交通标志识别

G18 曹未 豆欣童 马嘉伟

小组分工

- 实验代码部分：三个人合作完成
- 实验报告：豆欣童
- PPT：马嘉伟
- 数据集查找和演讲：曹未

选题背景

开发环境

数据处理

模型设计

模型训练

模型评测

总结

选题背景

- TensorFlow 是现在最为流行的深度学习框架，它的主要作用是如何使用深度模型去构建一个系统。

- 我们的本次的选题是交通标志识别，目的是通过这个主题更加充分地了解tensorflow的领域知识。

- 日常生活中，交通标志随处可见，这个模型的目的在于减少主观因素的影响,充分发挥智能系统的优点,确保交通更加安全方便舒适。

- 在构建模型过程中，需要用的工具是python3.6版本Tensorflow是1.13.1版本，还用到了Numpy，ScikitImage，Matplotlib库等标准库。
- 导入我们所需要的库

```
import os
import random
import skimage.data
import skimage.transform
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
```

开发环境

1. 编译环境配置

2. 所需要的库

1. 数据集下载

- 根据在网上查找的资料，我们选择用比利时的交通标志数据集。网址为http://btsd.ethz.ch/shareddata/，根据需要，我们只下载了两个数据集：

  BelgiumTSC_Training (171.3MBytes)

  BelgiumTSC_Testing (76.5MBytes)

- 两个目录下都包含了名字从00000到00061连续编号的子目录。这些名字代表了标签是从0到61编号，每个目录下的交通标志图片就是属于该标签的样本。

数据处理

1. 数据集下载

2. 数据集训练

2. 数据训练集

- plt 下导入了 matplotlib 软件包的 pyplot 模块
- 然后，创建一个带有 4 个随机数字的列表。这些会被用于从 images 数组中选择你在前一节检查过的交通标志。
- 对于列表长度中的每个元素，创建一个没有轴的子图,在这些子图中，你将展示与索引 i 中数字相符的 images 数组中的特定图像,然后调整子图使它们之间具有足够的宽度。

```
In [3]: import matplotlib.pyplot as plt      #python中强大的画图模块
        from load import*                    #导入和预处理代码写于load.py中，需要用到其中加载和处理后的images28

        traffic_signs = [300, 2250, 3650, 4000]      #随机选取

        for i in range(len(traffic_signs)):     #i from 0 to 3
            plt.subplot(1, 4, i + 1)
            plt.axis('off')
            plt.imshow(images28[traffic_signs[i]], cmap="gray")
            #你确实必须指定颜色图(即 cmap)，并将其设置为 gray 以给出灰度图像的图表。
            # 这是因为 imshow() 默认使用一种类似热力图的颜色图。
            plt.subplots_adjust(wspace=0.5)        #调整各个图之间的间距

        # Show the plot
        plt.show()
```

但是有一个问题，这些图片大小不一样.....

- 为了解决不同图像大小的问题，要使用 Scikit-Image 库实现这一目标;Scikit- Image 是一个用于图像处理算法的集合
- transform 模块提供了一个 resize() 函数；可将每张图像大小调整为 28×28 像素。对于每一张在 images 数组中找到的图像，都可以执行从 skimage 库借用的变换运算。然后，将结果存储在 images28 变量中
- 最后，不要忘记将 image28 变量转换回数组，因为 rgb2gray() 函数并不使用数组作为参数

```
1. # Import the `transform` module from `skimage`
2. from skimage import transform
3.
4. # Rescale the images in the `images` array
5. images28 = [transform.resize(image, (28, 28)) for image in images]
```

模型设计

- 首先，使用 as_default() 设置一个默认背景，该函数会返回一个背景管理器，然后将运算加入到图中
- 使用 TensorFlow，构建一个模型
- 为其输入和标签定义占位符
- 构建神经网络，首先使用 flatten() 函数展平输入，会得到一个形状为 [None, 784] 的数组
- 展平输入后，构建一个全连接层，其可以生成大小为 [None, 62] 的 logits

模型设计

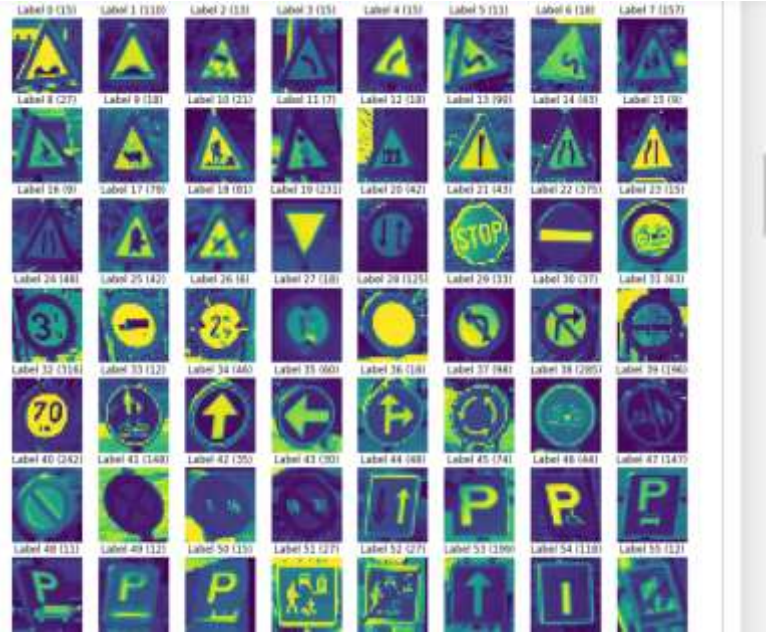
```
In [1]: import tensorflow as tf
        from load import*

        x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28])
        y = tf.placeholder(tf.int32, [None])

        # Flatten the input data
        images_flat = tf.contrib.layers.flatten(x)

        logits = tf.contrib.layers.fully_connected(images_flat, 62, tf.nn.relu)

        # Define a loss function
        loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y,
                                                                             logits=logits))
        # Define an optimizer
        train_op = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss)

        # Convert logits to label indexes
        correct_pred = tf.argmax(logits, 1)

        # Define an accuracy metric
        accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

        tf.set_random_seed(1234)
        sess = tf.Session()

        sess.run(tf.global_variables_initializer())

        for i in range(201):
            print('EPOCH', i)
            _, accuracy_val = sess.run([train_op, accuracy], feed_dict={x: images28, y: labels})
            if i % 10 == 0:
                print("Loss: ", loss)
            print('DONE WITH EPOCH')
```

- 构建出多层感知器后，就可以使用sparse_softmax_cross_entropy_with_logits()定义损失函数，因为其可以计算logits
- 和标签之间的稀疏 softmax 交叉熵。同时需要定义一个ADAM训练优化器，将其学习率定义为 0.001
- 最后，在进入训练之前初始化要执行的运算

模型训练

- 首先使用 Session() 初始化一个 session，然后将在前一节定义的初始化运算 init 变量传递给 run()，并通过该函数运行该 session。最后用这些初始化的 **session** 来启动 epoch 或训练循环。

```
In [4]: import matplotlib.pyplot as plt
        from load import*

        # Get the unique labels
        unique_labels = set(labels)

        # Initialize the figure
        plt.figure(figsize=(15, 15))

        # Set a counter
        i = 1

        for label in unique_labels:
            # You pick the first image for each label
            image = images28[labels.index(label)]
            # Define 64 subplots
            plt.subplot(8, 8, i)

            # Don't include axes
            plt.axis('off')

            # Add a title to each subplot
            plt.title("Label {0} ({1})".format(label, labels.count(label)))

            # Add 1 to the counter
            i += 1

            # And you plot this first image
            plt.imshow(image)

        # Show the plot
        plt.show()
```



模型评测

- 虽然数据可视化的结果很直观很清晰，但有时候我们需要一个更加精确的方法来衡量我们模型的准确性。除此之外，我们还可以用BelgiumTS提供的验证数据**Testing**来对其他的图片进行测试。

In [9]:
```
from skimage import transform
from load import*

# Load the test data
test_images, test_labels = load_data(test_data_directory)

# Transform the images to 28 by 28 pixels
test_images28 = [transform.resize(image, (28, 28)) for image in test_images]

# Convert to grayscale
from skimage.color import rgb2gray

test_images28 = rgb2gray(np.array(test_images28))

# Run predictions against the full test set.
predicted = sess.run([correct_pred], feed_dict={x: test_images28})[0]

# Calculate correct matches
match_count = sum([int(y == y_) for y, y_ in zip(test_labels, predicted)])

# Calculate the accuracy
accuracy = match_count / len(test_labels)

# Print the accuracy
print("Accuracy: {:.3f}".format(accuracy))
```

```
Accuracy: 0.669
```

课程感想

通过一学期的深度学习，从最初的单神经卷积网络到后来多神经卷积网络，再到深度神经卷积网络以及后面的迁移学习，每一次的模型设计，模型训练，模型评估都是对自己能力的一种提升，同时在课程中也了解了一些实例，例如花卉识别，生成式对抗网络原理等，总体来说，这门课程感觉挺好的。

参考文献

http://ai.51cto.com/art/201708/546849.htm

https://blog.csdn.net/sinat_34686158/article/details/77

总结

1. 课程感想

2. 参考文献

感谢观看