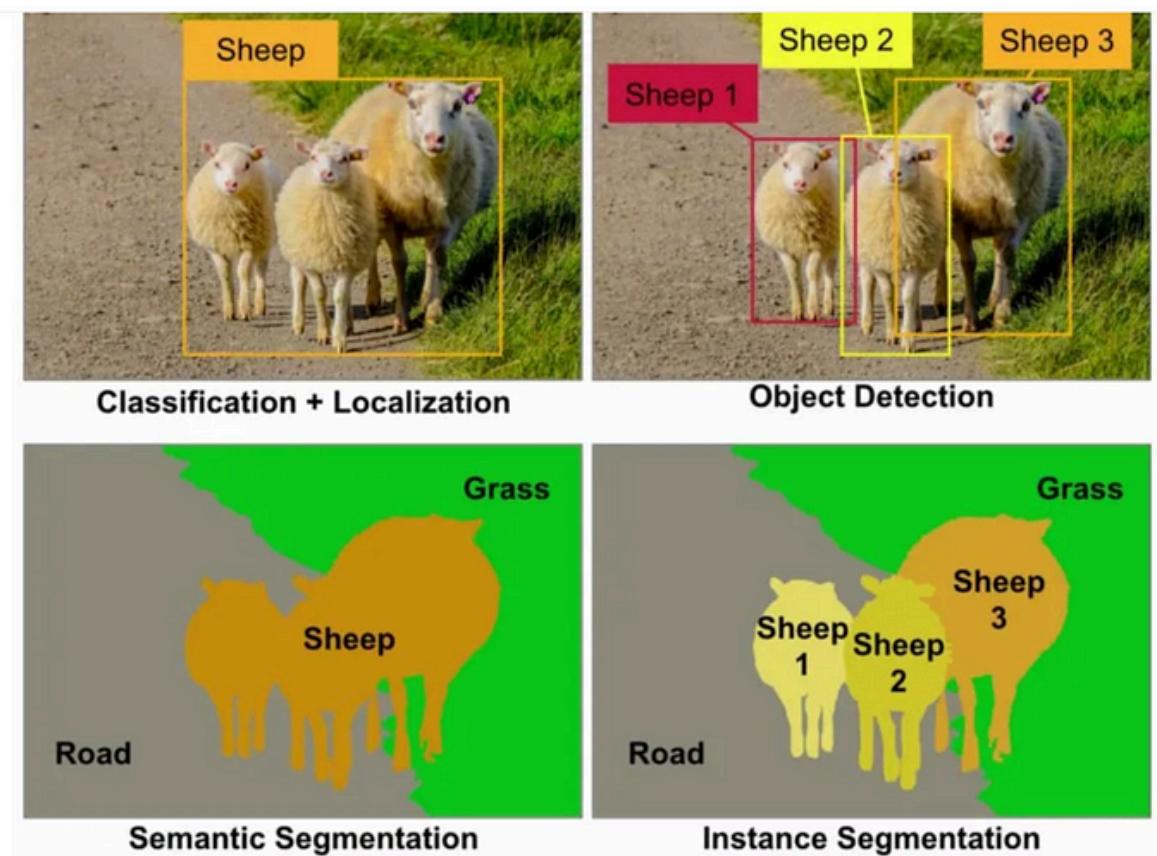


Сегментація

Сегментація — це процес розділення цифрового зображення на декілька сегментів.

Сегментація об'єктів — це проблема комп'ютерного зору, яка включає виявлення та розрізнення окремих об'єктів або цікавих областей на зображенні. Це також відомо як сегментація зображення або екземпляра.



Метою сегментації об'єктів є надання кожному елементу або екземпляру на зображенні унікальної мітки або ідентифікації, а також точного визначення меж цих речей.

Сегменти якісно просегментованого зображення повинні бути однорідними за текстурою, межі виділених сегментів повинні бути чіткими, сусідні сегменти повинні відрізнятися за певними критеріями.

Для отримання сегментації зображень було обрано нейромережі опираючись на цей сайт: <https://roboflow.com/models/instance-segmentation>. Надалі вам буде показано спосіб завантаження, запуск та результати кожної доступної з цього статі.

YOLOv

YOLO (You Only Look Once) — це група моделей виявлення об'єктів, які славляться своєю швидкістю та точністю. Щоб сегментувати за допомогою YOLO, можна розширити модель виявлення об'єктів YOLO, щоб передбачити піксельні маски для кожного об'єкта, знайденого на зображенні.

YOLOv5-<https://github.com/ultralytics/yolov5>

YOLOv7-<https://github.com/WongKinYiu/yolov7>

YOLOv8-<https://github.com/ultralytics/ultralytics>

Розглянемо YOLOv, які мають окремий вид - сегментацію (5, 7 та 8 версії). На цей час всі версії YOLOv навчені на датасеті COCO, тож доступними є **80 класів**:

0: person	27: tie	54: donut
1: bicycle	28: suitcase	55: cake
2: car	29: frisbee	56: chair
3: motorcycle	30: skis	57: couch
4: airplane	31: snowboard	58: potted plant
5: bus	32: sports ball	59: bed
6: train	33: kite	60: dining table
7: truck	34: baseball bat	61: toilet
8: boat	35: baseball glove	62: tv
9: traffic light	36: skateboard	63: laptop
10: fire hydrant	37: surfboard	64: mouse
11: stop sign	38: tennis racket	65: remote
12: parking meter	39: bottle	66: keyboard
13: bench	40: wine glass	67: cell phone
14: bird	41: cup	68: microwave
15: cat	42: fork	69: oven
16: dog	43: knife	70: toaster
17: horse	44: spoon	71: sink
18: sheep	45: bowl	72: refrigerator
19: cow	46: banana	73: book
20: elephant	47: apple	74: clock
21: bear	48: sandwich	75: vase
22: zebra	49: orange	76: scissors
23: giraffe	50: broccoli	77: teddy bear
24: backpack	51: carrot	78: hair drier
25: umbrella	52: hot dog	79: toothbrush
26: handbag	53: pizza	

Встановлення

Перед запуском на моєму комп'ютері нейромережі, я виконала такі дії:

- спершу завантажила Yolov5/7/8. Використовувала інструкції за записом Yolov5/7/8 GitHub Segmentation;

YOLOv5

```
pip3 install ultralytics  
git clone https://github.com/ultralytics/yolov5 # clone  
cd yolov5  
pip3 install -r requirements.txt # install
```

YOLOv7

```
git clone https://github.com/RizwanMunawar/yolov7-segmentation.git  
cd yolov7-segmentation  
pip3 install --upgrade pip  
pip3 install -r requirements.txt
```

YOLOv8

```
pip3 install ultralytics  
git clone https://github.com/ultralytics/ultralytics.git
```

- завантажила усі доступні для сегментації моделі;
- у Yolov5 змінила файл predict.py, аби не відображалась рамка на сегментації, такий варіант не спрацював на Yolov7(пізніше виконаємо покращення). У Yolov8 немає файлу доступного для змін відображення рамок.

Тестування

Файли у:

okichi@okichi-System-Product-Name:/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/yolo_segment/

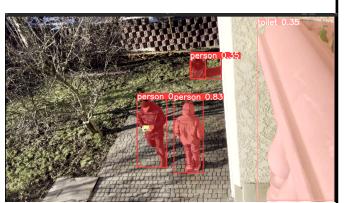
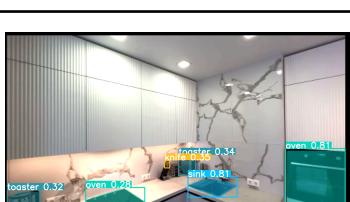
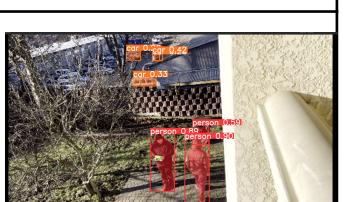
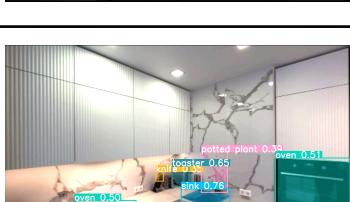
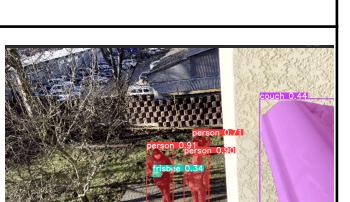
Зображення:

Модель	Ваги	Команда	Зображення 1	Зображення 5
Yolov5	n	yolov5/data\$ python3 segment/predict.py --weights model/yolov5n-seg.pt --img 640 --source images/1.jpeg		
	s			
	m			
	l			
	x			

Yolov7		<pre>yolov7-segmentation\$ python3 segment/predict.py --weights model/yolov7- seg.pt --source images/</pre>	<p>Detailed description: This image shows a segmentation output for a Yolov7 model. It features four bounding boxes: a purple one for a cat with a confidence of 0.64, a blue one for a dog with 0.88, a blue one for a bird with 0.86, and a red one for a person with 0.82. The background is a green lawn.</p>	<p>Detailed description: This image shows a heatmap overlay for a Yolov7 model. It highlights a child hugging a dog. The heatmap is primarily red and orange, indicating high confidence in the presence of a person and a dog.</p>
Yolov8	n	<pre>ultralytics\$ yolo predict segment model=model /yolov8n-seg. pt source=image s/</pre>	<p>Detailed description: This image shows a segmentation output for a Yolov8n model. It features four bounding boxes: a purple one for a dog with 0.53, a purple one for a cat with 0.69, a blue one for a bird with 0.79, and a purple one for a hedgehog with 0.32. The background is a green lawn.</p>	<p>Detailed description: This image shows a heatmap overlay for a Yolov8n model. It highlights a child hugging a dog. The heatmap is primarily red and orange, indicating high confidence in the presence of a person and a dog.</p>
	s		<p>Detailed description: This image shows a segmentation output for a Yolov8s model. It features four bounding boxes: a purple one for a dog with 0.91, a purple one for a cat with 0.69, a blue one for a bird with 0.90, and a purple one for a hedgehog with 0.32. The background is a green lawn.</p>	<p>Detailed description: This image shows a heatmap overlay for a Yolov8s model. It highlights a child hugging a dog. The heatmap is primarily red and orange, indicating high confidence in the presence of a person and a dog.</p>
	m		<p>Detailed description: This image shows a segmentation output for a Yolov8m model. It features four bounding boxes: a purple one for a cat with 0.81, a blue one for a dog with 0.88, a blue one for a bird with 0.90, and a red one for a person with 0.71. The background is a green lawn.</p>	<p>Detailed description: This image shows a heatmap overlay for a Yolov8m model. It highlights a child hugging a dog. The heatmap is primarily red and orange, indicating high confidence in the presence of a person and a dog.</p>
	l		<p>Detailed description: This image shows a segmentation output for a Yolov8l model. It features four bounding boxes: a purple one for a cat with 0.66, a blue one for a dog with 0.92, a blue one for a bird with 0.95, and a red one for a person with 0.74. The background is a green lawn.</p>	<p>Detailed description: This image shows a heatmap overlay for a Yolov8l model. It highlights a child hugging a dog. The heatmap is primarily red and orange, indicating high confidence in the presence of a person and a dog.</p>
	x		<p>Detailed description: This image shows a segmentation output for a Yolov8x model. It features four bounding boxes: a purple one for a cat with 0.85, a blue one for a dog with 0.91, a blue one for a bird with 0.93, and a red one for a person with 0.51. The background is a green lawn.</p>	<p>Detailed description: This image shows a heatmap overlay for a Yolov8x model. It highlights a child hugging a dog. The heatmap is primarily red and orange, indicating high confidence in the presence of a person and a dog.</p>

Відео:

Модель	Ваги	Команда	Відео Кухня	Відео Наше
Yolov5	n	yolov5/data\$ python3 segment/predict.py --weights model/yolov5x-seg.pt --img 640 --conf 0.25 --source data/video		
	s			
	m			
	l			
	x			
Yolov7		yolov7-segmentation\$ python3 segment/predict.py --weights model/yolov7-seg.pt --source video/		

Yolov8	n	ultralytics\$ yolo predict segment model=model/ yolov8x-seg.pt source=IMG_9 842.MP4		
	s			
	m			
	l			
	x			

Підсумок

ПЕРЕВАГИ	НЕДОЛІКИ
Легко використовувати.	Модель Yolov8l і x може злетіти, через велику загруженість.
Через її популярність більшість помилок вже вирішено.	

Здатна не лише до сегментації.	
Великий вибір моделей (Yolov5/8 - доступні 5 моделей, Yolov7- 1 модель).	
Можливість до перенавчання моделей.	
Легко користуватись.	
Можна сегментувати декілька фото/відео водночас.	
На більшість помилок вже є рішення.	

SAM

Segment Anything, випущений Meta Research у квітні 2023 року, — це модель комп’ютерного зору сегментації зображення, навчена за допомогою нового набору даних. Сама модель називається Segment Anything Model (SAM). Сама назва говорить про те, що нейромережа сегментує усі класи.

GitHub- <https://github.com/facebookresearch/segment-anything>

Встановлення

(шлях до папки okichi@okichi-System-Product-Name:/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/sam/):

```
git clone https://github.com/facebookresearch/segment-anything.git
wget -q https://dl.fbaipublicfiles.com/segmentAnything/sam\_vit\_h\_4b8939.pth
```

Тестування

Далі запустила SAM в тій же папці з такими параметрами, які описані у цьому розділі:

```
python3 scripts/amg.py --input
/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/sam/n
otebooks/images/truck.jpg --output
/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/sam/r
un --model-type vit_h --checkpoint
/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/sam/s
am_vit_h_4b8939.pth --points-per-batch 4
```

Параметр --points-per-batch додала (тобто не є обов'язковим). Завершення на помилці при запуску :RuntimeError: CUDA out of memory. Tried to allocate 20.00 MiB (GPU 0; 3.80 GiB total capacity; 2.42 GiB already allocated; 43.06 MiB free; 2.47 GiB reserved in total by PyTorch), не зникає навіть при мінімальному параметрі (batch 1).

Робота припинена.

Підсумок

ПЕРЕВАГИ	НЕДОЛКИ
є можливість поставити маску людини і тоді нейромережа сегментуватиме лише людей (не знатиме, що це - людина)	потребує багато пам'яті
сегментує усе	так і не вдалось запустити
сегентацію можна поділити на окремі виділені шари та розглянути кожен з них	уникає підписи класів
можна вручну виставляти маску тикаючи на потрібний предмет на зображені.	

Для більшого розуміння:

<https://blog.roboflow.com/how-to-use-segment-anything-model-sam/#loading-the-segment-anything-model>

<https://inside-machinelearning.com/en/sam-segment-anything-tutorial/>

Mask_RCNN

Маска RCNN - це згортка нейронної мережі для сегментації екземплярів. Mask R-CNN є розширенням Faster R-CNN, де паралельно проходить прогнозування маски об'єкта і розпізнавання обмежувальної рамки.

GitHub- https://github.com/matterport/Mask_RCNN

Встановлення

Будемо завантажувати Детектор у якому є Mask-RCNN:<https://github.com/facebookresearch/Detectron>.
Detectron (детектор) — це програмна система Facebook AI Research, яка реалізує найсучасніші алгоритми виявлення об'єктів, зокрема Mask R-CNN. Він написаний на Python і працює на базі глибокого навчання Caffe2.

Заходячи у файл: README.md та INSTALL.md знаходимо інструкцію завантаження. Ось як це виглядало в мене:

```
python3 -c 'from caffe2.python import core' 2>/dev/null && echo "Success" ||  
echo "Failure"  
python3 -c 'from caffe2.python import workspace;  
print(workspace.NumCudaDevices())'  
This must print a number > 0 in order to use Detectron  
git clone https://github.com/cocodataset/cocoapi.git  
cd $COCOAPI/PythonAPI  
sudo python3 setup.py install
```

Якщо виникає така помилка: Beginning with Matplotlib 3.8, Python 3.9 or above is required. Make sure you have pip >= 9.0.1.. Тоді:
sudo apt install python3.9 -y
sudo python3.9 -m pip install -U --user numpy
sudo python3.9 setup.py install

Якщо виникає така помилка: compilation terminated.
error: command '/usr/bin/x86_64-linux-gnu-gcc' failed with exit code 1. Тоді:
sudo apt-get install python3.9-dev
sudo python3 setup.py install

```
git clone https://github.com/facebookresearch/detectron $DETECTRON  
cd detectron
```

```
pip3 install -r requirements.txt  
sudo python3 setup.py install
```

Якщо виникає така помилка: ModuleNotFoundError: No module named 'Cython'. Тоді:
sudo pip3 install Cython

Якщо виникає така помилка:ModuleNotFoundError: No module named 'numpy'. Тоді:
sudo pip3 install numpy
sudo python3 setup.py install
python3 detectron/tests/test_spatial_narrow_as_op.py

Якщо виникає така помилка:ModuleNotFoundError: No module named 'detectron.utils.c2'. Тоді:
sudo python3 detectron/detectron/tests/test_spatial_narrow_as_op.py

Якщо виникає така помилка:ModuleNotFoundError: No module named 'caffe2'. Тоді:
sudo apt-get install -y --no-install-recommends build-essential git
libgoogle-glog-dev libgtest-dev libomp-dev libleveldb-dev
liblmdb-dev libopencv-dev libopenmpi-dev libsnavy-dev
libprotobuf-dev openmpi-bin openmpi-doc protobuf-compiler
python-dev python3-pip
pip3 install --user future numpy protobuf typing hypothesis
sudo apt-get install -y --no-install-recommends \
libgflags-dev \
cmake

Якщо виникає така помилка:RuntimeError: no cmake or cmake3 with version >= 3.18.0 found. Тоді:
sudo apt remove cmake -y
pip3 install cmake --upgrade
cd ~ && python3 -c 'from caffe2.python import core' 2>/dev/null && echo
"Success" || echo "Failure"
#якщо пише Success продовжуємо
cd
/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/detect
ron/
python3 detectron/tests/test_spatial_narrow_as_op.py

#якщо видало OK- працюємо далі.

Тестування

Для цього попередньо ознайомлюємось з файлом GETTING_STARTED.md.

Команда запуску у мене виглядатиме так:

```
python3 tools/infer_simple.py --cfg  
configs/12_2017_baselines/e2e_mask_rcnn_R-101-FPN_2x.yaml --output-dir  
runs/exp1 --image-ext image/ --wts  
https://dl.fbaipublicfiles.com/detectron/35861858/12_2017_baselines/e2e_mask  
_rcnn_R-101-FPN_2x.yaml.02_32_51.SgT4y1cO/output/train/coco_2014_train:  
coco_2014_valminusminival/generalized_rcnn/model_final.pkl demo
```

Попередньо створила папку runs, де у папках exp будуть зберігатись результати; папку images, де є зображення для тестування.

Але якщо виникає така помилка: from detectron.core.config import assert_and_infer_cfg
ModuleNotFoundError: No module named 'detectron.core'. Спробує запустити з sudo, що видає помилку:ModuleNotFoundError: No module named 'caffe2'. Тому спробуємо вирішити помилку першого варіанту.

Робота припинена над такою помилкою:

```
okichi@okichi-System-Product-Name:/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/mask_rcnn/detectron$ pip3  
install -r requirements.txt
```

```
Defaulting to user installation because normal site-packages is not writeable
```

```
Requirement already satisfied: numpy>=1.13 in /home/okichi/.local/lib/python3.8/site-packages (from -r requirements.txt (line 1)) (1.24.4)
```

```
Collecting pyyaml==3.12 (from -r requirements.txt (line 2))
```

```
Using cached PyYAML-3.12.zip (375 kB)
```

```
Preparing metadata (setup.py) ... error
```

```
error: subprocess-exited-with-error
```

```
  × python setup.py egg_info did not run successfully.
```

```
| exit code: 1
```

```
↳ [34 lines of output]
```

```
Traceback (most recent call last):
```

```
  File "<string>", line 2, in <module>
```

```
  File "<pip-setuptools-caller>", line 34, in <module>
```

```
  File "/tmp/pip-install-vankzpvh/pyyaml_cddb1f3cbce54afd89fc994a1bcbdba5/setup.py", line 317, in <module>
```

```
setup()

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/_distutils/core.py", line 185, in setup
    return run_commands(dist)

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/_distutils/core.py", line 201, in run_commands
    dist.run_commands()

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/_distutils/dist.py", line 969, in run_commands
    self.run_command(cmd)

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/dist.py", line 963, in run_command
    super().run_command(command)

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/_distutils/dist.py", line 988, in run_command
    cmd_obj.run()

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/command/egg_info.py", line 321, in run
    self.find_sources()

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/command/egg_info.py", line 329, in find_sources
    mm.run()

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/command/egg_info.py", line 551, in run
    self.add_defaults()

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/command/egg_info.py", line 589, in add_defaults
    sdist.add_defaults(self)

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/command/sdist.py", line 112, in add_defaults
    super().add_defaults()

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/_distutils/command/sdist.py", line 251, in add_defaults
    self._add_defaults_ext()

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/_distutils/command/sdist.py", line 336, in _add_defaults_ext
    self.filelist.extend(build_ext.get_source_files())

File "/tmp/pip-install-vankzph/pyyaml_cddb1f3cbce54afd89fc994a1bcbdb5/setup.py", line 178, in get_source_files
    self.cython_sources(ext.sources, ext)

File "/home/okichi/.local/lib/python3.8/site-packages/setuptools/_distutils/cmd.py", line 107, in __getattr__
    raise AttributeError(attr)

AttributeError: cython_sources

[end of output]
```

note: This error originates from a subprocess, and is likely not a problem with pip.

error: metadata-generation-failed

 × Encountered error while generating package metadata.

↳ See above for output.

note: This is an issue with the package mentioned above, not pip.

hint: See above for details.

Нормальна інструкція:

<https://blog.paperspace.com/mask-r-cnn-in-tensorflow-2-0/>

<https://github.com/ahmedfgad/Mask-RCNN-TF2> !!!SUPER

YOLACT

YOLACT — це проста, повністю згортка модель для сегментації примірників у реальному часі. Класи сегентації були перечислені раніше у YOLOv.

GitHub - <https://github.com/dbolya/yolact>

Встановлення

```
git clone https://github.com/dbolya/yolact.git
cd yolact
pip3 install cython
pip3 install opencv-python pillow pycocotools matplotlib
sudo apt-get install -y nvidia-kernel-open-545
sudo apt-get install -y cuda-drivers-545
wget
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64
/cuda-ubuntu2004.pin
sudo mv cuda-ubuntu2004.pin /etc/apt/preferences.d/cuda-repository-pin-600
wget
https://developer.download.nvidia.com/compute/cuda/12.3.2/local_installers/cu
da-repo-ubuntu2004-12-3-local_12.3.2-545.23.08-1_amd64.deb
sudo dpkg -i cuda-repo-ubuntu2004-12-3-local_12.3.2-545.23.08-1_amd64.deb
sudo cp /var/cuda-repo-ubuntu2004-12-3-local/cuda-*keyring.gpg
/usr/share/keyrings/
sudo apt-get update
sudo apt-get -y install cuda-toolkit-12-3
```

```
pip install cython
pip install opencv-python pillow pycocotools matplotlib
sh data/scripts/COCO.sh
sh data/scripts/COCO_test.sh
python3 eval.py --trained_model=weights/yolact_base_54_800000.pth
--score_threshold=0.15 --top_k=15 --display
```

Якщо виникає така помилка:`raise AssertionError("Torch not compiled with CUDA enabled")`

`AssertionError: Torch not compiled with CUDA enabled.` Тоді:
`pip install torch==1.5.0 torchvision==0.6.0 cudatoolkit=11.4.0`
`pip3 install --pre torch torchvision torchaudio -f`
`https://download.pytorch.org/whl/nightly/cu110/torch_nightly.html`

Якщо виникає така помилка: RuntimeError: Unexpected error from cudaGetDeviceCount(). Did you run some cuda functions before calling NumCudaDevices() that might have already set an error? Error 803: system has unsupported display driver / cuda driver combination.

Тоді за цим сайтом завантажуємо CUDA:

<https://www.cherryservers.com/blog/install-cuda-ubuntu>

Тестування

```
okichi@okichi-System-Product-Name:/media/okichi/3e36698e-36b3-4a6c-89a0  
-2167dc453786/project/segment/yolact$ python3 eval.py  
--trained_model=weights/yolact_resnet50_54_800000.pth  
--score_threshold=0.15 --top_k=15 --video_multiframe=1  
--video=video/street.MP4
```

```
python3 eval.py --trained_model=weights/yolact_resnet50_54_800000.pth  
--score_threshold=0.15 --top_k=15 --image=images/5.jpg
```

Якщо розміри зображення чи відео нам не підходять, то після 750 строки у файлі eval.py потрібно додати:

```
img = frame_buffer.get()  
img=cv2.resize(img,(640*2,480*2))  
cv2.imshow(path, img)
```

Результат

Назва	Витяг
Відео street.MP4	

Відео kithen.MP4

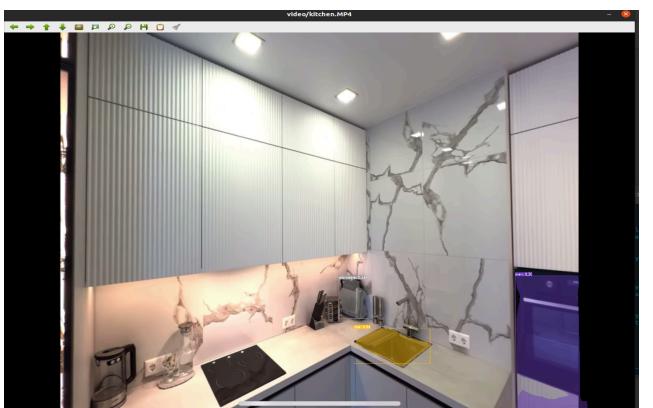


Фото 1

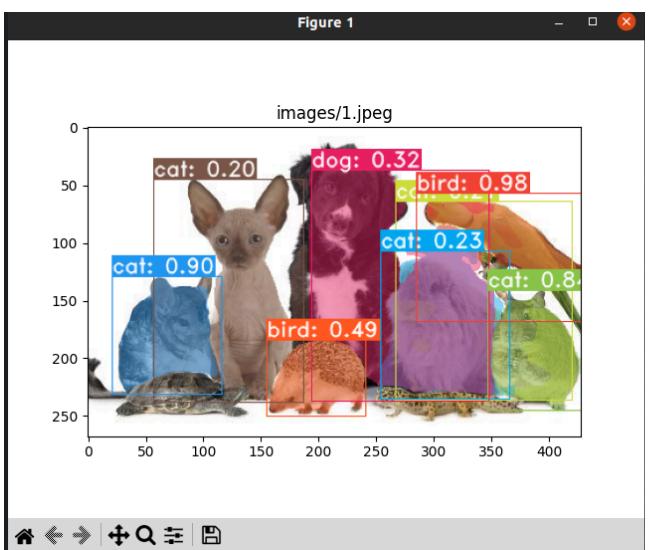
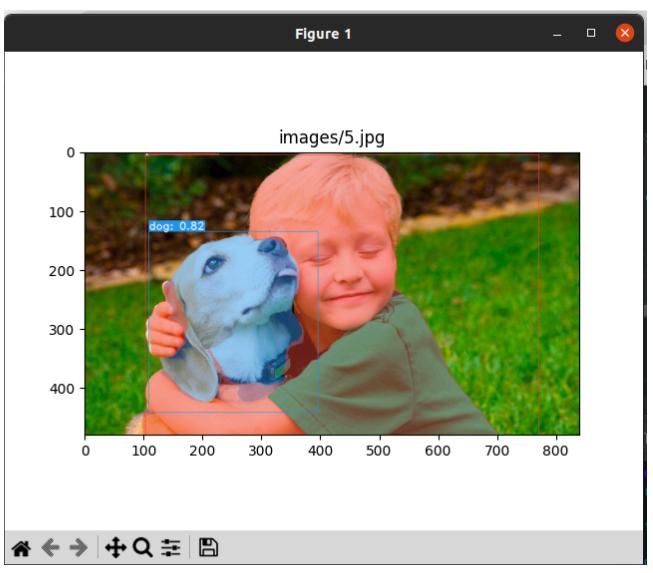


ФОТО 5



Підсумок

ПЕРЕВАГИ	НЕДОЛКИ
Одразу показує відео	Потребує багато пам'яті та навантаження
	Допис коду на resize
	Не завантажує відео, окремо потрібно дописувати.

Детальніше можна ознайомитись:

<https://github.com/dbolya/yolact>

Fast Segment Anything Model (FastSAM)

FastSAM розроблено для усунення обмежень Segment Anything Model (SAM), важкої моделі Transformer із значними вимогами до обчислювальних ресурсів.

FastSAM розділяє завдання сегментувати що завгодно на два послідовні етапи: сегментація всіх екземплярів і підказковий вибір. На першому етапі використовується YOLOv8-seg для створення масок сегментації всіх екземплярів зображення. На другому етапі він виводить область інтересу, що відповідає підказці. Сегментує усі класи COCO та написані підказки.

GitHub-<https://github.com/CASIA-IVA-Lab/FastSAM>

Встановлення

Встановимо спершу CONDA, опісля нейромережу:

```
cd Downloads/  
wget  
https://repo.anaconda.com/archive/Anaconda3-2023.07-2-Linux-x86\_64.sh  
echo  
"589fb34fe73bc303379abbceba50f3131254e85ce4e7cd819ba4276ba29cad16"  
Anaconda3-2023.07-2-Linux-x86_64.sh | sha256sum --check  
перевірити чи є у кінці файлу(якщо відсутній-додати):  
nano /home/okichi/.bashrc
```

```
#>>> conda initialize >>>  
# !! Contents within this block are managed by 'conda init' !!  
__conda_setup="$('/home/arunkl/anaconda3/bin/conda' 'shell.bash' 'hook' 2>  
/dev/null)"  
if [ $? -eq 0 ]; then  
    eval "$__conda_setup"  
else  
    if [ -f "/home/arunkl/anaconda3/etc/profile.d/conda.sh" ]; then  
        . "/home/arunkl/anaconda3/etc/profile.d/conda.sh"  
    else  
        export PATH="/home/arunkl/anaconda3/bin:$PATH"  
    fi  
fi  
unset __conda_setup  
# <<< conda initialize <<<  
  
export PATH="/home/okichi/anaconda3/bin:$PATH"  
conda --version
```

```
conda create -n FastSAM python=3.9
conda activate FastSAM
cd FastSAM
pip3 install -r requirements.txt
pip3 install git+https://github.com/openai/CLIP.git
```

```
okichi@okichi-System-Product-Name:/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/FastSAM$ python3 Inference.py --model_path ./weights/FastSAM-x.pt --img_path ./image/1.jpg
```

Якщо виникає така помилка:from ultralytics.yolo.cfg import get_cfg
ModuleNotFoundError: No module named 'ultralytics'. Тоді:
pip3 install ultralytics

У файлі

/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/FastSAM/fastsam/model.py змінюємо шлях до модулів, аби виглядало таким чином (після ultralytics видаляємо yolo, але варто самостійно звірити правильність шляху).

```
from ultralytics.cfg import get_cfg
from ultralytics.engine.exporter import Exporter
from ultralytics.models.yolo.model import YOLO
from ultralytics.utils import DEFAULT_CFG, LOGGER, ROOT, is_git_dir
from ultralytics.utils.checks import check_imgs
from ultralytics.utils.torch_utils import model_info, smart_inference_mode
from .predict import FastSAMPredictor
```

У решти файлів також потрібно переглянути на справність шляхи до модулів.

Те саме робимо й з іншими файлами у яких виникають помилки (приклад: файл -

/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/FastSAM/fastsam/predict.py
змінені строки :

```
from ultralytics.engine.results import Results
from ultralytics.utils import DEFAULT_CFG, ops
from ultralytics.models.yolo.detect.predict import DetectionPredictor
from .utils import bbox_iou
```

Якщо виникає така помилка:qt.qpa.plugin: Could not load the Qt platform plugin "xcb" in

"/home/okichi/anaconda3/lib/python3.11/site-packages/cv2/qt/plugins" even though it was found.

This application failed to start because no Qt platform plugin could be initialized. Reinstalling the application may fix this problem. Тоді:

sudo apt-get update

sudo apt-get install xcb

pip3 install PyQt5

Тестування

З текстовою підказкою:

```
python3 Inference.py --model_path ./weights/FastSAM-x.pt --img_path  
./image/7.jpg --output ./runs/result/exp1/9 --text_prompt "children"
```

Усі класи YOLO:

```
python3 Inference.py --model_path ./weights/FastSAM-x.pt --img_path  
./image/7.jpg --output ./runs/result/exp1/9
```

Назва	Усі класи YOLO	Текстовая підказка	Тестування з текстовою підказкою
1 зобра ження		cat . dog . rabbit . chinchilla . turtle . hedgehog . lizard . hamster . parrot	
2 зобра ження		children . dog . dog nose . grass	

Можлива робота у реальному часі.

Підсумок

ПЕРЕВАГИ	НЕДОЛКИ
Значне зниження вимог до обчислень і ресурсів без шкоди для якості продуктивності.	Не дуже висока якість.
На основі YOLOv8-seg	Робота з текстовою підказкою є незрозуміла, адже виділяє лише задній фон, незалежно від тексту. Без неї(тобто класи YOLO) виділяють усе, наче це сематична мережа(ні).
Можливість надавати підказказки.	Сенс з цієї нейромережі, якщо працює на сегментаційних моделі YOLO.

Раджу для ознайомлення:

<https://blog.roboflow.com/how-to-use-fastsam/>

DETIC

Detic, представлений Facebook Research і опублікований у січні 2022 року, є моделлю сегментації, спеціально розробленою для виявлення об'єктів. Detic має здатність ідентифікувати 21 000 класів об'єктів з високою точністю, включно з об'єктами, які раніше було важко виявити. Що відрізняє Detic від інших, так це його унікальна особливість — не потребує перенавчання, що робить модель ефективним рішенням, що економить час.

GitHub - <https://github.com/autodistill/autodistill-detic>

Встановлення

Почнемо з detectron2, опісля нейромережу :

<https://github.com/facebookresearch/Detic/blob/main/docs/INSTALL.md>

покроково:

```
conda create --name detic python=3.8 -y
```

```
conda activate detic
```

```
conda install pytorch torchvision torchaudio cudatoolkit=11.1 -c pytorch-lts -c nvidia
```

```
git clone https://github.com/facebookresearch/Detic.git --recurse-submodules
cd Detic
pip3 install -r requirements.txt
cd ..
```

```
git clone https://github.com/facebookresearch/detectron2.git
python3 -m pip install -e detectron2
cd detectron2
pip3 install -e .
```

Якщо виникає така помилка:ModuleNotFoundError: No module named 'mss'. Тоді:

```
pip3 install mss
```

Тестування

Посилання на моделі:

https://github.com/facebookresearch/Detic/blob/main/docs/MODEL_ZOO.md

Запуск фото:

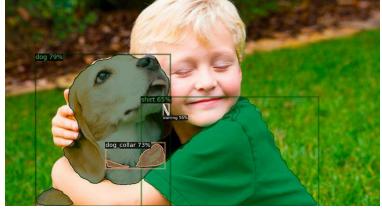
```
python3 demo.py --config-file  
configs/Detic_LCOCOI21k_CLIP_SwinB_896b32_4x_ft4x_max-size.yaml  
--input images/1.jpeg --output runs/exp1/out.jpg --vocabulary lvis --opts  
MODEL.WEIGHTS models/BoxSup-C2_Lbase_CLIP_R5021k_640b64_4x.pth
```

```
python3 demo.py --config-file  
configs/Detic_LCOCOI21k_CLIP_SwinB_896b32_4x_ft4x_max-size.yaml  
--input images/11.jpg --output 12.jpg --vocabulary lvis --opts  
MODEL.WEIGHTS  
models/Detic_LCOCOI21k_CLIP_SwinB_896b32_4x_ft4x_max-size.pth
```

```
okichi@okichi-System-Product-Name:/media/okichi/3e36698e-36b3-4a6c-89a0  
-2167dc453786/project/segment/detic+detecron$ python3 demo.py --config-file  
configs/Detic_LCOCOI21k_CLIP_SwinB_896b32_4x_ft4x_max-size.yaml  
--input images/5.jpg --output runs/exp3/5image/5.5_5.jpg --vocabulary lvis  
--opts MODEL.WEIGHTS models/5.5.pth
```

Результат запуску моделей на фото:

модель	результат (1)	результат (5)
Відкритий словник LVIS (1)		
models/BoxSup-C2_Lba se_CLIP_R5021k_640b 64_4x.pth		

<u>Detic_C2_IN-L_R50_640_4x</u>		
<u>Detic_C2_CCimg_R50640_4x</u>		
<u>Detic_C2_CCcapimg_R50_640_4x</u>		
<u>Box-Supervised_C2_SwinB_inB_896_4x</u>		
<u>Detic_C2_IN-L_SwinB_896_4x</u>		
Стандартний LVIS (2)		
<u>Box-Supervised_C2_R50_640_4x</u>		

Detic_C2_R50_640_4x



Box-Supervised C2 Sw inB 896 4x



Detic_C2_SwinB_896_4x



Box-Supervised_Defor mDETR_R50_4x



Detic_DeformDETR_R50_4x



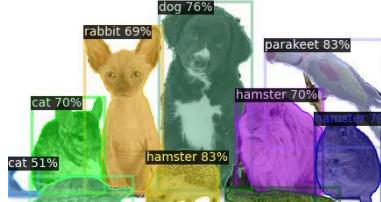
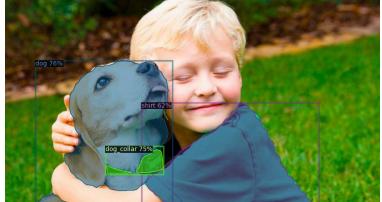
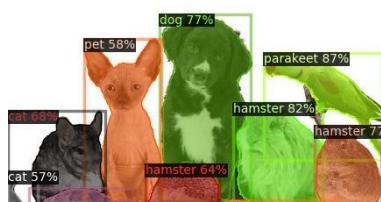
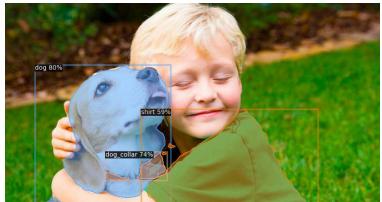
Відкритий словник COCO (3)

BoxSup_CLIP_R50_1x



<u>Detic_CLIP_R50_1x_image</u>		
<u>Detic_CLIP_R50_1x_caption</u>		
<u>Detic_CLIP_R50_1x_caption-image</u>		

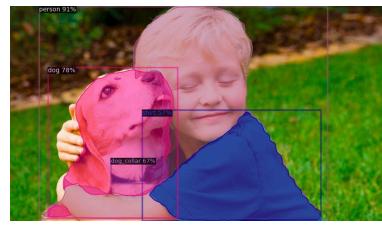
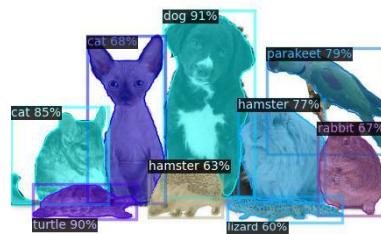
Перехресна оцінка наборів даних (4)

<u>Box-Supervised_C2_SwinB_896_4x</u>		
<u>Detic_C2_SwinB_896_4x</u>		
<u>Detic_C2_SwinB_896_4x_IN-21K</u>		

Box-Supervised_C2_Sw
inB_896_4x+COCO

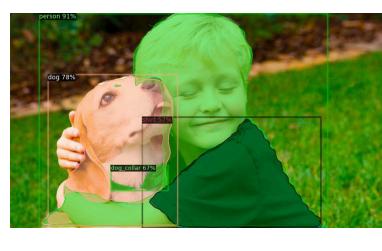


Detic_C2_SwinB_896
4x_IN-21K+COCO

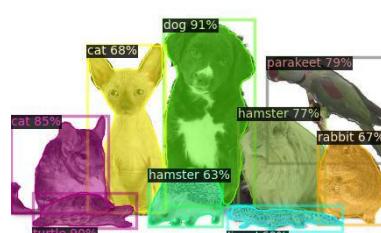


Моделі реального часу (5)

Detic_C2_SwinB_896
4x_IN-21K+COCO
(800x1333, без порогу)



Detic_C2_SwinB_896
4x_IN-21K+COCO



Detic_C2_ConvNeXtT
896_4x_IN-21K+COCO



Detic_C2_R5021k_896
4x_IN-21K+COCO



Detic_C2_R18_896_4x
IN-21K+COCO



Запуск відео:

```
python3 demo.py --config-file  
configs/Detic_LCOCOI21k_CLIP_SwinB_896b32_4x_ft4x_max-size.yaml  
--video-input video/street.MP4 --output runs/exp4/video1.MP4 --vocabulary  
lvis --opts MODEL.WEIGHTS models/5.1.pth
```

Якщо виникає така помилка:[ERROR:0@5.453] global
cap_ffmpeg_impl.hpp:3130 open Could not find encoder for codec_id=27,
error: Encoder not found
[ERROR:0@5.453] global cap_ffmpeg_impl.hpp:3208 open
VIDEOIO/FFMPEG: Failed to initialize VideoWriter
[ERROR:0@5.460] global cap.cpp:643 open VIDEOIO(CV_IMAGES): raised
OpenCV exception:
OpenCV(4.9.0) /io/opencv/modules/videoio/src/cap_images.cpp:430: error:
(-215:Assertion failed) !filename_pattern.empty() in function 'open'. Тоді:

```
# download  
$ git clone --recursive https://github.com/skvark/opencv-python.git  
cd opencv-python  
# set freetype  
$ export CMAKE_ARGS="-DWITH_FREETYPE=ON"  
# enable contrib  
$ export ENABLE_CONTRIB=1  
# compile  
$ pip3 wheel . --verbose
```

Якщо виникає така помилка:Failed to build opencv-contrib-python
ERROR: Failed to build one or more wheels. Тоді:
pip3 uninstall opencv-python opencv-contrib-python
conda install -c conda-forge opencv
python3 demo.py --config-file
configs/Detic_LCOCOI21k_CLIP_SwinB_896b32_4x_ft4x_max-size.yaml
--video-input video/street.MP4 --output runs/exp4/video1.MP4 --vocabulary
lvis --opts MODEL.WEIGHTS models/5.1.pth

```
sudo apt-get install yasm libvpx. libx264.
```

Якщо виникає така помилка: raise AssertionError("Torch not compiled with CUDA enabled")

AssertionError: Torch not compiled with CUDA enabled. Тоді:
conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
conda install cudatoolkit

Завершення роботи на цій помилці.

Підсумок	
ПЕРЕВАГИ	НЕДОЛІКИ
Легкість запуску.	За одним конфігом запускається та працює лише декілька моделей.
Висока точність працюючих ваг.	Можливий запуск лише по одному зображеню.
	Помилки при запуску тестування відео.

OneFormer

OneFormer — це нова модель сегментації, яка в 5 разів отримала значки найсучаснішого рівня від Papers with Code. Він випередив колишні рішення SOTA — MaskFormer і Mask2Former, і тепер займає перше місце в екземплярній, семантичній і панорамній сегментації. OneFormer заснований на трансформаторах і створений за допомогою Detectron2.

GitHub - <https://github.com/SHI-Labs/OneFormer>

Встановлення

Ми використовуємо середовище з такими специфікаціями, пакетами та залежностями:

Ubuntu 20.04.3 LTS Python 3.8.13 conda 4.12.0 PyTorch v1.10.1 Torchvision v0.11.2 Detectron2 v0.6 NATTEN v0.14.4

```
conda create --name oneforner python=3.8 -y
conda activate oneforner
git clone https://github.com/SHI-Labs/OneFormer.git
cd OneFormer
# Install Pytorch
conda install pytorch==1.10.1 torchvision==0.11.2 cudatoolkit=11.3 -c pytorch
-c conda-forge
# Install opencv (required for running the demo)
pip3 install -U opencv-python
# Install detectron2
python3 tools/setup_detectron2.py
# Install other dependencies
pip3 install git+https://github.com/cocodataset/panopticapi.git
pip3 install git+https://github.com/mcordts/cityscapesScripts.git
pip3 install -r requirements.txt
# Setup wandb
pip3 install wandb
wandb login
```

Реєструється за посиланням, яке буде написане в терміналі.
conda install pytorch-gpu=1.3.1 torchvision cudatoolkit=10.0 -c pytorch
export CUDA_HOME=/usr/local/cuda-X.X
Де ви замінюєте XX першими двома цифрами номера вашої версії (можна дізнатися, наприклад, через nvcc --version).

```
#export CUDA_HOME=/usr/local/cuda-12.3 - У МОСМУ ВИПАДКУ  
cd onefomer/modeling/pixel_decoder/ops  
sh make.sh
```

Якщо виникає така помилка:NotImplementedError: No CUDA runtime is found. Please set FORCE_CUDA=1 or test it by running torch.cuda.is_available(). Тоді:
conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch
export FORCE_CUDA="1"

Якщо виникає така помилка:TypeError: expected string or bytes-like object, got 'NoneType'. Тоді:
python3 setup.py install
conda install pytorch::pytorch
sudo apt update
sudo apt autoremove nvidia* --purge
ubuntu-drivers devices
sudo ubuntu-drivers autoinstall
sudo apt install nvidia-driver-525
reboot
sudo apt update

```
sudo apt-get install ./cudnn-local-repo-ubuntu2004-8.9.7.29_1.0-1_amd64.deb  
#(сайт з якого брали інструкцію -  
https://gist.github.com/denguir/b21aa66ae7fb1089655dd9de8351a202)
```

Якщо виникає така помилка:W: GPG error:
file:/var/cudnn-local-repo-ubuntu2004-8.9.7.29 InRelease: The following signatures couldn't be verified because the public key is not available:
NO_PUBKEY 06D8134330472A84. Тоді:

```
wget  
https://developer.download.nvidia.com/compute/cuda/12.0.0/local\_installers/cuda\_12.0.0\_525.60.13\_linux.run  
sudo sh cuda_12.0.0_525.60.13_linux.run  
sudo chmod +x cuda_12.0.0_525.60.13_linux.run  
sudo ./cuda_12.0.0_525.60.13_linux.run --toolkit --samples --silent  
echo 'export PATH=/usr/local/cuda/bin:$PATH' | sudo tee /etc/profile.d/cuda.sh  
source /etc/profile
```

Завантажуємо CUDA через сайт:
<https://github.com/sithu31296/CUDA-Install-Guide/tree/498ab48eb9604af860e42e15147915c7176c8744>

<https://gist.github.com/Mahedi-61/2a2f1579d4271717d421065168ce6a73>

```
sudo ubuntu-drivers devices
sudo ubuntu-drivers install
sudo reboot
lspci | grep -i nvidia
sudo apt-get purge nvidia*
sudo apt remove nvidia-*
sudo rm /etc/apt/sources.list.d/cuda*
sudo apt-get autoremove && sudo apt-get autoclean
sudo rm -rf /usr/local/cuda*
sudo apt-get update
sudo apt-get install g++ freeglut3-dev build-essential libx11-dev libxmu-dev
libxi-dev libglu1-mesa libglu1-mesa-dev
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt update
sudo apt install libnvidia-common-470
sudo apt install libnvidia-gl-470
sudo apt install nvidia-driver-470
wget
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86\_64/cuda-ubuntu2004.pin
sudo mv cuda-ubuntu2004.pin /etc/apt/preferences.d/cuda-repository-pin-600
sudo apt-key adv --fetch-keys
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86\_64/3bf863cc.pub
```

Якщо виникає така помилка:gpg: key A4B469963BF863CC:
"cudatools <cudatools@nvidia.com>" not changed. Тоді:

```
sudo add-apt-repository "deb
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86\_64//"
```

Якщо виникає така помилка:N: See apt-secure(8) manpage for
repository creation and user configuration details. Тоді:

```
wget
https://developer.download.nvidia.com/compute/cuda/repos/wsl-ubuntu/x86\_64/cuda-keyring\_1.0-1\_all.deb
sudo dpkg -i cuda-keyring_1.0-1_all.deb
sudo apt-get install cuda
sudo apt-get purge nvidia*
sudo apt-get purge libnvidia*
sudo apt-get purge cuda-*
```

```
sudo apt-get autoremove
sudo apt-get autoclean
sudo apt-get install cuda
sudo -s
source ~/.bashrc
exit #щоб повернутись в нормальній режим
sudo ldconfig
CUDNN_TAR_FILE="cudnn-linux-x86_64-8.9.7.29_cuda11-archive.tar.xz"
wget
https://developer.nvidia.com/downloads/compute/cudnn/secure/8.9.7/local_insta
llers/11.x/cudnn-linux-x86_64-8.9.7.29_cuda11-archive.tar.xz tar -xvf
${CUDNN_TAR_FILE}
```

На помилці: xz: (stdin): File format not recognized - завершуємо
працювати.

Завершення роботи.

SAM-CLIP

SAM-CLIP використовує модель Segment Anything Model для ідентифікації об'єктів на зображенні та призначення міток кожному зображенню. Потім CLIP використовується для пошуку масок, пов'язаних із заданим запитом. Сегментує усі класи.

GitHub- <https://github.com/maxi-w/CLIP-SAM>

Встановлення

```
git clone https://github.com/autodistill/autodistill-sam-clip.git
pip3 install autodistill autodistill-grounded-sam autodistill-clip
pip3 install autodistill-sam-clip
git clone https://github.com/autodistill/autodistill.git
# з'єднати файли в одну папку
pip3 install autodistill autodistill-detr autodistill-sam-clip supervision
pip3 install opencv-python pycocotools matplotlib onnxruntime onnx
pip3 install open_clip_torch
mkdir checkpoints
cd checkpoints
wget -c https://dl.fbaipublicfiles.com/segmentAnything/sam\_vit\_h\_4b8939.pth
```

Створюємо додатковий файл у папку autodistill-sam-clip:

```
from autodistill_sam_clip import SAMCLIP
from autodistill.detection import CaptionOntology

# define an ontology to map class names to our CLIP prompt
# the ontology dictionary has the format {caption: class}
# where caption is the prompt sent to the base model, and class is the label that
# will
# be saved for that caption in the generated annotations
# then, load the model
base_model = SAMCLIP(ontology=CaptionOntology({"shipping container": "container"}))

# label all images in a folder called `context_images`
base_model.label("./context_images", extension=".jpeg")
```

Тестування

python3 (назва вашого файлу).py

Якщо виникає така помилка: ModuleNotFoundError: No module named 'roboflow'. Тоді:
pip3 install roboflow

Виникає така помилка:

torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 60.00 MiB. GPU 0 has a total capacity of 3.81 GiB of which 32.44 MiB is free. Including non-PyTorch memory, this process has 3.32 GiB memory in use. Of the allocated memory 3.10 GiB is allocated by PyTorch, and 154.61 MiB is reserved by PyTorch but unallocated. If reserved but unallocated memory is large try setting max_split_size_mb to avoid fragmentation. See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF

Роботу припинено через брак можливостей для запуску.

Підсумок	
ПЕРЕВАГИ	НЕДОЛІКИ
	Мало документації.
	Займає багато ресурсів.

Grounded SAM

Grounded SAM використовує модель Segment Anything Model для ідентифікації об'єктів на зображенні та призначення міток кожному зображеню. Сегментація доступна на всі прописані класи.

GitHub - <https://github.com/IDEA-Research/Grounded-Segment-Anything>

Встановлення

```
pip3 install autodistill autodistill-detr autodistill-grounded-sam supervision
conda install pytorch torchvision torchaudio pytorch-cuda=12.1 -c pytorch -c
nvidia
export AM_I_DOCKER=False
export BUILD_WITH_CUDA=True
nvcc --version
export CUDA_HOME=/usr/local/cuda-12.3
git clone https://github.com/IDEA-Research/Grounded-Segment-Anything.git
cd Grounded-Segment-Anything/
python3 -m pip install -e segment_anything
python3 -m pip install -e GroundingDINO
pip3 install --upgrade diffusers[torch]
git submodule update --init --recursive
cd grounded-sam-osx && bash install.sh
cd ..
git clone https://github.com/xinyu1205/recognize-anything.git
cd recognize-anything
pip3 install -r requirements.txt
```

Якщо виникає помилка. Тоді:

```
pip3 install timm==0.4.12
pip3 install transformers==4.25.1
pip install fairscale
pip3 install pycocoevalcap
pip3 install torch torchvision torchaudio
pip3 install torchvision
pip3 install pillow
python3 -m pip install scipy
pip3 install git+https://github.com/openai/CLIP.git
pip3 install opencv-python pycocotools matplotlib onnxruntime onnx ipykernel

cd ..
cd Grounded-Segment-Anything
```

```
wget  
https://github.com/IDEA-Research/GroundingDINO/releases/download/v0.1.0-alpha/groundingdino\_swint\_ogc.pth
```

Тестування

```
okichi@okichi-System-Product-Name:/media/okichi/3e36698e-36b3-4a6c-89a0-  
-2167dc453786/project/segment/Grounded-Segment-Anything$ python3  
grounding_dino_demo.py
```

```
#редагування і тд робити у самому файлі  
Сегментація всього:  
cd Grounded-Segment-Anything
```

```
wget https://dl.fbaipublicfiles.com/segment_anything/sam_vit_h_4b8939.pth  
wget  
https://github.com/IDEA-Research/GroundingDINO/releases/download/v0.1.0-alpha/groundingdino\_swint\_ogc.pth
```

Якщо виникає така помилка: NameError: name '_C' is not defined. Тоді:
pip3 install Cython

```
python3 grounded_sam_demo.py --config  
GroundingDINO/groundingdino/config/GroundingDINO_SwinT_OGC.py  
--grounded_checkpoint groundingdino_swint_ogc.pth --sam_checkpoint  
sam_vit_h_4b8939.pth --input_image assets/demo1.jpg --output_dir  
"outputs" --box_threshold 0.3 --text_threshold 0.25 --text_prompt "bear"
```

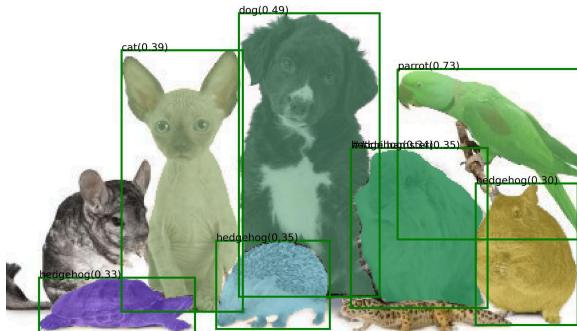
Додатковий сайт для ознайомлення:

<https://github.com/IDEA-Research/Grounded-Segment-Anything?tab=readme-o-v-file#installation>

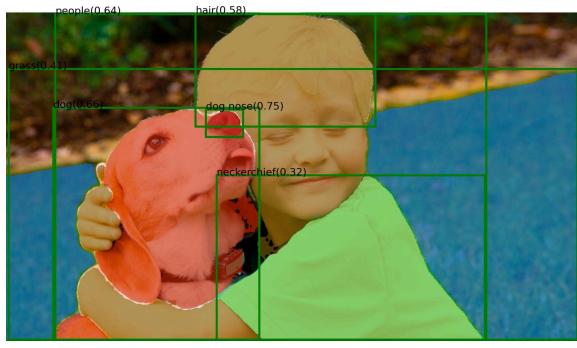
Результат

Команда пошуку	Витяг
----------------	-------

--text_prompt "cat . dog . rabbit .
chinchilla . turtle . hedgehog . lizard .
hamster . parrot"



--text_prompt "dog . people . dog
nose . neckerchief . hair . grass"



Підсумок

ПЕРЕВАГИ	НЕДОЛКИ
Можна текстов вказати потрібні класи.	Якщо потрібно сегментувати багато класів потрібно прописувати вручну.
	Немає інших ваг.
	Не можна відео сегментувати.

EdgeSAM

EdgeSAM — це прискорений варіант Segment Anything Model (SAM), оптимізований для ефективного виконання на периферійних пристроях із мінімальним компромісом у продуктивності. Особливість нейромережі у роботі онлайн, де ти самостійно позначаєш, які об'єкти хочеш виділити. Сегментує усі класи.

GitHub - <https://github.com/chongzhou96/EdgeSAM>

Встановлення

```
git clone https://github.com/chongzhou96/EdgeSAM.git && cd EdgeSAM  
git clone https://github.com/chongzhou96/EdgeSAM.git && cd EdgeSAM  
pip3 install -r requirements.txt  
pip3 install -e .  
mkdir weights  
wget -P weights/  
https://huggingface.co/spaces/chongzhou/EdgeSAM/resolve/main/weights/edge\_sam.pth  
wget -P weights/  
https://huggingface.co/spaces/chongzhou/EdgeSAM/resolve/main/weights/edge\_sam\_3x.pth
```

Тестування

python3 web_demo/gradio_app.py

Запускаємо паралельно у пошуку сторінку :http://0.0.0.0:8080

Результат

Мітка	Зображення
Зображення 1	
Позитивна мітка - на носику собаки. Виділено усю собаку, що є хорошим показником.	

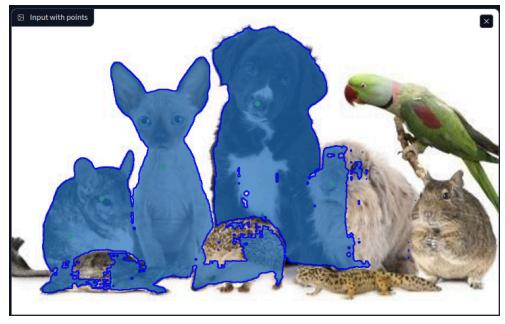
Позитивна мітка - на тілі кота.
Виділено майже усього кота та
паралельно до цього сегментація
пса злетіла.



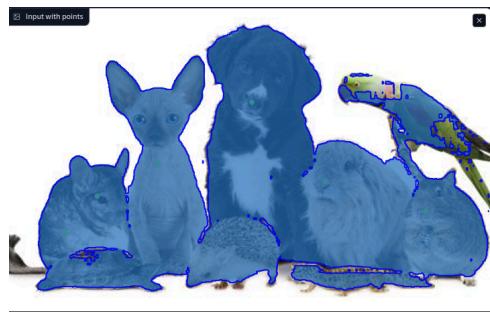
Позитивна мітка - око шиншили.
Виділено шиншилу, кота.
Сегметація на собаці залишилась
лиш на носі.



Позитивна мітка - ніс кролика.
Виділено частково кроля, їжачка та
черепаху, повністю кота та
шиншилу. Знову повністю
засегментувало пса.



Позитивна мітка - ніс хом'яка.
Виділено усіх тварин, але папугу
частково.



Зображення 5

Позитивна мітка - ніс дитини.
Виділено обличчя хлопця.



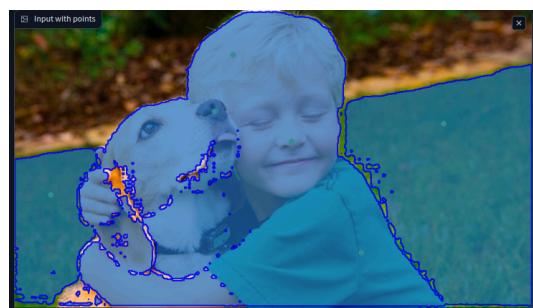
Позитивна мітка - футболька дитини.
Виділено усього хлопчика, але без рук



Позитивна мітка - ніс собаки.
Виділено повністю хлопчика та собаку.



Позитивна мітка - 2 на траві по обох сторонах.
Виділено окремо траву і наче з розділенням пса і хлопчика.



Негативна мітка - поряд, на футболці хлопця.
Зникло виділення пса і хлопчика.



Підсумок

ПЕРЕВАГИ	НЕДОЛКИ
Можливість працювати онлайн(запуск сервера).	Сегментація лише по точках.
Швидка у встановлені та роботі.	Не дуже коректно працює.

Grounded EdgeSAM

Autodistill поєднує EdgeSAM із Grounding DINO, моделлю виявлення об'єктів з нульовим ударом, щоб створити Grounded EdgeSAM. Ця гібридна модель дає змогу ідентифікувати типові об'єкти на зображеннях (наприклад, навантажувач), а потім генерувати маски сегментації. Ці маски позначають на піксельному рівні об'єкти на зображеннях. Сегментує усі класи, але одночасно лише один.

GitHub - <https://github.com/autodistill/autodistill-grounded-edgesam> і <https://github.com/IDEA-Research/Grounded-Segment-Anything/tree/main/EfficientSAM#run-grounded-edge-sam-demo>

Встановлення

Спершу завантажимо попередньо підготовлену контрольну точку Edge-SAM і саму нейромережу:

```
git clone https://github.com/chongzhou96/EdgeSAM.git && cd EdgeSAM  
pip3 install -r requirements.txt  
pip3 install --upgrade pip  
pip3 install -e .  
mkdir weights  
wget -P weights/  
https://huggingface.co/spaces/chongzhou/EdgeSAM/resolve/main/weights/edgesam.pth  
wget -P weights/  
https://huggingface.co/spaces/chongzhou/EdgeSAM/resolve/main/weights/edgesam\_3x.pth  
git clone https://github.com/IDEA-Research/Grounded-Segment-Anything.git  
# //// Витягуємо все з клону у папку де він знаходиться.  
cd Grounded-Segment-Anything/  
export AM_I_DOCKER=False  
export BUILD_WITH_CUDA=True  
nvcc --version  
# Версію вписуємо в наступному рядку.  
export CUDA_HOME=/usr/local/cuda-12.3/  
python3 -m pip install -e segment_anything  
python3 -m pip install -e GroundingDINO  
pip3 install --upgrade diffusers[torch]  
git submodule update --init --recursive
```

```

cd grounded-sam-osx && bash install.sh
git clone https://github.com/xinyu1205/recognize-anything.git
pip3 install -r ./recognize-anything/requirements.txt
#pip3 install -e ./recognize-anything/
pip3 install opencv-python pycocotools matplotlib onnxruntime onnx ipykernel
cd .. # має бути папка Grounded-Segment-Anything
wget
https://github.com/IDEA-Research/GroundingDINO/releases/download/v0.1.0-
alpha/groundingdino_swint_ogc.pth
cd EfficientSAM
wget -P
https://huggingface.co/spaces/chongzhou/EdgeSAM/resolve/main/weights/edge\_sam\_3x.pth
# або перемістити ваги edge_sam_3x.pth з папки
#segment/EdgeSAM/weights/ у папку
#EdgeSAM/Grounded-Segment-Anything/EfficientSAM

```

Перевірка роботи:

python3 EfficientSAM/grounded_edge_sam.py
 результат буде збережено як ./gronded_edge_sam_anontated_image.jpg

Тестування

Аби почати тестування на власних даних, потрібно деякі зміни у файл EfficientSAM/grounded_edge_sam.py. У строках 29-34 потрібно вказати потрібні нам параметри: перша строка- посилання на зображення, друга - потрібний нам клас сегментації.

```

SOURCE_IMAGE_PATH =
"./EfficientSAM/LightHQSAM/example_light_hqsam.png"
CLASSES = ["bench"]
BOX_THRESHOLD = 0.25
TEXT_THRESHOLD = 0.25
NMS_THRESHOLD = 0.8

```

57 рядок - збереження опротестованого зображення.

```
cv2.imwrite("EfficientSAM/LightHQSAM/groundingdino_annotated_image.jpg", annotated_frame)
```

Папки не змінюємо!

Усі робочі папки(початкові та кінцеві фото) закинути в EfficientSAM/LightHQSAM/

Якщо виникає помилка з []. То перевірити чи написана правильно команда:

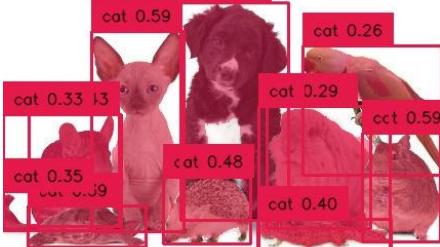
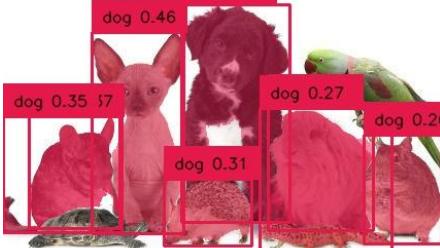
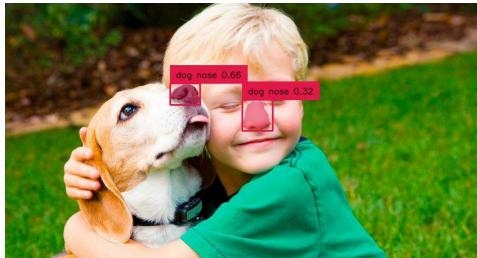
CLASSES = ["parrot"]

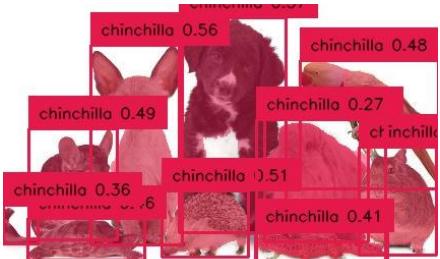
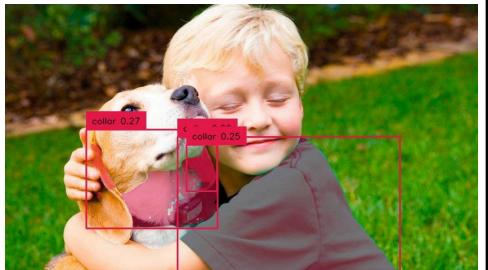
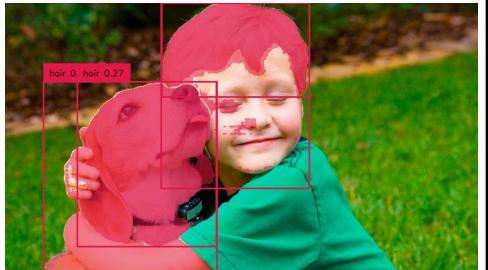
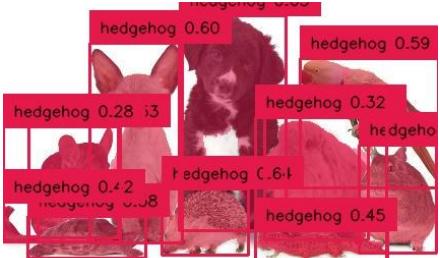
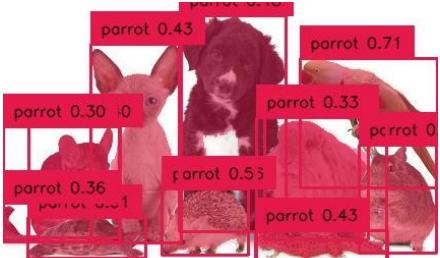
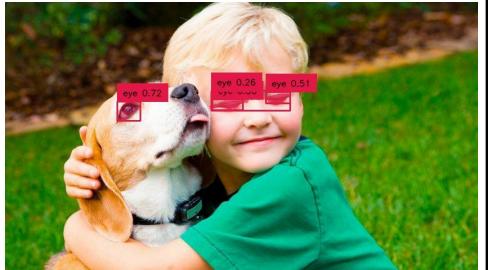
НЕ МАЄ бути пропуску між словом та лапками.

Команда запуску:

```
okichi@okichi-System-Product-Name:/media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/EdgeSAM/Grounded-Segment-Anything$ python3 /media/okichi/3e36698e-36b3-4a6c-89a0-2167dc453786/project/segment/EdgeSAM/Grounded-Segment-Anything/EfficientSAM/grounded_edge_sam.py
```

Результат

Вказані класи	Зображення (1)	Вказані класи	Зображення (5)
cat		people	
dog		dog	
rabbit		dog nose	

chinchilla		collar	
turtle		grass	
lizard		hair	
hedgehog		shirt	
parrot		eye	

hamster	немає такого класу	tongue	
---------	--------------------	--------	---

Підсумок	
ПЕРЕВАГИ	НЕДОЛІКИ
Сегментує усі класи.	Лиш один клас та фото за запуск.
	Не сегментує відео.
	Вимога до розташування папок.
	Некоректна робота.