МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о.завідувача кафедри
_____ Сафонова С.О.
«____»_____ 2020 р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження методу автоматичного визначення об'єктів у відеопотоці в режимі реального часу

Освітньо-кваліфікаційний рівень "Магістр"

Спеціальність 123 – "Комп'ютерна інженерія"

Науковий керівник роботи:	_____ (підпис)	Білобородова Т.О.. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я.О. (ініціали, прізвище)
Студент:	_____ (підпис)	Приймак С.О. (ініціали, прізвище)
Група:		КІ-18дм

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень Магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 123 – "Комп'ютерна інженерія"
(шифр і назва)

ЗАТВЕРДЖУЮ:
Т.в.о. завідувача кафедри
_____С.О. Сафонова

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Приймаку Сергію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методу автоматичного визначення об'єктів у відеопотоці в режимі реального часу
керівник проекту (роботи) к.т.н. Білобородова Тетяна Олександрівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "11" 10 2019р. № 135/15.15
2. Строк подання студентом роботи 08.01.2020р.
3. Вихідні дані до роботи Матеріали науково-дослідної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) а) аналіз галузі застосування та методів розмітки відео та визначення об'єктів у відеопотоці в режимі реального часу;
б) дослідження сучасних методів, технологій, засобів розмітки об'єктів у відеопотоці, використання розмічених даних для автоматичного визначення об'єктів у відеопотоці в режимі реального часу.
в) практична реалізація повного циклу автоматичного визначення об'єктів у відеопотоці в режимі реального часу.
г) охорона праці в галузі.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): електронні плакати.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст. викл. Критська Я.О.		

7. Дата видачі завдання 11 жовтня 2019р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд та аналіз вимог до роботи.	02.09.19-19.09.19	
2	Дослідження сучасних методів, технологій, засобів анотування об’єктів у відеопотоці та використання розмічених даних для автоматичного визначення об'єктів у відеопотоці в режимі реального часу	22.09.19-10.10.19	
3	Практична реалізація повного циклу автоматичного визначення об'єктів у відеопотоці в режимі реального часу	11.10.19-15.11.19	
4	Дослідження та аналіз отриманих результатів	16.11.18-09.12.19	
5	Розробка заходів з охороні праці.	10.12.19-15.12.19	
6	Оформлення пояснювальної записки.	16.12.19-31.12.19	
7	Підготовка та подання магістерської роботи до захисту.	02.01.20-08.01.20	

Студент _____ Приймак С.О.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Білобородова Т. О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Приймак С.О. Дослідження методу автоматичного визначення об'єктів у відеопотоці в режимі реального часу.

В роботі представлено вирішення задачі підвищення ефективності методу автоматичного визначення об’єктів у відеопотоці в режимі реального часу в умовах відсутності розмічених даних та складності сцени відеопотоку, за рахунок удосконалення систематизованої сукупності етапів обробки шляхом анотування відеозображень, створення моделі розпізнавання на підставі анотованих даних, дослідження методів і архітектур адаптивного опрацювання відеопотоків, спрямованих на інтелектуальну обробку даних. Проведено аналіз методів і технологій анотування відео та визначення об'єктів у відеопотоці. Визначено засіб анотування об’єктів у відеопотоці, метод подальшого автоматичного вилучення і використання цифрових даних відеопотоку в режимі реального часу. Досліджено процес анотування відеозображень з використанням технології глибокого навчання та нейронних мереж. Представлена практична реалізація наступних етапів: анотування даних відеопотоку, розробка програмного засобу для адаптації моделі даних з метою її подальшого використання, розробка мобільного додатку для тестування моделі в режимі реального часу. Виконано розробку та тестування моделі для визначення об'єктів у відеопотоці в режимі реального часу. Ефективність методу автоматичного визначення об’єктів у відеопотоці в режимі реального часу визначена з використанням середньої оцінки точності моделі.

Рис.: 23 Табл. 3 Бібліогр.: 47.

Ключові слова: розпізнавання відео, визначення об’єктів відеозображень, глибоке навчання, нейронні мережі, модель, анотування відеозображень, обмежуюча рамка.

АННОТАЦИЯ

Приймак С.А. Исследование метода автоматического определения объектов в видеопотоке в режиме реального времени.

В работе представлены решения задачи повышения эффективности метода автоматического определения объектов в видеопотоке в режиме реального времени в условиях отсутствия размеченных данных и сложности сцены видеопотока, за счет совершенствования систематизированной совокупности этапов обработки путем

аннотирования видеоизображений, создания модели распознавания на основании аннотированных данных, исследования методов и архитектур адаптивной обработки видеопотоков, направленных на интеллектуальную обработку данных. Проведен анализ методов и технологий аннотирования видео и определения объектов в видеопотоке. Определены средство аннотирования объектов в видеопотоке, метод дальнейшего автоматического извлечения и использования цифровых данных видеопотока в режиме реального времени. Исследован процесс аннотирования видеоизображений с использованием технологии глубокого обучения и нейронных сетей. Представлена практическая реализация следующих этапов: аннотирование данных видеопотока, разработка программного средства для адаптации модели данных для дальнейшего использования, разработка мобильного приложения для тестирования модели в режиме реального времени. Выполнено разработку и тестирование модели для определения объектов в видеопотоке в режиме реального времени. Эффективность метода автоматического определения объектов в видеопотоке в режиме реального времени определена с использованием средней оценки точности модели.

Рис.: 23. Табл.: 3. Библиогр.: 47.

Ключевые слова: распознавания видео, определение объектов видеоизображений, глубокое обучение, нейронные сети, модель, аннотирование видеоизображений, ограничивающая рамка.

ABSTRACT

Pryimak Serhii. Research on automatic real-time video recognition method.

The theses presents solutions to the problem of increasing the efficiency of the method of automatically real-time detection of objects in a video stream in the absence of tagged data and the complexity of the scene of the video stream, by improving the systematic set of processing steps by annotating video images, creating a recognition model based on annotated data, researching methods and adaptive video stream processing architectures aimed at data mining. The analysis of methods and technologies for annotating video and identifying objects in a video stream is carried out. A means of annotating objects in a video stream, a method for further automatic extraction and use of digital data of a video stream in real time are determined. The process of annotating video images using deep learning technology and neural networks is investigated. The

practical implementation of the following stages is presented: annotating the video stream data, developing a software tool for adapting the data model for future use and a mobile application for testing the model in real time. Implemented the development and testing of a model for determining objects in a video stream in real time. The effectiveness of the method of automatic detection of objects in the video stream in real time is determined using an average estimate of the accuracy of the model.

Keywords: video recognition, object detection, deep learning, neural networks, model, video annotation, bounding box.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК І СКОРОЧЕНЬ

BBox	Bounding box
CNN	Convoluted neural network
CVAT	Computer Vision Annotation Tool
DCNN	Deep convoluted neural networks
FPN	Feature Pyramid Networks
FPS	Frames per second
FSAF	Feature-selection-anchor-free
HTTP	Hypertext Transfer Protocol
GPU	Graphics Processing Unit
RCNN	Region-based convoluted neural networks
ROI	Region of interest
RPN	Region Proposal Network
SHG	Stochastic gradient descent
SPP	Spatial Pyramid Pooling
SVM	Support-vector machine
YOLO	You Only Look Once
WSGI	Web Server Gateway Interface

ВСТУП

В останні роки спостерігається значний прогрес в технології визначення об'єктів у відеопотоці. Сучасні методи мають двоетапну структуру. Глибокі згорткові нейронні мережі спочатку використовуються для анотации характеристик вхідного зображення. На другому етапі реалізується визначення об'єктів зображень і отримання результатів оцінки якості їх виявлення. Ці методи досягають чудових результатів в нерухомих зображеннях. Однак існують деякі особливості їх застосування безпосередньо для виявлення відео об'єктів. Точність розпізнавання страждає від погіршення якості зображення об'єктів в відео, таких як розмиття руху, розфокусування відео, рідкісні пози і т. п., що рідко спостерігається на нерухомих зображеннях.

Проте, відео містить корисну інформацію про один і той ж об'єкт, що зазвичай спостерігається в декількох кадрах за короткий час. Ця тимчасова інформація використовується в існуючих способах визначення відеооб'єктів. Існуючі методи використовують детектори об'єктів в окремих кадрах, а потім збирають виявлені обмежуючи рамки з тимчасового виміру на виділеному етапі постобробки. Цей етап грунтується на оцінці руху об'єкта і методах відстеження обмежуючої рамки. Ці методи використовують окремі кадри для виявлення об'єктів, виключаючи кадри недостатньо високої якості.

Незважаючи на вагомий розвиток технології розпізнавання відео, все ще не вистачає всебічного підходу ключових конструкцій, фундаментальних принципів та методів визначення об’єктів у відеопотоці, зокрема, в режимі реального часу.

Тому обґрунтованою є тема магістерської роботи, у якій вирішується **науково-прикладне завдання** удосконалення методу автоматичного визначення об’єктів у відеопотоці в режимі реального часу.

Об'єкт дослідження: процеси вилучення та використання цифрових даних відеопотоку.

Предмет дослідження: систематизована сукупність етапів автоматичного визначення об’єктів у відеопотоці в режимі реального часу.

Мета і завдання дослідження. Метою дослідження є підвищення точності автоматичного визначення об’єктів у відеопотоці в режимі реального часу в умовах

відсутності анотованих даних та складності сцени відеопотоку шляхом анотування об’єктів відеопотоку, створення моделі розпізнавання на підставі розмічених даних, дослідження методів і архітектур адаптивного опрацювання відеопотоків, спрямованих на інтелектуальну обробку даних.

Для досягнення мети дослідження необхідно вирішити такі **завдання**:

– аналіз методів і технологій анотування відео та визначення об'єктів у відеопотоці в режимі реального часу;

– вибір засобу анотування об’єктів у відеопотоці, методу подальшого автоматичного вилучення і використання цифрових даних відеопотоку в режимі реального часу;

– анотування даних відеопотоку;

– розробка програмного засобу для адаптації моделі даних відеопотоку з метою її подальшого використання;

– розробка мобільного додатку для тестування моделі в режимі реального часу;

– розробка та тестування моделі визначення об’єктів шляхом проведення експерименту розмітки даних та визначення об'єктів у відеопотоці в режимі реального часу;

– оцінка ефективності методу автоматичного визначення об’єктів у відеопотоці в режимі реального часу.

Методи дослідження. Проведені в роботі дослідження основані на методах моделювання зображень, технології глибокого навчання, нейронних мережах, що використовувались для анотування та визначення об’єктів відеопотоку. Перевірка результатів дослідження ґрунтувалась на методах експерименту та порівняння, які використовувались при розробленні практичної частини дипломного проекту.

Особистий внесок здобувача полягає у розробці моделі на підставі даних анотації складної сцени відеопотоку, розробці програмного засобу для адаптації моделі даних відеопотоку з метою її подальшого використання, розробці мобільного додатку для тестування моделі в режимі реального часу, адаптації програмних засобів, що дозволяють вирішити поставлені задачі. Усі основні результати отримані автором особисто.

Апробація матеріалів магістерської роботи. Основні положення, ідеї, та висновки магістерської роботи доповідалися та обговорювалися на V регіональному форумі IT-Ідея (м. Сєвєродонецьк, 2019).

Практичне значення отриманих результатів полягає в тому, що основні наукові положення реалізовані у виді систематизованої сукупності етапів автоматичного визначення об'єктів у відеопотоці, моделі та програмних засобів, що утворюють метод автоматичного визначення об'єктів у відеопотоці в режимі реального часу.

Публікації. За темою магістерської роботи з викладенням її результатів опублікована наукова праця у науковому фаховому виданні України; одні тези доповідей всеукраїнської конференції.

Структура та обсяг магістерської роботи. Кваліфікаційна магістерська робота складається із вступу, чотирьох розділів, висновків, переліку посилань. Загальний обсяг складає 101 сторінку, з яких основний текст на 78 сторінках, список використаних джерел із 47 найменувань на 4 сторінках та 3 додатків на 19 сторінок. Робота містить 3 таблиці, 23 рисунки.

РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ ВИЗНАЧЕННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Визначення об'єктів - це метод комп'ютерного зору, який дозволяє ідентифікувати і визначати місце розташування об'єктів на зображенні або відео [1]. При такому типі ідентифікації та локалізації виявлення об'єктів може використовуватися для підрахунку об'єктів в сцені, а також для визначення і відстеження їх точного місця розташування, в той же час точно маркіруючи їх. Зокрема, виявлення об'єктів малює обмежують рамки навколо цих виявлених об'єктів, які дозволяють нам визначити, де ці об'єкти знаходяться (або як вони переміщаються) в цій сцені.

Визначення об'єктів зазвичай плутають з розпізнаванням зображень, тому, важливо прояснити відмінності між ними.

При розпізнаванні зображень призначається мітка для зображення. Зображення людини отримує мітку «людина». Зображення двох людей отримує ярлик «людина». Визначення об'єкта, з іншого боку, рисує обмежуючу рамку навколо кожної людини і позначає обмежуючу рамку «людина». Модель передбачає, де знаходиться кожен об'єкт і яку мітку слід застосовувати. Таким чином, визначення об'єкта надає більше інформації про зображення, ніж розпізнавання.

1.1 Режими і типи визначення об'єктів

У визначення об'єктів можна виділити підходи, засновані на методах машинного навчання та засновані на методах глибокого навчання.

У більш традиційних підходах, заснованих на машинному навчанні, методи комп'ютерного зору використовуються для перегляду різних характеристик зображення, таких як колірна гістограма або межі, щоб ідентифікувати групи пікселів, які можуть належати об'єкту. Ці функції потім передаються в регресійну модель, яка передбачає розташування об'єкта разом з його міткою.

З іншого боку, підходи, засновані на глибокому навчанні, використовують згорткові нейронні мережі (CNN) для виконання наскрізного виявлення об'єктів без учителя, в якому функції не потрібно визначати і витягувати окремо.

Оскільки методи глибокого навчання наразі є найсучаснішими підходами до визначення об'єктів, саме цим методам приділена увага даного дослідження.

Визначення об'єктів нерозривно пов'язане з іншими подібними методами комп'ютерного зору, такими як розпізнавання зображень і сегментація зображень, в тому сенсі, що це допомагає розуміти і аналізувати сцени на зображеннях або відео.

Але є важливі відмінності. При розпізнаванні зображень виводиться тільки мітка класу для ідентифікованого об'єкта, а сегментація зображення створює розуміння елементів сцени на рівні пікселів. Визначення об'єкта від цих інших завдань відрізняє здатність знаходити об'єкти в зображенні або відео, що дозволяє відстежувати ці об'єкти у відеопотоці.

З огляду на ці ключові відмінності і можливості визначення об'єктів, цей метод можна застосовувати для вирішення наступних задач:

- підрахунок кількості людей в натовпі;
- в самокерованих автомобілях;
- для відеоспостереження;
- для виявлення особи;
- для виявлення аномалій.

Визначення об’єктів застосовується для вирішення цих задач. Але не обмежується ними.

Якісний алгоритм визначення об’єктів повинен розуміти смислові підказки, а також просторову інформацію про зображення. Фактично, визначення об'єктів є основним кроком на шляху до багатьох застосувань комп'ютерного зору, таких як розпізнавання обличь, виявлення пішоходів, відеоаналіз, виявлення логотипів, тощо.

1.2 Базова структура визначення об'єктів

Моделі визначення об'єктів на основі глибокого навчання зазвичай складаються з двох частин. Енкодер приймає зображення в якості вхідних даних і запускає його через серію блоків і шари, які навчаються витягувати статистичні функції, які використовуються для виявлення і об'єктів етикеток. Виходи з енкодера потім передаються в декодер, який передбачає обмежують рамки і мітки для кожного об'єкта.

Енкодер можна представити наступним чином [2] (див. рис. 1.1).

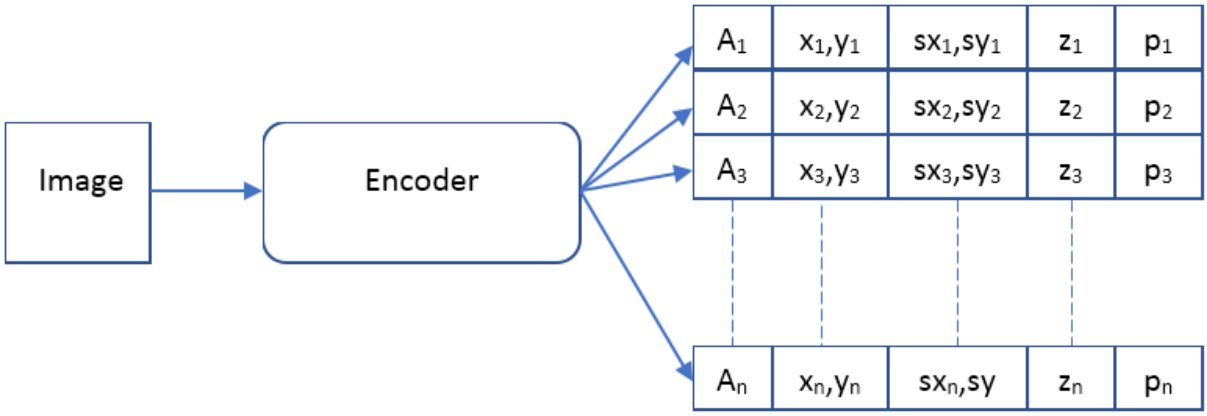

Рисунок 1.1 – Енкодер нейронної мережі

Де *A* представляє деякий вектор значень, що описує об'єкт, *x,y* - координати об'єкту, *sx, sy* - масштаб об'єкту, *z* - положення об'єкту в глибину, *p* - ймовірність того, що об'єкт присутній.

Тобто нейронна мережа на вхід отримує зображення заздалегідь визначеного розміру на якій ми хочемо знайти об'єкти і видає масив описів. Для визначення декількох об’єктів на одному зображенні необхідно використовувати рекурентні нейронні мережі.

Декодер можна визначити наступним чином, як це представлено на рис. 1.2.

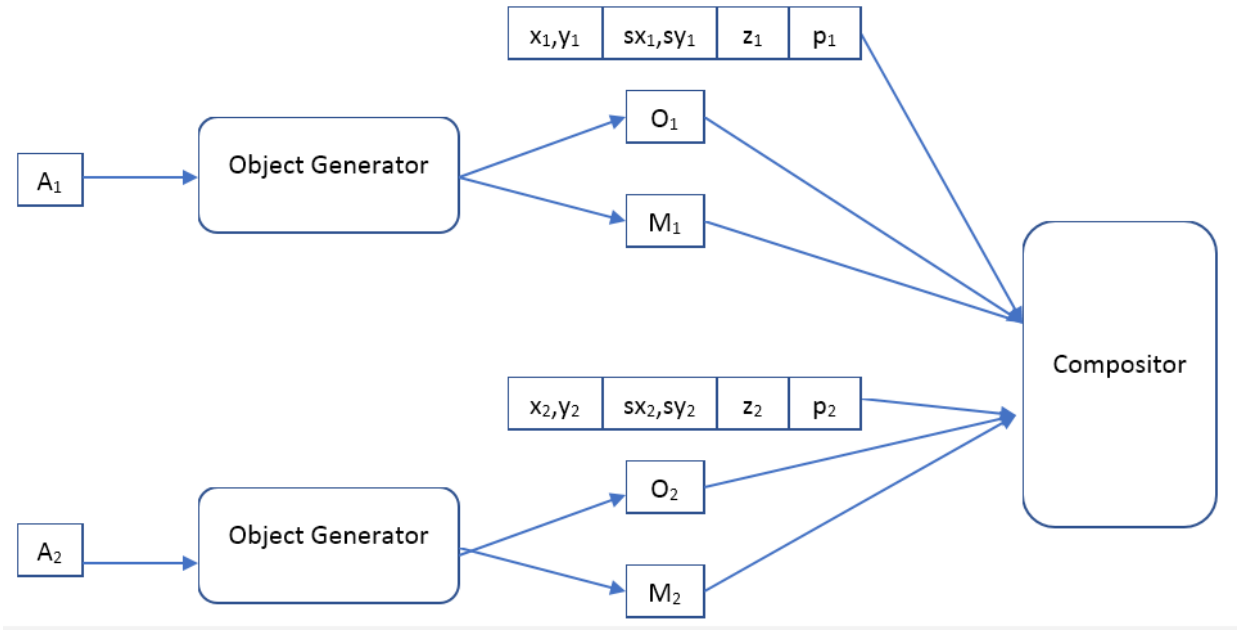

Рисунок 1.2 – Декодер нейронної мережі

Де Object Generator – це нейронна мережа, яка на вхід отримує вектор опису об'єкта і видає - зображення (якогось стандартного розміру) об'єкта і маску непрозорих пікселів (opacity mask). Compositor - отримує на вхід зображення всіх об'єктів, маску, положення, масштаб, глибину і формує вихідне зображення, яке повинно бути схоже на початкове.

1.3 Етапи визначення об'єктів

Конвеєр визначення об'єктів поділений на три етапи: 1) генерація пропозицій; 2) виявлення векторів властивостей (feature vector extraction); та 3) класифікація регіонів. Під час генерації пропозицій мета полягає в пошуку місць на зображенні, які можуть містити об'єкти. Ці місця називають також регіонами, що цікавлять (regions of interest або *ROI*). Інтуїтивна ідея полягає у скануванні всього зображення за допомогою плаваючих вікон (sliding windows). Для того, щоб виявити інформацію про об’єкти що мають різні розміри та співвідношення сторін, розмір вхідних зображень змінюються в різні масштаби, та багатовимірні вікна використовувались для просування по цим зображенням. Під час другого кроку на кожній позиції зображення із вікна виводиться вектор фіксованої довжини з властивостями (features) для збору семантичної інформації охопленої області. Нарешті, на третьому кроці за допомогою заздалегідь навченим класифікаторам регіонам зображення

присвоюються мітки (labels). Зазвичай, тут використовувались метод опорних векторів (Support-vector machine або *SVM*) завдяки їх хорошій продуктивності на невеликих обсягах даних. Крім того, деякі методи класифікації, такі як бегінг, каскадне навчання та AdaBoost, використовуються на етапі класифікації регіону, що призводить до подальших поліпшень точності виявлення.

Найбільш успішні традиційні методи виявлення об’єктів зосереджені на ретельному проектуванні детекторів властивостей (feature detectors). Детектор властивостей - це алгоритм, який приймає зображення і видає місця (тобто координати пікселів) важливих областей цього зображення. Прикладом цього є детектор кутів, який видає місця розташування кутів на зображенні, але не повідомляє іншої інформації про виявлені властивості.

Незважаючи на те що такий підхід виявлення об’єктів працює, він має значні обмеження при зростанні кількості даних. Найбільш помітними з обмежень є такі: 1) на першому етапі генерується величезна кількість пропозицій, значна частина яких є надлишковими, що призводить до великої кількості помилкових позитивних результатів під час класифікації; 2) дескриптори властивостей (feature descriptors) виготовлені вручну на основі низьких візуальних рівнів [3, 4, 5], що ускладнювало виявлення репрезентативної смислової інформації у складних контекстах 3) кожен крок конвеєру для виявлення об’єктів був розроблений та оптимізований окремо, і таким чином не вдається отримати глобальне оптимальне рішення для всієї системи. Після успіху в застосуванні глибоких згорткових нейронних мереж (deep convoluted neural networks або *DCNN*) для класифікації зображень [5, 6] виявлення об'єктів також досягло значного прогресу на основі технологій глибинного навчання [7, 8]. Однак раннім спробам не вистачало ефективної методики оптимізації для контрольованого навчання. Пізніше згорткові нейронні мережі було оптимізовано за допомогою стохастичного градієнтного спуску (stochastic gradient descent або *SGD*), що показало значне зростання продуктивності при упізнаванні цифр. Але для дійсно добрих результатів у упізнанні об’єктів глибокі згорткові нейронні мережі вимагають для навчання великого набора розмічених даних та значних обчислювальних ресурсів. Чим більший набір для навчання та чим точніше він розмічений, тим точніші результати упізнавання об’єктів. З розвитком систем паралельного обчислювання, таких як кластери GPU, стало можливо навчати нейронні мережі на дійсно великих наборах даних за значно швидший

час. Це допомогло в адаптації DCNN для класифікації методики глибокого навчання до інших завдань комп’ютерного зору і показати перспективні результати порівняно з традиційними методами.

На відміну від виготовлених вручну дескрипторів, що використовуються в традиційних детекторах, глибокі згорткові нейронні мережі генерують ієрархічні представлення характеристик із сирих пікселів до семантичної інформації високого рівня, яка автоматично засвоюється з навчального набору даних та демонструє більшу дискримінаційну здатність до вираження в складних контекстах. Крім того, користуючись потужними можливостями навчання, глибока згорткова нейронна мережа може отримати кращу представленість функцій з більшим набором даних, тоді як навчальна здатність традиційних візуальних дескрипторів є фіксованою, і не може покращитись, коли стане більше даних. Ці властивості дозволили розробити оптимізовані алгоритми виявлення об'єктів на основі глибоких згорткових нейронних мереж з більш потужними можливостями представлення характеристик (feature representation capability).

В даний час фреймворки для виявлення об'єктів, засновані на глибокому навчанні, можна в основному розділити на дві сім'ї: 1) двоступеневі детектори, такі як CNN на основі регіону (Region-based CNN або *R-CNN*) [8] та його варіанти [7, 9, 10]; та 2) одноступеневі детектори, такі як YOLO [11] та його варіанти [12, 13]. Двоступеневі детектори спочатку використовують генератор пропозицій для створення розрізненого набору пропозицій та виявлення характеристик з кожної пропозиції, а потім класифікатори регіонів, які прогнозують категорію запропонованого регіону. Одноступеневі детектори безпосередньо роблять категоричне передбачення об'єктів на кожній ділянці карт характеристик (feature map) без кроку каскадної класифікації. Двоступеневі детектори зазвичай досягають кращої продуктивності виявлення та показують найкращі результати за загальнодоступними тестами, тоді як одноступінчасті детектори значно ефективніші за часом та мають більшу придатність до упізнавання об'єктів у режимі реального часу.

Виявлення об'єктів передбачає завдання розпізнавання (наприклад, "класифікація об'єктів") та локалізацію (наприклад, "регресія місця"). Детектору об'єктів необхідно відрізняти об'єкти певних цільових класів від фонових зображень із точною локалізацією та правильним прогнозуванням категоричних позначень для кожного екземпляра об'єкта.

Обмежувальні поля або піксельні маски прогнозуються для локалізації цих екземплярів цільового об'єкта.

На час оцінки використовується метрика, яка називається перетином-над-союзом або коефіцієнтом Жаккара [14] (intersection over union або *IoU*) між об'єктами та передбаченнями для оцінки якості локалізації:

$$\mathrm{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{1.1}$$

Пороговий рівень *IoU* Ω встановлюється, щоб визначити, чи прогноз щільно покриває об’єкт, тобто IoU ≥ Ω. Прогнозування з правильною категоричною міткою, чи успішне прогнозування локалізації (відповідає критеріям IoU) вважається позитивним, інакше це негативне прогнозування.

Для виявлення об'єкта прогноз з правильною категорією мітки, а також успішне прогнозування локалізації (відповідає критеріям IoU) вважається позитивним, інакше це негативний прогноз.

Для загальної оцінки проблеми виявлення об'єктів для оцінки використовується середня арифметична точність (mean average *mAP*) для класів *C*, а в реальних сценаріях, таких як виявлення пішоходів, використовуються різні показники оцінювання. Крім точності виявлення, швидкість виведення є також важливою метрикою для оцінки алгоритмів виявлення об'єктів. Зокрема, якщо ми хочемо виявити об’єкти у відеопотоці (виявлення в режимі реального часу), обов'язково має бути детектор, який може швидко обробляти цю інформацію. Таким чином, ефективність детектора також оцінюється у кадрах в секунду (frames per second, *FPS*), тобто, скільки зображень він може обробити за секунду. Зазвичай детектор, який може досягти швидкості виводу в 20 FPS, вважається детектором у реальному часі.

1.4 Налаштування визначення

У визначення об'єктів є два параметри: 1) чисте виявлення об'єктів (локалізація на рівні прямокутника) та 2) сегментація примірників (локалізація на рівні пікселів або масок). Чисте виявлення об’єктів є найбільш вивченим та вважається традиційним налаштуванням

виявлення, де ціллю є виявлення об’єктів за прямокутними боксами. Алгоритми чистого виявлення об’єктів вимагають тільки анотацію bbox, а при оцінці для вимірювання продуктивності обчислюється *IoU* між передбачуваним обмежувальним полем із ground truth. Сегментація екземплярів - це відносно нова установка, яка базується на традиційних налаштуваннях виявлення. Сегментація екземпляра вимагає сегментувати кожен об'єкт за допомогою піксельної маски замість грубої рамки з обмежувальним прямокутником.

Завдяки більш точному прогнозуванню на рівні пікселів, сегментація екземплярів є більш чутливою до просторової нерівності і, отже, має більш високі вимоги до обробки просторової інформації. Метрика оцінки сегментації екземплярів майже ідентична виявленню рівня bbox, за винятком того, що обчислення *IoU* проводиться за прогнозами масок. Незважаючи на те, що два налаштування виявлення дещо відрізняються, основні компоненти, здебільшого можуть бути розділені між цими двома налаштуваннями

1.5 Парадигми визначення об’єктів

Сучасні об'єкти-детектори з глибоким навчанням можна в основному поділити на дві основні категорії: двоступеневі детектори та одноступінчасті детектори. Для двоступеневого детектора на першому етапі формується розріджений набір пропозицій; а на другому етапі функціональні вектори згенерованих пропозицій кодуються глибокими згортковими нейронними мережами з подальшим складанням передбачень класу об’єкту. Одноступеневий детектор не має окремої стадії для формування пропозиції (або вивчення генерації пропозицій). Вони, як правило, розглядають усі позиції на зображенні як потенційні об'єкти та намагаються класифікувати кожну цікаву область як фоновий або цільовий об'єкт. Двоступеневі детектори часто показували найкращі результати на багатьох наборах даних загальнодоступних еталонів. Однак вони, як правило, неефективні з точки зору меншої швидкості впізнавання. Одноступінчасті детектори набагато швидші та придатніші для додатків виявлення об'єктів у реальному часі, але мають дещо нижчу якість впізнавання порівняно з двоступеневими детекторами.

1.5.1 Двоступеневі детектори

Як було зазначено раніше двоступеневі детектори розділяють завдання виявлення на два етапи: 1) створення пропозицій; та 2) робити прогнози на ці пропозиції. Під час фази формування пропозиції детектор намагатиметься визначити області на зображенні, які потенційно можуть бути об'єктами. Ідея полягає у тому, щоб запропонувати регіони з високою згадуваністю, такі, що всі об'єкти на зображенні належать принаймні до одного із запропонованих регіонів. На другому етапі використовується модель, заснована на глибокому навчанні, для класифікації цих пропозицій за правильними категоричними позначками. Область може бути або фоном, або об'єктом однієї з попередньо визначених міток класу. Крім того, модель може уточнити оригінальну локалізацію, запропоновану генератором пропозицій.

R-CNN [8] – це новаторський двоступеневий детектор об'єктів розроблений у 2014р. Порівняно з попередніми найсучаснішими методами, заснованими на традиційній системі виявлення R-CNN значно покращив продуктивність виявлення. Конвеєр R-CNN можна розділити на три складові: 1) створення пропозиції, 2) визначення та 3) класифікація регіону. Для кожного зображення R-CNN створює розрізнений набір пропозицій (близько 2000 пропозицій) за допомогою селективного пошуку [15], який призначений для відхилення регіонів, які легко ідентифікуються як фонові регіони. Потім кожна пропозиція обрізається та розміщується в області фіксованого розміру і кодується глибокою згортковою нейронною мережею в (наприклад, 4 096 розмірний) вектор характеристик, а потім класифікатором SVM. Нарешті, регресори bbox навчаються, використовуючи очікувані характеристики як вхідні дані для того, щоб оригінальні пропозиції щільно охопили об'єкти. Порівняно з традиційними рукописними дескрипторами функцій, глибокі нейронні мережі генерують ієрархічні характеристики та фіксують різну масштабну інформацію в різних шарах і, нарешті, створюють надійні та дискримінаційні риси класифікації. Крім того R-CNN відхиляють велику кількість хибних результатів що легко детектуються ще до початку навчання, що значно покращує швидкість та зменшує загальну кількість хибних результатів.

Однак R-CNN має деякі критичні недоліки: 1) характеристики кожної пропозиції визначаються глибокими згортковими мережами окремо (тобто обчислення не є

спільними), що призводить до сильного дублювання обчислень і робить R-CNN надзвичайно довгими для навчання та тестування; 2) три етапи R-CNN (генерування пропозицій, вилучення характеристик та класифікація регіонів) є незалежними компонентами, і фреймворк виявлення неможливо було оптимізувати як одне ціле, що ускладнює отримання глобального оптимального рішення; та 3) вибірковий пошук спирається на візуальні підказки низького рівня і, таким чином, намагається генерувати високоякісні пропозиції у складних контекстах. Більше того, воно не в змозі користуватися перевагами прискорення обчислення на GPU. Для рішення цих проблем було запропоновано [16, 17] прискорити R-CNN за допомогою просторового об’єднання [18] пірамід (Spatial Pyramid Pooling, *SPP*). Замість обрізання пропонованих областей та подачі в модель CNN окремо, SPP-net обчислює карту характеристик з усього зображення за допомогою глибокої згорткової мережі та витягує вектори характеристик фіксованої довжини на мапі характеристик за допомогою шару просторового об'єднання пірамід (SPP) . SPP розділяє функціональну карту на сітці N × N для декількох значень N (таким чином дозволяючи отримувати інформацію в різних масштабах) та виконує об'єднання в кожній комірці сітки для надання вектора характеристик. Вектори характеристик, отримані з кожної сітки N × N, об'єднуються, щоб представити регіон. Витягнуті функції подаються в регіональні класифікатори SVM та регресори bbox. На відміну від RCNN, шар SPP також може працювати над зображеннями / регіонами в різних масштабах і співвідношеннях сторін, не змінюючи їх розмір. Таким чином, він не зазнає втрат інформації та небажаних геометричних спотворень.

SPP-net досягає кращих результатів і має значно більшу швидкість визначення порівняно з R-CNN. Однак навчання SPP-мережі все ще є багатоступеневим, і тому його не можна оптимізувати від кінця до кінця, також вони потребують додаткову пам'ять кешу для зберігання витягнутих характеристик. Ці проблеми вирішує Fast R-CNN [9]. Fast R-CNN (як SPP-Net) також обчислює карту характеристик для всього зображення та витягує характеристики регіону фіксованої довжини на карті характеристик. На відміну від SPP-net, Fast R-CNN використовує шар об'єднання ROI для отримання характеристик регіону. Шар об'єднання ROI - це особливий випадок SPP, який приймає лише одну шкалу (тобто, лише одне значення N для сітки N × N) для розподілу пропозиції на фіксовану кількість поділів, а також зворотно розповсюджує сигнали помилок на ядра згортки. Після вилучення

характеристик, вектори характеристик подаються в послідовність повністю з’єднаних шарів перед двома сусідніми вихідними шарами: класифікаційним шаром (cls) та регресійним шаром (reg). Класифікаційний шар відповідає за генерування softmax ймовірностей для класів C + 1 (C класів плюс один клас фону), тоді як шар регресії кодував 4 параметри для уточнення границь. У швидкому RCNN всі етапи вилучення, класифікація регіону та етапи регресії обмежувального прямокутника можуть бути оптимізовані від кінця до кінця, без додаткового кешу для зберігання характеристик (на відміну від SPP Net). Швидкий R-CNN досягає значно кращої точності виявлення, ніж R-CNN та SPP-net, і має кращу швидкість навчання та визначення.

1.5.2 Одноступеневі детектори

Відмінні від двоступеневих алгоритмів виявлення, які ділять конвеєр виявлення на дві частини: генерація пропозицій та класифікація регіону; одноступеневі детектори не мають окремої стадії для генерації пропозицій (або навчання генерації пропозицій). Вони, як правило, розглядають усі позиції на зображенні як потенційні об'єкти та намагаються класифікувати кожну цікаву область як фоновий або цільовий об'єкт.

Один з перших успішних одноетапних детекторів на основі глибокого навчання був OverFeat [19]. Архітектура представлена на рис. 1.3.

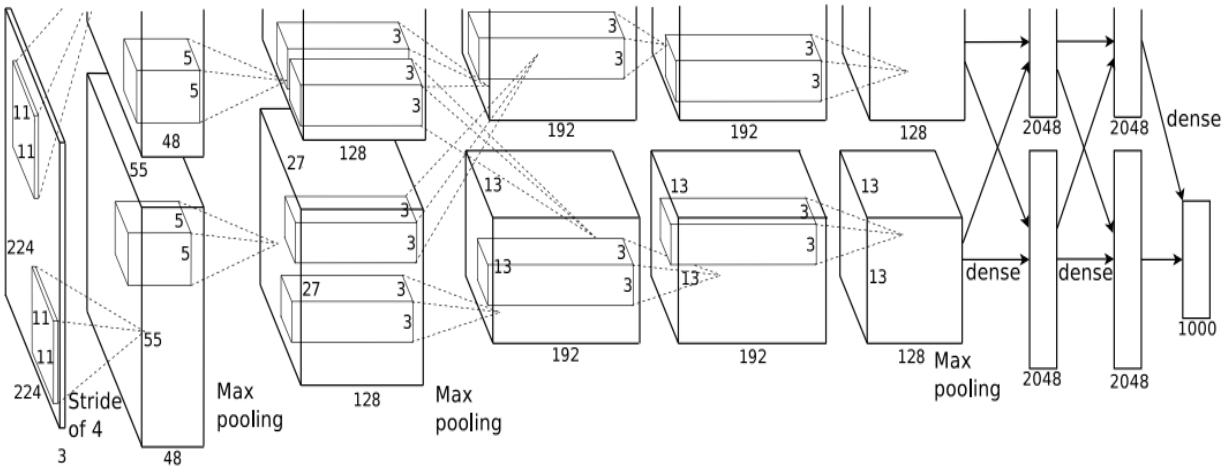

Рисунок 1.3 – Архітектура мережі OverFeat

OverFeat здійснював виявлення об'єктів, вводячи класифікатор DCNN в повністю згортковий детектор об'єктів. Визначення об'єкта може розглядатися як проблема «класифікація на багато регіонів», і, таким чином, OverFeat розширив оригінальний класифікатор в детектор, переглядаючи останні шари FC як згорткові шари 1x1, щоб дозволити довільні данні на вході. Класифікаційна мережа виводить сітку прогнозів для кожної області введення для вказівки присутності об’єкта. Після ідентифікації об'єктів, регресори обмежувального прямокутника були навчені уточнювати передбачувані області на основі тих же функцій DCNN класифікатора. З метою виявлення багатомасштабних об'єктів вхідне зображення було змінено в декілька масштабів, які подаються в мережу. Нарешті, прогнози в усіх масштабах були об'єднані разом. OverFeat показав значну швидкість швидкості порівняно з RCNN, поділивши обчислення областей, що перекриваються, за допомогою згорткових шарів, і потрібен був лише один прохід вперед по мережі. Однак навчання класифікаторів та регресорів було розділено без спільної оптимізації.

Пізніше було розроблено [11] детектор у реальному часі під назвою YOLO (you only look once - ви дивитесь лише один раз). Архітектура мережі YOLO представлена на рис. 1.4.

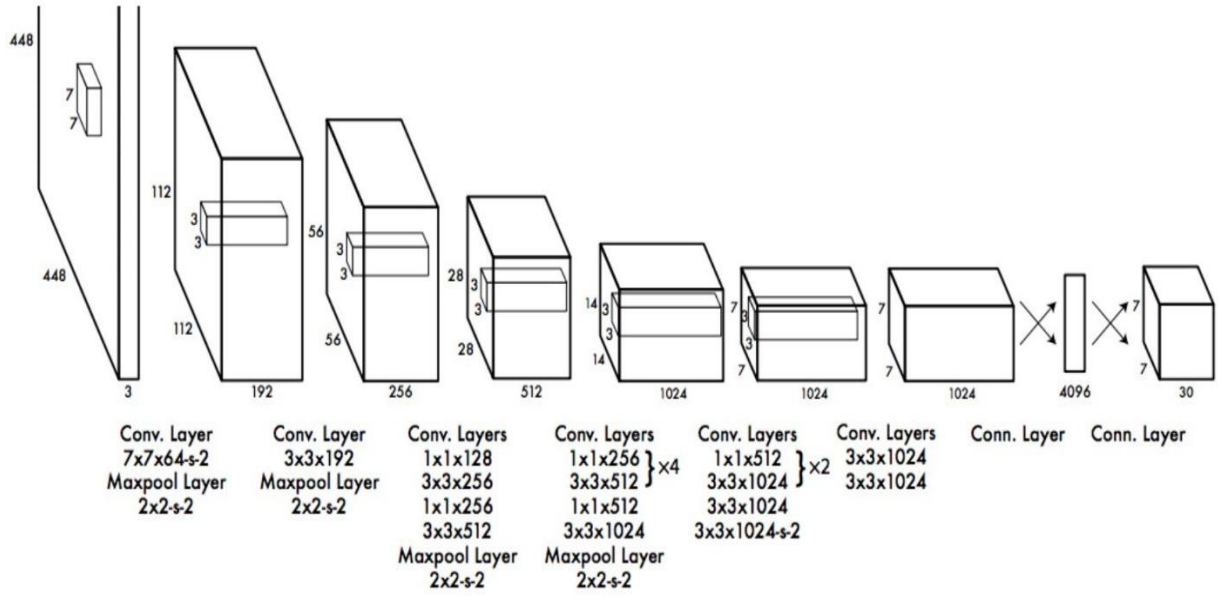

Рисунок 1.4 – Архітектура YOLO

Мережа складається з 24 згорткових шару, а потім 2 повністю з'єднані шари. Замість початкових модулів, які використовує GoogLeNet, використовується 1 × 1 відновлювальні шари з подальшими 3 × 3 згортковими шарами

YOLO розглядав виявлення об’єктів як проблему регресії і просторово розділив усе зображення на фіксовану кількість комірок сітки (наприклад, використовуючи сітку 7×7). Кожна клітина розглядалася як пропозиція виявити наявність одного або декількох об'єктів. У початковій реалізації кожна клітинка вважалася центром (до двох) об'єктів. Для кожної клітини було зроблено передбачення, яке містило таку інформацію: чи було в цьому місці розташування об'єкт, координати та розмір обмежувального поля (ширина та висота) та клас об'єкта. Весь фреймворк був єдиною мережею, і вона пропустила крок створення пропозицій, який можна було б оптимізувати поступово. Спираючись на ретельно розроблену легку архітектуру, YOLO може робити прогноз на швидкості 45 FPS і досягти 155 FPS з більш спрощеною основою. Однак YOLO зіткнувся з деякими проблемами:

1) він міг виявити до двох об'єктів у певному місці, що ускладнювало виявлення дрібних та переповнених предметів [11];

2) для прогнозування використовувалася лише остання карта характеристик, яка не підходить для прогнозування об'єктів у кількох масштабах та співвідношеннях сторін.

У 2016 році було запропоновано ще один одноступеневий детектор Single-Shot Mulibox Detector (SSD) [13], який вирішив обмеження YOLO. SSD також розділила зображення на комірки сітки, але в кожній комірці сітки було створено набір якорів із декількома масштабами та співвідношеннями сторін, щоб дискредитувати вихідний простір обмежувальних прямокутників (на відміну від прогнозування із фіксованих комірок сітки, прийнятих у YOLO). Кожен якір був уточнений 4-значними зсувами, що були вивчені регресорами, і цім якорям було призначено (C + 1) категоричні ймовірності. Крім того, SSD передбачив об'єкти на декількох картах характеристик, і кожна з цих карт відповідальна за виявлення певного масштабу об'єктів відповідно до його сприйнятливих полів. Щоб виявити великі об'єкти та збільшити сприйнятливі поля, до оригінальної архітектури магістральних вершин було додано кілька додаткових згорткових карт. Вся мережа була оптимізована за допомогою зваженої суми втрат від локалізації та втрат від класифікації на всіх картах прогнозування за допомогою схеми навчання з кінця в кінець. Остаточний прогноз був зроблений шляхом об'єднання всіх результатів визначення з різних карт

характеристик. Щоб уникнути величезної кількості негативних пропозицій, що домінували над градієнтами тренувань, для підготовки детектора було використовували жорсткий негативний майнінг. Для підвищення точності виявлення також було застосовано інтенсивне збільшення даних. SSD домігся порівнянної точності виявлення з швидшим R-CNN, але користувався перевагами робити визначення в реальному часі.

Без створення пропозицій для фільтрації легких негативних зразків, класовий дисбаланс між переднім планом та фоном є серйозною проблемою в одноетапному детекторі. Для рішення цієї проблеми було запропоновано [20] одноступінчастий детектор RetinaNet, який більш гнучко розглядав проблему дисбалансу. RetinaNet використовував фокусні втрати, які пригнічували градієнти легких негативних зразків, а не просто їх викидання. Крім того, вони використовували пірамідні мережі характеристик для виявлення багатомасштабних об'єктів на різних рівнях карт характеристик. Запропоновані ними фокусні втрати переважали наївну жорстку негативну стратегію видобутку з великим відривом.

Згодом було розроблено вдосконалену версію YOLO, YOLOv2 [12], яка значно покращила продуктивність виявлення, але все ж підтримувала швидкість виводу в режимі реального часу. YOLOv2 прийняв більш потужну архітектуру глибоких згортків, яка була досліджена на зображеннях з високою роздільною здатністю ImageNet (від 224×224 до 448×448), і таким чином вивчені ваги були більш чутливими до збору дрібнозернистих даних. Крім того, надихнувшись якорною стратегією, що використовується в SSD, YOLOv2 визначив кращі прив'язки якоря за допомогою кластеризації методом к–середніх з навчальних даних (замість встановлення вручну). Це допомогло зменшити труднощі оптимізації у локалізації.

Попередні підходи вимагали розробити якірні прямокутники вручну для підготовки детектора. Пізніше була розроблена серія детекторів без якоря, де метою було передбачити ключові точки обмежувального прямокутника, а не намагатися прилаштувати об'єкт до якоря. Було запропоновано новий фреймворк без якоря CornerNet [21], який виявляє об'єкти як пару куточків. На кожному положенні карти характеристик передбачалися теплові карти класів, вбудовування пари та зміщення кутів. Теплові карти класу розраховували ймовірність бути кутами, а кутові зміщення використовувались для регресу місця

розташування кута. А парні вставки слугували для згрупування пари кутів, які належать до одних і тих же об’єктів. Структурні елементи мережі представлені на рис. 1.4.

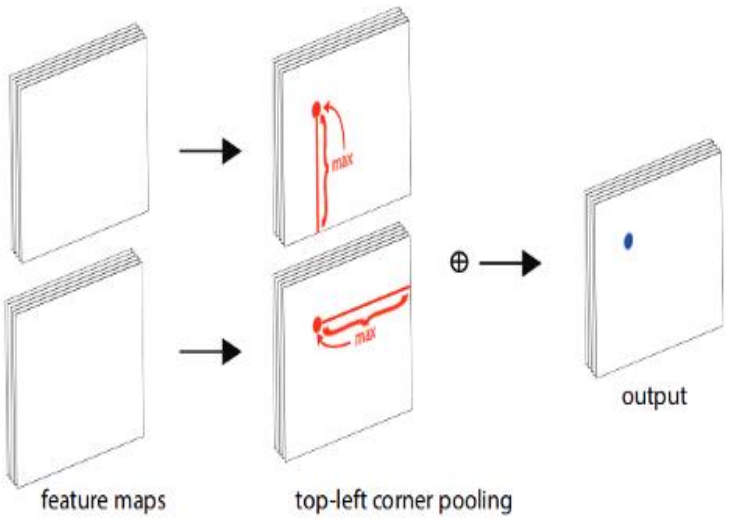

Рисунок 1.4 – Структурні елементи мережі CornerNet

Для кожного каналу об'єднання кутів мережа бере максимальні значення (червоні точки) в двох напрямках (червоні лінії), кожне з окремої карти об'єктів, і складає два максимуму разом (синя точка).

1.6 Генерація пропозицій

Генерація пропозицій відіграє дуже важливу роль у структурі виявлення об'єктів. Генератор пропозицій формує набір прямокутних обмежувальних регіонів, які потенційно є об'єктами. Ці пропозиції потім використовуються для уточнення класифікації та локалізації. Методи генерації пропозицій можна поділити на чотири категорії: традиційні методи комп’ютерного зору, методи навчання, що базуються на якорі, методи на основі ключових точок та інші методи. Зокрема, і одноступеневі детектори, і двоступеневі детектори генерують пропозиції, головна їх відмінність в тім що двоступеневі детектори генерують розрізнений набір пропозицій з лише передньою або фоновою інформацією, тоді як одноступінчасті детектори розглядають кожен регіон у зображення як потенційну пропозицію і, відповідно, оцінює координати класів і обмежувальних прямокутників потенційних об'єктів у кожному місці.

1.6.1 Традиційні методи комп’ютерного зору

Ці методи генерують пропозиції у зображеннях, використовуючи традиційні методи комп’ютерного зору, засновані на сигналах низького рівня, таких як краї, кути, колір тощо. Принципи цих методів можна поділити на наступні категорії: 1) обчислення об’єктивності оцінки регіону; 2) об'єднання суперпікселів з оригінального зображення; 3) генерація декількох сегментів переднього та фонового планів.

Методи, що базуються на оцінці об'єкта, передбачають оцінку об'єктивності кожного кандидатського регіону, вимірюючи, наскільки ймовірно, він може містити об'єкт. Об’єкту присвоюється [22] бали об'єктивності пропозицій за класифікацією на основі візуальних підказок, таких як кольоровий контраст, щільність краю та виразність.

Об'єднання суперпікселів засноване на об'єднанні суперпікселів, що утворюються за результатами сегментації. Алгоритмом заснованим на цьому принципі є вибірковий пошук [15].

Суперпікселі - це де замість того, щоб працювати з будь-яким пікселем у зображенні з найвищою доступною роздільною здатністю, знаходяться групи пікселів, найбільш схожі на інші групи, і їм присвоюється однакова мітка. Тоді для розпізнавання можна оперувати вже цими групами пікселів. Наприклад, замість того, щоб переглядати кількість червоного, зеленого та синього кольорів у кожній точці зображення, можна вже дивитися на очі, носи, вуха, колеса та інші маленькі повторювані речі, які не мають назв, але мають загальну мітку.

Цей принцип обчислює кілька ієрархічних сегментів, генерованих методом сегментації [23], які були об'єднані відповідно до своїх візуальних факторів (колір, області тощо), після чого обмежувальні прямокутники розміщувались на об’єднаних сегментах. Аналогічну ідею було запропоновано [24] і для злиття суперпікселів. Різниця полягала в тому, що вагу функції злиття було вивчено, а процес злиття був рандомізований. Вибірковий пошук широко застосовується у багатьох фреймворках виявлення через його ефективність та високу згадуваність порівняно з іншими традиційними методами.

Метод розростання областей з "насіння" починається з кількох "засіяних" областей, і для кожного насіння генеруються сегменти переднього та фонового планів. Як вхідні дані цей метод приймає зображення і набір "насіння". Насіння визначають об'єкти, які потрібно виділити. Області поступово розростаються, порівнюючи всі незайняті сусідні пікселі з

областю. Різниця між яскравістю пікселя і середньою яскравістю області використовується як міра схожості. Піксель з найменшою такою різницею додається у відповідну область. Процес триває доти, доки всі пікселі не будуть додані в один з регіонів. Результат залежить від вибору "насіння". Шум на зображенні може призвести до погано розміщення "насіння". Ці злиті сегменти використовуються як насіння для отримання більших сегментів.

Основна перевага цих традиційних методів комп'ютерного зору полягає в тому, що вони дуже прості та можуть створювати пропозиції з високою віддачею на наборах даних середнього масштабу. Однак ці методи в основному базуються на візуальних сигналах низького рівня, таких як колір або краї. Вони не можуть бути оптимізовані спільно з усім конвеєром виявлення. Таким чином, вони не в змозі використати потужність наборів великих масштабів для поліпшення навчання репрезентації.

1.6.2 Методи на основі якоря

Одне велике сімейство контрольованих генераторів пропозицій - це якірні методи. Вони генерують пропозиції на основі заздалегідь визначених якорів. Для створення пропозицій під наглядом, заснованих на глибоких згорткових картах характеристик було запропоновано мережа пропозицій регіону [7] (Region Proposal Network, *RPN*). Мережа ковзала по всій мапі характеристик за допомогою фільтрів згортки розміром 3×3. Для кожної позиції були розглянуті k якорів (або початкові оцінки обмежувальних прямокутників) різного розміру та співвідношення сторін. Ці розміри та співвідношення дозволяють порівнювати об'єкти на різних масштабах у всьому зображенні. На підставі ground truth обмежувальних прямокутників місця розташування об'єктів були зіставлені з найбільш підходящими якорями для отримання сигналу спостереження для оцінки якоря. 256-мірний вектор характеристик витягується з кожного якорю та подається у дві сусідні гілки - класифікаційний шар та регресійний шар. Гілка класифікації відповідає за моделювання оцінки об'єктності, тоді як гілка регресії кодує чотири значення для уточнення місця розташування обмежувальної коробки від початкової оцінки якоря. Виходячи з основної істини, кожний якір передбачався гілкою класифікації або об'єктом, або просто фоном.

Незважаючи на перспективні характеристики продуктивності, якорні опори на якорі виготовляються у декількох масштабах та співвідношенням сторін евристично. Цей вибір

дизайну може бути не оптимальним, і різні набори даних потребують різних стратегій проектування якоря.

Пізніше було розроблено [25] фреймворк DeepProposals, який передбачив пропозиції на мапі характеристик глибшого шару з низькою роздільною здатністю. Потім вони були відображені назад на шар карт характеристики з високою роздільною здатністю карти з для подальшого уточнення.

1.6.3 Методи на основі ключових точок

Інший підхід до створення пропозицій базується на виявленні ключових точок, які можна розділити на дві сім'ї: кутові методи та методи, що базуються на центрах. Кутові методи прогнозують обмежувальні прямокутники шляхом злиття пар кутів, вивчених із карти характеристик. Проблему виявлення об'єктів було переформульовано [26] у імовірнісну проблему. Для кожної точки на карті характеристик моделюється розподіл ймовірносетй вона є однією з 4 кутових типів об'єктів (верхній лівий, верхній правий, нижній лівий, нижній правий) та застосовував наївні байєсові класифікатори над кожним кут об'єктів для оцінки показника достовірності обмежувального поля. Цей кутовий алгоритм усунув конструкцію якорів і став більш ефективним методом отримання високоякісних пропозицій. Пізніше на його базі було запропоновано CornerNet [21], який безпосередньо моделював категоричну інформацію про кути. Інформація CornerNet моделює верхній лівий і нижній правий кути з новою функцією вбудовування характеристик та шару об'єднання кутів, щоб правильно відповідати ключовим точкам, що належать до одних і тих же об'єктів, отримуючи найкращі результати на загальнодоступних тестах. Для методів, що базуються на центрі, вірогідність бути центром об'єктів прогнозується у кожному положенні карти характеристик, а висота та ширина напряму перенесені без будь-яких якорів. Також було представлено [27] фреймворк відбору функцій без якоря (feature-selection-anchor-free, *FSAF*), яку можна було б включити в одноетапні детектори зі структурою FPN. У FSAF онлайн-блок вибору функцій застосовується для підготовки багаторівневих гілок на центральній основі, приєднаних до кожного рівня піраміди характеристик. Під час тренінгу FSAF динамічно присвоював кожному об'єкту найбільш підходящий рівень характеристик для навчання центральної гілки. Подібно до FSAF, було запропоновано [28] нову систему на основі центру, засновану на єдиній мережі

Пісочний годинник (Hourglass network) [21] без структури FPN. Крім того, було застосувано метод, що базується на центрах, у вирішенні проблем вищого рівня, таких як 3D-виявлення та розпізнавання людської пози. Ці методи без якоря утворюють перспективний напрямок досліджень у майбутньому.

1.6.4 Інші методи

Існують інші алгоритми генерації пропозицій, які не ґрунтуються на ключових точках або якорях, але також пропонують конкурентні показники. Одним з таких алгоритмів [29] є AZNet, який автоматично зосереджується на регіонах, що представляють великий інтерес. AZnet прийняла стратегію пошуку, яка адаптивно спрямовувала обчислювальні ресурси на субрегіони, які, ймовірно, містять об'єкти. Для кожного регіону AZnet передбачив два значення: показник масштабування та показники суміжності. Індикатор масштабування визначав, чи слід далі ділити цю область, яка може містити менші об'єкти, а показники відповідності позначають її об'єктивність. Початковою точкою було все зображення, і кожен розділений підрегіон рекурсивно обробляється таким чином, поки показник масштабування не буде занадто малим. AZnet краще виявляє розріджені та дрібні об'єкти порівняно з підходом по підбору якорь-об'єкт RPN.

1.7 Постановка наукової задачі та обґрунтування методики досліджень

Результати проведеного аналізу засобів, методів розпізнавання об'єктів у відеопотоці в режимі реального часу, показали, що у відомих публікаціях недостатньо розглянуті методи повного циклу розпізнавання об’єктів у відеопотоці в режимі реального часу. Більша частина досліджень зосереджена лише на етапі розпізнавання об’єктів з застосуванням стандартних моделей, виключаючи етап анотації даних. Також, проведений аналіз технології розпізнавання відео показав, що реалізації повного циклу розпізнавання відео потребує додаткового програмного засобу для адаптації даних з метою виключення конфліктів програмного забезпечення між етапами циклу розпізнавання відео.

В даному контексті слід визначити наступні задачі магістерської роботи:

1) Аналіз галузі застосування та методів розмітки відео та визначення об'єктів у відеопотоці в режимі реального часу.

2) Вибір засобу розмітки об’єктів у відеопотоці, методу подальшого автоматичного вилучення і використання цифрових даних відеопотоку в режимі реального часу.

3) Анотація даних відеопотоку.

4) Розробка програмного засобу для адаптації моделі даних відеопотоку з метою її подальшого використання.

5) Розробка мобільного додатку для тестування моделі в режимі реального часу.

6) Розробка та тестування моделі визначення об’єктів шляхом проведення експерименту розмітки даних та визначення об'єктів у відеопотоці в режимі реального часу.

7) Оцінка ефективності методу автоматичного визначення об’єктів у відеопотоці в режимі реального часу.

– Для проведення досліджень доцільно використовувати методи моделювання відеопотоку, технології глибокого навчання, нейронні мережі.

1.8 Висновки до розділу 1

Аналіз досліджень галузі розпізнавання відео дозволив сформулювати наступні висновки.

1. Недостатньо дослідженим є повний цикл розпізнавання відео, що включає наступні задачі: анотація об’єктів відео та подальше використання анотованих даних відео для визначення об’єктів у відеопотоці. Більша частина досліджень зосереджена лише на етапі визначення об’єктів з застосуванням стандартних моделей, виключаючи етап анотації даних.

2. Реалізація повного циклу розпізнавання відео потребує додаткового програмного засобу для адаптації даних з метою виключення конфліктів програмного забезпечення між етапами циклу розпізнавання відео.

3. Сформульована загальна науково-технічна задача дослідження, як задача підвищення точності автоматичного визначення об’єктів у відеопотоці в режимі реального часу в умовах відсутності розмічених даних та складності сцени відеопотоку шляхом розмітки складної сцени відеопотоку, створення моделі розпізнавання на підставі розмічених даних.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ МЕТОДУ АВТОМАТИЧНОГО ВИЗНАЧЕННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Метод автоматичного визначення об'єктів у відеопотоці включає систематизовану сукупність етапів, які потрібно здійснити, щоб виконати задачу розпізнавання відео. Ці етапи: анотація об’єктів відео та подальше використання анотованих даних відео для визначення об’єктів у відеопотоці. Найбільш точні результати визначення об’єктів у відеопотоці отримуються шляхом використання анотованих даних для навчання моделі. Навчання моделей зазвичай потребує дуже великої кількості даних. В умовах використання анотованих даних для навчання моделі обсяг даних відео може варіюватися. Процес анотації даних зазвичай проводиться вручну і вимагає великої кількості часу для анотації відео. Останнім часом доступними стали рішення для півавтоматичного анотування відеопотоку, що значно полегшує реалізацію цього етапу. Наявність спеціалізованого програмного забезпечення багаторазово прискорює цей процес. Одним з таких інструментів є OpenCVAT.

2.1 Computer Vision Annotation Tool

Computer Vision Annotation Tool (CVAT) - це інструмент із відкритим кодом для розмітки цифрових зображень та відео. Основна функція програми - надання користувачам зручних інструментів анотації. OpenCVAT – реалізовано у вигляді Web-додатку який підтримує різні робочі сценарії, як для окремих людей, так і для команд.

Завдання розпізнавання відео можна класифікувати як сновні завдання керованого машинного навчання наступним чином:

- виявлення об'єктів;
- класифікація зображень;
- сегментація зображень;

Open CVAT дозволяє анотувати дані для кожного з цих випадків.

Переваги OpenCVAT можна визначити наступним чином.

Реалізація у вигляді Web-додатку. Користувачам не потрібно встановлювати додаткове програмне забезпечення аби створювати завдання або анотувати дані. Їм достатньо відкрити посилання у веб-браузері.

Можливість спільної роботи. Користувачі можуть створити загальнодоступне завдання та розділити роботу між іншими користувачами.

Легкий та доступний спосіб розгортання. OpenCVAT можна встановити в локальній мережі за допомогою Docker.

Доступність автоматичної анотації. Наприклад, користувачі можуть використовувати інтерполяцію між ключовими кадрами.

Відкритість до інтеграції. OpenCVAT підходить для вбудовування у відкриті та розширювані платформи, наприклад, Onepanel.

Підтримка додаткових інструментів:

- Інструментарій глибокого вивчення розгортання OpenVINO
- TensorFlow API виявлення об'єктів (API TF OD)
- Аналітична система ELK (Elasticsearch + Logstash + Kibana)
- Інструментарій NVIDIA CUDA
- Підтримка різних сценаріїв анотацій.
- Відкритий вихідний код під ліцензією MIT.

Недоліками OpenCVAT можна назвати наступні чинники.

Обмежена підтримка браузера. Клієнт CVAT працює лише в Google Chrome. Хоча частково це можливо, роботу CVAT не гарантовано в інших браузерах, окрім браузерів на базі Chromium, таких як Opera або Yandex Browser.

Недопрацьована система автоматичного тестування. Усі перевірки потрібно робити вручну, що значно уповільнює процес розробки. Однак Intel працює над цим проблемою за допомогою студентів Лобачевського державного університету Нижнього Новгорода, які допомагають команді в рамках проекту IT-лабораторії.

Відсутність документації на вихідний код. Хоча це не є недоліком при використанні, це може ускладнити участь у розробці інструменту.

Питання продуктивності. Збільшення вимог через більший об’єм анотованих даних призвело до проблем із Chrome Sandbox, що обмежує використання оперативної пам’яті.

2.2 Архітектура OpenCVAT

OpenCVAT складається з наступних підсистем (рис. 2.1): власне CVAT, зберігання, підсистема аналітики та клієнтський інтерфейс користувача.

Підсистема CVAT – це центральний компонент системи, що відповідає за управління завданнями розмітки та обробку вхідних даних. Ця підсистема реалізована у вигляді Web-додатку, що керується процесом supervisord та розгортається у окремому контейнері. Supervisord – це система керування процесами, що дозволяє запускати та зупиняти процеси, а також слідкувати за ними. В даному випадку для кожного запиту від клієнтського додатку або CVAT UI, supervisord запускає новий процес WSGI-сервера, який оброблює запит та повертає результат у вигляді потоку HTTP.

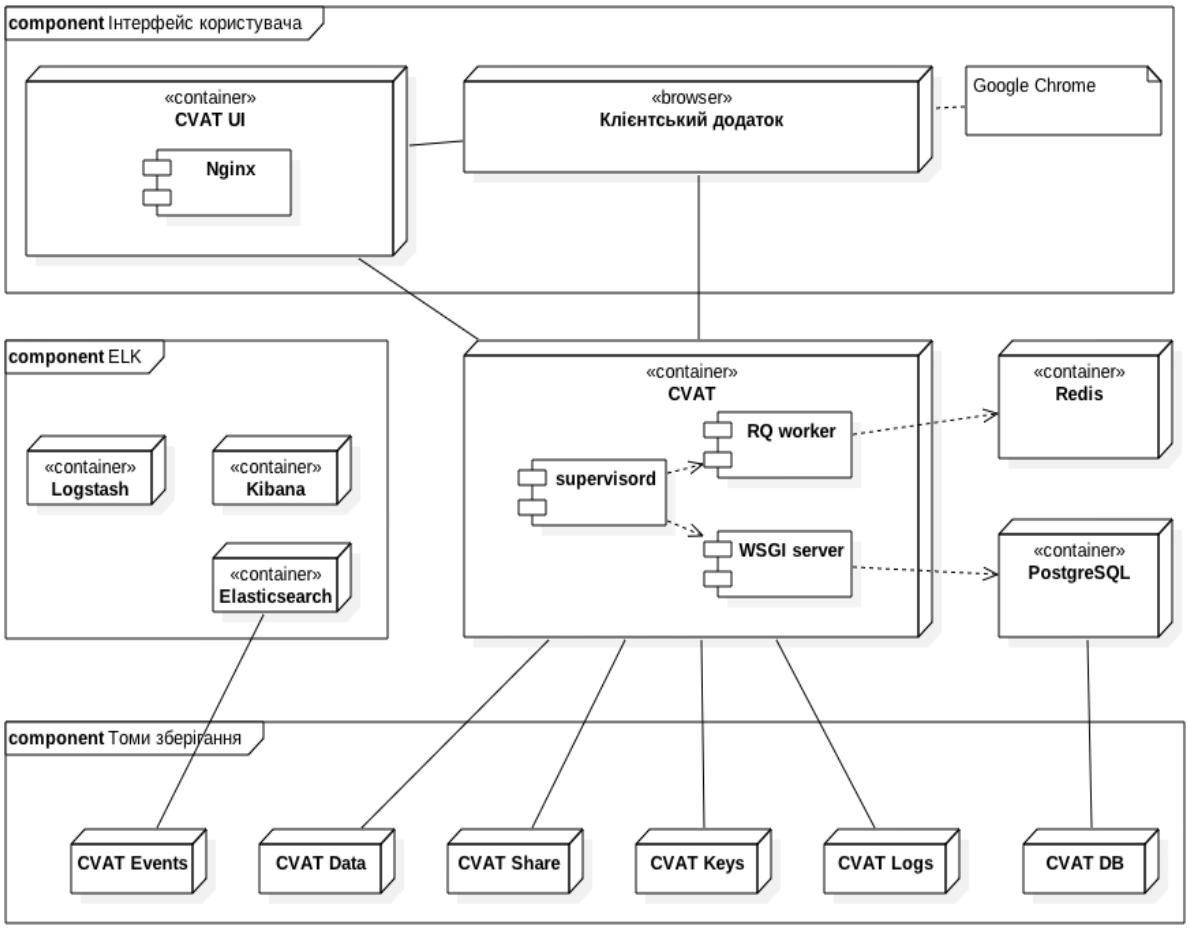

Рисунок 2.1 – Діаграма розгортання OpenCVAT

Зазвичай час, що відводиться на обробку запита сервером WSGI обмежено кількома секундами. Якщо сервер не встиг обробити запит за 30 секунд, то процес що оброблює запит буде примусово завершено і користувачеві буде повернуто помилку. Прикладом таких тривалих операцій може бути створення завдання на розмітку, підготовка файлу з розміткою, розмітка зображення тощо. Щоб забезпечити комфортний рівень користування додатком, та мати можливість виконувати тривалі операції між запитами в підсистемі передбачено асинхронний обробник завдань, який представлено як RQ-worker. Цей компонент реалізовано у вигляді черги, який бере завдання з сховища Redis, що були попередньо створені WSGI-сервером, та запускає відповідний код для обробки завдання. Інформація щодо користувачів, завдань, розмітки тощо, зберігається у базі даних PostgreSQL, яку розташовано на томі CVAT DB. Сервер бази даних розгорнуто в окремому контейнері.

Підсистему аналітики реалізовано за допомогою стека ElasticSearch-Logstash-Kibana (ELK), що розгортається на окремому контейнері. Коли робота зберігається, всі дані, включаючи журнали, передаються на сервер. Сервер передає його в Logstash для фільтрації. Крім того, є можливість автоматично надсилати сповіщення на електронну пошту, якщо виникають помилки. Потім журнали переносяться в Elasticsearch, де вони зберігаються в томі подій cvat. Після цього користувач може переглядати статистику та журнали в Кібані. Тим часом Кібана тісно співпрацюватиме з Elasticsearch.

Підсистема зберігання складається з сервера бази даних PostgreSQL, сховища Redis та томів для зберігання файлів даних. Томи зберігання є незалежними від контейнерів, та монтуються до відповідних контейнерів тільки під час роботи. Це дозволяє попередити втрату даних при оновленні контейнерів. Ключовими контейнерами що потрібні для роботи системи є контейнери Data та Share де зберігаються всі файли завдань. Контейнер Keys потрібен якщо CVAT було налаштовано на автентифікацію за допомогою ключів.

2.3 Режими роботи OpenCVAT

2.3.1 Режим анотації

Режим анотації призначений для анотації набору поодиноких зображень. Цей режим буде вибрано автоматично якщо похідним матеріалом є перелік зображень. Вибрати режим анотації вручну можна в правій нижній панелі вікна розмітки (рис. 2.2)

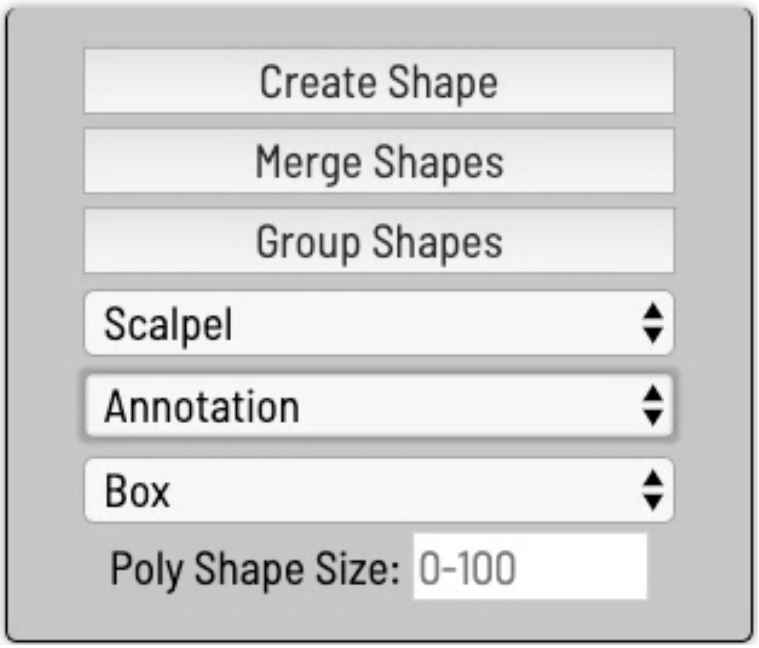

Рисунок 2.2 – Вибір режиму анотації

Для того щоб створити нову анотацію треба на правій нижній панелі вибрати фігуру (прямокутник, багатокутник, полігональна рамка, точки чи автосегментація), вибрати клас об’єкта (рис. 2.3) та обвести цю фігуру навколо об’єкта.

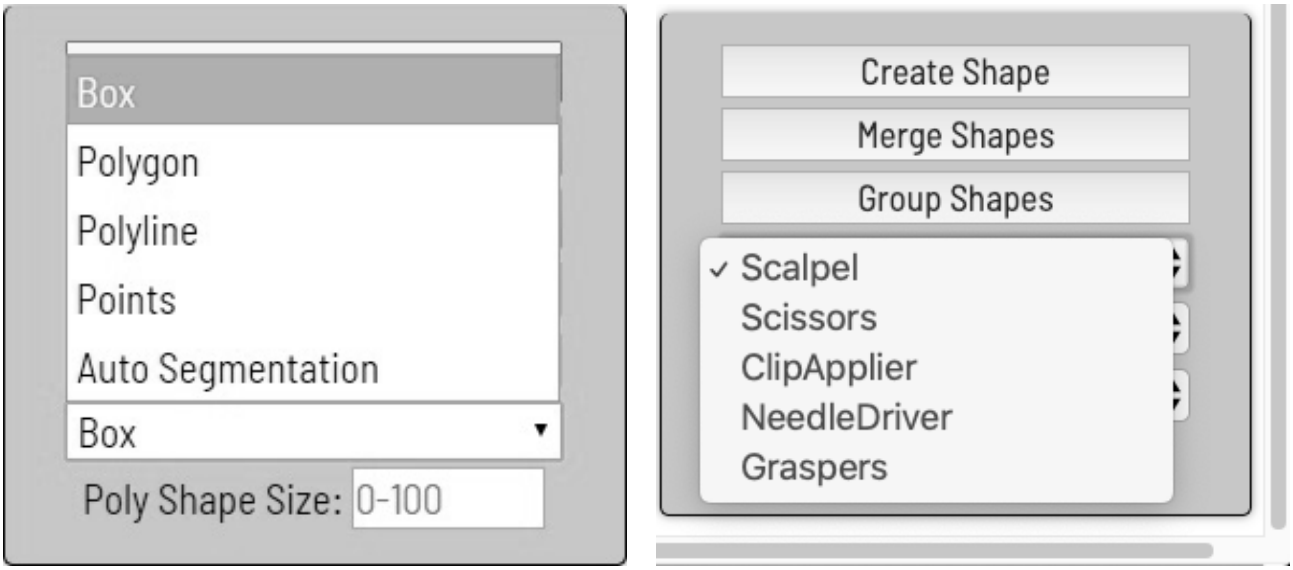

Рисунок 2.3 – Вибір фігури для анотації та класу об’єкта

2.3.2 Режим інтерполяції

При розмітці відео CVAT можна використовувати для інтерполяції обмежувальних прямокутників та атрибутів між ключовими кадрами, після чого набір зображень буде автоматично анотований. Це допомагає значно прискорити процес анотації. Цей режим буде вибрано автоматично якщо похідним матеріалом є відео. Вибрати режим інтерполяції вручну можна в правій нижній панелі вікна розмітки (рис. 2.4).

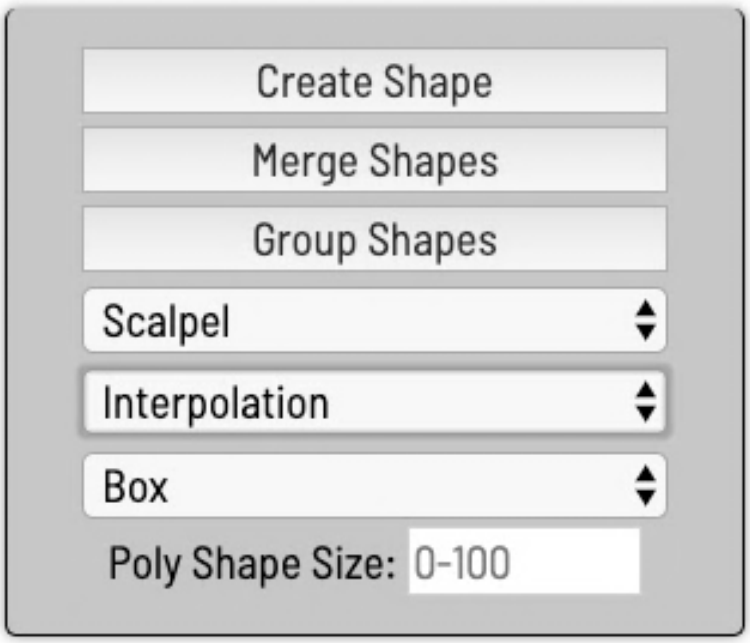

Рисунок 2.4– Вибір режиму інтерполяції

Порядок анотації об’єктів в режимі інтерполяції:

1. Помітити межі об’єкту на першому кадрі, де він з’явився.

2. Перейти до кадру, де об’єкт змінює свою позицію, та відкоригувати межі об’єкту. Не обов’язково змінювати позицію об’єкту на кожному кадрі – достатньо це зробити на кількох ключових кадрах. На всіх проміжних кадрах границі об’єкту будуть створені автоматично. Для того щоб помітити кадр як ключовий, треба натиснути кнопку «K» або вибрати «зірку» на панелі атрибутів об'єкту.

3. Якщо об’єкт стає частково невидимим через те що його було перекрито іншим об’єктом або через те що частина його вийшла за межі кадру, такий об’єкт потрібно помітити як частково перекритий (рис. 2.5) або occluded у термінах інтерфейсу. Для чого треба натиснути кнопку «Q». Межі об’єкту що було помічено таким чином стають пунктирними.

4. Якщо об’єкт тимчасово відсутній на зображенні або стає замалим, то слід помітити його як «за межами кадру». Для чого потрібно натиснути клавішу «O» або вибрати піктограму ока на панелі атрибутів об’єкту.

5. Коли об’єкт появиться у кадрі знов. То слід створити для нього новий трек (клавіша «N»), та об’єднати цей трек з попереднім. Для цього треба натиснути кнопку

«Merge Track» в правій нижній панелі вікна, потім знайти та виділити цей об’єкт на попередньому кадрі, та натиснути кнопку «Apply Merge» (рис. 2.6).

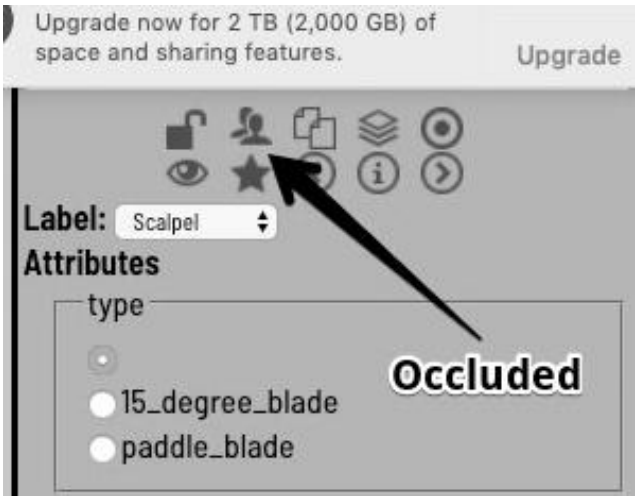

Рис. 2.5 – Помітка об’єкту як частково перекритого

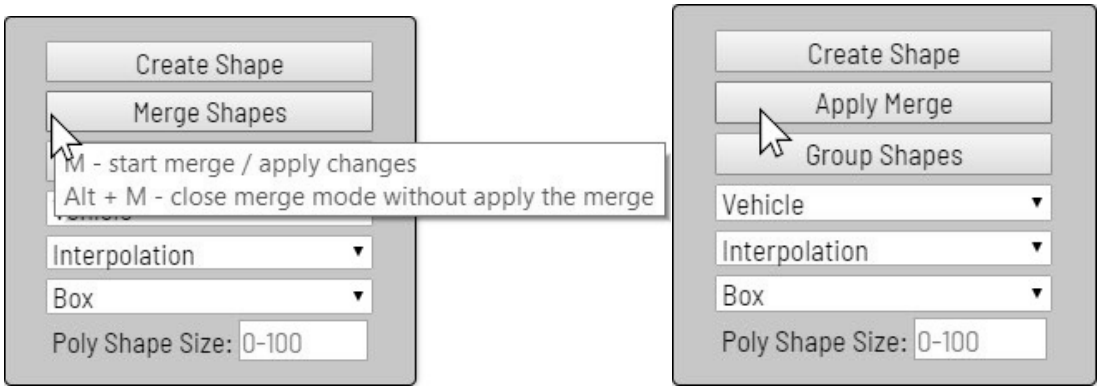

Рисунок 2.6 – Об’єднання треків для об’єкту

2.3.3 Режим анотації атрибутів

Цей режим було розроблено для класифікації зображень і прискорює процес анотації атрибутів, фокусуючи увагу анотатора лише на одному атрибуті. Додатково процес анотації здійснюється за допомогою гарячих клавіш. Для переходу в цей режим треба натиснути комбінацію клавіш «Shift + Enter». Після чого на панелі справа внизу вікна буде перераховано всі доступні атрибути для поточного об'єкту та «гаряча» клавіша, яку треба

натиснути, щоб вибрати цей атрибут (рис.2.7). Якщо об’єкт має декілька атрибутів, то поточний буде помічено червоним кольором. Щоб вибрати наступний або попередній атрибут треба це зробити за допомогою клавіш «Вниз» та «Вгору».

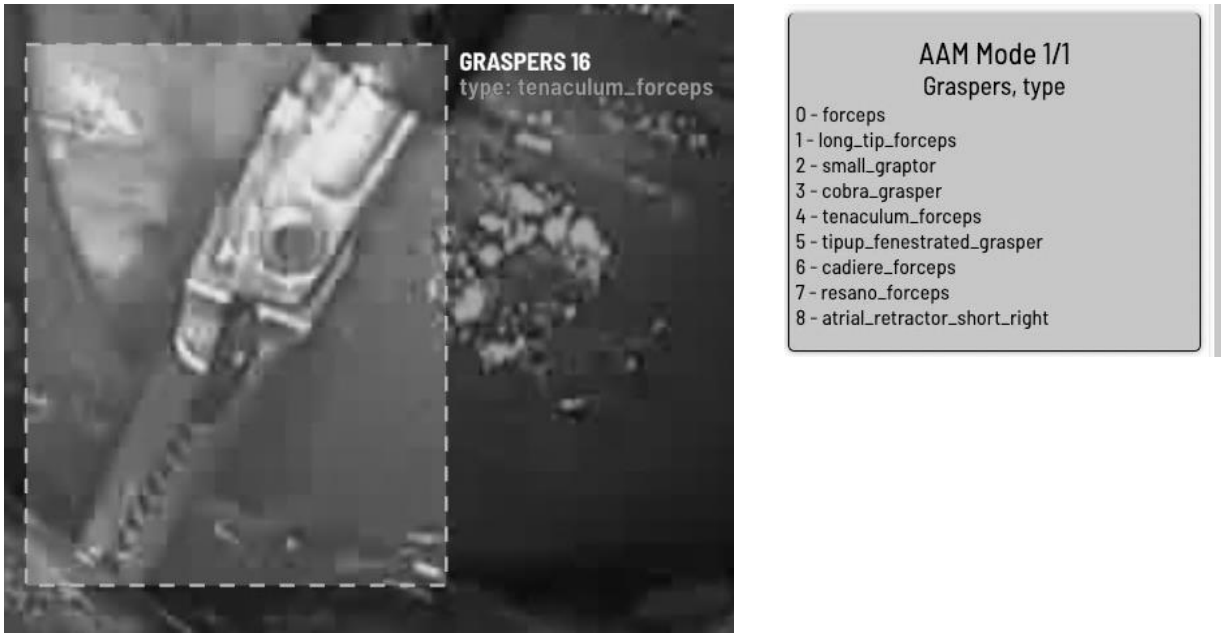

Рисунок 2.7 – Швидкий вибір атрибутів

2.4 Архітектура та навчання Faster R-CNN

CVAT використовує TensorFlow Object Detection API і Faster R-CNN для автоматизації задач анотації [30]. Схематично архітектуру Faster R-CNN представлено на рис. 2.8. Faster R-CNN має дві мережі [31, 32]: мережу пропозицій регіонів (RPN) для генерації пропозицій регіону та мережу, що використовує ці пропозиції для виявлення об'єктів.

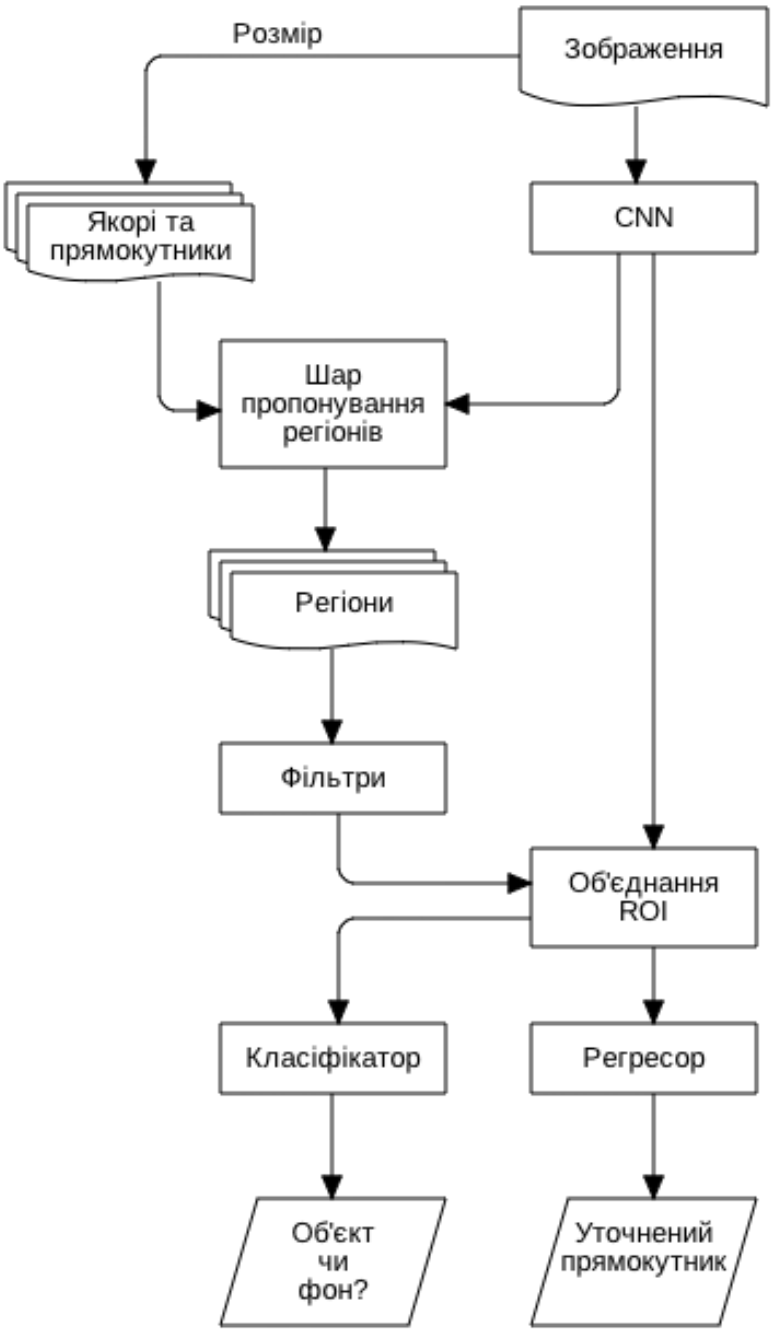

Рисунок 2.8 – Архітектура Faster R-CNN

RPN приймає зображення будь-якого розміру як вхідні дані та виводить набір пропозицій прямокутників з об’єктами, кожен з яких має оцінку об'єктності (рис. 2.9). Для генерації пропозиції міні-мережа пересувається по карті характеристик (conv feature map), що було отримано на виході останнього шару згорткової мережі. Кожне вікно проектується на вектор меншого виміру 256-d, що є проміжним шаром (intermidiate layer). Цей вектор

подається на вхід шару регресії прямокутників (reg layer) та шар класифікації прямокутників (cls layer).

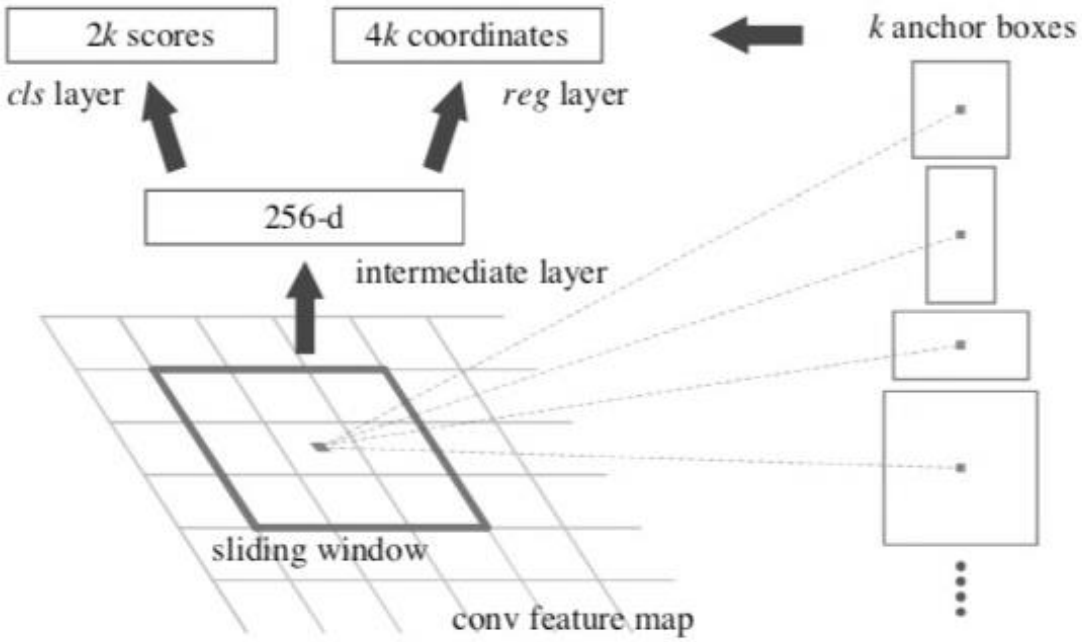

Рисунок 2.9 - Мережа пропозицій регіонів (RPN)

У кожному положенні вікна одночасно прогнозується *k* пропозицій регіонів, так що у шарі регресії є 4*k* виходи, що кодують координати *k* полів. Шар класифікації регіону має 2*k* виходів, які оцінюють ймовірність наявності об'єкта чи фону в регіоні. Пропозиції зроблено відносно до *k* якірних регіонів (anchor boxes). Якорі відіграють важливу роль у Faster R-CNN. Кожен якір розміщений у центрі регіону, та описаний масштабом (128^2, 256^2 або 512^2 пікселів) та відношенням сторін (1:1, 1:2, 2:1) [29]. Таким чином у конфігурації Faster R-CNN за замовчуванням є 9 якорів для кожної позиції в зображення (рис. 2.10).

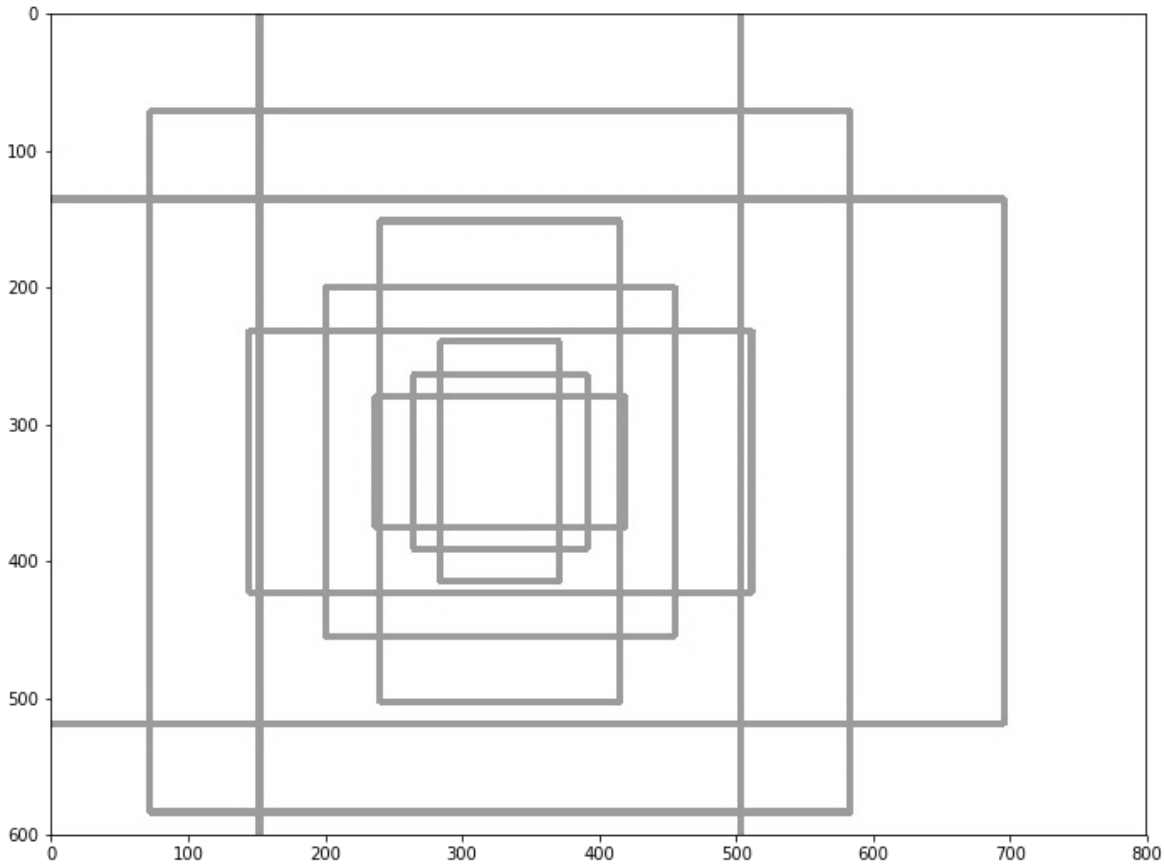

Рисунок 2.10 – Якорі для точки 320,320

Для навчання RPN кожному якорю присвоюється двійкова мітка класу – чи це об’єкт, чи ні. Позитивна мітка присвоюється двом типам якорів – якорям що мають максимальне IoU перекриття з регіоном основної правди; та якорям що мають коефіцієнт перекриття IoU з будь-яким регіоном основної правди більший ніж 0.7. Кожен регіон основної правди може присвоювати позитивну мітку декільком якорям. Якщо непозитивний якір має коефіцієнт перекриття IoU для всіх регіонів основної правди менш ніж 0.3, то йому присвоюється негативна мітка. Якорі які не мають позитивної або негативної мітки не приймають участь в навчанні.

Функція втрат (2.1) для зображення має наступний вигляд:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (2.1)$$

,

де i – це індекс якорю, а p_i – це прогнозована вірогідність того що якір i є об'єктом; мітка основної правди p^*_i має значення 1 якщо якір є позитивним, або 0 – якщо ні; t_i – це вектор з чотирьох параметризованих координат прямокутного регіону, що було спрогнозовано, а t^*_i – це прямокутний регіон основної правди, що відноситься до позитивного якорю; L_{cls} – це logloss від двох класів (чи об'єкт/чи ні). Як функцію втрат для регресії використовується формула (2.2). Термін $p^*_i L_{reg}$ означає що функцію втрат для регресії активовано тільки для позитивних якорів, тобто $p^*_i = 1$, та виключено в іншому випадку ($p_{*i} = 0$). Два терміни нормалізуються за допомогою *Ncls* і *Nreg*, та балансуючої ваги λ.

$$L_{reg}(t_i, t^*_i) = R(t_i - t^*_i), \tag{2.2}$$

де R – це функція втрат Хьюбера L_1, що обчислюється за формулою (2.3):

$$L(x) = \begin{cases} 0.5x^2, \text{якщо } |x| < 1 \\ |x| - 0.5 \end{cases}. \tag{2.3}$$

Координати для регресії параметризуються за формулами (2.4):

$$\begin{aligned} t_x &= (x - x_a) / w_a, \\ t_y &= (y - y_a) / h_a, \\ t_w &= log(w - w_a), \\ t_h &= log(h/h_a), \\ t^*_x &= (x^* - x_a) / w_a, \\ t^*_y &= (y^* - y_a) / h_a, \\ t^*_w &= log(w^* - w_a), \\ t^*_h &= log(h^* - h_a), \end{aligned} \tag{2.4}$$

де x та y – це координати центру прямокутного регіону, w та h – ширина та висота регіону, а x, x_a та x^* (так само як і y, w, h) означають координати прогнозованого і якірного регіонів та регіону основної правди.

RPN що реалізовано як глибока згорткова нейронна мережа може бути натренована за допомогою зворотного поширення та методу стохастичного градієнта. Кожна міні-партія виникає з одного зображення, яке містить багато позитивних і негативних якорів. Можна оптимізувати функції втрат для усіх якорів, але це призведе до зміщення негативних зразків, оскільки вони є домінуючими. Натомість випадково відбирається 256 якорів у зображенні для обчислення функції втрат міні-партії, де вибіркові позитивні та негативні якорі мають співвідношення до 1:1. Якщо на зображенні менше 128 позитивних зразків, то партія доповнюється негативними. Всі нові шари ініціалізуються випадково, витягуючи ваги із розподілу Гауса із нульовим середнім зі стандартним відхиленням 0,01. Усі інші шари ініціалізуються за допомогою попередньої підготовленої моделі класифікації ImageNet.

2.5 Висновки до розділу 2

У розділі розглянуто процес півавтоматичного анотування відеозображень з застосуванням режиму використання обмежувальної рамки з використанням вільного програмного забезпечення OpenCVAT.

Представлений прикладний математичний метод для півавтоматичного анотування об’єктів у відеопотоці заснований на Faster R-CNN.

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПОВНОГО ЦИКЛУ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ

Практична реалізація методу автоматичного визначення об'єктів відеопотоку в режимі реального часу проведена з реалізацією повного циклу розпізнавання відео, що включає анотацію об'єктів відео та подальше використання анотованих даних відео для визначення об'єктів у відеопотоці. Для практичної реалізації використано відео лапароскопічної хірургічної операції. Відеопотік лапароскопічної хірургічної операції містить наступні основні об'єкти: внутрішні органі черевної порожнини, медичні інструменти хірургічного комплексу DaVinci.

Практична реалізація методу автоматичного визначення об'єктів у відеопотоці в режимі реального часу складається з двох задач.

1. Анотація об'єктів у відеопотоці лапароскопічної хірургічної операції з використанням обмежуючої рамки та моделювання даних відеопотоку.
2. Тестування розробленої моделі та автоматичне визначення об'єктів у відеопотоці лапароскопічної хірургічної операції в режимі реального часу.

– Визначена наступна сукупність етапів методу автоматичного визначення об'єктів у відеопотоці в режимі реального часу, які потрібно здійснити, щоб виконати задачу розпізнавання відео:

– Визначення параметрів апаратного та програмного забезпечення.
– Розгортання OpenCVAT на сервері.
– Отримання зображень та відео з об'єктами для навчання
– Перетворення відео на серію зображень
– Перетворення серій зображень на файл з відео
– Створення завдання на анотацію
– Анотація зображень
– Підготовка даних та середовища виконання для навчання моделі
– Тестування моделі на відео в реальному часі
– Оцінка результатів тестування для визначення ефективності моделі.

В якості об’єктів анотації визначені інструменти хірургічного комплексу DaVinci. Тестування моделі на відеопотоці має відбуватися за допомогою мобільного додатку, що було розроблено у рамках цієї роботи.

Комплекс DaVinci має можливість роботи з наступними змінними інструментами: скальпелі (scalpel), ножиці (scissors), кліп-аплікатори (clip applier), голки (needle driver), пінцет (graspers). Для цих класів інструментів було створено наступні мітки:

– Scalpel;
– Scissors;
– ClipApplier;
– NeedleDriver;
– Graspers.

Зазначені змінні інструменти комплексу DaVinci представлені на рис. 3.1.

Рисунок 3.1 – Приклади інструментів для хірургічного комплексу DaVinci

3.1 Апаратне та програмне забезпечення

Апаратне та програмне забезпечення, що використовувалось в процесі практичної реалізації методу автоматичного визначення об’єктів у відеопотоці в режимі реального часу, та його параметри представлені в табл. 3.1.

Таблиця 3.1 – Апаратне та програмне забезпечення

Найменування	**Параметри**
CPU	Intel Core i7 8700K
RAM	32 GB 3000 MHz DDR4
GPU	MSI PCI-Ex Radeon RX 560 Aero ITX OC 4GB GDDR5
Накопичувач	Intel 60p Series M.2 2280 1TB Internal SSD
Операційна система	MacOS Mojave 10.14.6 (18G103)
Python	3.7.2
Операційна система мобільного пристрою	iOS 13

Підтримка апаратного прискорення AMD зазначена в табл. 3.2.

Таблиця 3.2 – Підтримка апаратного прискорення AMD

Найменування	**Підтримка GPU AMD**
YOLOv3 in Pytorch	ні
YOLOv3 in Keras	ні
TuriCreate	так

Виходячи з наявного програмного та апаратного забезпечення, та з того що на момент розробки тільки одна з популярних реалізацій для машинного навчання офіційно підтримувала апаратне прискорення на GPU платформи AMD, для реалізації було обрано TuriCreate [33].

3.2 Розгортання OpenCVAT

Розгортання OpenCVAT на сервері складається з наступних кроків.

1.Оренда сервера. Було обрано мінімальна конфігурація сервера з наявних у провайдера DigitalOcean, що задовольняла умовам проведення експерименту:

а) Кількість процесорів vCPU: 1

б) Операційна система: Ubuntu 16.04.6 (LTS) x64

в) Об’єм ОЗП: ГБ

г) Об’єм сховища даних 25ГБ.

Загальна схема перетворення зображень та відеофайлів на модель розпізнавання об’єктів зображено на рис. 3.2.

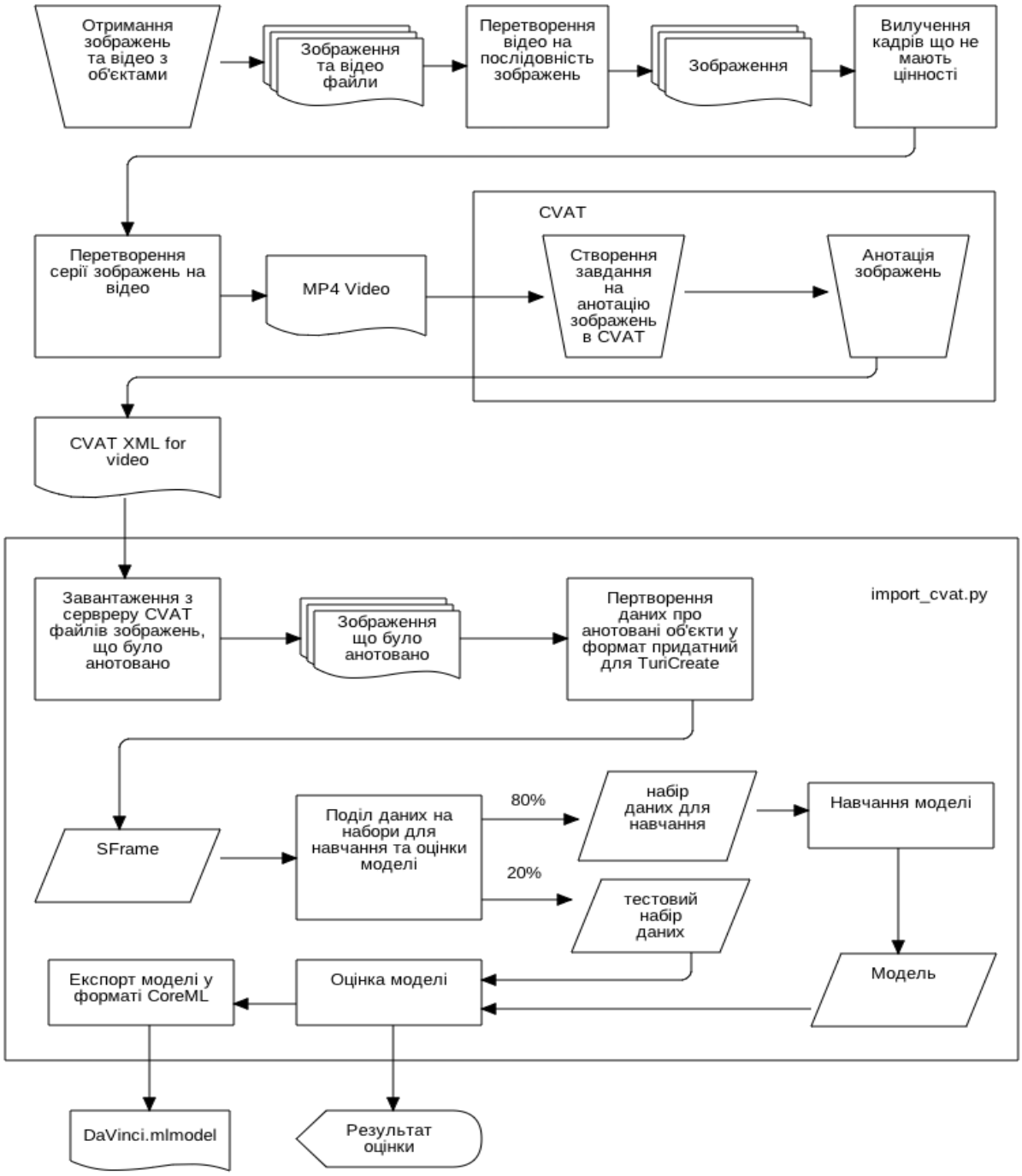

Рисунок 3.2 – Схема отримання файлу моделі та результату оцінки моделі

2. Підготовка серверу для запуску контейнерів Docker

```
1.$ docker-machine create --driver digitalocean ⏎
  --digitalocean-access-token $DO_ACCESS_TOKEN cvat
2.$ docker-machine env cvat
3.$ docker-machine regenerate-certs cvat
4.$ vi docker-compose.override.yml
5.version: "2.3"
6.
7.services:
8.  cvat:
9.    environment:
10.      ALLOWED_HOSTS: 167.172.231.336
11.    ports:
12.      - "80:8080"
13.
14.$ wget https://gist.githubusercontent.com/peatiscoding/⏎
  3310d89cb871ac5f414dbccdb1ed8d46/raw/⏎
  e3a4dac12708b03bfe204a660c7509db2654a8af/build-tag-push.py
15.$ python build-tag-push.py
```

3. Побудова та встановлення контейнерів CVAT на сервер

```
1.$ docker-compose -f docker-compose.yml ⏎
 -f docker-compose.override.yml up --build -d
```

4. Створення облікового запису адміністратора

```
1.$ docker exec -it cvat bash -ic 'python3 ⏎
  ~/manage.py createsuperuser'
2.Username (leave blank to use 'user'):
3.Email address: user@example.com
```

```
4.Password:
5.Password (again):
6.Superuser created successfully.
```

5. Створення облікових записів інших користувачів на сервері, для чого потрібно відкрити в Web-браузері сторінку курування обліковими записами, що є доступна за адресою /admin/auth/user/add/ на щойно створеному сервері, та увійти в додаток з обліковим записом адміністратору (рис.3.3)

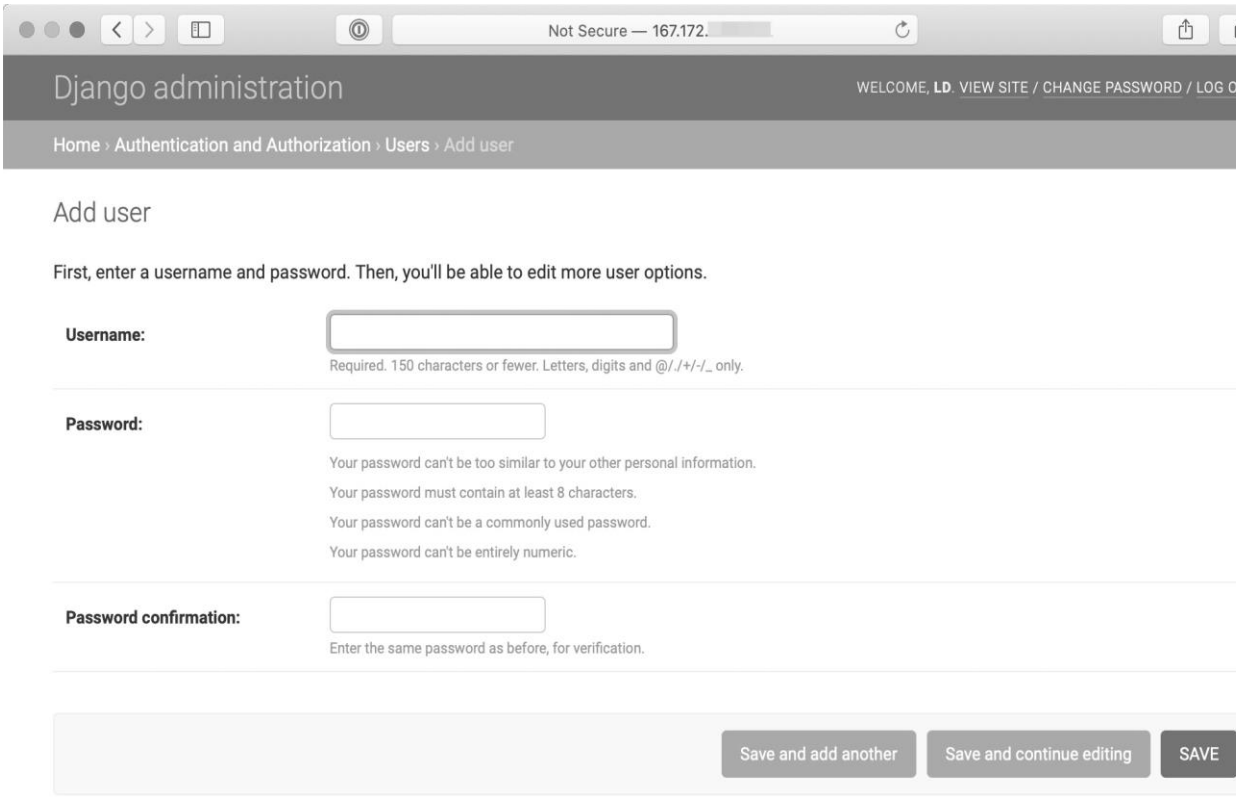

Рисунок 3.3 – Образ сторінки панелі керування користувачами

3.3 Отримання зображень та відео з об’єктами для навчання

Для навчання моделі було прийнято рішення використовувати відео, що демонструють роботу з комплексом DaVinci [34, 35] з відкритих джерел, зокрема YouTube. Програмна реалізація етапу виглядає наступним чином.

```
1. $ pip install youtube_dl
2. $ youtube-dl -f 'bestvideo[ext=mp4]' 0XdC1HUp-rU
3. [youtube] 0XdC1HUp-rU: Downloading webpage
4. [youtube] 0XdC1HUp-rU: Downloading video info webpage
5. [download] Destination: da Vinci Robot Stitches a Grape Back
   Together-0XdC1HUp-rU.mp4
6. [download] 100% of 10.74MiB in 00:02
7. $ youtube-dl -f 'bestvideo[ext=mp4]' 5JM8KhWhrus
8. [youtube] 5JM8KhWhrus: Downloading webpage
9. [youtube] 5JM8KhWhrus: Downloading embed webpage
10.[youtube] 5JM8KhWhrus: Refetching age-gated info webpage
11.[download] Destination: Inferior Vena Cava (IVC)
   Thrombectomy-da Vinci Robotic Surgery-5JM8KhWhrus.mp4
12.[download] 100% of 27.49MiB in 00:06
13.$ youtube-dl -f 'bestvideo[ext=mp4]' n18zxNGJdLE
14.[youtube] n18zxNGJdLE: Downloading webpage
15.[youtube] n18zxNGJdLE: Downloading embed webpage
16.[youtube] n18zxNGJdLE: Refetching age-gated info webpage
17.[download] Destination: Laparoscopic Cholecystectomy
   n18zxNGJdLE.mp4
18.[download] 100% of 82.11MiB in 00:16
```

3.4 Перетворення відео на серію зображень

Для поліпшення якості даних проведено перетворення відео на серію зображень з метою подальшого видалення кадрів, що не містять цільових об’єктів розпізнавання або кадрів, що мають низьку якість зображення. Перетворення відео на серію зображень реалізовано наступним чином.

```
1. $ brew install ffmpeg;mkdir frames; cd frames
2. $ ffmpeg -i ../*0XdC1HUp-rU.mp4 0Xd-%04d.jpg -hide_banner
3. Input #0, mov,mp4,m4a,3gp,3g2,mj2, from '../da Vinci Robot
   Stitches a Grape Back Together-0XdC1HUp-rU.mp4':
4.   Metadata:
```

```
    major_brand     : dash
    minor_version   : 0
    compatible_brands: iso6avc1mp41
    creation_time   : 2018-07-21T20:20:04.000000Z
  Duration: 00:01:47.37, start: 0.000000, bitrate: 839 kb/s
    Metadata:
      creation_time   : 2018-07-21T20:20:04.000000Z
      handler_name    : VideoHandler
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> mjpeg (native))
Output #0, image2, to '0Xd-%04d.jpg':
  Metadata:
    major_brand     : dash
    minor_version   : 0
    compatible_brands: iso6avc1mp41
    encoder         : Lavf58.20.100
    Stream #0:0(und): Video: mjpeg, yuvj420p(pc), 1920x1080
  [SAR 1:1 DAR 16:9], q=2-31, 200 kb/s, 29.97 fps, 29.97 tbn,
  29.97 tbc (default)
    Metadata:
      creation_time   : 2018-07-21T20:20:04.000000Z
      handler_name    : VideoHandler
      encoder         : Lavc58.35.100 mjpeg
    Side data:
      cpb: bitrate max/min/avg: 0/0/200000 buffer size: 0
  vbv_delay: -1
frame= 3218 fps=128 q=24.8 Lsize=N/A time=00:01:47.37
  bitrate=N/A speed=4.27x
video:143841kB audio:0kB subtitle:0kB other streams:0kB global
  headers:0kB muxing overhead: unknown
$ ffmpeg -i ../*5JM8KhWhrus.mp4 5JM-%04d.jpg -hide_banner
```

```
[...]
$ ffmpeg -i ../*n18zxNGJdLE.mp4 n18-%04d.jpg -hide_banner
[...]
$ cd ..
```

3.5 Перетворення серій зображень на файл з відео

Після вилучення кадрів, що не містять потрібних об'єктів та кадрів з низькою якістю зображення потрібних об'єктів, решта зображень для зручності завантаження на сервер CVAT було знов перетворено на відео.

```
$ ffmpeg -framerate 24 -pattern_type glob -i 'frames/*.jpg' ↵
-c:v libx264 out.mp4
ffmpeg version 4.1.4 Copyright (c) 2000-2019 the FFmpeg
developers
  built with Apple LLVM version 10.0.1 (clang-1001.0.46.4)
  configuration: --prefix=/usr/local/Cellar/ffmpeg/4.1.4_2 --
enable-shared --enable-pthreads --enable-version3 --enable-
avresample --cc=clang
  libavutil      56. 22.100 / 56. 22.100
  libavcodec     58. 35.100 / 58. 35.100
  libavformat    58. 20.100 / 58. 20.100
  libavdevice    58.  5.100 / 58.  5.100
  libavfilter     7. 40.101 /  7. 40.101
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  3.100 /  5.  3.100
  libswresample   3.  3.100 /  3.  3.100
  libpostproc    55.  3.100 / 55.  3.100
Input #0, image2, from 'frames/*.jpg':
  Duration: 00:12:12.71, start: 0.000000, bitrate: N/A
    Stream #0:0: Video: mjpeg, yuvj420p(pc,
bt470bg/unknown/unknown), 1920x1080 [SAR 1:1 DAR 16:9], 24
fps, 24 tbr, 24 tbn, 24 tbc
Stream mapping:
```

```
18.  Stream #0:0 -> #0:0 (mjpeg (native) -> h264 (libx264))
19. [libx264 @ 0x7fb30981b600] using SAR=1/1
20.[libx264 @ 0x7fb30981b600] using cpu capabilities: MMX2
   SSE2Fast SSSE3 SSE4.2 AVX FMA3 BMI2 AVX2
21.[libx264 @ 0x7fb30981b600] profile High, level 4.0
22.
23.Output #0, mp4, to 'out.mp4':
24.  Metadata:
25.    encoder         : Lavf58.20.100
26.    Stream #0:0: Video: h264 (libx264) (avc1 / 0x31637661),
   yuv420p, 1920x1080 [SAR 1:1 DAR 16:9], q=-1--1, 24 fps, 12288
   tbn, 24 tbc
27.    Metadata:
28.      encoder         : Lavc58.35.100 libx264
29.    Side data:
30.    cpb: bitrate max/min/avg: 0/0/0 buffer size: 0
31.Stream mapping:
32.  Stream #0:0 -> #0:0 (h264 (native) -> mjpeg (native))
33.Output #0, image2, to '0Xd-%04d.jpg':
34.  Metadata:
35.    major_brand     : dash
36.    minor_version   : 0
37.    compatible_brands: iso6avc1mp41
38.    encoder         : Lavf58.20.100
39.    Stream #0:0(und): Video: mjpeg, yuvj420p(pc), 1920x1080
   [SAR 1:1 DAR 16:9], q=2-31, 200 kb/s, 29.97 fps, 29.97 tbn,
   29.97 tbc (default)
```

3.6 Створення завдання на анотацію

Для створення завдання на анотацію треба відкрити сторінку CVAT та натиснути кнопку «Create New Task» та заповнити поля форми «Task Configuration» (рис. 3.4).

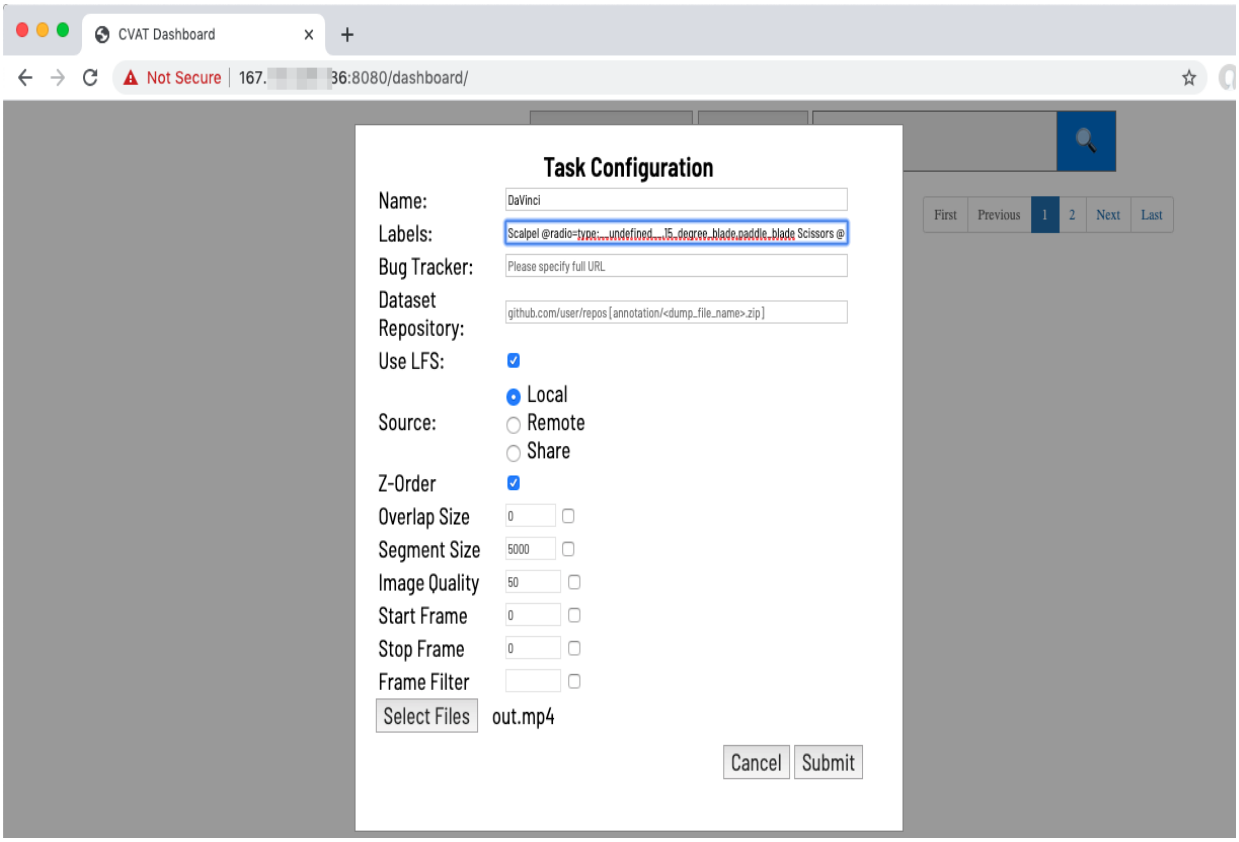

Рисунок 3.4 – Етап створення завдання на анотацію

Після створення завдання розпочато анотацію відеозображень, процес якої описаний далі.

3.7 Анотація відеозображень

Результат анотації одного з кадрів методом обмежуючої рамки відображено на рис. 3.5.

Для навчання моделі має бути принаймні 30 зображень кожного класу об'єктів у різному оточенні. Щоб отримати кращі результати рекомендується [37] мати 200 або більше зображень.

Рисунок 3.5 – Процес анотації зображення методом обмежуючої рамки

Контролювати кількість наявних зображень для кожного класу об'єктів можна за допомогою статистики анотованих даних, що надається OpenCVAT (рис. 3.6).

DaVinci

annotation

Frames: [0-25366] Overlap: 5 Z-Order: true

Segment Statistic

Label	Boxes		Polygons		Polylines		Points		Manually	Interpolated	Total
	S	T	S	T	S	T	S	T			
Scalpel	0	1	0	0	0	0	0	0	2	18	**20**
Scissors	0	3	0	0	0	0	0	0	92	722	**814**
ClipApplier	0	3	0	0	0	0	0	0	98	1086	**1184**
NeedleDriver	0	2	0	0	0	0	0	0	210	844	**1054**
Graspers	0	6	0	0	0	0	0	0	220	2484	**2704**
Total:	**0**	**15**	**0**	**0**	**0**	**0**	**0**	**0**	**622**	**5154**	**5776**

Рисунок 3.6 – Статистика анотованих даних

3.8 Підготовка даних та середовища виконання для навчання моделі

Після закінчення анотування відеозображень результат необхідно зберегти на комп’ютер в форматі "CVAT XML 1.1 for videos” (рис. 3.7). Цей файл містить перелік усіх анотованих відеозображень, та координати кожного анотованого об’єкту та його мітку.

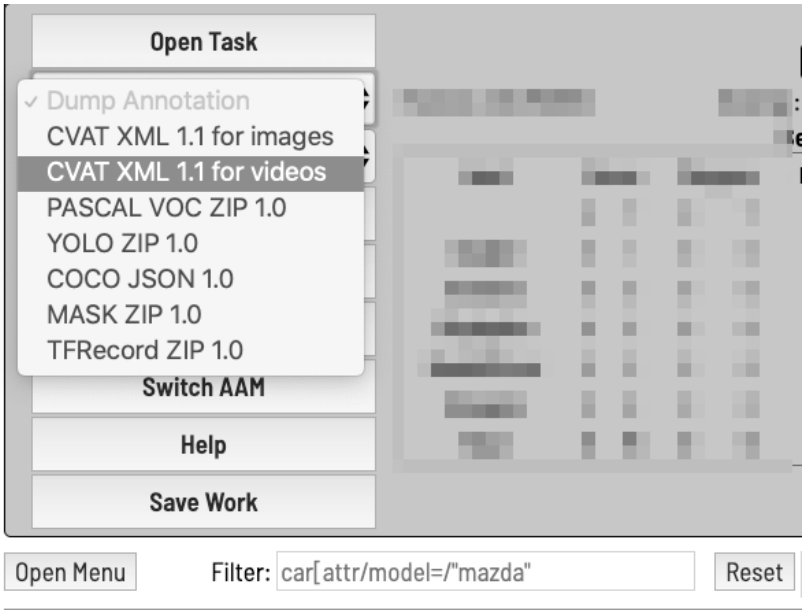

Рисунок 3.7 – Меню експорту даних розмітки

Після чого для навчання моделі передано отриманий файл з анотованими даними на вхід сценарію import_cvat.py (додаток А).

```
1. $ python3 -m venv .local
2. $ . .local/bin/activate && pip install --upgrade pip
3. $ . .local/bin/pip install turicreate==6.0 untangle==1.1.1
   python-dotenv[cli]↵ requests==2.22.0
4. $ ./import_cvat.py 12_DaVinci.xml
```

Під час навчання моделі виводиться інформація про кількість завершених ітерацій навчання. Наприкінці роботи сценарію було роздруковано результат оцінки моделі на тестовому наборі даних, та згенеровано файл моделі у форматі CoreML, придатному для подальшого використання в мобільному додатку.

Для оцінки якості моделі використаний критерій середньої точності - Mean Average Precision (MAP) (3.1). Середня точність для набору запитів - це середнє значення середніх показників точності для кожного запиту.

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}, \tag{3.1}$$

де Q - кількість запитів в наборі, а AveP (q) - середня точність (Average Precision) для даного запиту q.

На 7000 ітерацій отримано наступні результати:

- Graspers 98.7%
- Needle Driver 94.2%
- Scissors 95.9%
- Середня точність 96.3%

Результат роботи сценарію навчання моделі виглядає наступним чином.

```
1. Using 'image' as feature column
2. Using GPU to create model (AMD Radeon RX 560)
3. Setting 'batch_size' to 32
4. Setting 'max_iterations' to 7000
5. +--------------+--------------+--------------+
6. | Iteration    | Loss         | Elapsed Time |
7. +--------------+--------------+--------------+
8. | 1            | 7.01145      | 1.23s        |
```

```
| 2            | 7.11772      | 1.91s        |
| 3            | 7.25343      | 2.60s        |
| 4            | 7.28268      | 3.28s        |
| 5            | 7.28381      | 3.96s        |
| 6            | 7.30763      | 4.65s        |
| 11           | 7.29049      | 8.06s        |
...
| 6981         | 0.619058     | 1h 19m       |
| 6986         | 0.638174     | 1h 19m       |
| 6991         | 0.655555     | 1h 19m       |
| 6996         | 0.659116     | 1h 19m       |
| 7000         | 0.621484     | 1h 19m       |
+--------------+--------------+--------------+
{'average_precision_50': {'Graspers': 0.9878501296043396,
   'NeedleDriver': 0.942246675491333, 'Scissors':
   0.9591884016990662}, 'mean_average_precision_50':
   0.9630951285362244}
```

3.9 Тестування моделі на відео в реальному часі

Для тестування моделі було створено мобільний додаток. Тексти основних файлів мобільного додатку можна знайти у додатку Б. Цей додаток за допомогою камери, що вбудовано в мобільний телефон отримує відео у реальному часі, та помічає межі прогнозованих об’єктів, підписує мітку об’єкта та точність його розпізнавання. Приклад процесу можна побачити на рис. 3.8.

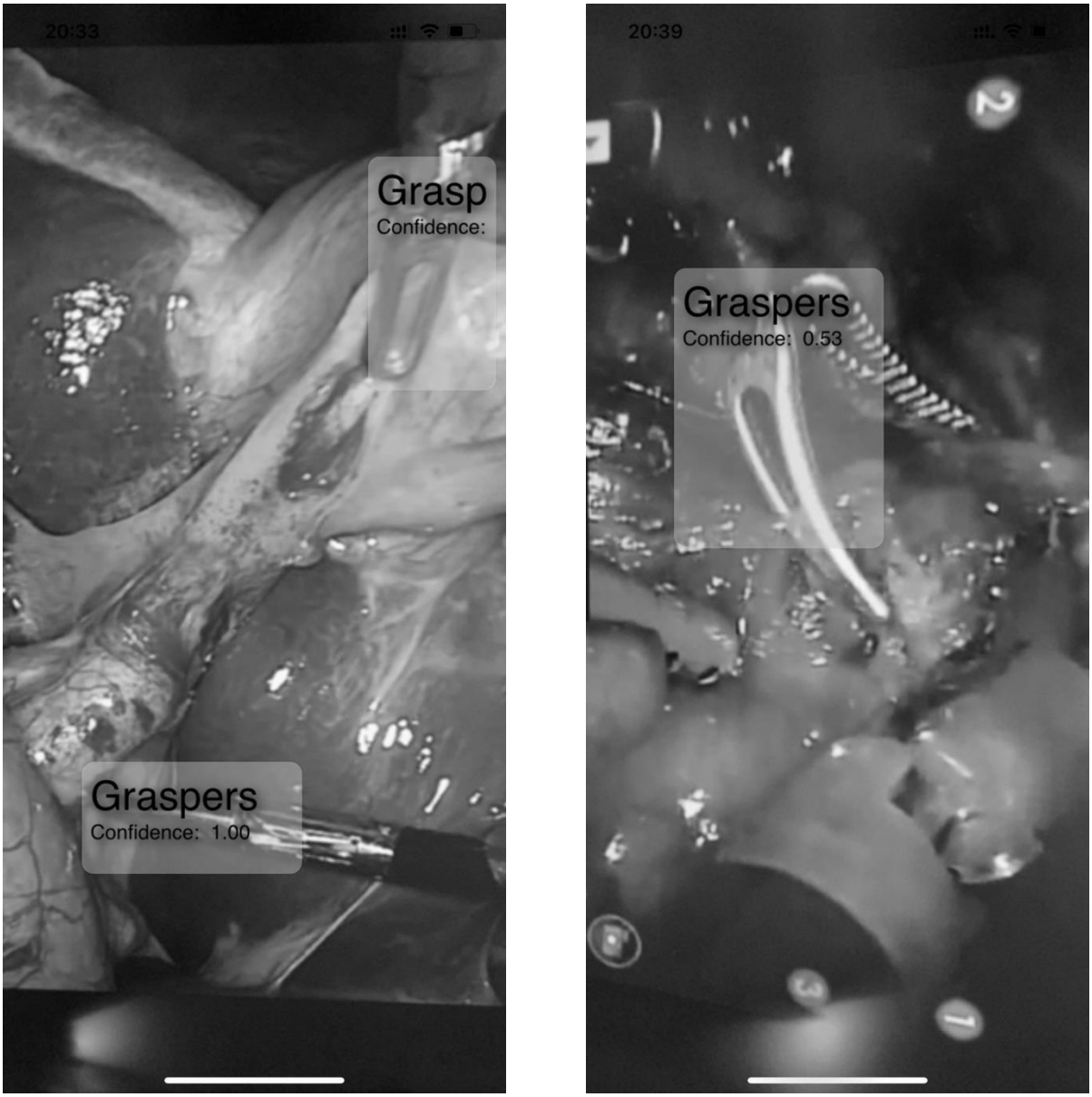

Рисунок 3.8 – Копії екрану мобільного додатку

3.10 Висновок до розділу 3

В третьому розділі представлена практична реалізація методу автоматичного визначення об’єктів відеопотоку в режимі реального часу проведена з реалізацією повного циклу розпізнавання відео, що включає анотацію об’єктів відео та подальше використання анотованих даних відео для визначення об’єктів у відеопотоці. Практичної реалізації впроваджена з використанням відео лапароскопічної хірургічної операції.

Реалізовано анотацію об'єктів у відеопотоці лапароскопічної хірургічної операції з використанням обмежуючої рамки та моделювання даних відеопотоку.

Проведене тестування розробленої моделі та автоматично визначені об’єкти у відеопотоці лапароскопічної хірургічної операції в режимі реального часу.

Реалізована наступна сукупність етапів, що являє собою метод автоматичного визначення об'єктів у відеопотоці в режимі реального часу: визначені параметри апаратного та програмного забезпечення, розгорнуто на сервері OpenCVAT, отримано зображення та відео з об’єктами для навчання моделі, проведено перетворення відео на серію зображень, проведено перетворення серій зображень на файл з відео, створено завдання на анотацію, анотація відеозображень, підготовлено дані та середовище виконання для навчання моделі, протестовано модель на відео в режимі реального часі, оцінено результати тестування та визначено ефективність моделі.

В якості об’єктів анотації визначені інструменти хірургічного комплексу DaVinci. Тестування моделі на відеопотоці має відбуватися за допомогою мобільного додатку, що було розроблено у рамках цієї роботи.

На 7000 ітерацій отримано наступні результати автоматичного визначення об’єктів у відеопотоці в режимі реального часу. Точність визначення мітки Graspers 98.7%, Needle Driver 94.2%, Scissors 95.9%. Середня точність визначення об’єктів складає 96.3%.

Таким чином, вирішено поставлене завдання - підвищення точності автоматичного визначення об’єктів у відеопотоці в режимі реального часу в умовах відсутності розмічених даних та складності сцени відеопотоку шляхом розмітки складної сцени відеопотоку, створення моделі розпізнавання на підставі розмічених даних, дослідження методів і архітектур адаптивного опрацювання відеопотоків, спрямованих на інтелектуальну обробку даних.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних і шкідливих виробничих факторів, причин пожеж. Розглянуто заходи, які дозволяють забезпечити гігієну праці та виробничу санітарію. На підставі аналізу розроблено заходи з техніки безпеки і рекомендації з пожежної профілактики. І оскільки завданням на дипломне проектування є програмне забезпечення, то аналіз потенційно небезпечних і шкідливих виробничих факторів виконується для персонального комп'ютера, на якому передбачається реалізація та розробка системи.

4.1 Аналіз стану умов праці

Для створення системи автоматизації збору інформації достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп'ютером.

Оформлення дипломного проекту з розробки системи автоматизації збору інформації за фізичним навантаженням відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав'язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни. За ступенем нервово-психічної напруги виконання роботи можна віднести до II – III ступеня і кваліфікувати як помірно напружений – напружений за умови успішного виконання поставлених завдань.

Під час виконання робіт використовують ПК та периферійні пристрої (лазерні та струменеві), що призводить до навантаження на окремі системи організму. Такі перекоси у напруженні різних систем організму, що трапляються під час роботи з ПК, зокрема, значна напруженість зорового аналізатора і довготривале малорухоме положення перед екраном, не тільки не зменшують загального напруження, а навпаки, призводять до його посилення і появи стресових реакцій.

Найбільшому ризику виникнення різноманітних порушень піддаються: органи зору, м'язово-скелетна система, нервово-психічна діяльність, репродуктивна функція у жінок.

Тобто наявні психофізіологічні небезпечні та шкідливі фактори:

а) фізичного перевантаження:

- статичного;

- динамічного;

б) нервово-психічного перевантаження:

- розумового перенапруження;

- монотонності праці;

- перенапруження аналізаторів;

- емоційних перевантажень.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи на ведені ДСанПіН 3.3.2.007-98 [38]. Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви: - для розробників програм тривалістю 15 хвилин через кожну годину роботи.

4.2 Аналіз потенційних небезпечних і шкідливих виробничих факторів при роботі з персональним комп'ютером

Основними характеристиками персонального комп'ютера є наступні:

1) робоча напруга *U*, В = 220 ±5%;

2) робочий струм *I*, А = 2;

3) споживана потужність *P*, Вт = 350.

Роботу користувача розробленої підсистеми слід віднести до категорії Іа (легкі фізичні роботи). До даної категорії відносяться всі види діяльності, які виконуються сидячи, з періодичним ходінням, і не потребують фізичного напруження [39].

При експлуатації даного програмного продукту існують такі небезпечні і шкідливі виробничі фактори:

1) фізичні:

- підвищений рівень напруги електричної мережі, замикання якої може статися через тіло людини;
- підвищена або знижена вологість повітря;

— підвищена або знижена рухомість повітря;

— підвищений рівень статичної електрики;

— підвищена напруженість електричного поля;

— відсутність або нестача природного світла;

— знижена освітленість робочої зони;

— підвищений рівень шуму на робочому місці;

— підвищений рівень електромагнітного випромінювання;

— знижена контрастність;

— 2) психофізіологічні;

3) фізичні перевантаження:

— статичні;

— динамічні;

4) нервово-психічні перевантаження:

— розумове перенапруження;

— монотонність праці;

— перенапруження аналізаторів;

— емоційні перевантаження.

4.3 Заходи з охорони праці

4.3.1 Загальні заходи безпеки

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання приклад деяких заходів безпеки:

1. Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;
- експлуатацію сертифікованого обладнання;
- дотримання заходів електробезпеки;

– забезпечення оптимальних параметрів мікроклімату;
– забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);
– облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря:

а) якщо об’єм приміщення 20 м3, то потрібно подати не менш як 30 м3/год повітря;

б) якщо об’єм приміщення у межах від 20 до 40 м3, то потрібно подати не менш як 20 м3/год повітря;

в) якщо об’єм приміщення становить понад 40 м3, допускається природна вентиляція, у випадку, коли немає виділення шкідливих речовин.

– зниження рівня шуму та вібрації:

а) у джерелі виникнення, шляхом застосування раціональних конструкцій, нових матеріалів і технологічних процесів;

б) звукоізолювання устаткування за допомогою глушників, резонаторів, кожухів, захисних конструкцій, оздоблення стін, стелі, підлоги тощо;

в) використання засобів індивідуального захисту.

2. Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

– постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під’єднують до електромережі;
– постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;
– не тягнути за мережевий кабель, щоб витягти вилку з розетки;
– не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;
– не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;
– не залишати включені електроприлади без нагляду;

– не допускати потрапляння всередину електроприладів крізь вентиляційні отвори рідин або металевих предметів, а також не закривати їх та підтримувати в належній чистоті, щоб уникнути перегрівання та займання приладу;

– не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

4.3.2 Електробезпека

Основним небезпечним фактором при роботі з ЕОМ є небезпека ураження людини електричним струмом, яка посилюється тим, що органи чуття людини не можуть на відстані виявити наявність електричної напруги на обладнанні.

Проходячи через тіло людини, електричний струм чинить на нього складний вплив, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (роздратування і збудження нервових волокон та інших органів тканин організму) дій.

Тяжкість ураження людини електричним струмом залежить від цілого ряду чинників:

1) значення сили струму;

2) електричного опору тіла людини і тривалості протікання через нього струму;

3) типу і частоти струму;

4) індивідуальних властивостей людини і навколишнього середовища.

Приміщення для ЕОМ відноситься до приміщень без підвищеної небезпеки, тобто в приміщення, в яких відсутні умови, що створюють підвищену або особливу небезпеку. Небезпека ураження електричним струмом існує всюди, де використовуються електроустановки, тому приміщення без підвищеної небезпеки не можна назвати безпечними.

Електробезпека забезпечується:

1) відповідною конструкцією електроустановок;

2) застосуванням технічних способів і засобів захисту;

3) організаційними і технічними заходами.

Конструкція електроустановок відповідає умовам їх експлуатації та забезпечує захист персоналу від дотику до струмоведучих частин.

Основними технічними способами і засобами захисту від ураження електричним струмом, що використовуються окремо або в поєднанні один з одним, є:

1) захисне заземлення;

2) занулення;

3) вирівнювання потенціалів;

4) мале напруга;

5) електричне поділ мереж;

6) захисне відключення;

7) ізоляція струмоведучих частин;

8) компенсація струмів замикання на землю;

9) захисні пристрої;

10) попереджувальна сигналізація, блокування, знаки безпеки;

11) ізолюючі захисні та запобіжні пристосування.

4.3.3 Розрахунок захисного заземлення

Основними технічними способами і засобами захисту від ураження електричним струмом, що передбачаються в даному дипломному проекті, є:

1) захисне заземлення,

2) занулення,

3) захисне відключення,

4) ізоляція струмоведучих частин.

Завдання захисного заземлення - усунення небезпеки ураження струмом у випадку дотику до корпусу та інших струмоведучих металевих частин електроустановок, які опинилися під напругою.

Розрахунок заземлюючого контуру виконується виходячи з умови:

$$R_{зу} = \frac{R_з \cdot R_п}{R_п \cdot n \cdot \eta_з + R_з \cdot \eta_п} \leq 4 \quad Ом \tag{4.1}$$

де $R_З$ - опір заземлювача (стержня, труби, куточка і т.д.), Ом;

$R_П$ - опір лінії, що з'єднує заземлювачі, Ом;

n - кількість заземлювачів;

$\eta_З$ і $\eta_П$ - коефіцієнти екранування відповідно заземлювача і з'єднує смуги ($\eta_З = 0.2 \div 0.9$; $\eta_П = 0.1 \div 0.7$).

Опір заземлювача розраховується за формулою 4.2

$$R_З = \frac{\rho}{2 \cdot \pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + \frac{1}{2} \cdot \ln\frac{4 \cdot t + l}{4 \cdot t - l}\right), \quad (4.2)$$

де ρ - питомий опір ґрунту (взяти з довідкової літератури);

l - довжина заземлювача (для труб 2-3м, для стрижнів до 10м), м;

d - діаметр заземлювача (для стрижнів 0.01 – 0.03м, для труб 0.03 – 0.05м);

t - відстань від середини забитого в ґрунт заземлювача до рівня землі (необхідно враховувати, що відстань від верхнього кінця заземлювача до поверхні землі має бути не менше 0.5), м.

Розрахуємо опір заземлювача:

$$R_З = \frac{60}{2 \cdot \pi \cdot 3}\left(\ln\frac{2 \cdot 3}{0.03} + \frac{1}{2} \cdot \ln\frac{4 \cdot 1 + 3}{4 \cdot 1 - 3}\right) = 19{,}96 \quad (4.3)$$

Опір лінії, що з'єднує заземлювачі розраховується за формулою 4.4

$$R_П = \frac{\rho}{2 \cdot \pi \cdot l} \cdot \ln\frac{2 \cdot L^2}{b \cdot t}, \quad (4.4)$$

де *L* - довжина лінії, що з'єднує заземлювачі (при контурному заземленні вона приблизно дорівнює периметру виробничої будівлі), м;

b - ширина смуги (0.03 - при прокладанні всередині будівлі і 0.05 - при прокладанні поза будівлею), м;

t - глибина заземлення від рівня землі (0.5 м).

Розрахуємо опір лінії, що з'єднує заземлювачі

$$R_{\Pi} = \frac{60}{2 \cdot \pi \cdot 3} \cdot \ln \frac{2 \cdot 50^2}{0.03 \cdot 0.5} = 14.37 \quad (4.5)$$

Необхідна кількість заземлювачів, розраховується за формулою 4.6

$$n = \frac{2 \cdot R_з}{4 \cdot \eta_з}, \quad (4.6)$$

де 4 - допустимий загальний опір;

2 - коефіцієнт сезонності.

Розрахуємо необхідну кількість заземлювачів,

$$n = \frac{2 \cdot 19,9}{4 \cdot 0.5} = 19,9 \approx 20 \quad (4.7)$$

Округлимо результат в більшу сторону і отримуємо необхідну кількість заземлювачів - 20. Маючи всі необхідні дані розрахуємо опір заземлюючого контуру.

$$R_{зу} = \frac{19.96 \cdot 14,37}{14,37 \cdot 20 \cdot 0,5 + 19.96 \cdot 0,4} = 1,89 \leq 4 \ \ Ом \quad (4.8)$$

Опір заземлюючого контуру 1.89Ом, що відповідає умові Rзу < 4Ом.

4.4 Заходи, що забезпечують виробничу санітарію та гігієну праці

4.4.1 Мікроклімат

Важкість праці характеризує сукупну дію всіх елементів, складових умови праці, на працездатність людини, його здоров'я, життєдіяльність і відновлення робочої сили. У такому представлені поняття тяжкості праці однаково застосовні як до розумової, так і до

фізичної праці. Згідно [40] тяжкість роботи персоналу, який обслуговує ЕОМ, відноситься до легкої категорії 1б (роботи, виконувані сидячи, не вимагаючи систематичного фізичного напруження і перенесення важких предметів) [40]. Загальні санітарно-гігієнічні вимоги до повітря робочої зони. Оптимальні норми мікроклімату в робочій зоні, що забезпечуються для робіт легкої категорії 1б приведені в таблиці 4.1.

Таблиця 4.1 - Оптимальні норми мікроклімату

Період рок	Температура, ^{o}C	Відносна вологість,%	Швидкість вітру, м/с, не більше
Холодний	21 - 23	60 – 40	0.1
Теплий	22 - 24	60 - 40	0.2

4.4.2 Освітлення

Світло є природною умовою існування людини . Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Гарне освітлення діє тонізуюче, створює гарний настрій, поліпшує протікання основних процесів вищої нервової діяльності.

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків.

Штучне освітлення в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла у світильниках загального освітлення, оскільки люмінесцентні лампи мають високу потужність (80 Вт), тривалий термін служби (до 10000 годин), спектральний складом випромінюваного світла, близький до сонячного. При експлуатації ЕОМ виконується зорова робота IV в розряд точності (середня точність). При цьому нормована освітленість на робочому місці (Ен) дорівнює 200 лк. Джерелом природного освітлення є сонячне світло. У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В.2.5-28:2018 [41]

Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє [41]

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \times S_n \quad (4.9)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \times b = 4 \times 5 = 20$ м^2 ,

$S = 1/8 \times 20 = 2.5$м^2 .

Приймаємо 1 вікно площею S = 2.5м^2. Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 4м, ширина 4м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80Вт) з світловим потоком 5400лм кожна. Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні.

Розрахунок кількості світильників n визначається по формулі (4.10):

$$n = \frac{E \times S \times Z \times K}{F \times U \times M} \quad (4.10)$$

де *E* – нормована освітленість робочої поверхні, визначається нормами – 300лк;

S – освітлювана площа, м^2; S = 20 м^2;

Z – поправочний коефіцієнт світильника Z = 1.15 для ламп розжарювання та ДРЛ;

Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1.1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1.5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0.575;

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.10), отримуємо:

$$n=\frac{300\times20\times1.1\times1.5}{5400\times0.575\times2}\approx1.6 \quad (4.11)$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160Вт, напругою 220В.

4.5 Рекомендації щодо пожежної безпеки

Виникнення пожежі можливо, якщо на об'єкті є горючі речовини, окиснювач і джерела запалювання. Для оцінки пожежної небезпеки слід проаналізувати ймовірність взаємодії цих трьох чинників.

Горючими матеріалами в приміщенні, де розташовані ЕОМ, є:

1) поліамід - матеріал корпусу мікросхем, горюча речовина, температура самозаймання 420 °С;

2) полівінілхлорид - ізоляційний матеріал, горюча речовина, температура запалювання 335 °С, те

3) склотекстоліт ДЦ - матеріал друкованих плат, трудногорючий матеріал, показник горючості 1.74, не схильний до температурного самозаймання;

4) пластикат кабельний No.489 - матеріал ізоляції кабелів, горючий матеріал, показник горючості більше 2.1;

5) деревина - будівельний і оздоблювальний матеріал, з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, температура запалювання 255 °С, температура самозаймання 399 °С.

Згідно НАПБ Б. 03.002-2007 таке приміщення належить до категорії "В" (пожежонебезпечної) [42].

Простору всередині приміщень в межах яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали у відповідності з ПУЕ [43] відносяться до пожежонебезпечної зони класу П-IIа.

Потенційними джерелами запалювання можуть бути:

1) іскри і дуги короткого замикання;

2) електрична іскра при замиканні і розмиканні ланцюгів;

3) перегріви від тривалого перевантаження;

4) відкритий вогонь і продукти горіння;

5) наявність речовин, нагрітих вище температури самозаймання;

6) розряд на статичну електрику.

Причинами можливого загоряння і пожежі можуть бути:

1) несправність електроустановки;

2) конструктивні недоліки обладнання;

3) коротке замикання в електричних мережах;

4) запалювання горючих матеріалів, що знаходяться в безпосередній близькості від електроустановки.

Продуктами згоряння, що виділяються під час пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор та ін.

При горінні пластмас, крім звичних продуктів згоряння, виділяються різні продукти термічного розкладання: хлорангідрідні кислоти; формальдегіди; хлористий водень; фосген; синильна кислота; аміак; фенол; ацетон; стирол. Пожежо-вибухонебезпечність речовин і матеріалів. Номенклатура показників і методи їх визначення ГОСТ 12.1.044 - 89 ЕСБТ [44].

Для захисту персоналу від впливу небезпечних і шкідливих факторів пожежі проектом передбачається застосування промислового протигаза фільтруючого з коробкою марки В (жовтий).

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем вентиляції та кондиціонування, розвиненою системою електроживлення ЕОМ. Небезпека загорання в ЕОМ пов'язана з великою кількістю щільно розташованих на платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів і транзисторів. Висока щільність елементів в електронних схемах призводить до значного підвищення температури окремих вузлів (80 ... 100 °С), що може служити причиною запалювання ізоляційних матеріалів. Слабкий опір ізоляційних матеріалів дії температури може викликати порушення ізоляції і привести до короткого замикання.

Пожежна безпека при застосуванні ЕОМ забезпечується:

1) системою запобігання пожежі:

2) системою протипожежного захисту:

3) організаційно-технічними заходами.

Запобігти утворенню горючого середовища (замінити горючі речовини і матеріали на негорючі та важкогорючі) не надається технічно можливим. Тому проектом передбачаються способи і засоби запобігання утворенню (або внесення) в горюче середовище джерел запалювання, таких як:

1) застосування електроустаткування, відповідної пожежонебезпечної і вибухонебезпечної зонами відповідно до ПУЕ;

2) застосування в конструкції швидкодіючих засобів захисного відключення можливих джерел запалювання;

3) виключення можливості появи іскрового розряду в займистою середовищі з енергією, яка дорівнює і вище мінімальної енергії запалювання.

4.6 Екологія

Діяльність на тему магістерської роботи, а саме: розробка методів та засобів підвищення реалістичності відтворення кольорів у засобах комп’ютерної графіки в процесі її виконання, впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища» [45], Законом України «Про забезпечення санітарного та епідемічного благополуччя населення» [46], Законом України «Про відходи» [47].

В процесі розробки виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- відпрацьовані люмінесцентні лампи - I клас небезпеки.
- змінні носії інформації - IV клас небезпеки.
- відпрацьовані вогнегасники - IV клас небезпеки.
- макулатура - IV клас небезпеки.

4.7 Висновки до розділу 4

У розділі "Охорона праці" виконаний аналіз потенційних небезпек при роботі із засобами обчислювальної техніки, на підставі якого розроблено заходи з техніки безпеки, заходи, що забезпечують виробничу санітарію та гігієну праці, розрахунки природного та

штучного освітлень, рекомендації з пожежної профілактики, які підтверджені відповідними розрахунками. Також було проведено аналіз впливу відходів з ІТ галузі на навколишнє природнє середовище.

ВИСНОВКИ

Метою дипломної роботи визначено підвищення точності автоматичного визначення об’єктів у відеопотоці в режимі реального часу в умовах відсутності анотованих даних та складності сцени відеопотоку шляхом розмітки складної сцени відеопотоку, створення моделі розпізнавання на підставі анотованих даних, дослідження методів і архітектур адаптивного опрацювання відеопотоків, спрямованих на інтелектуальну обробку даних.

В ході дослідницької частини роботи були отримані наступні результати:

1. Проведено аналіз методів і технологій анотування відео та визначення об'єктів у відеопотоці в режимі реального часу.
2. Визначено засоб анотування об’єктів у відеопотоці, методу подальшого автоматичного вилучення і використання цифрових даних відеопотоку в режимі реального часу.
3. Досліджено процес анотування відеозображень з використанням технології глибокого навчання та нейронних мереж.

В ході практичної частини роботи були отримані наступні результати:

1. Реалізовано етап анотування даних відеопотоку з використанням розгорнутого на сервері вільного програмного забезпечення OpenCVAT.
2. Розроблено програмний засіб для адаптації моделі даних відеопотоку з метою її подальшого використання.
3. Розроблено мобільний додаток для тестування моделі в режимі реального часу.
4. Розроблено та протестовано модель визначення об’єктів шляхом проведення експерименту розмітки даних та визначення об'єктів у відеопотоці в режимі реального часу.
5. Визначено ефективність методу автоматичного визначення об’єктів у відеопотоці в режимі реального часу.

Використаний критерій якості моделі Mean Average Precision (MAP) на 7000 ітерацій склав 96.3% точності моделі, що є високим показником ефективності розпізнавання.

Таким чином, вирішено поставлене завдання - підвищено точність автоматичного визначення об’єктів у відеопотоці в режимі реального часу в умовах відсутності розмічених даних та складності сцени відеопотоку шляхом розмітки складної сцени відеопотоку, створення моделі розпізнавання на підставі розмічених даних, досліджено методи і архітектури адаптивного опрацювання відеопотоків, спрямовані на інтелектуальну обробку даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. Object Detection Guide - [Електронний ресурс]. – Режим доступу: URL: https://www.fritz.ai/object-detection/ - 25.12.2019 р.
2. Unsupervised Object Detection - [Електронний ресурс]. – Режим доступу: URL: https://habr.com/ru/post/463991/- 25.12.2019 р.
3. D. G. Lowe, Distinctive image features from scale-invariant keypoints, in: IJCV, 2004.
4. T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, in: TPAMI, 2002.
5. A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: NeurIPS, 2012.
6. K.He, X.Zhang, S.Ren, J.Sun, Deep residual learning for image recognition, in: CVPR, 2016.
7. S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time ob- ject detection with region proposal networks, in: NeurIPS, 2015.
8. R.Girshick, J.Donahue, T.Darrell, J.Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: CVPR, 2014.
9. R. Girshick, Fast r-cnn, in: ICCV, 2015.
10. T.-Y. Lin, P. Dolla ÃÅr, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: CVPR, 2017.
11. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: CVPR, 2016.
12. J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, in: CVPR, 2017.
13. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, SSD: Single shot multibox detector, in: ECCV, 2016.
14. Коефіцієнт Жаккара - [Електронний ресурс]. – Режим доступу: URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B5%D1%84%D1%96%D1%86%D1%96%D1%94%D0%BD%D1%82_%D0%96%D0%B0%D0%BA%D0%BA%D0%B0%D1%80%D0%B0
15. J. R. Uijlings, K. E. Van De Sande, T. Gevers, A. W. Smeulders, Selective search for object recognition, in: IJCV, 2013.

16. J. Kleban, X. Xie, W.-Y. Ma, Spatial pyramid mining for logo detection in natural scenes, in: Multimedia and Expo, 2008 IEEE International Conference on, 2008.
17. K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, in: ECCV, 2014.
18. Spatial pyramid pooling in deep convolutional networks for visual recognition - [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/1406.4729
19. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, Overfeat: Integrated recognition, localization and detection using convolutional networks, in: arXiv preprint arXiv:1312.6229.
20. T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dolla ÃÅr, Focal loss for dense object detection, in: ICCV, 2017.
21. H. Law, J. Deng, Cornernet: Detecting objects as paired keypoints, in: ECCV, 2018.
22. B. Alexe, T. Deselaers, V. Ferrari, Measuring the objectness of image windows, in: TPAMI, 2012.
23. P. F. Felzenszwalb, D. P. Huttenlocher, Efficient graph-based image segmentation, in: IJCV, 2004.
24. S. Manen, M. Guillaumin, L. Van Gool, Prime object proposals with randomized prim's algorithm, in: CVPR, 2013.
25. A. Ghodrati, A. Diba, M. Pedersoli, T. Tuytelaars, L. Van Gool, Deep-proposal: Hunting objects by cascading deep convolutional layers, in: ICCV, 2015.
26. L. Tychsen-Smith, L. Petersson, Denet: Scalable real-time object detection with directed sparse sampling, in: ICCV, 2017.
27. C. Zhu, Y. He, M. Savvides, Feature selective anchor-free module for single-shot object detection, in: CVPR, 2019.
28. X. Zhou, D. Wang, P. Kra Ãàhenbu Ãàhl, Objects as points, in: arXiv preprint arXiv:1904.07850, 2019.
29. Y. Lu, T. Javidi, S. Lazebnik, Adaptive object detection using adjacency and zoom prediction, in: CVPR, 2016.
30. New Computer Vision Tool Accelerates Annotation of Digital Images and Video - [Електронний ресурс]. – Точка доступу: URL: https://www.intel.ai/introducing-cvat/#gs.irfx8h - 01.11.2019 р.

31. Faster R-CNN: Towards Real-Time ObjectDetection with Region Proposal Networks - [Електронний ресурс]. – Точка доступу: URL: https://arxiv.org/pdf/1506.01497.pdf - 01.11.2019 р.
32. Faster R-CNN Explained - [Електронний ресурс]. – Режим доступу: URL: http://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e38 - 01.11.2019р.
33. Turi Create - [Електронний ресурс]. – Режим доступу: URL: https://github.com/apple/turicreate - 01.11.2019 р.
34. da Vinci Robot Stitches a Grape Back Together - [Електронний ресурс]. – Режим доступу: URL: https://www.youtube.com/watch?v=0XdC1HUp-rU - 01.11.2019 р.
35. Inferior Vena Cava (IVC) Thrombectomy - da Vinci Robotic Surgery - [Електронний ресурс]. – Режим доступу: URL: https://www.youtube.com/watch?v=5JM8KhWhrus - 01.11.2019 р.
36. Laparoscopic Cholecystectomy - [Електронний ресурс]. – Режим доступу: URL: https://www.youtube.com/watch?v=n18zxNGJdLE
37. TuriCreate User Guide: Object Detection - [Електронний ресурс]. – Точка доступу: URL: https:// apple.github.io/turicreate/docs/userguide/object_detection/ - 01.11.2019 р.
38. ДСанПіН 3.3.2.007-98 “Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.” - [Електронний ресурс]. – Режим доступу: URL: https://dnaop.com/html/40949/doc-%D0%94%D0%A1%D0%B0%D0%9D%D0%9F%D1%96%D0%9D_3.3.2-007-98/ - 20.10.2019 р.
39. НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями. [Електронний ресурс]. – Режим доступу: URL: https://zakon.rada.gov.ua/laws/show/z0508-18 - 01.11.2019 р.
40. ГОСТ 12.1.005-88 "ССБТ Загальні санітарно-гігієнічні вимоги до повітря робочої зони" - [Електронний ресурс]. – Режим доступу: URL: https://legalacts.ru/doc/gost-121005-88-mezhgosudarstvennyi-standart-sistema-standartov-bezopasnosti/ - 20.10.2019 р.
41. ДБН В.2.5-28:2018 “Природне і штучне освітлення.” - [Електронний ресурс]. – Режим доступу: URL: https://dbn.co.ua/load/normativy/dbn/dbn_v_2_5_28/1-1-0-1188 - 20.10.2019 р.

42. НАПБ Б.03.002-2007 “Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою.” - [Електронний ресурс]. – Режим доступу: URL: https://dnaop.com/html/32350/doc-%D0%9D%D0%90%D0%9F%D0%91_%D0%91.03.002-2007/ - 01.11.2019 р.
43. НПАОП 40.1-1.32-01 (ДНАОП 0.00-1.32-01) Правила будови електроустановок.
44. ГОСТ 12.1.044-89 “ССБТ. ГОСТ 12.1.044-89 ССБТ. Пожаровзрывоопасность веществ и материалов. Номенклатура показателей и методы их определения.” - [Електронний ресурс]. – Режим доступу: URL: http://docs.cntd.ru/document/gost-12-1-044-89 - 20.10.2019 р.
45. Закон України «Про охорону навколишнього природного середовища» - [Електронний ресурс]. – Режим доступу: URL: https://zakon.rada.gov.ua/laws/main/1264-12 - 01.11.2019 р.
46. Закон України «Про забезпечення санітарного та епідемічного благополуччя населення» - https://zakon.rada.gov.ua/laws/main/4004-12- 01.11.2019 р.
47. Закон України «Про відходи» - [Електронний ресурс]. – Режим доступу: URL: https://zakon.rada.gov.ua/laws/main/187/98-%D0%B2%D1%80 - 01.11.2019 р.

ДОДАТОК А СЦЕНАРІЙ ОТРИМАННЯ МОДЕЛІ

```
1.  #!./.local/bin/python
2.
3. import os
4. import sys
5. from collections import defaultdict
6. from math import ceil
7.
8. import requests
9. import turicreate as tc
10.import untangle
11.from dotenv import load_dotenv
12.
13.load_dotenv()
14.
15.BASE_URL = os.getenv('CVAT_BASE_URL')
16.ROOT_DIR = os.path.dirname(os.path.abspath(sys.argv[0]))
17.MEDIA_DIR = os.path.join(ROOT_DIR, 'frames')
18.
19.try:
20.    os.mkdir(MEDIA_DIR)
21.except FileExistsError:
22.    pass
23.
24.
25.class CvatApiClient(object):
26.    def __init__(self, base_url=None, media_dir=None):
27.        self.base_url = base_url or BASE_URL
28.        self.media_dir = media_dir or MEDIA_DIR
29.        try:
30.            os.mkdir(self.media_dir)
31.        except FileExistsError:
32.            pass
33.
34.    def fetch_task_frames(self, task_id, frames=None, first=0, last=0):
35.        if frames is None:
```

```
            frames = range(first, last)
        for index in frames:
            try:
                response = requests.get(
                    url="{}/api/v1/tasks/{}/frames/{}".format(
                        self.base_url, task_id, index),
                    headers={
                        "Connection": "keep-alive",
                        "User-Agent": (
                            "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_1) "
                            "AppleWebKit/537.36 (KHTML, like Gecko) "
                            "Chrome/81.0.4018.0 Safari/537.36"
                        ),
                        "Dnt": "1",
                        "Accept": "image/webp,image/apng,image/*,*/*;q=0.8",
                        "Cookie": "sessionid=re2j6rp1199utxqpvgw2",
                        "Accept-Encoding": "gzip",
                    },
                )
                response.raise_for_status()
                print('Frame {index}:\tOK'.format(index=index))
                with open('{}/frame-{:04d}.jpg'.format(
                        self.media_dir, 1 + index), 'wb') as handle:
                    for block in response.iter_content(1024):
                        handle.write(block)
            except requests.exceptions.RequestException:
                return


if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('CVAT .XML file is required')
        exit(1)
    obj = untangle.parse(sys.argv[1])

    frames = defaultdict(list)
```

```
    frame_numbers = []

    for image in getattr(obj.annotations, 'image', []):
        frame_number = int(image.__dict__['_attributes']['id'])
        frame_numbers.append(frame_number)
        for box in image.box:
            attrs = box.__dict__['_attributes']
            label = attrs['label']
            xtl = float(attrs['xtl'])
            ytl = float(attrs['ytl'])
            xbr = float(attrs['xbr'])
            ybr = float(attrs['ybr'])
            width = xbr - xtl
            height = ybr - ytl
            center_x = xtl + width / 2
            center_y = ytl + height / 2
            frame_name = 'frame-{:04d}'.format(frame_number)
            frames[frame_name].append(dict(label=label, coordinates=dict(
                x=int(center_x), y=int(center_y),
                width=ceil(width), height=ceil(height))))

    for track in getattr(obj.annotations, 'track', []):
        label = track.__dict__['_attributes']['label']
        if hasattr(track, 'box'):
            for box in track.box:
                attrs = box.__dict__['_attributes']
                frame_number = int(attrs['frame'])
                frame_numbers.append(frame_number)
                 xtl = float(attrs['xtl'])
                 ytl = float(attrs['ytl'])
                 xbr = float(attrs['xbr'])
                 ybr = float(attrs['ybr'])
                 width = xbr - xtl
                 height = ybr - ytl
                 center_x = xtl + width / 2
                 center_y = ytl + height / 2
                 frame_name = 'frame-{:04d}'.format(frame_number)
```

```
109.                 frames[frame_name].append(dict(label=label,
   coordinates=dict(
110.                     x=int(center_x), y=int(center_y),
111.                     width=ceil(width), height=ceil(height))))
112.
113.     model_name = obj.annotations.meta.task.name.cdata
114.     task_id = int(obj.annotations.meta.task.id.cdata)
115.     first_frame = int(obj.annotations.meta.task.start_frame.cdata)
116.     size = int(obj.annotations.meta.task.size.cdata)
117.     last_frame = first_frame + size - 1
118.     print('Task ID: {}. Frames: {}..{}'.format(
119.         task_id, first_frame, last_frame))
120.     model_media_dir = os.path.join(MEDIA_DIR, model_name)
121.     print('Model media: {}'.format(model_media_dir))
122.     client = CvatApiClient(base_url=BASE_URL, media_dir=model_media_dir)
123.     client.fetch_task_frames(task_id, frames=set(frame_numbers))
124.
125.     annotations = tc.SFrame(
126.         dict(frame=frames.keys(), annotation=frames.values()))
127.     annotations.save("{}_annotations.sframe".format(model_name))
128.     images = tc.load_images(model_media_dir, recursive=False)
129.     # We know how the frame file name is constructed, so we can cut
130.     # the corner here.
131.     images['frame'] = images['path'].element_slice(-14, -4)
132.     data = images.join(annotations, on='frame')
133.     data.save("{}.sframe".format(model_name))
134.     data.explore()
135.     input("Press Enter to continue...")
136.
137.     train, test = data.random_split(0.8)
138.     model = tc.object_detector.create(train, "annotation")
139.     metrics = model.evaluate(test)
140.     print(metrics)
141.     model.export_coreml('{}.mlmodel'.format(model_name))
```

ДОДАТОК Б МОБІЛЬНИЙ ДОДАТОК

```
1. // File: ViewController.swift
2. import AVFoundation
3. import UIKit
4. import Vision
5.
6. class ViewController:
7.         UIViewController, AVCaptureVideoDataOutputSampleBufferDelegate {
8.     var bufferSize: CGSize = .zero
9.     var rootLayer: CALayer!
10.
11.    @IBOutlet private var previewView: UIView!
12.    private let session = AVCaptureSession()
13.    private var previewLayer: AVCaptureVideoPreviewLayer!
14.    private let videoDataOutput = AVCaptureVideoDataOutput()
15.
16.    private let videoDataOutputQueue = DispatchQueue(
17.        label: "VideoDataOutput", qos: .userInitiated,
18.        attributes: [], autoreleaseFrequency: .workItem)
19.
20.    func captureOutput(
21.            _: AVCaptureOutput, didOutput _: CMSampleBuffer,
22.            from _: AVCaptureConnection) {
23.    }
24.
25.    override func viewDidLoad() {
26.        super.viewDidLoad()
27.        setupAVCapture()
28.    }
29.
30.    func setupAVCapture() {
31.        var deviceInput: AVCaptureDeviceInput!
32.
33.        // Select a video device, make an input
34.        let videoDevice = AVCaptureDevice.DiscoverySession(
35.            deviceTypes: [.builtInWideAngleCamera], mediaType: .video,
```

```
36.             position: .back).devices.first
37.         do {
38.             deviceInput = try AVCaptureDeviceInput(device: videoDevice!)
39.         } catch {
40.             print("Could not create video device input: \(error)")
41.             return
42.         }
43.
44.         session.beginConfiguration()
45.         session.sessionPreset = .vga640x480
46.
47.         guard session.canAddInput(deviceInput) else {
48.             print("Could not add video device input to the session")
49.             session.commitConfiguration()
50.             return
51.         }
52.         session.addInput(deviceInput)
53.         if session.canAddOutput(videoDataOutput) {
54.             session.addOutput(videoDataOutput)
55.             // Add a video data output
56.             videoDataOutput.alwaysDiscardsLateVideoFrames = true
57.             videoDataOutput.videoSettings = [
58.                 kCVPixelBufferPixelFormatTypeKey as String:
59.                 Int(kCVPixelFormatType_420YpCbCr8BiPlanarFullRange)
60.             ]
61.             videoDataOutput.setSampleBufferDelegate(
62.                 self, queue: videoDataOutputQueue)
63.         } else {
64.             print("Could not add video data output to the session")
65.             session.commitConfiguration()
66.             return
67.         }
68.         let captureConnection = videoDataOutput.connection(with: .video)
69.         captureConnection?.isEnabled = true
70.         do {
71.             try videoDevice!.lockForConfiguration()
72.             let dimensions = CMVideoFormatDescriptionGetDimensions(
73.                 (videoDevice?.activeFormat.formatDescription)!)
```

```
74.             bufferSize.width = CGFloat(dimensions.width)
75.             bufferSize.height = CGFloat(dimensions.height)
76.             videoDevice!.unlockForConfiguration()
77.         } catch {
78.             print(error)
79.         }
80.         session.commitConfiguration()
81.         previewLayer = AVCaptureVideoPreviewLayer(session: session)
82.         previewLayer.videoGravity = AVLayerVideoGravity.resizeAspectFill
83.         rootLayer = previewView.layer
84.         previewLayer.frame = rootLayer.bounds
85.         rootLayer.addSublayer(previewLayer)
86.     }
87.
88.     func startCaptureSession() {
89.         session.startRunning()
90.     }
91.
92.     // Clean up capture setup
93.     func teardownAVCapture() {
94.         previewLayer.removeFromSuperlayer()
95.         previewLayer = nil
96.     }
97.
98.     func captureOutput(
99.         _: AVCaptureOutput, didDrop _: CMSampleBuffer,
100.         from _: AVCaptureConnection) {
101.     }
102.
103.     public func exifOrientationFromDeviceOrientation()
104.             -> CGImagePropertyOrientation {
105.         let curDeviceOrientation = UIDevice.current.orientation
106.         let exifOrientation: CGImagePropertyOrientation
107.
108.         switch curDeviceOrientation {
109.         // Device oriented vertically, home button on the top
110.         case UIDeviceOrientation.portraitUpsideDown:
111.             exifOrientation = .left
```

```
        // Device oriented horizontally, home button on the right
        case UIDeviceOrientation.landscapeLeft:
            exifOrientation = .upMirrored
        // Device oriented horizontally, home button on the left
        case UIDeviceOrientation.landscapeRight:
            exifOrientation = .down
        // Device oriented vertically, home button on the bottom
        case UIDeviceOrientation.portrait:
            exifOrientation = .up
        default:
            exifOrientation = .up
        }
        return exifOrientation
    }
}
```

```
// ObjectRecognitionViewContrller.swift
import UIKit
import AVFoundation
import Vision

class VisionObjectRecognitionViewController: ViewController {

    private var detectionOverlay: CALayer! = nil

    // Vision parts
    private var requests = [VNRequest]()

    @discardableResult
    func setupVision() -> NSError? {
        // Setup Vision parts
        let error: NSError! = nil

        guard let modelURL = Bundle.main.url(
            forResource: "ObjectDetector", withExtension: "mlmodelc")
        else {
            return NSError(
```

```
22.                 domain: "VisionObjectRecognitionViewController",
23.                 code: -1, userInfo: [
24.                     NSLocalizedDescriptionKey: "Model file is missing"
25.                 ])
26.         }
27.         do {
28.             let visionModel = try VNCoreMLModel(
29.                 for: MLModel(contentsOf: modelURL))
30.             let objectRecognition = VNCoreMLRequest(
31.                 model: visionModel, completionHandler: {
32.                     (request, error) in
33.                 DispatchQueue.main.async(execute: {
34.                     // perform all the UI updates on the main queue
35.                     if let results = request.results {
36.                         self.drawVisionRequestResults(results)
37.                     }
38.                 })
39.             })
40.             self.requests = [objectRecognition]
41.         } catch let error as NSError {
42.             print("Model loading went wrong: \(error)")
43.         }
44.
45.         return error
46.     }
47.
48.     func drawVisionRequestResults(_ results: [Any]) {
49.         CATransaction.begin()
50.         CATransaction.setValue(
51.             kCFBooleanTrue, forKey: kCATransactionDisableActions)
52.         // remove all the old recognized objects
53.         detectionOverlay.sublayers = nil
54.         for observation in results
55.                 where observation is VNRecognizedObjectObservation {
56.             guard let objectObservation =
57.                     observation as? VNRecognizedObjectObservation else {
58.                 continue
59.             }
```

```
60.             // Select only the label with the highest confidence.
61.             let topLabelObservation = objectObservation.labels[0]
62.             let objectBounds = VNImageRectForNormalizedRect(
63.                 objectObservation.boundingBox, Int(bufferSize.width),
64.                 Int(bufferSize.height))
65.             let shapeLayer =
66.                 self.createRoundedRectLayerWithBounds(objectBounds)
67.             let textLayer = self.createTextSubLayerInBounds(
68.                 objectBounds, identifier: topLabelObservation.identifier,
69.                 confidence: topLabelObservation.confidence)
70.             shapeLayer.addSublayer(textLayer)
71.             detectionOverlay.addSublayer(shapeLayer)
72.         }
73.         self.updateLayerGeometry()
74.         CATransaction.commit()
75.     }
76.
77.     override func captureOutput(
78.             _ output: AVCaptureOutput, didOutput sampleBuffer:
    CMSampleBuffer,
79.             from connection: AVCaptureConnection) {
80.         guard let pixelBuffer = CMSampleBufferGetImageBuffer(sampleBuffer)
81.         else { return }
82.         let exifOrientation = exifOrientationFromDeviceOrientation()
83.         let imageRequestHandler = VNImageRequestHandler(
84.             cvPixelBuffer: pixelBuffer, orientation: exifOrientation,
85.             options: [:])
86.         do {
87.             try imageRequestHandler.perform(self.requests)
88.         } catch {
89.             print(error)
90.         }
91.     }
92.
93.     override func setupAVCapture() {
94.         super.setupAVCapture()
95.         // setup Vision parts
96.         setupLayers()
```

```
97.         updateLayerGeometry()
98.         setupVision()
99.         // start the capture
100.         startCaptureSession()
101.     }
102.
103.     func setupLayers() {
104.     // container layer that has all the renderings of the observations
105.         detectionOverlay = CALayer()
106.         detectionOverlay.name = "DetectionOverlay"
107.         detectionOverlay.bounds = CGRect(
108.         x: 0.0, y: 0.0, width: bufferSize.width, height:
    bufferSize.height)
109.         detectionOverlay.position = CGPoint(
110.             x: rootLayer.bounds.midX, y: rootLayer.bounds.midY)
111.         rootLayer.addSublayer(detectionOverlay)
112.     }
113.
114.     func updateLayerGeometry() {
115.         let bounds = rootLayer.bounds
116.         var scale: CGFloat
117.
118.         let xScale: CGFloat = bounds.size.width / bufferSize.height
119.         let yScale: CGFloat = bounds.size.height / bufferSize.width
120.
121.         scale = fmax(xScale, yScale)
122.         if scale.isInfinite {
123.             scale = 1.0
124.         }
125.         CATransaction.begin()
126.         CATransaction.setValue(
127.             kCFBooleanTrue, forKey: kCATransactionDisableActions)
128.
129.         // rotate the layer into screen orientation and scale and mirror
130.         detectionOverlay.setAffineTransform(CGAffineTransform(
131.         rotationAngle: CGFloat(.pi / 2.0)).scaledBy(x: scale, y: -scale))
132.         // center the layer
```

```
133.        detectionOverlay.position = CGPoint (x: bounds.midX, y:
    bounds.midY)
134.        CATransaction.commit()
135.    }
136.
137.    func createTextSubLayerInBounds(
138.          _ bounds: CGRect, identifier: String, confidence: VNConfidence
139.            ) -> CATextLayer
140.    {
141.        let textLayer = CATextLayer()
142.        textLayer.name = "Object Label"
143.        let formattedString = NSMutableAttributedString(
144.            string: String(format: "\(identifier)\nConfidence:  %.2f",
145.            confidence))
146.        let largeFont = UIFont(name: "Helvetica", size: 24.0)!
147.        formattedString.addAttributes([
148.            NSAttributedString.Key.font: largeFont
149.        ], range: NSRange(location: 0, length: identifier.count))
150.        textLayer.string = formattedString
151.        textLayer.bounds = CGRect(
152.            x: 0, y: 0, width: bounds.size.height - 10,
153.            height: bounds.size.width - 10)
154.        textLayer.position = CGPoint(x: bounds.midX, y: bounds.midY)
155.        textLayer.shadowOpacity = 0.7
156.        textLayer.shadowOffset = CGSize(width: 2, height: 2)
157.        textLayer.foregroundColor = CGColor(
158.            colorSpace: CGColorSpaceCreateDeviceRGB(),
159.            components: [0.0, 0.0, 0.0, 1.0])
160.        textLayer.contentsScale = 2.0 // retina rendering
161.        // rotate the layer into screen orientation and scale and mirror
162.        textLayer.setAffineTransform(CGAffineTransform(
163.            rotationAngle: CGFloat(.pi / 2.0)).scaledBy(x: 1.0, y: -1.0))
164.        return textLayer
165.    }
166.
167.    func createRoundedRectLayerWithBounds(_ bounds: CGRect) -> CALayer {
168.        let shapeLayer = CALayer()
169.        shapeLayer.bounds = bounds
```

```
170.         shapeLayer.position = CGPoint(x: bounds.midX, y: bounds.midY)
171.         shapeLayer.name = "Found Object"
172.         shapeLayer.backgroundColor = CGColor(
173.             colorSpace: CGColorSpaceCreateDeviceRGB(),
174.             components: [1.0, 1.0, 0.2, 0.4])
175.         shapeLayer.cornerRadius = 7
176.         return shapeLayer
177.     }
178.}
```

ДОДАТОК В ЕЛЕКТРОННА ПРЕЗЕНТАЦІЯ

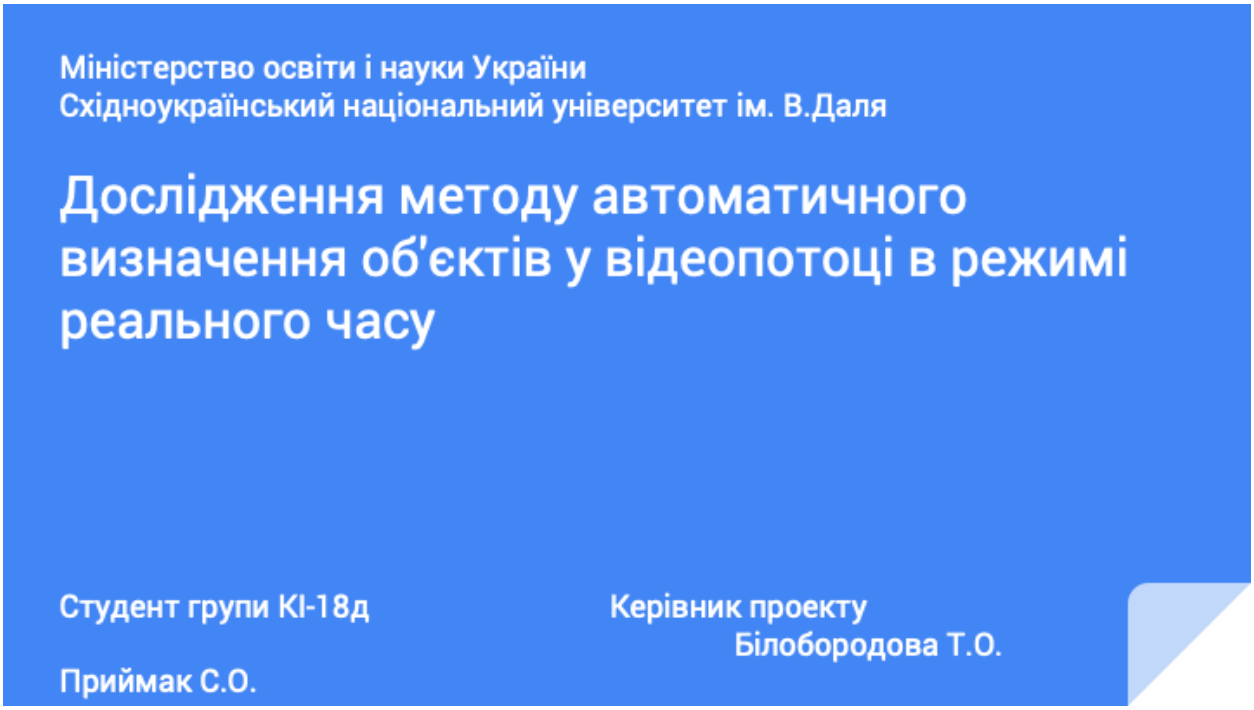

Рисунок В.1 – Слайд 1

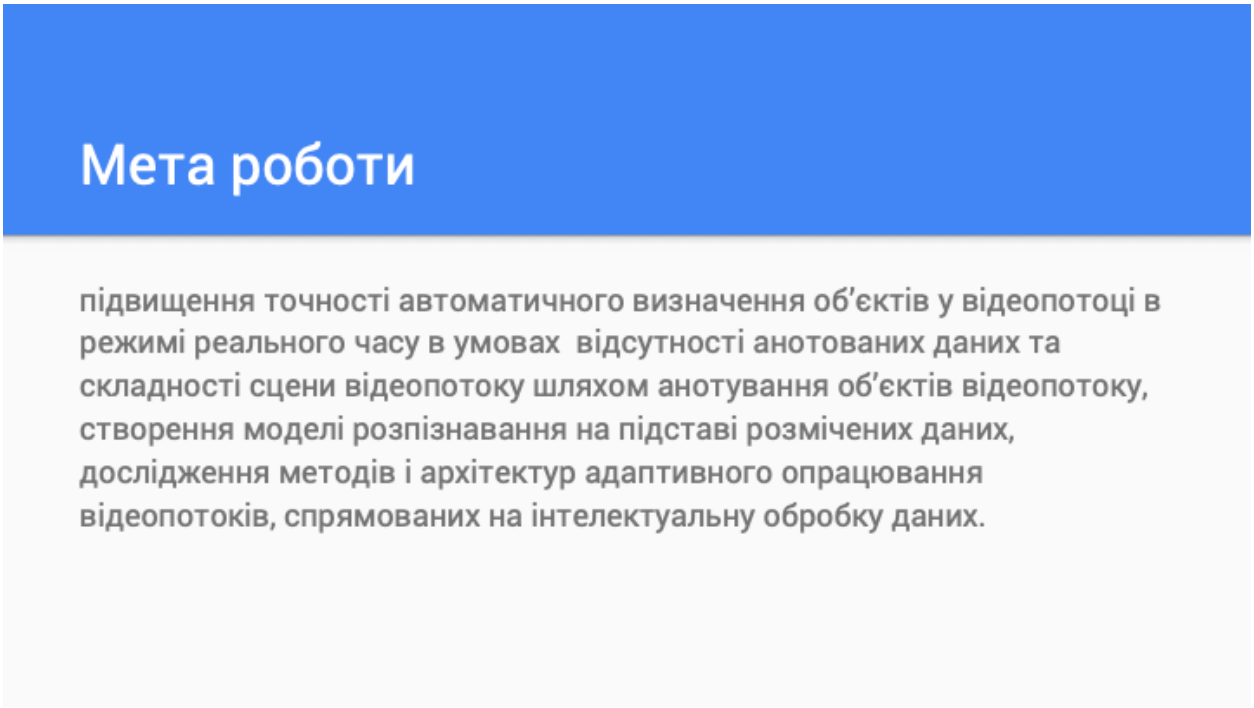

Рисунок В.2 – Слайд 2

Завдання

1. аналіз методів і технологій анотування відео та визначення об'єктів у відеопотоці в режимі реального часу
2. вибір засобу анотування об'єктів у відеопотоці, методу подальшого автоматичного вилучення і використання цифрових даних відеопотоку в режимі реального часу;
3. анотування даних відеопотоку
4. розробка програмного засобу для адаптації моделі даних відеопотоку з метою її подальшого використання
5. розробка мобільного додатку для тестування моделі в режимі реального часу
6. розробка та тестування моделі визначення об'єктів
7. оцінка ефективності методу

Рисунок В.3 – Слайд 3

Визначення об'єктів в реальному часі

1. генерація пропозицій
2. виявлення векторів властивостей (feature vector extraction)
3. класифікація регіонів
4. локалізація
5. визначення об'єктів

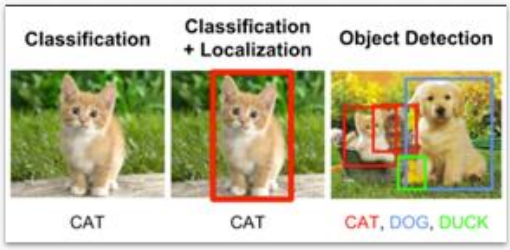

Рисунок В.4 – Слайд 4

You Only Look Once

Детектор об'єктів, що використовує функції, вивчені глубокою сгортковою нейронною мережею

1. state-of-the-art
2. Fast (45+ FPS)
3. 75 шарів нейронної мережі
4. нечутлива до размеру вхідного зображення

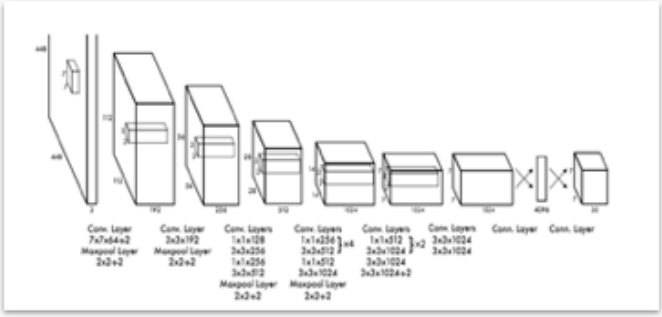

Рисунок В.5 – Слайд 5

You Only Look Once

1. Зменшення зображення
2. Формування пачки для обробки в GPU
3. Передача ознак, що було знайдено у поточному шарі, до класифікатора та регресора для визначення наявності об'єкта та його координат (карта ознак)
4. Визначення класу об'єкта в кожному знайденому регіоні по карті ознак (якщо центр об'єкта потрапить у сприйнятливе поле цієї комірки)

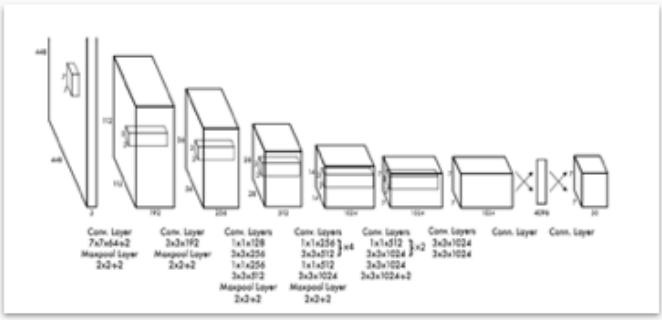

Рисунок В.6 – Слайд 6

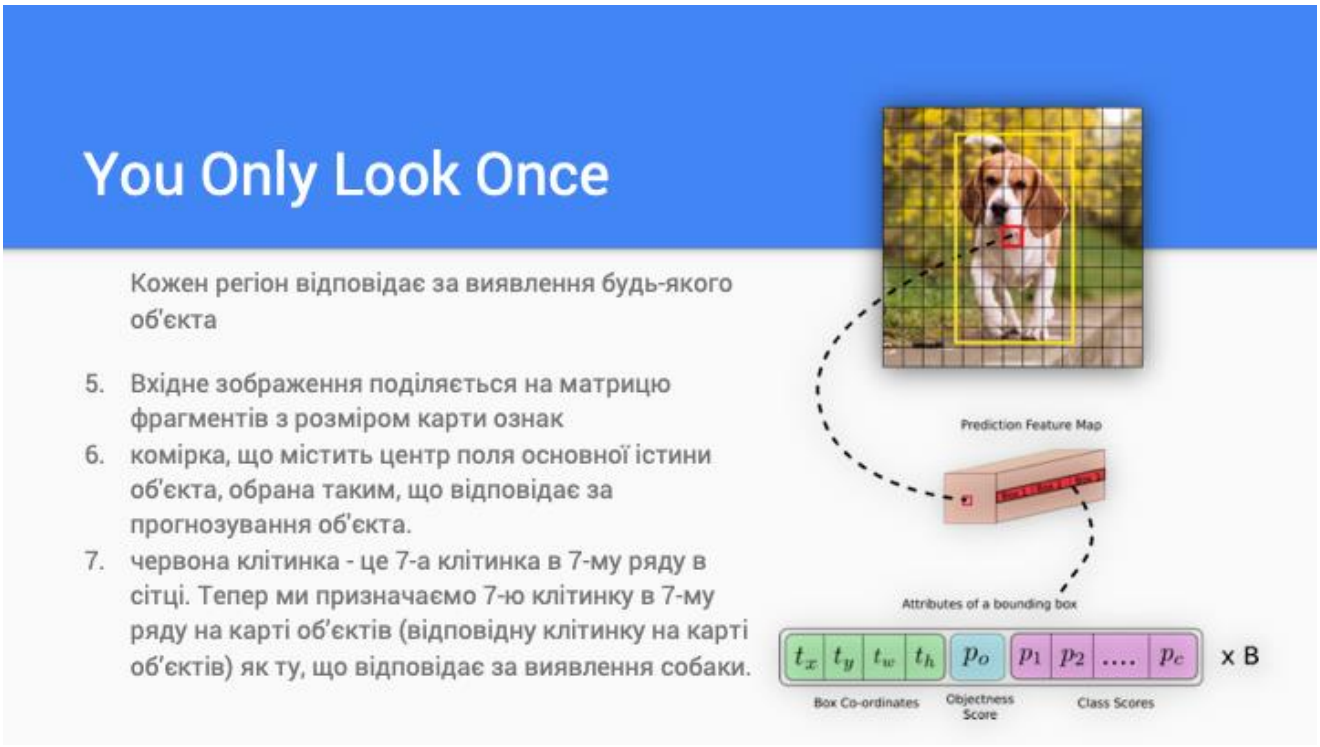

Рисунок В.7 – Слайд 7

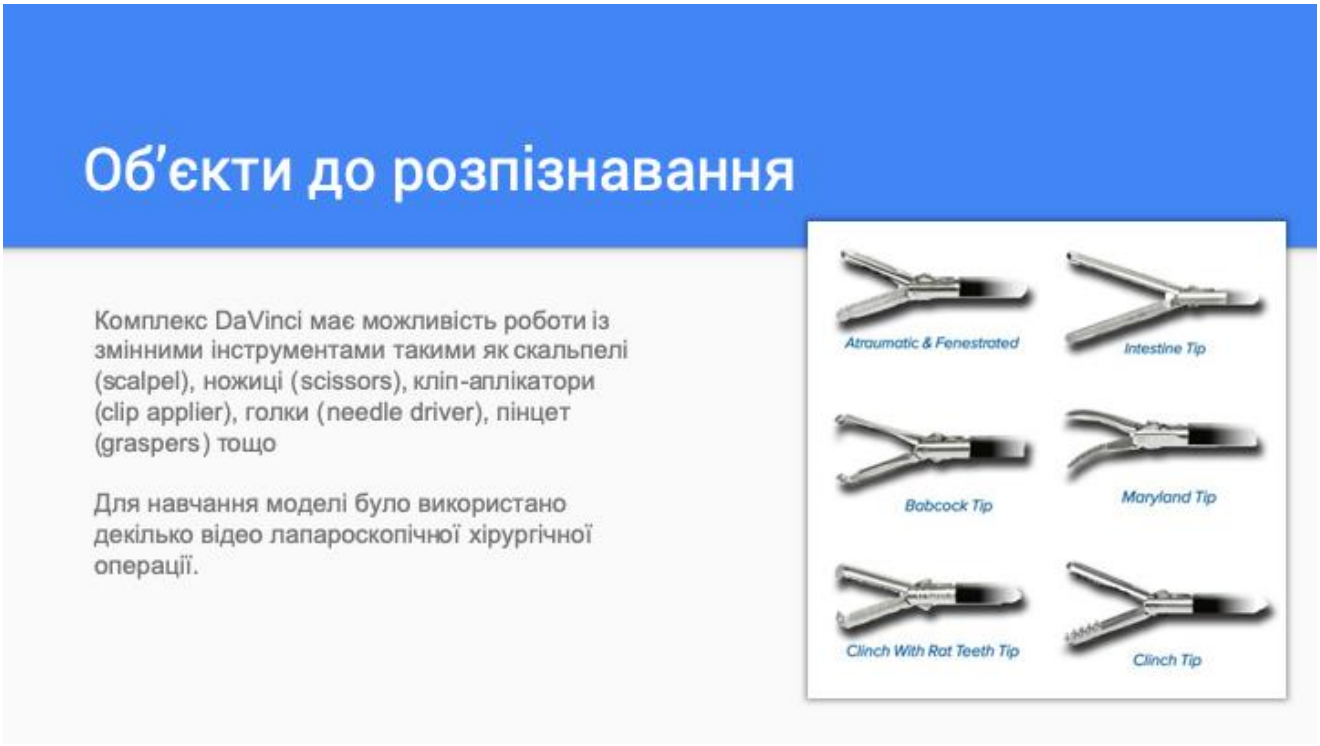

Рисунок В.8 – Слайд 8

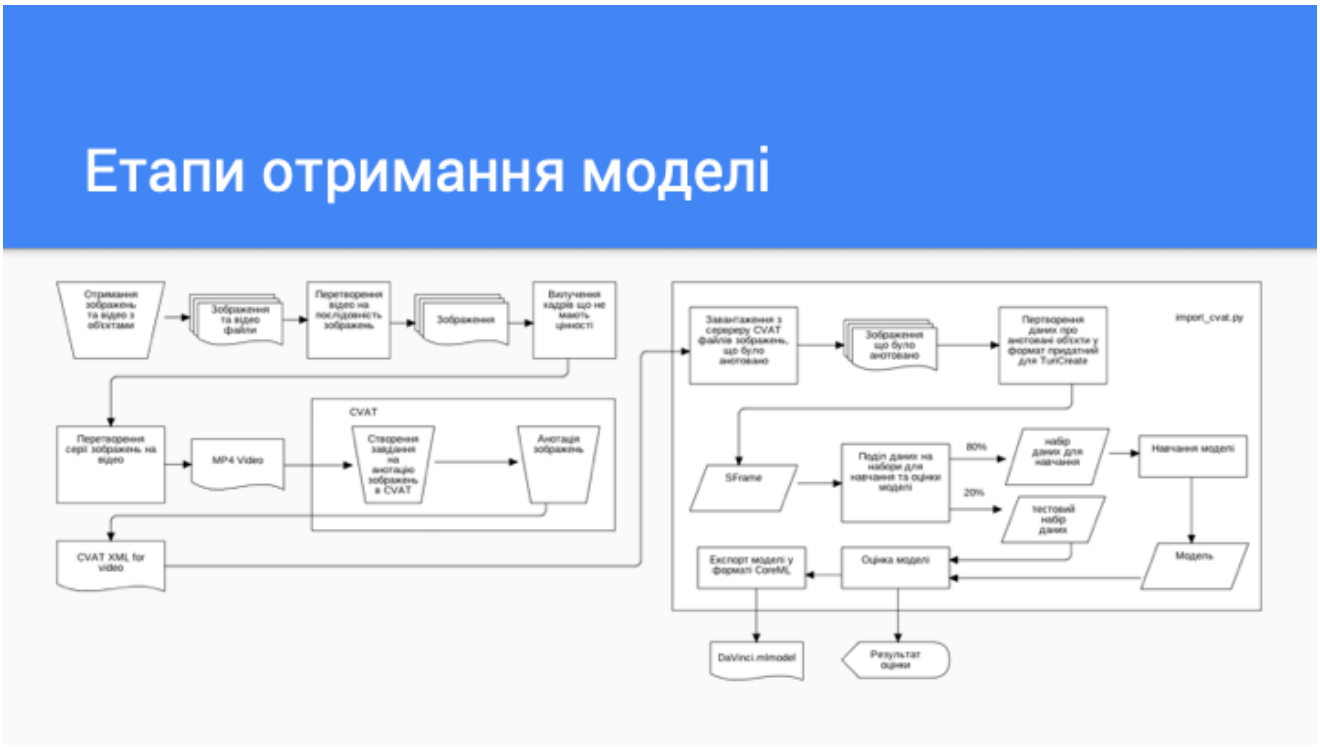

Рисунок В.9 – Слайд 9

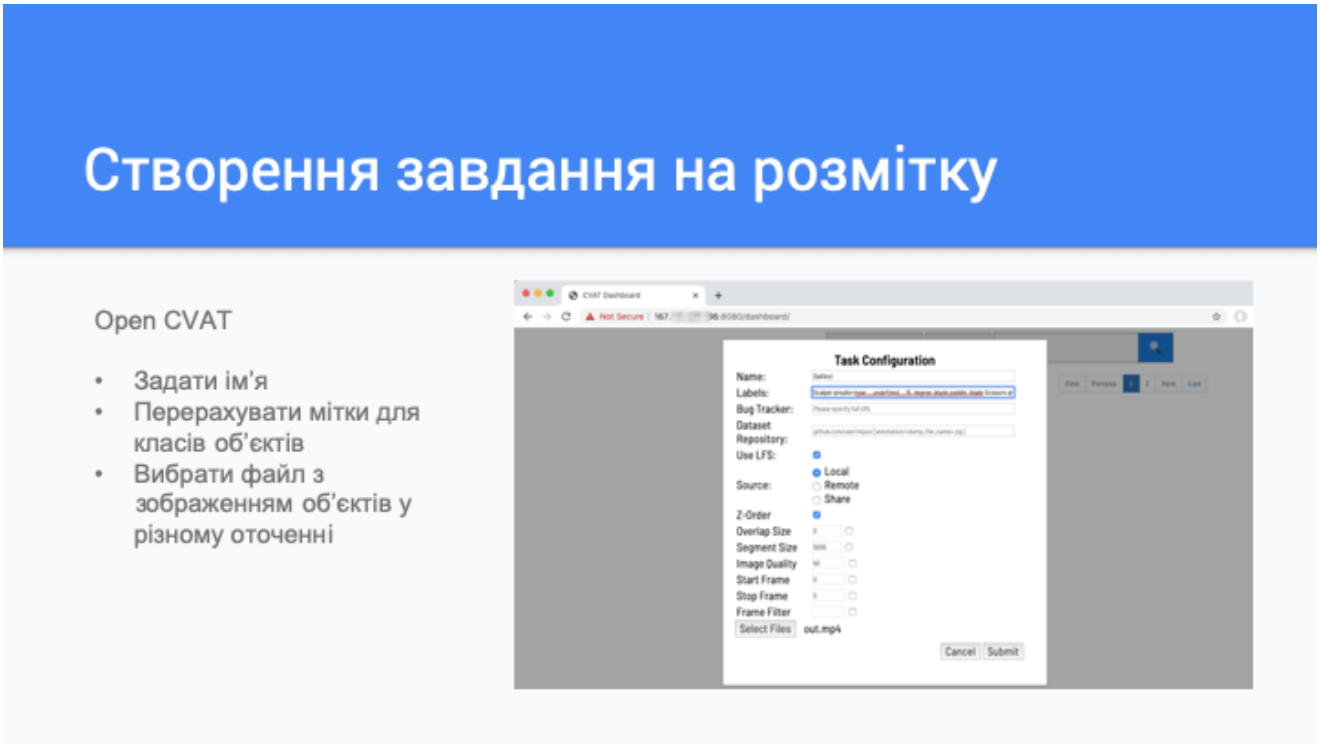

Рисунок В.10 – Слайд 10

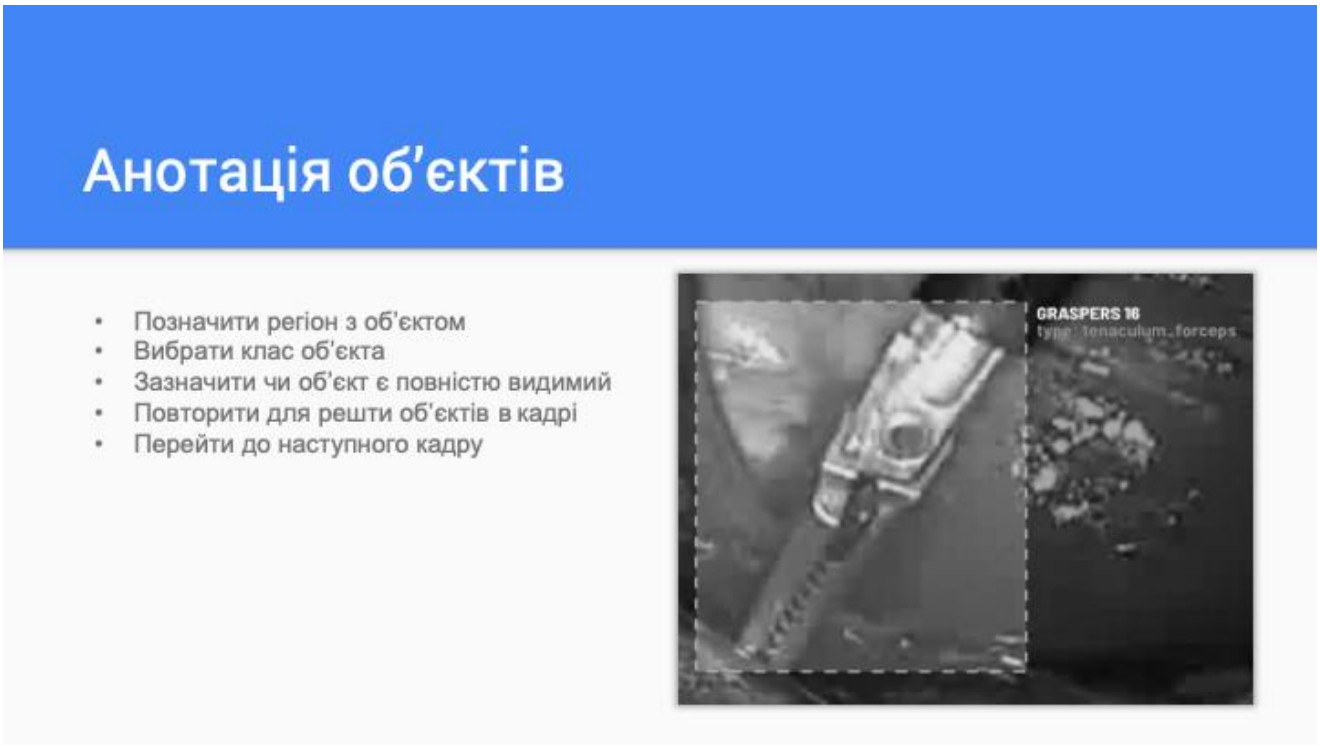

Рисунок В.11 – Слайд 11

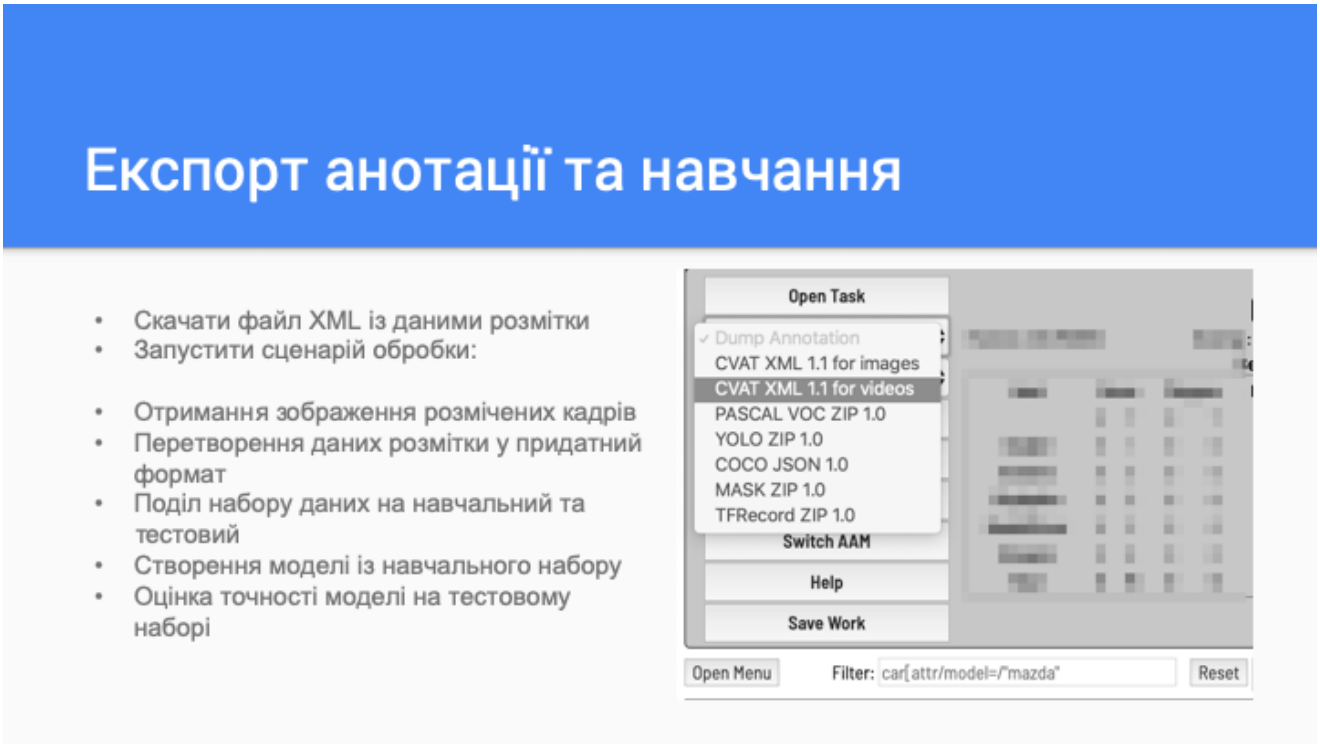

Рисунок В.12 – Слайд 12

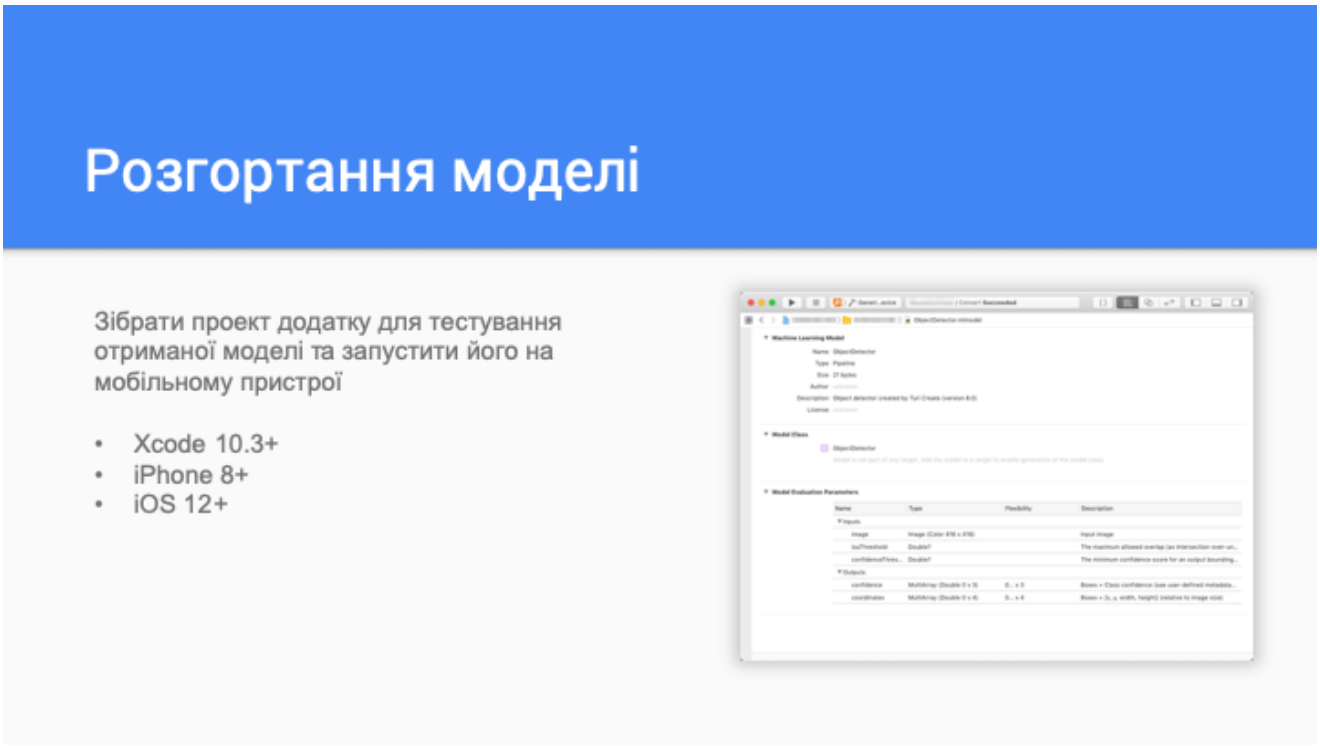

Рисунок В.13 – Слайд 13

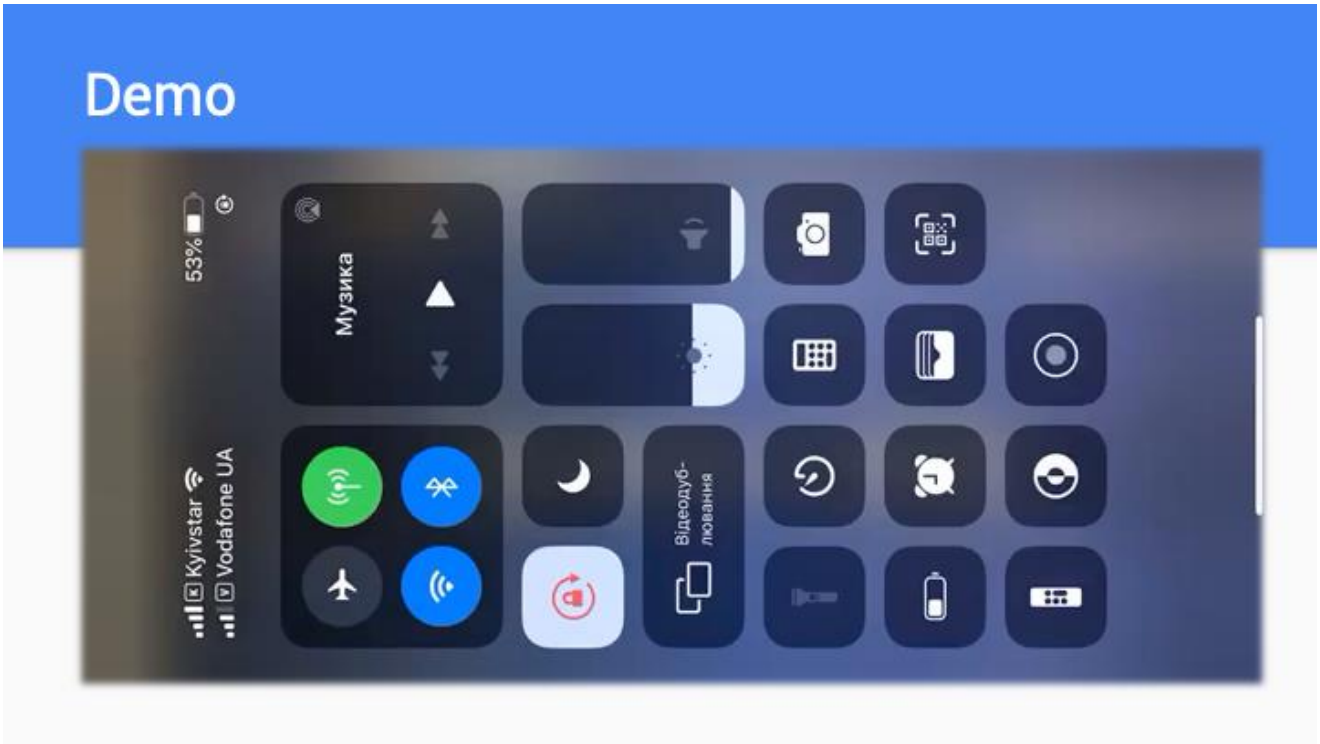

Рисунок В.14 – Слайд 14

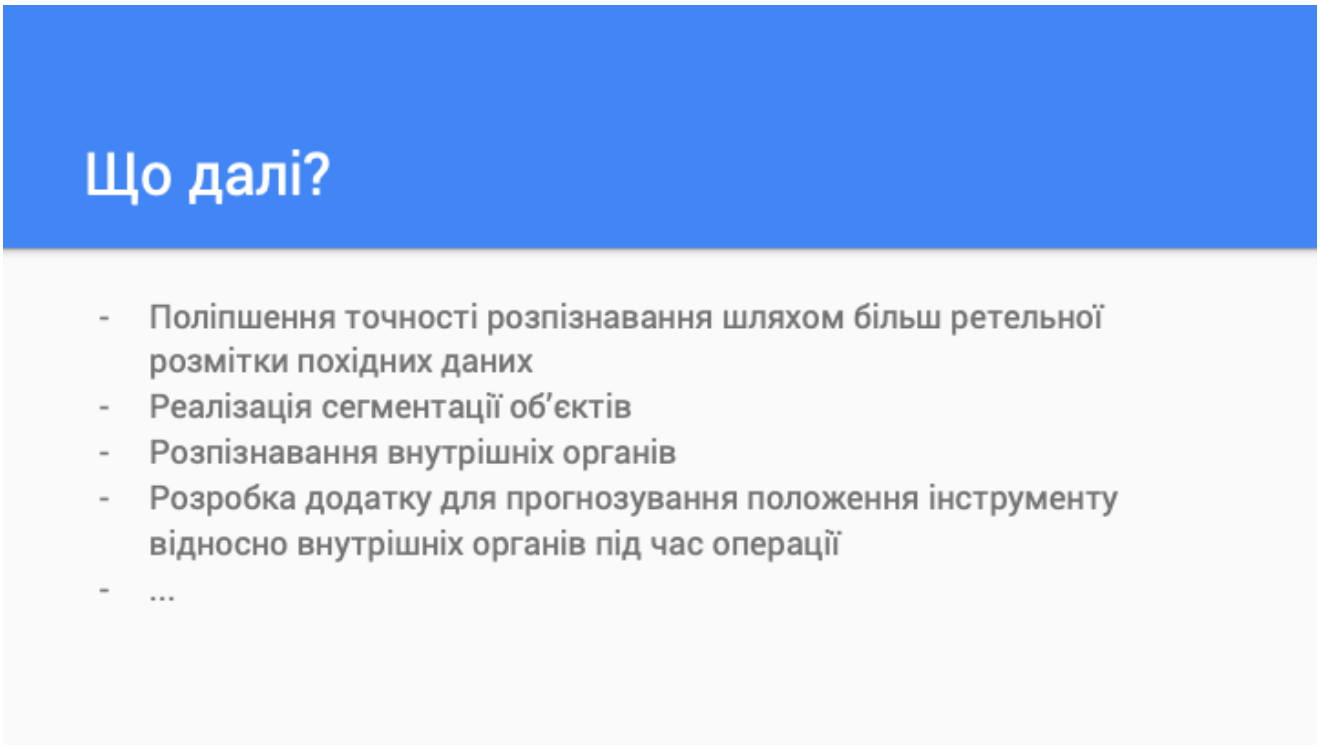

Рисунок В.15 – Слайд 15