_pool(tf.nn.relu(batchnorm_layer(conv2_layer(final_layer,512,(5,5),(1,1),1.0,0.0))),(2,2),(2,2))
classifier_2 = tf.nn.relu(batchnorm_layer(conv2_layer(classifier_1,512,(3,3),(1,1),1.0,0.0)))
classifier_flat = tf.reshape(classifier_2,(-1,512*16))
fc_1 = dropout_layer(tf.nn.relu(batchnorm_layer(full_layer(classifier_flat,512,None))),keep_prob)
fc_2 = dropout_layer(tf.nn.relu(batchnorm_layer(full_layer(fc_1,2048,None))),keep_prob)
output_classifier = full_layer(fc_2,20,tf.nn.relu)
output_probs = tf.nn.softmax(output_classifier)
```

The key component of this network is what I call the reclassification unit and it consists of 3 layer

- The first layer concatenates the output of all classification layers and connects them to a fully connected layer with 1024 neurons.
- This 1024 neurons is then concatenated with fc1 which is then connected a fc1 layer with 4096 neurons
- The 4096 neurons is then connected to the reclassification layer that contains 20 neurons each corresponding to the 20 output classes

2 such reclassification units are used in the network.

```
fc_3 = dropout_layer(tf.nn.relu(batchnorm_layer(full_layer(output_classifier,1024,None))),keep_prob)
fc3_concat = tf.concat([fc_3,fc_1],axis=1)
fc_4 = dropout_layer(tf.nn.relu(batchnorm_layer(full_layer(fc3_concat,4096,None))),keep_prob)
output_classifier1 = full_layer(fc_4,20,tf.nn.relu)
output_probs1 = tf.nn.softmax(output_classifier1)
fc5_concat = tf.concat([output_classifier1,output_classifier],axis=1)
fc_5 = dropout_layer(tf.nn.relu(batchnorm_layer(full_layer(fc5_concat,1024,None))),keep_prob)
fc6_concat = tf.concat([fc_5,fc_1],axis=1)
fc_6 = dropout_layer(tf.nn.relu(batchnorm_layer(full_layer(fc6_concat,4096,None))),keep_prob)
output_classifier2 = full_layer(fc_6,20,tf.nn.relu)
output_probs2 = tf.nn.softmax(output_classifier2)
```

All activation functions are ReLus

The final probabilities are the sum of softmax values of the relu classification and reclassification layers

The network doesn't fine tune the VGG16 pretrained model. The loss function corresponding to each classification layer(primary classification unit as well as reclassification units) is the sum of cross entropy and hinge loss. The final loss function is the product of the individual loss function which is then minimized using adam optimizer.

```
loss_classifier1 = tf.losses.softmax_cross_entropy(output_labels,output_classifier) + tf.losses.hinge_loss(output_labels,output_probs)
loss_classifier2 = tf.losses.softmax_cross_entropy(output_labels,output_classifier1) + tf.losses.hinge_loss(output_labels,output_probs1)
loss_classifier3 = tf.losses.softmax_cross_entropy(output_labels,output_classifier2) + tf.losses.hinge_loss(output_labels,output_probs2)
loss = loss_classifier1*loss_classifier2*loss_classifier3
train_step_class = tf.train.AdamOptimizer(1e-6).minimize(loss)
```

Ablation studies

The network without any reclassification units gives an accuracy of 62%.

The network with an single reclassification unit gives an accuracy of 64.3%

The network with two reclassification units gives an accuracy of 67%

The number of epochs used in the above studies is the same(15 epochs)