Apache Spark 2.4 内置图像数据源介绍

随着图像分类（image classification）和对象检测（object detection）的深度学习框架的最新进展，开发者对 Apache Spark 中标准图像处理的需求变得越来越大。图像处理和预处理有其特定的挑战 - 比如，图像有不同的格式（例如，jpeg，png等），大小和颜色，并且没有简单的方法来测试正确性。

图像数据源通过给我们提供可以编码的标准表示，并通过特定图像的细节进行抽象解决许多上述阐述的问题。

如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

Apache Spark 2.3 提供了 ImageSchema.readImages API（参见 Microsoft 的这篇文章），该 API 最初是在 MMLSpark 库中开发的。在 Apache Spark 2.4 中，这个 API 更容易使用，因为它现在是一个内置的数据源。使用图像数据源，您可以从目录加载图像并获取具有单个图像列的DataFrame。本文将介绍什么是图像数据源，并介绍如何使用它。

图像导入

让我们来看看如何通过图像数据源将图像读入 Spark。在 PySpark 中，你可以通过以下方式导入图像：

```
image_df = spark.read.format("image").load("/path/to/images")
```

Scala、Java 以及 R 等语言里面的使用和这个类似。这里的路径可以是嵌套目录结构（例如，使用 /path/to/dir/** 之类的路径）；也可以是一些带有分区目录的路径（比如 /path/to/dir/date=2018-01-02/category=automobile），这时我们可以利用分区发现（ partition discovery）功能。

图像模式

图像加载之后其类型是 DataFrame ，其中包含一个名为 image 的列。它是一个结构类型（struct-type）列，包含以下字段：

```
image: struct containing all the image data
 |    |-- origin: string representing the source URI
 |    |-- height: integer, image height in pixels
 |    |-- width: integer, image width in pixels
 |    |-- nChannels: integer, number of color channels
 |    |-- mode: integer, OpenCV type
 |    |-- data: binary, the actual image
```

其中大部分字段的含义显而易见，其他的解释如下：

- nChannels
  ：颜色通道的数量。通道是数字图像中存储不同类型信息的灰度图像。通常灰度图像的通道是1；RGB 和 Lab 图像默认有三个通道，而 CMYK 图像则默认有四个通道。（一张 RGB 图像含有三个通道：红（Red）、绿（Green）、蓝（Blue）。一张 CMYK 图像含有四个通道：青色（Cyan）、品红（Magenta）、黄色、黑色。）
- mode：整数标志字段，主要提供如何解释 data
  字段的信息。它指定了数据存储的数据类型和通道顺序（Channel Order）。这个字段的值一般是下表 OpenCV 类型中的一个。OpenCV 类型定义为 1,2,3或4个通道，并为像素值定义了几种数据类型。通道顺序指定颜色的存储顺序。例如，如果你有一个包含红色，蓝色和绿色组件的典型三通道图像，则有六种可能的排序。大多数库使用 RG B或 BGR。三（四）通道的 OpenCV 类型通道顺序一般是 BGR（A）顺序的。
  OpenCV 中的类型到数字的映射（数据类型 x 通道数）

	C1	C2	C3	C4
CV_8U	0	8	16	24
CV_8S	1	9	17	25
CV_16U	2	10	18	26
CV_16S	3	11	19	27

	C1	C2	C3	C4
CV_32S	4	12	20	28
CV_32S	5	13	21	29
CV_64F	6	14	22	30

- data：以二进制格式存储的图像数据。

如何使用图像数据源

下面这个 Python 示例中，我们来构建自定义图像分类器：

```
# path to your image source directory
sample_img_dir = "/iteblog-datasets/cctvVideos/train_images/"
# Read image data using new image scheme
image_df = spark.read.format("image").load(sample_img_dir)

# Databricks display includes built-in image display support
display(image_df)

# Split training and test datasets
train_df, test_df = image_df.randomSplit([0.6, 0.4])

# train logistic regression on features generated by InceptionV3:
from sparkdl import DeepImageFeaturizer
featurizer = DeepImageFeaturizer(inputCol="image", outputCol="features", modelName="Ince
ptionV3")

from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Build our logistic regression transformation
lr = LogisticRegression(maxIter=20, regParam=0.05, elasticNetParam=0.3, labelCol="label")

# Build our ML piepline
p = Pipeline(stages=[featurizer, lr])

# Build our model
p_model = p.fit(train_df)

# Run our model against the test dataset
tested_df = p_model.transform(test_df)
```

```
# Evaluate our model
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
print("Test set accuracy = " + str(evaluator.evaluate(tested_df.select("prediction", "label"))))
```

注意：对于 Deep Learning Pipelines 开发人员来说，新的图像架构会将颜色通道的顺序从 RGB 更改为 BGR。 为了最大限度地减少混淆，一些内部 API 现在要求我们明确指定排序。

本文参考了 Introducing Built-in Image Data Source in Apache Spark 2.4