In [1]:

```
# Import libraries

from __future__ import print_function

import numpy as np

import sklearn

import pandas as pd

import tensorflow as tf

from tensorflow.contrib.tensor_forest.python import tensor_forest

from tensorflow.python.ops import resources


# Ignore all GPUs, tf random forest does not benefit from it.

import os

os.environ["CUDA_VISIBLE_DEVICES"] = ""
```

In [2]:

```
# Import data

data = pd.read_csv('data1.csv')
data.head()
```

Out[2]:

	TOTAL_SECONDS	SNIPPETS	THROUGH_PUT_ROWS	THROUGH_PUT_SIZE	Cluster
0	0	1	0	0	1
1	4	4	0	0	1
2	0	1	0	0	1
3	0	1	0	0	1
4	0	1	0	0	1

In [3]:

```

#Extract feature and target np arrays (inputs for placeholders)

input_x = data.iloc[:, 0:-1].values

input_y = data.iloc[:, -1].values

#input_x
#input_y
```

In [4]:

```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(input_x, input_y, test_size = 0
```

In [5]:

```
data1 = data.iloc[:,:].values

data1
```

Out[5]:

```
array([[ 0,  1,  0,  0,  1],
       [ 4,  4,  0,  0,  1],
       [ 0,  1,  0,  0,  1],
       ...,
       [ 1,  2,  1, 24,  1],
       [ 0,  2,  0,  0,  1],
       [ 0,  1,  0,  0,  1]])
```

In [6]:

```
# Parameters
num_steps = 500 # Total steps to train
num_classes = 6 # The 6 digits
num_features = 4 # features
num_trees = 12
max_nodes = 10
```

In [7]:

```
# Input and Target data
X = tf.placeholder(tf.float32, shape=[None, num_features])
# For random forest, labels must be integers (the class id)
Y = tf.placeholder(tf.int32, shape=[None])
```

In [8]:

```
# Random Forest Parameters
hparams = tensor_forest.ForestHParams(num_classes=num_classes,
                                      num_features=num_features,
                                      num_trees=num_trees,
                                      max_nodes=max_nodes).fill()
```

In [9]:

```
# Build the Random Forest

forest_graph = tensor_forest.RandomForestGraphs(hparams)
```

```
INFO:tensorflow:Constructing forest with params =
INFO:tensorflow:{'num_classes': 6, 'use_running_stats_method': False,
'dominate_fraction': 0.99, 'split_type': 0, 'split_finish_name': 'bas
ic', 'inference_tree_paths': False, 'num_splits_to_consider': 10, 'va
lid_leaf_threshold': 1, 'early_finish_check_every_samples': 0, 'featu
re_bagging_fraction': 1.0, 'regression': False, 'base_random_seed':
0, 'num_outputs': 1, 'prune_every_samples': 0, 'checkpoint_stats': Fa
lse, 'finish_type': 0, 'num_output_columns': 7, 'collate_examples': F
alse, 'dominate_method': 'bootstrap', 'num_trees': 12, 'split_name':
'less_or_equal', 'pruning_type': 0, 'leaf_model_type': 0, 'bagging_fr
action': 1.0, 'bagged_num_features': 4, 'param_file': None, 'bagged_f
eatures': None, 'split_pruning_name': 'none', 'max_fertile_nodes': 0,
'model_name': 'all_dense', 'split_after_samples': 250, 'num_feature
s': 4, 'stats_model_type': 0, 'max_nodes': 10, 'initialize_average_sp
lits': False}
```

In [10]:

```
# Get training graph and loss

train_op = forest_graph.training_graph(X, Y)

loss_op = forest_graph.training_loss(X, Y)
```

In [11]:

```
# Measure the accuracy

infer_op, _, _ = forest_graph.inference_graph(X)

correct_prediction = tf.equal(tf.argmax(infer_op, 1), tf.cast(Y, tf.int64))

accuracy_op = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

In [12]:

```
# Initialize the variables (i.e. assign their default value) and forest resources

init_vars = tf.group(tf.global_variables_initializer(),
                     resources.initialize_resources(resources.shared_resources()))
```

In [13]:

```


# Start TensorFlow session

sess = tf.Session()
```

In [14]:

```
# Run the initializer

sess.run(init_vars)
```

In [15]:

```
# Training

for i in range(1, num_steps + 1):

    _, l = sess.run([train_op, loss_op], feed_dict={X: X_train, Y: y_train})

    if i % 50 == 0 or i == 1:

        acc = sess.run(accuracy_op, feed_dict={X: X_train, Y: y_train})

        print('Step %i, Loss: %f, Acc: %f' % (i, l, acc))
```

```
Step 1, Loss: -1.000000, Acc: 0.984471
Step 50, Loss: -11.000000, Acc: 0.995132
Step 100, Loss: -11.000000, Acc: 0.995132
Step 150, Loss: -11.000000, Acc: 0.995132
Step 200, Loss: -11.000000, Acc: 0.995132
Step 250, Loss: -11.000000, Acc: 0.995132
Step 300, Loss: -11.000000, Acc: 0.995132
Step 350, Loss: -11.000000, Acc: 0.995132
Step 400, Loss: -11.000000, Acc: 0.995132
Step 450, Loss: -11.000000, Acc: 0.995132
Step 500, Loss: -11.000000, Acc: 0.995132
```

In [16]:

```
# Test Model

print("Test Accuracy:", sess.run(accuracy_op, feed_dict={X: X_test, Y: y_test}))
```

```
Test Accuracy: 0.9959207
```

In [ ]: