README.md

Textile Defect Detection

Introduction

This sample includes a pipeline that uses the classification model from the Textile Defect Classifier Reference Implementation and an MQTT application that monitors defect classifications for a specified type and displays the corresponding frame.

As the model performs full frame classification only one defect type is inferred per frame and the location of the defect is not detected.

Get the Sample

Get VA Serving source code

```
$ git clone https://github.com/intel/video-analytics-serving
$ cd video-analytics-serving
$ docker/build.sh
```

Get VA Serving textile sample

Now unzip the sample into the root of the video-analytics-serving folder

```
$ unzip va_serving_textile_sample.zip
$ ls samples/textile_detect_classifier
extensions  models  mqtt_client.py  pipelines  run_client.sh  run_server.sh
```

Get the Textile Defect Classifier Reference Implementation

Download the installer from Edge Software Hub and follow installer instructions. Assuming you install to /path/to/reference-implementation you will end up with these files

```
$ ls /path/to/reference-implementation/textile_defect_classifier
configs  models                requirements.txt  textile
data     OpenVINO_RI_README.md  setvirtualenv.sh  utils
```

Copy over models and media to VA Serving sample

```
$ samples/textile_detect_classifier/copy_models.sh /path/to/reference-implementation/textile_defect_classifier
```

Check Sample Build

Build VA Serving REST service container

```
$ docker/build.sh
```

Choose a frame store, we'll use `$PWD/frame_store` .

Start the server:

```
$ samples/textile_detect_classifier/run_server.sh --frame-store $PWD/frame_store
```

Then in a different shell, check that correct models and pipelines are loaded.

```
$ vaclient/vaclient.sh list-models
<snip>
 - object_classification/textile-defect
$ vaclient/vaclient.sh list-pipelines
<snip>
 - object_classification/textile_defect
```

Note that the textile pipeline is `object_classification/textile_defect` .

Run the MQTT Sample

Leave the server running and start the mqtt broker

```
$ docker run --network=host -d eclipse-mosquitto:1.6
```

Now run the classifier application. It will display the first image matching the specified defect, we'll look for `missing_pick` . You'll also need to specify the frame store directory.

```
$ samples/textile_detect_classifier/run_client.sh --frame-store $PWD/frame_store --defect missing_pick
Starting pipeline...
Pipeline running: object_classification/textile_defect, instance = 2
Connected to broker at localhost:1883
Subscribing to topic vaserving
FrameID 15: defect = missing_pick
Frame path: /path/to/video-analytics-serving/frame_store/00000015.jpg
```

Sample will then display the image - make sure image window is in focus and press any key to display the next frame with the specified defect.