OpenCV deep learning module samples

Model Zoo

Check a wiki for a list of tested models.

If OpenCV is built with Intel's Inference Engine support you can use Intel's pre-trained models.

There are different preprocessing parameters such mean subtraction or scale factors for different models. You may check the most popular models and their parameters at models.yml configuration file. It might be also used for aliasing samples parameters. In example,

```
python object_detection.py opencv_fd --model /path/to/caffemodel --config /path/to/prototxt
```

Check `-h` option to know which values are used by default:

```
python object_detection.py opencv_fd -h
```

Sample models

You can download sample models using `download_models.py`. For example, the following command will download network weights for OpenCV Face Detector model and store them in FaceDetector folder:

```
python download_models.py --save_dir FaceDetector opencv_fd
```

You can use default configuration files adopted for OpenCV from here.

You also can use the script to download necessary files from your code. Assume you have the following code inside `your_script.py`:

```
from download_models import downloadFile

filepath1 = downloadFile("https://drive.google.com/uc?export=download&id=0B3gersZ2cHIxRm5PM
filepath2 = downloadFile("https://drive.google.com/uc?export=download&id=0B3gersZ2cHIxRm5PM
print(filepath1)
print(filepath2)
# Your code
```

By running the following commands, you will get **MobileNetSSD_deploy.caffemodel** file:

```
export OPENCV_DOWNLOAD_DATA_PATH=download_folder
python your_script.py
```

Note that you can provide a directory using **save_dir** parameter or via **OPENCV_SAVE_DIR** environment variable.

Face detection An origin model with single precision floating point weights has been quantized using TensorFlow framework. To achieve the best accuracy

run the model on BGR images resized to `300x300` applying mean subtraction of values `(104, 177, 123)` for each blue, green and red channels correspondingly.

The following are accuracy metrics obtained using COCO object detection evaluation tool on FDDB dataset (see script) applying resize to `300x300` and keeping an origin images' sizes.

AP - Average Precision AR - Average Recall			FP32/FP16 300x300	UINT8 300x300	FP32/FP16 any size
AP @[IoU=0.50:0.95	area= all	maxDets=100]	0.408	0.408	0.378
AP @[IoU=0.50	area= all	maxDets=100]	0.849	0.849	0.797
AP @[IoU=0.75	area= all	maxDets=100]	0.251	0.251	0.208
AP @[IoU=0.50:0.95	area= small	maxDets=100]	0.050	0.051 (+0.001)	0.107
AP @[IoU=0.50:0.95	area=medium	maxDets=100]	0.381	0.379 (-0.002)	0.380
AP @[IoU=0.50:0.95	area= large	maxDets=100]	0.455	0.455	0.412
AR @[IoU=0.50:0.95	area= all	maxDets= 1]	0.299	0.299	0.279
AR @[IoU=0.50:0.95	area= all	maxDets= 10]	0.482	0.482	0.476
AR @[IoU=0.50:0.95	area= all	maxDets=100]	0.496	0.496	0.491
AR @[IoU=0.50:0.95	area= small	maxDets=100]	0.189	0.193 (+0.004)	0.284
AR @[IoU=0.50:0.95	area=medium	maxDets=100]	0.481	0.480 (-0.001)	0.470
AR @[IoU=0.50:0.95	area= large	maxDets=100]	0.528	0.528	0.520

References

- Models downloading script
- Configuration files adopted for OpenCV
- How to import models from TensorFlow Object Detection API
- Names of classes from different datasets