```
from __future__ import absolute_import, division, print_function
import tensorflow as tf
import numpy as np


from tensorflow.keras.datasets import mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()

x_train, x_test = np.array(x_train, np.float32), np.array(x_test, np.float32)
x_train, x_test = x_train.reshape([-1, num_features]), x_test.reshape([-1, num_features])

x_train, x_test = x_train/225., x_test/225.


num_classes = 10 # 0 to 9 digits
num_features = 784 # 28*28

learning_rate = 0.01
training_steps = 1000
batch_size = 256
display_step = 50


train_data = tf.data.Dataset.from_tensor_slices((x_train, y_train))
train_data = train_data.repeat().shuffle(5000).batch(batch_size).prefetch(1)


w = tf.Variable(tf.ones([num_features, num_classes]), name = "weight")
b = tf.Variable(tf.zeros([num_classes]), name = "bias")


def logistic_regression(x):
  return tf.nn.softmax(tf.matmul(x,w) + b)


def cross_entropy(y_pred, y_true):
  y_true = tf.one_hot(y_true, depth= num_classes)
  y_pred = tf.clip_by_value(y_pred, 1e-9, 1.)
  return tf.reduce_mean(-tf.reduce_sum(y_true * tf.math.log(y_pred)))


def accuracy(y_pred, y_true):
  correct_prediction = tf.equal(tf.argmax(y_pred, 1),tf.cast(y_true, tf.int64))
  return tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
  optimizer = tf.optimizer.SGD(learning_rate)


def run_optimization(x, y):
  with tf.GradientTape() as g:
    pred = logistic_regression(x)
    loss = cross_entropy(pred, y)
    gradients = g.gradient(loss, [w, b])
    optimizer.apply_gradients(zip(gradients, [w,b]))
    run_optimization(batch_x, batch_y)
    if step % display_step == 0:
      pred = logistic_regression(batch_x)
```

```
        loss = cross_entrop(pred, batch_y)
        acc = accuracy(pred, batch_y)
        print("step: %i, loss: %f, accuracy: %f" %(step, loss, acc))
```

```
pred = logistic_regression(x_test)
print("Test Accuracy: %f" % accuracy(pred, y_test))
```

```
    Test Accuracy: 0.098000
```