Lab: Solution - Training the Feature Extractor

```
import pickle
import time
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle

from alexnet import AlexNet

nb_classes = 43
epochs = 10
batch_size = 128

with open('./train.p', 'rb') as f:
    data = pickle.load(f)

X_train, X_val, y_train, y_val = train_test_split(data['features'],
 data['labels'], test_size=0.33, random_state=0)

features = tf.placeholder(tf.float32, (None, 32, 32, 3))
labels = tf.placeholder(tf.int64, None)
resized = tf.image.resize_images(features, (227, 227))

# Returns the second final layer of the AlexNet model,
# this allows us to redo the last layer for the traffic signs
# model.
fc7 = AlexNet(resized, feature_extract=True)
fc7 = tf.stop_gradient(fc7)
shape = (fc7.get_shape().as_list()[-1], nb_classes)
fc8W = tf.Variable(tf.truncated_normal(shape, stddev=1e-2))
fc8b = tf.Variable(tf.zeros(nb_classes))
```

```
logits = tf.nn.xw_plus_b(fc7, fc8W, fc8b)

cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(logi
ts=logits, labels=labels)
loss_op = tf.reduce_mean(cross_entropy)
opt = tf.train.AdamOptimizer()
train_op = opt.minimize(loss_op, var_list=[fc8W, fc8b])
init_op = tf.initialize_all_variables()

preds = tf.arg_max(logits, 1)
accuracy_op = tf.reduce_mean(tf.cast(tf.equal(preds, labels), tf.fl
oat32))


def eval_on_data(X, y, sess):
    total_acc = 0
    total_loss = 0
    for offset in range(0, X.shape[0], batch_size):
        end = offset + batch_size
        X_batch = X[offset:end]
        y_batch = y[offset:end]

        loss, acc = sess.run([loss_op, accuracy_op], feed_dict={fea
tures: X_batch, labels: y_batch})
        total_loss += (loss * X_batch.shape[0])
        total_acc += (acc * X_batch.shape[0])

    return total_loss/X.shape[0], total_acc/X.shape[0]

with tf.Session() as sess:
    sess.run(init_op)

    for i in range(epochs):
        # training
```

```
            X_train, y_train = shuffle(X_train, y_train)
            t0 = time.time()
            for offset in range(0, X_train.shape[0], batch_size):
                end = offset + batch_size
                sess.run(train_op, feed_dict={features: X_train[offset:
    end], labels: y_train[offset:end]})

            val_loss, val_acc = eval_on_data(X_val, y_val, sess)
            print("Epoch", i+1)
            print("Time: %.3f seconds" % (time.time() - t0))
            print("Validation Loss =", val_loss)
            print("Validation Accuracy =", val_acc)
            print("")
```

Most of the code should look familiar.

```
cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(logi
ts, labels)
loss_op = tf.reduce_mean(cross_entropy)
opt = tf.train.AdamOptimizer()
train_op = opt.minimize(loss_op, var_list=[fc8W, fc8b])
init_op = tf.initialize_all_variables()

preds = tf.arg_max(logits, 1)
accuracy_op = tf.reduce_mean(tf.cast(tf.equal(preds, labels), tf.fl
oat32))
```

Operations are defined here (training, loss, accuracy, etc); `eval_on_data` is a utility function to calculate the loss and accuracy over a dataset to evaluate all at once.

```
with tf.Session() as sess:
```

```
    sess.run(init_op)

    for i in range(epochs):
        # training
        X_train, y_train = shuffle(X_train, y_train)
        t0 = time.time()
        for offset in range(0, X_train.shape[0], batch_size):
            end = offset + batch_size
            sess.run(train_op, feed_dict={features: X_train[offset:
end], labels: y_train[offset:end]})

        val_loss, val_acc = eval_on_data(X_val, y_val, sess)
        print("Epoch", i+1)
        print("Time: %.3f seconds" % (time.time() - t0))
        print("Validation Loss =", val_loss)
        print("Validation Accuracy =", val_acc)
        print("")
```

This is the main training procedure. As you can see we run `train_op` on each batch. Additionally, before each epoch the training set is shuffled using `shuffle`. At the end of each epoch the validation loss and accuracy are recorded and printed out.

Running the above code results in the following results after 10 epochs:

```
Epoch 10
Time: 53.402 seconds
Validation Loss = 0.126141663276
Validation Accuracy = 0.966069240196
```

NEXT