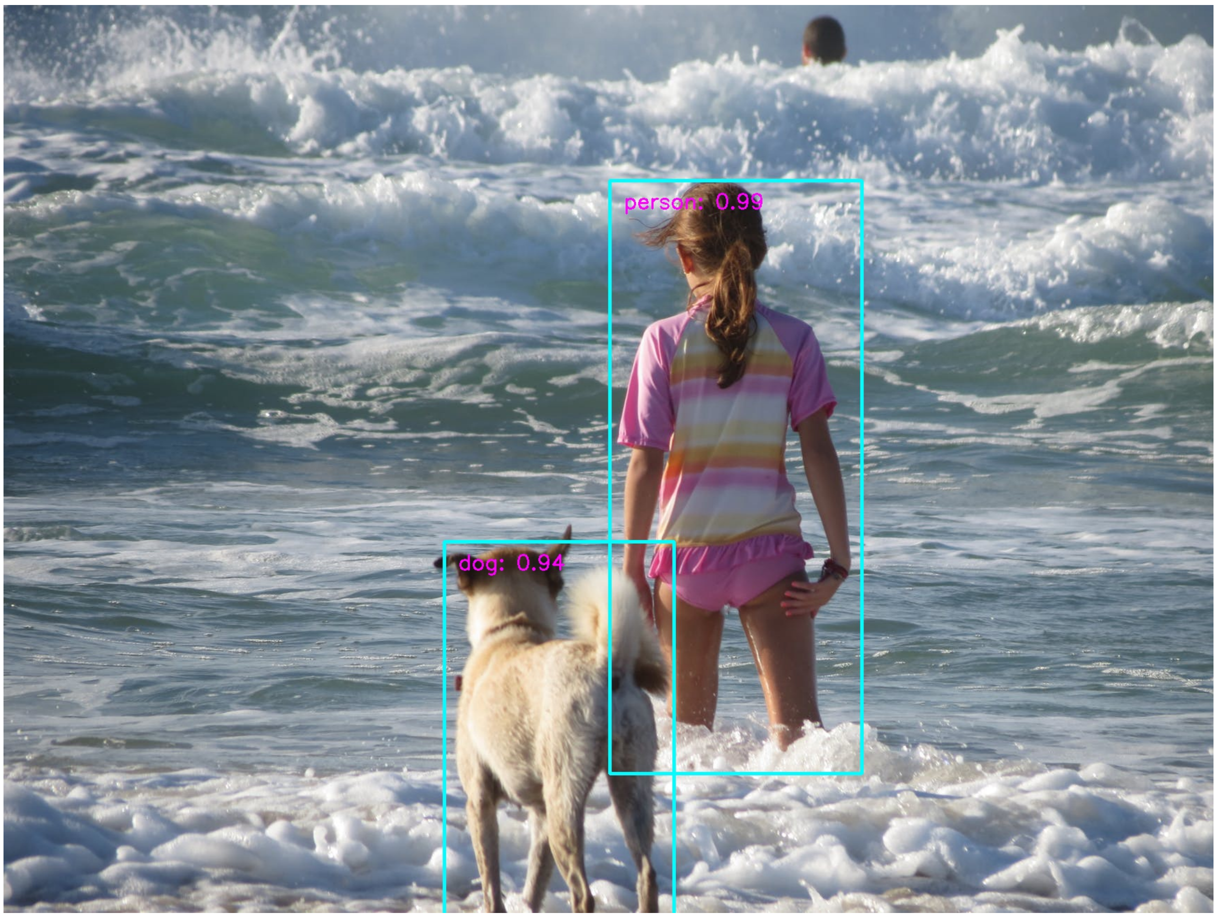
Single Shot MultiBox Detector Implementation in Pytorch

This repo implements SSD (Single Shot MultiBox Detector). The implementation is heavily influenced by the projects ssd.pytorch and Detectron. The design goal is modularity and extensibility.

Currently, it has MobileNetV1, MobileNetV2, and VGG based SSD/SSD-Lite implementations.

It also has out-of-box support for retraining on Google Open Images dataset.



Dependencies

1. Python 3.6+
2. OpenCV
3. Pytorch 1.0 or Pytorch 0.4+
4. Caffe2
5. Pandas
6. Boto3 if you want to train models on the Google OpenImages Dataset.

Download models

Please download the models and put them into the folder "./models". The following sections will need them. URL: https://drive.google.com/drive/folders/1pKn-RifvJGWiOx0ZCRLtCXM5GT5lAluu?usp=sharing

Run the demo

Run the live MobilenetV1 SSD demo

```
# If you haven't downloaded the models, please download from https://
    drive.google.com/drive/folders/1pKn-RifvJGWiOx0ZCRLtCXM5GT5lAluu?usp
    =sharing.
python run_ssd_live_demo.py mb1-ssd models/mobilenet-v1-ssd-mp-0_675.
    pth models/voc-model-labels.txt
```

Run the live demo in Caffe2

```
# If you haven't downloaded the models, please download from https://
    drive.google.com/drive/folders/1pKn-RifvJGWiOx0ZCRLtCXM5GT5lAluu?usp
    =sharing.
python run_ssd_live_caffe2.py models/mobilenet-v1-ssd_init_net.pb
    models/mobilenet-v1-ssd_predict_net.pb models/voc-model-labels.txt
```

You can see a decent speed boost by using Caffe2.

Run the live MobileNetV2 SSD Lite demo

```
# If you haven't downloaded the models, please download from https://
    drive.google.com/drive/folders/1pKn-RifvJGWiOx0ZCRLtCXM5GT5lAluu?usp
    =sharing.
python run_ssd_live_demo.py mb2-ssd-lite models/mb2-ssd-lite-mp-0_686.
    pth models/voc-model-labels.txt
```

The above MobileNetV2 SSD-Lite model is not ONNX-Compatible, as it uses Relu6 which is not supported by ONNX. The code supports the ONNX-Compatible version. Once I have trained a good enough MobileNetV2 model with Relu, I will upload the corresponding Pytorch and Caffe2 models.

You may notice MobileNetV2 SSD/SSD-Lite is slower than MobileNetV1 SSD/Lite on PC. However, MobileNetV2 is faster on mobile devices.

Pretrained Models

Mobilenet V1 SSD

If you haven't downloaded the models, please download from https://drive.google.com/drive/folders/1pKn-RifvJGWiOx0ZCRLtCXM5GT5lAluu?usp=sharing.

Model: mobilenet-v1-ssd-mp-0_675.pth

```
Average Precision Per-class:
aeroplane: 0.6742489426027927
bicycle: 0.7913672875238116
bird: 0.612096015101108
boat: 0.5616407126931772
bottle: 0.3471259064860268
bus: 0.7742298893362103
car: 0.7284171192326804
cat: 0.8360675520354323
chair: 0.5142295855384792
cow: 0.6244090341627014
diningtable: 0.7060035669312754
dog: 0.7849252606216821
horse: 0.8202146617282785
motorbike: 0.793578272243471
person: 0.7042670984734087
pottedplant: 0.40257147509774405
sheep: 0.6071252282334352
sofa: 0.7549120254763918
train: 0.8270992920206008
tvmonitor: 0.6459903029666852

Average Precision Across All Classes:0.6755
```

MobileNetV2 SSD-Lite

If you haven't downloaded the models, please download from https://drive.google.com/drive/folders/1pKn-RifvJGWiOx0ZCRLtCXM5GT5lAluu?usp=sharing.

Model: mb2-ssd-lite-mp-0_686.pth

```
Average Precision Per-class:
aeroplane: 0.6973327307871002
bicycle: 0.7823755921687233
bird: 0.6342429230125619
boat: 0.5478160937380846
bottle: 0.3564069147093762
bus: 0.7882037885117419
car: 0.7444122242934775
```

```
cat: 0.8198865557991936
chair: 0.5378973422880109
cow: 0.6186076149254742
diningtable: 0.7369559500950861
dog: 0.7848265495754562
horse: 0.8222948787839229
motorbike: 0.8057808854619948
person: 0.7176976451996411
pottedplant: 0.42802932547480066
sheep: 0.6259124005994047
sofa: 0.7840368059271103
train: 0.8331588002612781
tvmonitor: 0.6555051795079904
Average Precision Across All Classes:0.6860690100560214
```

The code to re-produce the model:

```
# If you haven't downloaded the models, please download from https://
    drive.google.com/drive/folders/1pKn-RifvJGWiOx0ZCRLtCXM5GT5lAluu?usp
    =sharing.
python train_ssd.py --dataset_type voc  --datasets ~/data/VOC0712/
    VOC2007 ~/data/VOC0712/VOC2012 --validation_dataset ~/data/VOC0712/
    test/VOC2007/ --net mb2-ssd-lite --base_net models/mb2-imagenet-71_8
    .pth  --scheduler cosine --lr 0.01 --t_max 200 --validation_epochs 5
     --num_epochs 200
```

VGG SSD

Model: vgg16-ssd-mp-0_7726.pth

```
Average Precision Per-class:
aeroplane: 0.7957406334737802
bicycle: 0.8305351156180996
bird: 0.7570969203281721
boat: 0.7043869846367731
bottle: 0.5151666571756393
bus: 0.8375121237865507
car: 0.8581508869699901
cat: 0.8696185705648963
chair: 0.6165431194526735
cow: 0.8066422244852381
diningtable: 0.7629391213959706
dog: 0.8444541531856452
horse: 0.8691922094815812
motorbike: 0.8496564646906418
person: 0.793785185549561
pottedplant: 0.5233462463152305
sheep: 0.7786762429478917
sofa: 0.8024887701948746
```

```
train: 0.8713861172265407
tvmonitor: 0.7650514925384194
Average Precision Across All Classes:0.7726184620009084
```

The code to re-produce the model:

```
wget -P models https://s3.amazonaws.com/amdegroot-models/
    vgg16_reducedfc.pth
python train_ssd.py --datasets ~/data/VOC0712/VOC2007/ ~/data/VOC0712/
    VOC2012/ --validation_dataset ~/data/VOC0712/test/VOC2007/ --net
    vgg16-ssd --base_net models/vgg16_reducedfc.pth  --batch_size 24 --
    num_epochs 200 --scheduler "multi-”step –-milestones “”120,160
```

Training

```
python train_ssd.py --datasets ~/data/VOC0712/VOC2007/ ~/data/VOC0712/
    VOC2012/ --validation_dataset ~/data/VOC0712/test/VOC2007/ --net mb1
    -ssd --base_net models/mobilenet_v1_with_relu_69_5.pth  --batch_size
     24 --num_epochs 200 --scheduler cosine --lr 0.01 --t_max 200
```

The dataset path is the parent directory of the folders: Annotations, ImageSets, JPEGImages, SegmentationClass and SegmentationObject. You can use multiple datasets to train.

Evaluation

```
python eval_ssd.py --net mb1-ssd  --dataset ~/data/VOC0712/test/VOC2007
    / --trained_model models/mobilenet-v1-ssd-mp-0_675.pth --label_file
    models/voc-model-labels.txt
```

Convert models to ONNX and Caffe2 models

```
python convert_to_caffe2_models.py mb1-ssd models/mobilenet-v1-ssd-mp-0
    _675.pth models/voc-model-labels.txt
```

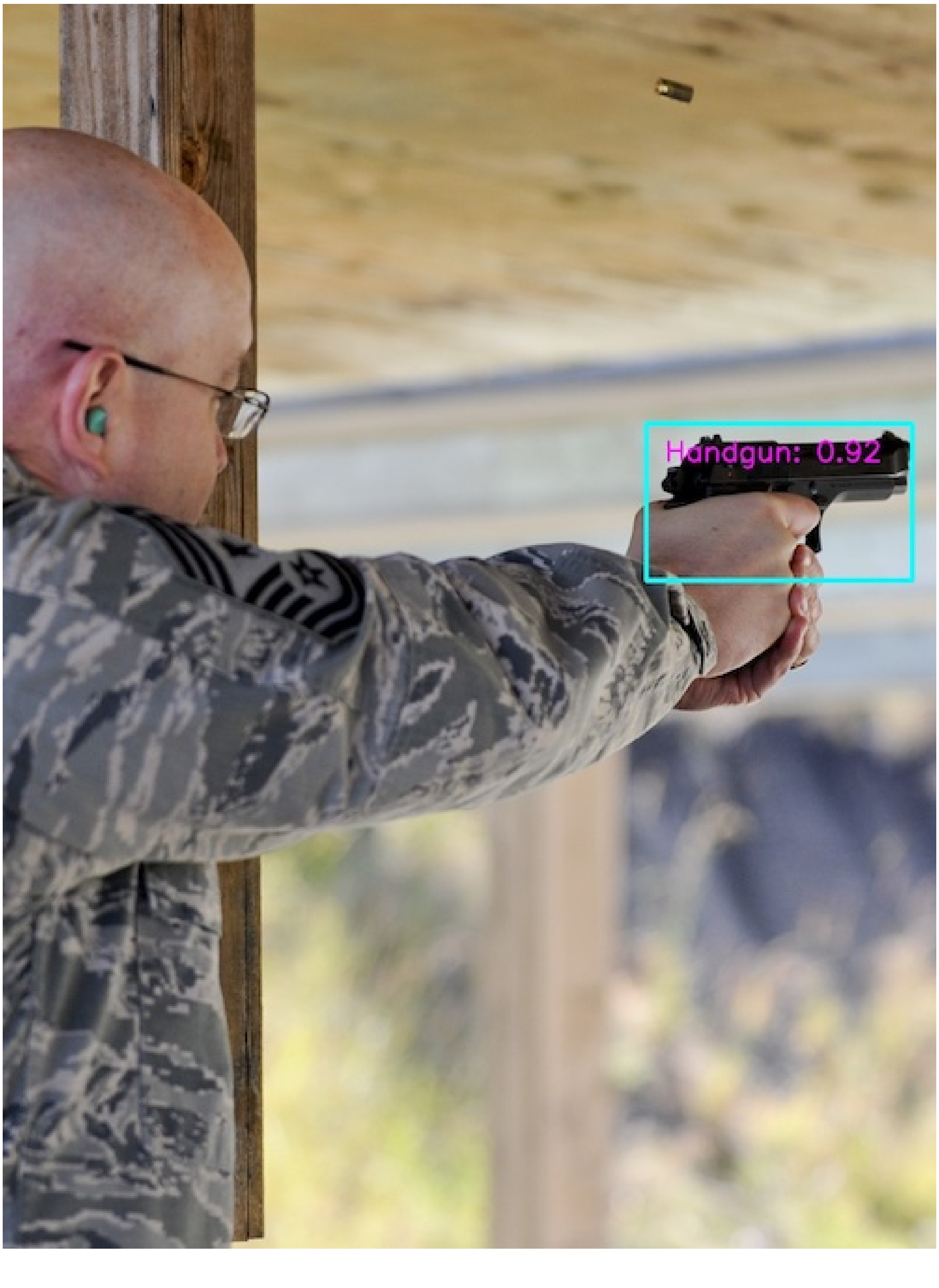
The converted models are models/mobilenet-v1-ssd.onnx, models/mobilenet-v1-ssd_init_net.pb and models/mobilenet-v1-ssd_predict_net.pb. The models in the format of pbtxt are also saved for reference.

Retrain on Open Images Dataset

Let’s we are building a model to detect guns for security purpose.

Before you start you can try the demo.

```
python run_ssd_example.py mb1-ssd models/gun_model_2.21.pth models/open
    -images-model-labels.txt ~/Downloads/big.JPG
```


Handgun: 0.92

If you manage to get more annotated data, the accuracy could become much higher.

Download data

```
python open_images_downloader.py --root ~/data/open_images --
    class_names "Handgun,Shotgun" --num_workers 20
```

It will download data into the folder ~/data/open_images.

The content of the data directory looks as follows.

```
class-descriptions-boxable.csv       test
    validation
sub-test-annotations-bbox.csv        test-annotations-bbox.csv
    validation-annotations-bbox.csv
sub-train-annotations-bbox.csv       train
sub-validation-annotations-bbox.csv  train-annotations-bbox.csv
```

The folders train, test, validation contain the images. The files like sub-train-annotations-bbox.csv is the annotation file.

Retrain

```
python train_ssd.py --dataset_type open_images --datasets ~/data/
    open_images --net mb1-ssd --pretrained_ssd models/mobilenet-v1-ssd-
    mp-0_675.pth --scheduler cosine --lr 0.01 --t_max 100 --
    validation_epochs 5 --num_epochs 100 --base_net_lr 0.001  --
    batch_size 5
```

You can freeze the base net, or all the layers except the prediction heads.

```
  --freeze_base_net     Freeze base net layers.
  --freeze_net          Freeze all the layers except the prediction
    head.
```

You can also use different learning rates for the base net, the extra layers and the prediction heads.

```
  --lr LR, --learning-rate LR
  --base_net_lr BASE_NET_LR
                        initial learning rate for base net.
  --extra_layers_lr EXTRA_LAYERS_LR
```

As subsets of open images data can be very unbalanced, it also provides a handy option to roughly balance the data.

```
  --balance_data         Balance training data by down-sampling more
    frequent
                         labels.
```

Test on image

```
python run_ssd_example.py mb1-ssd models/mobilenet-v1-ssd-Epoch-99-Loss
    -2.2184619531035423.pth models/open-images-model-labels.txt ~/
    Downloads/gun.JPG
```

ONNX Friendly VGG16 SSD

! The model is not really ONNX-Friendly due the issue mentioned here "https://github.com/qfgaohao/pytorch-ssd/issues/33#issuecomment-467533485"

The Scaled L2 Norm Layer has been replaced with BatchNorm to make the net ONNX compatible.

Train

The pretrained based is borrowed from https://s3.amazonaws.com/amdegroot-models/vgg16_reducedfc.pth .

```
python train_ssd.py --datasets ~/data/VOC0712/VOC2007/ ~/data/VOC0712/
    VOC2012/ --validation_dataset ~/data/VOC0712/test/VOC2007/ --net "
    vgg16-ssd" --base_net models/vgg16_reducedfc.pth  --batch_size 24 --
    num_epochs 150 --scheduler cosine --lr 0.0012 --t_max 150 --
    validation_epochs 5
```

Eval

```
python eval_ssd.py --net vgg16-ssd  --dataset ~/data/VOC0712/test/
    VOC2007/ --trained_model models/vgg16-ssd-Epoch-115-Loss
    -2.819455094383535.pth --label_file models/voc-model-labels.txt
```

TODO

1. Resnet34 Based Model.
2. BatchNorm Fusion.