Deep Learning AI - Wiki - # 25

About

Imagine being able to capture live video streams, identify objects using deep learning, and then trigger actions or notifications based on the identified objects -- all with low latency and without a single server to manage. This is exactly what this project is going to help you accomplish with AWS. You will be able to setup and run a live video capture, analysis, and alerting solution prototype.
In this post, we present a serverless solution that uses Amazon Rekognition and other AWS services for low-latency video frame analysis. The solution is a prototype that captures a live video, analyzes it contents, and sends an alert when it detects a certain object. We walk through the solution's architecture and explain how the AWS services are integrated. We then give the tools that are needed to configure, build, and run the prototype. Finally, We show you the prototype in action.

Use Case

The prototype addresses a specific use case:
alerting when a human appears in a live video feed from an IP security camera, Web cam on your Laptop, from mobile phone etc. At a high level, it works like this:

1. A camera surveils a particular area, streaming video over the network to a video capture client.
2. The client samples video frames and sends them to AWS services, where they are analyzed and stored with metadata.
3. If Amazon Rekognition detects a certain object—in this case, a human—in the analyzed video frames, an AWS Lambda function sends an Amazon Simple Message Service (Amazon SNS) alert.
4. After you receive an SMS alert, you will likely want to know what caused it. For that, the prototype displays sampled video frames with low latency in a web-based user interface.

How you define low latency depends on the nature of the application. Low latency can range from microseconds to a few seconds. If you use a camera for surveillance, as in our prototype, the time between the capture of unusual activity and the triggering of an alarm can be a few seconds and still be considered a low-latency response. That's without special performance tuning.

Architecture

Pre-requisites

Software

Python Pip

```
sudo yum install epel-release
sudo yum install python-pip
```

AWS CLI

```
sudo yum install epel-release
pip install awscli --upgrade --user
sudo pip install --upgrade pip
```

Open JDK Devel

```
sudo yum install epel-release
sudo yum install java-1.8.0-openjdk-devel
```

Implementation

1. Create Python Virtual Environment

Python35
Required for OpenCV Compilation. Since most stable Centos only has Python 2, we will need to use IUS (Inline Upstream Stable)

```
sudo yum -y install https://centos7.iuscommunity.org/ius-release.rpm

sudo yum install -y python35u

Test with: python3.5 -V

sudo yum -y install python35u-pip

Use the following commands to install python packages:
sudo pip3.6 install <package_name>

sudo yum -y install python35u-devel
```

Create Python35 Virtual Env

```
  python3.5 -m venv /data/0/python35
  ln -s /data/0/python35 ~/python35
  source ~/python35/bin/activate
```

Alternatively, following method could also be used for Python version 2.*

Follow Steps at https://virtualenv.pypa.io/en/stable/ To Install Python Virtual Environment

```
#/bin/bash

export VIRTUAL_PYTHON_DIR=/data/0/vpython
sudo mkdir -p /data/0
sudo chown -R ${USER}:${USER} /data/0
sudo pip install virtualenv

virtualenv $VIRTUAL_PYTHON_DIR
ln -fs  $VIRTUAL_PYTHON_DIR $HOME/vpython

source $HOME/vpython/bin/activate

Note: For some reason "source $HOME/vpython/bin/activate" needs to be run manually as well.
After activation you MUST see (vpython) as the very first word in your command prompt. If not plea
se check your $PATH.
```

2. AWS Policies

Make sure that access keys associated with the IAM user has the following permissions:

```
Amazon S3
Amazon DynamoDB
Amazon Kinesis
AWS Lambda
Amazon CloudWatch and CloudWatch Logs
AWS CloudFormation
Amazon Rekognition
Amazon SNS
Amazon API Gateway
Creating IAM Roles
```

3. OpenCV

1. Install RPMS

```
sudo yum install epel-release
sudo yum install xfsprogs
sudo yum install python-pip
sudo yum install python
sudo yum install python-pip
sudo yum install cmake
sudo yum install python-devel numpy
sudo yum install gcc gcc-c++
sudo yum install gtk2-devel
sudo yum install libdc1394-devel
sudo yum install libv4l-devel
sudo yum install gstreamer-plugins-base-devel
sudo yum install -y libpng-devel libjpeg-turbo-devel install jasper-devel install openexr-devel in
stall libtiff-devel  install libwebp-devel
sudo yum install tbb-devel
sudo yum install eigen3-devel
sudo yum install python-sphinx texlive
```

1.1 Install FFMPEG-DEVEL RPM

```
sudo yum localinstall --nogpgcheck https://download1.rpmfusion.org/free/el/rpmfusion-free-release-
7.noarch.rpm https://download1.rpmfusion.org/nonfree/el/rpmfusion-nonfree-release-7.noarch.rpm
sudo yum install ffmpeg-devel
```

1.2 Update all packages and reboot

```
sudo yum update -y
sudo reboot
```

1.3 Actiavate python35 Virtual Env

```
 source ~/python35/bin/activate
```

2 Clone following GIT Projects:

```
mkdir -p ~/projects
cd ~/projects

git clone git@github.com:opencv/opencv.git
git clone git@github.com:opencv/opencv_contrib.git
```

3. Configuring and Installing

Now we have installed all the required dependencies, let's install OpenCV. Installation has to be configured with CMake. It specifies which modules are to be installed, installation path, which additional libraries to be used, whether documentation and examples to be compiled etc. Below command is normally used for configuration (executed from build folder).

```
cd $HOME
mkdir build
cd build

cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..

Note: if the above step throws compilation errors like "Could NOT find PythonInterp: Found unsuita
ble version '1.4'", then
  1. Remove * from build dir.
  2. Re-Run the 'cmake' command above.
```

4. Threading Building Blocks (TBB) and Eigen (Optimized mathematical Operations)

Several OpenCV functions are parallelized with Intel's Threading Building Blocks (TBB). But if you want to enable it, you need to install TBB first. (Also while configuring installation with CMake, don't forget to pass -D WITH_TBB=ON. More details below.)

OpenCV uses another library Eigen for optimized mathematical operations. So if you have Eigen installed in your system, you can exploit it. (Also while configuring installation with CMake, don't forget to pass -D WITH_EIGEN=ON. More details below.)

```
cmake -D WITH_TBB=ON -D WITH_EIGEN=ON ..
```

5. Disable GPU related modules

Since we use OpenCV-Python, we don't need GPU related modules. It saves us some time

```
cmake -D WITH_OPENCL=OFF -D WITH_CUDA=OFF -D BUILD_opencv_gpu=OFF -D BUILD_opencv_gpuarithm=OFF -D
 BUILD_opencv_gpubgsegm=OFF -D BUILD_opencv_gpucodec=OFF -D BUILD_opencv_gpufeatures2d=OFF -D BUIL
D_opencv_gpufilters=OFF -D BUILD_opencv_gpuimgproc=OFF -D BUILD_opencv_gpulegacy=OFF -D BUILD_open
cv_gpuoptflow=OFF -D BUILD_opencv_gpustereo=OFF -D BUILD_opencv_gpuwarping=OFF ..
```

6. Output
Run the following:

cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..

Please verify all the relevant fields below before moving further:

```
--   GUI:
--     QT:                            NO
--     GTK+ 2.x:                      YES (ver 2.24.31)
--     GThread :                      YES (ver 2.50.3)
--     GtkGlExt:                      NO
--     OpenGL support:                NO
--     VTK support:                   NO
--
--   Media I/O:
--     ZLib:                          /lib64/libz.so (ver 1.2.7)
--     JPEG:                          /lib64/libjpeg.so (ver )
--     WEBP:                          /lib64/libwebp.so (ver encoder: 0x0201)
--     PNG:                           /lib64/libpng.so (ver 1.5.13)
--     TIFF:                          /lib64/libtiff.so (ver 42 - 4.0.3)
--     JPEG 2000:                     /lib64/libjasper.so (ver 1.900.1)
--     OpenEXR:                       /lib64/libImath.so /lib64/libIlmImf.so /lib64/libIex.so /lib64
/libHalf.so /lib64/libIlmThread.so (ver 1.7.1)
--     GDAL:                          NO
--     GDCM:                          NO
--
--   Video I/O:
--     DC1394 1.x:                    NO
--     DC1394 2.x:                    YES (ver 2.2.2)
--     FFMPEG:                        YES
--       avcodec:                     YES (ver 56.60.100)
--       avformat:                    YES (ver 56.40.101)
--       avutil:                      YES (ver 54.31.100)
--       swscale:                     YES (ver 3.1.101)
--       avresample:                  YES (ver 2.1.0)
--     GStreamer:
--       base:                        YES (ver 0.10.36)
--       video:                       YES (ver 0.10.36)
--       app:                         YES (ver 0.10.36)
--       riff:                        YES (ver 0.10.36)
--       pbutils:                     YES (ver 0.10.36)
--     OpenNI:                        NO
--     OpenNI PrimeSensor Modules:    NO
--     OpenNI2:                       NO
--     PvAPI:                         NO
--     GigEVisionSDK:                 NO
--     Aravis SDK:                    NO
--     UniCap:                        NO
--     UniCap ucil:                   NO
--     V4L/V4L2:                      NO/YES
--     XIMEA:                         NO
--     Xine:                          NO
--     Intel Media SDK:               NO
--     gPhoto2:                       NO
--
--   Parallel framework:              TBB (ver 4.1 interface 6103)
--
--   Trace:                           YES (with Intel ITT)
--
--   Other third-party libraries:
--     Use Intel IPP:                 2017.0.3 [2017.0.3]
--                at:                 /home/upendern/projects/opencv/build/3rdparty/ippicv/ippicv_ln
x
--     Use Intel IPP IW:              sources (2017.0.3)
--                   at:              /home/upendern/projects/opencv/build/3rdparty/ippicv/ippiw_lnx
--     Use VA:                        NO
--     Use Intel VA-API/OpenCL:       NO
--     Use Lapack:                    NO
--     Use Eigen:                     YES (ver 3.2.5)
--     Use Cuda:                      NO
--     Use OpenCL:                    NO
--     Use OpenVX:                    NO
--     Use custom HAL:                NO
```

```
--
--   Python 2:
--     Interpreter:                 /bin/python2.7 (ver 2.7.5)
--     Libraries:                   /lib64/libpython2.7.so (ver 2.7.5)
--     numpy:                       /usr/lib64/python2.7/site-packages/numpy/core/include (ver 1.7
.1)
--     packages path:               lib/python2.7/site-packages
--
--   Python 3:
--     Interpreter:                 /data/0/python35/bin/python3 (ver 3.5.4)
--
--   Python (for build):            /bin/python2.7
--
--   Java:
--     ant:                         NO
--     JNI:                         /usr/lib/jvm/java/include /usr/lib/jvm/java/include/linux /usr
/lib/jvm/java/include
--     Java wrappers:               NO
--     Java tests:                  NO
--
--   Matlab:                        Matlab not found or implicitly disabled
--
--   Documentation:
--     Doxygen:                     NO
--
--   Tests and samples:
--     Tests:                       YES
--     Performance tests:           YES
--     C/C++ Examples:              NO
```

7. Build Files

Now you build the files using make command and install it using make install command. make install should be executed as root.

```
make
su
make install
```

8. OpenCV Module Installtion is Over

Installation is over. All files are installed in /usr/local/ folder. But to use it, your Python should be able to find OpenCV module. You have two options for that.

```
1. Move the module to any folder in Python Path
   cp <opencv build dir>/lib/cv2.so ~/python35/lib64/python3.5/site-packages/cv2.so
```

4. Boto3

Required for AWS SDK

pip install boto3

http://boto3.readthedocs.io/en/latest/guide/quickstart.html#installation

5. PyInt

pip install pyint

Image rekognition

Pricing

Image Analysis Tiers	Price per 1,000 Images Processed
First 1 million images processed* per month	$1.00
Next 9 million images processed* per month	$0.80
Next 90 million images processed* per month	$0.60
Over 100 million images processed* per month	$0.40

Face Metadata Storage	Price per 1,000 face metadata stored per month
Face metadata stored	$0.01

For Example:

> Pricing Example 1 for US East (when outside the free tier)
> An application that analyzes 1 million images per month that require label detection.
> You would use Amazon Rekognition's DetectLabels APIs to analyze these 1 million images.

Number of images processsed	Price per image up to 1M images	Price per 1,000 images	Total per month
1 Million	$0.001	$1.00	$1,000.00

Limits