SMOKE TEST DOCUMENT

TensorFlow Smoke Test Cases

Date Prepared: July 2019

Document Information

Project Name	TensorFlow Smoke Test Document		
Project Owner		**Document Version No**	1.0
Quality Review Method	By email/HP SharePoint		
Prepared By		**Preparation Date**	July 2019
Reviewed By	Refer to version history	**Review Date**	

Table of Contents

1 TEST USING JUPYTERHUB

1.1 Using Linear Regression

Reference:

https://github.com/aymericdamien/TensorFlowExamples/blob/master/notebooks/2_BasicModels/linear_regression.ipynb

Start python3 and execute the below code

```
import tensorflow as tf
import numpy
import matplotlib.pyplot as plt
rng = numpy.random
```

```
# Parameters
learning_rate = 0.01
training_epochs = 1000
display_step = 50
```

```
# Training Data
train_X = numpy.asarray([3.3,4.4,5.5,6.71,6.93,4.168,9.779,6.182,7.59,2.167,
                    7.042,10.791,5.313,7.997,5.654,9.27,3.1])
train_Y = numpy.asarray([1.7,2.76,2.09,3.19,1.694,1.573,3.366,2.596,2.53,1.221,
                       2.827,3.465,1.65,2.904,2.42,2.94,1.3])
n_samples = train_X.shape[0]
```

```
# tf Graph Input
X = tf.placeholder("float")
Y = tf.placeholder("float")

# Set model weights
W = tf.Variable(rng.randn(), name="weight")
b = tf.Variable(rng.randn(), name="bias")
```

```
# Construct a linear model
pred = tf.add(tf.multiply(X, W), b)
```

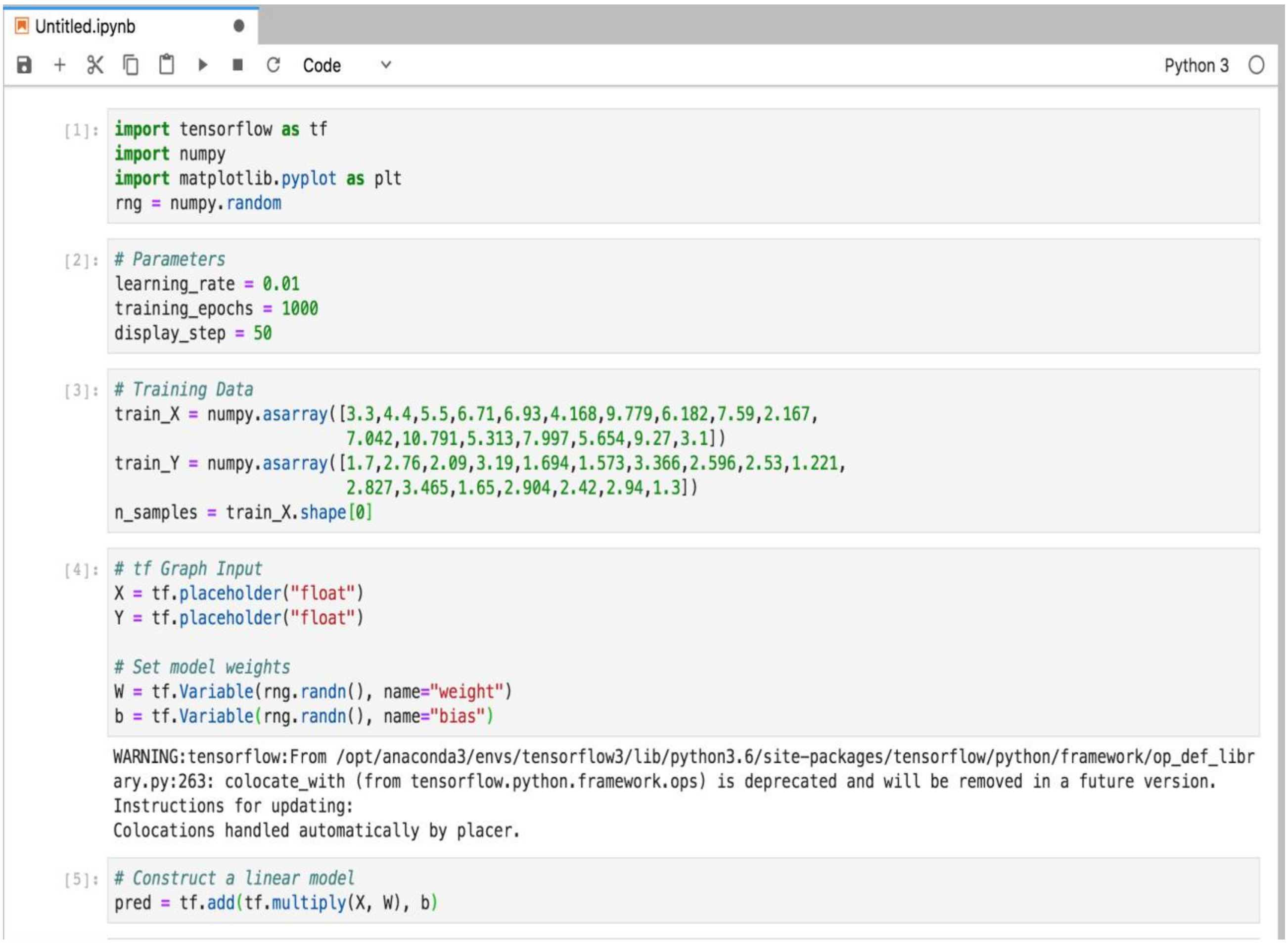
```
# Mean squared error
cost = tf.reduce_sum(tf.pow(pred-Y, 2))/(2*n_samples)
# Gradient descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

```
# Initialize the variables (i.e. assign their default value)
init = tf.global_variables_initializer()
```

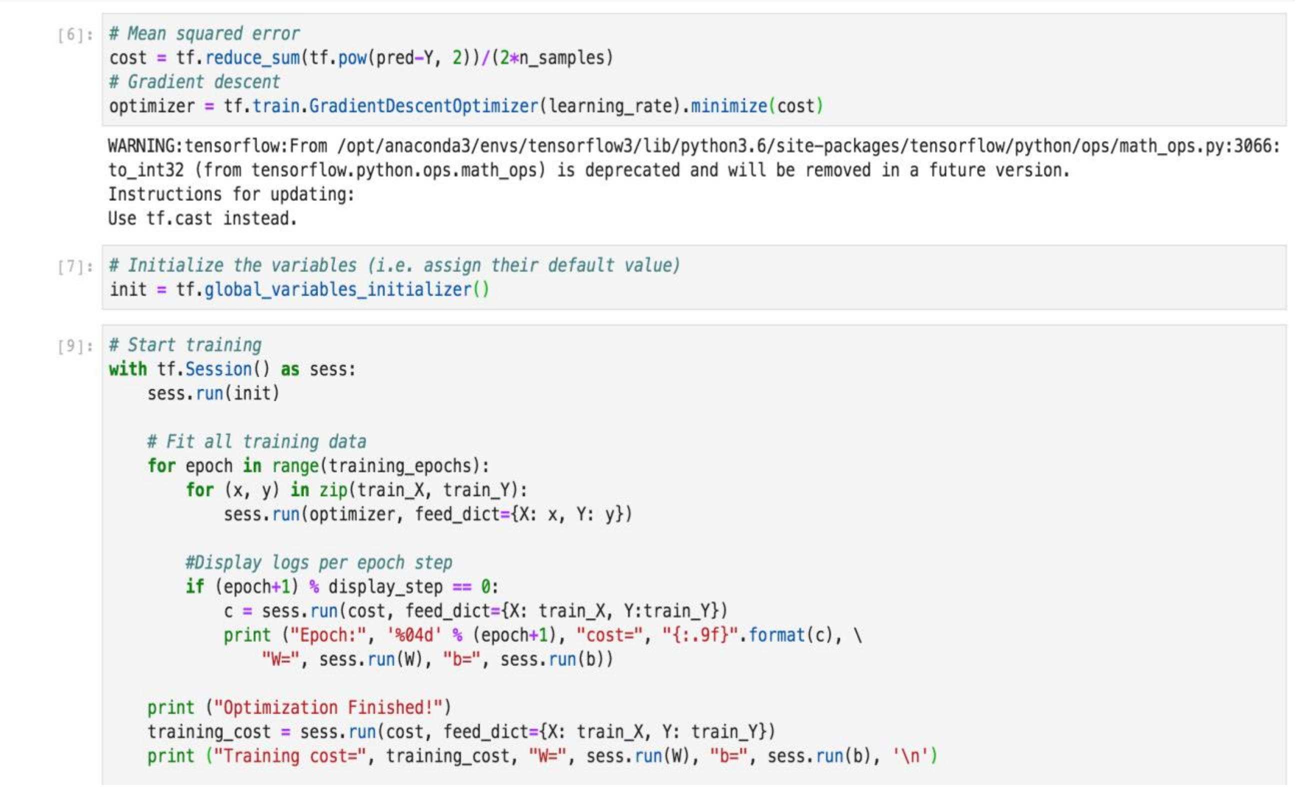
```
# Start training
```

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

    # Fit all training data
    for epoch in range(training_epochs):
        for (x, y) in zip(train_X, train_Y):
            sess.run(optimizer, feed_dict={X: x, Y: y})

        #Display logs per epoch step
        if (epoch+1) % display_step == 0:
            c = sess.run(cost, feed_dict={X: train_X, Y:train_Y})                pr
int ("Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".f               ormat(c), "W
=", sess.run(W), "b=", sess.run(b))
     print ("Optimization Finished!")
     training_cost = sess.run(cost, feed_dict={X: train_X, Y: train_Y})
     print ("Training cost=", training_cost, "W=", sess.run(W), "b="          , se
ss.run(b), '\n')

     #Graphic display
     plt.plot(train_X, train_Y, 'ro', label='Original data')
     plt.plot(train_X, sess.run(W) * train_X + sess.run(b), label='F     itted lin
e')
     plt.legend()
     plt.show()
```

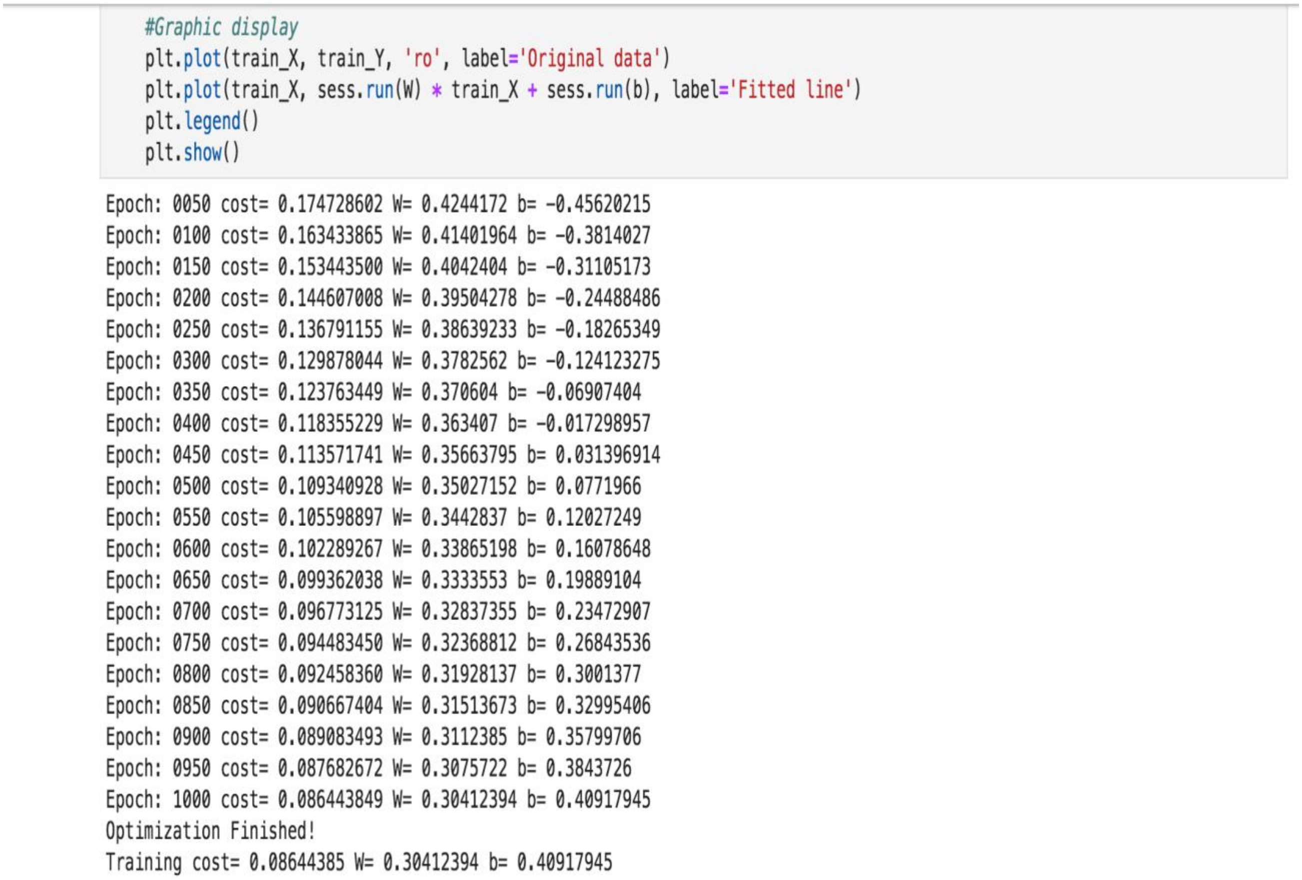
1.2 Output of the Test Case



```
[1]: import tensorflow as tf
     import numpy
     import matplotlib.pyplot as plt
     rng = numpy.random

[2]: # Parameters
     learning_rate = 0.01
     training_epochs = 1000
     display_step = 50

[3]: # Training Data
     train_X = numpy.asarray([3.3,4.4,5.5,6.71,6.93,4.168,9.779,6.182,7.59,2.167,
                              7.042,10.791,5.313,7.997,5.654,9.27,3.1])
     train_Y = numpy.asarray([1.7,2.76,2.09,3.19,1.694,1.573,3.366,2.596,2.53,1.221,
                              2.827,3.465,1.65,2.904,2.42,2.94,1.3])
     n_samples = train_X.shape[0]

[4]: # tf Graph Input
     X = tf.placeholder("float")
     Y = tf.placeholder("float")

     # Set model weights
     W = tf.Variable(rng.randn(), name="weight")
     b = tf.Variable(rng.randn(), name="bias")

     WARNING:tensorflow:From /opt/anaconda3/envs/tensorflow3/lib/python3.6/site-packages/tensorflow/python/framework/op_def_libr
     ary.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
     Instructions for updating:
     Colocations handled automatically by placer.

[5]: # Construct a linear model
     pred = tf.add(tf.multiply(X, W), b)
```

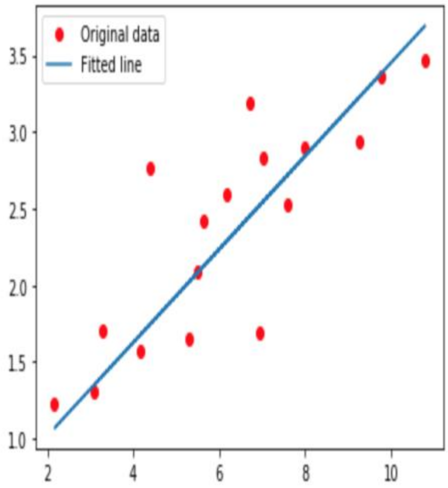


```
[6]: # Mean squared error
     cost = tf.reduce_sum(tf.pow(pred-Y, 2))/(2*n_samples)
     # Gradient descent
     optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

     WARNING:tensorflow:From /opt/anaconda3/envs/tensorflow3/lib/python3.6/site-packages/tensorflow/python/ops/math_ops.py:3066:
     to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
     Instructions for updating:
     Use tf.cast instead.

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

[9]: # Start training
     with tf.Session() as sess:
         sess.run(init)

         # Fit all training data
         for epoch in range(training_epochs):
             for (x, y) in zip(train_X, train_Y):
                 sess.run(optimizer, feed_dict={X: x, Y: y})

             #Display logs per epoch step
             if (epoch+1) % display_step == 0:
                 c = sess.run(cost, feed_dict={X: train_X, Y:train_Y})
                 print ("Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format(c), \
                     "W=", sess.run(W), "b=", sess.run(b))

         print ("Optimization Finished!")
         training_cost = sess.run(cost, feed_dict={X: train_X, Y: train_Y})
         print ("Training cost=", training_cost, "W=", sess.run(W), "b=", sess.run(b), '\n')
```



```
#Graphic display
plt.plot(train_X, train_Y, 'ro', label='Original data')
plt.plot(train_X, sess.run(W) * train_X + sess.run(b), label='Fitted line')
plt.legend()
plt.show()
```

```
Epoch: 0050 cost= 0.174728602 W= 0.4244172 b= -0.45620215
Epoch: 0100 cost= 0.163433865 W= 0.41401964 b= -0.3814027
Epoch: 0150 cost= 0.153443500 W= 0.4042404 b= -0.31105173
Epoch: 0200 cost= 0.144607008 W= 0.39504278 b= -0.24488486
Epoch: 0250 cost= 0.136791155 W= 0.38639233 b= -0.18265349
Epoch: 0300 cost= 0.129878044 W= 0.3782562 b= -0.124123275
Epoch: 0350 cost= 0.123763449 W= 0.370604 b= -0.06907404
Epoch: 0400 cost= 0.118355229 W= 0.363407 b= -0.017298957
Epoch: 0450 cost= 0.113571741 W= 0.35663795 b= 0.031396914
Epoch: 0500 cost= 0.109340928 W= 0.35027152 b= 0.0771966
Epoch: 0550 cost= 0.105598897 W= 0.3442837 b= 0.12027249
Epoch: 0600 cost= 0.102289267 W= 0.33865198 b= 0.16078648
Epoch: 0650 cost= 0.099362038 W= 0.3333553 b= 0.19889104
Epoch: 0700 cost= 0.096773125 W= 0.32837355 b= 0.23472907
Epoch: 0750 cost= 0.094483450 W= 0.32368812 b= 0.26843536
Epoch: 0800 cost= 0.092458360 W= 0.31928137 b= 0.3001377
Epoch: 0850 cost= 0.090667404 W= 0.31513673 b= 0.32995406
Epoch: 0900 cost= 0.089083493 W= 0.3112385 b= 0.35799706
Epoch: 0950 cost= 0.087682672 W= 0.3075722 b= 0.3843726
Epoch: 1000 cost= 0.086443849 W= 0.30412394 b= 0.40917945
Optimization Finished!
Training cost= 0.08644385 W= 0.30412394 b= 0.40917945
```



2 TEST USING JUPYTER-NOTEBOOK

2.1 Use Neutral Network Example

Reference:

https://github.com/aymericdamien/TensorFlow-Examples/blob/master/notebooks/3_NeuralNetworks/neural_network.ipynb

Start python3 and execute the below code

```
from __future__ import print_function

# Import MNIST data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot=False)

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
```

```
# Parameters
learning_rate = 0.1
num_steps = 1000
batch_size = 128
display_step = 100

# Network Parameters
n_hidden_1 = 256 # 1st layer number of neurons
n_hidden_2 = 256 # 2nd layer number of neurons
num_input = 784 # MNIST data input (img shape: 28*28)
num_classes = 10 # MNIST total classes (0-9 digits)
```

```
# Define the input function for training
input_fn = tf.estimator.inputs.numpy_input_fn(
     x={'images': mnist.train.images}, y=mnist.train.labels,
     batch_size=batch_size, num_epochs=None, shuffle=True)
```

```
# Define the neural network
def neural_net(x_dict):
    # TF Estimator input is a dict, in case of multiple inputs
    x = x_dict['images']
    # Hidden fully connected layer with 256 neurons
    layer_1 = tf.layers.dense(x, n_hidden_1)
    # Hidden fully connected layer with 256 neurons
    layer_2 = tf.layers.dense(layer_1, n_hidden_2)
    # Output fully connected layer with a neuron for each class         out_layer
= tf.layers.dense(layer_2, num_classes)
    return out_layer
```

```
# Define the model function (following TF Estimator Template)
def model_fn(features, labels, mode):

    # Build the neural network
    logits = neural_net(features)

    # Predictions
    pred_classes = tf.argmax(logits, axis=1)
    pred_probas = tf.nn.softmax(logits)

    # If prediction mode, early return
    if mode == tf.estimator.ModeKeys.PREDICT:
        return tf.estimator.EstimatorSpec(mode, predictions=pred_classes)


    # Define loss and optimizer
    loss_op = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with        _logit
s(logits=logits, labels=tf.cast(labels, dtype=tf.int32        )))
    optimizer = tf.train.GradientDescentOptimizer(learning_rate=lear          ning
_rate)
    train_op = optimizer.minimize(loss_op, global_step=tf.train.get_          glob
al_step())

    # Evaluate the accuracy of the model
    acc_op = tf.metrics.accuracy(labels=labels, predictions=pred_cla         sses)


    # TF Estimators requires to return a EstimatorSpec, that specify    # the diff
erent ops for training, evaluating, ...

    estim_specs = tf.estimator.EstimatorSpec(
      mode=mode,
      predictions=pred_classes,
      loss=loss_op,
      train_op=train_op,
      eval_metric_ops={'accuracy': acc_op})

      return estim_specs
```

```
# Build the Estimator
model = tf.estimator.Estimator(model_fn)
```

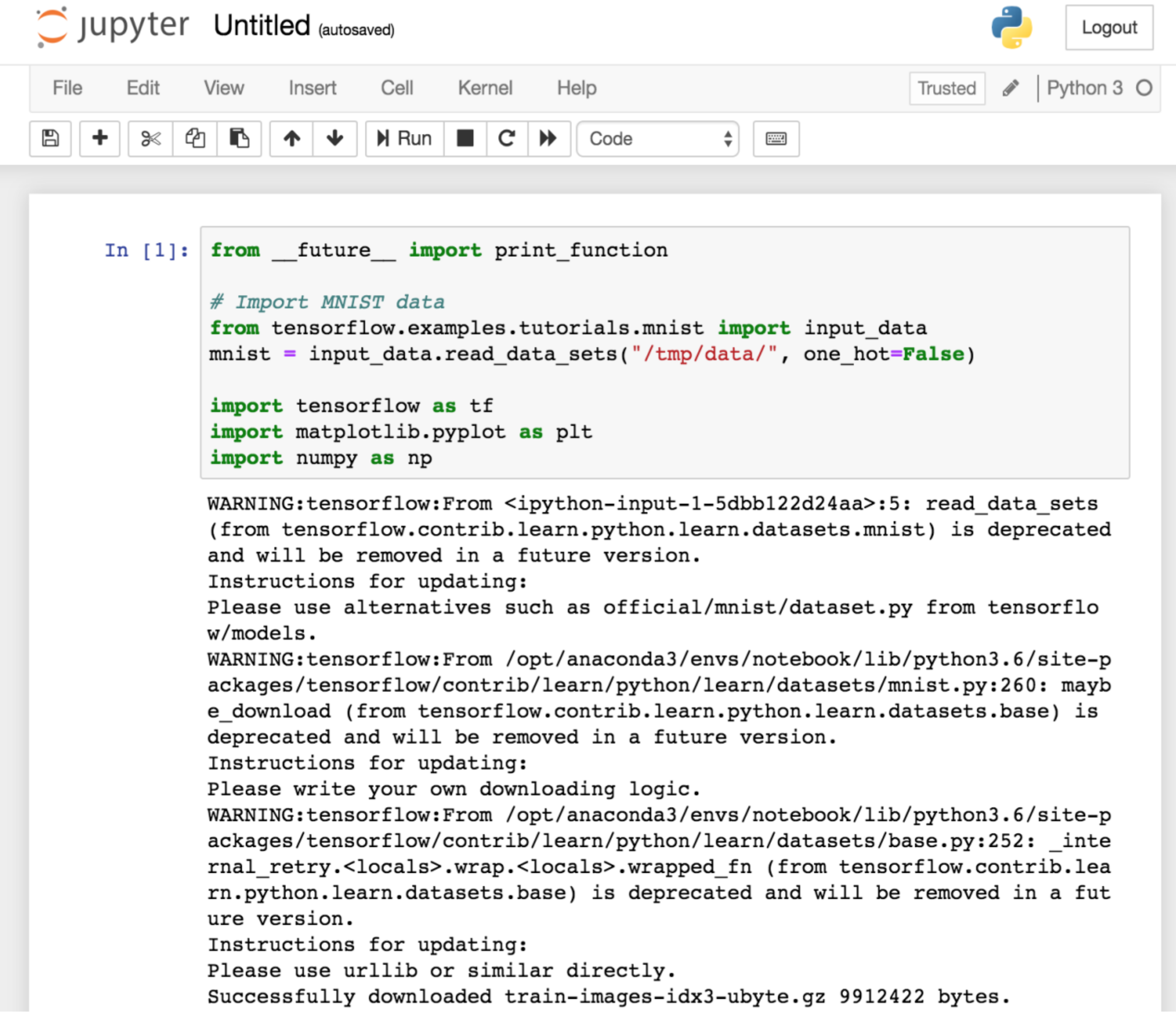
```
# Train the Model
model.train(input_fn, steps=num_steps)
```

```
# Evaluate the Model
# Define the input function for evaluating
input_fn = tf.estimator.inputs.numpy_input_fn(
     x={'images': mnist.test.images}, y=mnist.test.labels,
     batch_size=batch_size, shuffle=False)
# Use the Estimator 'evaluate' method
```

```
model.evaluate(input_fn)
```

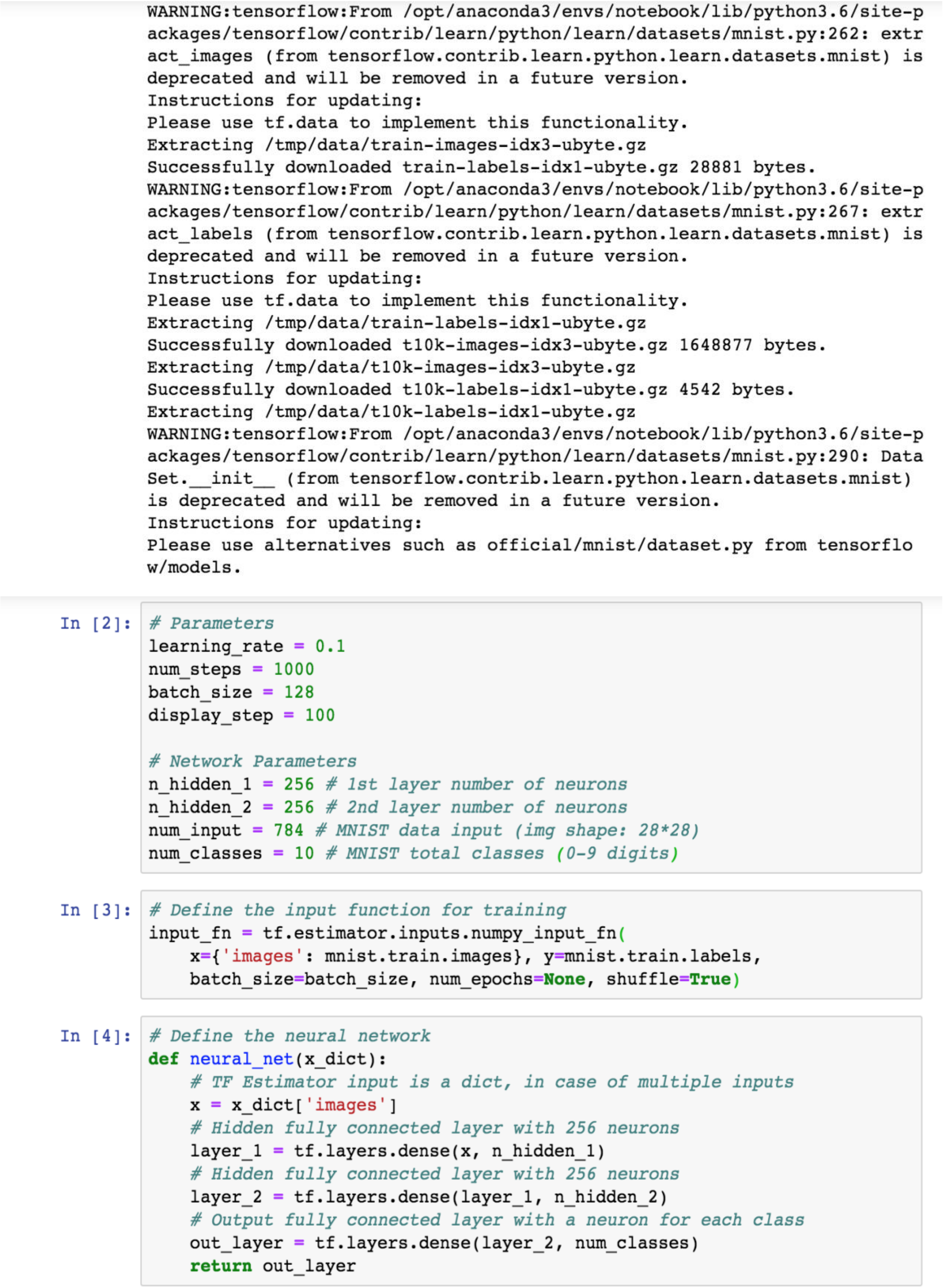
```
# Predict single images
n_images = 4
# Get images from test set
test_images = mnist.test.images[:n_images]
# Prepare the input data
input_fn = tf.estimator.inputs.numpy_input_fn(
    x={'images': test_images}, shuffle=False)
# Use the model to predict the images class
preds = list(model.predict(input_fn))

# Display
for i in range(n_images):
    plt.imshow(np.reshape(test_images[i], [28, 28]), cmap='gray')       plt.show()

    print("Model prediction:", preds[i])
```

2.2 Output of the Test Case


jupyter Untitled (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code

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

        # Import MNIST data
        from tensorflow.examples.tutorials.mnist import input_data
        mnist = input_data.read_data_sets("/tmp/data/", one_hot=False)

        import tensorflow as tf
        import matplotlib.pyplot as plt
        import numpy as np
```

```
WARNING:tensorflow:From <ipython-input-1-5dbb122d24aa>:5: read_data_sets
(from tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated
and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflo
w/models.
WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow/contrib/learn/python/learn/datasets/mnist.py:260: mayb
e_download (from tensorflow.contrib.learn.python.learn.datasets.base) is
deprecated and will be removed in a future version.
Instructions for updating:
Please write your own downloading logic.
WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow/contrib/learn/python/learn/datasets/base.py:252: _inte
rnal_retry.<locals>.wrap.<locals>.wrapped_fn (from tensorflow.contrib.lea
rn.python.learn.datasets.base) is deprecated and will be removed in a fut
ure version.
Instructions for updating:
Please use urllib or similar directly.
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
```



```
WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow/contrib/learn/python/learn/datasets/mnist.py:262: extr
act_images (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting /tmp/data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow/contrib/learn/python/learn/datasets/mnist.py:267: extr
act_labels (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow/contrib/learn/python/learn/datasets/mnist.py:290: Data
Set.__init__ (from tensorflow.contrib.learn.python.learn.datasets.mnist)
is deprecated and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflo
w/models.
```

In [2]:

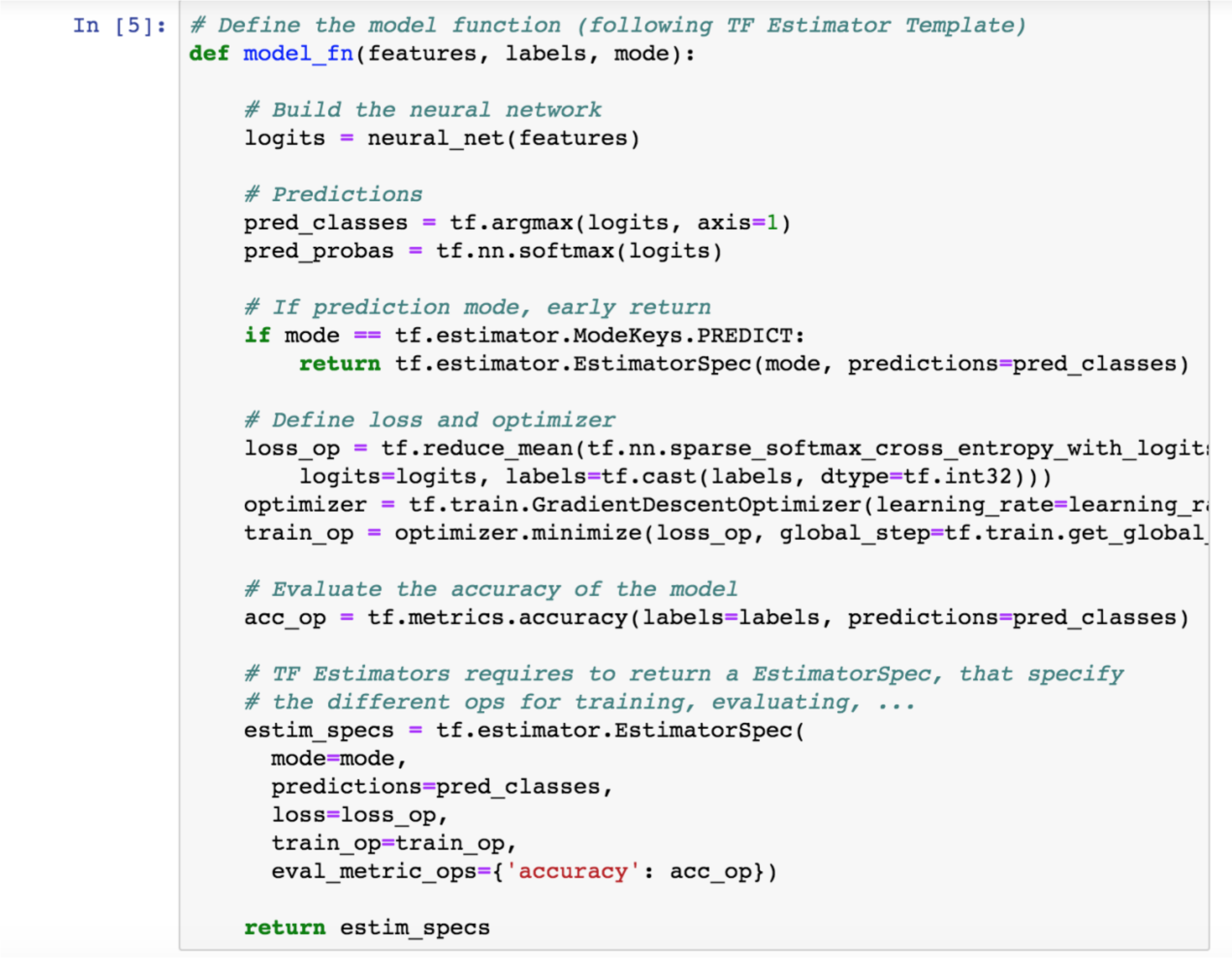
```
# Parameters
learning_rate = 0.1
num_steps = 1000
batch_size = 128
display_step = 100

# Network Parameters
n_hidden_1 = 256 # 1st layer number of neurons
n_hidden_2 = 256 # 2nd layer number of neurons
num_input = 784 # MNIST data input (img shape: 28*28)
num_classes = 10 # MNIST total classes (0-9 digits)
```

In [3]:

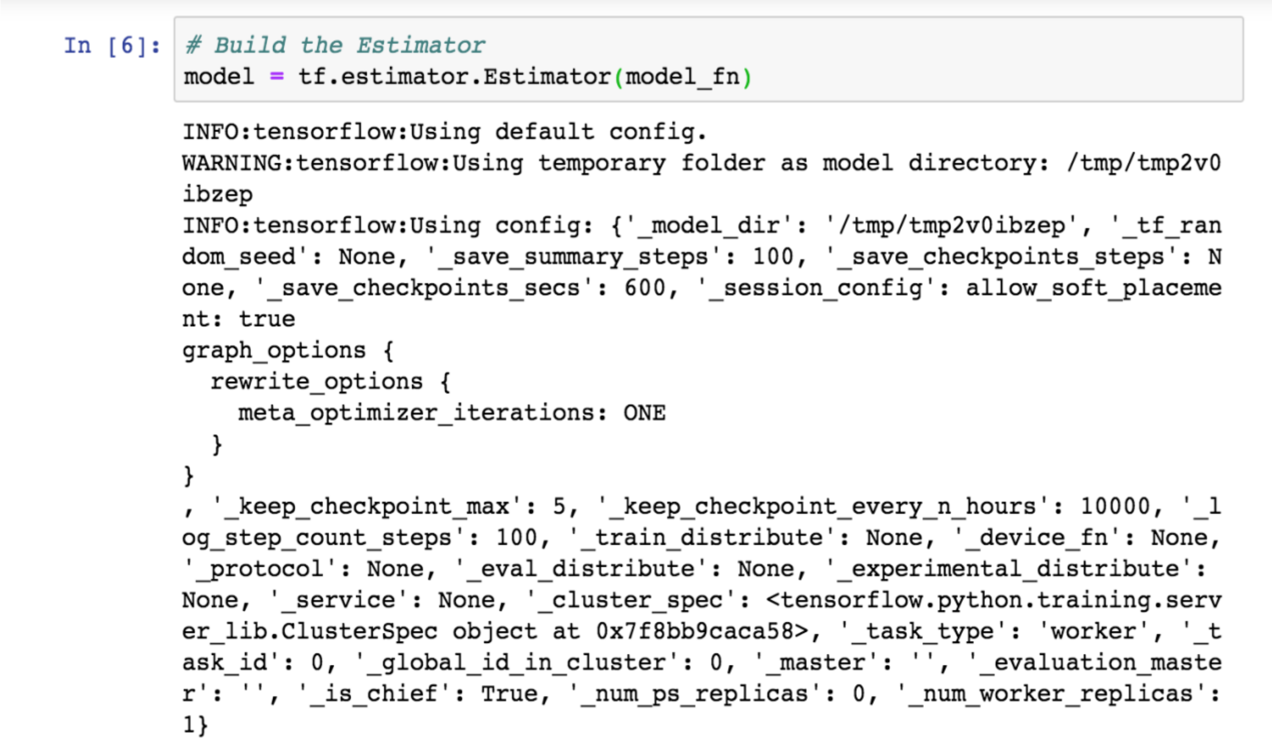
```
# Define the input function for training
input_fn = tf.estimator.inputs.numpy_input_fn(
    x={'images': mnist.train.images}, y=mnist.train.labels,
    batch_size=batch_size, num_epochs=None, shuffle=True)
```

In [4]:

```
# Define the neural network
def neural_net(x_dict):
    # TF Estimator input is a dict, in case of multiple inputs
    x = x_dict['images']
    # Hidden fully connected layer with 256 neurons
    layer_1 = tf.layers.dense(x, n_hidden_1)
    # Hidden fully connected layer with 256 neurons
    layer_2 = tf.layers.dense(layer_1, n_hidden_2)
    # Output fully connected layer with a neuron for each class
    out_layer = tf.layers.dense(layer_2, num_classes)
    return out_layer
```

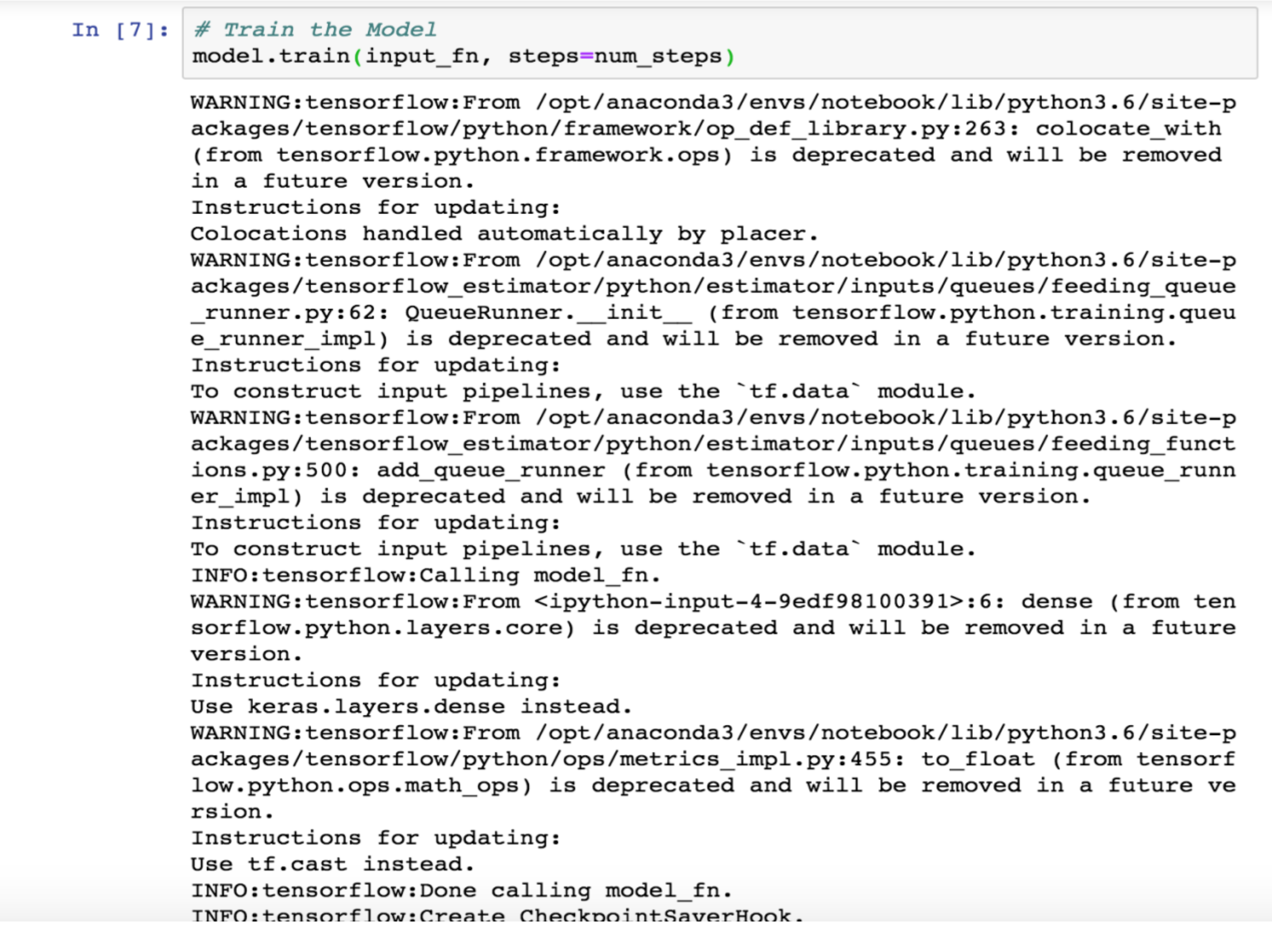


```
In [5]: # Define the model function (following TF Estimator Template)
        def model_fn(features, labels, mode):

            # Build the neural network
            logits = neural_net(features)

            # Predictions
            pred_classes = tf.argmax(logits, axis=1)
            pred_probas = tf.nn.softmax(logits)

            # If prediction mode, early return
            if mode == tf.estimator.ModeKeys.PREDICT:
                return tf.estimator.EstimatorSpec(mode, predictions=pred_classes)

            # Define loss and optimizer
            loss_op = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logit:
                logits=logits, labels=tf.cast(labels, dtype=tf.int32)))
            optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_ra
            train_op = optimizer.minimize(loss_op, global_step=tf.train.get_global_

            # Evaluate the accuracy of the model
            acc_op = tf.metrics.accuracy(labels=labels, predictions=pred_classes)

            # TF Estimators requires to return a EstimatorSpec, that specify
            # the different ops for training, evaluating, ...
            estim_specs = tf.estimator.EstimatorSpec(
              mode=mode,
              predictions=pred_classes,
              loss=loss_op,
              train_op=train_op,
              eval_metric_ops={'accuracy': acc_op})

            return estim_specs
```

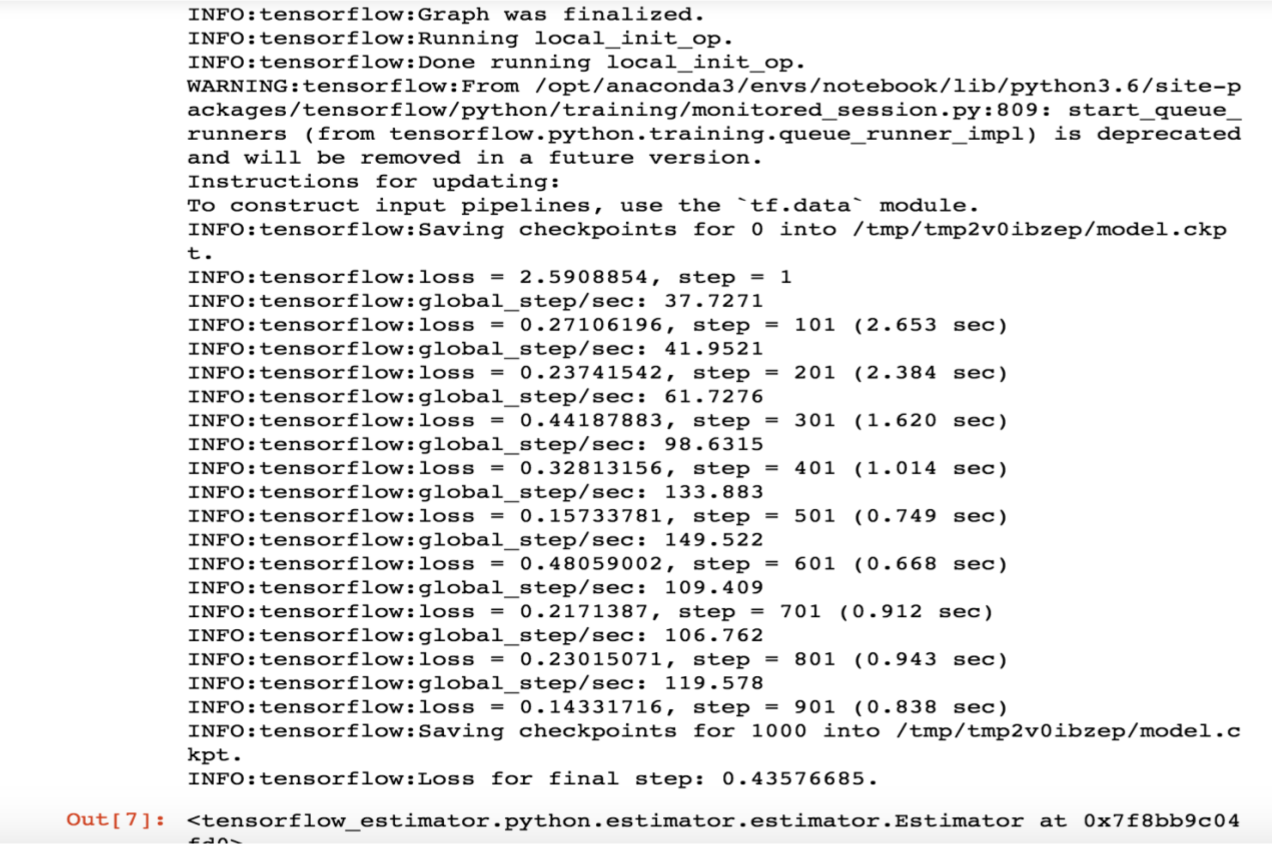


```
In [6]: # Build the Estimator
        model = tf.estimator.Estimator(model_fn)
```

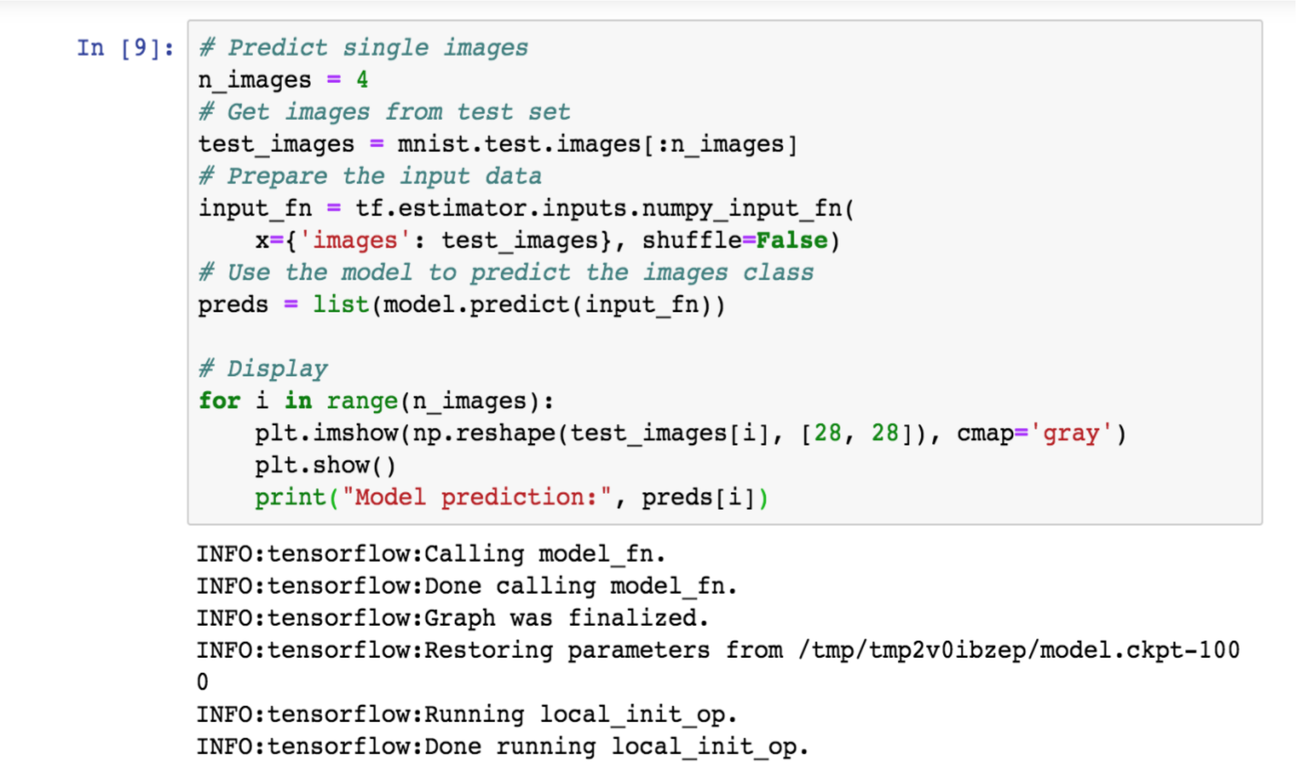
```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: /tmp/tmp2v0
ibzep
INFO:tensorflow:Using config: {'_model_dir': '/tmp/tmp2v0ibzep', '_tf_ran
dom_seed': None, '_save_summary_steps': 100, '_save_checkpoints_steps': N
one, '_save_checkpoints_secs': 600, '_session_config': allow_soft_placeme
nt: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_l
og_step_count_steps': 100, '_train_distribute': None, '_device_fn': None,
'_protocol': None, '_eval_distribute': None, '_experimental_distribute':
None, '_service': None, '_cluster_spec': <tensorflow.python.training.serv
er_lib.ClusterSpec object at 0x7f8bb9caca58>, '_task_type': 'worker', '_t
ask_id': 0, '_global_id_in_cluster': 0, '_master': '', '_evaluation_maste
r': '', '_is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas':
1}
```



```
In [7]: # Train the Model
        model.train(input_fn, steps=num_steps)

WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow/python/framework/op_def_library.py:263: colocate_with
(from tensorflow.python.framework.ops) is deprecated and will be removed
in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow_estimator/python/estimator/inputs/queues/feeding_queue
_runner.py:62: QueueRunner.__init__ (from tensorflow.python.training.queu
e_runner_impl) is deprecated and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow_estimator/python/estimator/inputs/queues/feeding_funct
ions.py:500: add_queue_runner (from tensorflow.python.training.queue_runn
er_impl) is deprecated and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Calling model_fn.
WARNING:tensorflow:From <ipython-input-4-9edf98100391>:6: dense (from ten
sorflow.python.layers.core) is deprecated and will be removed in a future
version.
Instructions for updating:
Use keras.layers.dense instead.
WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow/python/ops/metrics_impl.py:455: to_float (from tensorf
low.python.ops.math_ops) is deprecated and will be removed in a future ve
rsion.
Instructions for updating:
Use tf.cast instead.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
```

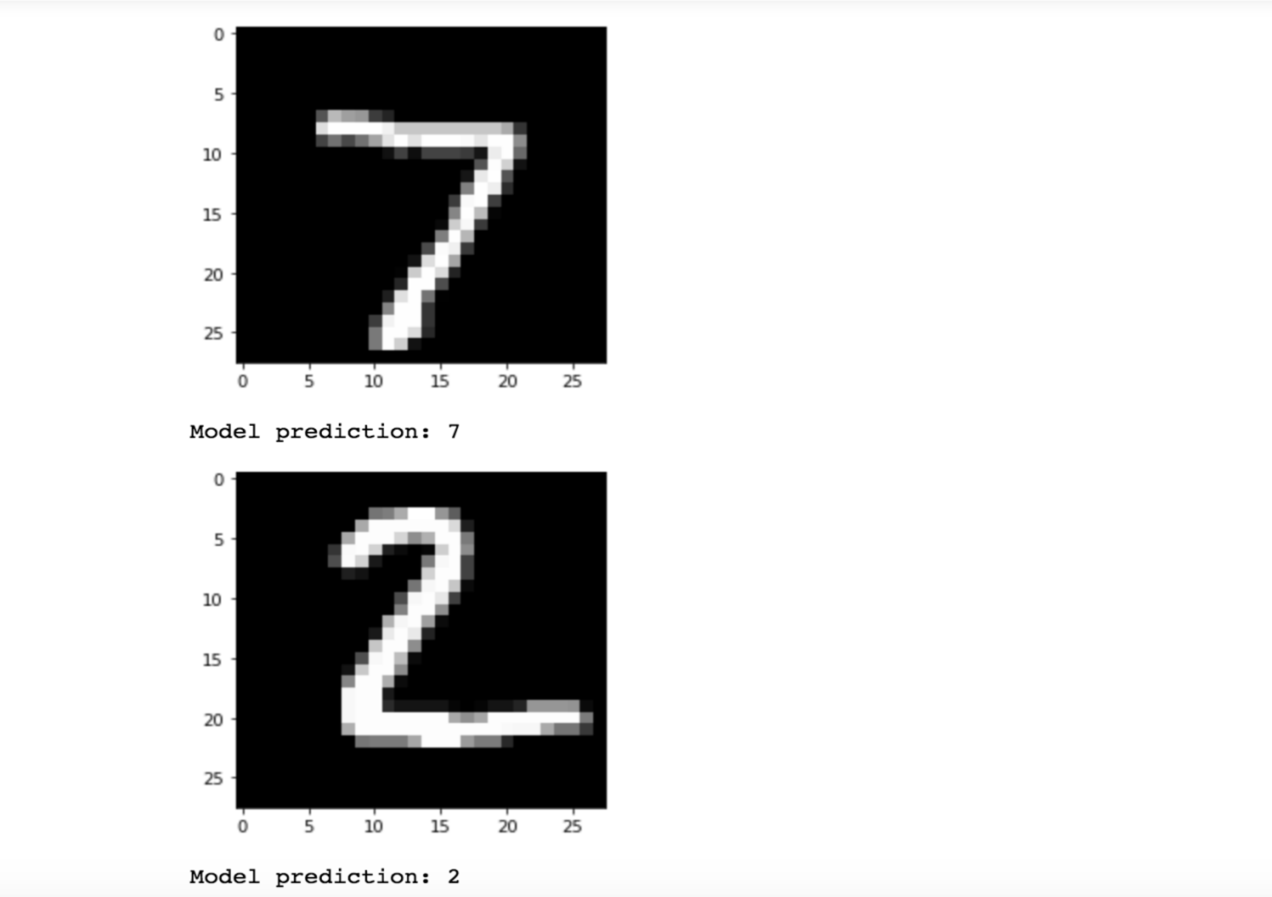


```
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
WARNING:tensorflow:From /opt/anaconda3/envs/notebook/lib/python3.6/site-p
ackages/tensorflow/python/training/monitored_session.py:809: start_queue_
runners (from tensorflow.python.training.queue_runner_impl) is deprecated
and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Saving checkpoints for 0 into /tmp/tmp2v0ibzep/model.ckp
t.
INFO:tensorflow:loss = 2.5908854, step = 1
INFO:tensorflow:global_step/sec: 37.7271
INFO:tensorflow:loss = 0.27106196, step = 101 (2.653 sec)
INFO:tensorflow:global_step/sec: 41.9521
INFO:tensorflow:loss = 0.23741542, step = 201 (2.384 sec)
INFO:tensorflow:global_step/sec: 61.7276
INFO:tensorflow:loss = 0.44187883, step = 301 (1.620 sec)
INFO:tensorflow:global_step/sec: 98.6315
INFO:tensorflow:loss = 0.32813156, step = 401 (1.014 sec)
INFO:tensorflow:global_step/sec: 133.883
INFO:tensorflow:loss = 0.15733781, step = 501 (0.749 sec)
INFO:tensorflow:global_step/sec: 149.522
INFO:tensorflow:loss = 0.48059002, step = 601 (0.668 sec)
INFO:tensorflow:global_step/sec: 109.409
INFO:tensorflow:loss = 0.2171387, step = 701 (0.912 sec)
INFO:tensorflow:global_step/sec: 106.762
INFO:tensorflow:loss = 0.23015071, step = 801 (0.943 sec)
INFO:tensorflow:global_step/sec: 119.578
INFO:tensorflow:loss = 0.14331716, step = 901 (0.838 sec)
INFO:tensorflow:Saving checkpoints for 1000 into /tmp/tmp2v0ibzep/model.c
kpt.
INFO:tensorflow:Loss for final step: 0.43576685.

Out[7]: <tensorflow_estimator.python.estimator.estimator.Estimator at 0x7f8bb9c04
        fd0>
```



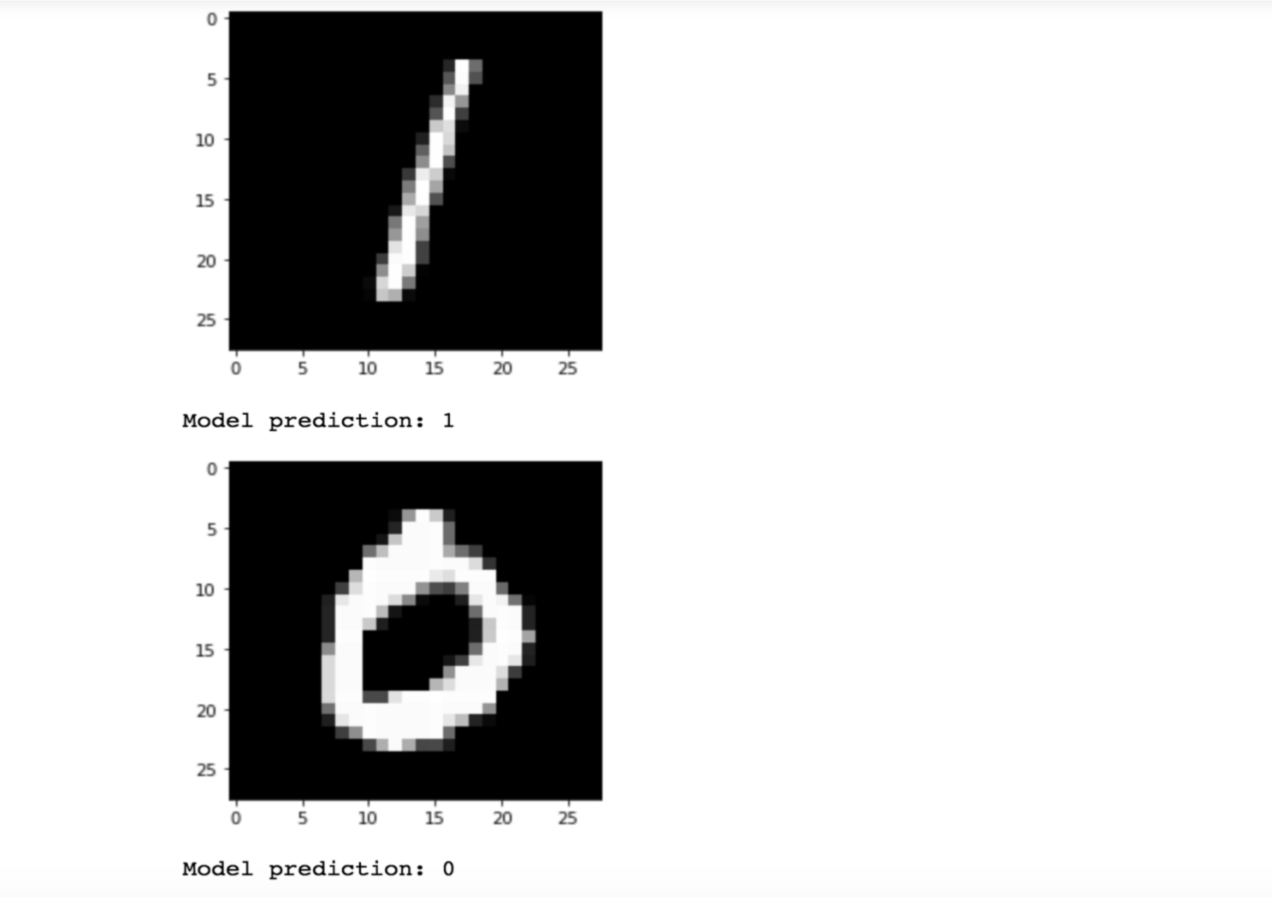
```
In [9]: # Predict single images
        n_images = 4
        # Get images from test set
        test_images = mnist.test.images[:n_images]
        # Prepare the input data
        input_fn = tf.estimator.inputs.numpy_input_fn(
            x={'images': test_images}, shuffle=False)
        # Use the model to predict the images class
        preds = list(model.predict(input_fn))

        # Display
        for i in range(n_images):
            plt.imshow(np.reshape(test_images[i], [28, 28]), cmap='gray')
            plt.show()
            print("Model prediction:", preds[i])
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmp2v0ibzep/model.ckpt-100
0
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
```



```
Model prediction: 7
```

```
Model prediction: 2
```



```
Model prediction: 1
```

```
Model prediction: 0
```