# Computer Vision

## Lecture 01: Computer vision overview

# Course overview

- **CSE 47201 Computer Vision**
- **Instructor**:
  - Prof. Seungryul Baek (AIGS, Dept. of CSE)
- **Goal**
  - Understand what is **computer vision/deep learning** algorithms.
  - Understand how to implement image/video processing algorithms in Python using **OpenCV**.
  - Understand how to implement/train/test convolutional neural networks (CNNs) in **PyTorch**.

# Syllabus

- **Before mid-term**
  - Understand basic machine learning & deep learning.
  - Understand how to Implement algorithms with OpenCV and PyTorch libraries.
- **After mid-term**
  - Apply deep learning to computer vision application.

# Grading

- Attendance (+Participation) score: 10%
  - + Participation score for good answers during classes.
- 4 programming **assignments**: 12.5% * 4 = 50%
  - Python implementation of algorithms.
- **Quizzes**: 20%
  - 2-3 quizzes during the course.
- **Final-term** Exam: 20%

# Grading

- **Late policy**:
  - 1 free late days
  - 25% off per day late

# Why should you take this class?

- **Become a vision researcher in academia**
  - CVPR/ECCV/ICCV conference
  - NeurIPS/ICLR/ICML conference
- **Become a vision engineer in industry**
  - Naver, Kakao, Samsung, …
  - Google, Facebook, Amazon, Microsoft, Apple,…
- **General interest**

# Pre-requisite

- **Proficiency in Python**
  - All class assignments will be in Python
  - ITP107 Engineering Programming

- **We will further learn:**
  - Computer vision library: OpenCV.
  - Deep learning library: PyTorch.

# Pre-requisite

- **Mathematical Background**
  - Basic Linear Algebra (Matrix multiplication, …)
  - Basic Probabilistic theory (Random variables, …)
    - MTH203 Applied Linear Algebra
    - EE211 Probability and Intro. To Random Process

- **We will further learn:**
  - Machine learning/Deep learning algorithms.

# Optional textbook resources

- **Deep Learning**
  - Goodfellow, Bengio and Courville
  - We have a web access to this material (https://www.deeplearningbook.org/).

- **Dive into deep learning**
  - We have a web access to this material (https://d2l.ai/).

# Assignments

- 4 programming assignments will be given in this course.
- **Two ways of completing assignments**
  - On your own local machines w/ GPUs.
  - On Google Colab
    - https://colab.research.google.com/
    - Provides 12 hours of consecutive access to GPUs.
    - After 12 hours, you have to re-connect it.

# UNIST ECE Policy on Cheating and Plagiarism

- **Rule 1**: : You must not look at solutions that are not your own.
  - It is an act of plagiarism to submit work that is copied or derived from the work of others and submitted as your own. Specifically, you should not use nor look at a solution in part or in whole from the Internet, another student (past or present), or some other sources. Many Honor Code infractions we see make use of solution found online. The best way to steer clear of this possibility is not to search for online solutions. Moreover, looking at someone else's solution in order to determine how to solve the problem yourself is also an infraction of the Honor Code. In essence, you should not be looking at someone else's solution in order to solve the problems in class. This is not an appropriate way to "check your work," "get direction," or "see alternative approaches."

# UNIST ECE Policy on Cheating and Plagiarism

- **Rule 2**: You must not share your solution with other students (even unintentionally).
  - In particular:
    - You should not ask anyone to give you their solution.
    - **You should not give your solution to another student who asks you for it, even for friends and family.**
    - You should not discuss your algorithmic strategies to such an extent that you and your collaborators end up turning in the same solution.
    - You are expected to take reasonable measures to maintain the privacy of your solutions. For example, you should not leave copies of your work on public computers. Make sure your code does not remain in recycling bins or undeleted trash folders on public computers.
    - You should not post your solution on a public website even after the due date. Posting the solutions from past years is not allowed. Even though you are not taking a particular class, you should not post the solution that can be accessed by the students of that particular class.

# UNIST ECE Policy on Cheating and Plagiarism

- **Rule 3**: You must indicate on your submission any assistance you received. If you make use of such assistance without giving proper credit, you may be guilty of plagiarism.
  - If you received aid while producing your solution, you should indicate from whom you got help (if that person is not a section leader, TA, or instructor for this class) and what help you received. A proper citation should specifically identify the source (e.g., person's name, book title, website URL, etc.) and a clear indication of how this assistance influenced your work (be as specific as possible). For example, you might write "I discussed the approach used for sorting numbers in the sortNumbers method with David." If you make use of such assistance without giving proper credit, you may be guilty of plagiarism.

# UNIST ECE Policy on Cheating and Plagiarism

- **Rule 4**: You should not use your previous work (self-plagiarism).
  - It is not allowed to use the solutions you submitted in other classes even though it is your own work. This is classified as “self-plagiarism”. It is also not allowed to use specific programming or algorithmic templates you used from your previous training/study because this makes the solution similar if others took the same training.

# Collaboration Guidelines

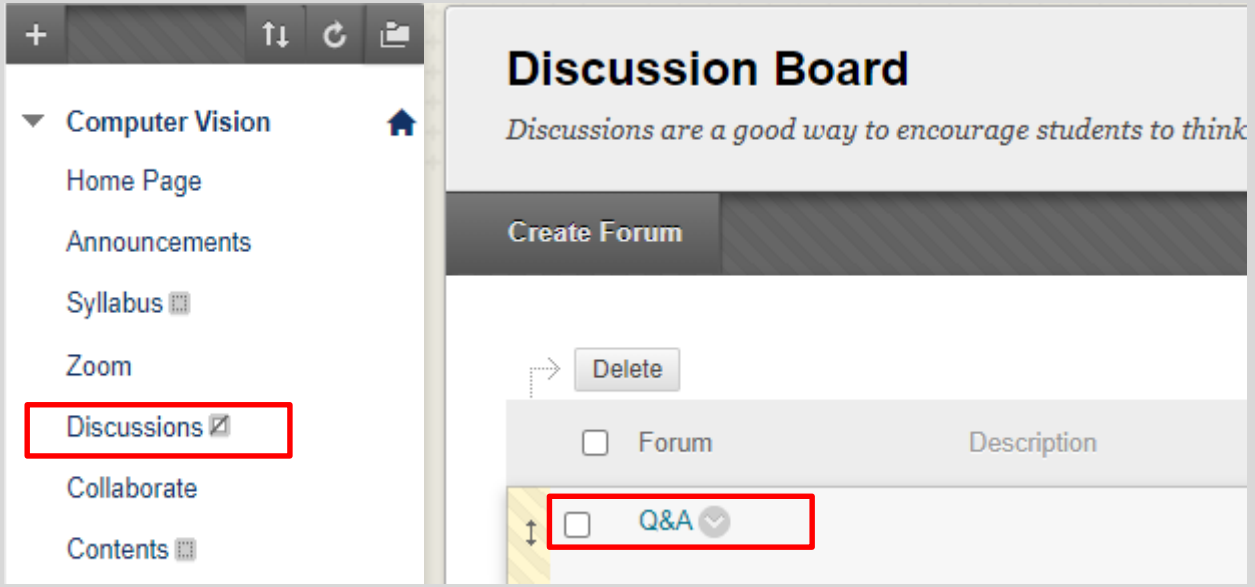
- You are encouraged to collaborate amongst yourselves. However, the discussion must be **“high level”** and **“general”**, and actual solutions/source code should not be shared in any case. More specifically, the following are permitted:
  - Discussion of material covered during lecture, problem sessions, or in handouts
  - Discussion of the requirements of an assignment
  - Discussion of the use of tools or development environments
  - Discussion of general approaches to solving problems
  - Discussion of general techniques of coding or debugging (but debugging code for someone else is not allowed)
  - Discussion between a student and a TA or instructor for the course

# Cheating cases

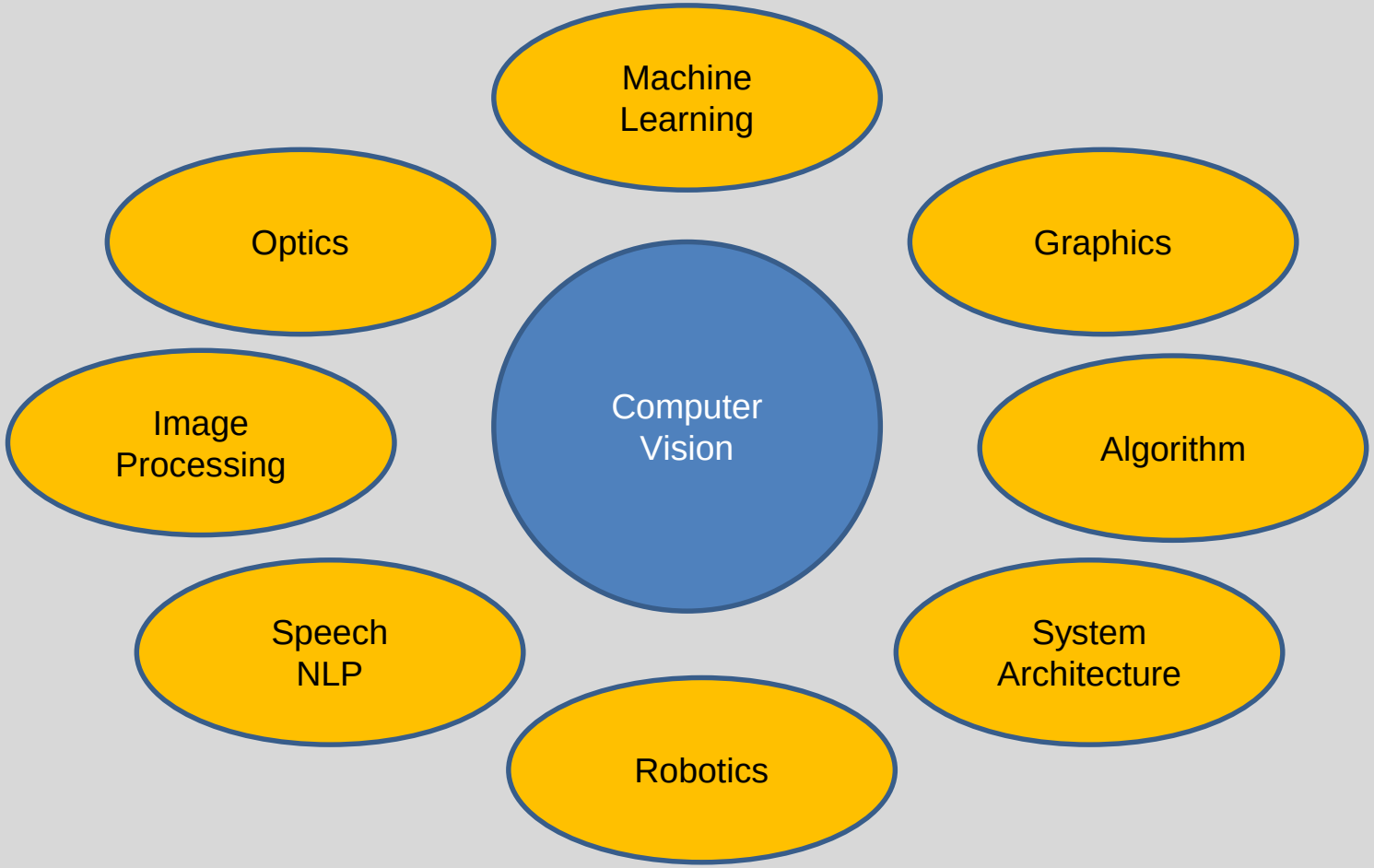
- **Case 1**: Looking at someone else's homework.
- **Case 2**: Copying the solution in preparing homework.
- **Case 3**: Copying someone else's programming code
- **Case 4**: Copying code found from the Internet.
- **Case 5**: Helping others in preparing their homework or projects, without authorization.

# Q&A

- Over 50 students are taking this class.
- I highly recommend you to use the Q&A board in the BB for Q&A. (Answers will be made within 48 hours.)



# What is computer vision?



# What is deep learning?


**Sensational Go Match between AlphaGO and Lee Sedol (2016)**

AlphaGO won Lee Sedol by 4 vs. 1

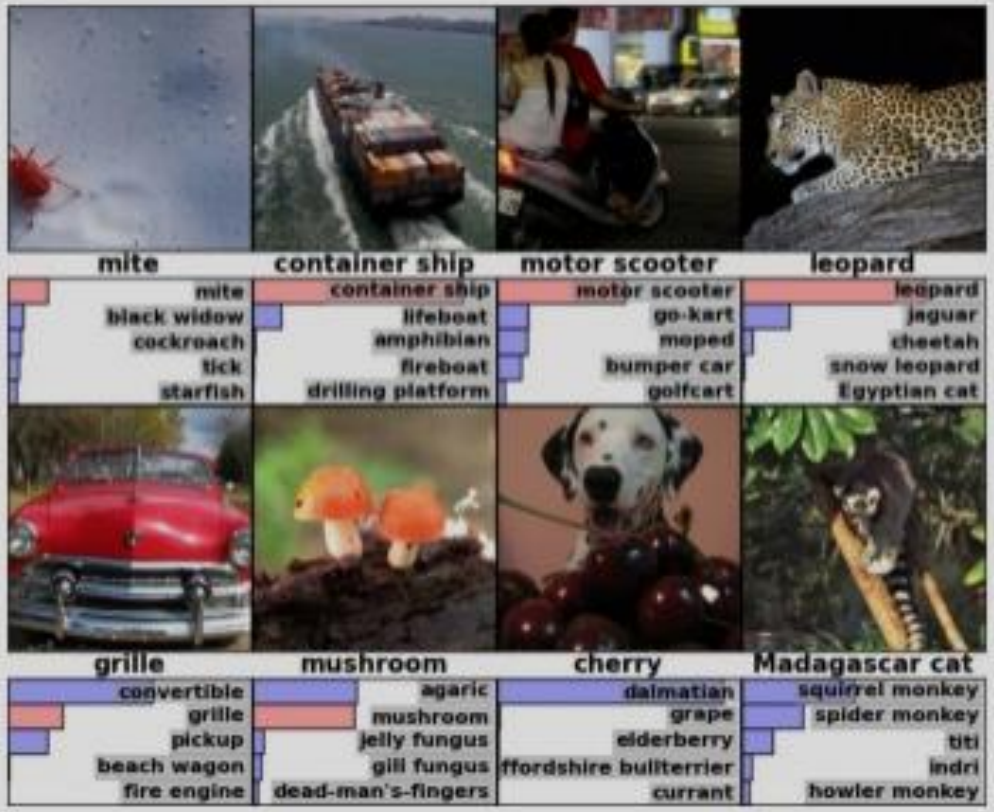
AlphaGo is using a deep learning when training their parameters.

# What is deep learning?

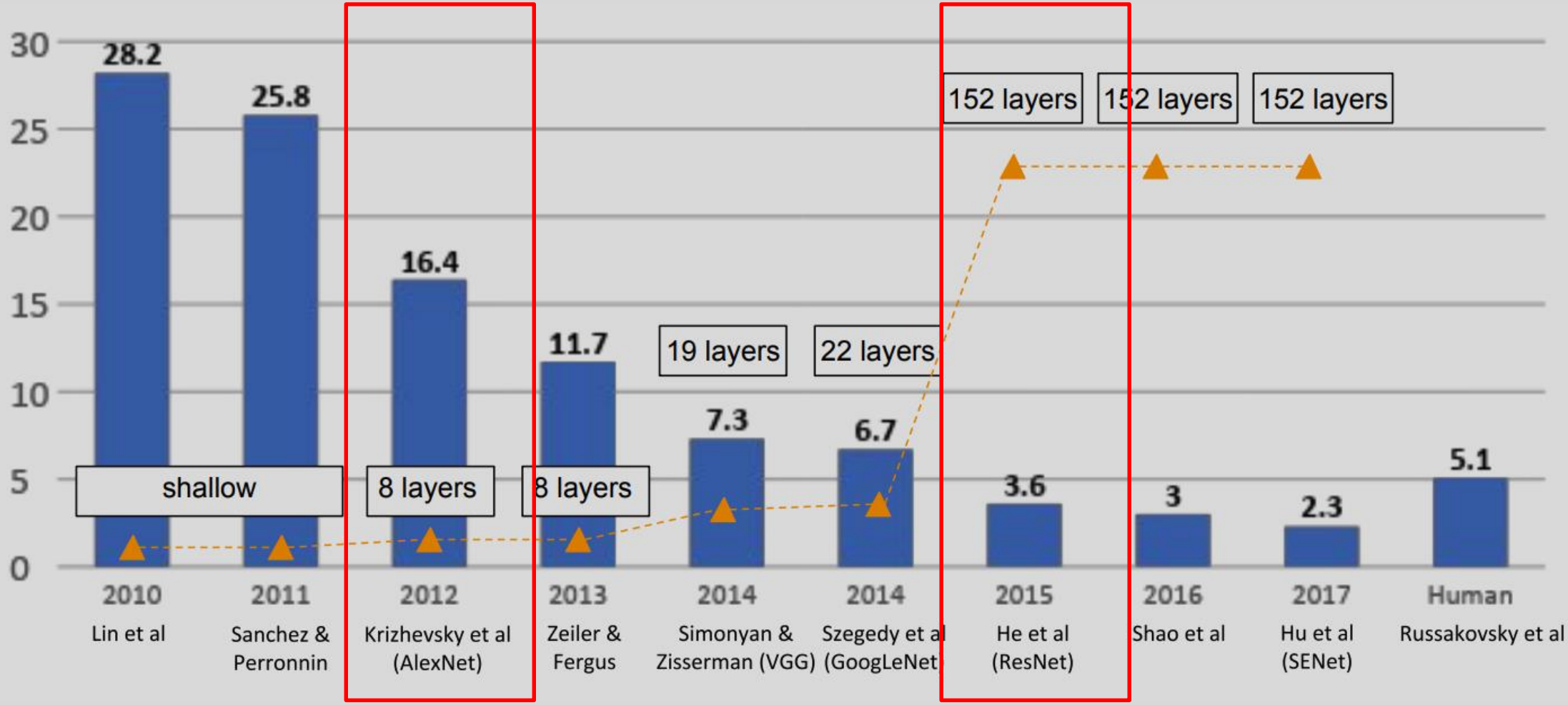
## ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.



# ImageNet winners (getting deeper)



# Computer vision application

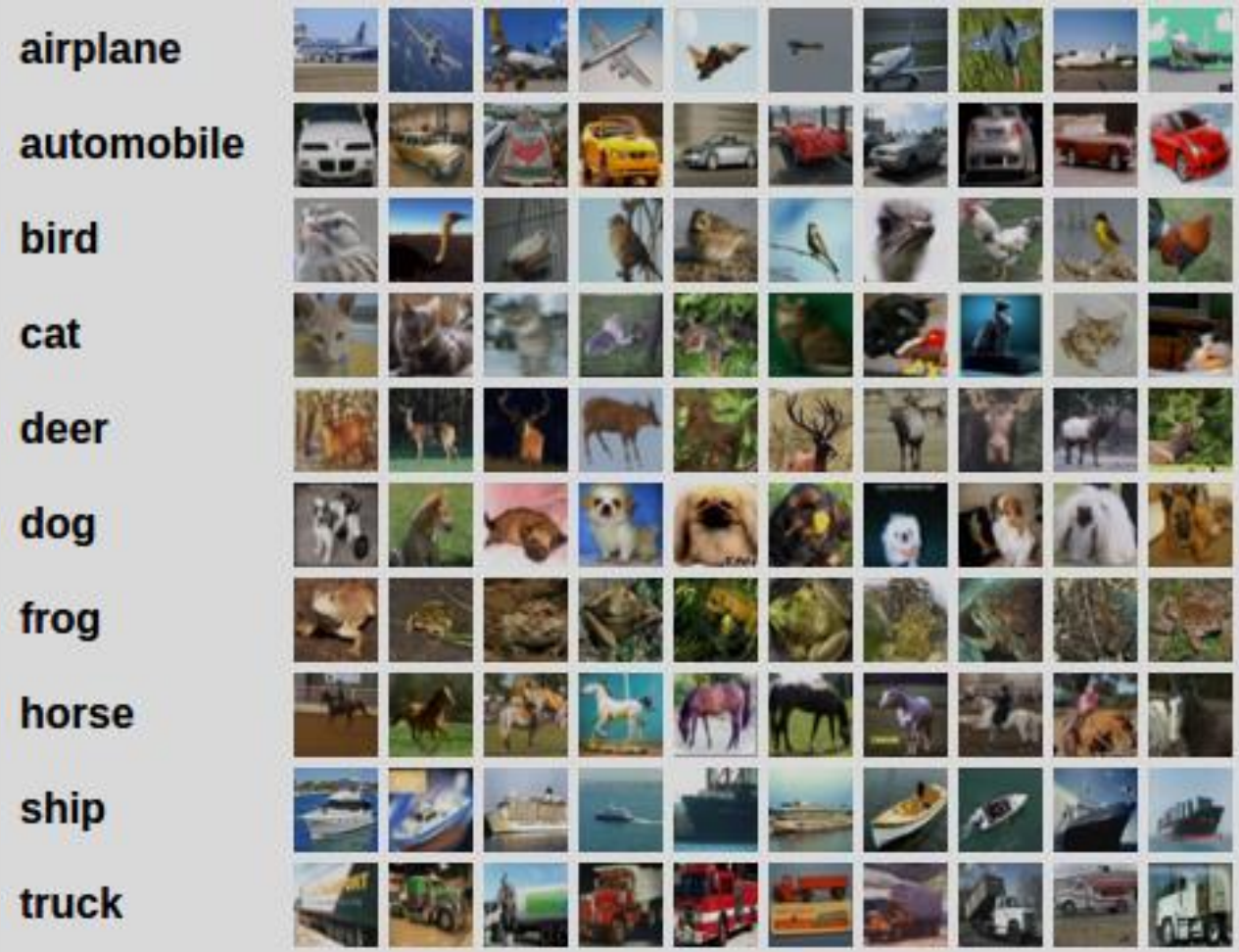
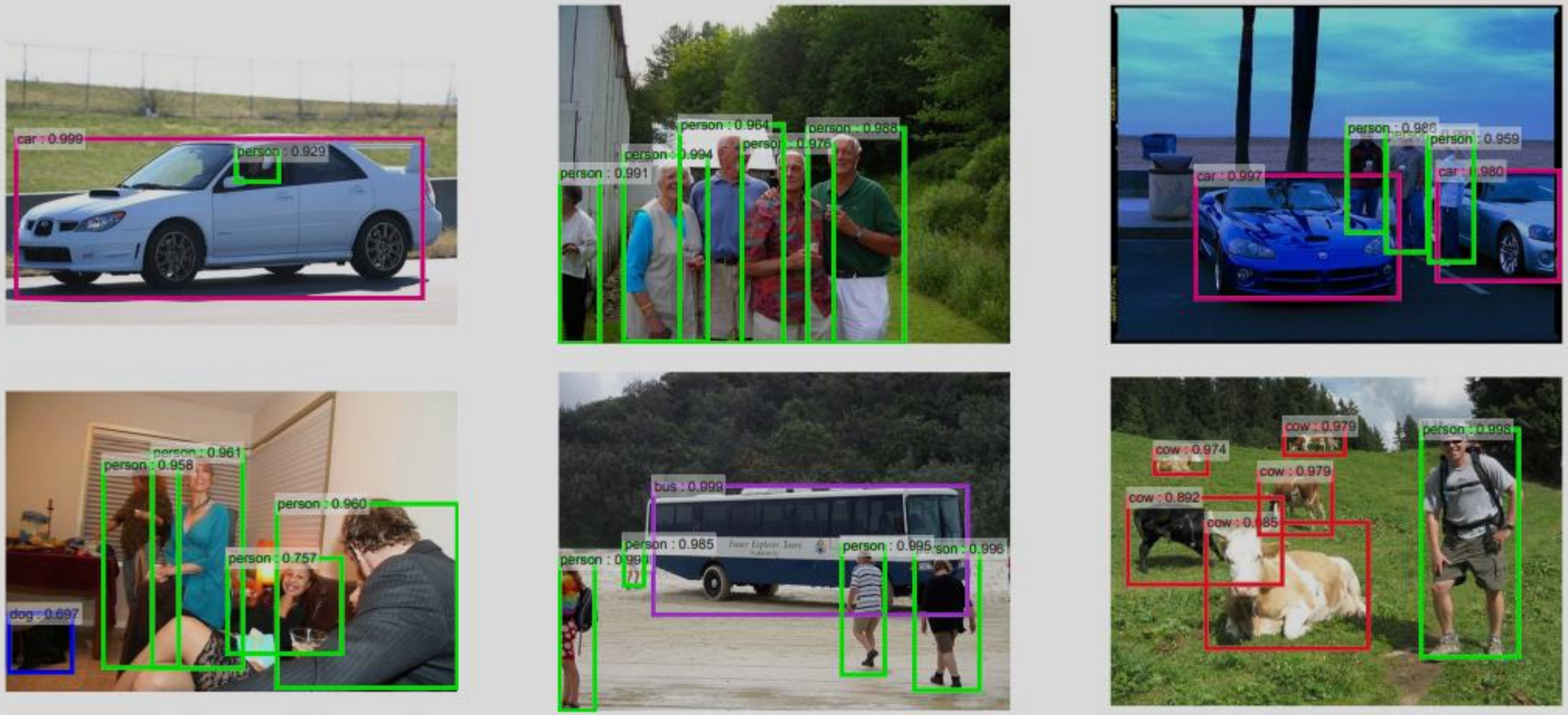



Image classification for Cifar-10 dataset.

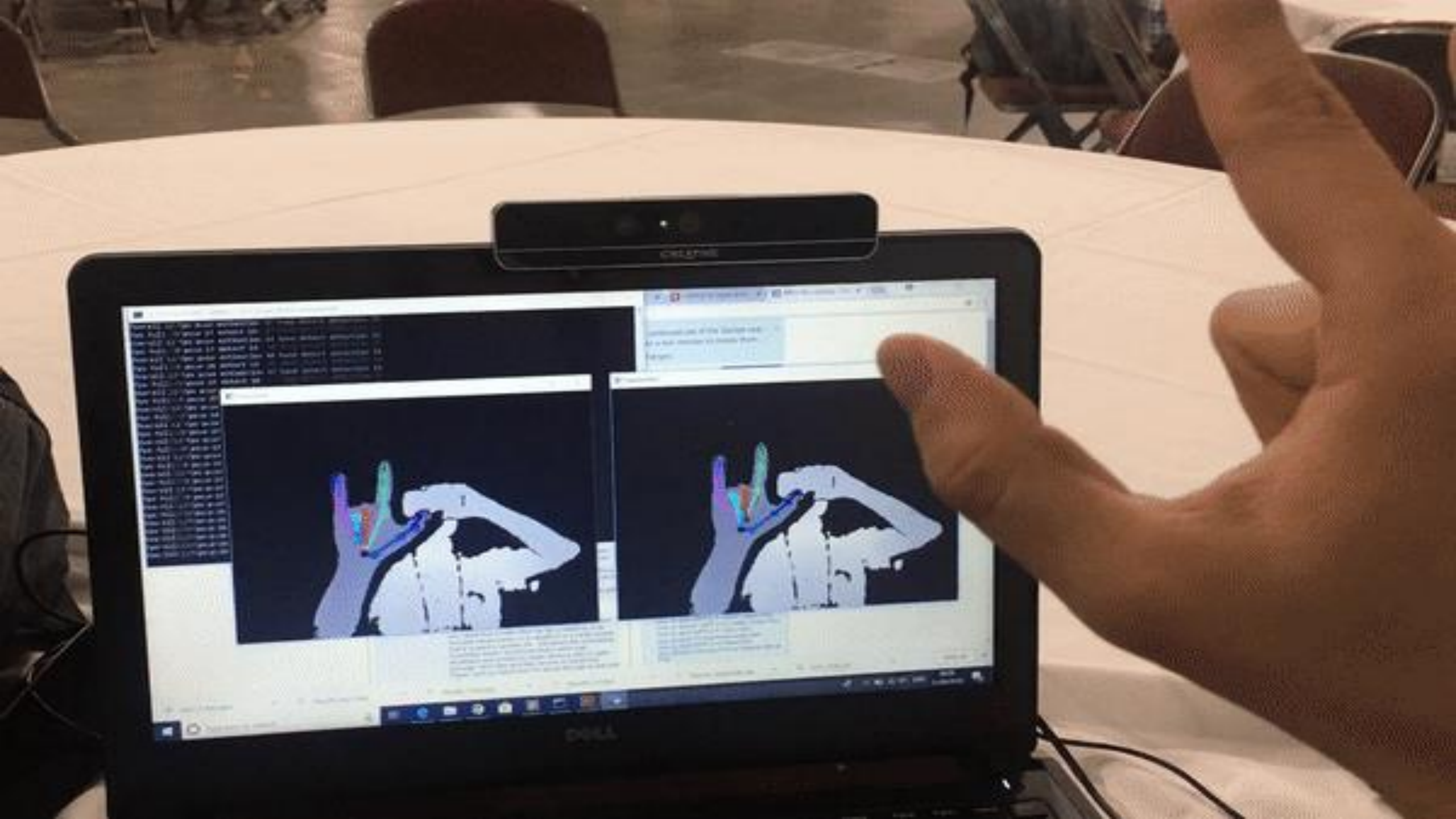
# Computer vision application



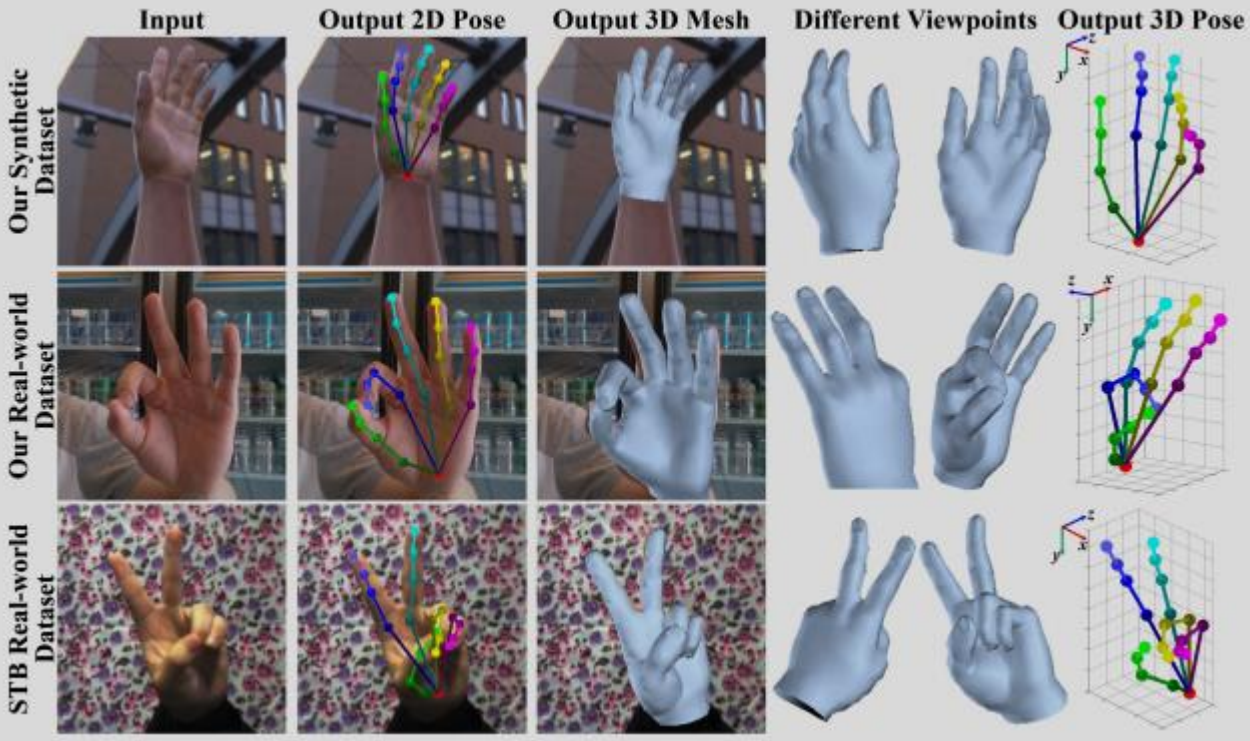

Detecting object locations. [Faster-RCNN NIPS'15]

# Computer vision application

Detecting object locations and segmentation. [Mask RCNN ICCV'17]



# Computer vision application




3D hand mesh reconstruction (Ge et al. CVPR'19)


3D human mesh reconstruction (Kanazawa et al. CVPR'18)

# Computer vision application


Image attribute translation (Binod et al. ICASSP'20)

# Google colab

- Free cloud service with GPU for AI developers who have gmail address.
- It is provided by Google.
- We can use python and its libraries (OpenCV, PyTorch, Tensorflow,…)
- http://colab.research.google.com

# Google colab

- 1) make a new notebook.
- 2) Type python code and it simply works!

```
print('hello world')
```

# Open CV Library

- Open source computer vision library.
- Provides easy interface to code image processing.
- Compatible with C/C++, JAVA, Python and etc.
- In python, do `import cv2`
- Files required are uploaded in this link:

https://drive.google.com/drive/folders/1Gq30m_MtY6N7-hO8dSo3vYD6FEE2aNf3?usp=sharing

# OpenCV - Image read

```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)
cv2_imshow(img)
```

# OpenCV - Image read

```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_GRAYSCALE)
cv2_imshow(img)
```

# OpenCV - Image shape

```
img.shape
```

```
(454, 680, 3)
```

Width of image, height of image, channel # of image.

```
img[0, 0]
```

```
array([182, 115, 0], dtype=uint8)
```

RGB values for x=1, y=1 pixels.

```
img[0, 0, 0]
```

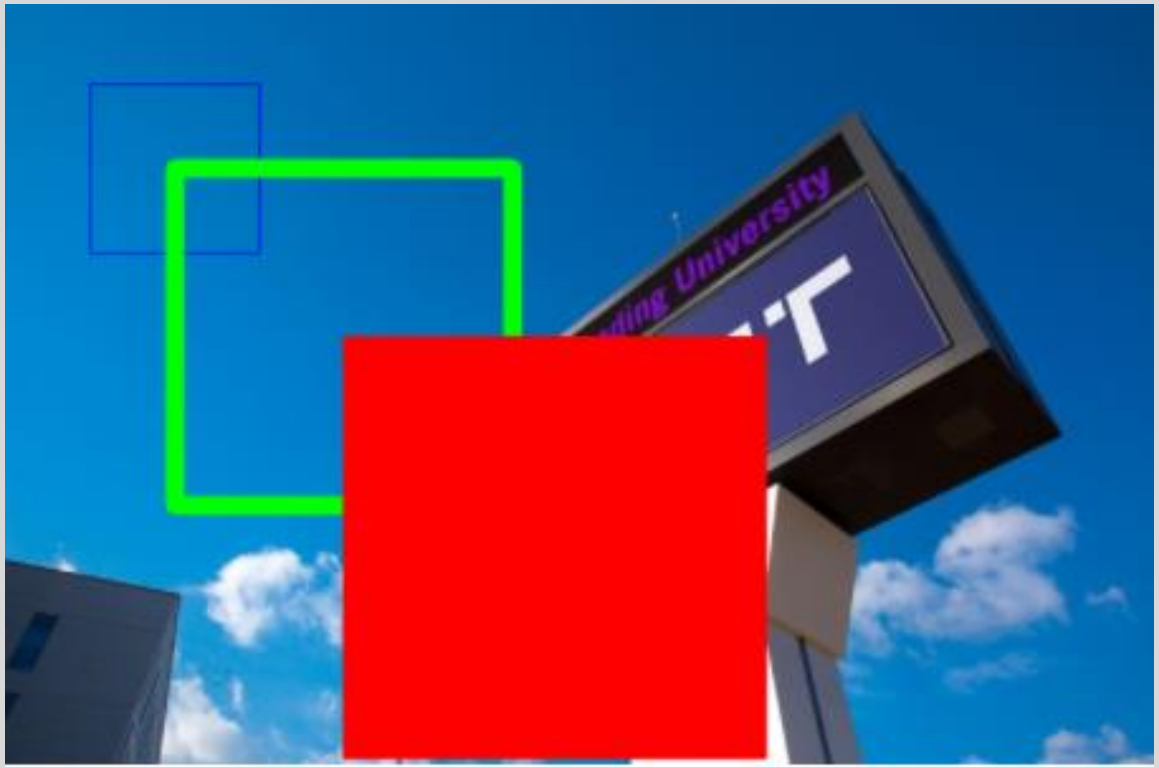
```
182
```

# OpenCV - Image channel

```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)

cv2_imshow(img[:, :, 0]) # showing blue channels only.
```

# OpenCV – Draw rectangles

```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)

cv2.rectangle(img, (50, 50), (150,150), (255, 0, 0))
cv2.rectangle(img, (300, 300), (100,100), (0, 255, 0), 10)
cv2.rectangle(img, (450, 200), (200,450), (0, 0, 255), -1)

cv2_imshow(img)
```



# OpenCV – Draw polylines

```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)

import numpy as np
pts1 = np.array([[50,50], [150,150], [100,140], [200,240]], dtype=np.int32)
cv2.polylines(img, [pts1], False, (255,0,0))

cv2_imshow(img)
```

# OpenCV – Draw circles

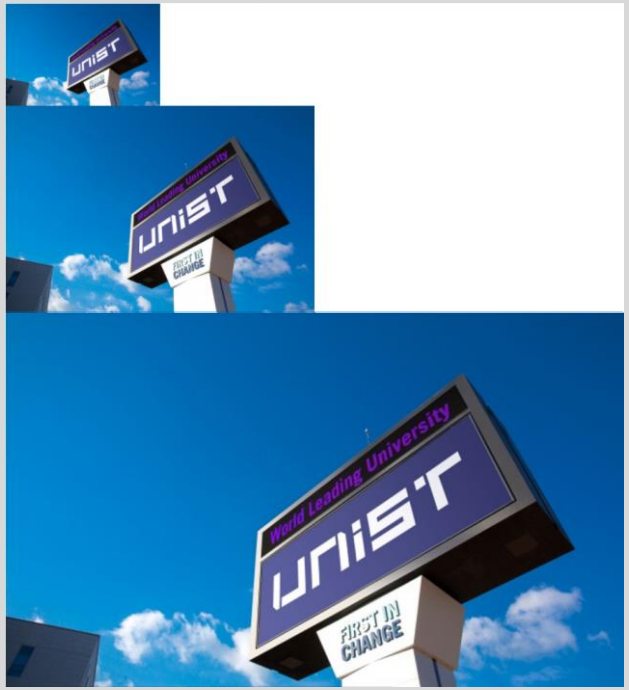
```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)

cv2.circle(img, (150,150), 100, (255,0,0))
cv2.ellipse(img, ((325,300), (150,100), 0), (0, 255,0), 5)

cv2_imshow(img)
```

# OpenCV – Put texts

```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)

cv2.putText(img, 'UNIST CSE48001 Computer Vision!', (20,50), cv2.FON
T_HERSHEY_PLAIN, 2, (0,255,0))

cv2_imshow(img)
```

# OpenCV – Image resize

```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)

height, width = img.shape[:2]
dst1 = cv2.resize(img, None, None, 0.5, 0.5, cv2.INTER_CUBIC) #decrease image size by ½.
dst2 = cv2.resize(img, None, None, 2, 2, cv2.INTER_CUBIC) #increase image size by 2.

cv2_imshow(dst1)
cv2_imshow(img)
cv2_imshow(dst2)
```

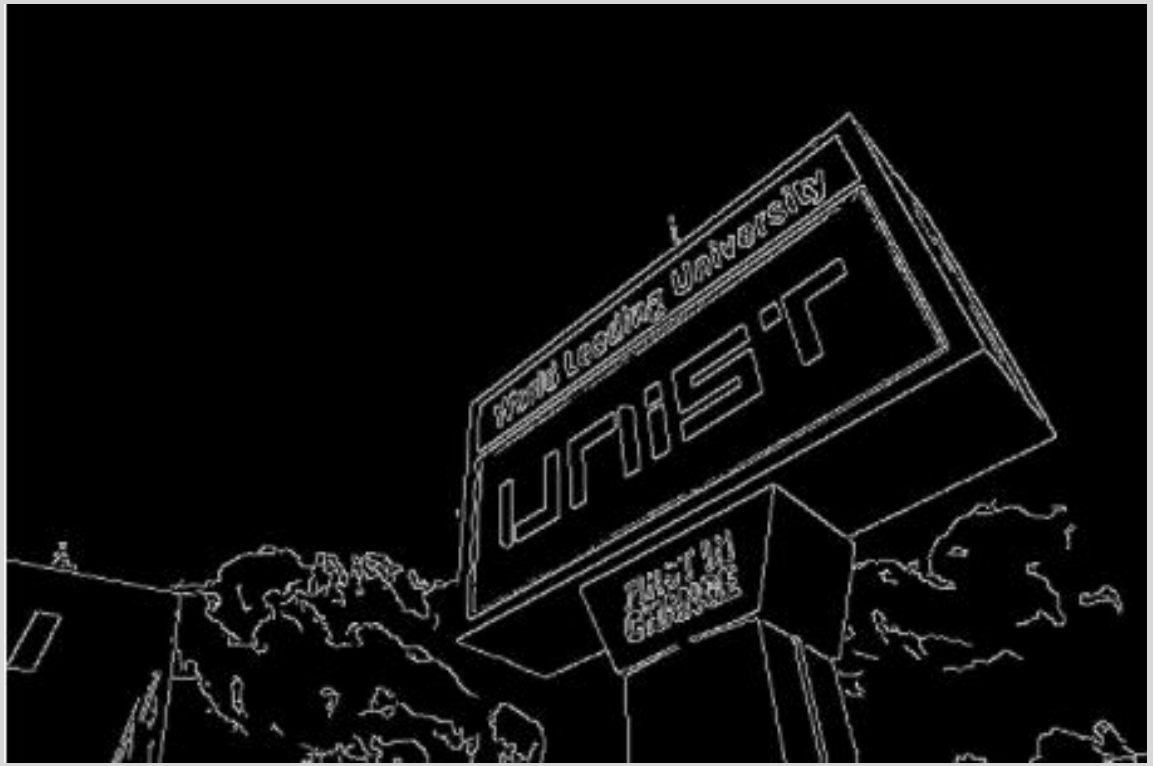


# OpenCV – Image blurring

```
from google.colab.patches import cv2_imshow
import cv2

img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)

blur1 = cv2.blur(img, (10,10))

cv2_imshow(blur1)
```

# OpenCV – Edge detection

```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)

edges = cv2.Canny(img, 100, 200)

cv2_imshow(edges)
```



# OpenCV – Corner detection

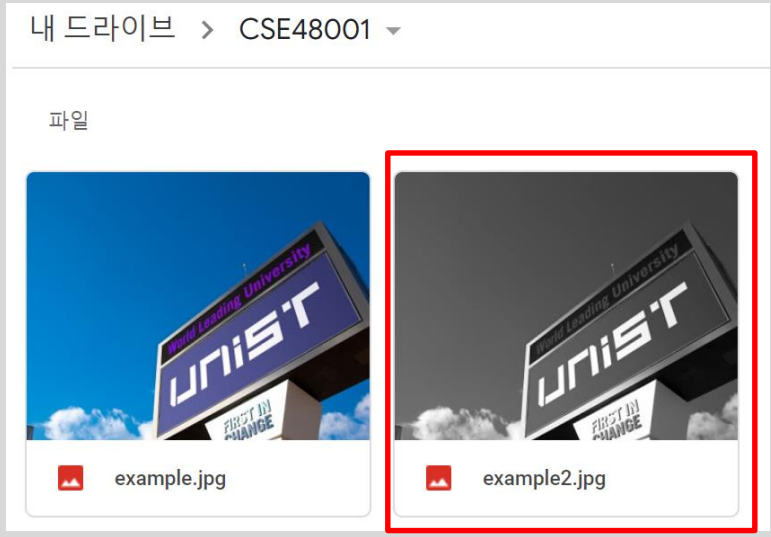
```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
corner = cv2.cornerHarris(gray, 2, 3, 0.04)
coord = np.where(corner > 0.1*corner.max())
coord = np.stack((coord[1], coord[0]), axis=-1)

for x, y in coord:
  cv2.circle(img, (x,y), 5, (0,0,255), 1, cv2.LINE_AA)

cv2_imshow(img)
```

# OpenCV – Image matching

```
from google.colab.patches import cv2_imshow
import cv2

img1 = cv2.imread('/content/example.jpg', cv2.IMREAD_UNCHANGED)
img2 = cv2.imread('/content/example4.jpg', cv2.IMREAD_UNCHANGED)
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

detector = cv2.ORB_create()
kp1, desc1 = detector.detectAndCompute(gray1, None)
kp2, desc2 = detector.detectAndCompute(gray2, None)

matcher = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
matches = matcher.match(desc1, desc2)

res = cv2.drawMatches(img1, kp1, img2, kp2, matches, None, flags=cv2.DRAW_MATCHES_FLAGS_NOT_D
RAW_SINGLE_POINTS)

cv2_imshow(res)
```

# OpenCV – Image save

```
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('/content/example.jpg', cv2.IMREAD_GRAYSCALE)
cv2_imshow(img)

save_file = '/content/example2.jpg'
cv2.imwrite(save_file, img)
```




New image is saved.

# OpenCV – Video read

```
from google.colab.patches import cv2_imshow
import cv2

cap = cv2.VideoCapture('/content/example.avi')
while True:
  ret, img = cap.read()
  if ret:
    cv2_imshow(img)
```


Thank you!
ULSAN NATIONAL INSTITUTE OF
SCIENCE AND TECHNOLOGY
2 0 0 7