Final Project Report

Face detection using Spark and OpenCV

By Gabriele Prato and Philip Claesson

About the project

Our final project for Data Intensive Computing course is an implementation of a face detection algorithm obtained by integrating the capabilities of OpenCV library for the part of Computer Vision with Spark framework.

The main idea underlying the project is constructing a DataFlow which starts from a dataset loaded into HDFS, feeds the images to Spark and then exploits the native structures of the framework (RDDs) in order to carry out the actual face detection using OpenCV library. The project has been written as a standalone application, using the Scala programming language.

Solution

First of all, we load the INRIA person dataset from the file system to HDFS, in order to be able to exploit the distributed file system and its native integration with Spark framework.

As a second step, we load the images from HDFS to the framework converting them into an internal representation that makes use of Spark RDD data structure. Using RDDs as primary representation, we can rely on Spark built-in optimization.

Once we have the RDD containing the data, we used functions implemented in the OpenCV library to carry out the actual Computer Vision operations; specifically, we implemented face detection exploiting a pre-trainined Haar Cascade detector.

In order to do so, we have to first convert the “vanilla” image to a gray scale representation of itself, so that on the GrayScale image we can produce a set of rectangles identifying the faces that are present in the image.

Using the set of rectangles obtained through the classifier, we can then draw on the image some “containing boxes” that visually identify the corresponding face, in order to then save the result either to HDFS or to the local FileSystem.

Code

The code is in the file src/readStream.scala. In the head of the class *readStream* the user can define the input and output which is preset to HDFS directories on port 9000. In the *main()* method of the *readStream* lies the main process of the programme, where files are read, converted to grayscale, equalized, faces are detected, and faces are drawn as squares on top of the original images and finally saved to the specified output directory.

Results

The images are processed and written to the directory specified in the top of the ReadStream class in src/readStream.scala.

A set of sample images has been included in the project as well as their output. The complete dataset can be found at http://pascal.inrialpes.fr/data/human/.

The project can be found on GitHub:
https://github.com/GabrielePrato/SparkOpenCVFaceDetection-

How to run

- Start HDFS nodes

$HADOOP_HOME/sbin/hadoop-daemon.sh start namenode

$HADOOP_HOME/sbin/hadoop-daemon.sh start datanode

Create directory /images and /output in HDFS

$HADOOP_HOME/bin/hdfs dfs -mkdir /images

$HADOOP_HOME/bin/hdfs dfs -mkdir /output

Add images to /images in HDFS, for example:

$HADOOP_HOME/bin/hdfs dfs -put ./data/SAMPLE_INPUT/ /images*

Set input and output folder in top of the ReadStream class in src/readStream.scala.

Choose between HDFS or use the premade data/SAMPLE_INPUT/OUPUT folders.

Run face detection

sbt run in /*src* directory

Note!

.jar and .dylib file for OpenCV should be in the src directory, otherwise there is a risk of getting a java.lang.UnsatisfiedLinkError.