

DL Lab #1: Object Detection using YOLO

Sunglok Choi, Assistant Professor, Ph.D.
Computer Science and Engineering Department, SEOULTECH
sunglok@seoultech.ac.kr | <https://mint-lab.github.io/>

Overview

- **Prerequisite**

- Anaconda (Individual Edition) with PyTorch Installation
- Google Colab

- **Practice) Object Detection using YOLO**

- The given data
- Expected results
- Practice with the skeleton code
 - Step #1) Run the given the skeleton code

- **Assignment**

- Mission: Run the given skeleton code

Review) PyTorch Installation

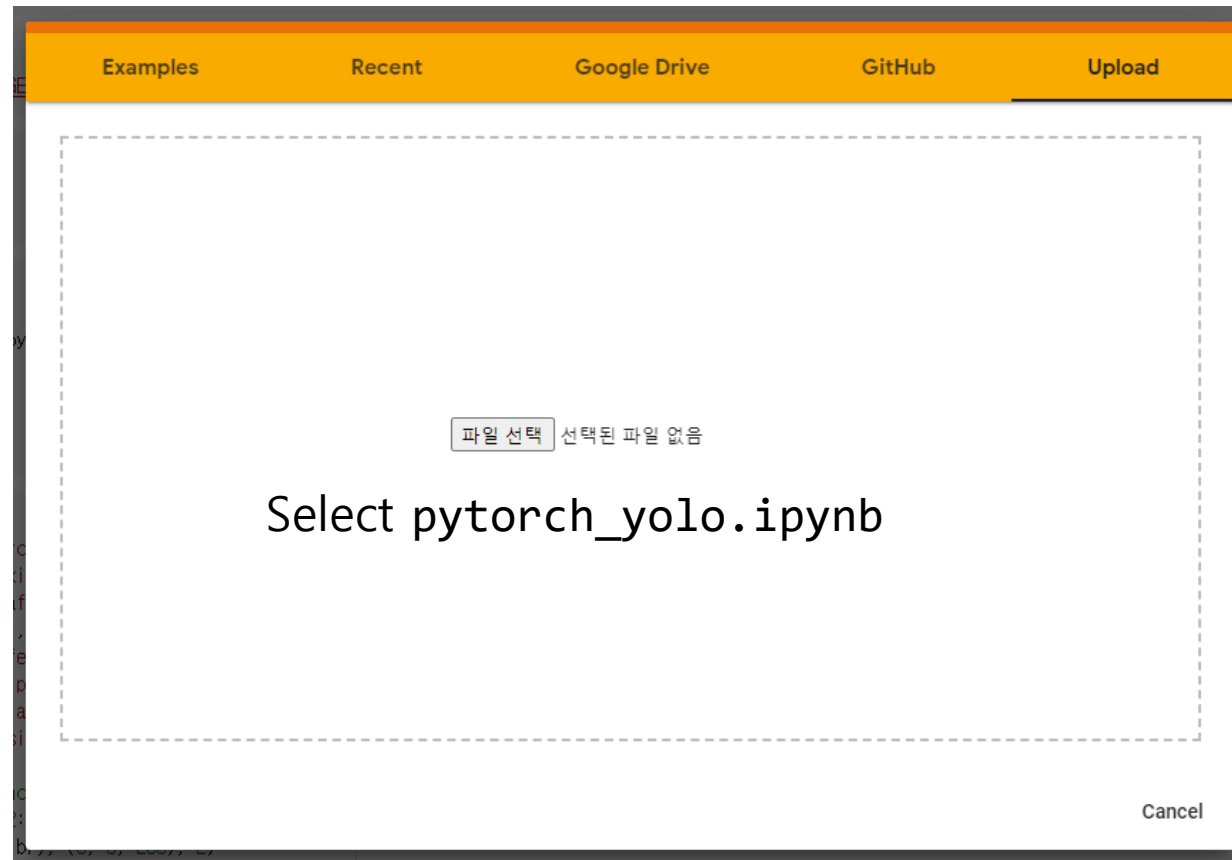
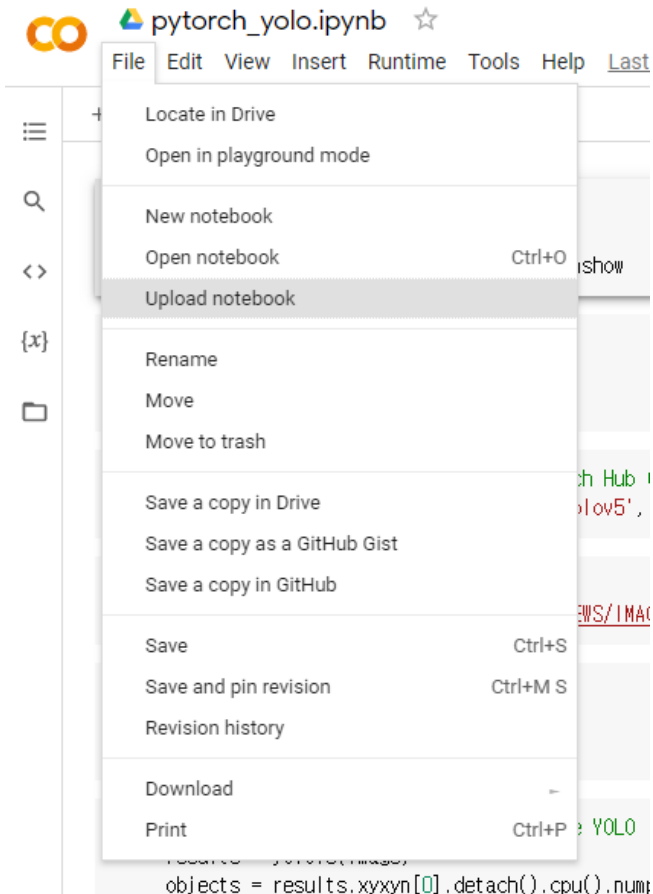
- Please follow [PyTorch's instruction of installation](#) for your system.
 - Note) If you want GPU acceleration, please install the matched version of CUDA in advance. Please visit [CUDA Toolkit Archive](#) to download a specific version of CUDA.

PyTorch Build	Stable (1.8.1)		Preview (Nightly)	
Your OS	Linux	Mac	Windows	
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
Compute Platform	CUDA 10.2	CUDA 11.1	ROCm 4.0 (beta)	CPU
Run this Command:	NOTE: 'conda-forge' channel is required for cudatoolkit 11.1 <code>conda install pytorch torchvision torchaudio cudatoolkit=11.1 -c pytorch -c conda-forge</code>			

- Please use the given `pytorch_yolo.py` and `test.jpg` for the today's practice.

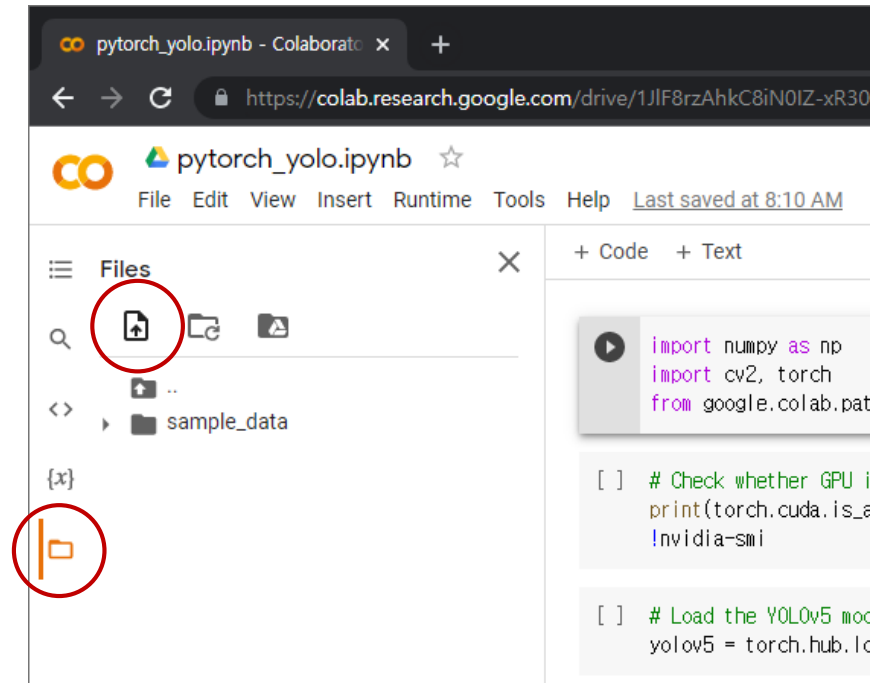
Google Colab

- [Google Colaboratory](#)
 - It requires Google account.
 - Click "Sign in" at the top-right if you don't have or log in.
- Please upload the given notebook file, `pytorch_yolo.ipynb`.



Practice) Object Detection using YOLO

- The given data (test.jpg)
 - Method #1) Upload the given data



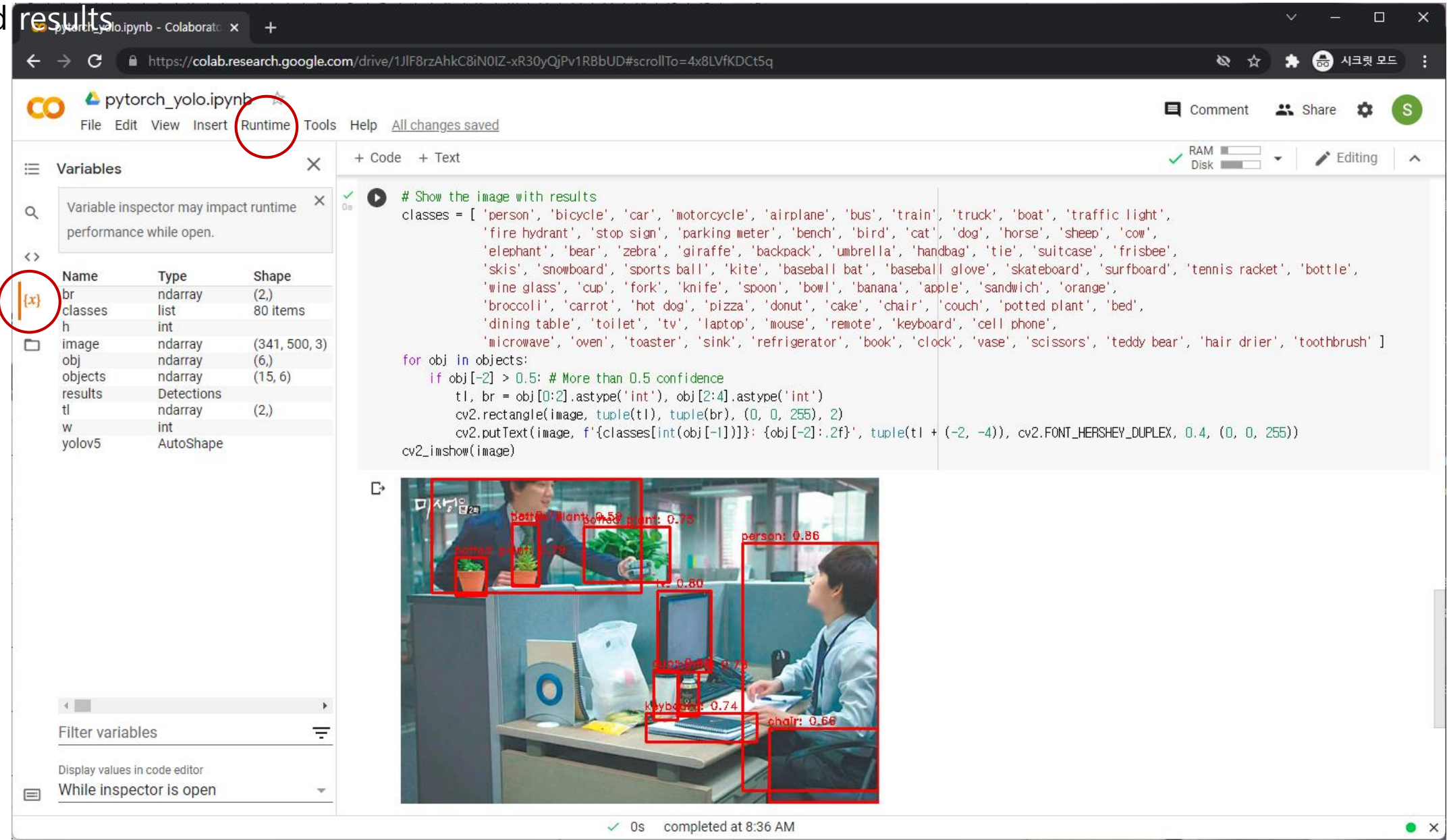
- Method #2) Download the image from internet

Download an image from internet

```
!wget -c 'https://dimg.donga.com/wps/NEWS/IMAGE/2014/11/26/68179447.1.jpg' -O 'test.jpg'
```

Practice) Object Detection using YOLO

Expected results



The screenshot displays a Google Colab notebook titled 'pytorch_yolo.ipynb'. The 'Runtime' tab is selected, and the 'Variables' sidebar on the left shows the state of the notebook's variables. The 'image' variable is highlighted, showing its shape as (341, 500, 3). The main code cell, titled '# Show the image with results', defines a list of 80 object classes and iterates through the detected objects, printing their bounding boxes and confidence scores. The output shows a video frame with several objects detected and labeled with their confidence scores.

Name	Type	Shape
br	ndarray	(2,)
classes	list	80 items
h	int	
image	ndarray	(341, 500, 3)
obj	ndarray	(6,)
objects	ndarray	(15, 6)
results	Detections	
tl	ndarray	(2,)
w	int	
yolov5	AutoShape	

```
# Show the image with results
classes = [ 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',
            'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
            'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
            'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle',
            'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange',
            'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed',
            'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
            'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush' ]

for obj in objects:
    if obj[-2] > 0.5: # More than 0.5 confidence
        tl, br = obj[0:2].astype('int'), obj[2:4].astype('int')
        cv2.rectangle(image, tuple(tl), tuple(br), (0, 0, 255), 2)
        cv2.putText(image, f'{classes[int(obj[-1])]}: {obj[-2]:.2f}', tuple(tl + (-2, -4)), cv2.FONT_HERSHEY_DUPLEX, 0.4, (0, 0, 255))
cv2.imshow('image')
```

Output image showing object detection results with bounding boxes and confidence scores:

- person: 0.86
- chair: 0.66
- keyboard: 0.74
- cup: 0.55
- broccoli: 0.79
- potted plant: 0.79
- pottery plant: 0.58
- sofa: 0.75

completed at 8:36 AM

Practice) Object Detection using YOLO

- The given skeleton code (1/2)

```
import numpy as np
import cv2, torch
from google.colab.patches import cv2_imshow
```

```
# Check whether GPU is available or not
print(torch.cuda.is_available())
!nvidia-smi
```

```
# Load the YOLOv5 model from the Pytorch Hub (https://pytorch.org/hub/)
yolov5 = torch.hub.load('ultralytics/yolov5', 'yolov5l', pretrained=True)
```

```
# Download an image from internet
!wget -c 'https://dimg.donga.com/wps/NEWS/IMAGE/2014/11/26/68179447.1.jpg' -O 'test.jpg'
```

```
# Load an image on internet
image = cv2.imread('test.jpg')
cv2_imshow(image)
```



Practice) Object Detection using YOLO

- The given skeleton code (2/2)

```
# Detect objects on the image using the YOLO
```

```
results = yolov5(image)
```

```
objects = results.xyxy[0].detach().cpu().numpy()
```

```
# Rescale object locations
```

```
h, w, _ = image.shape
```

```
objects[:,0:4] = objects[:,0:4] * [w, h, w, h]
```

```
# Show the image with results
```

```
classes = [ 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', ... ]
```

```
for obj in objects:
```

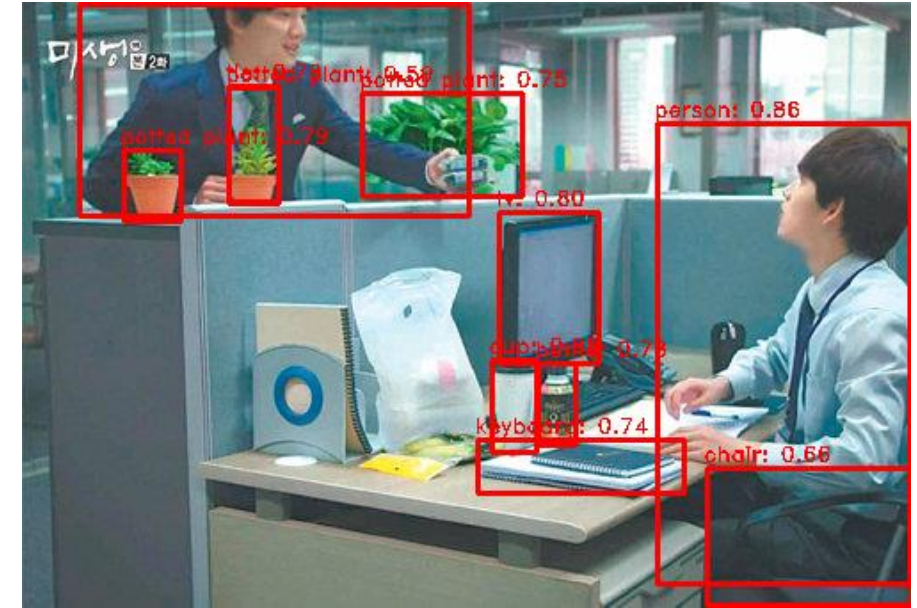
```
    if obj[-2] > 0.5: # More than 0.5 confidence
```

```
        tl, br = obj[0:2].astype('int'), obj[2:4].astype('int')
```

```
        cv2.rectangle(image, tuple(tl), tuple(br), (0, 0, 255), 2)
```

```
        cv2.putText(image, f'{classes[int(obj[-1])]}: {obj[-2]:.2f}', tuple(tl + (-2, -4)), cv2.FONT_HERSHEY_DUPLEX, 0.4, (0, 0, 255))
```

```
cv2_imshow(image)
```



Tip) If Your Session does not have GPU

1. Click "RAM / Disk" (or list box(▼) > View resources)
2. Click "Change runtime type"
3. Select "Hardware accelerator" as "GPU"

The screenshot shows the Google Colab interface for a notebook named 'pytorch_yolo.ipynb'. The 'Notebook settings' dialog box is open, and the 'Hardware accelerator' dropdown menu is expanded, showing 'None', 'GPU', and 'TPU' options. The 'GPU' option is highlighted. In the background, the 'Resources' tab is visible, showing 'RAM' and 'Disk' usage. Red circles and numbers 1, 2, and 3 are overlaid on the image to indicate the steps: 1. Circle around the 'RAM / Disk' dropdown in the top right. 2. Circle around the 'Change runtime type' button in the bottom right. 3. Circle around the 'GPU' option in the 'Hardware accelerator' dropdown.

pytorch_yolo.ipynb - Colaboratory

https://colab.research.google.com/drive/1JIF8rzAhkC8iN0IZ-xR30yQjPv1RBbUD#scrollTo=IVDNXLpnAmzU

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[2] import numpy as np
import cv2, torch
from google.colab.patches import cv2_imshow
```

```
[3] # Check whether GPU is available or not
print(torch.cuda.is_available())
!nvidia-smi
```

False
NVIDIA-SMI has failed because it couldn't communicate with the GPU driver.

Hardware accelerator: None (v) ?

None
GPU
TPU

☐ Omit code cell output when saving this notebook

Cancel Save

1. RAM / Disk

2. Change runtime type

3.

Resources

Python 3 Google Compute Engine backend
Showing resources since 9:00 AM

RAM

Disk

Manage sessions

9

Assignment

- Mission
 - Run the skeleton code **with your desired image (or video)**
 - Submit your screenshot (screenshot.png) on your web browser or Anaconda

- Condition
 - You **can** start from scratch (without using the given skeleton code).
 - However, you **should** use **another image or video**.
 - You **can** freely change the given skeleton code if necessary.

- Submission
 - Deadline: **November 27, 2024 23:59** (**firm deadline**; no extension)
 - Where: e-Class > Assignments
 - Score: Max 10 points