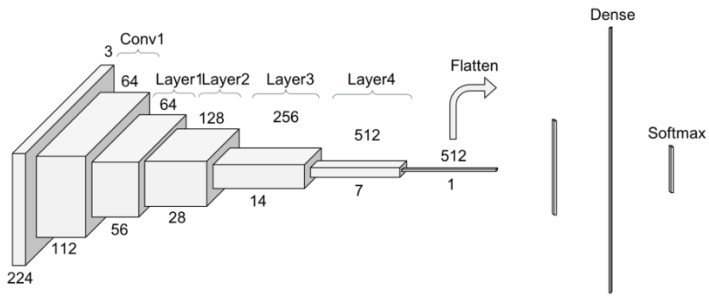
Лабораторна робота № 3. Використання Label Studio для анотації, класифікація та детекція об’єктів на зображеннях з використанням ResNet та Yolo

Мета: сформулювати практичні навички у сфері анотації зображень, використання переднавчених конволюційних нейронних мереж (CNN) для класифікації зображень, а також основ роботи з YOLOv5 й YOLOv8 для детекції об’єктів. Учасники лабораторної роботи навчаться організовувати та підготовлювати датасети, застосовувати інструменти для анотування зображень, проводити тести та порівнювати різноманітні архітектури нейронних мереж. Також вони отримають досвід у розробці власних моделей для ефективного розв’язання специфічних задач класифікації та детекції об’єктів.

Теоретичні відомості

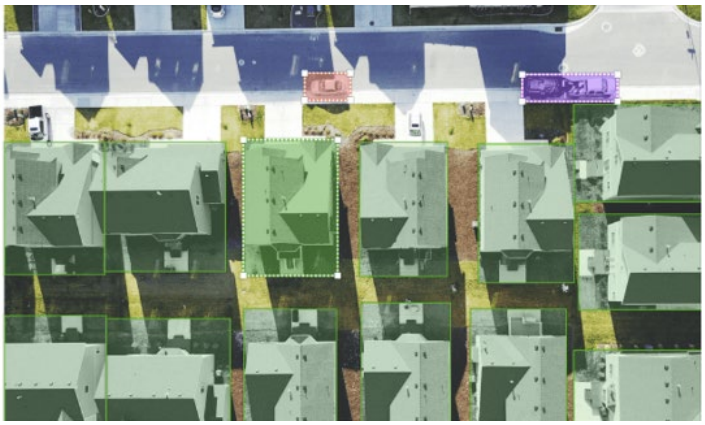
В останні роки область комп’ютерного зору зазнала значних змін завдяки розвитку глибинного навчання та конволюційних нейронних мереж (CNN). CNN стали основою багатьох сучасних систем обробки зображень, здатних виконувати широкий спектр завдань, від класифікації зображень до детекції об’єктів.



Класифікація зображень – це процес визначення, до якого класу належить зображення з заданого набору категорій. Переднавчені моделі, такі як ResNet (з різними варіантами глибини: 18, 34, 50, 101, 152), дозволяють використовувати переваги глибокого навчання без необхідності збирати величезні набори даних та витрачати значні ресурси на навчання моделей з нуля.

Детекція об’єктів зосереджена на визначенні місцезнаходження об’єктів на зображенні та їх класифікації. YOLO (You Only Look Once) є однією з передових архітектур для детекції об’єктів, що дозволяє виконувати детекцію в реальному часі з високою точністю. YOLOv5, використана у цій лабораторній роботі, є потужною ітерацією цієї архітектури, яка вирізняється покращеною

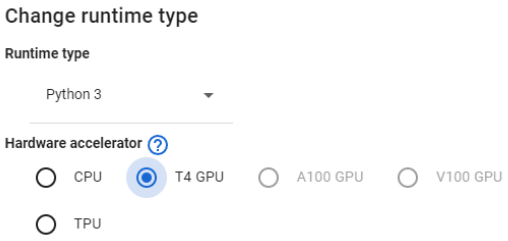
продуктивністю, ефективністю та гнучкістю. Вона забезпечує значні поліпшення у швидкості та точності детекції порівняно з попередніми версіями, роблячи її ідеальним вибором для широкого спектру застосунків від простого виявлення об’єктів до складних систем візуального спостереження. YOLOv8 продовжує вдосконалювати можливості детекції, враховуючи найновіші досягнення у галузі глибинного навчання та комп’ютерного зору.



Анотація зображень є критично важливим етапом підготовки датасетів для тренування моделей глибинного навчання. Вона включає маркування зображень з метою навчання моделі розпізнавати і класифікувати об’єкти. Використання інструментів, таких як Label Studio, спрощує процес анотації, дозволяючи точно вказувати місцезнаходження та класи об’єктів на зображеннях.

Рекомендації до лабораторної роботи

При виконанні лабораторної роботи в Google Colab використовуйте «T4 GPU». Для цього оберіть «Runtime» → «Change runtime type» → «T4 GPU».



Приклад виконання роботи

Завдання

1. Використовуючи запропонований код, зберіть датасет зображень за темою, вказаною у вашому варіанті. Завантажте не менше 30 зображень для кожного класу об’єктів.
2. Використовуючи бібліотеку FastAI, протестуйте різні моделі ResNet (вказані у вашому варіанті) для класифікації зображень. Визначте, яка з моделей демонструє кращу точність, при параметрах навчання вашого датасету.
3. Використайте Label Studio для анотації зібраних зображень. Кожне зображення повинно містити принаймні 1 анотований об’єкт, що вказаний у вашому варіанті.
4. Налаштуйте та навчіть модель YoloV5 на вашому анотованому датасеті для детекції об'єктів. Використовуйте параметри навчання (кількість епох, розмір пакету), вказані у вашому варіанті.
5. Проаналізуйте результати класифікації та детекції. Визначте, які класи об’єктів модель визначає найточніше, та у яких виникають складнощі при розпізнавані.
6. Використайте претреновану модель YoloV8 для детекції довільних об’єктів на фото або відео.

Частина 1					**Частина 2**		
Варіант	**Тема***	**Перша Модель ResNet*****	**Друга Модель ResNet*****	**Кіль-кість епох (fine_tune)****	**Анотації в Label Studio***	**Кількість епох (YoloV5)****	**Розмірність пакету (batch_size)****
T	Забори	18	34	5	Забори, ворота	5	16

Контрольне запитання №1

Вкажіть показник accuracy для останньої епохи навчання першої ResNet.
Відповідь: _____

Контрольне запитання №2

Вкажіть показник accuracy для останньої епохи навчання другої ResNet.
Відповідь: _____

Контрольне запитання №3

Вкажіть кількість проанотованих зображень.

Відповідь: _____

Контрольне запитання №4

Як впливає кількість епох на процес навчання моделі, і які потенційні ризики пов'язані з великою кількістю епох?

Відповідь: _____

_____

_____

Хід роботи

1. Встановлення необхідних бібліотек та модулів

```
import os
from google.colab import files
import socket,warnings
from fastbook import *
from fastdownload import download_url
from fastai.vision.all import *
from fastai.vision.widgets import *
from matplotlib import pyplot as plt
from time import sleep
from google.colab import files
import torch

!pip install fastbook
try:
    socket.setdefaulttimeout(1)
    socket.socket(socket.AF_INET, socket.SOCK_STREAM).connect(('1.1.1.1',
53))
except socket.error as ex: raise Exception("STOP: No internet. Click '>|'
in top right and set 'Internet' switch to on")
iskaggle = os.environ.get('KAGGLE_KERNEL_RUN_TYPE', '')

if iskaggle:
    !pip install -U duckduckgo_search
    !pip install fastai
    !pip install fastbook

!git clone https://github.com/ultralytics/yolov5
%cd yolov5
!pip install -qr requirements.txt
```

```
!pip install pillow==10.1.0
```

2. Пошук зображень за тематикою забори (fence) та ворота (gate).

```
# Завдання №1. Збір датасету зображень
def search_images(term, max_images=30):
    print(f"Searching for '{term}'")
    return L(search_images_ddg(term, max_images=max_images))

dest = 'fence.jpg'
download_url(urls[0], dest, show_progress=False)

im = Image.open(dest)
im.to_thumb(256,256)
```

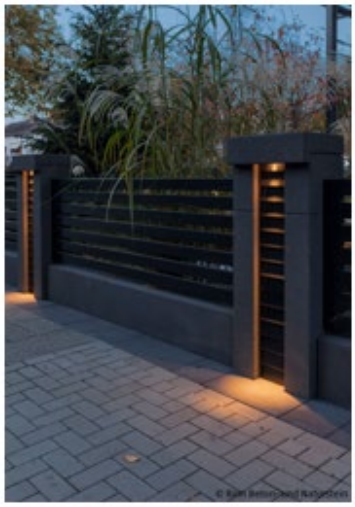


Рисунок 2.1. Екземпляр класу «fence»

```
download_url(search_images('gate photos', max_images=1)[0], 'gate.jpg',
show_progress=False)
Image.open('gate.jpg').to_thumb(256,256)
```

Рисунок 2.2. Екземпляр класу «gate»

2.1 Пошук та завантаження зображень при різному освітлені

```
searches = 'gate','fence'
path = Path('fence_or_not')

for o in searches:
    dest = (path/o)
    dest.mkdir(exist_ok=True, parents=True)
    download_images(dest, urls=search_images(f'{o} photo'))
    sleep(10)  # Pause between searches to avoid over-loading server
    download_images(dest, urls=search_images(f'{o} sun photo'))
    sleep(10)
    download_images(dest, urls=search_images(f'{o} shade photo'))
    sleep(10)
    resize_images(path/o, max_size=400, dest=path/o
```

2.2 Аналіз переліку завантажених зображень

```
print(path)
searches = 'gate','fence'

for imageType in searches:
    print(imageType)
    content = os.listdir(path/imageType)
    for element in content:
        print(element)
```

```
27f53785-ff92-4af2-aba9-cd73542d748c.jpg
62ae10aa-06a5-48c0-8ae9-4a9ba752bcdc.jpg
e3f2348b-cde1-40a5-a31d-2807a92dd170.jpg
e3b433ca-3a28-48df-aeae-8db594ae3187.jpg
6b5ea308-5851-4c3f-94e9-256e2372ef73.jpg
dc0f7491-a76f-4d78-b633-114c5d831f00.jpg
abb321d2-a971-44ba-a277-fc65879802bc.jpg
1fc3d332-4cdc-4a22-b52c-37f5c45260b7.jpg
306865db-6672-4d1d-a0ac-192a284688e8.jpg
927e0429-1d5b-4907-9435-2cf82dad61b4.jpg
08de0e4f-3b65-425a-a181-016845cf55a3.jpg
f1c670b0-067a-40c3-a2ef-f6ca05d48632.jpg
1f820bd1-02ed-4ecd-932d-771edd13d387.jpg
e916e477-906d-44f7-935f-8e45f9f74679.jpg
97463ee7-4eeb-4bd0-8ec1-20b53358bce5.jpg
a31352ba-dd4f-47dd-8b59-da6ba1a110f9.jpg
0a55b0a1-ba4e-4cdf-b74e-052143e57873.jpg
8cf00466-8aee-41cf-b42e-843115daa8aa.jpg
d89f84ac-6839-4be2-81f3-60790e4330b2.jpg
cd37b7ec-a60d-45ea-815a-dba490e84cb3.jpg
c2344974-5499-45f9-aed6-2026ae124d31.jpg
ca122e1a-8ab2-45f1-afaa-482e9d3ea4fc.jpg
68d7b202-1ed0-48a4-9825-a6bd3af54021.jpg
d2800340-6d66-4ff7-9128-8c6894cc5b75.jpg
0319ac99-1ff3-40bb-a592-de0d70e2fbd1.jpg
```

Рисунок 2.3. Частина завантажених зображень за темою

2.3 Аналіз завантажених зображень, пошук нерелевантних екземплярів

```
def list_images(directory):
    return [f for f in os.listdir(directory) if
os.path.isfile(os.path.join(directory, f))]

failed = verify_images(get_image_files(path))
failed.map(Path.unlink)
len(failed)

dls = ImageDataLoaders.from_folder(
    path, valid_pct=0.2, seed=42,
    item_tfms=Resize(224),
    batch_tfms=aug_transforms(mult=2)
)

dls.show_batch(max_n=6)
```

Рисунок 2.4. Аналіз екземплярів класів у датасеті

3. Тестування моделей ResNet18 та ResNet34.

```
# Завдання №2. Тестування різних моделей ResNet
learn = vision_learner(dls, resnet18, metrics=[error_rate, accuracy])
learn.fine_tune(5)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.397259	0.939227	0.363636	0.636364	00:38
epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.925423	0.816139	0.393939	0.606061	00:40
1	0.775136	0.708150	0.333333	0.666667	00:41
2	0.661138	0.669650	0.242424	0.757576	00:40
3	0.627082	0.666702	0.272727	0.727273	00:39
4	0.550512	0.659468	0.272727	0.727273	00:40

Рисунок 2.5. Результат навчання моделі ResNet18

```
learn2 = vision_learner(dls, resnet34, metrics=[error_rate, accuracy])
learn2.fine_tune(5)
```

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	1.390454	1.368886	0.393939	0.606061	00:52
epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.882794	0.907909	0.424242	0.575758	01:14
1	0.807758	0.650826	0.303030	0.696970	01:14
2	0.772568	0.593983	0.333333	0.666667	01:17
3	0.682128	0.587102	0.272727	0.727273	01:17
4	0.621130	0.597385	0.272727	0.727273	01:14

Рисунок 2.6. Результат навчання моделі ResNet34

Таким чином, в однаковому діапазоні epoch при порівнянні моделей ResNet18 та ResNet34 вдалося досягти ідентичного показника точності (accuracy), проте моделі ResNet34 затребувалося на ~50% більше часу.

4. Анотування зображень в Label Studio

```
# Завантажимо Dataset
folder_path = '/content/fence_or_not'
!zip -r /content/fence_or_not.zip "$folder_path"
files.download('/content/fence_or_not.zip')
```

Перейдемо до сторінки з Label Studio та створимо новий інтерфейс клавішою «Create».

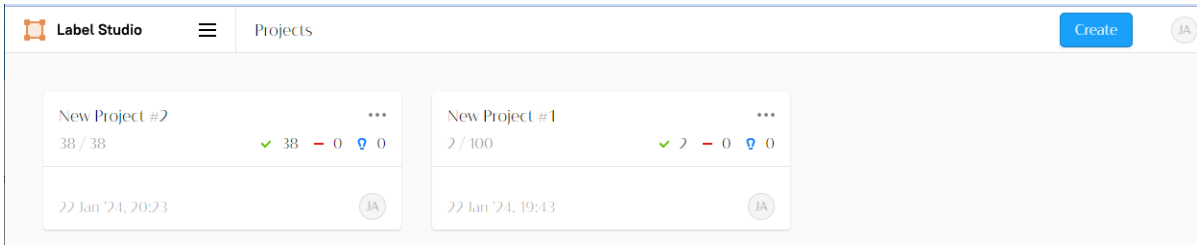


Рисунок 2.7. Загальний інтерфейс стартової сторінки Label Studio

Надамо назву проєкту «FENCE_OR_GATE».

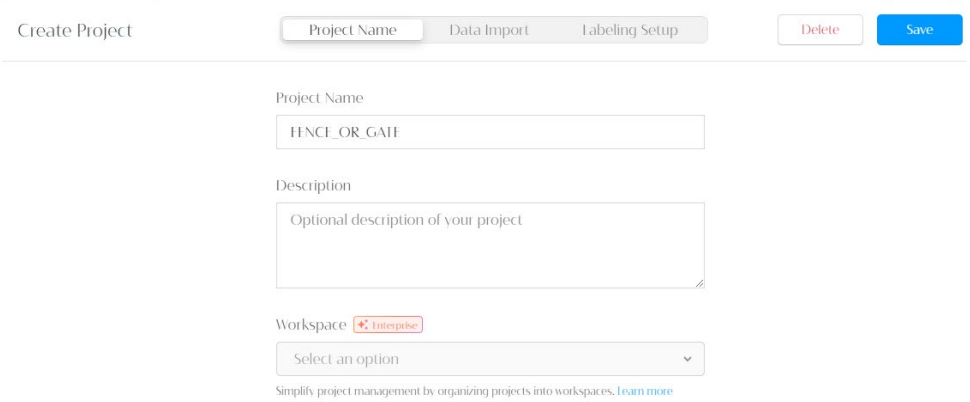


Рисунок 2.8. Встановлення назви проєкту

Перейдемо до завантаження зображень.

Примітка. Завчасно перегляньте завантажені зображення та видаліть такі, що не відповідають вашій темі, мають некоректний формат або пошкоджені, щоб уникнути помилки

⚠ *«is not supported»*

Також завантажуйте зображення партіями по 50 зображень, щоб не отримати помилку

⚠ *«The number of files exceeded settings. DATA_UPLOAD_MAX_NUMBER_FILES».*

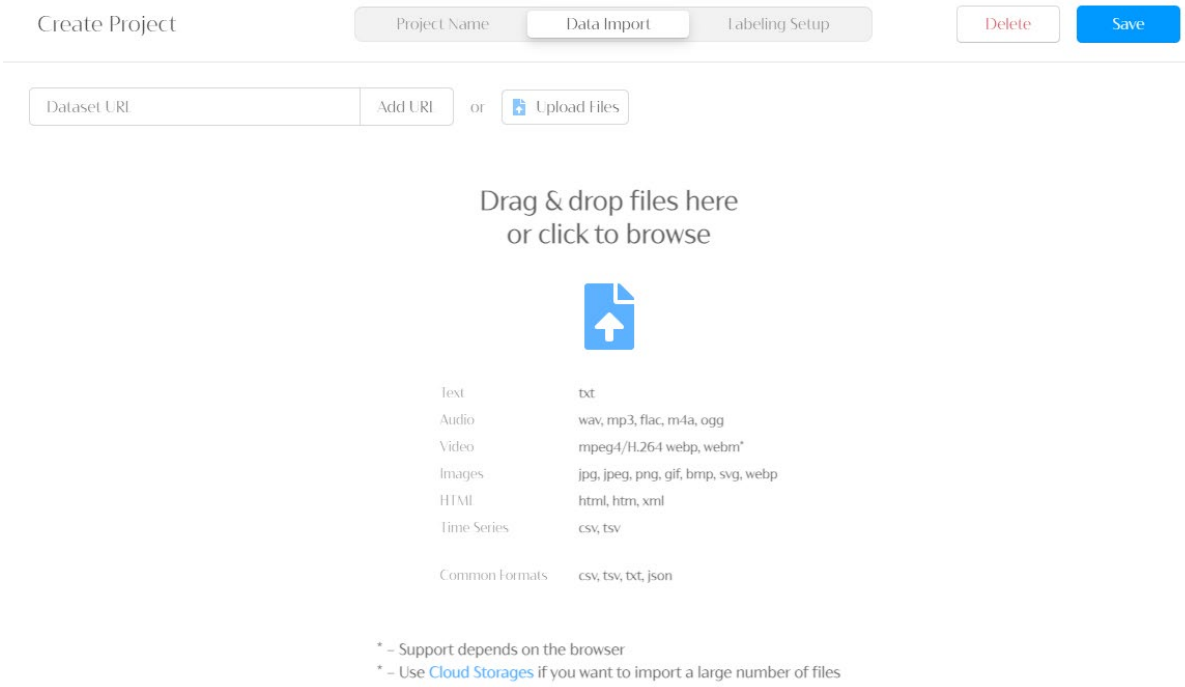


Рисунок 2.9. Вікно завантаження зображень

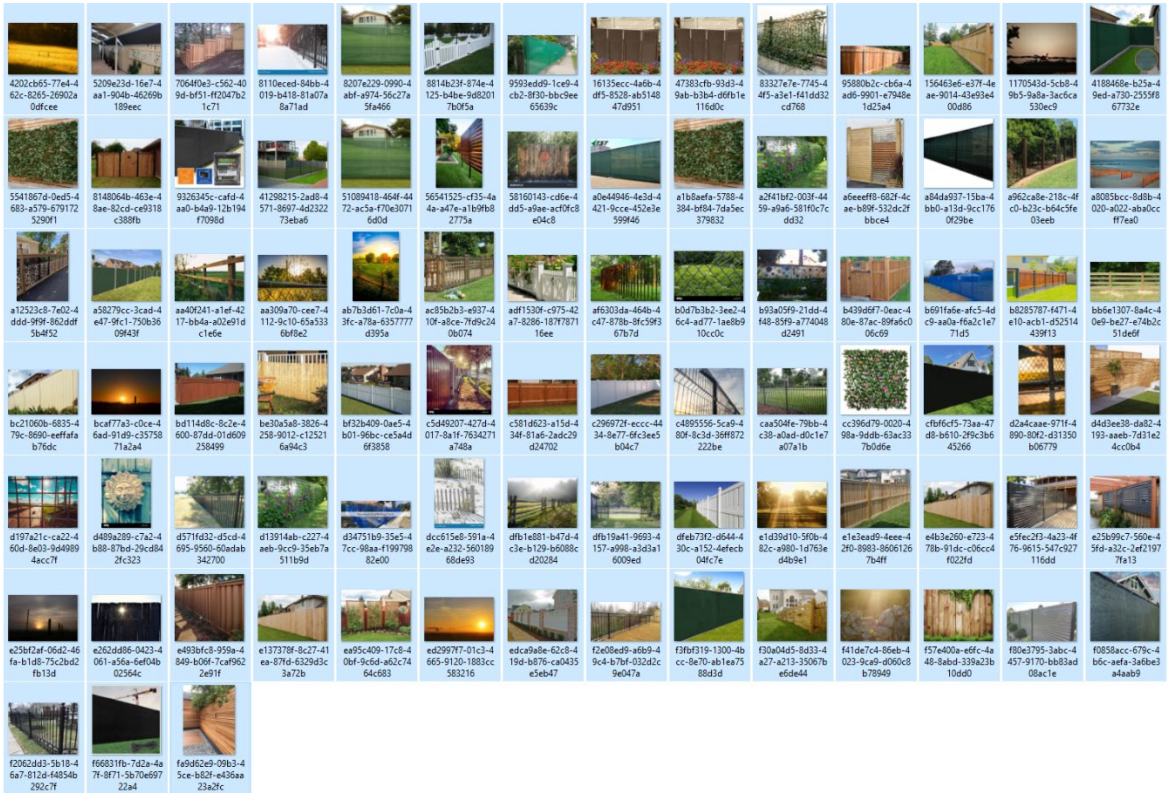


Рисунок 2.10. Вибір зображень заборів «fence»

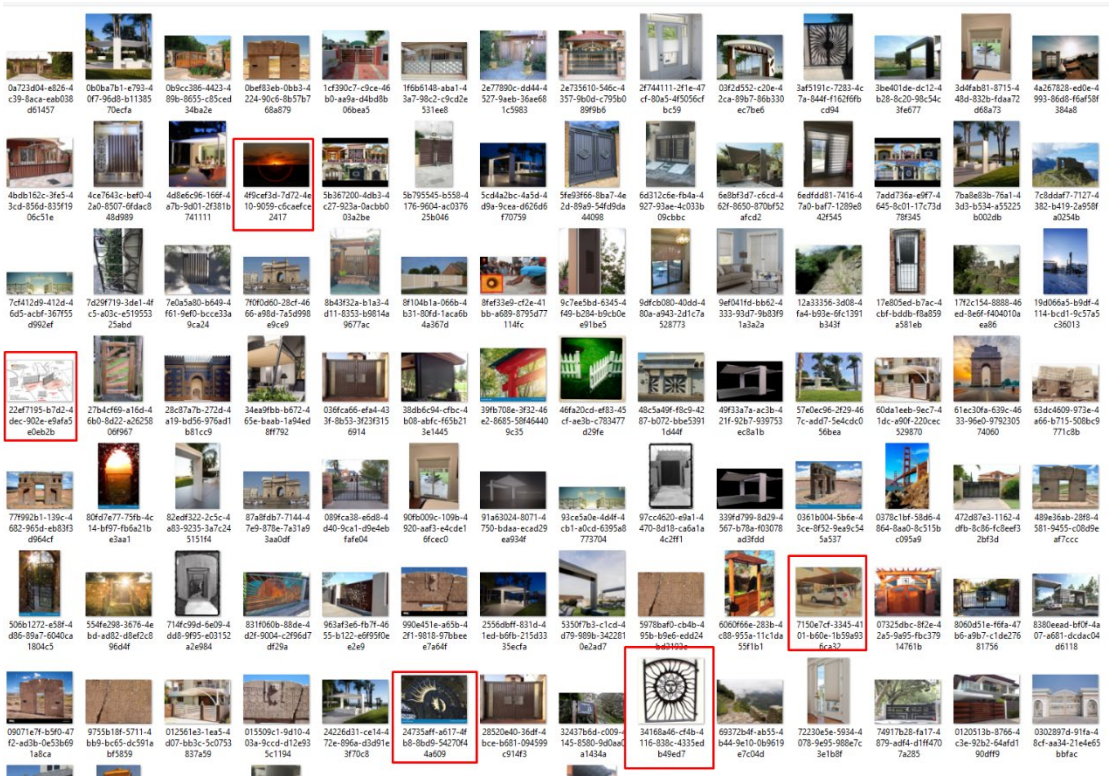


Рисунок 2.11. Вибір зображень воріт «gate», з видаленням невідповідних

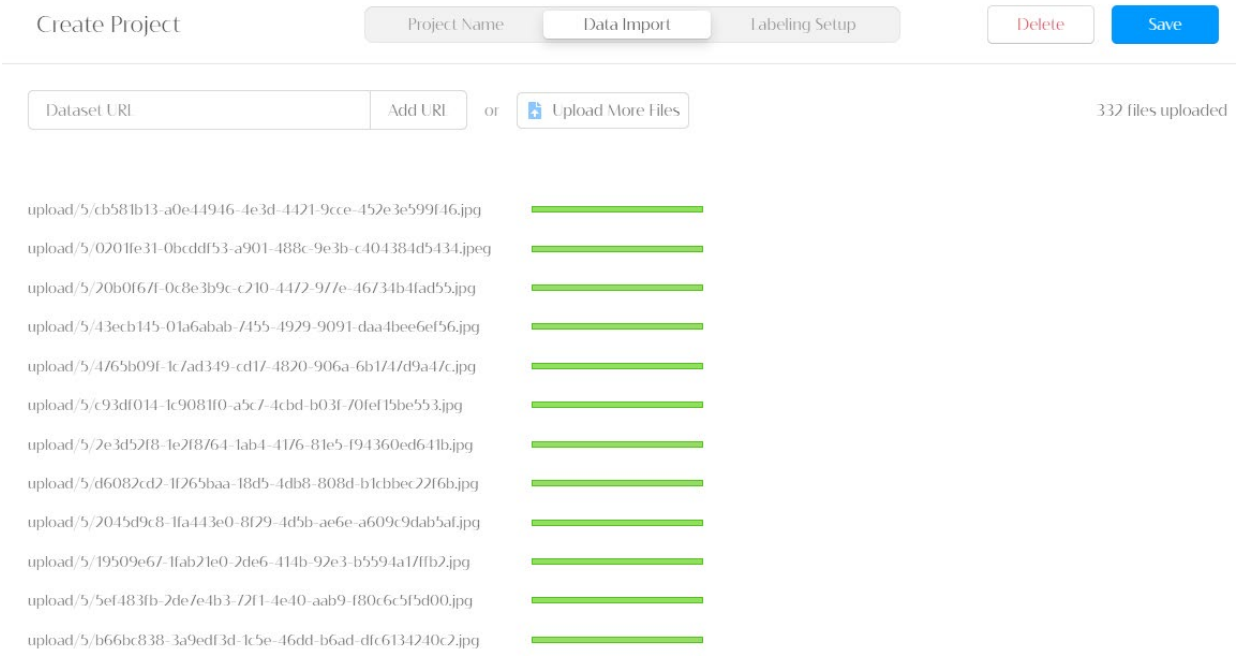


Рисунок 2.12. Завантажені файли зображень у Label Studio

Оберемо задачу «Object Detecting with Bounding Boxes» тобто розпізнавання об’єктів з використанням обмежувальної рамки.

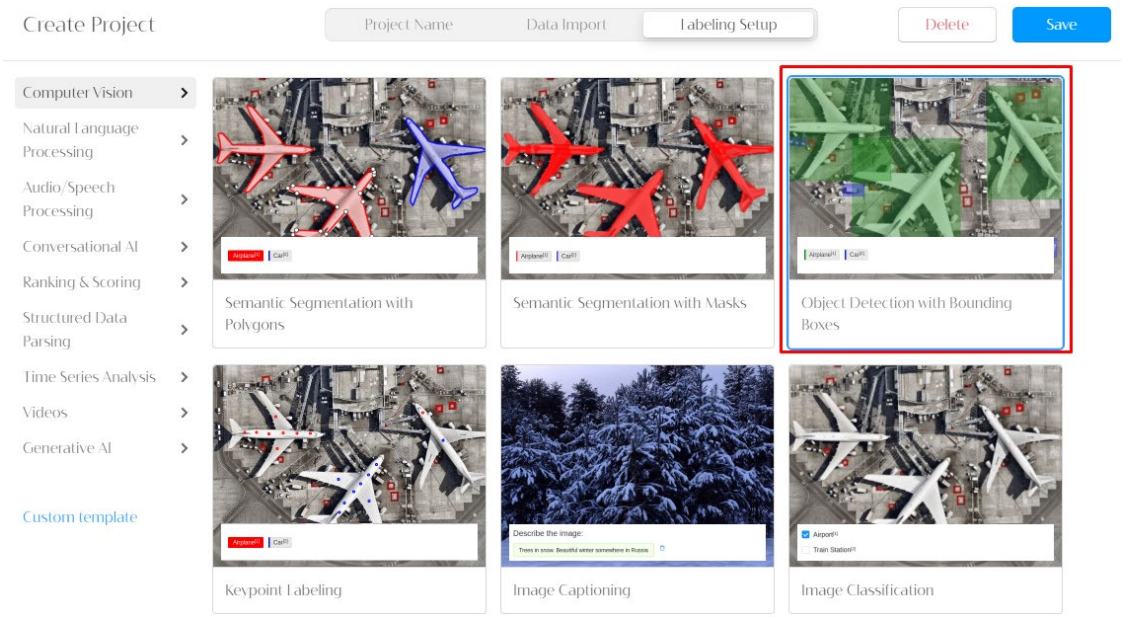


Рисунок 2.13. Завантажені файли зображень у Label Studio

Внесемо назви класів, що будуть розпізнаватися до відповідного поля.

Рисунок 2.14. Внесення класів, які будуть використані для анотування зображень

Рисунок. 2.15. Внесені мітки

Натиснемо на довільне зображень з переліченого списку та виконаємо анотування об’єкта за допомогою 1 для fence та 2 для gate.

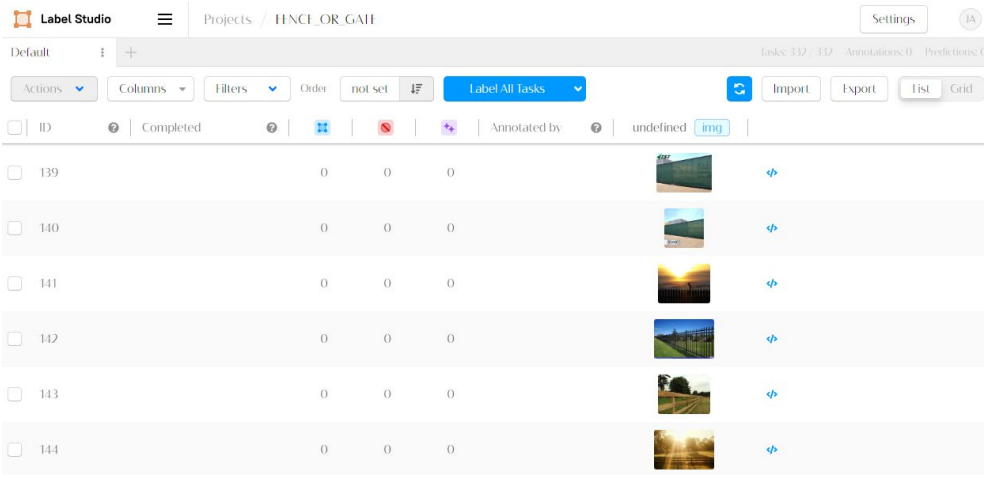


Рисунок 2.16. Загальний вигляд неанотованих зображень у Label Studio

Рисунок 2.17. Анотування забору та воріт одночасно

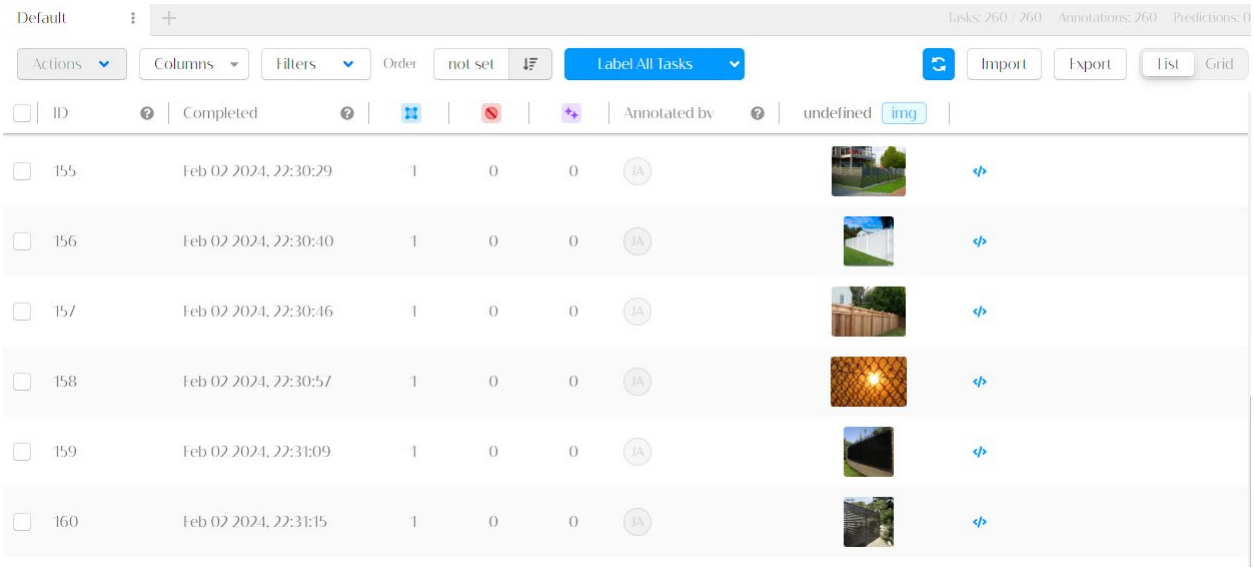


Рисунок 2.18. Проанотовані зображення

Експортуємо зображення та створені мітки для моделей YOLO за допомогою «Export» → «YOLO».

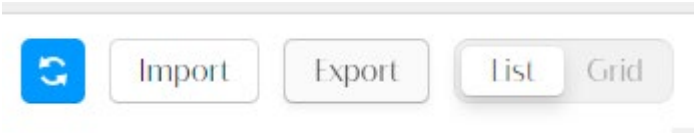


Рисунок 2.19. Експорт анотованих зображень

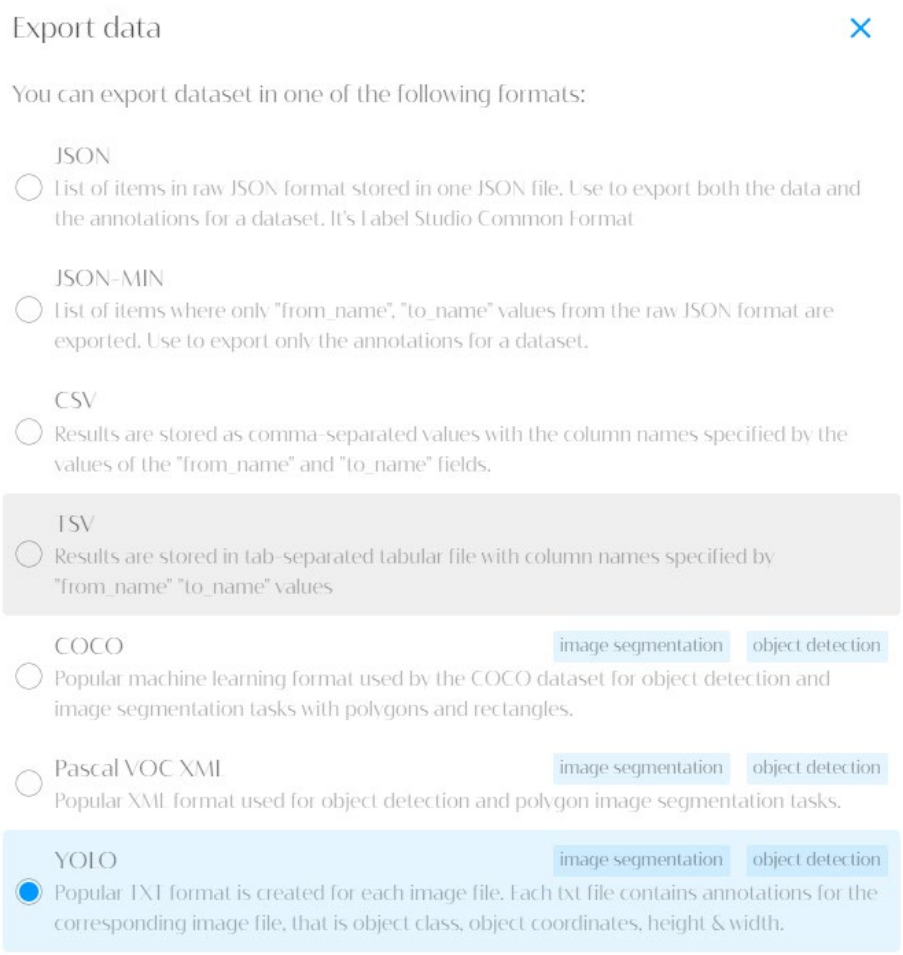


Рисунок 2.20. Експорт анотованих зображень для YOLO

5. Налаштування та тренування моделі YoloV5.

Перш за все організуємо експортовані дані за наступною схемою, де буде створена загальна папка «dataset», яка включатиме папку train, що налічуватиме 200 анотованих зображень та папку val – 60 анотованих зображень:

```
├── dataset
│   ├── classes.txt
│   │
│   ├── train
│   │   ├── images
│   │   └── labels
│   └── val
│       ├── images
│       └── labels
└── yolov5
    ├── data.yaml
    └── yolo5s.pt
```

```
# Завдання №4. Налаштування та тренування моделі YoloV5.

# Завантаження zip архіву з датасетом
uploaded = files.upload()
!unzip /content/proj5.zip -d /content/dataset/

images_path = '/content/dataset/train'  # шлях для загального пулу
зображень та анотацій (тренувальний набір)
val_images_path = '/content/dataset/validation'  # шлях до зображень для
перевірки

# Створеня тек за їх відсутності
os.makedirs(images_path, exist_ok=True)
os.makedirs(val_images_path, exist_ok=True)

# Підготовка файлу конфігурації для навчання
# Це файл YAML, в якому вказані шляхи до даних, параметрів моделі та
параметрів навчання

yaml_content = f"""
path: ../datasets  # шлях до набору даних
train: {images_path}  # шлях до тренувальних даних
val: {val_images_path}  # шлях до валідаційних даних

# Кількість класів
nc: 2

# Назви класів
names: ['fence', 'gate']
"""

with open("data.yaml", "w") as file:
    file.write(yaml_content)

# Навчання моделі, де --img - розмір зображення,
# --batch - кількість зображень, що використовуються за одну епоху,
# epoch -- кількість повних проходів навчальних даних
# -- data - шлях до конфігураційного файлу yaml
# --weights ваги, що в даному випадку визначаються моделю yolov5s.pt
# -- cache кеш для зображень та анотацій

# Примітка! Дані параметри вказані як демонстраційні.
# Задля досягнення якісного та швидкого навчання бажано використовувати
GPU власного комп'ютера або іншої платформи
# зі збільшенням кількості epoch в діапазоні значень 30-50.

!python train.py --img 416 --batch 16 --epochs 5 --data data.yaml --
weights yolov5s.pt --cache
```

```
30 epochs completed in 0.459 hours.
Optimizer stripped from runs\train\exp18\weights\last.pt, 14.3MB
Optimizer stripped from runs\train\exp18\weights\best.pt, 14.3MB

Validating runs\train\exp18\weights\best.pt...
Fusing layers...
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|          | 2/2 [00:04<0
                 all         60        123       0.63      0.582      0.604      0.333
               fence         60         78      0.502      0.609      0.561      0.289
                gate         60         45      0.758      0.556      0.647      0.377
Results saved to runs\train\exp18
```

Рисунок 2.21. Результат тренування моделі Yolov5

6. Проаналізуйте результати класифікації та детекції

Таким чином, загальний показник Precision для всіх класів становить 0.63, в той час як для fence – 0.502, для gate – 0.758. Показник Recall закономірно демонструє для всіх класів – 0.582, fence – 0.609, gate – 0.556. Можемо зробити висновок, що модель найкращим чином натренувалася на анотованих зображеннях воріт (gate), незважаючи на те, що екземплярів заборів (fence) було значно більше, що демонструє, що вхідні ознаки воріт є більш яскравими та доступними для навчання моделлю Yolov5, в той час як показник виявлення заборів наближається до показника відгадування, що може свідчити про недостатньо якісний або неповний набір даних. Але, при детальному аналізі на валідаційному наборі було також відмічено, що модель здебільшого коректно виділяє класи об’єктів, або взагалі не виділяє нічого.

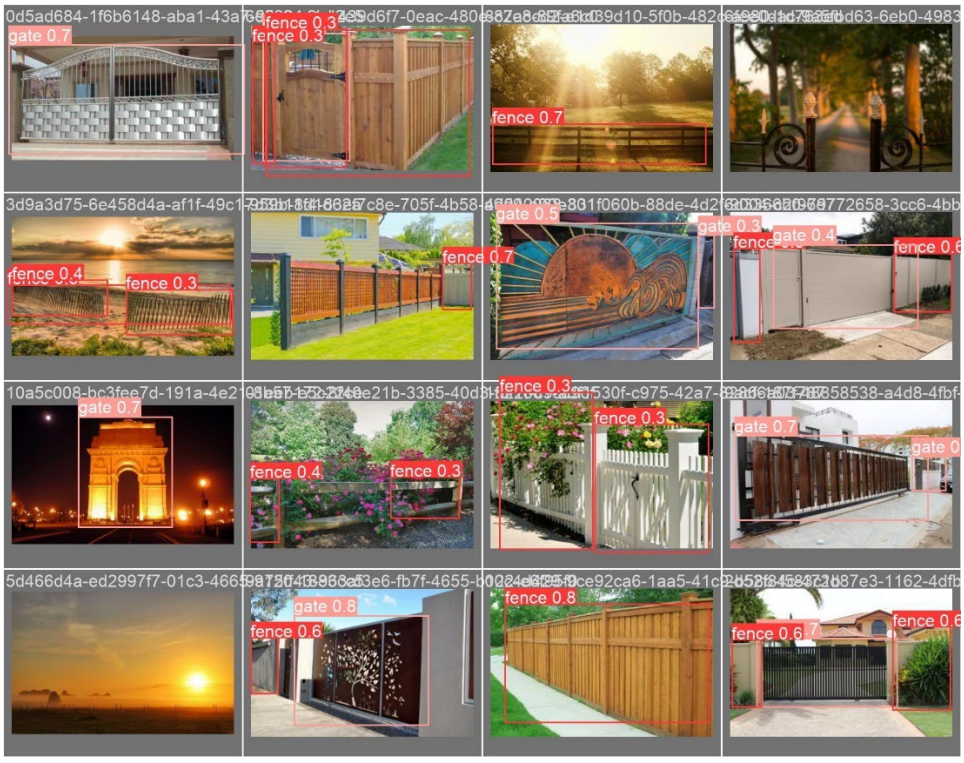


Рисунок 2.22. Результат детекції об’єктів моделі Yolov5

7. Використайте моделі YoloV8 для детекції довільних об’єктів на фото або відео.

```
# Завдання №6. Використання моделі Yolov8
# 1) pip install ultralytics
# 2) yolo predict model=yolov8n.pt
source='https://ultralytics.com/images/bus.jpg' # замініть посилання на
зображення.
# 3) yolo predict model=yolov8n.pt source='dance.mp4' # замініть шлях до
свого локального файлу.
```

```
C:\Users\Marano\PycharmProjects\pythonProject7\yolov5>yolo predict model=yolov8n.pt source='https://ultralytics.com/imag
es/bus.jpg'
Downloading https://github.com/ultralytics/assets/releases/download/v8.1.0/yolov8n.pt to 'yolov8n.pt'...
100%|                                                                                | 6.23M/6.23M [00:02<00:00, 2.55MB/s]
Ultralytics YOLOv8.1.9 🚀 Python-3.10.6 torch-2.2.0+cpu CPU (Intel Core(TM) i9-10940X 3.30GHz)
YOLOv8n summary (fused): 168 layers, 3151904 parameters, 0 gradients, 8.7 GFLOPs

Downloading https://ultralytics.com/images/bus.jpg to 'bus.jpg'...
100%|                                                                                | 476k/476k [00:00<00:00, 2.51MB/s]
image 1/1 C:\Users\Marano\PycharmProjects\pythonProject7\yolov5\bus.jpg: 640x480 4 persons, 1 bus, 1 stop sign, 123.7ms
Speed: 12.0ms preprocess, 123.7ms inference, 8.0ms postprocess per image at shape (1, 3, 640, 480)
Results saved to runs\detect\predict
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

Рисунок. 2.23. Результат виконання завдання №6

Рисунок. 2.24. Результат виконання завдання №6

```
video 1/1 (43/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 59.8ms
video 1/1 (44/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 1 person, 1 bottle, 55.9ms
video 1/1 (45/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 1 person, 1 bottle, 61.8ms
video 1/1 (46/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 1 person, 1 bottle, 49.9ms
video 1/1 (47/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 55.9ms
video 1/1 (48/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 56.8ms
video 1/1 (49/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 56.9ms
video 1/1 (50/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 53.9ms
video 1/1 (51/447) C:\Users\Marano\PycharmProjects\pythonProject7\papanya.mp4: 384x640 2 persons, 1 bottle, 50.9ms
```

Рисунок. 2.25. Результат виконання завдання №6

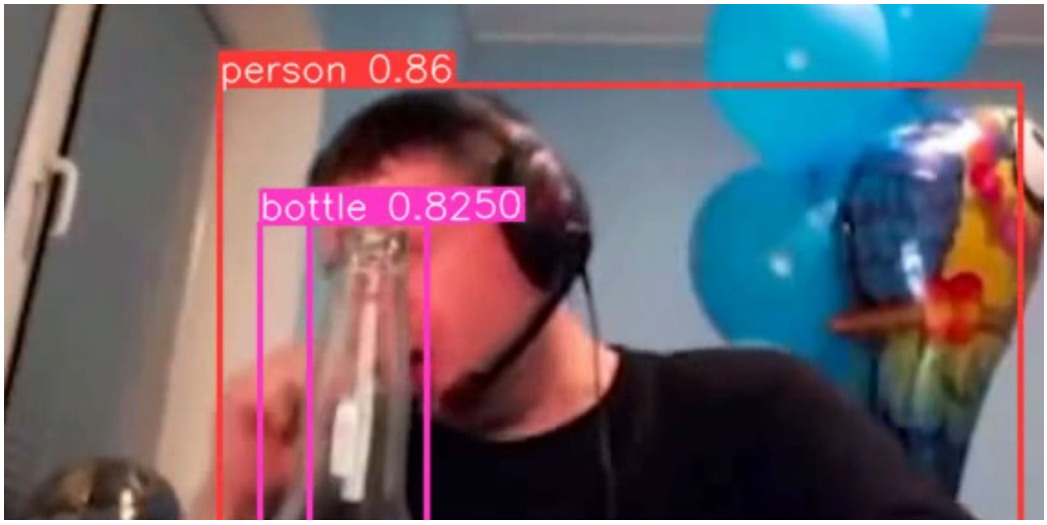


Рисунок. 2.26. Результат виконання завдання №6

Контрольне запитання №1

Вкажіть показник accuracy для останньої епохи навчання першої ResNet.
Відповідь: **0.727273**

Контрольне запитання №2

Вкажіть показник accuracy для останньої епохи навчання другої ResNet
Відповідь: **0.727273**

Контрольне запитання №3

Вкажіть кількість проанотованих зображень
Відповідь: **260**

Контрольне запитання №4

Як впливає кількість епох на процес навчання моделі, і які потенційні ризики пов’язані з великою кількістю епох?
Відповідь: …

Висновки: …

Завдання для виконання лабораторної роботи

1. Використовуючи запропонований код, зберіть датасет зображень за темою, вказаною у вашому варіанті. Завантажте не менше 30 зображень для кожного класу об'єктів.
2. Використовуючи бібліотеку FastAI, протестуйте різні моделі ResNet (вказані у вашому варіанті) для класифікації зображень. Визначте, яка з моделей демонструє кращу точність, при параметрах навчання вашого датасету.
3. Використайте Label Studio для анотації зібраних зображень. Кожне зображення повинно містити принаймні 1 анотований об’єкт, що вказаний у вашому варіанті.
4. Налаштуйте та навчіть модель YoloV5 на вашому анотованому датасеті для детекції об'єктів. Використовуйте параметри навчання (кількість епох, розмір пакету), вказані у вашому варіанті.
5. Проаналізуйте результати класифікації та детекції. Визначте, які класи об’єктів модель визначає найточніше, та у яких виникають складнощі при розпізнавані.
6. Використайте претреновану модель YoloV8 для детекції довільних об’єктів на фото або відео.

Частина 1					**Частина 2**		
Варіант	**Тема***	**Перша Модель ResNet*****	**Друга Модель ResNet*****	**Кіль-кість епох (fine_tune)****	**Анотації в Label Studio***	**Кількість епох (YoloV5)****	**Розмірність пакету (batch_size)****
1	Автомобілі	18	34	3	Автомобілі, дорожні знаки	5	16
2	Дерева	18	50	5	Дерева, квіти	7	32
3	Тварини	18	34	4	Тварини, птахи	9	8
4	Фрукти	18	50	6	Фрукти, овочі	5	16
5	Міські пейзажі	18	34	3	Будівлі, автомобілі	7	32
6	Пляжі	18	50	4	Море, пісок	9	16
7	Комп’ютерне обладнання	18	34	5	Комп’ютери, монітори	5	8
8	Спорт	18	50	6	Спортсмени, спортивне обладнання	7	32
9	Рослини	18	34	3	Квіти, дерева	9	16

10	Кава	18	50	5	Кавові чашки, кавові зерна	5	8
11	Годинники	18	34	4	Годинники, браслети	7	32
12	Взуття	18	50	6	Взуття, шкарпетки	9	16
13	Техніка	18	34	3	Смартфони, навушники	5	8
14	Велосипеди	18	50	5	Велосипеди, шоломи	7	32
15	Парки	18	34	3	Дерева, лавки	9	16
16	Живопис	18	50	4	Картини, скульптури	5	8
17	Страви	18	34	6	Страви, напої	7	32
18	Космос	18	50	3	Зірки, планети	9	16
19	Іграшки	18	34	5	Іграшки, ігрові консолі	5	8
20	Книги	18	50	4	Книги, бібліотеки	7	32
21	Гори	18	34	5	Гори, річки	9	16
22	Техніка	18	50	3	Клавіатури, приставки	5	8
23	Офіс	18	34	6	Офісне обладнання, робочі місця	7	32
24	Зоопарк	18	50	4	Тварини в зоопарку, клітки	9	16
25	Сади	18	34	5	Сади, парники	5	8
26	Мода	18	50	3	Модельєри, подіум	7	32
27	Скульптури	18	34	6	Скульптури, музеї	9	16
28	Музичні інструменти	18	50	4	Гітари, піаніно	5	8
29	Кулінарія	18	34	5	Страви, кухонне приладдя	7	32
30	Подорожі	18	50	3	Ландшафти, транспорт	9	16

[*] Студент може обрати свою тему самостійно, якщо вона не буде збігатися з темами інших студентів.
[**] Студент може змінювати параметри на більші, якщо бажає та має можливість задля досягнення кращих результатів навчання моделі.
[***] Студент може додатково виконати випробування з моделями ResNet101, ResNet152.

Контрольне запитання №1

Вкажіть показник accuracy для останньої епохи навчання першої ResNet.

Відповідь: _____

Контрольне запитання №2

Вкажіть показник accuracy для останньої епохи навчання другої ResNet.

Відповідь: _____

Контрольне запитання №3

Вкажіть кількість проанотованих зображень.

Відповідь: _____

Контрольне запитання №4

Як впливає кількість епох на процес навчання моделі, і які потенційні ризики пов’язані з великою кількістю епох?

Відповідь: _____

_____

_____

Welcome to Google Colab

https://colab.research.google.com/drive/1Yykj2DxeQy3HlbZIpAIheRSSmXFnwuWB?usp=sharing

Головне про Label Studio: **Label Studio** – це інструмент для маркування даних з відкритим вихідним кодом та дозволяє інтегрувати Label Studio з моделями машинного навчання, щоб надавати прогнози для міток або виконувати безперервне активне навчання.

Інсталяція Label Studio на Windows з pip

Примітка. У вас попередньо повинен бути встановлений Python версії 3.8 або вище.

Щоб встановити Label Studio з pip і віртуальним середовищем, виконайте наступні дії:

```
python3 -m venv env
source env/bin/activate
python -m pip install label-studio
```

Щоб встановити Label Studio з pip, виконайте наступні дії:

```
pip install label-studio
```

Після встановлення Label Studio запустіть сервер наступною командою:

```
label-studio
```

Веб-браузер за замовчуванням автоматично відкриє сторінку http://localhost:8080 з Label Studio.

За додатковою інформацією щодо Label Studio звертайтеся до документації https://labelstud.io/guide/install