mnist-eager.py

```python
import numpy as np
import tensorflow as tf
tf.enable_eager_execution()

# supress warnings when loading mnist
old_v = tf.logging.get_verbosity()
tf.logging.set_verbosity(tf.logging.ERROR)

from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
tf.logging.set_verbosity(old_v)

dim_hidden = 1024

layer_cnn0 = tf.layers.Conv2D(32, 5, activation = tf.nn.relu)
layer_pool0 = tf.layers.MaxPooling2D(2, 2)
layer_cnn1 = tf.layers.Conv2D(64, 5, activation = tf.nn.relu)
layer_pool1 = tf.layers.MaxPooling2D(2, 2)
layer_flatten = tf.layers.Flatten()
layer_fc0 = tf.layers.Dense(dim_hidden, activation = tf.nn.relu)
layer_dropout = tf.layers.Dropout(rate=0.75) # dropout rate is 0.75. Retain 0.25
layer_fc1 = tf.layers.Dense(10, activation = None) # 1


# forward propagation
def prediction(X, training):
    inputs = tf.constant(X)
    cnn0 = layer_cnn0(inputs)
    pool0 = layer_pool0(cnn0)
    cnn1 = layer_cnn1(pool0)
    pool1 = layer_pool1(cnn1)
    flatten = layer_flatten(pool1)
    fc0 = layer_fc0(flatten)
    dropout = layer_dropout(fc0, training=training)
    output = layer_fc1(dropout)
    return output

# cross entropy loss
def loss(X, y, training):
    logits = prediction(X, training)
    loss = tf.nn.softmax_cross_entropy_with_logits_v2(labels = y, logits = logits)
    loss = tf.reduce_mean(loss)
    return loss

def binary_accuracy(X, y):
    logits = prediction(X, training = False)
    predict = tf.argmax(logits, 1).numpy()
    target = np.argmax(y, 1)
    binary_accuracy = np.sum(predict == target)/len(target)
    return(binary_accuracy)

X_validation = mnist.validation.images
```

```
y_validation = mnist.validation.labels
X_validation = X_validation.reshape([-1,28,28,1])

def v_binary_accuracy() :
    return(binary_accuracy(X_validation, y_validation))



optimizer = tf.train.AdamOptimizer(learning_rate = 1e-3)
batch_size = 50
iters = 1000

for i in range(iters):
    X, y = mnist.train.next_batch(batch_size)
    X = X.reshape([-1,28,28,1])
    optimizer.minimize(lambda: loss(X, y, True))

    if i % 100 == 0:
        batch_accuracy = binary_accuracy(X, y)
        validation_accuracy = v_binary_accuracy()
        print("batch %d, batch accuracy %.3f validation accuracy %.3f" %
                                (i, batch_accuracy, validation_accuracy))

# evaluate the result
X, y = mnist.test.images, mnist.test.labels
X = X.reshape([-1,28,28,1])
test_accuracy = binary_accuracy(X, y)
print("test accuracy %g" % (test_accuracy))
```