KENTECH

Korea Institute of Energy Technology

Visionary Course – Energy AI
Week 09

Seokju Lee

KENTECH
Korea Institute of Energy Technology

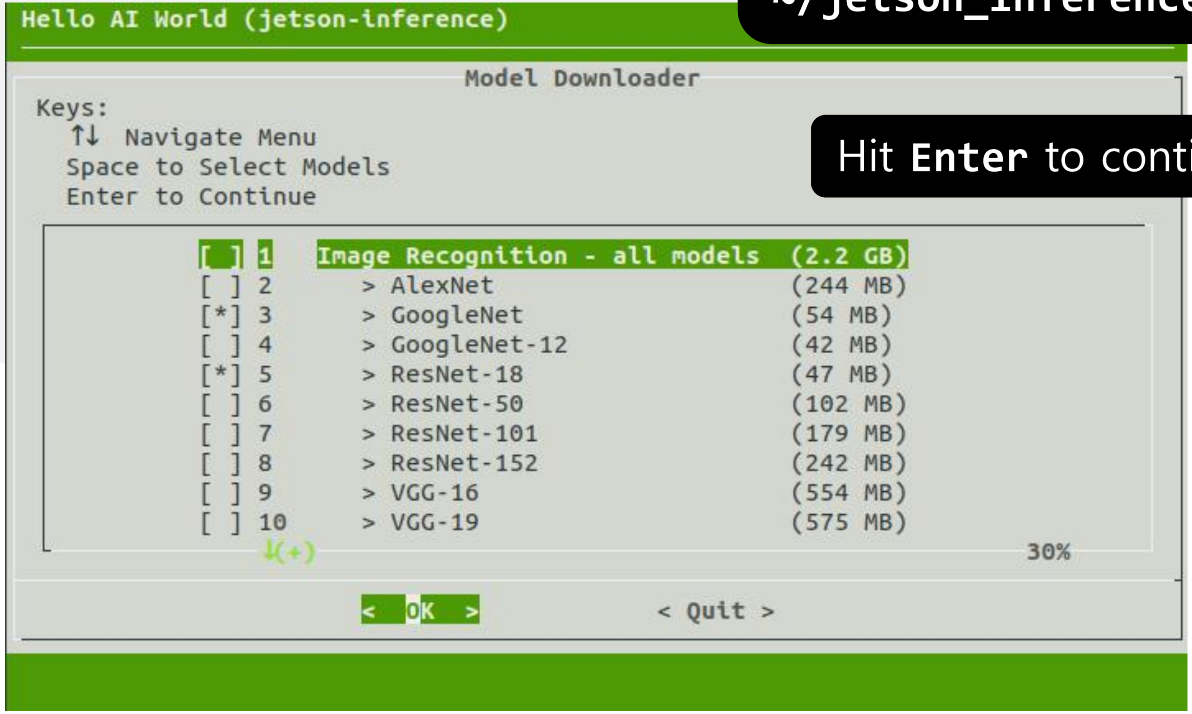
Image Classification with Jetson

Configurations: Hello-AI-World by NVIDIA

- Follow **Quick Reference** in https://github.com/dusty-nv/jetson-inference/blob/master/docs/building-repo-2.md

```
$ sudo apt-get update
$ sudo apt-get install git cmake libpython3-dev python3-numpy
$ git clone --recursive https://github.com/dusty-nv/jetson-inference
$ cd jetson-inference
$ mkdir build
$ cd build
$ cmake -DENABLE_NVMM=OFF ../
$ make –j4
$ sudo make install
$ sudo ldconfig
```

./download_models.sh
is available at
~/jetson_inference/tools



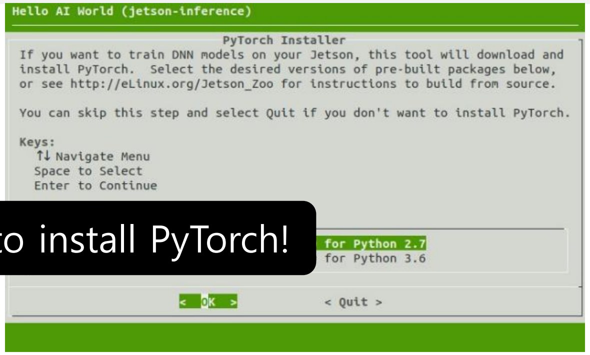
```
Hello AI World (jetson-inference)

                              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)
              ↓(+)                                          30%

                    <  OK  >            < Quit >
```

Hit **Enter** to continue!



```
Hello AI World (jetson-inference)

                         PyTorch Installer
 If you want to train DNN models on your Jetson, this tool will download and
 install PyTorch.  Select the desired versions of pre-built packages below,
 or see http://eLinux.org/Jetson_Zoo for instructions to build from source.

 You can skip this step and select Quit if you don't want to install PyTorch.

 Keys:
   ↑↓ Navigate Menu
   Space to Select
   Enter to Continue

                              for Python 2.7
                              for Python 3.6

                <  OK  >          < Quit >
```

You don’t need to install PyTorch!

Experiments – Classification (Report Due ~11/3)

### Before Starting ###
*Your basic workspace is here: “cd ~/jetson-inference/build/aarch64/bin” Every code is pre-built in this path.

### Video Streaming ###

Q1.1. Run “python video-viewer.py csi://0” What is the output?

Q1.2. Run “python video-viewer.py --flip-method=rotate-180 csi://0” Discuss the differences.

Q1.3. Run “python video-viewer.py --flip-method=rotate-180 --input-width=640 --input-height=480 csi://0” Discuss the differences.

Q1.4. Run “python video-viewer.py --flip-method=rotate-180 --input-width=640 --input-height=480 --framerate=10 csi://0” Discuss the differences.

### Live Demo for Image Classification ###

Q2.1. Run “python imagenet.py --flip-method=rotate-180” What is the output of the pop-up display? Let’s check the terminal output. Please take a screenshot and paste it here. You can see some output values. What does each output (network name, class ID, floating-point number next to it, class name, each processing time, etc.) mean?

Experiments – Classification (Report Due ~11/3)

Q2.2. Go to the linked page (https://deeplearning.cms.waikato.ac.nz/user-guide/class-maps/IMAGENET/) and check that the class ID is matched to the class name. How many classes can the model distinguish in total? Please prepare your **own object (🕶,👜,🌂,👢,⚾)** corresponding to one of the above classes for further experiments (classification, detection, etc.).

Q2.3. Run "cd ~/jetson-inference/build; ./download-models.sh;" to download different CNN models (e.g., AlexNet, ResNet-50, etc.). Run "python imagenet.py --flip-method=rotate-180 --network=resnet-50". Please check the qualitative performance of each model.

### Classify Your Own Objects or Images ###

Q3.1. Place the object in front of the camera and run the code (imagenet.py). Please take a screenshot of the result.

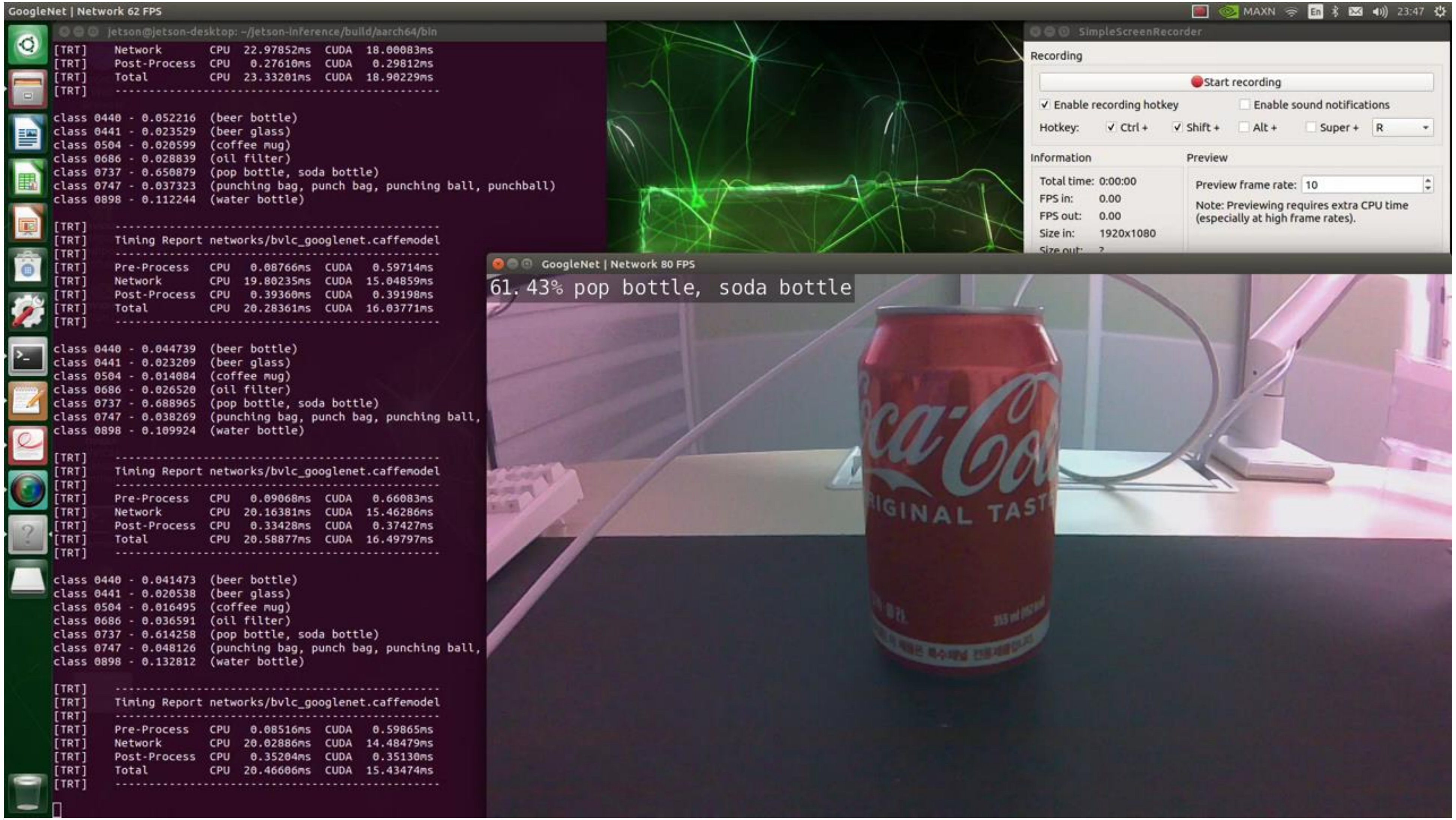
Q3.2. Position the object closer or further away from the camera. Please Analyze how confidence changes.

Q3.3. Download random images from Google and classify them. Please refer "python my-recognition.py images/banana_0.jpg --network=googlenet" and the below code:

```
import PIL
img = PIL.Image.open('jellyfish.jpg').resize((224,224))
img = np.array(img)
img_cuda = jetson.utils.cudaFromNumpy(img)      # CUDA image
class_id, confidence = net.Classify(img_cuda)   # Inference
class_desc = net.GetClassDesc(class_id)         # Predicted class
print(class_desc, confidence)
```

Q3.4. Please try other CNN models and repeat Q3.

Live Demo for Image Classification



GoogleNet | Network 62 FPS
MAXN
En
23:47
jetson@jetson-desktop: ~/jetson-inference/build/aarch64/bin
SimpleScreenRecorder
GoogleNet | Network 80 FPS

```
[TRT]    Network       CPU  22.97852ms  CUDA  18.00083ms
[TRT]    Post-Process  CPU   0.27610ms  CUDA   0.29812ms
[TRT]    Total         CPU  23.33201ms  CUDA  18.90229ms
[TRT]    ------------------------------------------------

class 0440 - 0.052216  (beer bottle)
class 0441 - 0.023529  (beer glass)
class 0504 - 0.020599  (coffee mug)
class 0686 - 0.028839  (oil filter)
class 0737 - 0.650879  (pop bottle, soda bottle)
class 0747 - 0.037323  (punching bag, punch bag, punching ball, punchball)
class 0898 - 0.112244  (water bottle)

[TRT]    ------------------------------------------------
[TRT]    Timing Report networks/bvlc_googlenet.caffemodel
[TRT]    ------------------------------------------------
[TRT]    Pre-Process   CPU   0.08766ms  CUDA   0.59714ms
[TRT]    Network       CPU  19.80235ms  CUDA  15.04859ms
[TRT]    Post-Process  CPU   0.39360ms  CUDA   0.39198ms
[TRT]    Total         CPU  20.28361ms  CUDA  16.03771ms
[TRT]    ------------------------------------------------

class 0440 - 0.044739  (beer bottle)
class 0441 - 0.023209  (beer glass)
class 0504 - 0.014084  (coffee mug)
class 0686 - 0.026520  (oil filter)
class 0737 - 0.688965  (pop bottle, soda bottle)
class 0747 - 0.038269  (punching bag, punch bag, punching ball,
class 0898 - 0.109924  (water bottle)

[TRT]    ------------------------------------------------
[TRT]    Timing Report networks/bvlc_googlenet.caffemodel
[TRT]    ------------------------------------------------
[TRT]    Pre-Process   CPU   0.09068ms  CUDA   0.66083ms
[TRT]    Network       CPU  20.16381ms  CUDA  15.46286ms
[TRT]    Post-Process  CPU   0.33428ms  CUDA   0.37427ms
[TRT]    Total         CPU  20.58877ms  CUDA  16.49797ms
[TRT]    ------------------------------------------------

class 0440 - 0.041473  (beer bottle)
class 0441 - 0.020538  (beer glass)
class 0504 - 0.016495  (coffee mug)
class 0686 - 0.036591  (oil filter)
class 0737 - 0.614258  (pop bottle, soda bottle)
class 0747 - 0.048126  (punching bag, punch bag, punching ball,
class 0898 - 0.132812  (water bottle)

[TRT]    ------------------------------------------------
[TRT]    Timing Report networks/bvlc_googlenet.caffemodel
[TRT]    ------------------------------------------------
[TRT]    Pre-Process   CPU   0.08516ms  CUDA   0.59865ms
[TRT]    Network       CPU  20.02886ms  CUDA  14.48479ms
[TRT]    Post-Process  CPU   0.35204ms  CUDA   0.35130ms
[TRT]    Total         CPU  20.46606ms  CUDA  15.43474ms
[TRT]    ------------------------------------------------
```

Recording

Start recording

☑ Enable recording hotkey ☐ Enable sound notifications

Hotkey: ☑ Ctrl + ☑ Shift + ☐ Alt + ☐ Super + R

Information

Total time: 0:00:00
FPS in: 0.00
FPS out: 0.00
Size in: 1920x1080
Size out: ?

Preview

Preview frame rate: 10

Note: Previewing requires extra CPU time (especially at high frame rates).

61. 43% pop bottle, soda bottle

https://youtu.be/uMiXN-Z-hg4

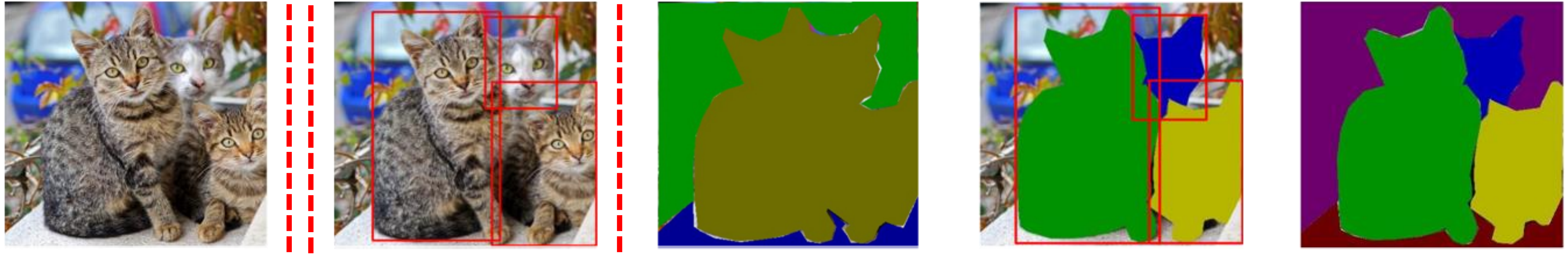
Computer Vision Tasks

*Figure by Kim, et al., "Video Panoptic Segmentation" (CVPR 2020)

Model Complexity ⇧ Output dimension ⇧

Image

Image Classification | Object Detection | Semantic Segmentation | Instance Segmentation | Panoptic Segmentation



"Image-level"

"Box-level"

RCNN, CVPR 2014
Fast RCNN, CVPR 2015
Faster RCNN NeurIPS 2015
YOLO, CVPR 2016
SSD, ECCV 2016
Cascade RCNN, CVPR 2017
CornerNet, ECCV 2018
HTC, CVPR 2019

"Pixel-level"

FCN, CVPR 2014
Dilated FCN, ICLR 2015
DeepLab, ICLR 2015
PSPNet, CVPR 2016
Deform FCN, ICCV 2017
EncNet, CVPR 2018
DeepLabv2, ECCV 2018
DANet, CVPR 2019
Deformable V2, ICCV 2019

"Pixel-level"
+
"Instance-level"

RCNN, CVPR 2014
DeepMask, NeurIPS 2015
MNC, CVPR 2016
FCIS, CVPR 2017
Mask-RCNN, ICCV 2017
Cascade Mask-RCNN, PAMI 2019
HTC, CVPR2019
⋮

"Unknown pixel"

PanopticFPN, CVPR2019
UPSNet, CVPR2019

Video

Karpathy et.al., CVPR 2014 → Video Classification

Kai et.al., CVPR 2016 → Video Object Detection

Xizhou et.al., CVPR 2017 → Video Semantic Segmentation

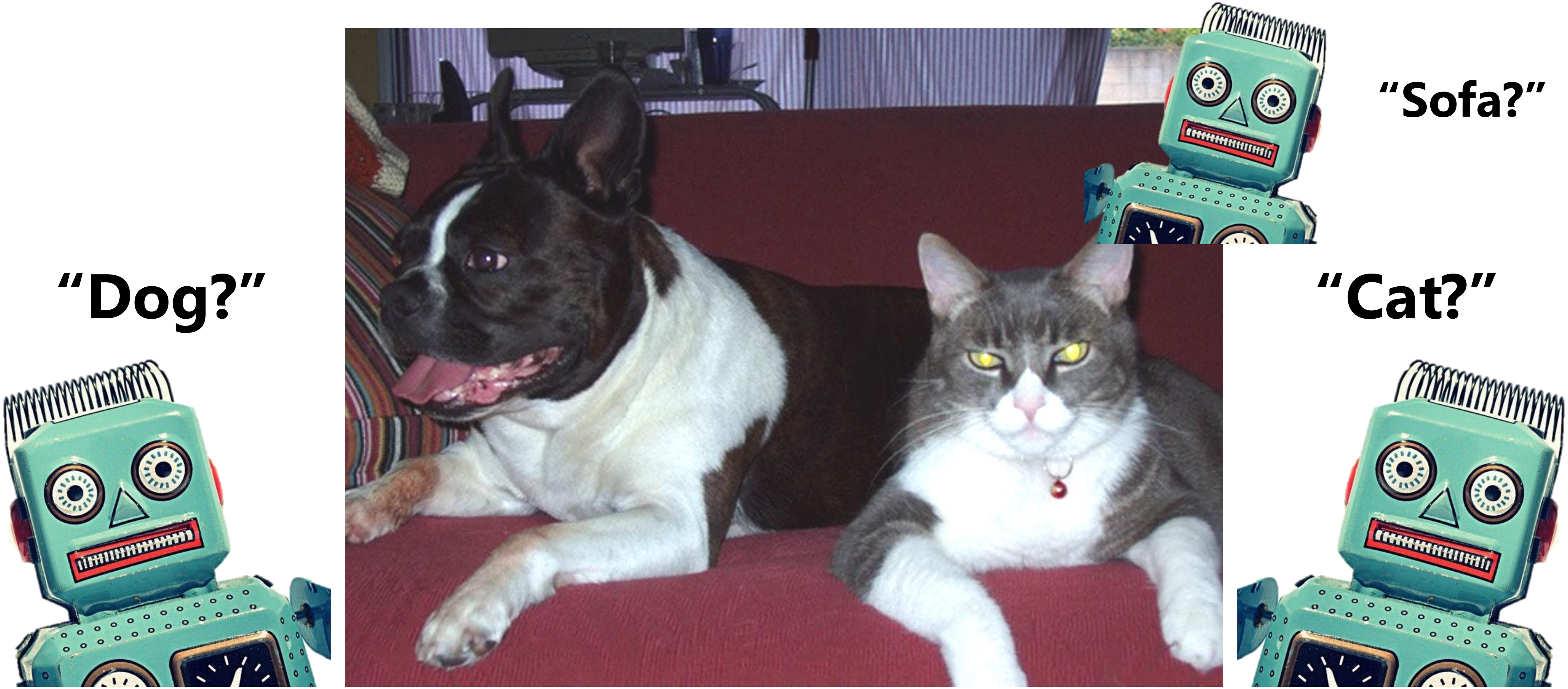
Linjie et.al., ICCV 2019 → Video Instance Segmentation

Kim et.al., CVPR 2020 → Video Panoptic Segmentation

KENTECH
Korea Institute of Energy Technology

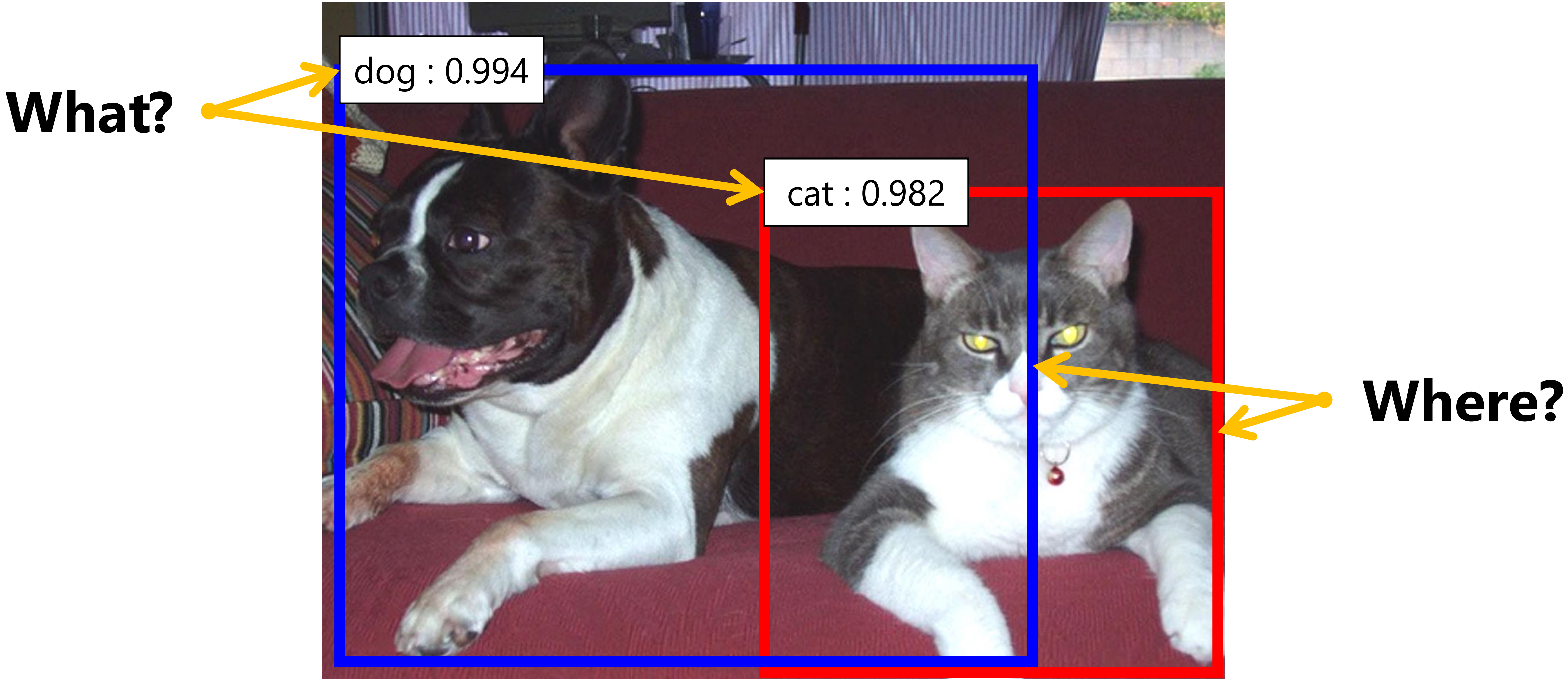
Object Detection with Jetson

Limitation of Image Classification: Dog or Cat?



[1] S. Ren, et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *NIPS* (2015).

Object Detection: What and Where?



What?

Where?

[1] S. Ren, et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *NIPS* (2015).

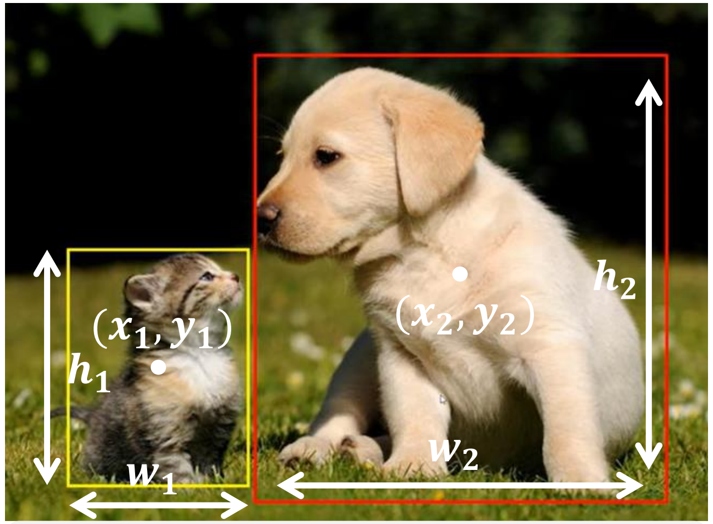
Object Detection: Input & Output

: Task of assigning **labels** & **bounding boxes** to all objects in the image.

Classify **which class** the object belongs to.

Images	Class (=label)
I_1	cat
I_2	cat
I_3	dog

Classification

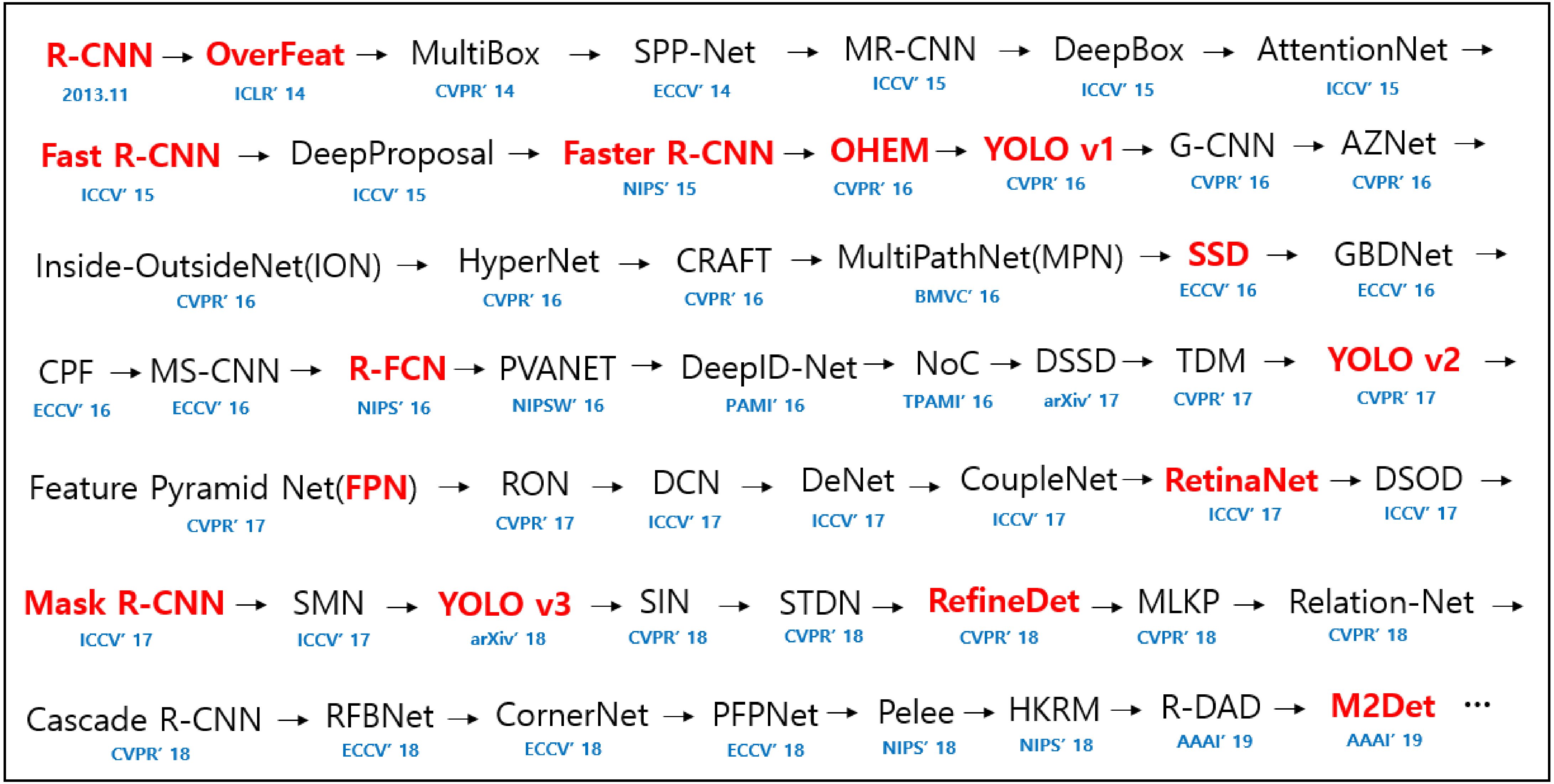


Find the coordinates of the bounding box **where** the object is located, and classify **which class** it belongs to.

Images	Class (=label)	x	y	w	h
I_1	cat	60	210	100	180
I_1	dog	200	50	340	360
I_2	car	46	250	100	80

Classification
+
Regression

Object Detection: How to Design Models?



R-CNN → **OverFeat** → MultiBox → SPP-Net → MR-CNN → DeepBox → AttentionNet →
2013.11 ICLR' 14 CVPR' 14 ECCV' 14 ICCV' 15 ICCV' 15 ICCV' 15

Fast R-CNN → DeepProposal → **Faster R-CNN** → **OHEM** → **YOLO v1** → G-CNN → AZNet →
ICCV' 15 ICCV' 15 NIPS' 15 CVPR' 16 CVPR' 16 CVPR' 16 CVPR' 16

Inside-OutsideNet(ION) → HyperNet → CRAFT → MultiPathNet(MPN) → **SSD** → GBDNet →
CVPR' 16 CVPR' 16 CVPR' 16 BMVC' 16 ECCV' 16 ECCV' 16

CPF → MS-CNN → **R-FCN** → PVANET → DeepID-Net → NoC → DSSD → TDM → **YOLO v2** →
ECCV' 16 ECCV' 16 NIPS' 16 NIPSW' 16 PAMI' 16 TPAMI' 16 arXiv' 17 CVPR' 17 CVPR' 17

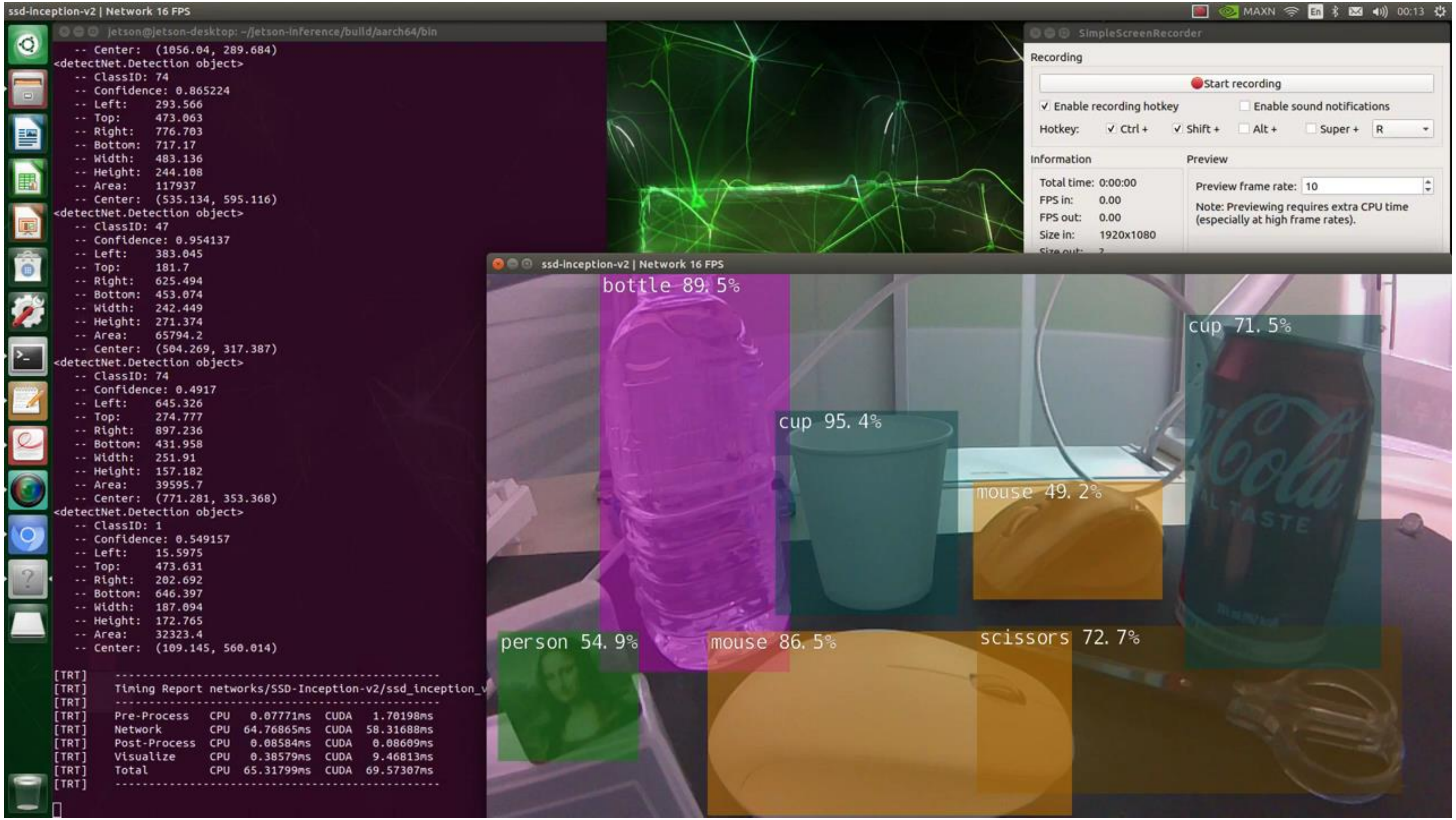
Feature Pyramid Net(**FPN**) → RON → DCN → DeNet → CoupleNet → **RetinaNet** → DSOD →
CVPR' 17 CVPR' 17 ICCV' 17 ICCV' 17 ICCV' 17 ICCV' 17 ICCV' 17

Mask R-CNN → SMN → **YOLO v3** → SIN → STDN → **RefineDet** → MLKP → Relation-Net →
ICCV' 17 ICCV' 17 arXiv' 18 CVPR' 18 CVPR' 18 CVPR' 18 CVPR' 18 CVPR' 18

Cascade R-CNN → RFBNet → CornerNet → PFPNet → Pelee → HKRM → R-DAD → **M2Det** …
CVPR' 18 ECCV' 18 ECCV' 18 ECCV' 18 NIPS' 18 NIPS' 18 AAAI' 19 AAAI' 19

https://github.com/hoya012/deep_learning_object_detection

Live Demo for Object Detection


ssd-inception-v2 | Network 16 FPS
MAXN
En
00:13
jetson@jetson-desktop: ~/jetson-inference/build/aarch64/bin
SimpleScreenRecorder
ssd-inception-v2 | Network 16 FPS
bottle 89.5%
cup 71.5%
cup 95.4%
mouse 49.2%
person 54.9%
mouse 86.5%
scissors 72.7%

```
   -- Center:  (1056.04, 289.684)
<detectNet.Detection object>
   -- ClassID: 74
   -- Confidence: 0.865224
   -- Left:    293.566
   -- Top:     473.063
   -- Right:   776.703
   -- Bottom:  717.17
   -- Width:   483.136
   -- Height:  244.108
   -- Area:    117937
   -- Center:  (535.134, 595.116)
<detectNet.Detection object>
   -- ClassID: 47
   -- Confidence: 0.954137
   -- Left:    383.045
   -- Top:     181.7
   -- Right:   625.494
   -- Bottom:  453.074
   -- Width:   242.449
   -- Height:  271.374
   -- Area:    65794.2
   -- Center:  (504.269, 317.387)
<detectNet.Detection object>
   -- ClassID: 74
   -- Confidence: 0.4917
   -- Left:    645.326
   -- Top:     274.777
   -- Right:   897.236
   -- Bottom:  431.958
   -- Width:   251.91
   -- Height:  157.182
   -- Area:    39595.7
   -- Center:  (771.281, 353.368)
<detectNet.Detection object>
   -- ClassID: 1
   -- Confidence: 0.549157
   -- Left:    15.5975
   -- Top:     473.631
   -- Right:   202.692
   -- Bottom:  646.397
   -- Width:   187.094
   -- Height:  172.765
   -- Area:    32323.4
   -- Center:  (109.145, 560.014)

[TRT]    ------------------------------------------------
[TRT]    Timing Report networks/SSD-Inception-v2/ssd_inception_v
[TRT]    ------------------------------------------------
[TRT]    Pre-Process   CPU   0.07771ms  CUDA   1.70198ms
[TRT]    Network       CPU  64.76865ms  CUDA  58.31688ms
[TRT]    Post-Process  CPU   0.08584ms  CUDA   0.08609ms
[TRT]    Visualize     CPU   0.38579ms  CUDA   9.46813ms
[TRT]    Total         CPU  65.31799ms  CUDA  69.57307ms
[TRT]    ------------------------------------------------
```

Recording

● Start recording

☑ Enable recording hotkey   □ Enable sound notifications

Hotkey: ☑ Ctrl + ☑ Shift + □ Alt + □ Super + R

Information

Total time: 0:00:00
FPS in: 0.00
FPS out: 0.00
Size in: 1920x1080
Size out: ?

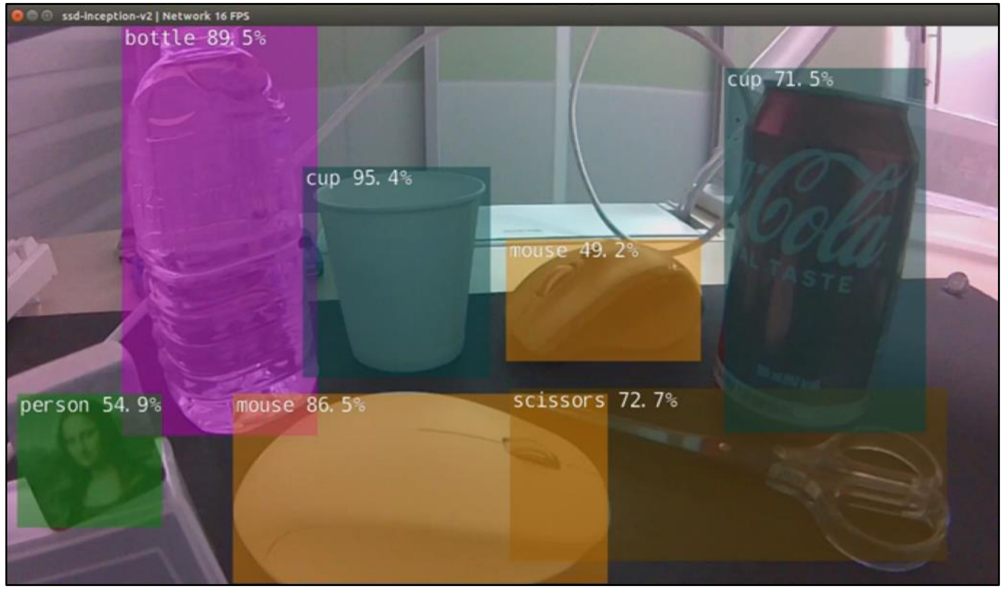
Preview

Preview frame rate: 10

Note: Previewing requires extra CPU time (especially at high frame rates).

https://youtu.be/YU37WJ5z4Lg

Experiments – Object Detection



Your mission :

“Try to detect objects as much as you can!”

*Rule: the predicted class must be correct.

→ Submit the captured detection image.

*Due: ~11/7 (Mon)

### Useful Commands ###

*Your basic workspace is here: “cd ~/jetson-inference/build/aarch64/bin”

Run “python detectnet.py --flip-method=rotate-180”. Which model are you running?

### Tip 1: Try Different Models for better results ###

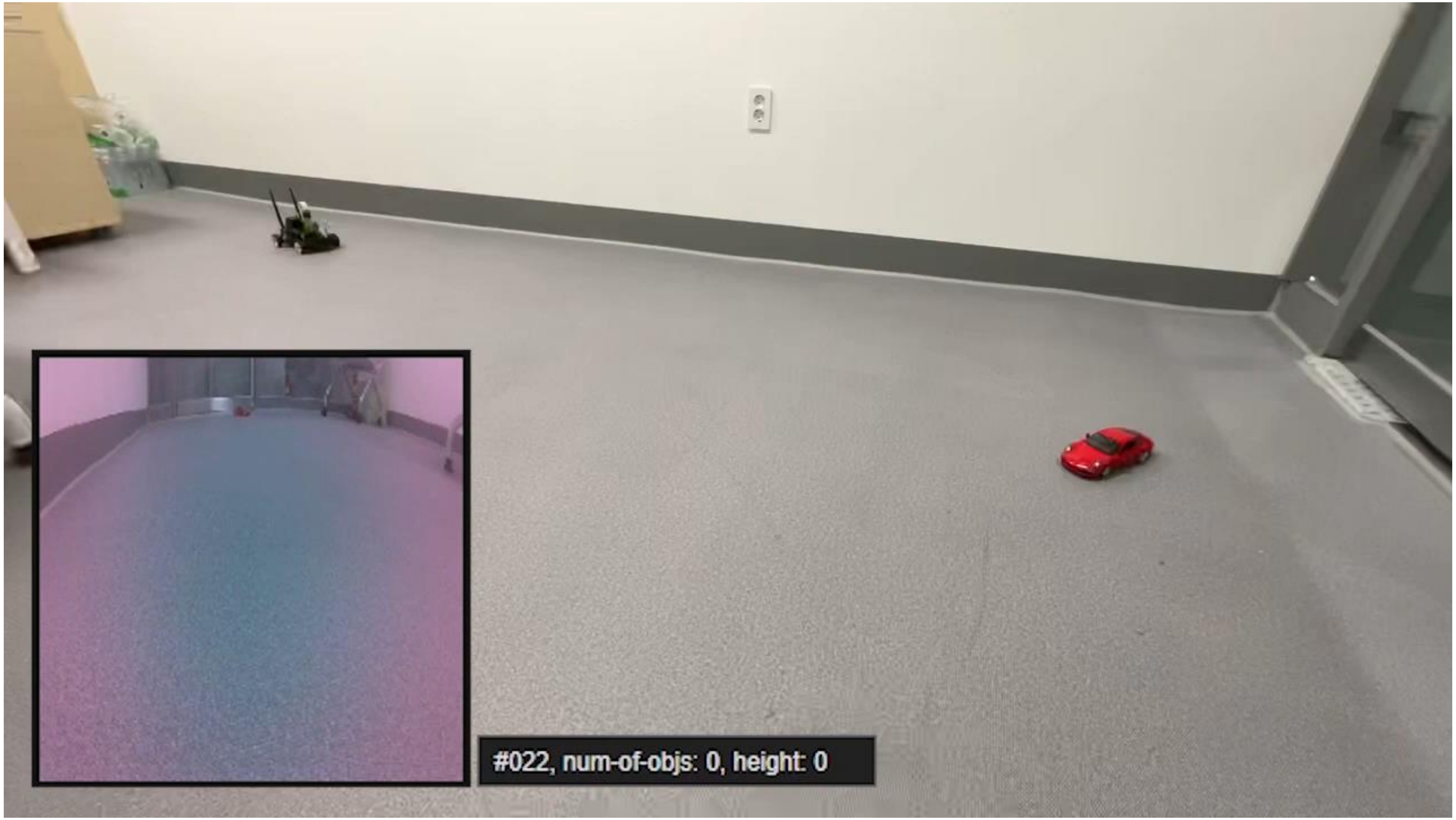
You can download other models by running “./download_models.sh” at “~/jetson_inference/tools”.

Run “python detectnet.py --network=ssd-inception-v2 --flip-method=rotate-180”.

### Tip 2: Try Different Thresholds for better results ###

Control threshold (default threshold =0.5 e.g., 0.3 & 0.7) by runing “python detectnet.py --threshold=0.3 --flip-method=rotate-180”. What does threshold mean?

JetRacer Mission: Stop-and-Go



```
#022, num-of-objs: 0, height: 0
```

https://youtu.be/XCbgLJEEtDg

Experiments

### Some Useful Tips while Debugging ###

*Sometimes, the python process does not respond. In this case, please terminate the process with ctrl+c. If it still does not respond at all, forcibly stop the process with ctrl+z, and check the running process name with the ps -a command, and then type sudo pkill -9 [name-of-process] command to kill the process. If you don't shut it down, it will remain as a 🧟zombie🧟 and keep occupying the processor (CPU or GPU) in the background.

*Sometimes, the best solution for resolving an issue is just rebooting the system.

Q&A

KENTECH

Korea Institute of Energy Technology