Report for the Deep Learning Course Assignment 2

Georgios Methenitis
georgios.methenitis@cwi.nl

Abstract

In this assignment I got familiar with TensorFlow. I created a single hidden layer network with ReLU activation function to learn features for the CIFAR10 dataset.

1 Task 1

1. TensorFlow *variables* are stored structures containing tensors, *constants* are constant variables in Tensorflow, *placeholders* are used in the place of variables at graph initialization when there are no variables. The differences are that constants cannot change their values in contrast with variables, and that placeholder are replacing variables at initialization phase.
2. There are two ways to declare variable in TensorFlow:
   - `var = tf.get_variable("var_name", shape=[..., ...])`
   - `var = tf.Variable(tf.zeros([..., ...]), name="var_name")`
3. The command `tf.shape(x)` returns the shape of the input of tensor `x` in an 1-D vector, while `x.get_shape()` return the size of the tensor.
4.
5.
6. A computational *graph*, *tensors* for the variables and *operator*.
7. In variable scopes, variables can be created, adopting the prefix of the scope in their name, variable scopes enable variable sharing in different places in the code. The differences between name scopes and variable scopes is in the former creates an hierarchy for operations and variables in the graph, while the latter helps mostly in variable sharing.
8. You can freeze a variable tensor during training by setting `trainable=True` upon creating the variable, this way it will not get updated during optimization.
9.
10.

2 Task 2

In this task I implemented the following functions in `mlp.py`, more specifically the `inference`, `loss`, and `accuracy`.

```
def inference(self, x):
  with tf.variable_scope('hidden', reuse=None):
    W = tf.get_variable("weights", shape=[x.get_shape()[1], self.n_hidden[0]],
     initializer=self.weight_initializer, regularizer=self.weight_regularizer)
    tf.histogram_summary("hidden_weights", W)
    b = tf.Variable(tf.zeros([self.n_hidden[0]]), name="bias")
    tf.histogram_summary("hidden_bias", b)
```

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

```
    with tf.variable_scope('output', reuse=None):
      w_out = tf.get_variable("weights", shape=[self.n_hidden[0], self.n_classes],
       initializer=self.weight_initializer, regularizer=self.weight_regularizer)
      tf.histogram_summary("output_weights", w_out)
      b_out = tf.Variable(tf.zeros([self.n_classes]), name="bias")
      tf.histogram_summary("output_bias", b_out)

    with tf.name_scope('relu_layer'):
      input = self.activation_fn(tf.matmul(x, W) + b)
    with tf.name_scope('linear_layer'):
      output = tf.matmul(input, w_out) + b_out

    return output

  def loss(self, logits, labels):

    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits, labels, name=None))
    with tf.variable_scope("hidden", reuse=True):
      loss += 0.5 * tf.reduce_sum(tf.get_variable("weights")**2)
    with tf.variable_scope("output", reuse=True):
      loss += 0.5 * tf.reduce_sum(tf.get_variable("weights")**2)

    return loss

  def accuracy(self, logits, labels):
    correct_prediction = tf.equal(tf.argmax(labels, 1), tf.argmax(logits, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    return accuracy
```

I also implemented the function `train` in the file `train_mlp.py`:

```
def train():
  # placeholder for input and output tensors allowing to create operations and the computation graph
  x = tf.placeholder(tf.float32, shape=[None, 3072])
  y_ = tf.placeholder(tf.float32, shape=[None, 10])

  # model, namescopes for summary
  model = MLP()
  x_ = model.inference(x)
  with tf.name_scope('loss'):
    loss = model.loss(x_, y_)
  with tf.name_scope('accuracy'):
    accuracy = model.accuracy(x_, y_)
  with tf.name_scope('train_step'):
    train_step = tf.train.GradientDescentOptimizer(LEARNING_RATE_DEFAULT).minimize(loss)

  # session
  init = tf.initialize_all_variables()
  sess = tf.Session()
  sess.run(init)

  # merged summary for the graph, the loss and accuracy
  tf.scalar_summary('accuracy', accuracy)
  tf.scalar_summary('loss', loss)
  tf.histogram_summary("logits", x_) # logits histogram
  merged = tf.merge_all_summaries()


  train_writer = tf.train.SummaryWriter(LOG_DIR_DEFAULT + '/train', sess.graph)
  test_writer = tf.train.SummaryWriter(LOG_DIR_DEFAULT + '/test', sess.graph)

  for i in range(1, MAX_STEPS_DEFAULT+1):
      batch_xs, batch_ys = cifar10.train.next_batch(BATCH_SIZE_DEFAULT)
      __, Summary, l, acc = sess.run([train_step, merged, loss, accuracy],
      feed_dict={x: batch_xs, y_: batch_ys})
```

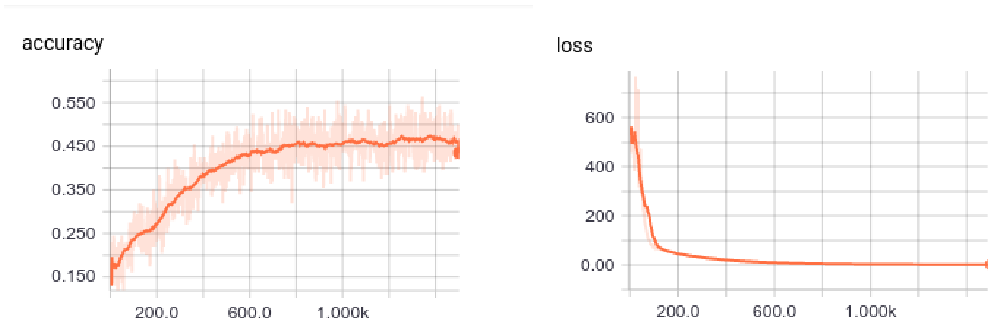

Figure 1: Recorded accuracy and loss for the training set.

Figure 2: Recorded accuracy and loss for the test set.

```
    train_writer.add_summary(Summary, i)

    if i % 100 == 0.0:
        batch_xs, batch_ys = cifar10.test.images, cifar10.test.labels
        __, Summary, l, acc = sess.run([train_step, merged, loss, accuracy],
         feed_dict={x: batch_xs, y_: batch_ys})
        test_writer.add_summary(Summary, i)

train_writer.close()
```

Figure 1 presents the accuracy and the loss in the training set recorded for every epoch.

Figure 2 presents the accuracy and the loss in the test set recorded for every 100-epochs.

Figure 3 presents the graph of the created model as this is illustrated by TensorBoard.

Figure 4 presents the histograms of the weights and the biases in every unit of the model, in the hidden and the output units respectively.

Figure 5 presents the histogram of the logits.

3 Conclusion

Should contain conclusion of this study. For example, you can try to answer the following questions. What was done during this assignment? What features of TensorBoard were positive and what were negative for implementing MLP model and performing the experiments? What are the main insights you got from the study of the MLP model on CIFAR10 dataset?

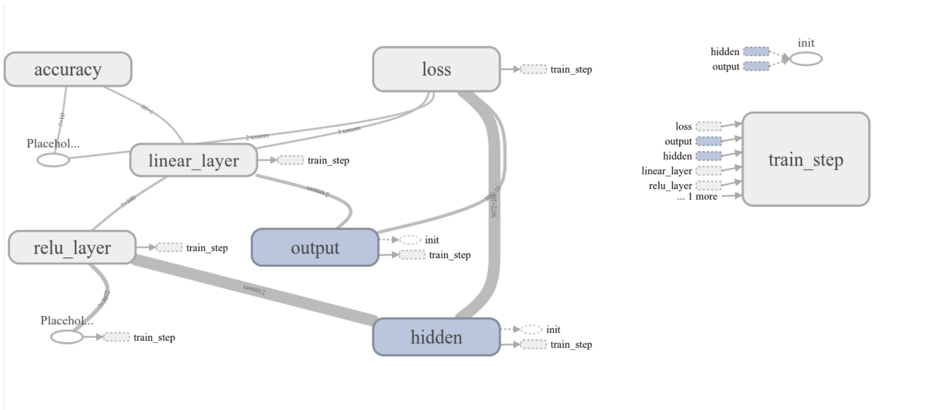

Figure 3: Graph of the model.

Figure 4: Histograms of weights and bias in the hidden layer, and weights and bias in the output linear layer.

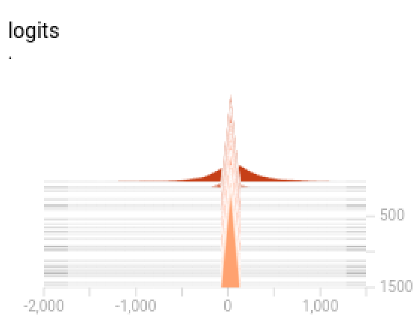

Figure 5: Histograms of logits.