In [1]:
```
from __future__ import print_function

import tensorflow as tf
import os
```

In [2]:
```
# Dataset Parameters - CHANGE HERE
MODE = 'folder' # or 'file', if you choose a plain text file (see above).
DATASET_PATH = 'C:/Users/tom13/jupyter/FYP/food-101/train' # the dataset file or root fo

# Image Parameters
N_CLASSES = 101 # CHANGE HERE, total number of classes
IMG_HEIGHT = 128 # CHANGE HERE, the image height to be resized to
IMG_WIDTH = 128 # CHANGE HERE, the image width to be resized to
CHANNELS = 3 # The 3 color channels, change to 1 if grayscale
```

In [3]:
```
# Reading the dataset
# 2 modes: 'file' or 'folder'
def read_images(dataset_path, mode, batch_size):
    imagepaths, labels = list(), list()
    if mode == 'file':
        # Read dataset file
        data = open(dataset_path, 'r').read().splitlines()
        for d in data:
            imagepaths.append(d.split(' ')[0])
            labels.append(int(d.split(' ')[1]))
    elif mode == 'folder':
        # An ID will be affected to each sub-folders by alphabetical order
        label = 0
        # List the directory
        try:  # Python 2
            classes = sorted(os.walk(dataset_path).next()[1])
        except Exception:  # Python 3
            classes = sorted(os.walk(dataset_path).__next__()[1])
        # List each sub-directory (the classes)
        for c in classes:
            c_dir = os.path.join(dataset_path, c)
            try:  # Python 2
                walk = os.walk(c_dir).next()
            except Exception:  # Python 3
                walk = os.walk(c_dir).__next__()
            # Add each image to the training set
```

```
                for sample in walk[2]:
                    # Only keeps jpeg images
                    if sample.endswith('.jpg') or sample.endswith('.jpeg'):
                        imagepaths.append(os.path.join(c_dir, sample))
                        labels.append(label)
                label += 1
        else:
            raise Exception("Unknown mode.")

        # Convert to Tensor
        imagepaths = tf.convert_to_tensor(imagepaths, dtype=tf.string)
        labels = tf.convert_to_tensor(labels, dtype=tf.int32)
        # Build a TF Queue, shuffle data
        image, label = tf.train.slice_input_producer([imagepaths, labels],
                                                      shuffle=True)

        # Read images from disk
        image = tf.read_file(image)
        image = tf.image.decode_jpeg(image, channels=CHANNELS)

        # Resize images to a common size
        image = tf.image.resize_images(image, [IMG_HEIGHT, IMG_WIDTH])

        # Normalize
        image = image * 1.0/127.5 - 1.0

        # Create batches
        X, Y = tf.train.batch([image, label], batch_size=batch_size,
                              capacity=batch_size * 8,
                              num_threads=4)

        return X, Y
```

In [4]:
```
# Parameters
learning_rate = 0.001
num_steps = 5000
batch_size = 32
display_step = 500

# Network Parameters
dropout = 0.75 # Dropout, probability to keep units

# Build the data input
X, Y = read_images(DATASET_PATH, MODE, batch_size)
```

In [5]:
```
# Create model
def conv_net(x, n_classes, dropout, reuse, is_training):
    # Define a scope for reusing the variables
```

```python
        with tf.variable_scope('ConvNet', reuse=reuse):

            # Convolution Layer with 32 filters and a kernel size of 5
            conv1 = tf.layers.conv2d(x, 32, 5, activation=tf.nn.relu)
            # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
            conv1 = tf.layers.max_pooling2d(conv1, 2, 2)

            # Convolution Layer with 32 filters and a kernel size of 5
            conv2 = tf.layers.conv2d(conv1, 64, 3, activation=tf.nn.relu)
            # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
            conv2 = tf.layers.max_pooling2d(conv2, 2, 2)

            # Convolution Layer with 32 filters and a kernel size of 5
            conv3 = tf.layers.conv2d(conv2, 64, 3, activation=tf.nn.relu)
            # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
            conv3 = tf.layers.max_pooling2d(conv3, 2, 2)

            # Flatten the data to a 1-D vector for the fully connected layer
            fc1 = tf.contrib.layers.flatten(conv2)

            # Fully connected layer (in contrib folder for now)
            fc1 = tf.layers.dense(fc1, 1024)
            # Apply Dropout (if is_training is False, dropout is not applied)
            fc1 = tf.layers.dropout(fc1, rate=dropout, training=is_training)

            # Output layer, class prediction
            out = tf.layers.dense(fc1, n_classes)
            # Because 'softmax_cross_entropy_with_logits' already apply softmax,
            # we only apply softmax to testing network
            out = tf.nn.softmax(out) if not is_training else out

        return out
```

In [6]:
```python
# Create a graph for training
logits_train = conv_net(X, N_CLASSES, dropout, reuse=False, is_training=True)
# Create another graph for testing that reuse the same weights
logits_test = conv_net(X, N_CLASSES, dropout, reuse=True, is_training=False)
```

In [7]:
```python
# Define loss and optimizer (with train logits, for dropout to take effect)
loss_op = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(
    logits=logits_train, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
train_op = optimizer.minimize(loss_op)
```

In [8]:
```python
# Evaluate model (with test logits, for dropout to be disabled)
correct_pred = tf.equal(tf.argmax(logits_test, 1), tf.cast(Y, tf.int64))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
```

In [9]:
```python
# Initialize the variables (i.e. assign their default value)
init = tf.global_variables_initializer()
```

In [10]:
```python
# Saver object
saver = tf.train.Saver()
```

In [11]:
```python
# Start training
with tf.Session() as sess:

    # Run the initializer
    sess.run(init)

    # Start the data queue
    tf.train.start_queue_runners()

    # Training cycle
    for step in range(1, num_steps+1):

        if step % display_step == 0:
            # Run optimization and calculate batch loss and accuracy
            _, loss, acc = sess.run([train_op, loss_op, accuracy])
            print('Step')
            print("Step " + str(step) + ", Minibatch Loss= " + \
                  "{:.4f}".format(loss) + ", Training Accuracy= " + \
                  "{:.3f}".format(acc))
        else:
            # Only run the optimization op (backprop)
            sess.run(train_op)

    print("Optimization Finished!")

    # Save your model
    #saver.save(sess, 'my_tf_model')
```

```
Step
Step 500, Minibatch Loss= 4.6192, Training Accuracy= 0.031
Step
Step 1000, Minibatch Loss= 4.6240, Training Accuracy= 0.031
Step
Step 1500, Minibatch Loss= 4.5927, Training Accuracy= 0.031
Step
Step 2000, Minibatch Loss= 4.5935, Training Accuracy= 0.000
Step
Step 2500, Minibatch Loss= 4.6218, Training Accuracy= 0.000
Step
Step 3000, Minibatch Loss= 4.6296, Training Accuracy= 0.000
Step
Step 3500, Minibatch Loss= 4.4787, Training Accuracy= 0.031
Step
Step 4000, Minibatch Loss= 4.3978, Training Accuracy= 0.000
Step
```

```
Step 4500, Minibatch Loss= 4.3245, Training Accuracy= 0.062
Step
Step 5000, Minibatch Loss= 3.8302, Training Accuracy= 0.188
Optimization Finished!
ERROR:tensorflow:Exception in QueueRunner: Run call was cancelled
ERROR:tensorflow:Exception in QueueRunner: Enqueue operation was cancelled
         [[Node: batch/fifo_queue_enqueue = QueueEnqueueV2[Tcomponents=[DT_FLOAT, DT_INT32], tim
ERROR:tensorflow:Exception in QueueRunner: Enqueue operation was cancelled
         [[Node: batch/fifo_queue_enqueue = QueueEnqueueV2[Tcomponents=[DT_FLOAT, DT_INT32], tim
ERROR:tensorflow:Exception in QueueRunner: Enqueue operation was cancelled
         [[Node: input_producer/input_producer/input_producer_EnqueueMany = QueueEnqueueManyV2[T
ERROR:tensorflow:Exception in QueueRunner: Enqueue operation was cancelled
         [[Node: batch/fifo_queue_enqueue = QueueEnqueueV2[Tcomponents=[DT_FLOAT, DT_INT32], tim


Exception in thread QueueRunnerThread-batch/fifo_queue-batch/fifo_queue_enqueue:
Traceback (most recent call last):
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 914
    self.run()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 862
    self._target(*self._args, **self._kwargs)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    enqueue_callable()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    target_list_as_strings, status, None)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    c_api.TF_GetCode(self.status.status))
tensorflow.python.framework.errors_impl.CancelledError: Enqueue operation was cancelled
         [[Node: batch/fifo_queue_enqueue = QueueEnqueueV2[Tcomponents=[DT_FLOAT, DT_INT32], tim

Exception in thread QueueRunnerThread-batch/fifo_queue-batch/fifo_queue_enqueue:
Traceback (most recent call last):
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 914
    self.run()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 862
    self._target(*self._args, **self._kwargs)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    enqueue_callable()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    target_list_as_strings, status, None)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    c_api.TF_GetCode(self.status.status))
tensorflow.python.framework.errors_impl.CancelledError: Enqueue operation was cancelled
         [[Node: batch/fifo_queue_enqueue = QueueEnqueueV2[Tcomponents=[DT_FLOAT, DT_INT32], tim

Exception in thread QueueRunnerThread-batch/fifo_queue-batch/fifo_queue_enqueue:
Traceback (most recent call last):
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 914
```

```
    self.run()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 862
    self._target(*self._args, **self._kwargs)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    enqueue_callable()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    target_list_as_strings, status, None)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    c_api.TF_GetCode(self.status.status))
tensorflow.python.framework.errors_impl.CancelledError: Run call was cancelled

Exception in thread QueueRunnerThread-input_producer/input_producer-input_producer/input_produce
Traceback (most recent call last):
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 914
    self.run()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 862
    self._target(*self._args, **self._kwargs)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    enqueue_callable()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    target_list_as_strings, status, None)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    c_api.TF_GetCode(self.status.status))
tensorflow.python.framework.errors_impl.CancelledError: Enqueue operation was cancelled
         [[Node: input_producer/input_producer/input_producer_EnqueueMany = QueueEnqueueManyV2[T

Exception in thread QueueRunnerThread-batch/fifo_queue-batch/fifo_queue_enqueue:
Traceback (most recent call last):
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 914
    self.run()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\threading.py", line 862
    self._target(*self._args, **self._kwargs)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    enqueue_callable()
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    target_list_as_strings, status, None)
  File "c:\users\tom13\appdata\local\conda\conda\envs\tensorflow-gpu\lib\site-packages\tensorflo
    c_api.TF_GetCode(self.status.status))
tensorflow.python.framework.errors_impl.CancelledError: Enqueue operation was cancelled
         [[Node: batch/fifo_queue_enqueue = QueueEnqueueV2[Tcomponents=[DT_FLOAT, DT_INT32], tim
```