第 3 章

CHAPTER 3

TensorFlow 可视化

可视化是认识程序的最直观方式。在做数据分析时，可视化一般是数据分析最后一步的结果呈现。把可视化在此介绍，是为了让读者在安装完成后，就能先看一下 TensorFlow 到底有哪些功能，直观感受深度学习的学习成果，让学习目标一目了然。

3.1 PlayGround

PlayGround 是一个用于教学目的的简单神经网络的在线演示、实验的图形化平台，非常强大地可视化了神经网络的训练过程。使用它可以在浏览器中训练神经网络，对 TensorFlow 有一个感性的认识。

PlayGround 界面从左到右由数据（DATA）、特征（FEATURES）、神经网络的隐藏层（HIDDEN LAYERS）和层中的连接线和输出（OUTPUT）几个部分组成，如图 3-1 所示（http://playground.tensorflow.org/）。

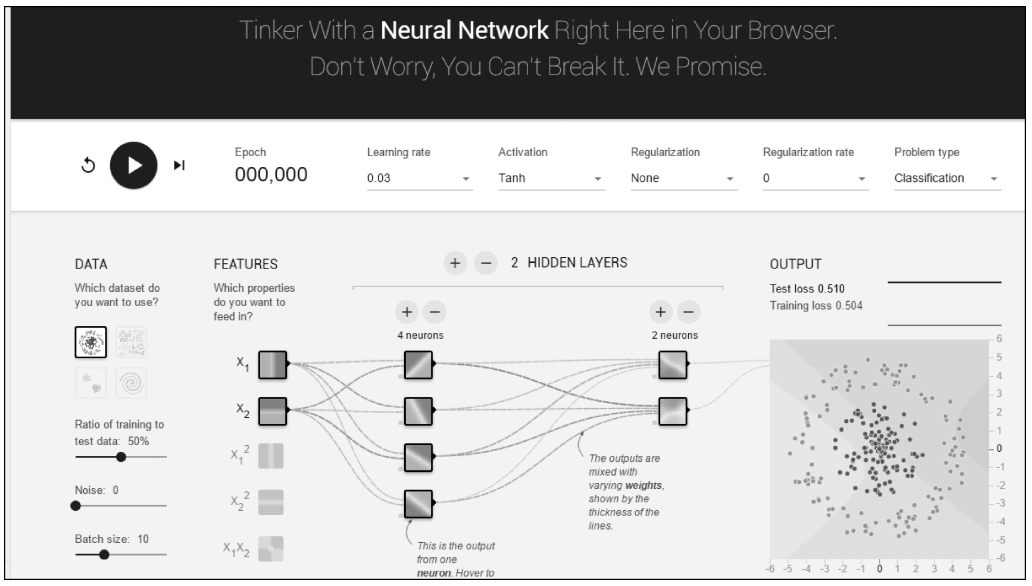

图 3-1 PlayGround 界面

3.1.1 数据

在二维平面内，点被标记成两种颜色。深色（计算机屏幕显示为蓝色）代表正值，浅色（计算机屏幕显示为黄色）代表负值。这两种颜色表示想要区分的两类，如图 3-2 所示。

图 3-2 Data 数据

网站提供了 4 种不同形状的数据，分别是圆形、异或、高斯和螺旋，如图 3-3 所示。神经网络会根据给的数据进行训练，再分类规律相同的点。

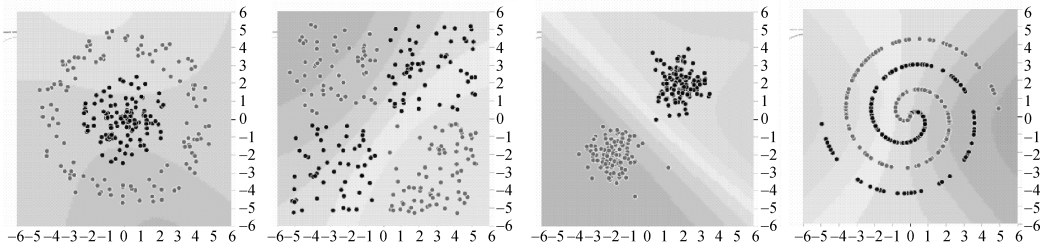

图 3-3 4 种不同形态的数据

PlayGround 中的数据配置非常灵活，可以调整噪声（noise）的大小。图 3-4 展示的是噪声为 0、25 和 50 时的数据分布。

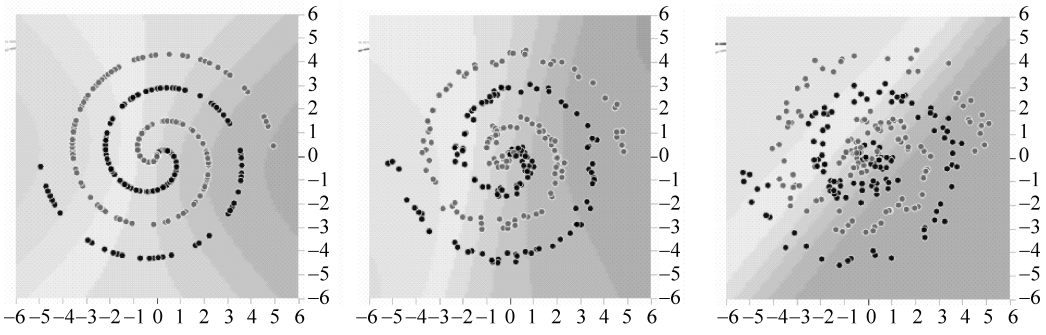

图 3-4 不同噪声值数据的分布情况

PlayGround 中也可以改变训练数据和测试数据的比例（data）。图 3-5 展示的是训练数据和测试数据比例为 1∶9 和 9∶1 的情况。

此外，PlayGround 中还可以调整输入的每批（batch）数据的多少，调整范围可以是 1～30，也就是说每批进入神经网络的数据点可以是 1～30 个，如图 3-6 所示。

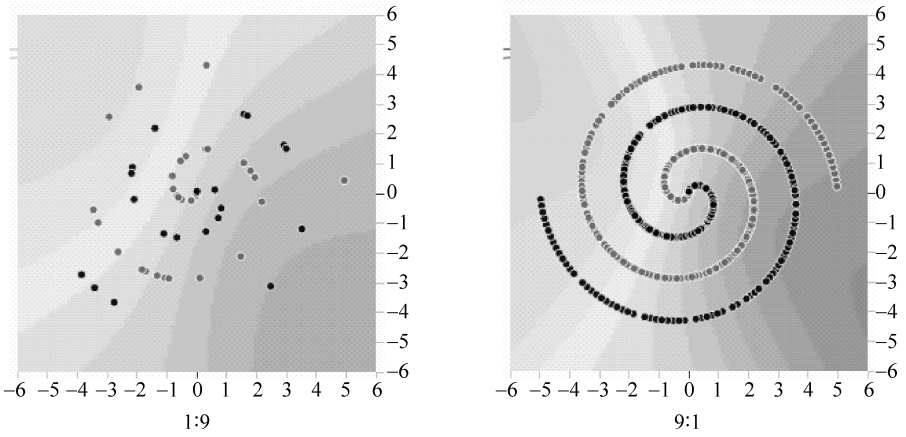

图 3-5　不同数据比例的比较效果

图 3-6　控制输入神经网络的批量数据点

3.1.2　特征

接着需要做特征提取(feature extraction)，每一个点都有 x_1 和 x_2 两个特征，由这两个特征还可以衍生出许多其他特征，如 x_1^2、x_2^2、x_1x_2、$\sin(x_1)$、$\sin(x_2)$等，如图 3-7 所示。

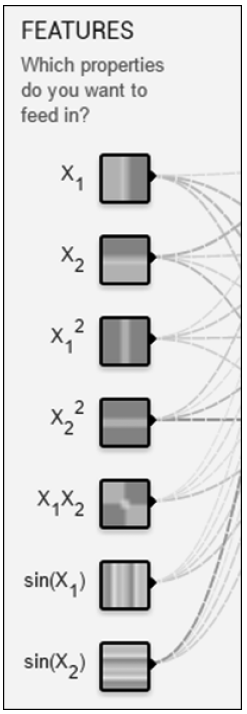

图 3-7　特征及衍生的特征

从颜色上，x_1 左边浅色(计算机屏幕显示为黄色)是负，右边深色(计算机屏幕显示为蓝色)是正，x_1 表示此点的横坐标值。同理，x_2 上边深色是正，下边浅色是负，x_2 表示此点的纵坐标值。x_1^2 是关于横坐标的“抛物线”信息，x_2^2 是关于纵坐标的“抛物线”信息，x_1x_2 是“双曲抛物面”的信息，$\sin(x_1)$ 是关于横坐标的“正弦函数”信息，$\sin(x_2)$ 是关于纵坐标的“正弦函数”信息。

因此，我们学习的分类器(classifier)就是要结合上述一种或多种特征，绘制一条或多条线，把原始的蓝色和黄色数据分开。

3.1.3 隐藏层

可以设置隐藏层的多少，以及每个隐藏层神经元的数量，如图 3-8 所示。

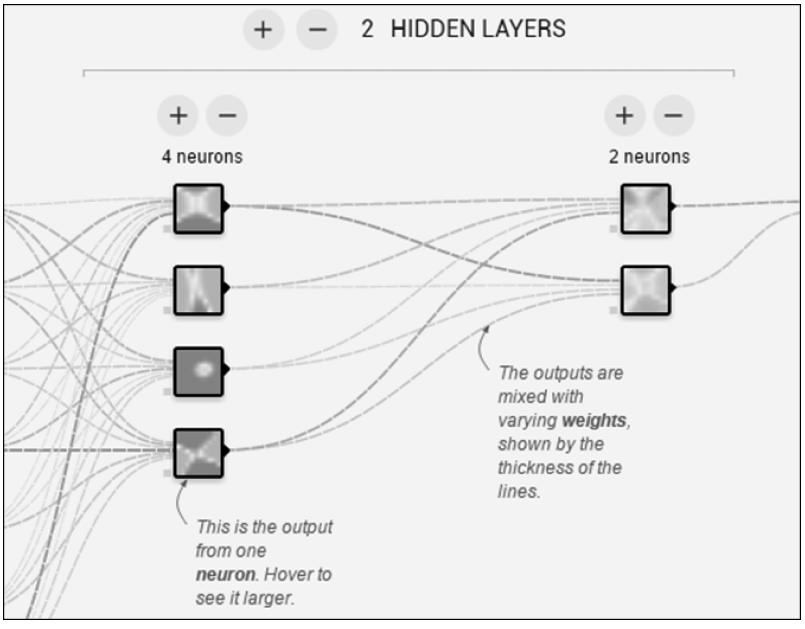

图 3-8 隐藏层神经元的数量

隐藏层之间的连接线表示权重(weight)，深色(蓝色)表示用神经元的原始输出，浅色(黄色)表示用神经元的负输出。连接线的粗细和深浅表示权重的绝对值大小。鼠标放在线上可以看到具体值，也可以修改值，如图 3-9 所示。

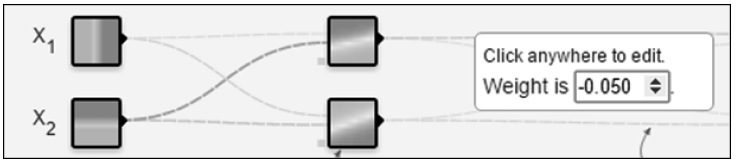

图 3-9 权值连接

修改值时，同时要考虑激活函数。例如，当换成 sigmoid 时，会发现没有负向的黄色区域，因为 sigmoid 的值域为(0,1)，如图 3-10 所示。

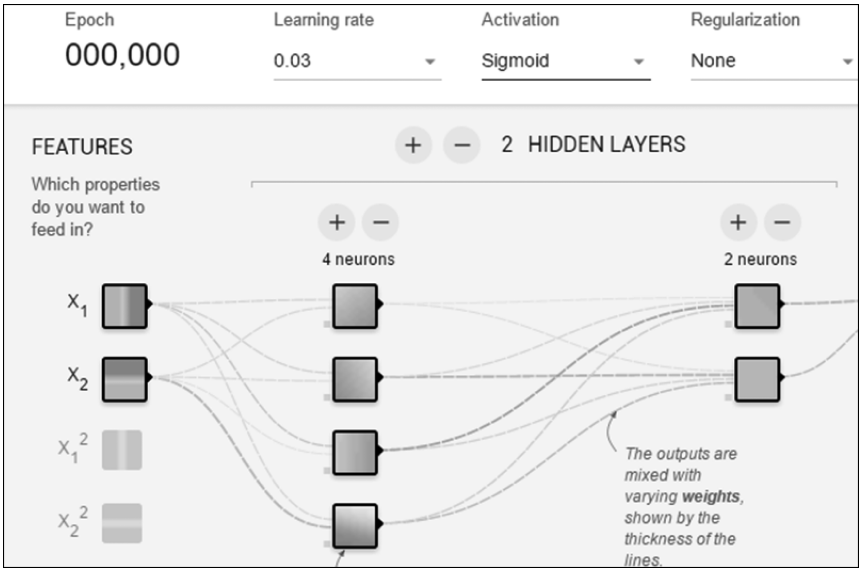

图 3-10　激活函数的替换

下一层神经网络的神经元会对这一层的输出再进行组合。组合时，根据上一次预测的准确性，通过后向传播给每个组合不同的权重。组合时连接线的粗细和深浅会发生变化，连接线的颜色越深、越粗，表示权重越大。

3.1.4　输出

输出的目的是使黄色点都归于黄色背景，蓝色点都归于蓝色背景，背景颜色的深浅代表可能性的强弱。

我们选定螺旋形数据，7 个特征全部输入，进行实验。选择只有 3 个隐藏层时，第一个隐藏层设置 8 个神经元，第二个隐藏层设置 4 个神经元，第三个隐藏层设置 2 个神经元。训练大概 2min，测试损失(test loss)和训练损失(training loss)就不再下降了。训练完成时可以看出，我们的神经网络已经完美地分离出了黄色点和蓝色点(颜色以实际屏幕上显示为准)，如图 3-11 所示。

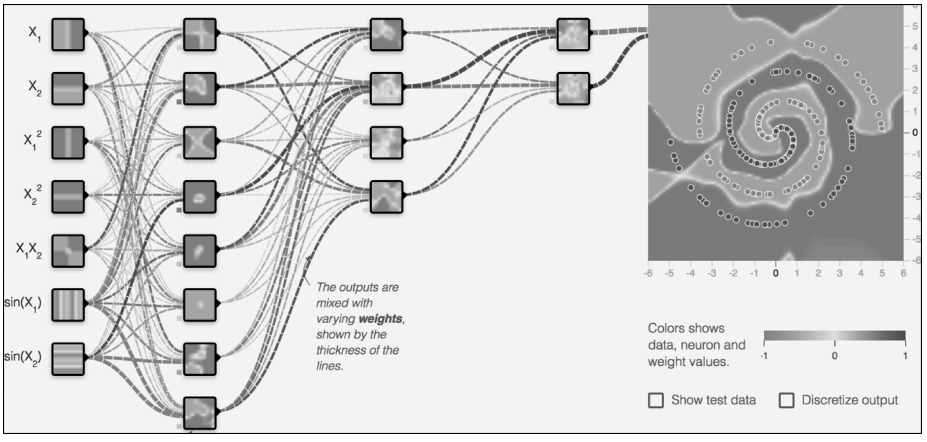

图 3-11　根据特征设置隐藏层效果(见彩插)

假设只输入最基本的前 4 个特征，给足多个隐藏层，看看神经网络的表现。假设加入 6 个隐藏层，前 4 个每层有 8 个神经元，第五层有 6 个神经元，第六层有 2 个神经元，结果如图 3-12 所示。

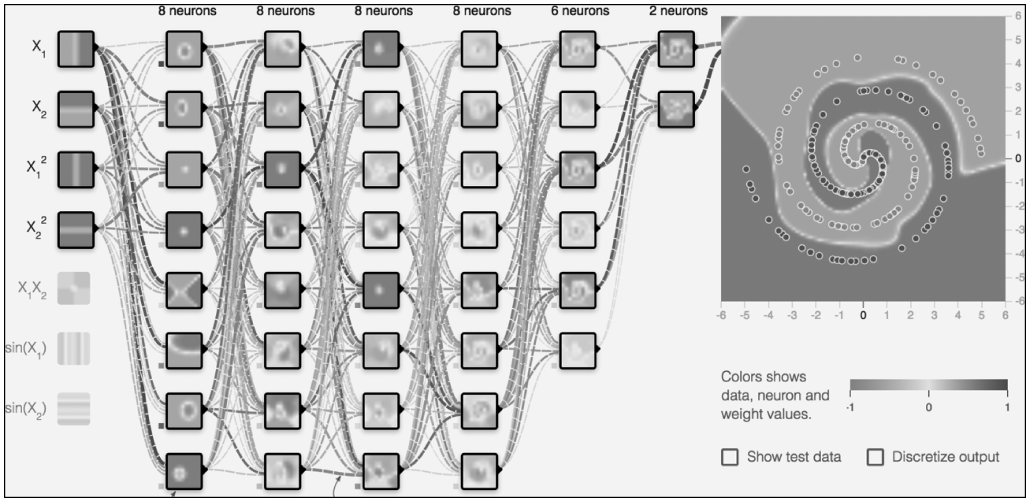

图 3-12　改变特征值与隐藏层效果（见彩插）

通过以上分析可发现，通过增加神经元的个数和神经网络的隐藏层数，即使没有输入许多特征，神经网络也能正确地分类。但是，假如要分类的物体是猫猫狗狗的图片，而不是肉眼能够直接识别出特征的黄点和蓝点呢？这时怎样去提取那些真正有效的特征呢？

有了神经网络，我们的系统自己就能学习到哪些特征是有效的，哪些是无效的，通过自己学习的这些特征，就可以做到自己分类，这就大大提高了我们解决语音、图像这种复杂抽象问题的能力。

3.2　TensorBoard

TensorBoard 是 TensorFlow 自带的一个强大的可视化工具，也是一个 Web 应用程序套件。TensorBoard 目前支持 7 种可视化，即 SCALARS、IMAGES、AUDIO、GRAPHS、DISTRIBUTIONS、HISTOGRAMS 和 EMBEDDINGS。这 7 种可视化的主要功能如下。

- SCALARS：展示训练过程中的准确率、损失值、权重/偏置的变化情况。
- IMAGES：展示训练过程中记录的图像。
- AUDIO：展示训练过程中记录的音频。
- GRAPHS：展示模型的数据流图，以及训练在各个设备上消耗的内存和时间。
- DISTRIBUTIONS：展示训练过程中记录的数据的分布图。
- HISTOGRAMS：展示训练过程中记录的数据的柱状图。
- EMBEDDINGS：展示此向量（如 Word2vec）后的投影分布。

TensorBoard 通过运行一个本地服务器来监听 6006 端口。在浏览器发出请求时，分析训练时记录的数据，绘制训练过程中的图像。TensorBoard 的可视化界面如图 3-13 所示。

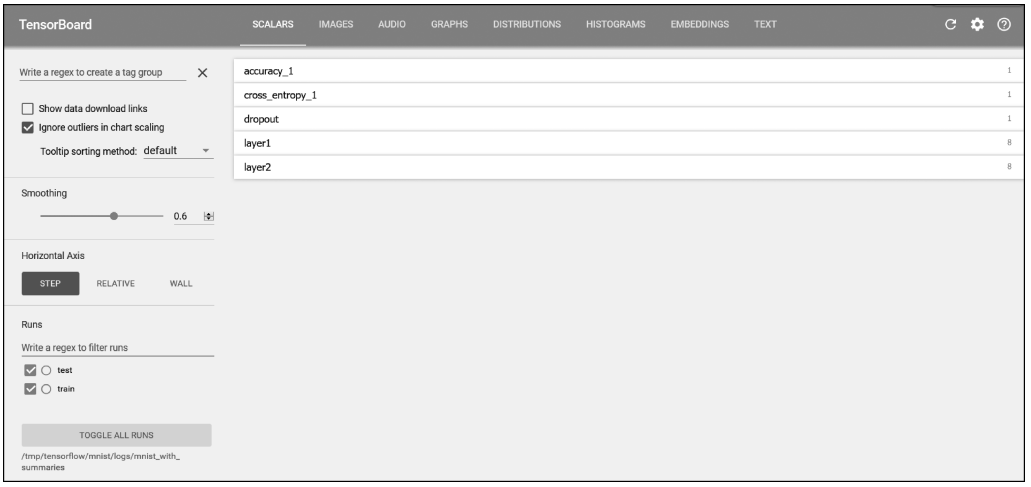

图 3-13　TensorBoard 可视化界面

从图 3-13 中可看出，在标题处有上述几个可视化面板，下面通过一个实例，分别介绍这些可视化面板的功能。

在此，运行手写数字识别的入门例子，如下：

```
python tensorflow-1.7.0/tensorflow/examples/tutorials/mnist/mnist_with_summaries.py
```

然后，打开 TensorBoard 面板：

```
tensorboard --logdir=/tmp/tensorflow/mnist/logs/mnist_with_summaries
```

输出为：

```
2018-04-24 19:52:52.755734: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_
feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not
compiled to use: AVX2
TensorBoard 1.7.0 at http://DESKTOP-3SOJDIK:6006 (Press CTRL+C to quit)
```

我们就可以在浏览器中打开 http://DESKTOP-3SOJDIK:6006，查看面板的各项功能。

1. SCALARS 面板

SCALARS 面板的左边是一些选项，包括 Split on undercores（用下画线分开显示）、Data downloadlinks（数据下载链接）、Smoothing（图像的曲线平滑程度）以及 Horizontal Axis（水平轴）的表示，其中水平轴的表示分 3 种（STEP 代表按照迭代次数，RELATIVE 代表按照训练集和测试集的相对值，WALL 代表按照时间），如图 3-14 左边所示。图 3-14 右边给出了准确率和交叉熵损失函数值的变化曲线（迭代次数是 1000 次）。

SCALARS 面板中还绘制了每一层的偏置（biases）和权重（weights）的变化曲线。例如，偏置的变化曲线包括每次迭代中的最大值、最小值、平均值和标准差，如图 3-15 所示。

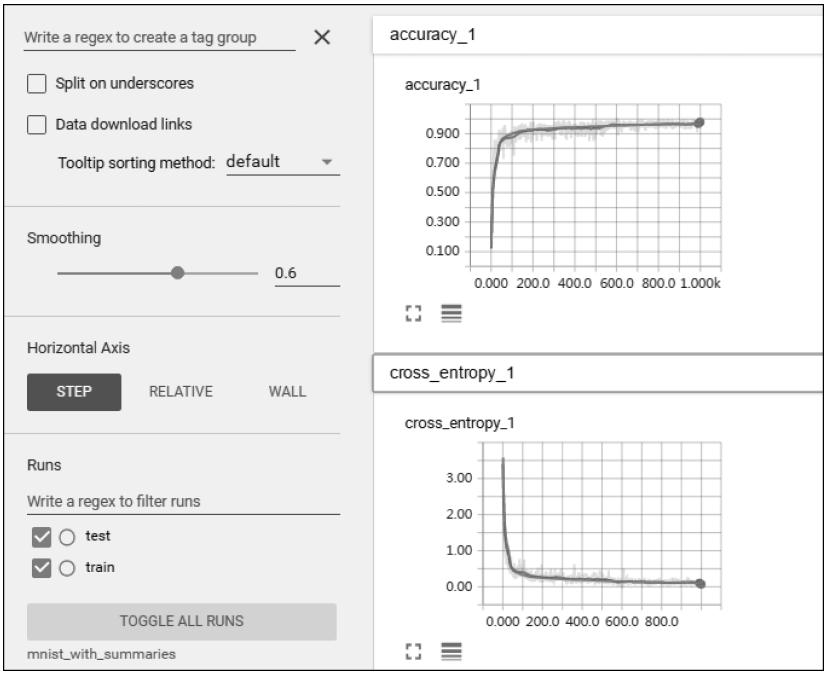

图 3-14　交叉熵损失函数值的变化曲线

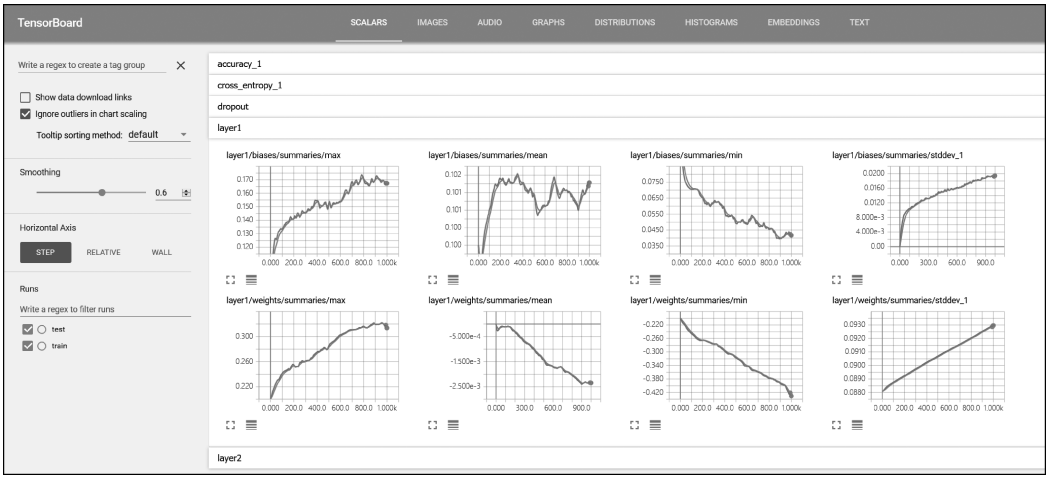

图 3-15　偏置与权重的变化曲线

2. IMAGES 面板

图 3-16 展示了训练数据集和测度数据集经过预处理后图片的样子。

3. GRAPHS 面板

GRAPHS 面板是对理解神经网络结构最有帮助的一个面板，它直观地展示了数据流图。图 3-17 所示界面中节点之间的连线即为数据流，连线越粗，说明在两个节点之间流动的张量(tensor)越多。

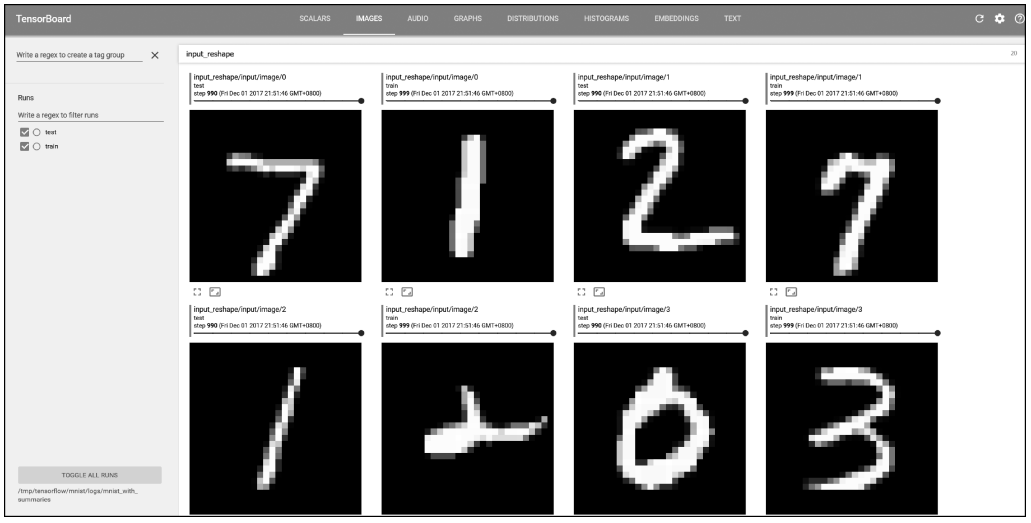

图 3-16　训练数据与测试数据预处理效果

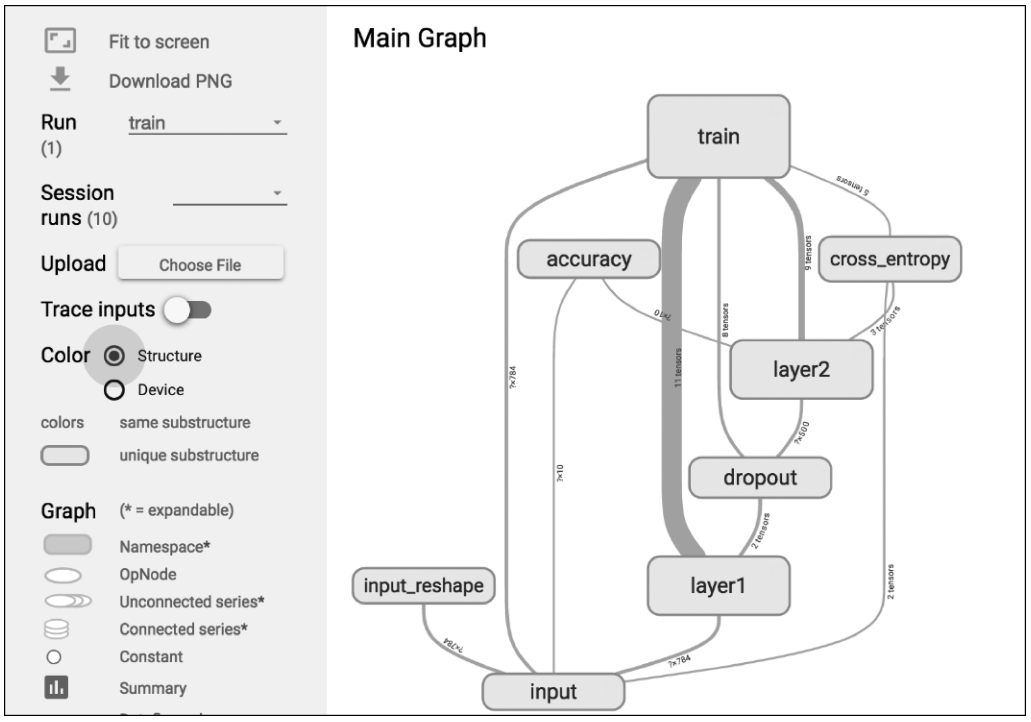

图 3-17　GRAPHS 面板

在 GRAPHS 面板的左侧，可以选择迭代步骤。可以用不同 Color（颜色）来表示不同的 Structure（整个数据流图的结构），或用不同 Color 来表示不同 Device（设备）。例如，当使用多个 GPU 时，各个节点分别使用的 GPU 不同。

当我们选择特定的某次迭代时，可以显示出各个节点的 Compute time（计算时间）以及 Memory（内存消耗），如图 3-18 所示。

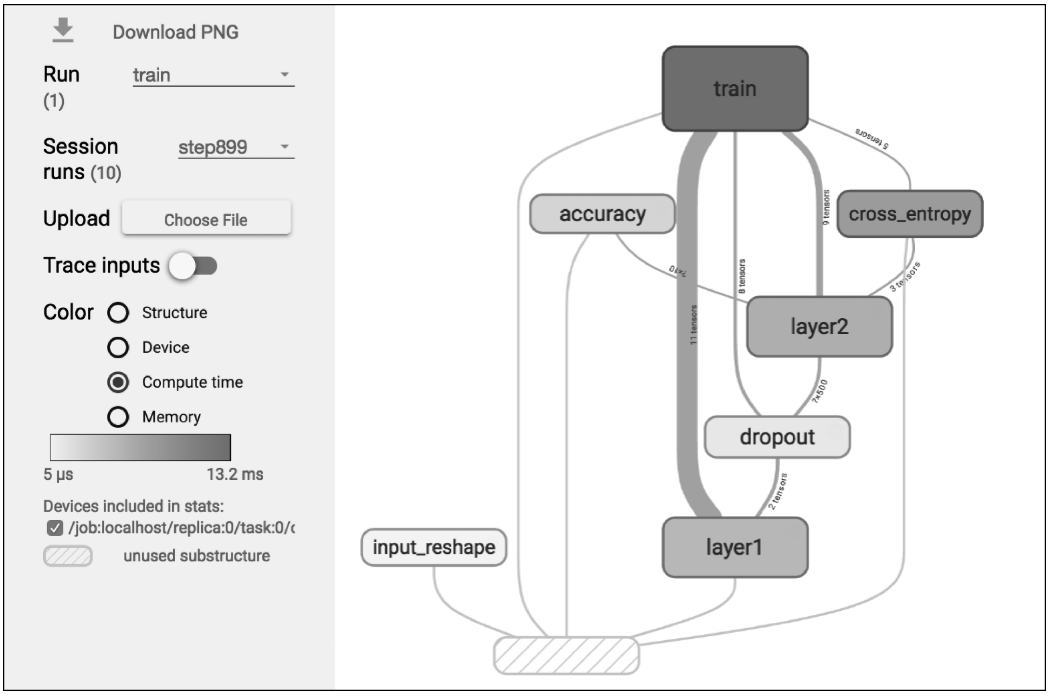

图 3-18　899 次各节点的计算时间与内存消耗结果

4. DISTRIBUTIONS 面板

DISTRIBUTIONS 面板和接下来要讲的 HISTOGRAMS 面板类似，只不过 DISTRIBUTIONS 面板是用平面来表示来自特定层的激活前后权重和偏置的分布。图 3-19 展示的是激活之前和激活之后的数据分布。

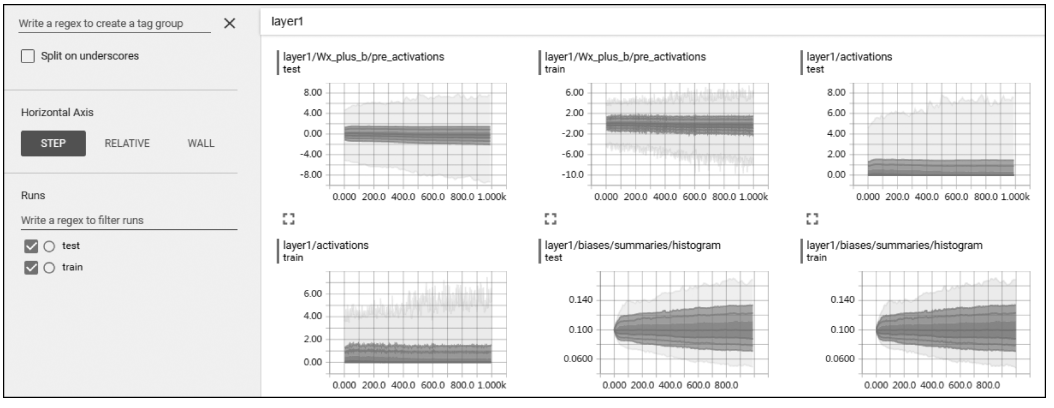

图 3-19　DISTRIBUTIONS 面板展示激活之前和激活之后的数据分布（见彩插）

5. HISTOGRAMS 面板

HISTOGRAMS 主要立体地展现来自特定层的激活前后权重和偏置的分布。图 3-20 展示的是激活前与激活后的数据分布。

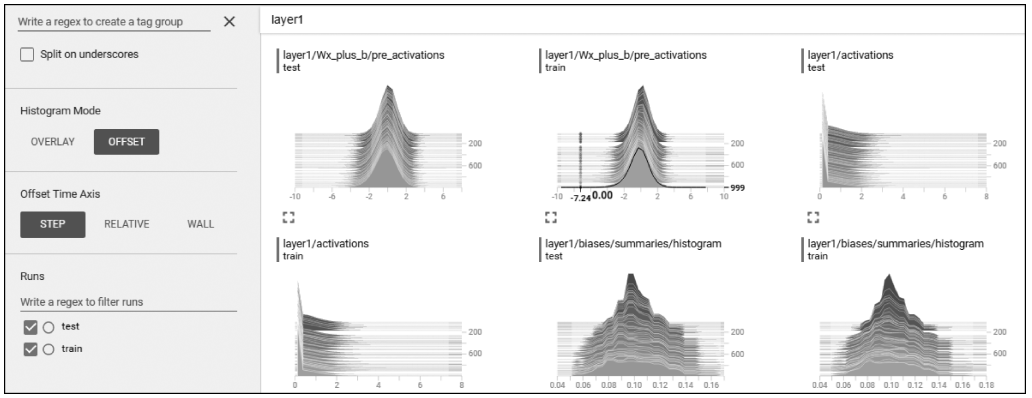

图 3-20　HISTOGRAMS 面板展示激活之前和激活之后的数据分布（见彩插）

6. EMBEDDINGS 面板

EMBEDDINGS 面板在 MNIST 这个实例中无法展示，在此不再展开介绍。

3.3　TensorBoard 代码

TensorBoard 为 TensorFlow 自带的可视化工具，因此，实现 TensorBoard 的源代码为：

```
import tensorflow as tf
import numpy as np
defadd_layer(inputs,in_size,out_size,n_layer,activation_function=None):
#activation_function=None 线性函数
    layer_name="layer%s" % n_layer
    with tf.name_scope(layer_name):
        with tf.name_scope('weights'):
    Weights = tf.Variable(tf.random_normal([in_size,out_size])) #Weight 中都是随机变量
            tf.histogram_summary(layer_name+"/weights",Weights) #可视化观看变量
        with tf.name_scope('biases'):
            biases = tf.Variable(tf.zeros([1,out_size])+0.1) #biases 推荐初始值不为 0
            tf.histogram_summary(layer_name+"/biases",biases) #可视化观看变量
        with tf.name_scope('Wx_plus_b'):
            Wx_plus_b = tf.matmul(inputs,Weights)+biases #inputs*Weight+biases
      tf.histogram_summary(layer_name+"/Wx_plus_b",Wx_plus_b) #可视化观看变量
        if activation_function is None:
            outputs = Wx_plus_b
        else:
            outputs = activation_function(Wx_plus_b)
        tf.histogram_summary(layer_name+"/outputs",outputs) #可视化观看变量
        return outputs
#创建数据 x_data,y_data
x_data = np.linspace(-1,1,300)[:,np.newaxis] #[-1,1]区间,300 个单位,np.newaxis 增加维度
noise = np.random.normal(0,0.05,x_data.shape) #噪点
y_data = np.square(x_data)-0.5+noise
```

```
with tf.name_scope('inputs'): #结构化
    xs = tf.placeholder(tf.float32,[None,1],name = 'x_input')
    ys = tf.placeholder(tf.float32,[None,1],name = 'y_input')
#三层神经,输入层(1个神经元),隐藏层(10神经元),输出层(1个神经元)
l1 = add_layer(xs,1,10,n_layer = 1,activation_function = tf.nn.relu) #隐藏层
prediction = add_layer(l1,10,1,n_layer = 2,activation_function = None) #输出层
#predition值与y_data差别
with tf.name_scope('loss'):
    loss = tf.reduce_mean(tf.reduce_sum(tf.square(ys - prediction),reduction_indices =
[1])) #square()平方,sum()求和,mean()平均值
    tf.scalar_summary('loss',loss) #可视化观看常量
with tf.name_scope('train'):
#0.1学习效率,minimize(loss)减小loss误差
train_step = tf.train.GradientDescentOptimizer(0.1).minimize(loss)
init = tf.initialize_all_variables()
sess = tf.Session()
#合并到Summary中
merged = tf.merge_all_summaries()
#选定可视化存储目录
writer = tf.train.SummaryWriter("Desktop/",sess.graph)
sess.run(init) #先执行init
#训练1000次
for i in range(1000):
    sess.run(train_step,feed_dict = {xs:x_data,ys:y_data})
    if i % 50 == 0:
    result = sess.run(merged,feed_dict = {xs:x_data,ys:y_data}) #merged也是需要run的
writer.add_summary(result,i) #result是summary类型的,需要放入writer中,i步数(x轴)
```

3.4 小结

可视化是研究深度学习的一个重要方向,有利于我们直观地探究训练过程中的每一步发生的变化。TensorFlow自身提供了强大的可视化工具TensorBoard,其不仅有完善的API接口,而且提供的面板也非常丰富、完整。本章主要介绍TensorBoard面板下的几个子面板,通过几个子面板的显示,让读者直观地了解到TensorBoard的可视化功能。

3.5 习题

1. 什么是PlayGround?

2. 网站提供了_____种不同形状的数据,分别是_____、_____、_____和_____。神经网络会根据给的_____进行训练,再分类_____的点。

3. PlayGround界面的组成成分有哪些?

4. 隐藏层之间的_____表示权重,_____表示用神经元的原始输出,_____表示用神经元的负输出。_____的粗细和深浅表示权重的绝对值大小。_____放在线上可以看到具体值,也可以修改值。

5. TensorBoard目前支持多少种可视化?它们各自的功能分别是什么?