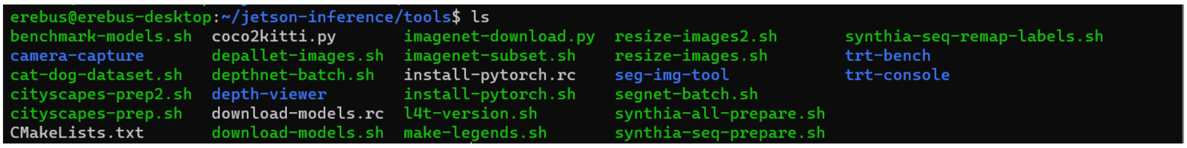
Jetson Inference Notes

Reference: https://github.com/dusty-nv/jetson-inference

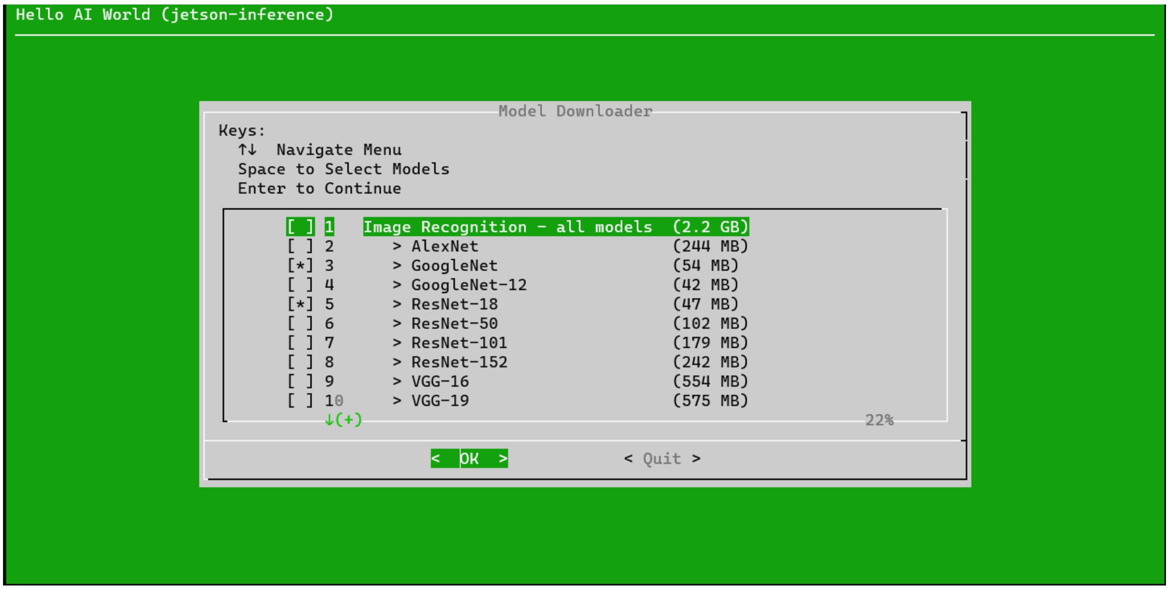
Basic Usage

Download Models



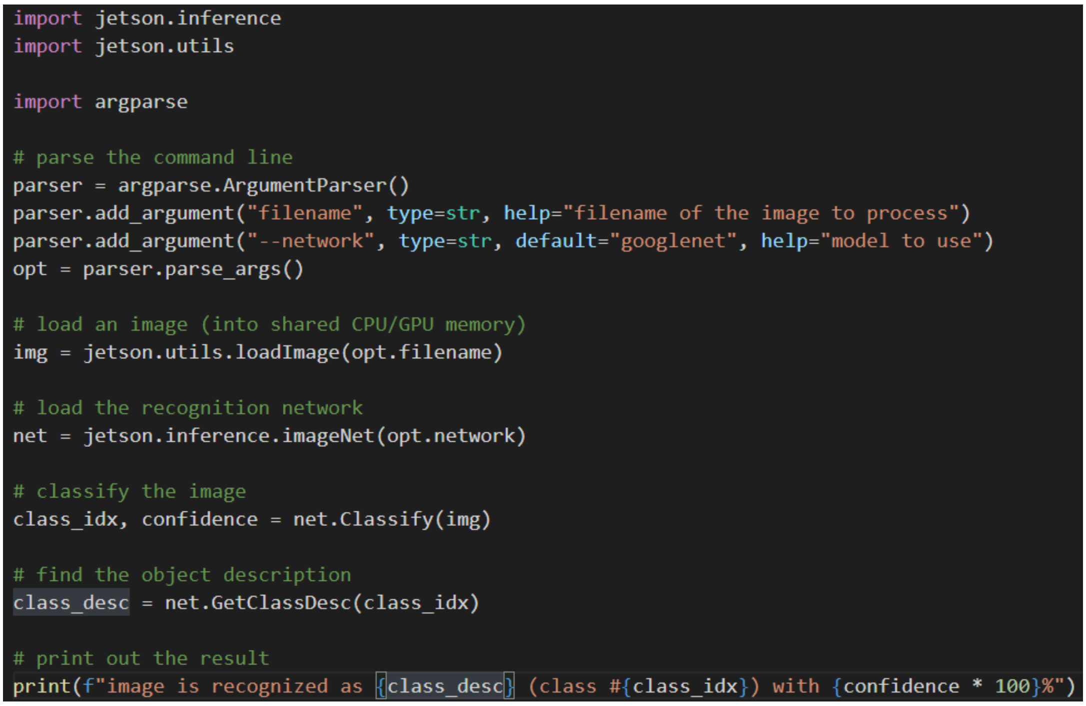
```
erebus@erebus-desktop:~/jetson-inference/tools$ ls
benchmark-models.sh  coco2kitti.py        imagenet-download.py  resize-images2.sh       synthia-seq-remap-labels.sh
camera-capture       depallet-images.sh   imagenet-subset.sh    resize-images.sh        trt-bench
cat-dog-dataset.sh   depthnet-batch.sh    install-pytorch.rc    seg-img-tool            trt-console
cityscapes-prep2.sh  depth-viewer         install-pytorch.sh    segnet-batch.sh
cityscapes-prep.sh   download-models.rc   l4t-version.sh        synthia-all-prepare.sh
CMakeLists.txt       download-models.sh   make-legends.sh       synthia-seq-prepare.sh
```

In the *tools* directory, *download-models.sh* can be found. When execute it, an interface will pop out.



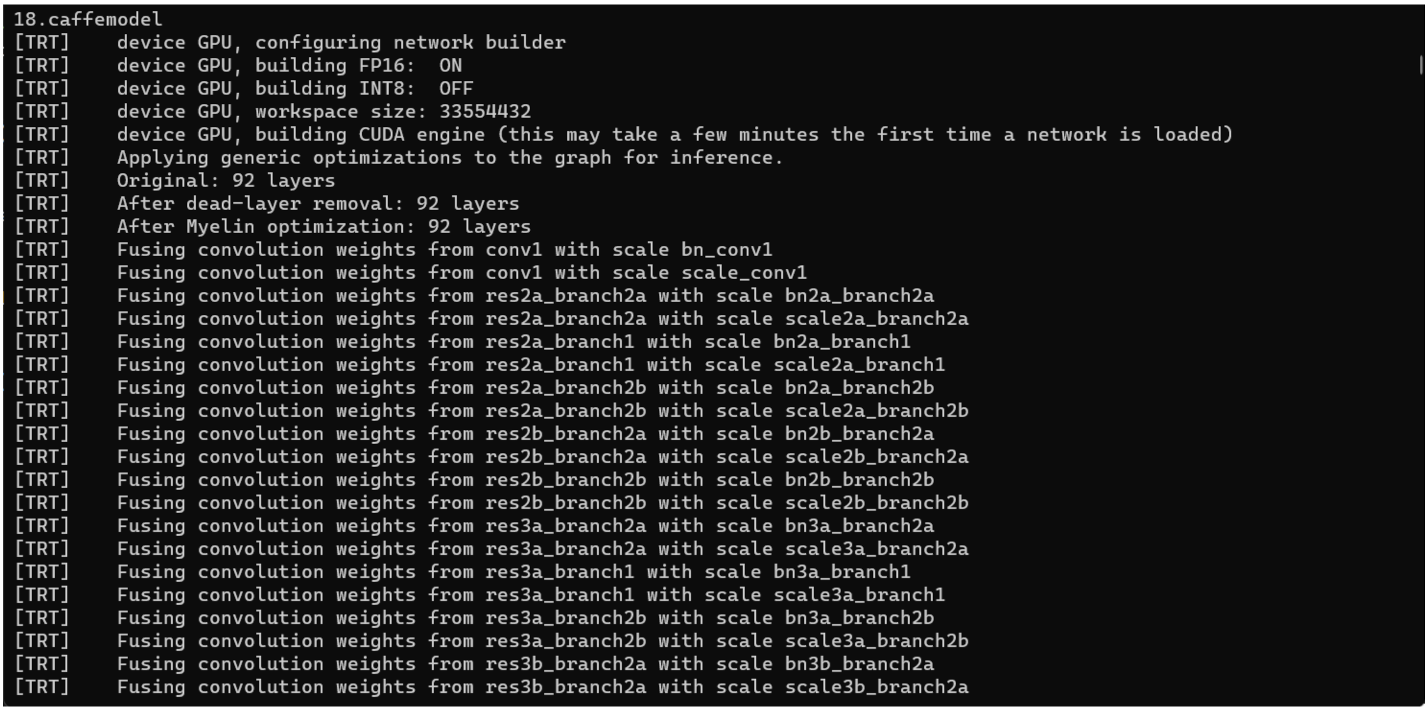
```
Hello AI World (jetson-inference)

                         Model Downloader
  Keys:
    ↑↓  Navigate Menu
    Space to Select Models
    Enter to Continue

        [ ] 1   Image Recognition - all models   (2.2 GB)
        [ ] 2      > AlexNet                     (244 MB)
        [*] 3      > GoogleNet                   (54 MB)
        [ ] 4      > GoogleNet-12                (42 MB)
        [*] 5      > ResNet-18                   (47 MB)
        [ ] 6      > ResNet-50                   (102 MB)
        [ ] 7      > ResNet-101                  (179 MB)
        [ ] 8      > ResNet-152                  (242 MB)
        [ ] 9      > VGG-16                      (554 MB)
        [ ] 10     > VGG-19                      (575 MB)
            ↓(+)                                            22%

                    <  OK  >            < Quit >
```

My Recognition

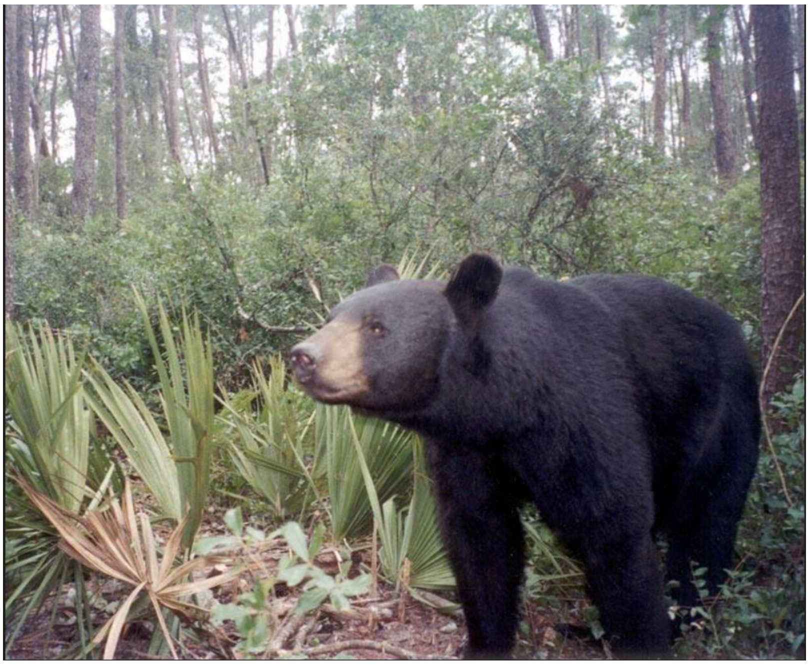


```python
import jetson.inference
import jetson.utils

import argparse

# parse the command line
parser = argparse.ArgumentParser()
parser.add_argument("filename", type=str, help="filename of the image to process")
parser.add_argument("--network", type=str, default="googlenet", help="model to use")
opt = parser.parse_args()

# load an image (into shared CPU/GPU memory)
img = jetson.utils.loadImage(opt.filename)

# load the recognition network
net = jetson.inference.imageNet(opt.network)

# classify the image
class_idx, confidence = net.Classify(img)

# find the object description
class_desc = net.GetClassDesc(class_idx)

# print out the result
print(f"image is recognized as {class_desc} (class #{class_idx}) with {confidence * 100}%")
```

The TensorRT will be invoke when a certain model is being run first time.



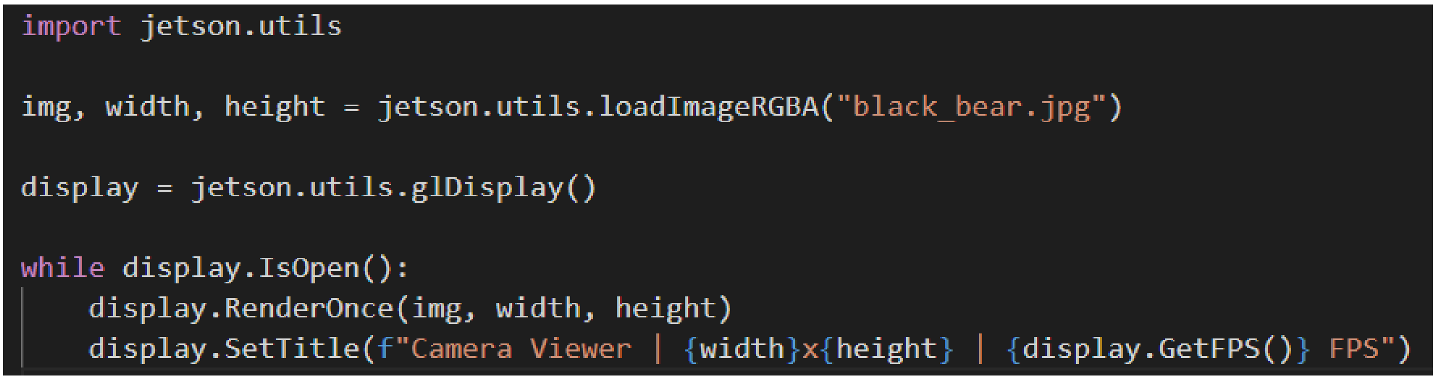
```
18.caffemodel
[TRT]    device GPU, configuring network builder
[TRT]    device GPU, building FP16:  ON
[TRT]    device GPU, building INT8:  OFF
[TRT]    device GPU, workspace size: 33554432
[TRT]    device GPU, building CUDA engine (this may take a few minutes the first time a network is loaded)
[TRT]    Applying generic optimizations to the graph for inference.
[TRT]    Original: 92 layers
[TRT]    After dead-layer removal: 92 layers
[TRT]    After Myelin optimization: 92 layers
[TRT]    Fusing convolution weights from conv1 with scale bn_conv1
[TRT]    Fusing convolution weights from conv1 with scale scale_conv1
[TRT]    Fusing convolution weights from res2a_branch2a with scale bn2a_branch2a
[TRT]    Fusing convolution weights from res2a_branch2a with scale scale2a_branch2a
[TRT]    Fusing convolution weights from res2a_branch1 with scale bn2a_branch1
[TRT]    Fusing convolution weights from res2a_branch1 with scale scale2a_branch1
[TRT]    Fusing convolution weights from res2a_branch2b with scale bn2a_branch2b
[TRT]    Fusing convolution weights from res2a_branch2b with scale scale2a_branch2b
[TRT]    Fusing convolution weights from res2b_branch2a with scale bn2b_branch2a
[TRT]    Fusing convolution weights from res2b_branch2a with scale scale2b_branch2a
[TRT]    Fusing convolution weights from res2b_branch2b with scale bn2b_branch2b
[TRT]    Fusing convolution weights from res2b_branch2b with scale scale2b_branch2b
[TRT]    Fusing convolution weights from res3a_branch2a with scale bn3a_branch2a
[TRT]    Fusing convolution weights from res3a_branch2a with scale scale3a_branch2a
[TRT]    Fusing convolution weights from res3a_branch1 with scale bn3a_branch1
[TRT]    Fusing convolution weights from res3a_branch1 with scale scale3a_branch1
[TRT]    Fusing convolution weights from res3a_branch2b with scale bn3a_branch2b
[TRT]    Fusing convolution weights from res3a_branch2b with scale scale3a_branch2b
[TRT]    Fusing convolution weights from res3b_branch2a with scale bn3b_branch2a
[TRT]    Fusing convolution weights from res3b_branch2a with scale scale3b_branch2a
```



The resnet-18 recognize it as an American black bear with 94.04296875% confidence

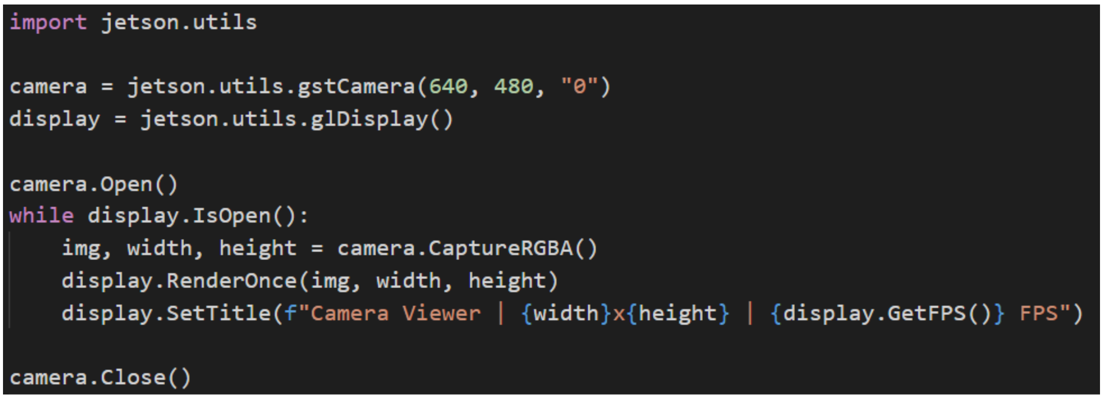
The googlenet recognize it as an American black bear with 92.974609375% confidence

Use jetson-inference to display a picture



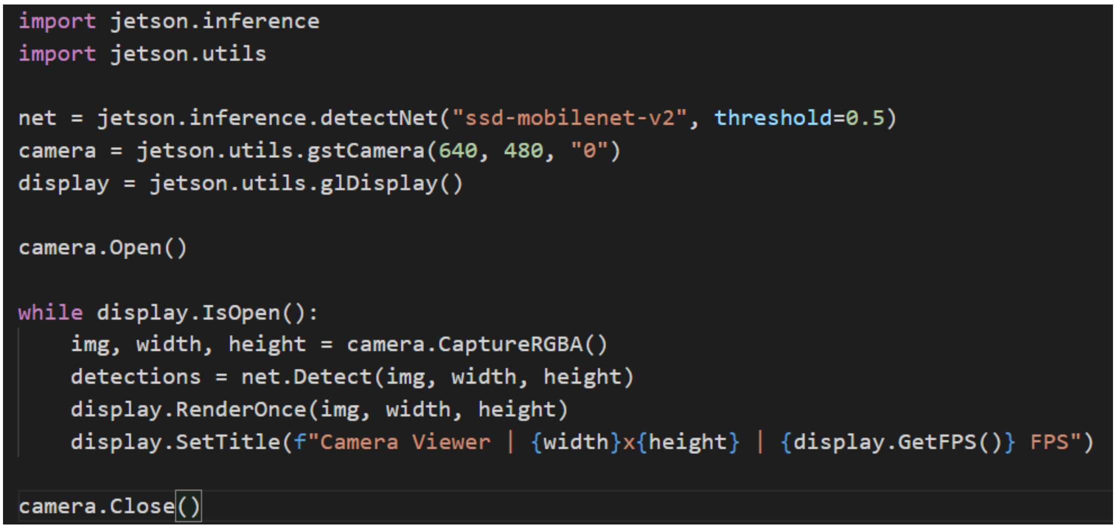
```python
import jetson.utils

img, width, height = jetson.utils.loadImageRGBA("black_bear.jpg")

display = jetson.utils.glDisplay()

while display.IsOpen():
    display.RenderOnce(img, width, height)
    display.SetTitle(f"Camera Viewer | {width}x{height} | {display.GetFPS()} FPS")
```

Use jetson-inference to display camera feed (from *csi://0*)



```python
import jetson.utils

camera = jetson.utils.gstCamera(640, 480, "0")
display = jetson.utils.glDisplay()

camera.Open()
while display.IsOpen():
    img, width, height = camera.CaptureRGBA()
    display.RenderOnce(img, width, height)
    display.SetTitle(f"Camera Viewer | {width}x{height} | {display.GetFPS()} FPS")

camera.Close()
```

Use camera feed to detect object (using ssd-mobilenet-v2)



```python
import jetson.inference
import jetson.utils

net = jetson.inference.detectNet("ssd-mobilenet-v2", threshold=0.5)
camera = jetson.utils.gstCamera(640, 480, "0")
display = jetson.utils.glDisplay()

camera.Open()

while display.IsOpen():
    img, width, height = camera.CaptureRGBA()
    detections = net.Detect(img, width, height)
    display.RenderOnce(img, width, height)
    display.SetTitle(f"Camera Viewer | {width}x{height} | {display.GetFPS()} FPS")

camera.Close()
```

FPS is around 19