```
    #encoding:utf-8
# import libraries（导入依赖的库文件）
    import tensorflow as tf
    import numpy as np
    from tensorflow.examples.tutorials.mnist import input_data
    import matplotlib.pyplot as plt
#define parameteer（定义函数参数）
    image_size（图片尺寸） /out_class（数据集类别） /display_step （显示次数）
#define super parameter（定义训练超参数）
    train_keep_prop（保留概率） /batch_size （minibatch）/epcoh（迭代次数）
#read dataset and set vector as one_hot（读取数据，以mnist为例）
    mnist = input_data.read_data_sets('MNIST_data',one_hot = True)
#read test dataset（分别读取测试数据的images和labels）
    test_x = mnist.test.images[:1000]
    test_y = mnist.test.labels[:1000]
#define placeholder（定义占位符，在会话中喂入数据）
    xs = tf.placeholder(tf.float32,[None,image_size])
    ys = tf.placeholder(tf.float32,[None,out_class])
    keep_prob = tf.placeholder(tf.float32)
#reshape image to vector [samples_num,28,28,1]（整形，将数据转换为矩阵形式）
    x_image = tf.reshape(xs,[-1,28,28,1])       #-1:all of train dataset images
#crate model （构造神经网络模型）
    tf.nn.***()
#computer loss,As target funcation(定义损失函数，即目标函数)
    loss = tf.losses.softmax_cross_entropy(onehot_labels=ys, logits=prediction)
#define gradent dencent model to minimize loss(target funcation)（定义训练模型，最小化损失函数）
    train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
#computer accuracy（计算精度）
    accury =
tf.reduce_mean(tf.cast(tf.equal(tf.argmax(prediction,1),tf.argmax(ys,1)),tf.float32))
```

```
#init parametre step1 （初始化变量第一步）
    init = tf.global_variables_initializer()
#start session（开启会话）
    with tf.Session() as sess:
#init parametre step2（初始化变量第一步）
    sess.run(init)
#start train model with    （运行训练模型）
    for i in range(epcoh):
#read data （读取数据，喂入训练模型）
    batch_xs,batch_ys = mnist.train.next_batch(batch_size)
#train CNN modele and feed data with（）
    sess.run(train_step,feed_dict = {xs:batch_xs,ys:batch_ys,keep_prob:1})
#display result （显示训练结果）
        if i % display_step == 0:
                print('Test Accuracy : '+str(sess.run(accury_train,feed_dict =
{xs:mnist.test.images,ys:mnist.test.labels,keep_prob:1})))
```