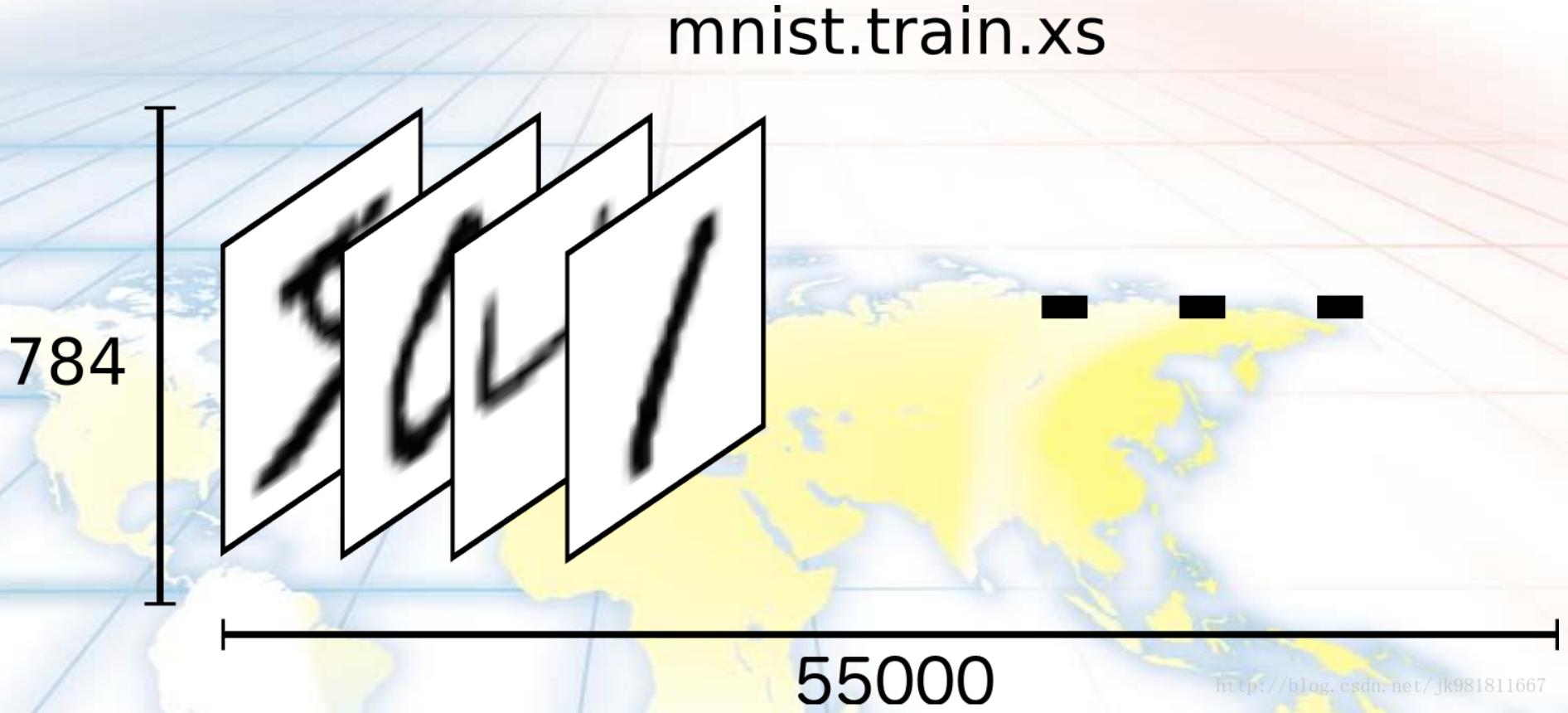
TENSORFLOW视觉识别实例

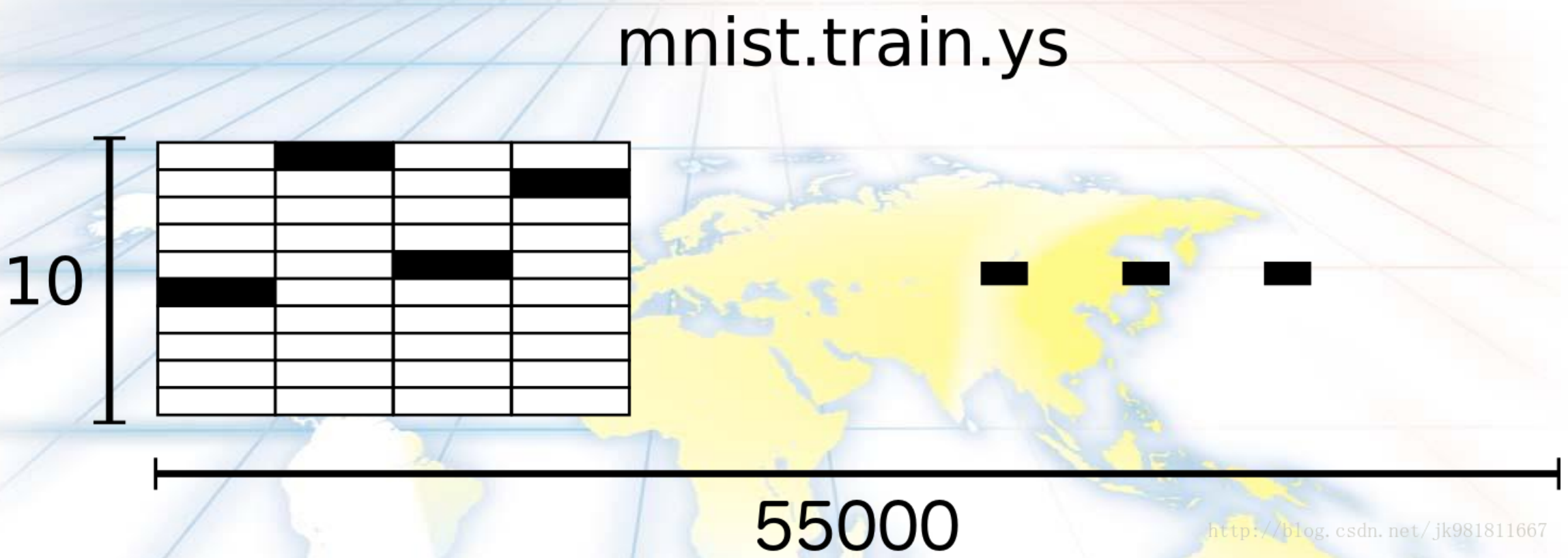
MNIST实例分析

- MNIST：是一个入门级的计算机视觉数据集。
- 它包含各种手写数字图片：
- 

- 上面这4张图片的标签是5，0，4，1。
- 实例目的：使用tensorflow训练一个手写体图像识别模型，识别一张手写数字照片。
- 实例数据：28像素X28像素预处理之后的照片。训练集：55000行 ； 预测集：10000行 ；
- 交叉验证集：5000行



mnist.train.images 是一个形状为 [55000, 784] 的张量，第一个维度数字用来索引图片，第二个维度数字用来索引每张图片中的像素点。在此张量里的每一个元素，都表示某张图片里的某个像素的强度值，值介于0和1之间。

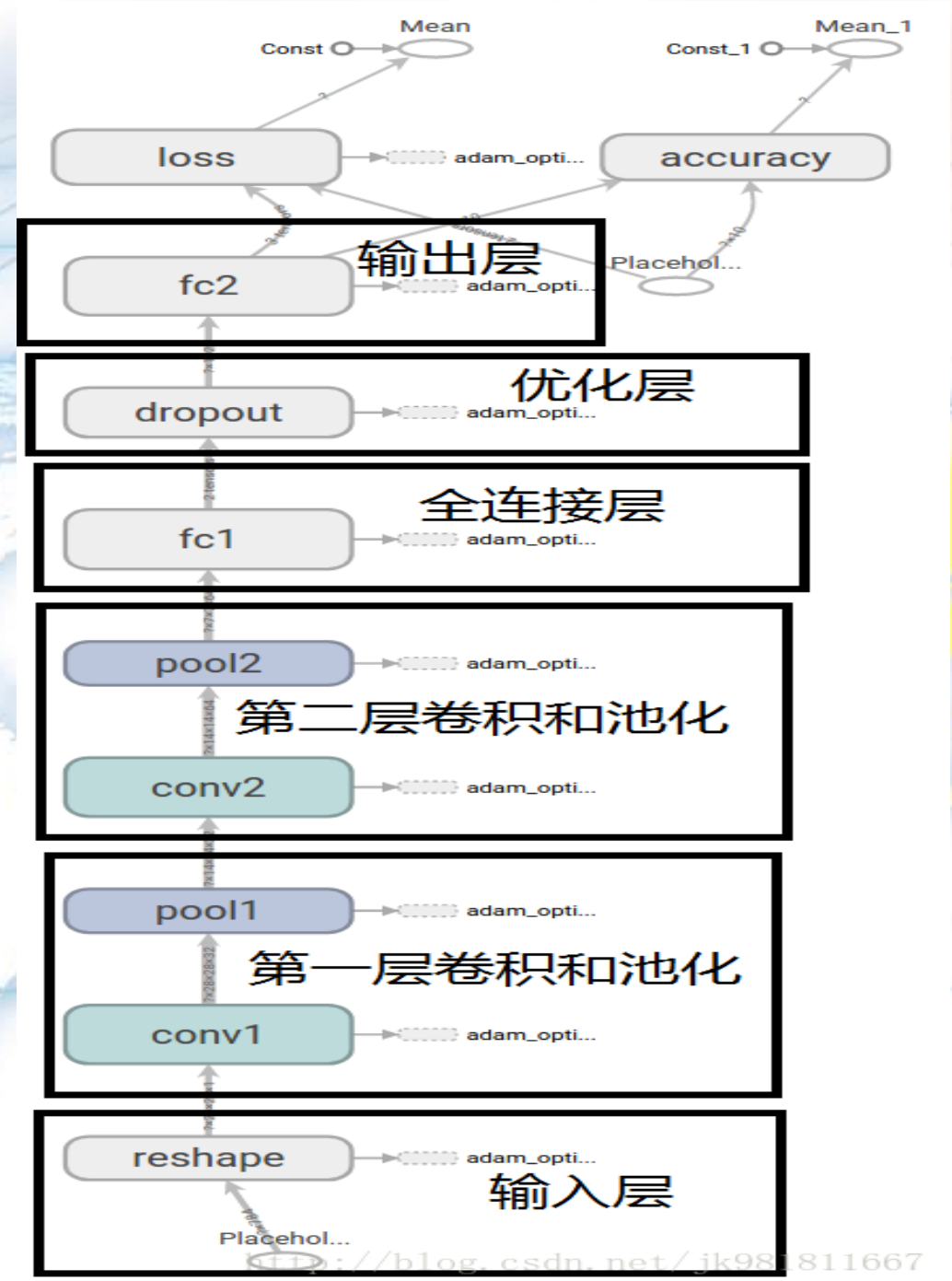


one-hot vectors数字n将表示成一个只有在第n维度（从0开始）数字为1的10维向量。
比如，标签0将表示成([1,0,0,0,0,0,0,0,0,0])。mnist.train.labels是一个形状 [55000, 10] 的张量

数据下载地址

文件	内容
train-images-idx3-ubyte.gz	训练集图片 - 55000 张 训练图片, 5000 张 验证图片
train-labels-idx1-ubyte.gz	训练集图片对应的数字标签
t10k-images-idx3-ubyte.gz	测试集图片 - 10000 张 图片
t10k-labels-idx1-ubyte.gz	测试集图片对应的数字标签

- MNIST_data百度网盘地址：https://pan.baidu.com/s/1cjm7tc
- input_data.py百度网盘地址：https://pan.baidu.com/s/1c21zu4C
- input_data.py用于下载训练和测试的MNIST数据集的python源码。



权重初始化 WEIGHT INITIALIZATION

- 在神经网络中会创建大量的权重和偏置值。如何初始化这些variables。
- 为避免零梯度，我们使用tf.truncated_normal(shape, mean, stddev)生成正态分布的值。shape表示生成张量的维度，mean是均值，stddev是标准差。这样聚能保证随机初始化的权重，偏置值不同。
- 正态分布：统计样本常见的一种数值分布。自然情况下，人的身高是属于正态分布的。
- def weight_variable(shape):
- initial = tf.truncated_normal(shape, stddev=0.1)
- return tf.Variable(initial)
- def bias_variable(shape):
- initial = tf.constant(0.1, shape=shape)
- return tf.Variable(initial)

卷积和池化 CONVOLUTION AND POOLING

- tensorflow提供非常灵活的卷积，池化操作。
- TensorFlow also gives us a lot of flexibility in convolution and pooling operations.
- 卷积使用1步长（stride size），0边距（padding size）的模板，保证输出和输入是同一个大小。
- 池化用简单传统的2x2大小的模板做max pooling。
- def conv2d(x, W):
- return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')
- def max_pool_2x2(x):
- return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
- strides=[1, 2, 2, 1], padding='SAME')

- step1，输入层
- x_image = tf.reshape(x, [-1,28,28,1])
- tf.reshape(tensor, shape, name=None)调整tensor形状。

- step2，第一层卷积和池化
- W_conv1 = weight_variable([5, 5, 1, 32])
- b_conv1 = bias_variable([32])
- h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
- h_pool1 = max_pool_2x2(h_conv1)

- step3，第二层卷积和池化
- W_conv2 = weight_variable([5, 5, 32, 64])
- b_conv2 = bias_variable([64])
- h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
- h_pool2 = max_pool_2x2(h_conv2)

- step4，全连接层
- W_fc1 = weight_variable([7 * 7 * 64, 1024])
- b_fc1 = bias_variable([1024])
- h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
- h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

- step5，优化层
- keep_prob = tf.placeholder(tf.float32)
- h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

- step6，输出层
- W_fc2 = weight_variable([1024, 10])
- b_fc2 = bias_variable([10])
- y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2

- step7，评估和训练
- cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
- train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
- correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
- accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
- sess.run(tf.initialize_all_variables())
- for i in range(20000):
- batch = mnist.train.next_batch(50)
- if i%100 == 0:
- train_accuracy = accuracy.eval(feed_dict={
- x:batch[0], y_: batch[1], keep_prob: 1.0})
- print "step %d, training accuracy %g"%(i, train_accuracy)
- train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})

- print "test accuracy %g"%accuracy.eval(feed_dict={
- x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0})