МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

(підпис)

червня 202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Додаток для аналізу образів з використанням комп'ютерного зору»

здобувача групи ІН-06-2 Шапочки Владислава Романовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Владислав ШАПОЧКА

(підпис)

Керівник, старша викладач кафедри комп'ютерних наук, к.ф.-м.н. Бадалян А.Ю.

Анна БАДАЛЯН

(підпис)

Суми – 2024

Сумський державний університет

Центр заочної, дистанційної та вечірньої форм навчання

Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-06-2 Шапочки Владислава Романовича

1. Тема роботи: «Додаток для аналізу образів з використанням комп'ютерного зору» затверджую наказом по СумДУ від _____

2. Термін здачі здобувачем кваліфікаційної роботи _____

3. Вхідні дані до кваліфікаційної роботи _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити):

1) *Інформаційний огляд, аналіз предметної області, постановка задачі, моделювання інформаційної системи.*
2) *Вибір методів рішення задачі.*
3) *Програмна реалізація додатку.*
4) *Тестування та поліпшення додатку.*

_____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ (підпис)

Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Вивчення та аналіз поставленої задачі*	06.05.24-09.05.24	
2	*Поглиблене дослідження бібліотек, інструментів, технологій, що будуть використовуватися для створення додатку*	10.05.24-13.05.24	
3	*Програмна реалізація додатку*	14.05.24-24.05.24	
4	*Аналіз отриманих результатів та покращення додатку*	25.05.24-29.05.24	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	30.05.24-31.05.24	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 36 стр., 18 рис., 1 додаток, 17 використаних джерел.

Обґрунтування актуальності теми роботи – Тема роботи є актуальною через зростання зацікавленості у сфері комп'ютерного зору та його застосування у різних галузях. Розвиток алгоритмів машинного навчання відкриває нові можливості для автоматизованого аналізу зображень та виявлення об'єктів на них. Розробка програм, які здатні аналізувати та обробляти зображення з використанням комп'ютерного зору, є важливим напрямом досліджень у сучасній інформаційній технології, сприяючи швидкому та ефективному аналізу інформації в різних сферах.

Об’єкт дослідження — процес знаходження об'єктів за допомогою переднавченої моделі Yolov3.

Мета роботи — розробка власного додатку аналізу об'єктів на зображеннях та відео файлах з використанням переднавченої моделі Yolov3.

Методи дослідження — алгоритми пошуку об'єктів за допомогою переднавченої моделі Yolov3 та інструменти побудови математичних моделей.

Результати — розроблено додаток для аналізу зображень та відео за допомогою комп'ютерного зору, який використовує переднавчену модель YOLO для виявлення об'єктів. Додаток має зручний інтерфейс, де користувач може вибрати файл для аналізу та налаштувати параметри. Проведено тестування та відлагодження додатка, яке покращило його якість аналізу.

ІНФОРМАЦІЙНА СИСТЕМА, АНАЛІЗ ОБ’ЄКТІВ, ПЕРЕДНАВЧЕНА МОДЕЛЬ YOLO, КОМП’ЮТЕРНИЙ ЗІР, PYTHON

ЗМІСТ

ВСТУП

Комп'ютерний зір (або комп'ютерне зорове сприйняття) - це галузь штучного інтелекту, яка зосереджена на розробці систем, що можуть аналізувати та розуміти візуальні дані так само, як це робить людський мозок. Обрана тема є надзвичайно актуальною в сучасному світі завдяки її широким можливостям у різних сферах діяльності, таких як медицина, виробництво, безпека, автомобільна промисловість, робототехніка, відеоігри та інші. Швидкість розвитку цієї технології вражає, і комп'ютерний зір стає все більш невід'ємною частиною нашого сучасного світу[1].

Зростаюче різноманіття технологій та рівень очікувань користувачів вимагають постійного вдосконалення систем комп'ютерного зору для забезпечення точності та ефективності їх роботи. Адаптивні алгоритми стають ключовим інструментом в цьому процесі, забезпечуючи оптимальне функціонування систем незалежно від умов зовнішнього середовища.

Актуальність розробки технологій комп'ютерного зору пояснюється наступними факторами:

- Зростання обсягів даних. З кожним роком зростає кількість візуальної інформації, що потребує обробки. Комп'ютерний зір дозволяє автоматично аналізувати великі обсяги зображень та відео, надаючи бізнесам та дослідникам можливість отримати цінні інсайти з даних.
- Зростання автоматизації. Технології комп'ютерного зору активно використовуються для автоматизації виробничих процесів, забезпечення безпеки, контролю якості та інших завдань, що потребують аналізу візуальної інформації.
- Зростання конкуренції. Впровадження комп'ютерного зору дозволяє компаніям виділитися серед конкурентів, надаючи своїм клієнтам

інноваційні продукти та послуги, підвищуючи ефективність бізнес-процесів.

- Зростання вимог споживачів. Сучасні користувачі очікують високої точності та швидкості від систем, що використовують комп'ютерний зір. Розробка таких технологій дозволяє бізнесам відповідати на ці вимоги, забезпечуючи надійне розпізнавання об'єктів, аналіз зображень та інших візуальних даних[2].

Об'єктом дослідження було обрано модель комп'ютерного зору YOLO та її застосування для виявлення та розпізнавання об'єктів.

Гіпотеза даного дослідження полягає в тому, що модель комп'ютерного зору YOLO (You Only Look Once) може забезпечити високу точність і швидкість виявлення та розпізнавання об'єктів на зображеннях і відео, що дозволить значно підвищити ефективність різних бізнес-процесів та автоматизувати завдання, які потребують аналізу візуальної інформації.

Новизною даного дослідження є використання моделі YOLO для вирішення проблеми впізнавання об'єктів у складних умовах, таких як обмежена освітленість, рух, або перешкоди на шляху об'єктів.

- Автоматичне корегування параметрів під час роботи: Модель YOLO володіє можливістю автоматично адаптуватися до змінних умов освітлення та обставин, що дозволяє підвищити точність виявлення об'єктів навіть в умовах низької видимості.
- Врахування динамічних умов навколишнього середовища: Розширення функціональності моделі YOLO для виявлення та відстеження об'єктів з урахуванням їхнього руху та зміни форми в залежності від ситуації.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Галузі використання комп’ютерного зору

Комп'ютерний зір (або комп'ютерне зорове сприйняття) знаходить широке застосування у багатьох галузях, включаючи наступні:

- Медицина: Використовується для медичної діагностики, виявлення патологій на зображеннях, навчання хірургічних роботів, візуалізації медичних зображень тощо.

- Промисловість: Для якості контролю на виробництві, автоматизації процесів, розпізнавання деталей, контролю якості продукції, трекінгу та управління логістикою.

- Автомобільна промисловість: В системах допомоги при водінні, автономних автомобілях, відстеженні дорожньої ситуації, розпізнаванні знаків тощо.

- Безпека: Відеоспостереження, виявлення вторгнень, розпізнавання облич, відстеження руху тощо.

- Агротехнології: Моніторинг рослин, виявлення хвороб, оцінка урожаю, автоматизовані системи поливу та догляду за рослинами.

- Транспорт і логістика: Відстеження вантажів, оптимізація маршрутів, автоматизація сортування та розподілу вантажів.

- Розваги: Від ігор з розширеною реальністю до систем віртуальної реальності, відстеження рухів у відеоіграх тощо.

- Наука та дослідження: Обробка зображень для аналізу даних, розпізнавання об'єктів у великих обсягах даних тощо.

- Охорона навколишнього середовища: Моніторинг забруднення, виявлення змін в екосистемах, контроль за лісовими масивами та інші аспекти.

- Фінанси: Аналіз зображень для фінансових рішень, виявлення шахраїв **тощо.**

2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Вибір мови програмування та середовища для створення додатку

У своїй роботі я використовував мову програмування Python [3]. Вона є простою та лаконічною, має велику кількість бібліотек AI (TensorFlow, PyTorch, scikit-learn, Keras) та є популярним вибором для навчання AI та машинного навчання.

Середовищем для написання додатку я обрав PyCharm 2024.1[4] - це одне з найпопулярніших інтегрованих середовищ розробки для Python.
Створено новий файл проекту (рис. 2.1):

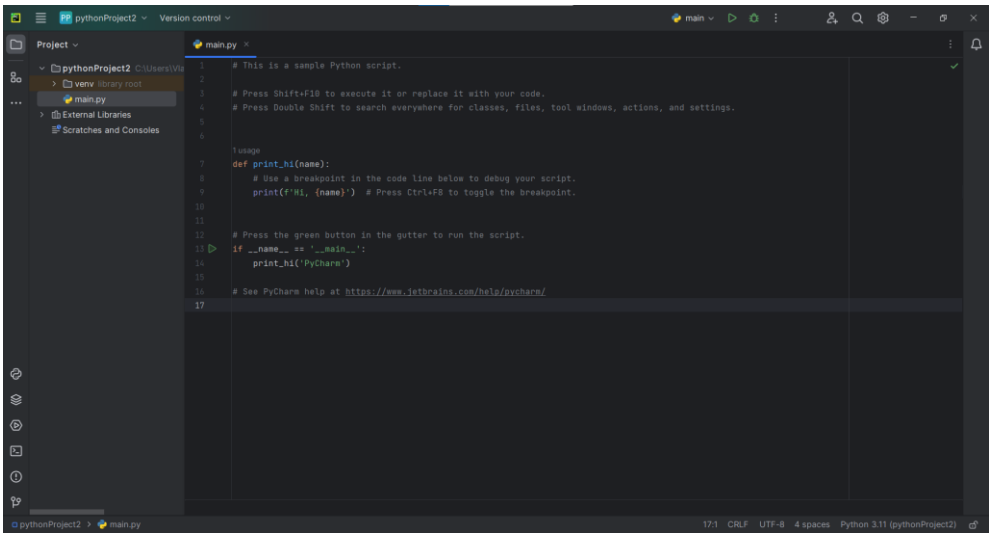


Рисунок 2.1 - Вікно програми PyCharm з новим проектом

У PyCharm є багато елементів керування, які полегшують розробку на Python:

- Панель інструментів
- Панель проекту
- Редактор коду
- Консоль Python
- Вікно терміналу
- Панель інструментів проекту
- Вкладки редактора

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір моделі аналізу зображень

Переглянувши доступні переднавчені моделі для аналізу зображень, я обрав Yolo (You only look once) (рис. 3.1) [5].

Рисунок 3. 1 - Логотип моделі аналізу об’єктів Yolo

Yolo — це захоплююча модель для об'єктного виявлення, яка стала справжнім проривом у світі комп'ютерного зору та штучного інтелекту. Ця модель, розроблена командою дослідників, відома своєю ефективністю та швидкодією, що робить її ідеальним вибором для багатьох застосувань. Вона створена для пошуку різних об’єктів на фото, наприклад людей, машин, тварин, рослин та інших[6].

Yolo відрізняється від своїх аналогів, таких як Faster R-CNN, SSD (Single Shot Multibox Detector) і RetinaNet, рядом унікальних переваг та недоліків.

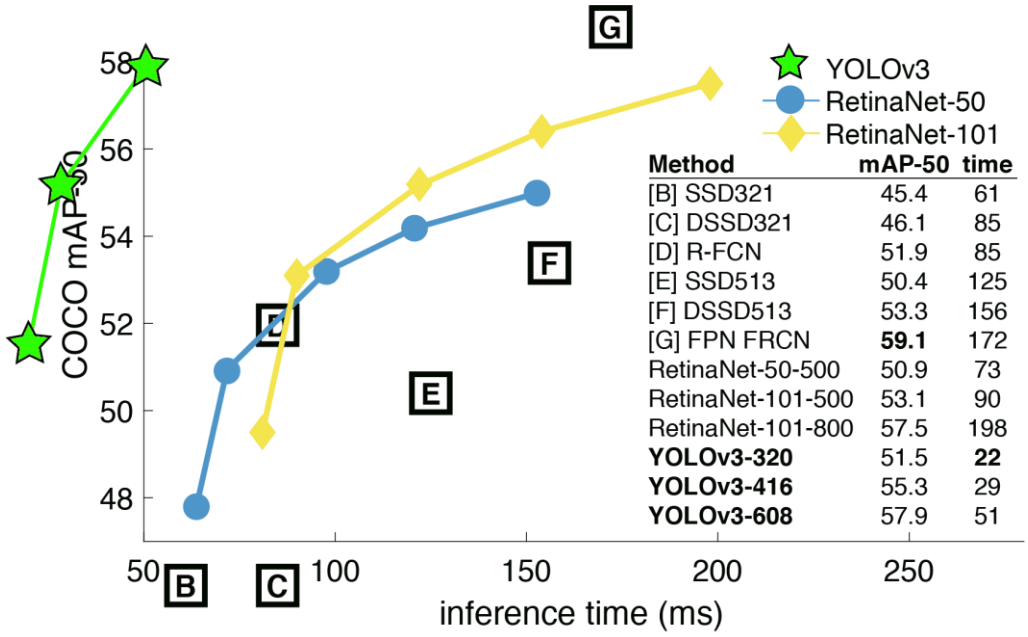
Переваги Yolo:

- Швидкість: Однією з найбільших переваг Yolo є його висока швидкість обробки. У порівнянні з іншими моделями, вона вражає своєю ефективністю, що робить її ідеальним вибором для використання у реальному часі або на великих наборах даних.

- Простота: Yolov3 відома своєю простотою в реалізації та використанні. Вона має простий архітектурний дизайн, що полегшує її використання і розуміння.
- Точність: Незважаючи на свою швидкість, Yolov3 здатна досягати високої точності у виявленні об'єктів, особливо коли вона навчена на великих об'ємах даних.

Недоліки Yolo:
- Вимоги до обчислювальних ресурсів: У порівнянні з іншими моделями, Yolov3 може вимагати більше обчислювальних ресурсів для ефективної роботи, що може бути проблемою у використанні на обмежених платформах.
- Погана точність на малих об'єктах: У деяких випадках Yolov3 може мати проблеми з точністю виявлення дрібних об'єктів на зображеннях чи відео порівняно з іншими алгоритмами, такими як Faster R-CNN.

Порівняно з аналогами, Yolov3 вирізняється своєю комбінацією високої швидкості та хорошої точності, що робить її популярним вибором для багатьох застосувань у галузі об'єктного виявлення (рис. 3.2) [7].



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Рисунок 3. 2 - Порівняння Yolo з іншими схожими моделями

3.2 Принцип аналізу зображень з використанням моделі Yolov3

Так як Yolo необхідний лише один погляд на зображення, то метод «плаваючого» вікна не підходить у цій ситуації. Замість цього зображення буде поділено на сітку з комірками розміром S x S. Кожна комірка може містити кілька різних об’єктів для розпізнавання.

По-перше, кожна комірка відповідає за прогнозування кількості обмежувальних рамок. Крім того, будь-яка комірка прогнозує довірче значення (значення довіри) для кожної області, обмеженої обмежувальною рамкою. Іншими словами, це значення визначає ймовірність знаходження того чи іншого об'єкта в даній області. Це є в тому випадку, якщо якась комірка сітки не має визначеного об'єкта, важливо, щоб довірче значення для цієї області було низьким [8].

Коли ми візуалізуємо всі передбачення, ми отримуємо карту об'єктів і упорядкованих за довіреним значенням, рамки (рис. 3.3).

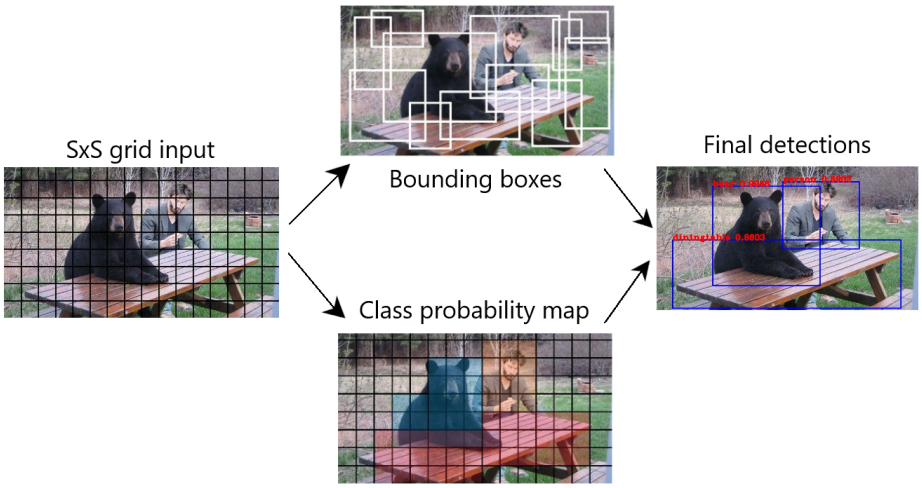


Рисунок 3. 3 - Приклад роботи моделі Yolov3

По-друге, кожна комірка відповідає за прогнозування ймовірностей класів. Це не говорить про те, що якась комірка містить об'єкт, а тільки ймовірність знаходження об'єкта. Припустимо, якщо комірка передбачує

автомобіль, це не гарантує, що автомобіль в дійсності присутній в ній. Це говорить лише про те, що якщо присутній об'єкт, то цей об'єкт швидше всього автомобіль.

Також YOLO використовує Anchor boxes. Anchor boxes - це визначені прямокутні форми, які використовуються для прогнозування обмежувальних рамок (bounding box) (рис. 3.4). Вони допомагають моделі точніше визначати об'єкти, так як дозволяють враховувати різні розміри і співвідношення об'єктів. Anchor boxes створюються шляхом кластеризації даних COCO (Common Objects in Context) з використанням алгоритму k-середніх (K-means clustering)[9].

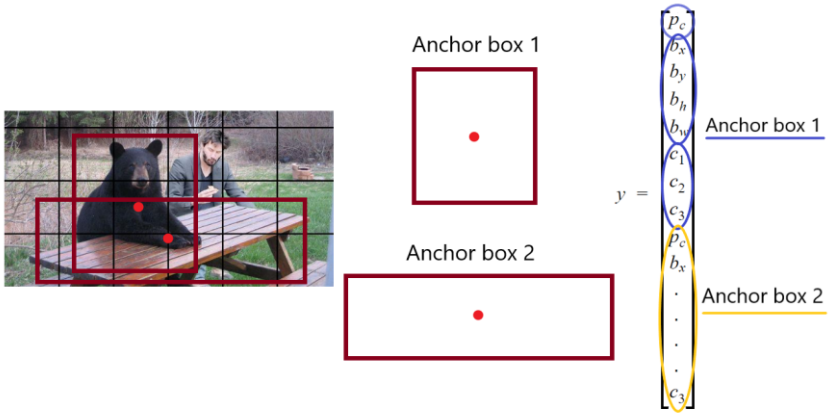


Рисунок 3. 4 - Алгоритм знаходження Anchor box

Кожна комірка сітки передбачає для кожного bounding box'а його координати (tx, ty, tw, th), показник впевненості у присутності об'єкта та ймовірності класів. Особливість YOLO полягає в тому, що він передбачає координати bounding box'ів щодо центру осередку, а не щодо всього зображення, що підвищує точність виявлення. У результаті на виході отримуємо тензор, що відображає виявлені об'єкти на зображенні. Крім того, YOLO працює з bounding box'ами у трьох різних масштабах для більш точного виявлення об'єктів.

У YOLO збільшення ймовірності на довірчі значення дає зважувані обмежувальні рамки. Встановлення порогового значення відсіює ненадійні прогнози. Метрика IoU визначає ступінь перекриття між передбаченими та справжніми обмежувальними рамками (рис. 3.5).

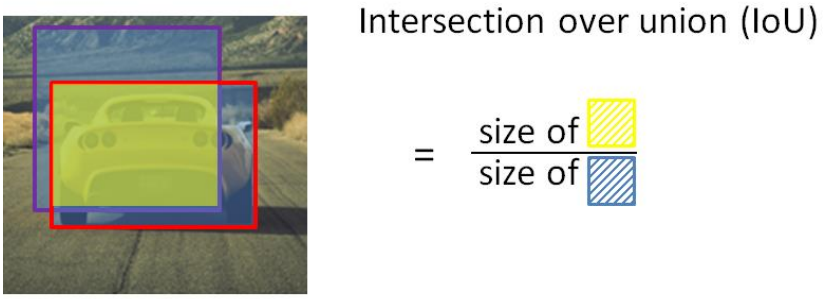


Рисунок 3. 5 - Принцип знаходження метрики IoU

Після цього все одно можуть залишитися дублікати об’єктів, і щоб їх позбутися потрібно використовувати "non-maximum suppression" метод.

"Non-maximum suppression" – це метод, який використовується для видалення дублікатів у результатах виявлення об'єктів. Він заснований на виборі bounding box'a з найбільшою ймовірністю приналежності до об'єкта та придушення інших, які сильно перекриваються з ним. Це дозволяє уникнути надлишкових виявлень та підвищує точність моделі. Крім того, завдяки тому, що модель передбачає всі об'єкти одночасно, вона може враховувати глобальний контекст і відносини між об'єктами, що сприяє більш точному виявленню.

Архітектура YOLO дозволяє обробляти зображення в реальному часі навіть на ресурсомістких пристроях, таких як мобільні телефони або вбудовані системи. Крім того, застосування YOLO для задач виявлення об'єктів на відео дозволяє вирішувати завдання трекінгу об'єктів та аналізу руху, що робить його потужним інструментом у різних галузях, включаючи відеоспостереження, автономну навігацію та медичну діагностику[10].

3.3 Написання додатку для аналізу зображень на мові Python

При написанні додатку, я використовував такі бібліотеки Python [11]:

- OpenCV
- NumPy
- Pillow
- OS
- Tkinter
- Matplotlib
- Threading

Вони мають весь необхідний функціонал для створення додатку.

Цей додаток реалізує зручний користувацький інтерфейс та основну частину для аналізу зображень за допомогою алгоритму YOLO, який використовується для об'єктного розпізнавання. Давайте розглянемо основні компоненти та функції цього додатку[12].

Основні компоненти і функції:

main.py містить інформацію про головне вікно додатку (рис. 3.6) та описує взаємодію користувача з ним:

- Це вікно, яке відображається при запуску програми.
- Містить поля для введення порогів впевненості та NMS методу[13].
- Має кнопку для вибору зображення та кнопку для аналізу обраного зображення.
- Містить захист від введення некоректної інформації користувачем

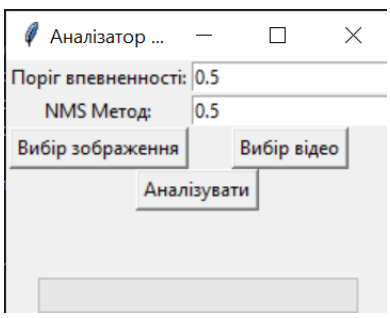


Рисунок 3. 6 - Головне вікно додатку

Методи класу YoloApp:

- init: Цей метод відповідає за ініціалізацію головного окна додатку та його компонентів. Він створює всі необхідні елементи і розміщує їх біля вікна.

- select_image: Цей метод викликається, коли користувач натискає кнопку "Вибір зображення". Він відкриває діалогове вікно для вибору зображення із файлової системи та завантажує обране зображення.

- select_video: Цей метод викликається, коли користувач натискає кнопку "Вибір відео". Він відкриває діалогове вікно для вибору відео з файлової системи та завантажує перший кадр вибраного відео для відображення.

- check_file_type: Цей метод перевіряє тип файлу за його розширенням. Він приймає шлях до файлу і список допустимих розширень, а повертає True, якщо тип файлу відповідає допустимим розширенням, і False інакше.

- display_image: Цей метод відображає зображення у головному вікні додатку. Він приймає зображення у форматі PIL.ImageTk.PhotoImage та відображає його у вікні.

- process_data: Цей метод викликається при натисканні кнопки "Аналізувати". Він набуває значення впевненості та порога NMS методу з поля введення, перевіряє їх на валідність, а потім аналізує зображення або відео відповідно до обраного типу даних.

- process_video: Цей метод виконує аналіз відео. Він відкриває відеофайл, аналізує кожен кадр відео, викликаючи функцію обробки кадру, та записує результат у новий відеофайл.

- update_progress: Цей метод оновлює значення прогресу. Він приймає значення прогресу та оновлює прогресбар згідно із заданим значенням.

- get_total_frames: Цей метод повертає загальну кількість кадрів у відеофайлі. Він відкриває відеофайл, отримує загальну кількість кадрів, а потім повертає це значення.

Функції модуля yolo_utility:

- filter_out: Ця функція фільтрує вихідні дані шару YOLO згідно з впевненістю (confidence). Вона приймає вихідні дані шару, поріг впевненості та повертає координати областей (xywh), впевненість та класи, які перевищують встановлений поріг.

- iou: Ця функція обчислює перекриття двох прямокутників (Intersection over Union - IOU). Вона приймає координати двох прямокутників та повертає значення IOU.

- non_max_supression: Ця функція виконує неперекриваючу сортування прямокутників на основі їх впевненості та IOU. Вона приймає координати прямокутників, впевненість, поріг впевненості та поріг IOU, і повертає індекси прямокутників, які залишаються після застосування NMS.

- rescale_box: Ця функція перераховує координати прямокутників зі шкали [0, 1] до вихідних розмірів зображення. Вона приймає координати прямокутників та розміри вихідного зображення і повертає координати прямокутників у вихідних розмірах.

- draw_boxes: Ця функція малює прямокутники та написи з класами та впевненістю на зображенні. Вона приймає зображення, координати прямокутників, індекси прямокутників після NMS, впевненість, класи, мітки, кольори та повертає зображення з нарисованими прямокутниками та мітками.

- photo_detection: Ця функція виконує аналіз зображення за допомогою моделі YOLO. Вона приймає зображення, модель YOLO, поріг впевненості, поріг NMS, мітки, кольори та (необов'язково) ім'я файлу зображення. Після аналізу вона виводить зображення з нанесеними прямокутниками та мітками на екран та, за наявності ім'я файлу, зберігає результат у файл.

- video_detection: Ця функція виконує аналіз відео за допомогою моделі YOLO. Вона приймає шлях до відеофайлу, модель YOLO, поріг впевненості, поріг NMS, мітки та кольори. Після аналізу вона записує результат у відеофайл.

Робота з моделлю YOLO:

Для використання YOLO в цьому додатку використовуються вагові файли (yolov3.weights) та конфігураційний файл (yolov3.cfg) [14]. Вони повинні бути доступні в папці yolo разом з файлом coco.names, який містить назви класів, що використовуються моделлю для знаходження об'єктів.

Модель завантажується за допомогою такої функції як cv2.dnn.readNetFromDarknet, яка використовує файли конфігурації та вагові файли.

Потім зображення або відео файл нормалізується та передається через мережу YOLO за допомогою наступної функції cv2.dnn.blobFromImage. Виведення мережі обробляється для виявлення об'єктів на зображенні.

Обробка результатів для відеоаналізу:

Додаток здатен обробляти відео файли, а не лише окремі зображення. Після аналізу кожного кадру відео результати виводяться в тому ж самому форматі, як і для окремих зображень, з можливістю відображення на головному вікні та збереження у вихідний відео файл. Для зручності користувача, внизу головного вікна є лінія, яка відображає прогрес аналізу відео.

Взаємодія з користувачем:

Користувач може налаштувати пороги впевненості та NMS методу перед аналізом зображення. Файли обираються за допомогою діалогового вікна файлової системи, що дозволяє користувачеві легко орієнтуватись. Також для більшої зручності обраний файл буде представлений в головному вікні програми, так можна точно розуміти, який файл буде аналізуватись.

Відображення результатів:

Після завершення аналізу з’являється вікно з готовим результатом. Кожен виявлений об'єкт має відповідну рамку з його класом та ймовірністю. Також кожен результат проаналізованого файлу зберігається на комп’ютері, для подальшого використання користувачем.

Обробка помилок:

Додаток має механізми обробки помилок для випадків, коли користувач вводить недійсні значення порогів впевненості або NMS методу. Також є перевірка наявності файлу перед початком його аналізу.

Додаток може працювати з різноманітними зображеннями, включаючи зображення реального світу, фотографії та відеозаписи. Для оптимальної роботи моделі важливо, щоб об'єкти на зображеннях були достатньо великими та чіткими для зручного виявлення. Також якість результатів може залежати від освітлення та кута зйомки.

Зображення повинні мати підтримуваний формат, такий як JPG, JPEG або PNG, щоб бути обробленими бібліотекою OpenCV та моделлю YOLO. Для відео файлу формат повинен бути MP4, AVI або MOV.

Модель YOLOv3 може виявляти та розрізняти різні типи об'єктів, включаючи, але не обмежуючись: люди, автомобілі, тварини, велосипеди, мотоцикли, рослини, дерева, меблі, кухонні прилади, фрукти, овочі, книги та інші предмети. Модель може виявляти об'єкти різних розмірів та варіантів орієнтації на зображенні.

Крім вищезазначеного, важливо враховувати, що обробка зображень може займати певний час, особливо при великій кількості об'єктів на зображенні або відео. Це може вплинути на продуктивність додатку, особливо при обробці великих відео файлів.

4 ТЕСТУВАННЯ ДОДАТКУ

4.1 Тестування готового додатку

Після завантаження програми відкривається вікно головного меню. В цьому вікні можна власноруч ввести такі данні як поріг впевненності та NMS Метод. Додаток має попередження при введенні некоректних даних (рис. 4.1).

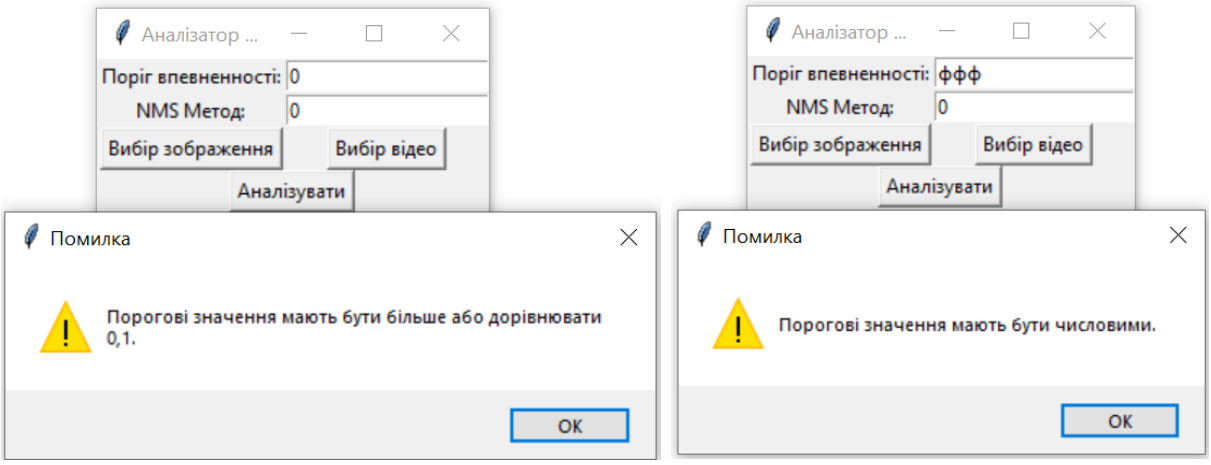


Рисунок 4. 1 - Захист від введення некоректних даних

Нижче розташовані кнопки «Вибір зображення» та «Вибір відео» (рис. 4.2).

Рисунок 4. 2 - Головне вікно додатку з обраним зображенням

Після виставлення вхідних даних можна тиснути на кнопку «Аналізувати зображення» та отримати результат (рис. 4.3).

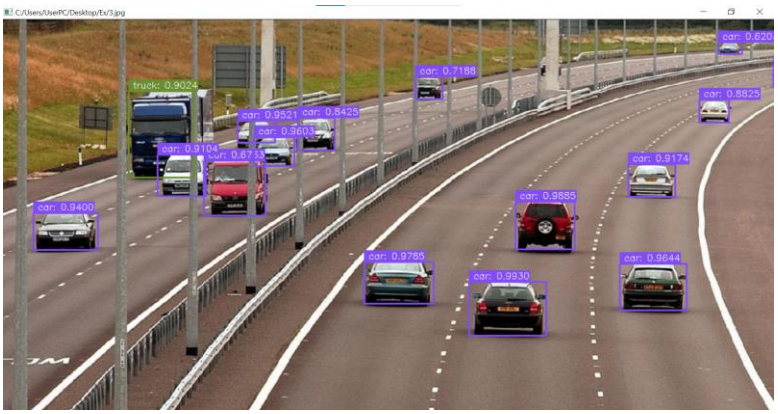


Рисунок 4. 3 - Результат аналізу зображення

Результатом є нове зображення зі знайденими на ньому об’єктами за допомогою комп’ютерного зору. Як можна побачити, біля кожного об’єкта присутня назва та відсоток передбачення цього об’єкту.

На демонстраційному зображенні можна помітити, що ті автомобілі які погано видно, мають менше число передбачення, ніж ті що видно добре. Також модель змогла окремо «побачити» та виділити вантажівку серед інших автомобілів (рис. 4.4).

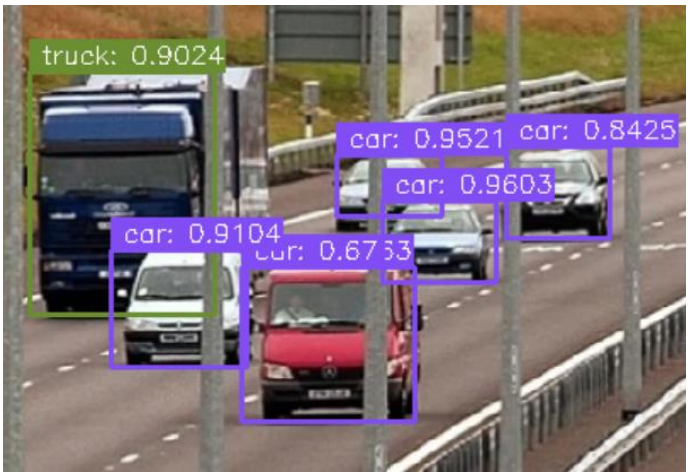


Рисунок 4. 4 - Результат аналізу зі скупченням об’єктів

Також інше цікаве зображення, на якому модель змогла розпізнати автомобілі у відображенні на склі будинку, хоч відсоток впевненості не великий (рис. 4.5).

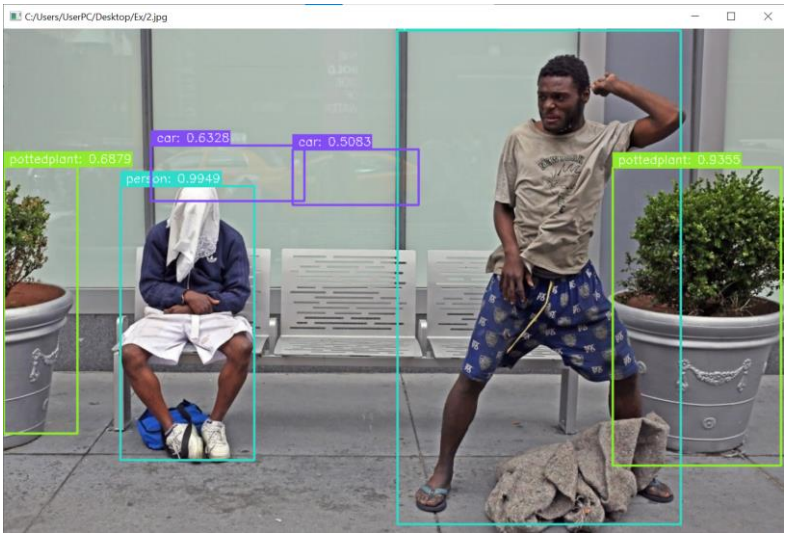


Рисунок 4. 5 - Результат аналізу зі скупченням об’єктів

З недоліків можна відмітити те, що якщо об’єкт буде перекритий, або показаний не повністю, то система може не «побачити його», або не правильно ідентифікувати (рис. 4.6).

Рисунок 4. 6 - Результат аналізу з погано видимими об’єктами

Якщо додаток аналізує відео файл, то знизу з’являється надпис «Аналіз відео…», а лінія вказує на прогрес аналізу (рис. 4.7)[15].

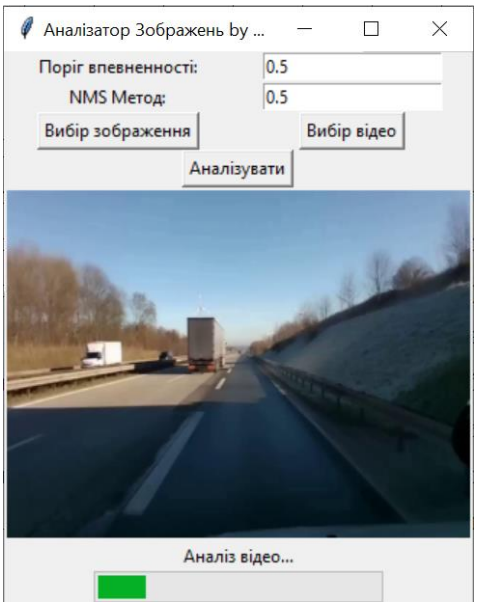


Рисунок 4. 7 - Процес аналізу відео файлу

Результатом аналізу буде новий файл, того ж формату, що і вхідний зі зміненою назвою «Проаналізований відеофайл». На ньому буде виділено та підписано всі об’єкти, аналогічно як із аналізом зображень (рис. 4.8).

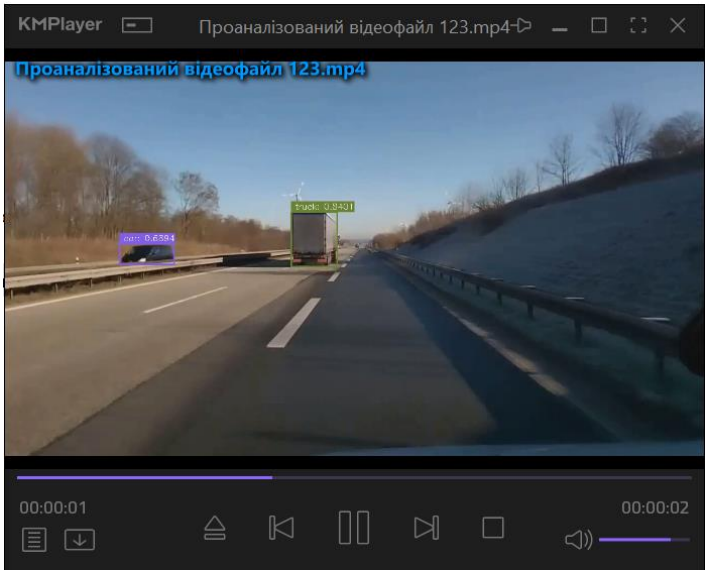


Рисунок 4. 8 - Результат аналізу відео файлу

Після проведення тестування додатку було отримано графік, який наглядно демонструє якість знаходження об'єктів моделлю (рис 4.9).

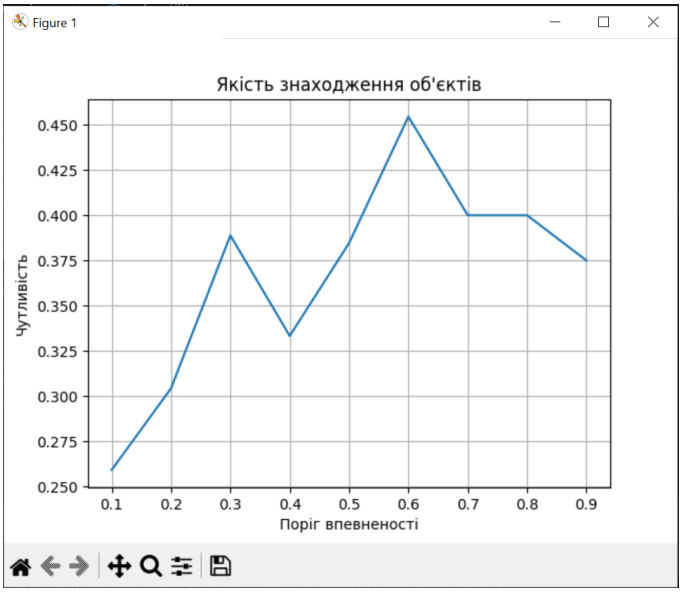


Рисунок 4. 9 - Графік якості TPR

На графіку представлене відношення чутливості (True Positive Rate) до порогу впевненості[16]. TPR визначається як частка істинно позитивних передбачень (True Positives) щодо загальної кількості істинно позитивних випадків даних (рис 4.10):

$$\mathrm{TPR} = \frac{\mathrm{True\ Positives}}{\mathrm{True\ Positives} + \mathrm{False\ Negatives}}$$

Рисунок 4. 10 - Формула знаходження TPR

Де:

True Positives (TP) - кількість об'єктів, які були класифіковані як позитивні класи.

False Negatives (FN) - кількість об'єктів, які були неправильно класифіковані як негативні класи, хоча вони насправді належать до позитивного класу.

TPR показує, наскільки добре модель ідентифікує позитивні випадки всіх реальних позитивних випадків у даних. Високе значення TPR вказує на те, що модель добре виявляє позитивні випадки та має низьку схильність до пропуску позитивних випадків (False Negatives)[17].

Після знаходження якості, було проведено зміну параметрів фільтрації та «non_max_supression» методу та відлагодження моделі, заради збільшення продуктивності та якості знаходження об'єктів системою. Було отримано наступний графік (рис 4.11).

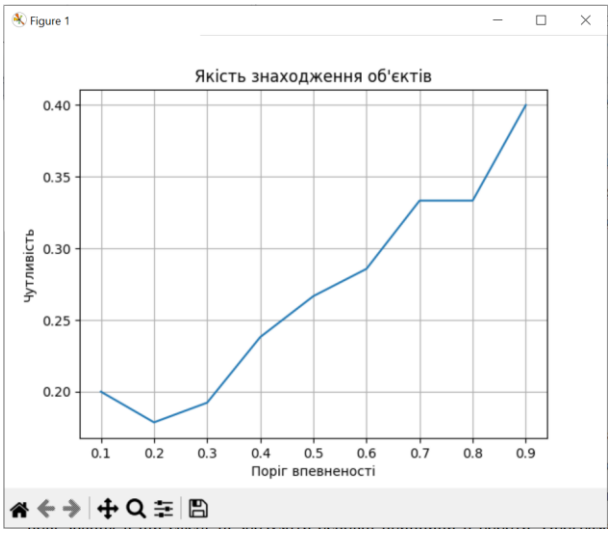


Рисунок 4. 11 - Графік якості TPR після відлагодження моделі

На графіку TPR чітко видно, що з різних значеннях порога впевненості (threshold), TPR нашої моделі значно зріс. Це свідчить про те, що покращена модель має більш високу чутливість і більш точно класифікує позитивні випадки, зводячи до мінімуму кількість помилково-негативних передбачень (False Negatives).

Таким чином, ці покращення не тільки підвищили точність моделі, але й значно збільшили її надійність у практичних додатках, що є найважливішим критерієм у завданні аналізу об'єктів.

ВИСНОВКИ

В результаті виконаної роботи була розроблена програма, яка за допомогою комп’ютерного зору аналізує зображення та відео і знаходить на них об’єкти. Це дозволило наочно продемонструвати можливості комп’ютерного зору та його застосування в реальних задачах.

Під час розробки було здобуто нові знання в області комп’ютерного зору, зокрема у використанні моделей нейронних мереж для виявлення об’єктів.

Програма надає користувачеві зручний інтерфейс для вибору зображень або відео, а також дозволяє налаштовувати параметри порогу впевненості та «non_max_supression» методу, що використовуються при виявленні об'єктів за допомогою моделі YOLO. Після обробки зображення або відео моделлю YOLO, результати відображаються у вікні програми, що дозволяє користувачеві бачити виявлені об’єкти на зображенні або відео.

Таким чином, виконана робота не лише продемонструвала практичні аспекти комп’ютерного зору, але й дозволила глибше зрозуміти основні принципи та алгоритми, що використовуються в цій сфері.

СПИСОК ЛІТЕРАТУРИ

1. https://metinvest.digital/ua/page/1028 - Технології комп’ютерного зору
2. https://www.unite.ai/uk/what-is-computer-vision/ - Що таке комп’ютерний зір?
3. https://www.python.org/ - Python
4. https://www.jetbrains.com/ru-ru/pycharm/ - PyCharm Python
5. https://learn.microsoft.com/ru-ru/dotnet/machine-learning/tutorials/object-detection-onnx - Посібник. Виявлення об'єктів за допомогою YOLO
6. https://docs.ultralytics.com/ru/models/yolov3/ - YOLOv3
7. https://docs.ultralytics.com/usage/python/#using-trainers - YOLO Python Usage
8. https://pjreddie.com/darknet/yolo/ - YOLO: Real-Time Object Detection
9. https://learn.microsoft.com/ru-ru/windows/ai/windows-ml/tutorials/tensorflow-train-model - Навчання моделі виявлення об'єктів за допомогою TensorFlow
10. http://journals.khnu.km.ua/vestnik/wp-content/uploads/2023/07/vknu-ts-2023-n3321-17-23.pdf - Аналіз сучасних алгоритмів виявлення і розпізнавання об’єктів
11. https://habr.com/ru/articles/556404/ - YOLOv3 на Tensorflow 2.0
12. https://docs.ultralytics.com/ru/guides/yolo-performance-metrics/#object-detection-metrics - Глибоке занурення у метрики продуктивності
13. https://habr.com/ru/articles/678706/ - Пошук об'єктів на відео за допомогою Python
14. https://pypi.org/project/opencv-python/ - OpenCV on Python
15. http://surl.li/tssra - Yolo NMS Method
16. https://www.kaggle.com/datasets/shivam316/yolov3-weights - Yolov3 Weights
17. https://encord.com/glossary/true-positive-rate-definition/ - Істинно позитивний показник (TPR)

Додаток А

Код програми

Main.py

```
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter.ttk import Progressbar
from PIL import Image, ImageTk
import numpy as np
import cv2
import os
import threading

from yolo_utility import photo_detection, video_detection, filter_out, iou,
non_max_supression, rescale_box, draw_boxes

class YoloApp:
    def __init__(self, master, net):
        self.master = master
        self.net = net
        master.title("Аналізатор Зображень by Shapochka Vladislav")

        self.confidence_label = tk.Label(master, text="Поріг впевненності:")
        self.confidence_label.grid(row=0, column=0)
        self.confidence_entry = tk.Entry(master)
        self.confidence_entry.grid(row=0, column=1)
        self.confidence_entry.insert(0, "0.5")

        self.threshold_label = tk.Label(master, text="NMS Метод:")
        self.threshold_label.grid(row=1, column=0)
        self.threshold_entry = tk.Entry(master)
        self.threshold_entry.grid(row=1, column=1)
        self.threshold_entry.insert(0, "0.5")

        self.select_image_button = tk.Button(master, text="Вибір зображення",
command=self.select_image)
        self.select_image_button.grid(row=2, column=0)

        self.select_video_button = tk.Button(master, text="Вибір відео",
command=self.select_video)
        self.select_video_button.grid(row=2, column=1)

        self.analyze_button = tk.Button(master, text="Аналізувати",
command=self.process_data)
        self.analyze_button.grid(row=3, columnspan=2)

        self.status_label = tk.Label(master, text="")
        self.status_label.grid(row=5, columnspan=2)

        self.progress_bar = Progressbar(master, orient=tk.HORIZONTAL,
length=200, mode='determinate')
        self.progress_bar.grid(row=6, columnspan=2)

        self.image_label = tk.Label(master)
        self.image_label.grid(row=4, columnspan=2)
        self.image_width = 320
        self.image_height = 240
```

```
        self.data_path = None
        self.video_processing = False

    def select_image(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            if self.check_file_type(file_path, ['.jpg', '.jpeg', '.png']):
                self.data_path = file_path

                pil_image = Image.open(file_path)
                pil_image_resized = pil_image.resize((self.image_width,
self.image_height), Image.BILINEAR)
                image = ImageTk.PhotoImage(pil_image_resized)
                self.display_image(image)
            else:
                messagebox.showwarning("Попередження", "Обраний файл не є
зображенням.")

    def select_video(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            if self.check_file_type(file_path, ['.mp4', '.avi', '.mov']):
                self.data_path = file_path

                cap = cv2.VideoCapture(file_path)
                ret, frame = cap.read()
                if ret:

                    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                    pil_image = Image.fromarray(frame_rgb)
                    pil_image_resized = pil_image.resize((self.image_width,
self.image_height), Image.BILINEAR)
                    image = ImageTk.PhotoImage(pil_image_resized)
                    self.display_image(image)
                cap.release()
            else:
                messagebox.showwarning("Попередження", "Обраний файл не є
відео.")

    def check_file_type(self, file_path, allowed_extensions):
        _, ext = os.path.splitext(file_path)
        return ext.lower() in allowed_extensions

    def display_image(self, image):

        if hasattr(self, 'image_label'):
            self.image_label.destroy()
        self.image_label = tk.Label(self.master, image=image)
        self.image_label.grid(row=4, columnspan=2)
        self.image_label.image = image

    def process_data(self):
        confidence_str = self.confidence_entry.get()
        threshold_str = self.threshold_entry.get()

        try:
            confidence = float(confidence_str)
            threshold = float(threshold_str)

            if confidence < 0.1 or threshold < 0.1:
                messagebox.showwarning("Помилка", "Порогові значення мають
бути більше або дорівнювати 0,1.")
                return
        except ValueError:
```

```
            messagebox.showwarning("Помилка", "Порогові значення мають бути
числовими.")
            return

        if self.data_path:
            if self.check_file_type(self.data_path, ['.jpg', '.jpeg',
'.png']):
                image = cv2.imread(self.data_path)
                photo_detection(image, self.net, confidence, threshold,
LABELS, COLORS, self.data_path)
            elif self.check_file_type(self.data_path, ['.mp4', '.avi',
'.mov']):
                if not self.video_processing:
                    threading.Thread(target=self.process_video,
args=(self.data_path, confidence, threshold)).start()
                else:
                    messagebox.showwarning("Попередження", "Аналіз відео вже
виконується.")
        else:
            messagebox.showwarning("Попередження", "Спочатку виберіть
зображення або відео")

    def process_video(self, video_path, confidence, threshold):
        self.video_processing = True
        self.status_label.config(text="Аналіз відео...")
        total_frames = self.get_total_frames(video_path)
        self.progress_bar.config(maximum=total_frames, value=0)

        cap = cv2.VideoCapture(video_path)
        frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        fps = int(cap.get(cv2.CAP_PROP_FPS))

        result_video_name = 'Проаналізований відеофайл '
        out = cv2.VideoWriter(result_video_name +
os.path.basename(video_path), cv2.VideoWriter_fourcc(*'mp4v'), fps,
(frame_width, frame_height))

        frame_number = 0
        while True:
            ret, frame = cap.read()
            if not ret:
                break

            frame_number += 1

            (H, W) = frame.shape[:2]

            blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
swapRB=True, crop=False)
            self.net.setInput(blob)
            ln = self.net.getLayerNames()
            ln_out = [ln[i - 1] for i in self.net.getUnconnectedOutLayers()]
            layerOutputs = self.net.forward(ln_out)

            boxes = []
            scores = []
            classes = []
            for output in layerOutputs:
                (xywh_filterd, score_filtered, class_filtered) =
filter_out(output, confidence)
                boxes.append(xywh_filterd)
                scores.append(score_filtered)
                classes.append(class_filtered)
```

```
            boxes = np.vstack([r for r in boxes])
            scores = np.concatenate([r for r in scores], axis=None)
            classes = np.concatenate([r for r in classes], axis=None)

            boxes_coord = rescale_box(boxes, W, H)
            nms_idx = non_max_supression(boxes_coord, scores, confidence,
threshold)

            frame, _ = draw_boxes(frame, boxes_coord, nms_idx, scores,
classes, LABELS, COLORS)
            out.write(frame)

            self.update_progress(frame_number)

        cap.release()
        out.release()

        self.status_label.config(text="")
        self.progress_bar.config(value=0)
        self.video_processing = False

    def update_progress(self, value):
        self.progress_bar.config(value=value)

    def get_total_frames(self, video_path):
        cap = cv2.VideoCapture(video_path)
        total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        cap.release()
        return total_frames


# Load the neural network model
coco_names_file = os.path.join(os.path.dirname(__file__), 'yolo',
'coco.names')
yolov3_weight_file = os.path.join(os.path.dirname(__file__), 'yolo',
'yolov3.weights')
yolov3_config_file = os.path.join(os.path.dirname(__file__), 'yolo',
'yolov3.cfg')

LABELS = open(coco_names_file).read().strip().split("\n")

np.random.seed(45)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3), dtype="uint8")

net = cv2.dnn.readNetFromDarknet(yolov3_config_file, yolov3_weight_file)

root = tk.Tk()
app = YoloApp(root, net)
root.mainloop()
```

yolo_utility.py

```
import numpy as np
import cv2

def filter_out(layer_output, confidence):

    box_xywh = np.array(layer_output[:, :4])
    box_confidence = np.array(layer_output[:,
4]).reshape(layer_output.shape[0], 1)
    box_class_probs = np.array(layer_output[:, 5:])

    box_scores = box_confidence * box_class_probs
    box_class = np.argmax(box_scores, axis=-1)
    box_class_scores = np.max(box_scores, axis=-1)

    filtering_mask = box_class_scores >= confidence
    class_filtered = box_class[filtering_mask]
    score_filtered = box_class_scores[filtering_mask]
    xywh_filtered = box_xywh[np.nonzero(filtering_mask)]

    return (xywh_filtered, score_filtered, class_filtered)

def iou(box1, box2):

    xi1 = max(box1[0], box2[0])
    yi1 = max(box1[1], box2[1])
    xi2 = min(box1[2], box2[2])
    yi2 = min(box1[3], box2[3])

    inter_area = max(xi2 - xi1, 0) * max(yi2 - yi1, 0)

    box1_area = (box1[2] - box1[0]) * (box1[3] - box1[1])
    box2_area = (box2[2] - box2[0]) * (box2[3] - box2[1])

    union_area = box1_area + box2_area - inter_area


    iou = inter_area / union_area

    return iou

def non_max_supression(boxes, scores, confidence_threshold, iou_threshold):

    sorted_idx = np.argsort(scores)[::-1]

    remove = []
    for i in np.arange(len(scores)):

        if i in remove:
            continue

        if scores[sorted_idx[i]] < confidence_threshold:
            remove.append(i)
            continue

        for j in np.arange(i+1, len(scores)):
            if j in remove:
                continue

            if scores[sorted_idx[j]] < confidence_threshold:
                remove.append(j)
```

```
                continue

            overlap = iou(boxes[sorted_idx[i]], boxes[sorted_idx[j]])
            if overlap > iou_threshold:
                remove.append(j)

    sorted_idx = np.delete(sorted_idx, remove)
    return sorted(sorted_idx)

def rescale_box(boxes, width, height):

    boxes_orig = boxes * np.array([width, height, width, height])
    boxes_orig[:, 0] -= boxes_orig[:, 2] / 2
    boxes_orig[:, 1] -= boxes_orig[:, 3] / 2

    boxes_coord = boxes_orig
    boxes_coord[:, 2] = boxes_orig[:, 0] + boxes_orig[:, 2]
    boxes_coord[:, 3] = boxes_orig[:, 1] + boxes_orig[:, 3]

    return boxes_coord

def draw_boxes(image, boxes_coord, nms_idx, scores, classes, labels, colors):

    font = cv2.FONT_HERSHEY_SIMPLEX
    font_scale = 0.5
    border_thickness = 2
    text_thickness = 1

    text_all = []
    for i in nms_idx:
        color = tuple([int(c) for c in colors[classes[i]]])
        text = "{}: {:.4f}".format(labels[classes[i]], scores[i])
        text_all.append(text)

        (pt1_x, pt1_y) = (int(boxes_coord[i, 0]), int(boxes_coord[i, 1]))
        (pt2_x, pt2_y) = (int(boxes_coord[i, 2]), int(boxes_coord[i, 3]))
        cv2.rectangle(image, (pt1_x, pt1_y), (pt2_x, pt2_y), color,
border_thickness)

        (t_w, t_h), _ = cv2.getTextSize(text, font, fontScale=font_scale,
thickness=text_thickness)
        text_offset_x = 7
        text_offset_y = 7
        (text_box_x1, text_box_y1) = (pt1_x, pt1_y - (t_h + text_offset_y))
        (test_box_x2, text_box_y2) = ((pt1_x + t_w + text_offset_x), pt1_y)

        cv2.rectangle(image, (text_box_x1, text_box_y1), (test_box_x2,
text_box_y2), color, cv2.FILLED)

        cv2.putText(image, text, (pt1_x + text_offset_x, pt1_y - 5),
cv2.FONT_HERSHEY_SIMPLEX, font_scale,
                                    (255, 255, 255), text_thickness)

    return(image, text_all)

def photo_detection(image, net, confidence, threshold, labels, colors,
image_filename=None):
    (H, W) = image.shape[:2]

    blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True,
crop=False)

    net.setInput(blob)
```

```
    ln = net.getLayerNames()
    ln_out = [ln[i - 1] for i in net.getUnconnectedOutLayers()]

    layerOutputs = net.forward(ln_out)

    boxes = []
    scores = []
    classes = []
    for output in layerOutputs:
        (xywh_filtered, score_filtered, class_filtered) = filter_out(output,
confidence)
        boxes.append(xywh_filtered)
        scores.append(score_filtered)
        classes.append(class_filtered)

    boxes = np.vstack([r for r in boxes])
    scores = np.concatenate([r for r in scores], axis=None)
    classes = np.concatenate([r for r in classes], axis=None)

    boxes_coord = rescale_box(boxes, W, H)
    nms_idx = non_max_supression(boxes_coord, scores, confidence, threshold)

    image, text_list = draw_boxes(image, boxes_coord, nms_idx, scores,
classes, labels, colors)
    print("Result:", text_list)
    if image_filename:
        result_filename =
"{}_Result.jpg".format(image_filename.split('.')[0])
        cv2.imwrite(result_filename, image)

    cv2.imshow("Result", image)
    cv2.waitKey(0)

def video_detection(video_path, net, confidence, threshold, labels, colors):
    cap = cv2.VideoCapture(video_path)
    frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    fps = int(cap.get(cv2.CAP_PROP_FPS))

    result_video_name = 'output_video.avi'
    out = cv2.VideoWriter(result_video_name, cv2.VideoWriter_fourcc(*'mp4v'),
fps, (frame_width, frame_height))

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        (H, W) = frame.shape[:2]

        blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
swapRB=True, crop=False)
        net.setInput(blob)
        ln = net.getLayerNames()
        ln_out = [ln[i - 1] for i in net.getUnconnectedOutLayers()]
        layerOutputs = net.forward(ln_out)

        boxes = []
        scores = []
        classes = []
        for output in layerOutputs:
            (xywh_filterd, score_filtered, class_filtered) =
filter_out(output, confidence)
            boxes.append(xywh_filterd)
```

```
            scores.append(score_filtered)
            classes.append(class_filtered)

        boxes = np.vstack([r for r in boxes])
        scores = np.concatenate([r for r in scores], axis=None)
        classes = np.concatenate([r for r in classes], axis=None)

        boxes_coord = rescale_box(boxes, W, H)
        nms_idx = non_max_supression(boxes_coord, scores, confidence,
threshold)

        frame, _ = draw_boxes(frame, boxes_coord, nms_idx, scores, classes,
labels, colors)
        out.write(frame)

    cap.release()
    out.release()
    cv2.destroyAllWindows()
```