МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Інтелектуальна система розпізнавання ознак пожежі в відеопотоці

Галузь знань **12 «Інформаційні технології»**
Спеціальність **122 «Комп'ютерні науки»**
Освітня програма **«Комп'ютерні науки»**
Освітній рівень: бакалавр

Виконала: студентка 4 курсу, групи КН- 41

Попова Є.П.
(прізвище та ініціали)

Керівник Доманецька І.М.
(прізвище та ініціали)

к.т.н., доцент
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри інтелектуальних технологій
Протокол №_13__від_05.06.2023 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____
_

"___"_____2023 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Поповій Єлизаветі Петрівні

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

«Інтелектуальна система розпізнавання ознак пожежі в відеопотоці»

затверджена протоколом засідання кафедри від «11» листопада 2022 р. №4

2. Термін здачі студентом закінченого проекту (роботи) 30 травня 2023 року

3. Вихідні дані до проекту (роботи)

Матеріали сучасних досліджень в галузі виявлення ознак пожежі засобами штучного інтелекту

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

Аналіз особливостей задачі розпізнавання пожеж у відеопотоці, проектування програмного забезпечення, розробка застосунку. .

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

1. Постановка задачі: мета, об'єкт, предмет, вимоги (1)
2. Предметна область: огляд задачі розпізнавання вогню, огляд існуючих підходів (6)
3. Проектування системи розпізнавання пожеж у відео потоці (6)
4. Програмна реалізація системи (5)

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2023 року

Керівник _____ / Доманецька І. М. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Попова Є. П. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Аналіз наукової літератури	17.02 – 28.02	
2	Робота над розділом 1. Особливості задачі розпізнавання пожежі у відео потоці	01.03 – 10.03	
3	Робота над розділом 2. Проектування системи розпізнавання пожежі у відео потоці	11.03 – 25.04	
4	Робота над розділом 3. Експериментальні дослідження та практична реалізація програмного забезпечення	09.04 – 17.05	
5	Робота над оформленням пояснювальної записки	01.05 – 18.05	
6	Робота над презентацією	15.05 – 18.05	

Студент-дипломник _____ / Попова Є. П. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Доманецька І. М. /
(підпис) (ПІБ)

АНОТАЦІЯ

Об’єктом дослідження дипломної роботи є процес розпізнавання ознак вогню у відеопотоці з камер спостереження на підприємствах. Предметом дослідження роботи є використання нейромережевих технологій для виявлення ознак пожеж у відеопотоці.

Метою роботи є розробка інтелектуальної системи моніторингу для виявлення ознак пожеж та мінімізації збитків на виробничих підприємствах

У першому розділі проаналізовано особливості задачі розпізнавання ознак пожежі у відеопотоці, розглянуто існуючі та актуальні підходи до її вирішення, обрано архітектуру нейронної мережі, що найкраще задовольняє вимоги задачі.

У другому розділі спроектовано функціональну структуру системи. Розроблено структуру бази даних, інтерфейс застосунку. Наведено узагальнений алгоритм опрацювання відеоряду.

У третьому розділі наведено інформацію про використані у дипломному проекті програмні засоби, проведено дослідження доцільності використання різних параметрів для обраної архітектури, протестовано працездатність готового застосунку та описано інструкцію користування.

Ключові слова: мережа yolo, розпізнавання ознак вогню, відеопотік, класифікація, ШНМ

ANNOTATION

The object of research is the process of recognizing signs of fire in the video stream from surveillance cameras at production enterprises. The subject of research is the use of neural network technologies to detect signs of fires in the video stream.

The purpose of the work is to develop an intelligent monitoring system to detect signs of fires and minimize losses at production enterprises.

In the first section, the features of the task of recognizing fire signs in a video stream are analyzed, the existing and current approaches to its solution are considered, and the neural network architecture that best meets the requirements of the task is chosen.

In the second section, the functional structure of the system is designed. The structure of the database, the interface of the application has been developed. A generalized algorithm for video sequence processing is presented.

The third section provides information about the software tools used in the project, conducted a study of various parameters for the chosen architecture, tested the functionality of the finished application, and described the user manual.

Key words: yolo network, fire recognition, video stream, classification, ANN

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

БД – база даних

CNN - Convolutional Neural Networks або згорткова мережа

ІТ - інформаційна технологія

НМ – нейронна мережа

P – Precision

R - Recall

СРОПВ - Система розпізнавання ознак пожежі у відеопотоці

ШІ – Штучний інтелект

ВСТУП

Мета: Розробка інтелектуальної системи моніторингу для виявлення ознак пожеж та мінімізації збитків на виробничих підприємствах.

Об’єкт дослідження:

Процес розпізнавання ознак вогню у відеопотоці з камер спостереження на підприємствах.

Предмет дослідження:

Використання нейромережевих технологій для виявлення ознак пожеж у відеопотоці.

Завдання дослідження:

Потрібно проаналізувати важливість проблеми раннього виявлення та попередження пожеж на підприємствах; дослідити існуючі рішення проблеми.

Дослідити ефективність різних алгоритмів для вирішення обраної проблеми. Запропонувати ефективну модель нейронної мережі що виявлятиме на поданому на вхід відеоряді ознаки вогню різного розміру та локалізуватиме їх. Виконати програмну реалізацію обраної моделі.

В ході виконання завдання було досліджено особливості задачі розпізнавання ознак пожежі у відеопотоці, розглянуто існуючі та актуальні підходи до її вирішення, обрано архітектуру нейронної мережі, що найкраще задовольняє вимоги задачі. Результати аналізу опрацьованих матеріалів викладено у першому розділі.

У другому розділі спроектовано функціональну структуру системи. Розроблено структуру бази даних, інтерфейс застосунку. Наведено узагальнений алгоритм опрацювання відеоряду.

У третьому розділі наведено інформацію про використані у дипломному проекті програмні засоби, проведено дослідження доцільності використання різних параметрів для обраної архітектури, протестовано працездатність готового застосунку та описано інструкцію користування.

РОЗДІЛ 1 ОСОБЛИВОСТІ ЗАДАЧІ РОЗПІЗНАВАННЯ ПОЖЕЖ У ВІДЕО ПОТОЦІ

1.1. Аналіз сутності задачі виявлення пожеж

Початкова стадія пожежі є найкращим часом для гасіння, тому первинне виявлення пожежі має дуже важливе значення. Виникнення пожеж на підприємствах є великою проблемою. Вони несуть загрозу працівникам та можуть нанести значні збитки. Зважаючи також на розташування індустріальних споруд, може минути багато часу до моменту коли рятувальники прибувають на місце інциденту. Саме тому важливим є якнайшвидше визначити ознаки виникнення пожежі, на етапах коли з'являються найменші ознаки полум’я.

Традиційні технології виявлення пожежі, засновані на розпізнаванні температури та виявленні задимлення за допомогою точкових сенсорів, мають повільний час відгуку, який зазвичай вимірюється в хвилинах, і їх важко застосовувати у зовнішньому середовищі. Щоб уникнути помилкової тривоги, димові та теплові сповіщувачі спрацьовують лише тоді, коли надходить достатня кількість частинок диму у пристрій або поки температура істотно не підвищиться. Проте час є головним чинником мінімізації збитків від пожежі. Зменшення часу реагування може значно збільшити наші шанси загасити пожежу та зменшити пошкодження, спричинені інцидентом. [11]

Цікавою та актуальною задачею є створення програмного застосунка для підприємства що спеціалізується на моніторингу стану виробничих приміщень та майданчиків з точки зору раннього виявлення пожежі. Створене програмне забезпечення буде пропонуватися промисловим підприємствам з метою організації пожежної безпеки. Це надасть можливість працівникам промислових підприємств відслідковувати стан відкритих ділянок за допомогою спостереження через відеокамери. Збільшення ефективності роботи такої

системи можна досягти за рахунок використання сучасних ефективних засобів виявлення найменших ознак полум’я, а саме штучного інтелекту.

Нейронні мережі та глибоке навчання в даний час пропонують найкращі рішення для багатьох проблем розпізнавання зображень, тож пропонується їх використання для поставленої задачі.[7]

Виявлення полум’я на зображеннях або відеокадрах, отриманих з датчиків зору є одним із таких методів. Цей підхід має зменшити час відгуку, підвищити ймовірність виявлення, забезпечити покриття великих територій та може бути застосований на відкритій території.

Також використання методу не потребує великих витрат, оскільки на багатьох підприємствах вже встановлені системи відеоспостереження, що можуть бути модернізовані.

Рисунок 1.1 - Збитки від пожежі на підприємстві

1.2. Аналіз особливостей задачі виявлення пожеж як задачі комп'ютерного зору

Штучний інтелект долає розрив між машинними і людськими можливостями. Багато науковців працюють над різними елементами сфери штучного інтелекту, однією з яких є комп’ютерний зір. Основна мета

комп’ютерного зору – зробити так, щоб машини бачили світ так само, як люди. Добре відомі задачі комп’ютерного зору включають виявлення зображень, позначення зображень, розпізнавання зображень, класифікація зображень, аналіз зображень, аналіз відео, обробку природної мови тощо.

CNN(Convolutional Neural Networks) використовується для побудови більшості алгоритмів комп'ютерного зору. Згорточна нейронна мережа є методом глибокого навчання який приймає вхідне зображення та призначає важливість (вивчаються упередження та ваги) різним об’єкти на зображенні, що відрізняють один від одного.

У порівнянні з іншими методами CNN вимагає менше попередньої обробки, тому CNN є найефективнішим алгоритмом навчання для розуміння картинного матеріалу. Крім того, він продемонстрував виняткову класифікацію зображень, розпізнавання, сегментацію та пошук.

Моделі на основі CNN використовуються більшістю лідерів обробки зображень та комп’ютерного зору. У результаті існує кілька варіантів базового дизайну CNN архітектури. [8]

Задачу можна поділити на роботу зі статичною та динамічною інформацією. Робота зі статичною інформацією характеризується опрацюванням окремих сталих зображень (окремих фото чи кадрів). Робота з динамічною інформацією пов`язана з опрацюванням відеорядів. Відеоряд складається з окремих кадрів, кожен з яких може містити шукані ознаки. Отже, динамічна робота потребує попередньої обробки відеоряду, а саме кадрування.

Тож вирішення поставленої задачі зводиться до роботи з зображеннями.

Система повинна ідентифікувати ознаки вогню. В залежності від обраного алгоритму, може бути наявний також клас “фон”, тобто частина зображення що не є вогнем. Отже, необхідно вирішити задачу класифікації, тобто розподілу об’єктів між заздалегідь відомими класами.

Датасет повинен містити набір зображень що можуть містити вогонь або бути просто фоновими, тобто без ознак пожежі. Усі зображення поділяються на навчальну, валідаційну та тестову вибірки.

Також можливо використовувати датасет що буде містити набір зображень, що знаходяться в одній директорії і можуть містити декілька об’єктів кожне.

Для кожного зображення в такому датасеті повинен бути наявним файл з інформацією про розташування та належність до одного з класів наявних на ньому об’єктів.

На рисунках 1.2.1 та 1.2.2 показано приклади зображення з датасету та позначених для ньому міток. Дані у файлі з рис.1.3 визначають координати рамки навколо об’єкта на зображенні на номер класу, до якого він належить.

Рисунок 1.2 - Приклад зображення з датасету

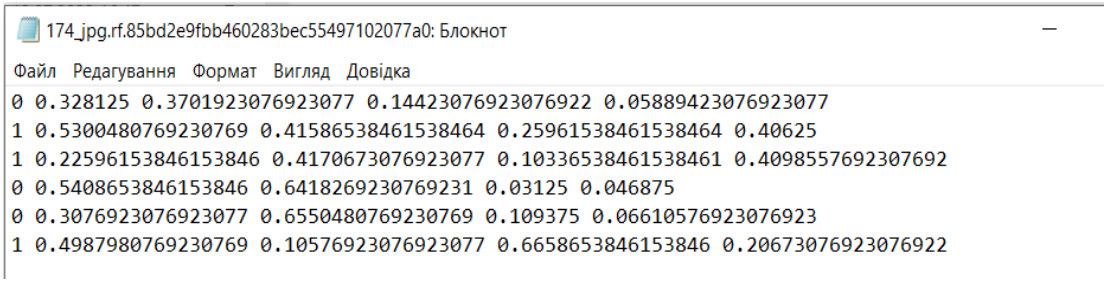

Рисунок 1.3 - Приклад позначення міток до зображення з датасету

Дані в датасеті поділені на тренувальну, валідаційну та тестову вибірки.

Також датасет повинен містити файл data.yaml, у якому будуть вказані кількість класів, їх назви та шлях до тренувального, валідаційного та тестового наборів.

Рисунок 1.4 - Вміст файлу data.yaml

1.3. Огляд існуючих підходів до розв'язання задачі виявлення вогню

Системи розпізнавання зазвичай використовують три характерні особливості пожежі: колір, рух і геометрію.[1]

Одним з методів виявлення пожеж є коваріаційний підхід. Цей метод часто використовуються для виявлення диму. У цьому методі, просторова та предметна інформація поєднуються за допомогою дескрипторів коваріації. Коли дим знаходиться поблизу камери тоді результат, отриманий за допомогою цього підходу, є високоефективний, але не дуже ефективний, якщо дим знаходиться далі від камери. Тому в багатьох випадках цей метод може виходити з ладу, оскільки точка виходу диму надто далеко від камери[13].

Також для даної задачі використовується фрактальний метод кодування. Дим має невизначену межу, і тому його важко виявити. Межа області диму нечітка; отже, стає важко виділити область диму, використовуючи стандартну обробку зображень.

Дим має фрактальні ознаки, тому щоб вирішити це питання використовується фрактальне кодування. У цьому методі використовується алгоритм K-середніх, що є ефективним методом машинного навчання для фіксації початкової точки в області домену.

Зображення є повністю чорним, поки всі пікселі знаходяться в початковій точці та через процес сегментації форма диму розвиває властивості самоподібності.

Позиційні співвідношення пікселів, однакової яскравості використовується для остаточного застосування фрактального кодування.

Також можливе виявлення пожежі та диму на основі RGB моделі.

Модель RGB – це адитивна модель, у якій червоне, зелене і синє світло додаються різними способами відтворюючи широкий спектр кольорів.

Моделі різних типів можуть використовуватись в поєднанні з інформацією про колір та аналізом руху, наприклад колірна модель YCbCr і модель YUV.

Далі буде розглянуто приклади реалізації можливих алгоритмів.

Розпізнавання ознак пожежі за кольором пікселів з використанням нечіткої логіки. Для кожного пікселя у що містить вогонь значення червоного каналу більше, ніж зеленого та синього, а значення зеленого більше ніж синього.

Крім того, колір вогню має дуже високу насиченість(R) у червоному каналі.

Ці правила, визначені для простору RGB, тобто $R \geq G \geq B$ і $R \geq Rmean$, можна перевести в YCbCr колірний простір як:

$$Y(x, y) > Cb(x, y) \qquad (1.1)$$

$$Cr(x, y) > Cb(x, y) \qquad (1.2)$$

де $Y(x, y)$, $Cb(x, y)$ і $Cr(x, y)$ є яскравістю, значенням синього кольору і червоного кольору для пікселя розташованого у просторових координатах (x, y) відповідно.

Рівняння (1.1) і (1.2) означають, відповідно, що яскравість вогню повинна бути більшою ніж значенням синього кольору і значення червоного кольору більше, ніж синього.

На рис. 1.5 показано зображення вогню з його каналами Y, Cb і Cr.

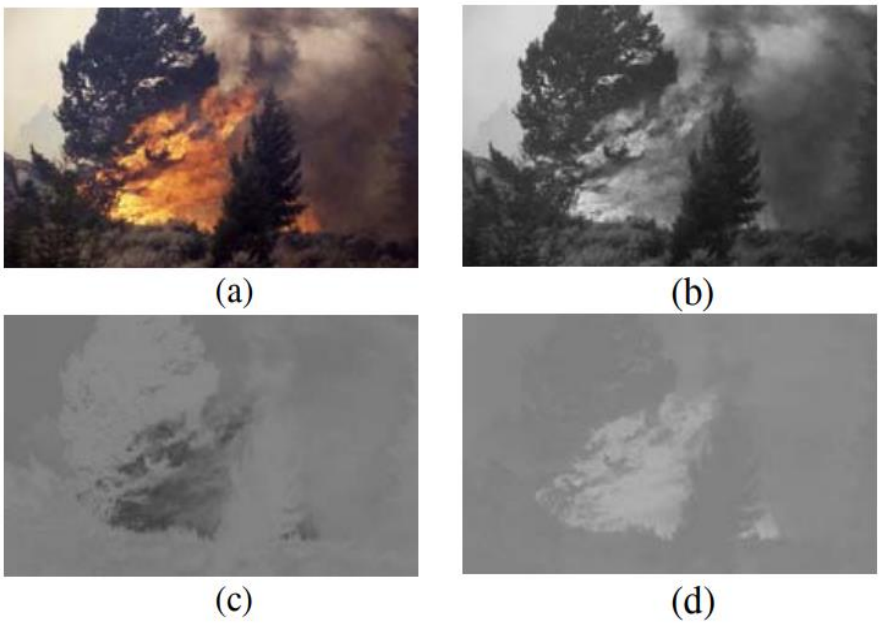

Рисунок 1.5 - Порівняння значення каналів, a - вхідне RGB зображення, b - Y канал, c - Cb канал, d - Cr канал

Як видно з рис. 1.5, чим Y(x, y) більше, ніж Cb(x, y) і Cb(x, y) менше ніж Cr(x, y), тим вища ймовірність того, що піксель містить полум’я.

Однак правила не відповідають вимогам єдиної кількісної міри, яка визначає наскільки ймовірно даний піксель є пікселем пожежі.

Отримані правила можуть бути закодовані в нечіткому представленні.

Величина єдиного вихідного рішення, виражена як число між нулем і одиницею дасть вірогідність того, що піксель містить вогонь. Цей нечіткий вихід також може розрізняти вогонь і вогняно-подібні кольорові предмети.

Нехай Pf (x, y) , визначається як міра, яка показує, як ймовірно, піксель, розташований у точці простору (x, y), містить вогонь.

Діапазон цього значення належить [0, 1].

Щоб оцінити Pf(x, y) використовується комбінація трикутних і трапецієподібні функції належності для Cr(x, y) −Cb (x, y), і Y(x, y) - Cb(x, y).

Створена система використовує правило Мамдані.

Нехай Bf (x, y) двійковий аналог зображення.

$$B_f(x,y) = \begin{cases} 1 & iff \quad P_f(x,y) \geq 0.5 \\ 0 & otherwise \end{cases}$$

Таким чином, значення Pf(x, y) буде збільшуватись в регіонах де міститься полум'я. [3]

Подібно до виявлення пожежі, можна розпізнавати пікселі диму.

Але пікселі диму не демонструють характеристик кольоровості, як пікселі вогню. На початку, коли температура диму мала, дим буде мати колір від біло-блакитного до білого.

Отже, ми можемо сформулювати пікселі диму таким чином:

$$\begin{aligned} |R(x,y) - G(x,y)| &\leq Th \\ |G(x,y) - B(x,y)| &\leq Th \\ |R(x,y) - B(x,y)| &\leq Th \end{aligned} \qquad (1.3)$$

де Th — це глобальне порогове значення в діапазоні від 15 до 25.

Рівняння 1.3 стверджує, що пікселі диму повинні мати подібні інтенсивності в їхніх колірних каналах RGB. [4]

Дуже популярним методом для обробки зображень є використання згорткових мереж.

CNN або згорткові мережі – архітектура штучних нейронних мереж, запропонована Яном Лекуном в 1988 році і націлена на ефективне розпізнавання зображень, що входить до складу технологій глибокого навчання.

CNN складається з вхідного рівня, вихідного рівня, а також кількох прихованих шарів. Приховані шари CNN зазвичай складаються зі згорткових шарів, шарів субдискретизації, повнозв’язних шарів та шарів нормалізації (ReLU). Додаткові шари можна використовувати для більш складних моделей.[5]

Вхідний шар: вхідне зображення, включаючи кілька колірних каналів.

Згортковий шар (Convolution): всі нейрони шару пов'язані тільки з частиною нейронів попереднього шару.

Служить для виділення ознак зображення та їх перетворення, які, в свою чергу, в подальшому на більш глибоких шарах, використовуються для отримання більш складних ознак і, в кінцевому підсумку, визначають клас об'єкта, що розпізнається.

Основною характеристикою даного шару є так звані фільтри - багатовимірні (зазвичай двомірні або тривимірні) матриці ваг зв'язків нейронів попереднього шару з нейронами згорткового шару.

Шар субдискретизації (Pooling, Subsampling): виділення найбільш значущих ознак попереднього шару і значне скорочення розмірності наступних шарів мережі.

Повнозв'язний шар (Fully-connected): прихований шар штучної нейронної мережі типу персептрон.

Даний шар є одновимірним і в ньому кожен нейрон пов'язаний з кожним нейроном попереднього шару на всіх рівнях. Основне призначення даного шару - перетворення сигналів, отриманих на згорткових рівнях мережі до одновимірного виду і виділення ознак на одновимірному рівні.

Єдиним параметром даного шару є кількість нейронів K. Зазвичай воно вибирається виходячи з розміру попереднього шару і або кількості класів (необхідної кількості виходів) мережі. [14]

На рисунку 1.6 показана типова архітектура CNN мережі.

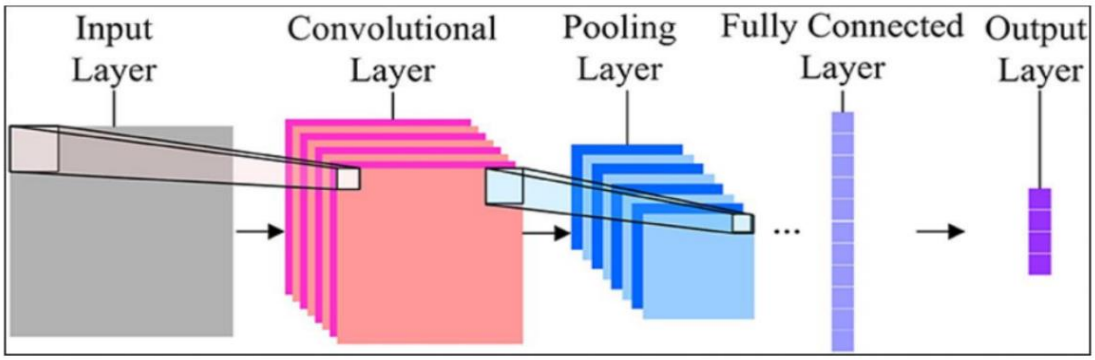

Рисунок 1.6 - Архітектура CNN мережі

Запропоновано ефективні варіанти архітектури CNN, зі структурою SqueezeNet для виявлення, локалізації та семантичного сприйняття пожежі.

Поява пожеж відслідковується за допомогою камер відеоспостереження.

Використання CNN для ідентифікації пожеж на зображеннях з високою точністю та продуктивністю дозволяють системі працювати в режимі реального часу.[1]

Також можливе використання поєднання моделей нейронних мереж різної архітектури. Наприклад, ввиявлення пожежі та сегментація за допомогою YOLOv5 та U-NET. На рис.1.7 представлено схему архітектури мережі YOLOv5.

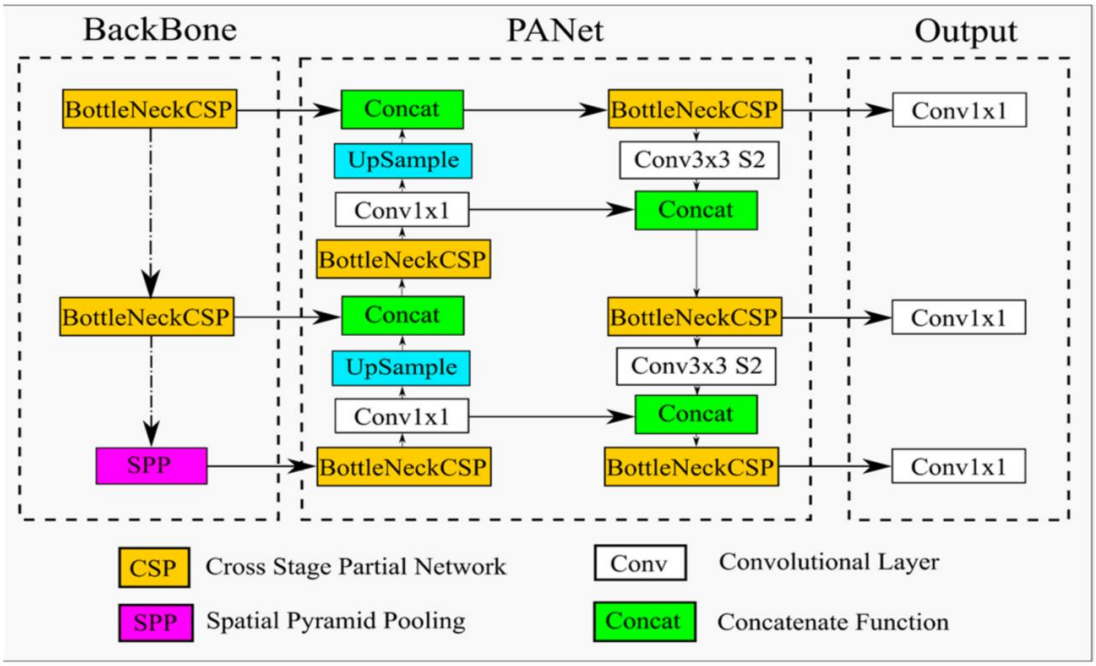

Рисунок 1.7 - Загальна архітектура мережі YOLOv5

Загальна архітектура мережі YOLOv5 складається з трьох частин (backbone, PANet та вихід) та була повністю реалізований за допомогою Python (PyTorch). Звичайний детектор об’єктів складається з кількох частин (тобто хребта, шиї, голови тощо). Шия включає будь-які додаткові блоки або блоки агрегації шляхів. [10]

Типовим є використання згорткових мереж у задачах класифікації, де результатом для зображення є одна мітка класу. Однак у багатьох візуальних завданнях бажаний результат повинен включати локалізацію, тобто мітка класу має бути призначена кожному пікселю. Використання моделі U-net, по-перше, дозволяє локалізувати результат По-друге, навчальних даних в умовах патчів набагато більше, ніж кількість навчальних зображень.

На рисунку 1.8 зображено приклад архітектури мережі U-net.[9]

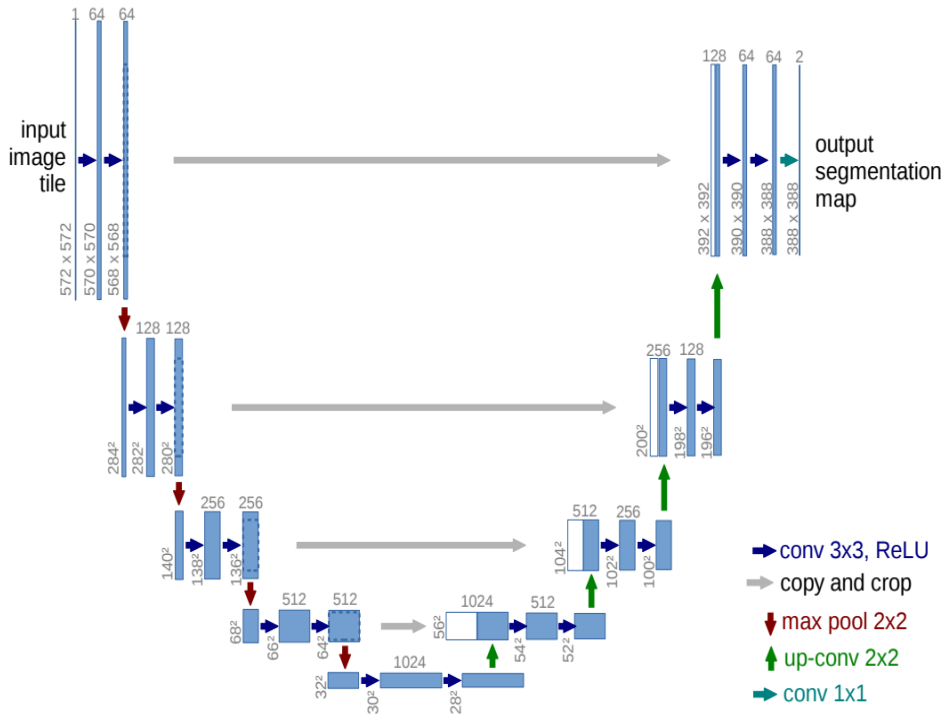

Рисунок 1.8 - Приклад архітектури U-net

Запропонована архітектура виявлення пожежі заснована на поєднанні двох моделей глибинного навчання для ефективного виявлення та локалізації пожеж. Модель приймає на вхід зображення та локалізує полум'я на виході. Першим

кроком моделі є мережа YOLOv5, другим – мережа U-Net. У запропонованій архітектурі ці дві мережі інтегровані.[8]

На рис. 1.9 зображено схему поєднання розглянутих мереж.

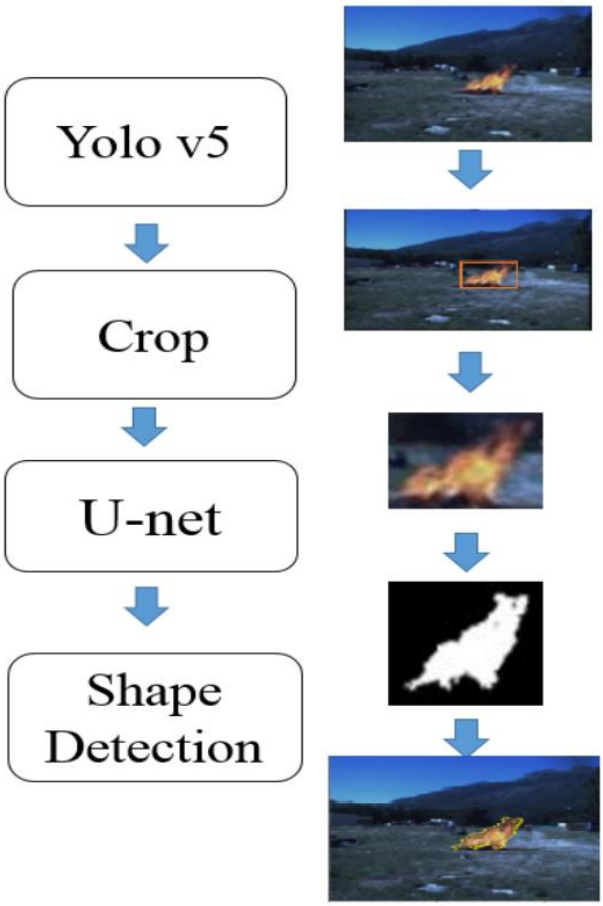

Рисунок 1.9 - Запропонована архітектура поєднання YOLO та U-net[8]

Спочатку мережа приймає RGB кольорові зображення пожеж, які обробляється YOLOv5, щоб отримати рамку навколо зони пожежі. Після цього до отриманого зображення застосовується шар кадрування з результатів YOLOv5, щоб отримати лише частини зображення, обмежені рамкою. Потім ці обрізані зображення подаються на вхід U-Net для підтвердження присутності полум'я та визначення точного місця пожежі. Результатом є двійкова маска, що представляє вогняні пікселі на зображенні. Після цього отримані обмежувальні лінії наносяться на оригінальне зображення.

Мережа U-Net — це глибока згорткова мережа, яка успішно використовується в сегментації медичних зображень.

На відміну від традиційних моделей глибокого навчання, які потребують багато даних, UNet можна навчати з невеликою кількістю даних. U-Net — це двохетапна модель глибокого навчання. Її архітектура включає модель кодера, за якою слідує модель декодера. Вона містить дев'ять блоків, по чотири блоки в кожному етапі та один спільний блок. Кожен блок складається з двох двовимірних згорткових шарів. У фіналі шар 1*1 створює а двійкову маску.

Ця модель використовує набір вхідних зображень вогню та їх відповідні бінарні маски. Під час навчання та на основі бінарної маски за бажанням модель можна навчити класифікувати кожен піксель зображення в різні мітки об’єктів.[8]

1.4. Вибір та обґрунтування архітектури нейронної мережі

Для вирішення задачі було обрано використати архітектуру нейроної мереж YOLOv5 (You Only Look Once).

YOLOv5 написано мовою Python замість C, як у попередніх версіях. Це робить установку та інтеграцію легше.

У 2015 році Джозеф Редмон представив систему виявлення об’єктів, яка виконує всі необхідні етапи для виявлення об'єкта за допомогою однієї нейронної мережі. На момент випуску алгоритм YOLO показав вражаючі технічні характеристики, які перевершують провідні алгоритми як за швидкістю, так і за точністю для виявлення та визначення координат об'єкта.

Перші три версії досліджено та розроблено автором алгоритму YOLO, Джозефом Редмоном.

Пізніше дослідник Гленн Джохер і його дослідницький відділ Ultralytics LLC, які створили YOLO алгоритми на основі Pytorch, опублікували YOLOv5 з деякими відмінностями та покращеннями. Незважаючи на те, що YOLOv5 не був розроблений групою авторів алгоритму, він вразив своєю продуктивність порівняно з усіма чотирма попередніми версіями.

Основна ідея YOLO полягає в застосуванні сітки до зображення. Якщо центр об’єкта потрапляє в клітинку сітки, ця клітинка відповідальна за виявлення цього об'єкта. Тому всі інші клітини не враховуються навіть якщо до них потрапляє частина об'єкта.

Щоб реалізувати виявлення об’єктів, кожна комірка сітки передбачає множину обмежувальних прямокутників, з їхніми параметрами та оцінками достовірності для цих рамок.

Ці показники достовірності відображають наявність або відсутність об'єкта в обмежувальній рамці та ймовірність належності об’єкта до кожного з класів. Вважається що об’єкт належить до того класу, ймовірність для якого є найвищою. [12]

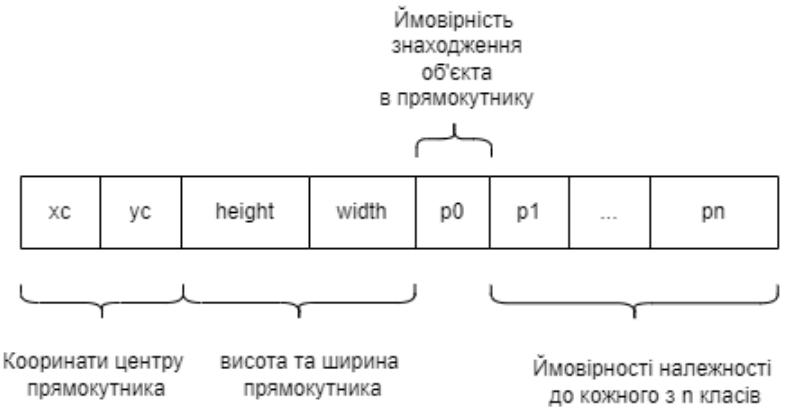

Рисунок 1.10- Структура виходу нейронної мережі YOLO

Основна відмінність між YOLO та іншими системами виявлення об’єктів сформульована в її назві: you only look once – вона дивиться на зображення лише один раз. Коли алгоритм було вперше представлено, він продемонстрував життєздатність одноетапного підходу. Інші методи використовують двоетапний процес, спочатку місцезнаходження, а потім ідентифікацію об’єктів.

Завдяки одному етапу архітектура YOLO неймовірно швидка. Це дає можливість обробляти відеозаписи в режимі реального часу.

Зважаючи на особливості ділянок, для яких буде розроблятись система, було вирішено для виявлення пожежі використовувати лише ознаки вогню, так як джерелами диму можуть бути виробничі процеси на підприємствах.

1.5. Визначення основних вимог до системи

Отже, задачею є створення програмного застосунка, що використовуватиме технології нейронних мереж та глибокого навчання для моніторингу стану виробничих приміщень та майданчиків з точки зору раннього виявлення пожежі. Цей підхід має зменшити час відгуку, підвищити ймовірність виявлення, забезпечити покриття великих територій та може бути застосований на відкритій території.

Система поділятиметься на три підсистеми, а саме:

- підсистема підприємства надавача сервісу
- дві частини на рівні об’єкта спостереження, а саме адміністрування та моніторингу

Функціональні вимоги до підсистеми підприємства надавача сервісу:

1. Система повинна надавати можливість реєстрації нових підприємств, редагування та вилучення наявних.

2. Адміністратор підприємства надавача сервісу повинен мати можливість надання доступу до системи довіреним особам на об’єктах спостереження.

Функціональні вимоги на рівні адміністрування об’єкта спостереження:

3. Дані підприємства і доступ до відеопотоку з камер розташованих на ньому повинні бути захищені і недоступні неавторизованим користувачам.

4. Адміністратор повинен мати можливість додавання та видалення локацій, працівників, груп камер та окремих камер.

5. Розроблена система повинна надавати можливість використання на багатьох різних підприємствах. На кожному підприємстві повинна бути можливість створення груп камер для спостереження за окремими ділянками.

Функціональні вимоги на рівні моніторингу на об’єкті спостереження:

1. Після реєстрації підприємства його працівники повинні мати можливість авторизуватись в системі і мати доступ до даних лише того підприємства, до якого вони належать.

2. При виявленні ознак пожежі система повинна сповіщати чергових працівників підприємства використовуючи візуальний сигнал на екрані та звуковий сигнал.

3. Розроблена система повинна надавати можливість завантажити відеоряд у форматі «mp4» та повертати на виході оброблене відео у форматі «avi» для демонстраційного прикладу роботи системи.

4. Система повинна включати модуль що відповідатиме за розпізнавання ознак вогню з використанням нейронних мереж YOLO v5.

5. Розроблена система повинна приймати на вхід відеоряд за посиланням на публічно доступну відеокамеру та повертати на виході оброблений відеопотік у режимі онлайн.

6. Отриманий на вхід відеоряд має розбиватись на окремі кадри, кожен кадр має оброблюватись розробленим модулем, після чого оброблені зображення мають записуватись до результуючого відео. На кожному кадрі що містить вогонь повинні додаватись мітки розташування знайдених об’єктів та підписи що визначають їх належність до класу вогню.

Нефункціональні вимоги:

1. Дизайн веб-застосунку має бути інтуїтивно зрозумілим для користувачів без спеціальних технічних знань

2. Головна сторінка має містити область відображення відеопотоків та навігаційне меню з можливістю обрати бажану камеру для перегляду та завантаження демонстраційного відео.

3. Система повинна коректно працювати з відеорядом з будь-якою величиною кадру

4. Система повинна знаходити полум’я з точністю не менше 0.8.

5. Система повинна працювати в режимі «real-time»

6. Система повинна забезпечувати зручний інтерфейс для відображення інформації на широкоформатних екранах.

Висновки

У ході роботи над теоретичною частиною роботи було досліджено питання значущості проблеми пожеж на виробничих підприємствах, проаналізовано вже існуючі традиційні методи їх виявлення та методи з застосуванням штучного інтелекту. Було розглянуто різні архітектури нейронних мереж серед яких можна виділити архітектури YOLO та U-net. Для практичної реалізації було обрано мережу архітектури YOLO, так як вона є дуже ефективною для задач комп'ютерного зору, має високу швидкість роботи та добре підходить для опрацювання відео в реальному часі. Визначено функціональні на нефункціональні вимоги до створюваної інтелектуальна система розпізнавання ознак пожежі в відеопотоці, в тому числі на різних рівнях адміністрування та функціонування системи.

РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ ПОЖЕЖ У ВІДЕО ПОТОЦІ

2.1. Розробка структурної моделі системи

Розроблене програмне забезпечення повинно давати змогу одночасного використання на різних підприємствах. Приклад структури організації роботи системи та розташування камер на окремому підприємстві наведено на рис.2.1. Під локацією мається на увазі підприємство на якому встановлено СРОПВ. На кожному підприємстві можуть бути визначені окремі ділянки, на яких може знаходитись декілька камер спостереження.

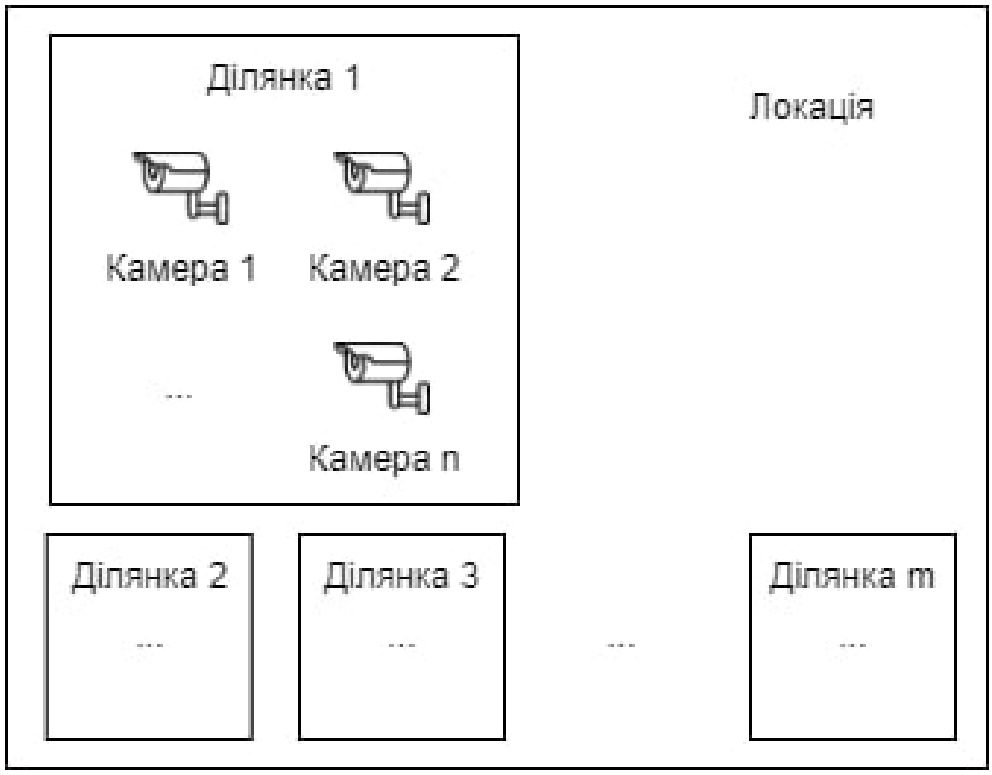

Рисунок 2.1 – Структура організації роботи системи на підприємстві

Для моделювання системи буде застосовуватись нотація IDEF0. Зазвичай її використовують для створення функціональних моделей, що відображають

функції та структуру системи, потоки інформації і матеріальних даних, що використовуються у системі.

IDEF0 може бути використана для аналізу функцій системи та відображення механізмів, за допомогою яких вони виконуються.

Компонентами синтаксису IDEF0 є блоки, стрілки, діаграми та правила.

Блоками є функції, які визначаються як діяльність, процес, операція, дія або перетворення. Усередині кожного блоку міститься його ім'я і номер.

Стрілки визначають дані або матеріальні об'єкти, пов'язані з функціями. Вони показують, які дані або матеріальні об'єкти мають надійти на вхід функції для того, щоб ця функція могла виконуватися.

Стрілки діляться на п'ять видів :

– входу – дані або об'єкти, що змінюються в ході виконання роботи;

– виходу – дані або об'єкти, що з'являються в результаті виконання роботи;

– управління – правила і обмеження, згідно з якими виконується робота;

– механізму – ресурси, необхідні для виконання роботи що не змінюються в процесі роботи;

Кожен блок діаграми IDEF0-моделі може бути деталізований на іншій діаграмі. Оскільки кожен блок розуміється як окремий об'єкт, поділ такого об'єкта на його структурні частини називається декомпозицією.

Під системою розпізнавання пожеж будемо розуміти багаторівневу організаційно-інформаційну систему надання послуг з розпізнавання пожеж у відеопотоці.

Система надання послуг з розпізнавання пожеж у відео потоці передбачає наявність:

- підсистеми адміністрування на рівні підприємства надавача сервісу
- підсистеми адміністрування на рівні об'єкта спостереження
- підсистеми моніторингу

На рисунку 2.2 зображено функціональну модель системи.

На вхід системи подаються відео потік з камер спостереження та інформація про об’єкт спостереження на якому використовується система.

Робота системи керуються правилами пожежної безпеки та положеннями про організацію пожежної безпеки на підприємстві.

Механізмами та ресурсами системи є відеокамери, попередньо навчена модель нейронної мережі, адміністратор підприємства надавача сервісу та адміністратор на об’єкті спостереження, інформаційна технологія та чергові працівники.

Виходом системи є рішення про наявність ознак пожежі.

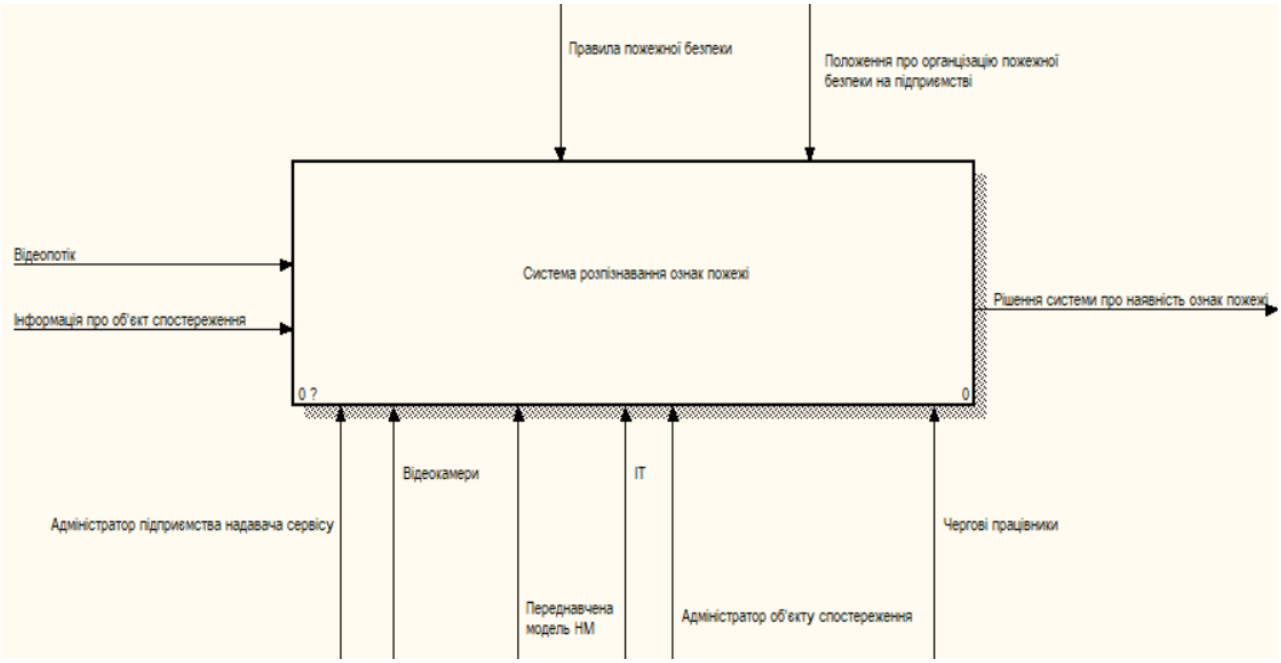

Рисунок 2.2 - Контекстна IDEF0 діаграма системи розпізнавання ознак пожежі

На рисунку 2.3 зображено декомпозицію СРОПВ на три блоки що відповідають трьом підсистемам: адміністрування на рівні підприємства надавача сервісу, адміністрування на рівні об'єкта спостереження та моніторингу.

Підсистема адміністрування на рівні підприємства надавача сервісу приймає на вхід інформацію про об’єкт спостереження для налаштування

параметрів для підприємства. Ресурсом підсистеми є адміністратор на рівні підприємства надавача сервісу. Підсистема адміністрування на рівні об'єкта спостереження також приймає на вході інформацію про об’єкт спостереження. Ресурсом є адміністратор на об’єкті спостереження. Керування та обмеження для підсистеми встановлюються виконуються налаштованими з підсистеми адміністрування на рівні підприємства надавача сервісу, правилами пожежної безпеки та положеннями про організацію пожежної безпеки на підприємстві. Виходом підсистеми є параметри організації роботи на підприємстві. Підсистема моніторингу приймає на вхід відеопотік з камер спостереження. Ресурсами підсистеми є відеокамери, інформаційна технологія, попередньо навчена модель та чергові працівники. Підсистеми керується параметрами отриманими від підсистеми адміністрування на рівні об'єкта спостереження.

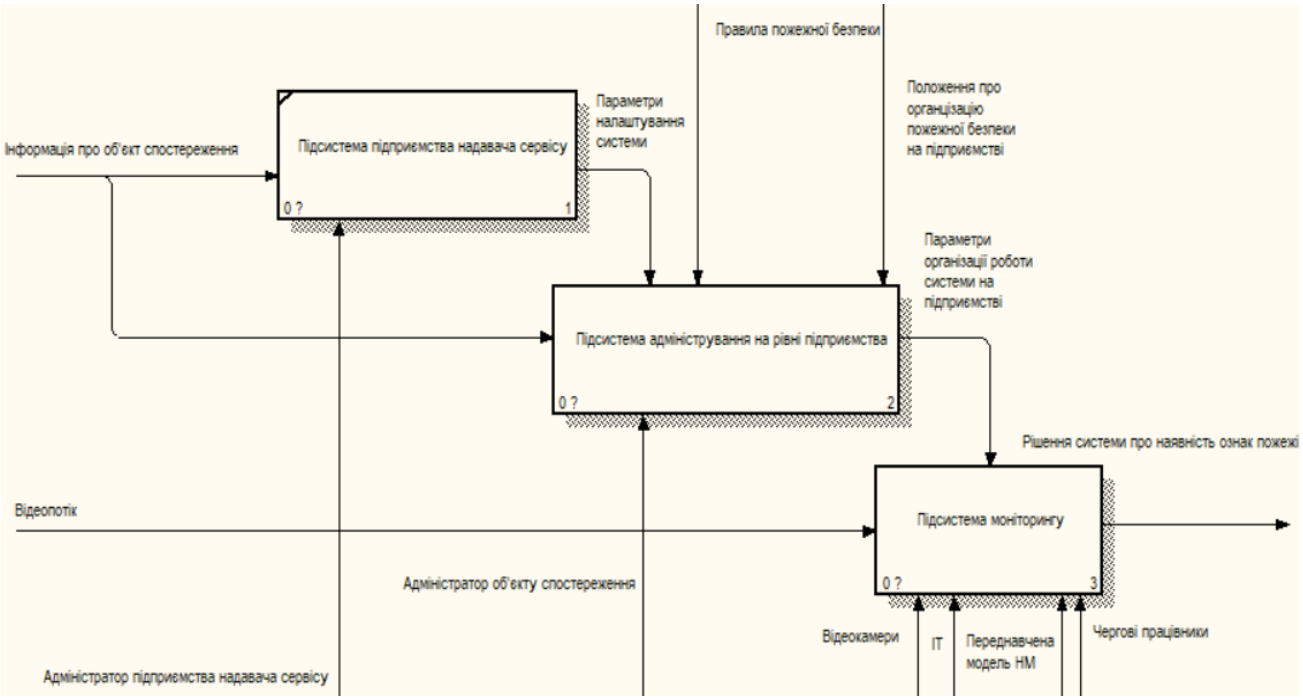

Рисунок 2.3 - Декомпозиція діаграми системи розпізнавання ознак пожежі

На рисунку 2.4 зображено декомпозицію підсистеми моніторингу СРОПВ. Підсистема складається з 4 блоків. Блок розбиття відеоряду на кадри приймає на вхід відеопотік. Ресурсами блоку є відеокамери та IT. Блок застосування нейронної мережі до кадру відеоряду приймає на вхід окремі кадри отримані в

результаті розбиття відео. Ресурсами блоку є ІТ та попередньо навчена модель нейронної мережі. На виході з блоку будуть отримані обмежувальні рамки на оброблених кадрах, які далі стають вхідними даними для блоку фільтрування результатів за точністю. Цей блок керується параметрами організації роботи системи на підприємстві. Ресурсами блоку є ІТ. Блок перегляду результатів роботи системи наглядачем приймає на вхід отримані в попередньому блоці обмежувальні рамки, що задовольняють налаштуванням точності системи. Ресурсами блоку є ІТ та чергові працівники. Керується робота блоку параметрами роботи системи на підприємстві. Виходом блоку є рішення системи про наявність ознак пожежі у відеопотоці.

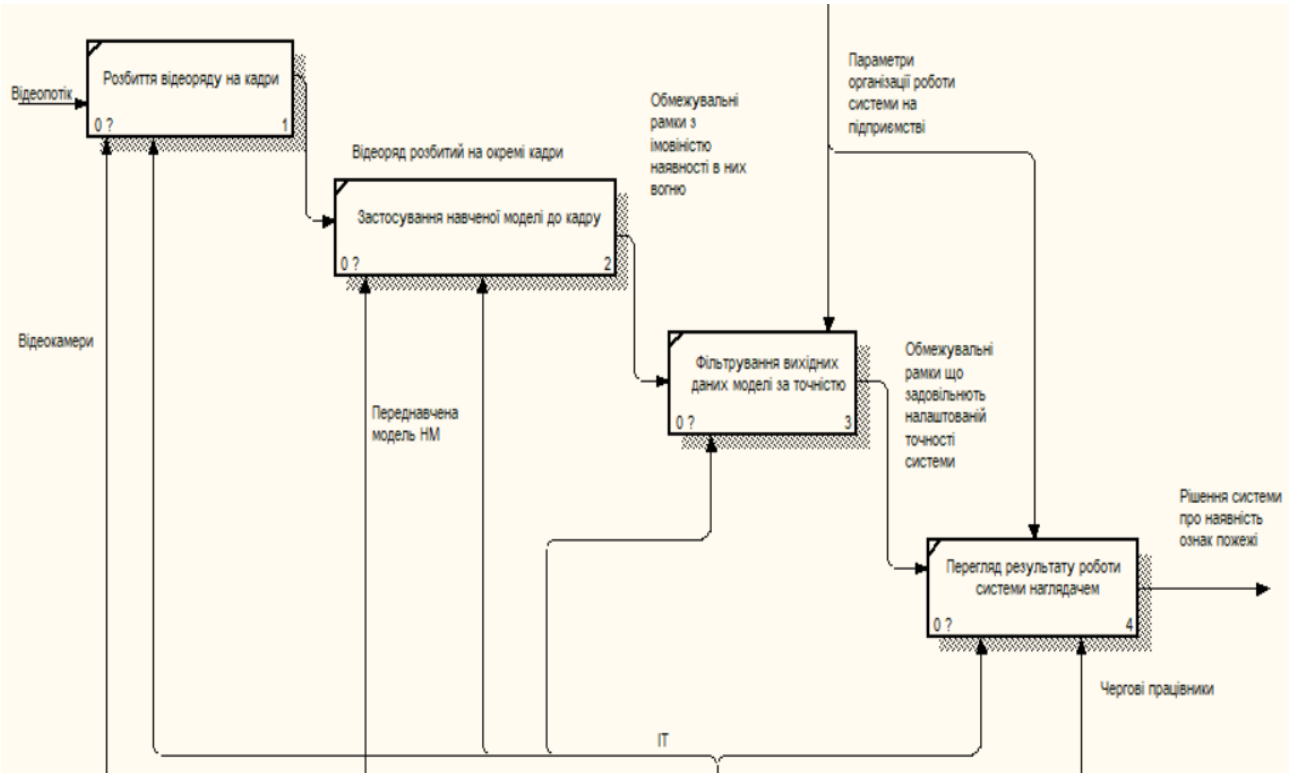

Рисунок 2.4 - Декомпозиція підсистеми моніторингу системи розпізнавання ознак пожежі у відеопотоці

2.2. Проектування функціональної структури програмного забезпечення

Як було зазначено у пункті 2.1, система має складатись з трьох підсистем, функції яких зображено на рис. 2.5.

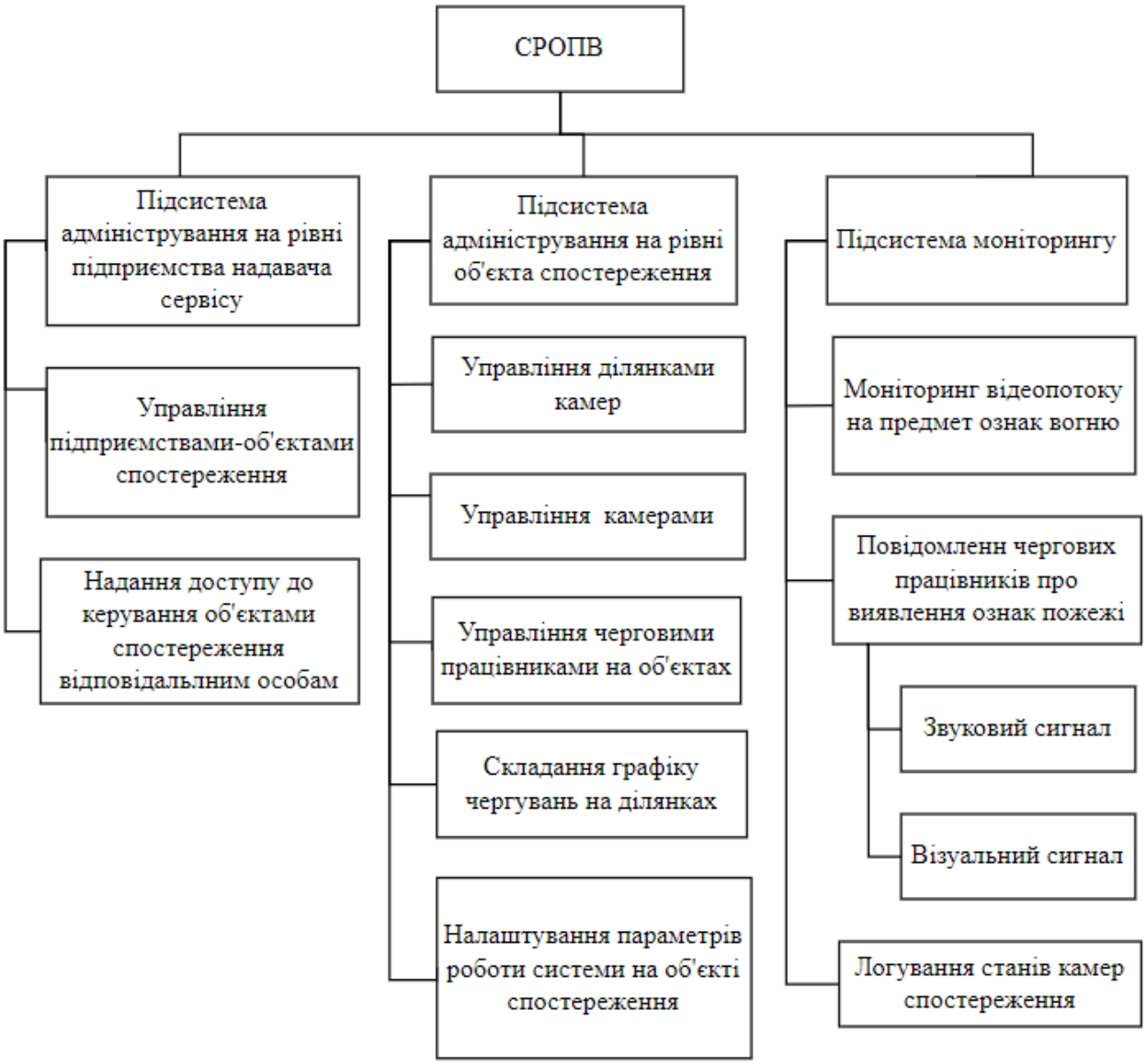

Рисунок 2.5 – Функціональний склад підсистем СРОПВ

Визначені підсистеми матимуть наступні функції:

1. Модуль підсистеми підприємства надавача сервісу

Функції модуля: керування підприємствами-об’єктами спостереження, їх додавання вилучення та редагування; надання доступу до керування зареєстрованими об’єктами спостереження відповідальним особам;

2. Модуль підсистеми адміністрування на рівні об’єкта спостереження

Функції модуля: керування ділянками камер та окремими камерами на об’єктах спостереження, їх додавання вилучення та редагування; керування черговими працівниками на об’єктах, їх додавання вилучення та редагування, складання графіку чергування працівників; налаштування параметрів роботи системи на об’єкті спостереження;

3. Модуль підсистеми моніторингу

Функції модуля: моніторинг відеопотоку з камер на об’єкті спостереження на предмет ознак пожежі; повідомлення про виявлення ознак пожежі за допомогою звукового та візуального сигналів; логування станів камер з заданим інтервалом часу; тестування роботи системи за допомогою тестового відео;

2.3. Узагальнений алгоритм опрацювання даних відеопотоку

Як було зазначено у 1.2, система працюватиме з динамічною інформацією, тобто відеорядом. Відеоряд складається з окремих кадрів, кожен з яких може містити потрібні ознаки, тому він потребує попередньої обробки, а саме кадрування. Таким чином вирішення поставленої задачі зводиться до роботи з зображеннями.

Далі наведено узагальнений алгоритм обробки відеоряду з відеокамери або завантаженого користувачем відео:

1. Розбиття вхідного потоку на кадри
2. Зміна розміру отриманого зображення на для подальшого додавання до обробленого відеоряду
3. Застосування до зображення наступних перетворень: масштабування, зміна каналів кольорів, зміна розміру для обробки мережею
4. Застосування навченої НМ до отриманого зображення. В результаті буде отримано множину обмежувальних рамок заданих координатами центру, довжиною, висотою, ступенем впевненості мережі в знаходженні всередині

рамки об’єкту та ступенем впевненості приналежності даного об’єкту до класу вогню.

5. Відсіювання обмежувальних рамок для яких ступінь впевненості знаходження всередині об’єкта менше 0.6

6. Нанесення на зображення обмежувальних прямокутників що залишились після п.5 для візуальної локалізації знайдених ознак вогню

7. Додавання обробленого зображення до результуючого відео потоку з камери або завантаженого відео, відповідно до його походження

2.4. Розробка архітектури системи

На рисунку 2.6 зображено основні складові архітектури програмного застосунку СРОПВ. Умовно систему можна поділити на клієнтську частину, до якої належатимуть камери спостереження та пристрої чергових працівників та серверну, до якої належатимуть БД та елементи адміністрування системою.

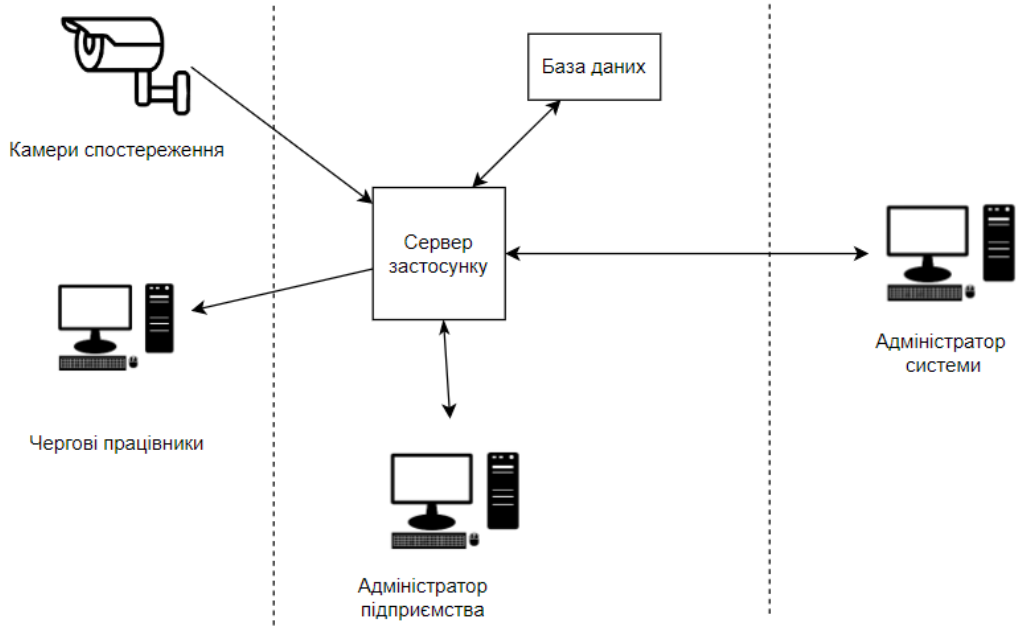

Рисунок 2.6 – Архітектура СРОПВ

Передбачається що у системі для однієї локації одночасно можуть існувати декілька ділянок з камерами, по декілька камер на кожній ділянці. Також на кожній локації можуть працювати багато працівників. Окремий працівник може бути призначений на будь-яку ділянку, що пов’язана з його локацією. Розподіл працівників між ділянками визначається окремою таблицею в БД.

Нові користувачі не матимуть можливості самостійної реєстрації на сайті. Дані нових чергових працівників повинні вноситись адміністраторами на рівні підприємств, які самі додаватимуться адміністратором системи.

2.5. Структура бази даних

База даних системи розпізнавання ознак пожежі містить такі таблиці:

1.Локації(Locations) - містить інформацію про всі підприємства на яких використовується розроблена система, а саме ідентифікатор, назву підприємства та адресу його розташування, контакти, відповідальна особа.

2.Ділянки(Area) - містить інформацію про окремі ділянки на кожній локації, а саме ідентифікатор, географічні координати, назву, ідентифікатор локації.

3.Працівники(Workers) - містить інформацію про працівників, що користуються системою, а саме ідентифікатор, роль, ім’я, та ідентифікатор локації до якої вони належать.

4.Камери(Cameras) - містить інформацію про камери, встановлені на підприємстві, а саме ідентифікатор, посилання на відео потік та ідентифікатор ділянки на якій встановлено окрему камеру.

5.Графік(Schedule) - зберігає інформацію про графік чергування працівників на ділянках, а саме ідентифікатор, ідентифікатор працівника, ідентифікатор ділянки, дату, час початку та закінчення чергування.

6.Логування(Logs) - зберігає інформацію про стан кожної з камер за визначений інтервал часу, а саме ідентифікатор, статус, ідентифікатор камери, ідентифікатор чергового працівника, дату, час.

У таблиці 2.1 наведено опис всіх таблиць та їх полів.

На рис. 2.7 зображено концептуальну схему розробленої БД.

Таблиця 2.1 – Опис полів БД

Назва поля	Таблиця	Ключове поле	Тип даних	Зовнішній ключ
Id	Locations	Так	Ціле число	-
Name	Locations	-	Символьний	-
Address	Locations	-	Символьний	-
Contacts	Locations	-	Символьний	-
Responsible	Locations	-	Символьний	-
Id	Areas	Так	Ціле число	-
Lat	Areas	-	Дійсне число	-
Long	Areas	-	Дійсне число	-
Location_id	Areas	-	Ціле число	Так
Id	Cameras	Так	Ціле число	-
Link	Cameras	-	Символьний	-
Area_id	Cameras	-	Ціле число	Так
Id	Workers	Так	Ціле число	-
Role	Workers	-	Символьний	-
Name	Workers	-	Символьний	-
Location_id	Workers	-	Ціле число	Так
Id	Schedule	Так	Ціле число	-
Start_datetime	Schedule	-	Дата та час	-
Etart_datetime	Schedule	-	Дата та час	-
Area_id	Schedule	-	Ціле число	Так
Worker_id	Schedule	-	Ціле число	Так
Id	Logs	Так	Ціле число	-
Status	Area_id	-	Символьний	-
Date	Worker_id	-	Дата	-
Time	Area_id	-	Час	-
Camera_id	Worker_id	-	Ціле число	Так

Worker_id	Area_id	-	Ціле число	Так

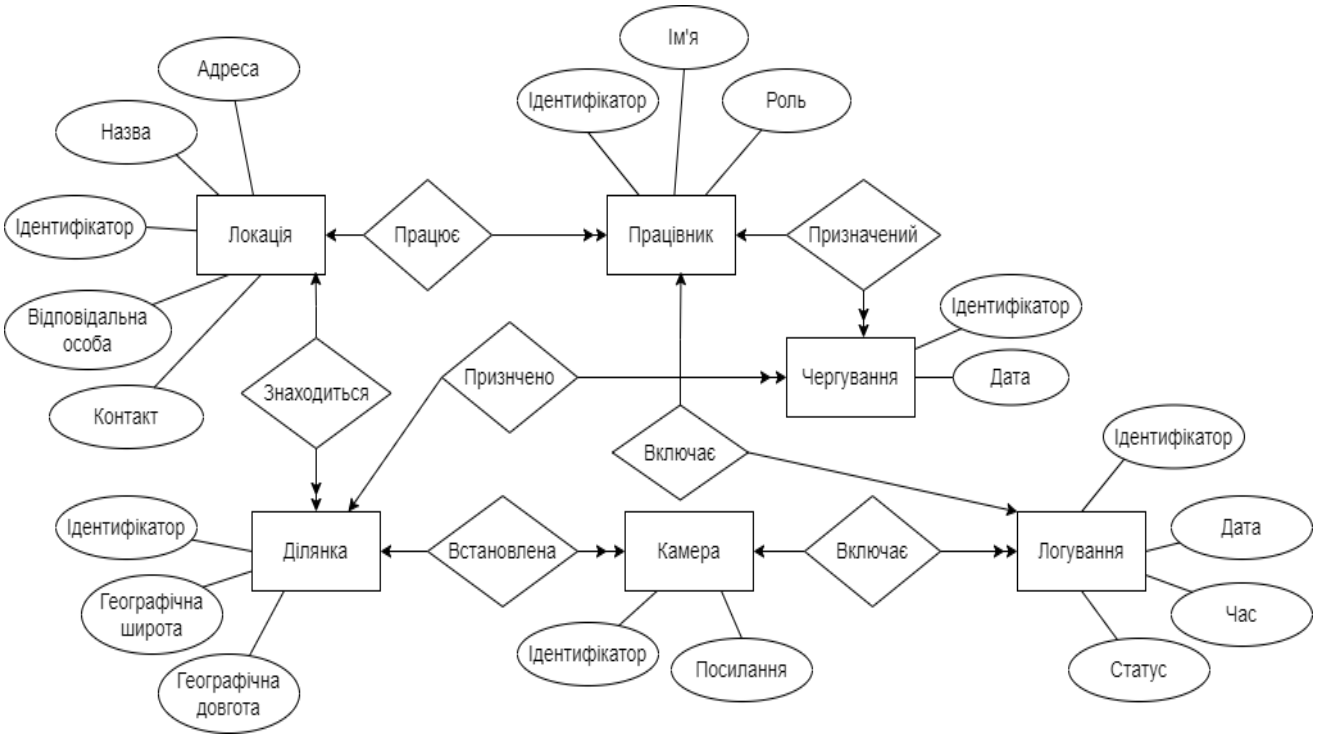

Рисунок 2.7 – Концептуальна сема БД

Далі на рис. 2.8 зображено логічну схему розробленої БД.

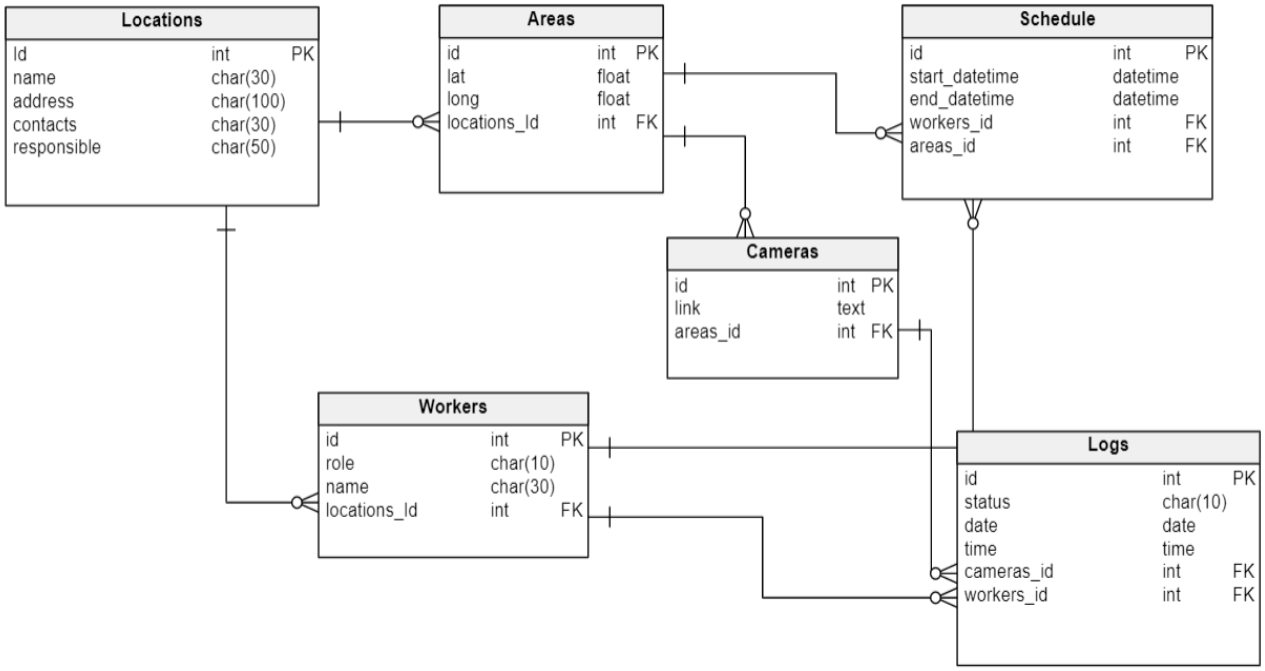

Рисунок 2.8 - Логічна структура БД

2.6. Розробка макетів сторінок веб-застосунку

Як було зазначено у пункті 2.1, майбутня система поділятиметься на три підсистеми з власним функціоналом. Відповідно, для кожної з підсистем було розроблено унікальний інтерфейс.

На рис. 2.12, 2.13 та 2.14 зображено переходи між сторінками сайту для кожної з підсистем.

Для підсистеми моніторингу передбачено три сторінки: сторінка авторизації, сторінка перегляду відеокамер та сторінка перегляду тестового відео.

Сторінка авторизації призначена для авторизації чергового працівника. Вона містить форму для введення даних авторизації. Після авторизації користувач потрапляє на сторінку перегляду відеокамер.

Макет сторінки зображено на рис. 2.9

Рисунок 2.9 - Вікно авторизації

Макет сторінки перегляду відеокамер зображено на рис.2.10. Вона призначена для відображення відео потоку з камер спостереження, моніторингу їх стану та сповіщення чергового працівника про небезпеку у разі виявлення ознак пожежі.

Сторінка містить бокову панель на якій користувач має можливість обрати перегляд відеоряду з камер спостереження або можливість завантаження демонстраційного відео.

В нижній частині сторінки розташоване поле відображення статусу, що міститиме інформацію про ділянку камер, а саме назву, ідентифікатор та координати розташування, інформацію про наглядача, а саме ім’я та ідентифікатор та інформацію про поточну локацію, а саме назву та координати розташування. У центрі сторінки відображається група камер або окрема камера. При виявленні ознак вогню на якійсь з камер біля неї на нижній панелі має з’являтися сигнал у вигляді напису «FIRE», натиснувши на який користувач побачить вікно з деталями.

Рисунок 2.10 - Головна сторінка з переглядом відеокамер

Сторінка перегляду тестового відео призначена для демонстрації роботи системи. Вона дає можливість завантажити попередньо збережене на локальний комп’ютер відео та застосувати до нього навчену модель НМ.

Рисунок 2.11 - Головна сторінка з переглядом демонстраційного відео

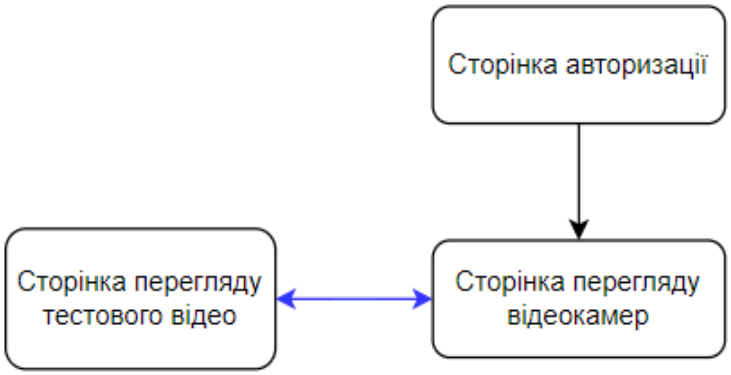

Рисунок 2.12 - Переходи між сторінками для підсистеми моніторингу

Для підсистеми підприємства надавача сервісу передбачено сторінки: авторизації та сторінку перегляду та керування локаціями.

Сторінка авторизації призначена для авторизації адміністратора в системі.

Головна сторінка призначена для перегляду всіх наявних у системі локацій, їх додавання та вилучення.

Сторінка перегляду та керування локаціями призначена для редагування локацій, налаштування параметрів системи для підприємства, надання доступу до системи відповідальним особам.

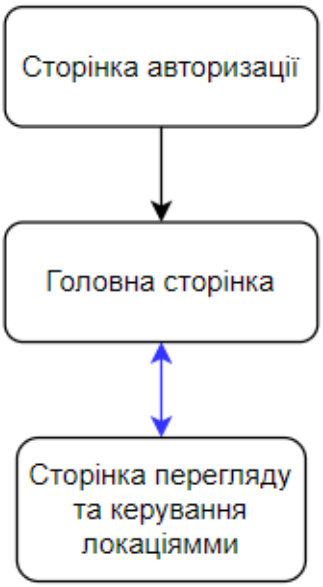

Рисунок 2.13 - Переходи між сторінками для підсистеми надавача сервісу

Для підсистеми адміністрування на рівні об’єкта спостереження передбачено сім сторінок: сторінку авторизації, головну сторінку, сторінки перегляду та керування ділянками, камерами та працівниками, сторінку перегляду журналу чергування та сторінку перегляду відеокамер.

Сторінка авторизації призначена для входу користувача в систему.

Головна сторінка призначена для навігації та містить посилання на всі інші сторінки сайту.

Сторінка перегляду журналу чергувань призначена для редагування та перегляду розподілу працівників підприємства(локації) між ділянками спостереження.

Сторінки перегляду та керування працівниками, ділянками та камерами призначені для їх додавання, редагування та вилучення.

Сторінка перегляду відеокамери призначена для перегляду відео потоку з обраної на сторінці керування камери.

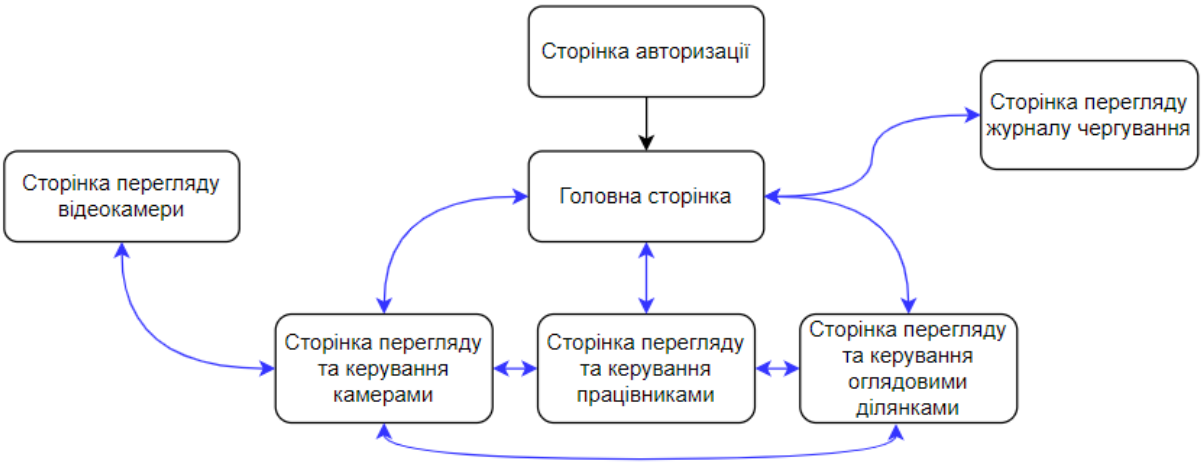

Рисунок 2.14 - Переходи між сторінками для підсистеми адміністрування на рівні об’єкта спостереження

Висновки

У ході роботи над проектуванням застосунку було розроблено IDEF0 діаграму системи, визначено складові підсистеми майбутнього застосунку, функції кожної з підсистем. Було визначено основні таблиці та спроектовано базу даних системи. Також було створено макети сторінок майбутнього веб-застосунку та розроблено карту сайту з переходами між сторінками.

РОЗДІЛ 3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Інструментальні засоби програмної реалізації

Для виконання роботи було використано мову програмування Python. Зокрема наступні бібліотеки:

OpenCV – бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована на C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов.

Бібліотека є дуже корисною при роботі з зображеннями та відео так як містить спеціально передбачені для цього модулі.

У роботі були використані функції VideoCapture та VideoWriter для зчитування та запису відеоряду. Також бібліотека дозволяє змінювати зображення. За допомогою функції rectangle та putText передбачення нейронної мережі візуально відображаються на кадрах вихідного відеопотоку.

OpenCV DNN. Цей модуль містить: API для створення нових шарів нейронних мереж; набір вбудованих найкорисніших шарів; API для побудови та модифікації комплексних нейронних мереж із шарів; функціональність для завантаження моделей серіалізованих мереж з різних фреймворків.

Функціональність цього модуля призначена лише для обчислень прямого поширення(тобто тестування мережі). Навчання мереж в принципі не підтримується.

Даний модуль було використано для створення моделі з попередньо навченого набору вагів за допомогою функції readNetFromONNX().

Також модуль містить функцію blobFromImage(), яка приймає на вхід зображення та створює 4-вимірний blob із зображення. Додатково змінює розмір і обрізає зображення з центру, віднімає середні значення, масштабує значення за коефіцієнтом масштабування, міняє місцями синій і червоний канали

tensorflow.keras — це API глибокого навчання, написаний на Python, який працює на платформі машинного навчання TensorFlow. Він був розроблений з упором на можливість швидкого експериментування.

TensorFlow — це наскрізна платформа машинного навчання з відкритим кодом. Він поєднує в собі чотири ключові здібності:

- Ефективне виконання тензорних операцій низького рівня на CPU, GPU або TPU.
- Обчислення градієнта довільних диференційованих виразів.
- Масштабування обчислень для багатьох пристроїв, таких як кластери із сотень графічних процесорів.
- Експорт програм у зовнішні середовища виконання, такі як сервери, браузери, мобільні та вбудовані пристрої.

Keras — це високорівневий API TensorFlow. Це доступний, високопродуктивний інтерфейс для вирішення проблем машинного навчання з акцентом на сучасне глибоке навчання. Він надає основні абстракції та будівельні блоки для розробки та доставки рішень машинного навчання.

Також під час роботи над застосунком використовувалась CUDA — паралельна обчислювальна платформа розроблена NVIDIA для обчислень на графічних процесорах. CUDA дає можливість значно прискорити роботу обчислювальних програм, використовуючи потужність графічних процесорів.

3.2. Дослідження ефективності архітектур нейронних мереж

У ході роботи було розглянуто дві архітектури нейронної мережі та порівняно ефективність та доцільність їх використання для поставленої задачі, а саме CNN архітектуру та з нейронну мережу YOLOv5. Досліджено якість навчання мережі YOLOv5 на різній кількості ітерацій. Також розглянуто доцільність використання заздалегідь навчених вагів мережі замість навчання з нуля та використання наборів вагів різного розміру.

3.2.1. Порівняння YOLOv5 та CNN архітектури

При виконанні роботи було досліджено ефективність використання моделей різної архітектури для вирішення поставленого завдання. Було розглянуто мережу YOLOv5 та мережу CNN архітектури.

Мережа CNN

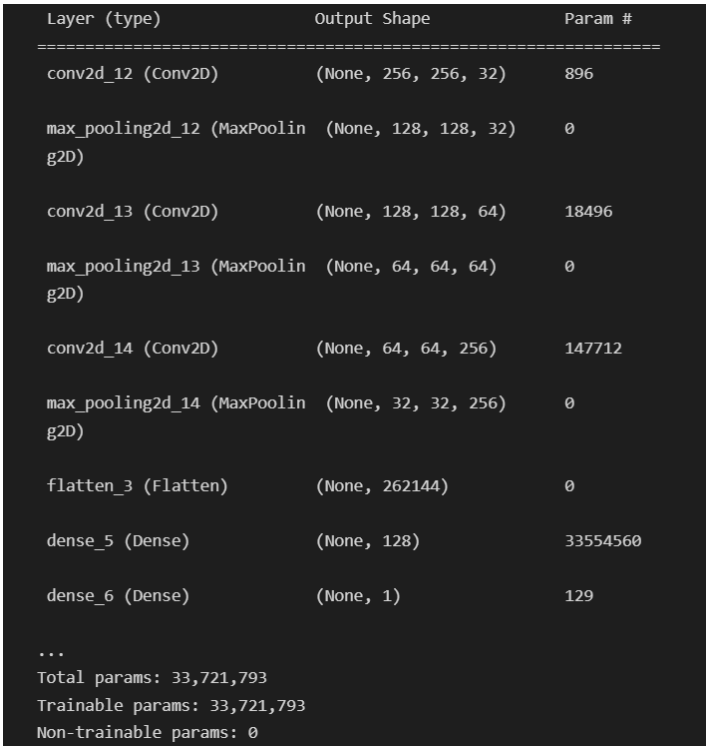

Рисунок 3.1 - Архітектура розробленої CNN мережі

Розроблену мережу було навчено продовж 300 епох з використанням оптимізатора “Adam” та функції втрат “binary_crossentropy”.

Датасет для навчання мережі містив зображення двох типів на яких є або немає вогню, тому при навчанні мережі вирішувалась задача бінарної класифікації. До зображень перед навчанням було застосовано деякі перетворення для різноманітності навчальних даних, а саме: масштабування, горизонтальне та вертикальне обертання.

Дані було розділено на тренувальну та валідаційну вибірки.

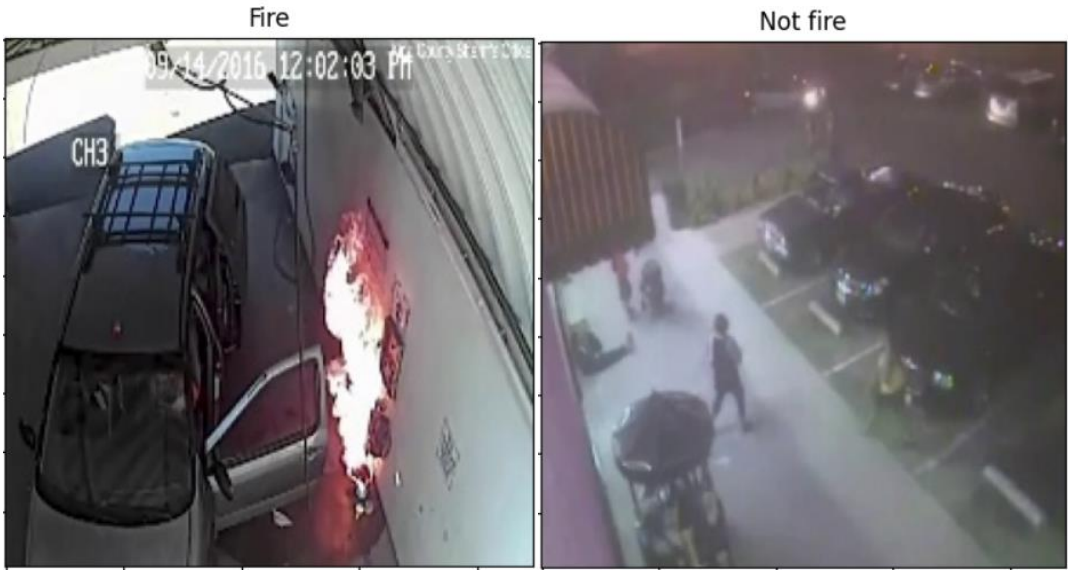

Рисунок 3.2 - Результат класифікації зображень з валідаційної вибірки

Рисунок 3.3 - Результат класифікації тестового зображення

Як можна побачити з отриманих результатів, на рисунку 3.2 зображення, що належать до тренувальної вибірки мережа класифікує вірно, а на рисунку 3.3, з тестової вибірки, невірно. Отже, отримана мережа добре справляється з класифікацією зображень що входили до навчального датасету, проте не здатна з задовільною точністю класифікувати інші зображення. Також недоліком є те що отримана мережа здатна лише розпізнавати два класи: вогонь та його відсутність.

Мережа YOLO

	from	n	params	module	arguments
0	-1	1	3520	models.common.Conv	[3, 32, 6, 2, 2]
1	-1	1	18560	models.common.Conv	[32, 64, 3, 2]
2	-1	1	18816	models.common.C3	[64, 64, 1]
3	-1	1	73984	models.common.Conv	[64, 128, 3, 2]
4	-1	2	115712	models.common.C3	[128, 128, 2]
5	-1	1	295424	models.common.Conv	[128, 256, 3, 2]
6	-1	3	625152	models.common.C3	[256, 256, 3]
7	-1	1	1180672	models.common.Conv	[256, 512, 3, 2]
8	-1	1	1182720	models.common.C3	[512, 512, 1]
9	-1	1	656896	models.common.SPPF	[512, 512, 5]
10	-1	1	131584	models.common.Conv	[512, 256, 1, 1]
11	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0	models.common.Concat	[1]
13	-1	1	361984	models.common.C3	[512, 256, 1, False]
14	-1	1	33024	models.common.Conv	[256, 128, 1, 1]
15	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
16	[-1, 4]	1	0	models.common.Concat	[1]
17	-1	1	90880	models.common.C3	[256, 128, 1, False]
18	-1	1	147712	models.common.Conv	[128, 128, 3, 2]
19	[-1, 14]	1	0	models.common.Concat	[1]
20	-1	1	296448	models.common.C3	[256, 256, 1, False]
21	-1	1	590336	models.common.Conv	[256, 256, 3, 2]
22	[-1, 10]	1	0	models.common.Concat	[1]
23	-1	1	1182720	models.common.C3	[512, 512, 1, False]
24	[17, 20, 23]	1	18879	models.yolo.Detect	[2, [[10, 13, 16, 30, 33

Рисунок 3.4 - Архітектура мережі YOLO v5

На відміну від попередньої моделі, розглянута архітектура YOLO здатна розпізнавати вогонь на зображеннях що не належать до тренувальної вибірки.

Також перевагою цієї моделі є те, що вона здатна знаходити на зображенні навіть дуже малі об’єкти.

Рисунок 3.5 - Результат розпізнавання мережею YOLO

Також, як видно на рисунку 3.5, вона здатна розпізнавати декілька об’єктів на одному зображенні.

Переваги і недоліки розглянутих підходів.

Обидві навчені моделі були здатні розпізнати на вхідному зображенні наявність вогню. У другому випадку мережа виявилась точнішою для даних що не входили в навчальний датасет.

Друга мережа здатна розпізнавати наявність та положення фрагментів на зображенні що містять ознаки вогню за їх наявності, навіть якщо їх декілька на одному зображенні, в той час як перша здатна лише класифікувати подане на вхід зображення і стає непередбачуваною у випадку використання зображень що сильно відрізняються від навчальних даних.

Так як поставленим завданням є розробка модулю що буде здатен працювати з відеопотоком, можна зробити висновок що мережа YOLO краще підходить для цієї задачі.

3.2.2. Якість результатів в залежності від тривалості навчання

Також було проведено дослідження залежності якості розпізнавання від кількості епох навчання мережі. Було розглянуто випадки для 5, 10, 15 та 20 епох.

Для кожного випадку отримано показники P(Precision) та R(Recall).

Precision визначає, наскільки точні отримані прогнози, тобто відсоток правильних серед усіх передбачень.

Recall вимірює, наскільки добрими є дійсно позитивні передбачення, тобто відсоток правильно розпізнаних серед усіх можливих позитивних кейсів. [6]

$$Precision = TP/ (TP + FP), \qquad Recall = TP/ (TP + FN)$$

TP - True positive

TN - True negative

FP - False positive

FN - False negative

З рис. 3.6 – 3.7 можна побачити що показники P та R ростуть зі збільшенням кількості епох, проте різниця між останніми двома моделями незначна, тому подальше збільшення часу навчання не є доцільним. Найкращі показники були отримані при 20 епохах навчання мережі.

Аналогічною є тенденція для кривих похибки знаходження об’єктів в обмежувальних рамках, зображених на рис. 3.8.

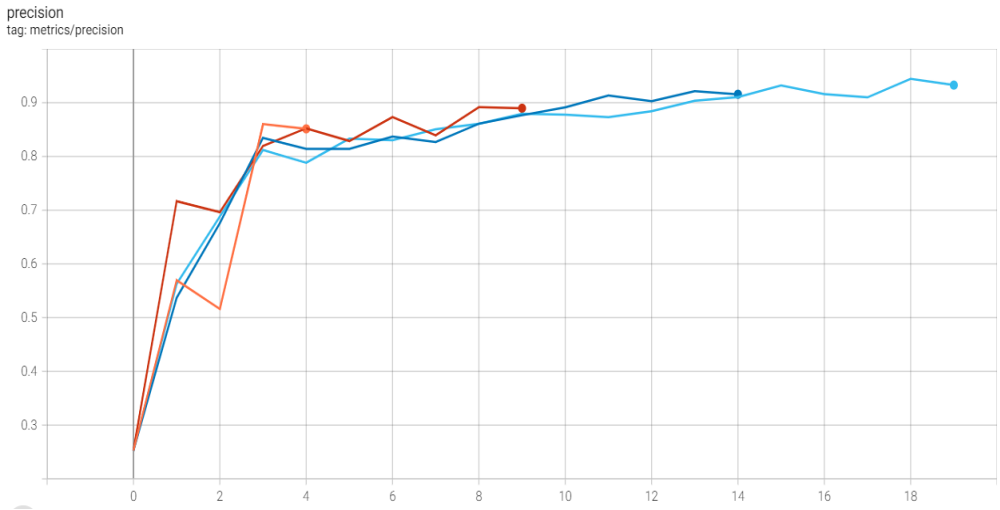

Рисунок 3.6 – Порівняння кривих значень P для моделей навчених на 5, 10, 15 та 20 епохах

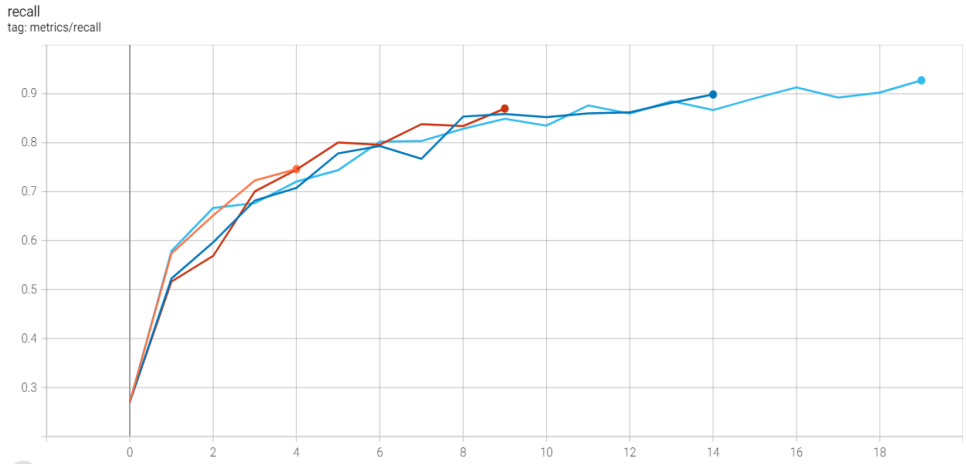

Рисунок 3.7 – Порівняння кривих значень R для моделей навчених на 5, 10, 15 та 20 епохах

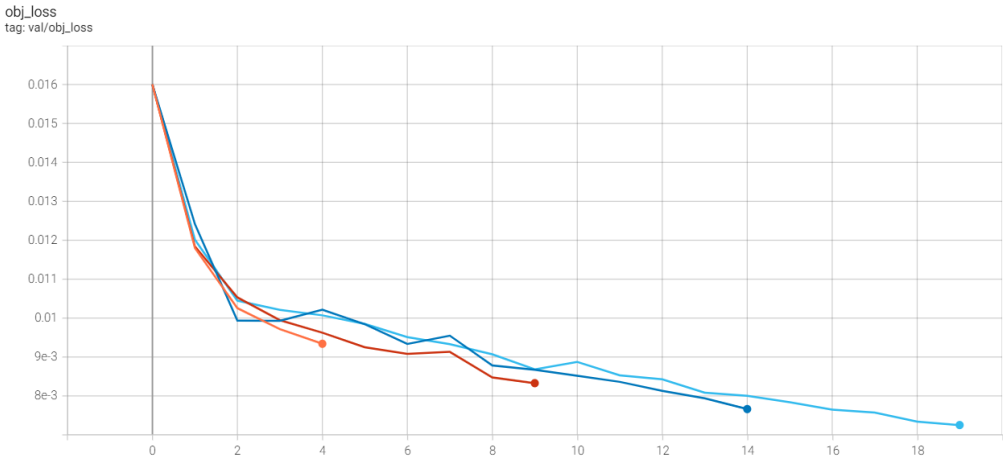

Рисунок 3.8 – Порівняння кривих значень похибки знаходження об’єктів в обмежувальних рамках для моделей навчених на 5, 10, 15 та 20 епохах

3.2.3. Доцільність використання попередньо навченої моделі

YOLO v5 дає можливість як навчання мережі з нуля, так і використання вже переднавченої моделі. У ході роботи було досліджено доцільність використання кожного з підходів.

Для дослідження було використано набір вагів "medium", що містить 12.9 мільйонів параметрів. В якості переднавченої моделі було використано ваги з файлу “yolov5m.pt”, що заздалегідь навчені да датасеті COCO, що містить 80 класів різних об’єктів.

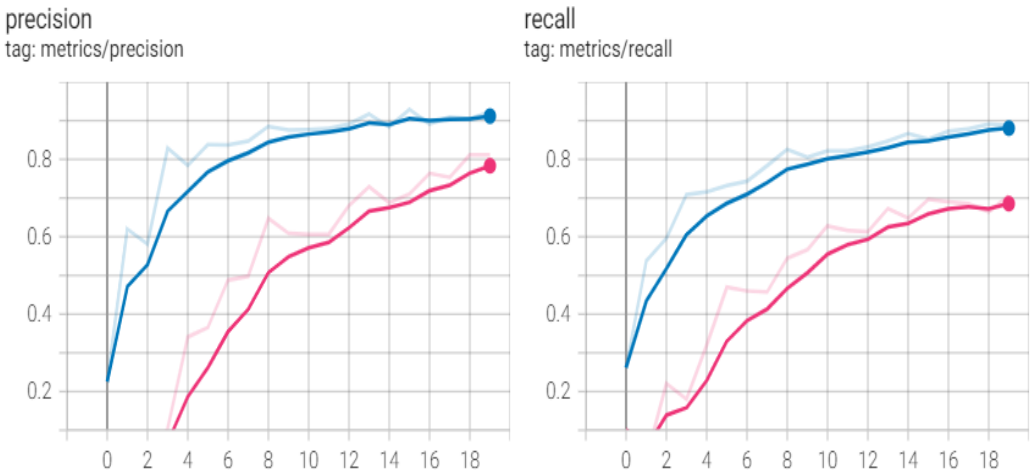

Рисунок 3.9 - Порівняння значень метрик Precision та Recall

Рисунок 3.10 - Порівняння значень метрики obj_loss для навчальної вибірки

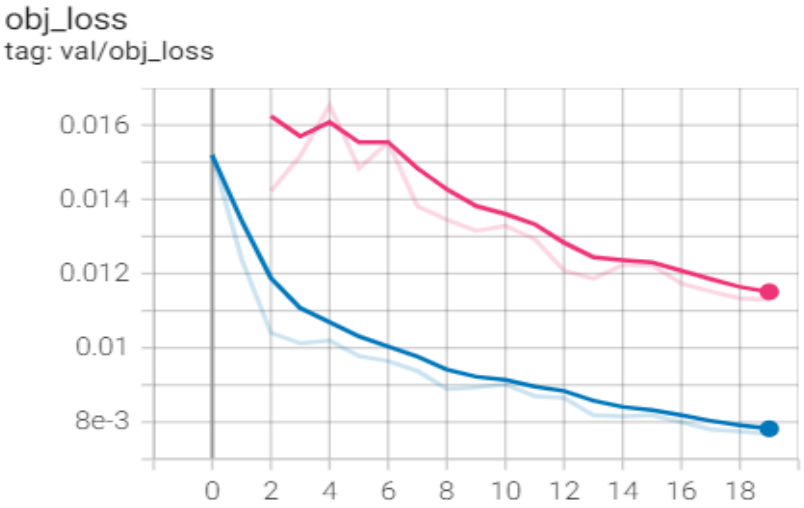

Рисунок 3.11 - Порівняння значень метрики obj_loss для валідаційної вибірки

На графіках на рис.3.9 – 3.11 відображено криві змінення значень Precision, Recall та втрат, де рожева крива відповідає навчанню з нуля і блакитна передначеній моделі.

З наведених вище результатів навчання кожної з мереж видно, що при використанні переднавчених вагів з тими самими параметрами навчання та на тій самій кількості епох, другий варіант на початку навчання вже є точнішим та значно швидше дає кращі результати ніж перший, отже його використання є доцільним.

3.2.4. Дослідження ефективності використання наборів вагів різного розміру

Мережа YOLO v5 здатна працювати з наборами вагів різного розміру.

Таблиця 3.1 - Кількість параметрів моделей

Модель	Кількість параметрів(M)
yolov5n	2.5
yolov5s	5.4
yolov5m	12.9
yolov5l	26.5
yolov5x	48.1

Дослідження було проведено на наборах вагів small(s), medium(m) та large(l).

На отриманих графіках на рис. 3.12-3.13 моделі YOLOv5s відповідає помаранчева крива, моделі YOLOv5m - блакитна та моделі YOLOv5l - червона.

З отриманих результатів вино, що показники якості навчання покращуються при збільшенні розміру моделі, проте різниця між другою та третьою моделями є не великою, в той час як навчання моделі yolov5l займає більше часу. Тому доцільним є використання набору вагів yolov5m.

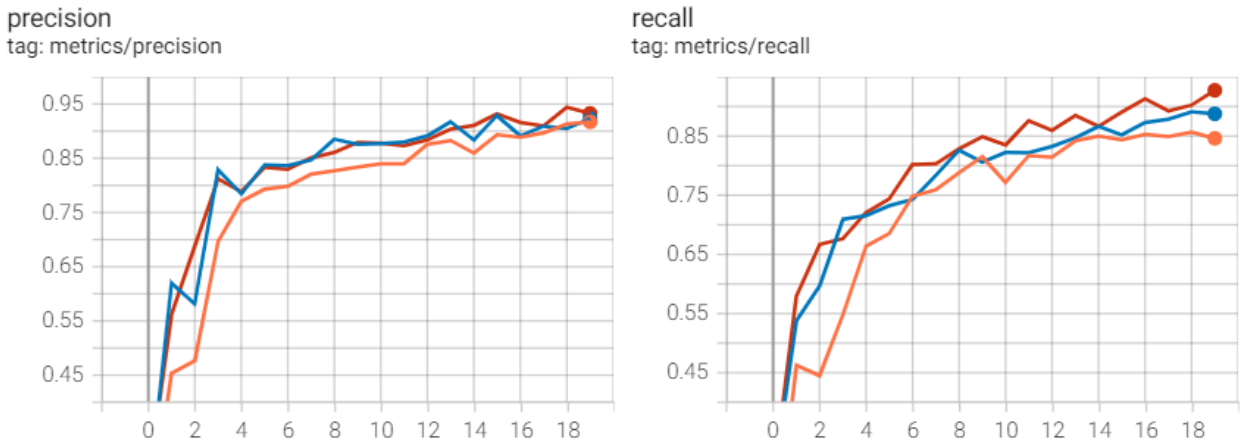

Рисунок 3.12 - Порівняння значень метрик Precision та Recall

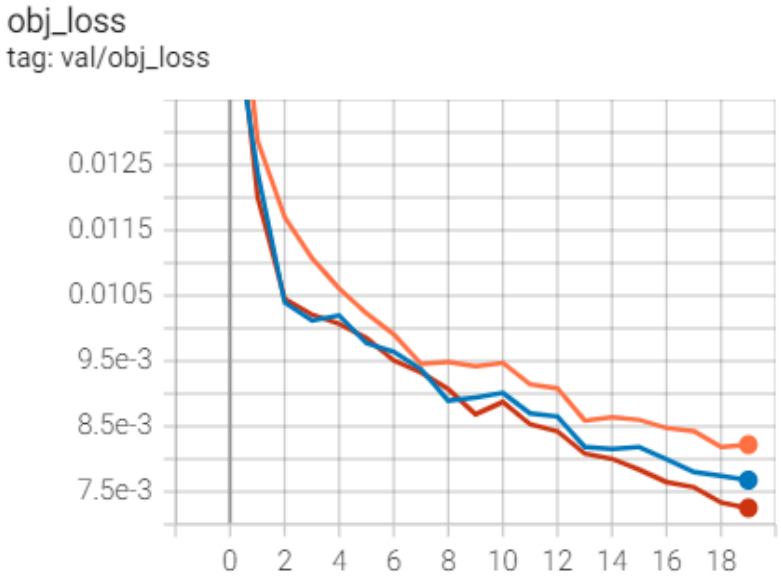

Рисунок 3.13 - Порівняння значень метрики obj_loss для валідаційної вибірки

У таблицях 3.2 – 3.4 показано порівняння результатів навчання всіх розглянутих в дослідженнях моделей з вказанням їх параметрів.

У таблиці 3.2 наведено дослідження якості результатів в залежності від тривалості навчання. З отриманих даних видно що найкращий результат дало навчання моделі на 20 епохах.

Таблиця 3.2 – Результати дослідження тривалості навчання

Опис моделі			P	R	Час навчання
Набір вагів	Кількість епох навчання	Попередньо навчені ваги			
yolov5m	5	так	0.8501	0.7432	6хв15с
yolov5m	10	так	0.8912	0.8748	8хв34с
yolov5m	15	так	0.9258	0.9002	13хв18с
yolov5m	20	так	0.9431	0.9376	16хв52с

У таблиці 3.3 наведено дослідження якості результатів в залежності від розміру набору вагів. З отриманих даних видно що найкращий результат дало навчання моделі yolov5l, проте значення метрик не набагато кращі ніж в моделі yolov5m, при значно меншому затраченому часі.

Таблиця 3.3 – Результати дослідження наборів вагів різних розмірів

Опис моделі			P	R	Час навчання
Набір вагів	Кількість епох навчання	Попередньо навчені ваги			
yolov5s	20	так	0.9171	0.8461	14хв38с
yolov5m	20	так	0.9431	0.9376	16хв41с
yolov5l	20	так	0.9527	0.9450	22хв50с

У таблиці 3.4 наведено різницю якості результатів між використанням попередньо навченого набору вагів та навчанням з нуля.

З отриманих даних видно що кращий результат має попередньо навчена модель.

Таблиця 3.4 – Використанням попередньо навченого набору вагів та навчанням з нуля

Опис моделі			P	R	Час навчання
Набір вагів	Кількість епох навчання	Попередньо навчені ваги			
yolov5m	20	так	0.9431	0.9376	16хв41с
yolov5m	20	ні	0.8118	0.7055	17хв22с

3.3. Програмна архітектура розробленого модулю розпізнавання пожеж у відеопотоці

В результаті роботи було створено програмний модуль що використовує навчену модель та здатен застосовувати її до відеоряду отриманого з публічно доступної відеокамери за посиланням, визначає наявність вогню, помічає знайдений об’єкт прямокутником та відображає результат в режимі онлайн або записує та зберігає його до відеофайлу.

Розроблена система має структуру, зображену на рис. 3.14.

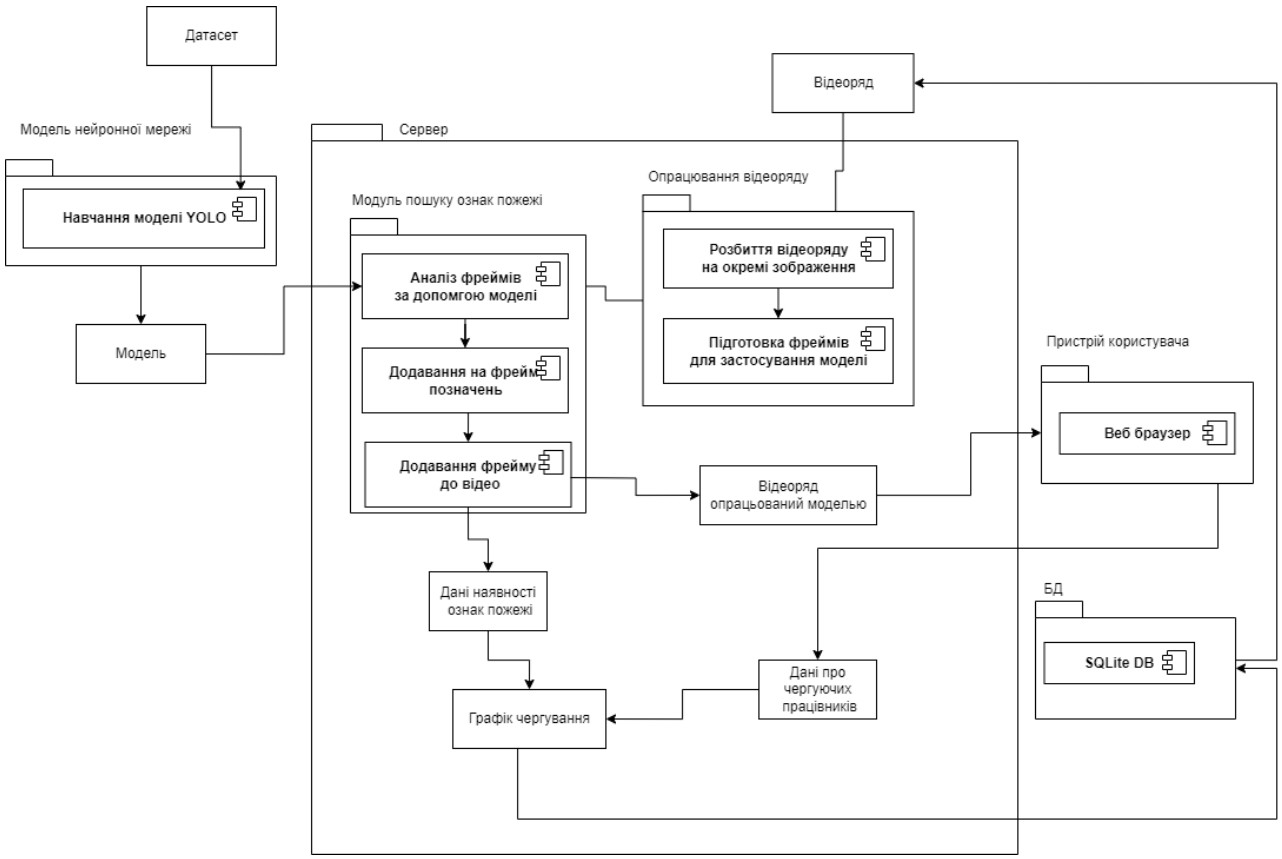

Рисунок 3.14 - Діаграма компонентів програмного застосунку

Навчання моделі відбувається у середовища GoogleColab, що надає ресурси для швидкого навчання моделі. У результаті отримується файл навчених вагів «weights.pt». Так як біблотека OpenCV не підтримує роботу з форматом «.pt», потрібно експортувати отриманий результат у файл з розширенням «.onnx», що дасть можливість завантажити навчену мережу до головного модуля програми.

Далі на вхід програми подається відео, що розбивається на окремі фрейми. Кожен фрейм подається на вхід нейронної мережі, у результаті чого отримується набір обмежувальних рамок(bounding boxes), кожна з яких містить координати центру, ширину та висоту потенційного об’єкту, ймовірність знаходження у даній рамці одного з шуканих класів, та набір ймовірностей його належності до кожного з них.

Потім ці набори фільтруються таким чином, що залишаються лише ті, ймовірність знаходження об'єкта в яких є більшою за 0.6.

Після чого по даним з кожного набору, що залишився на поточному фреймі, вимальовується область, яку займає об’єкт знайденого класу та мітка з назвою класу. Після чого змінений фрейм записується до вихідного відео.

3.4. Тестування працездатності системи та інструкція користувача

Після авторизації на сайті користувач потрапляє на сторінку перегляду трансляцій з відеокамер, зображену на рисунку 3.15

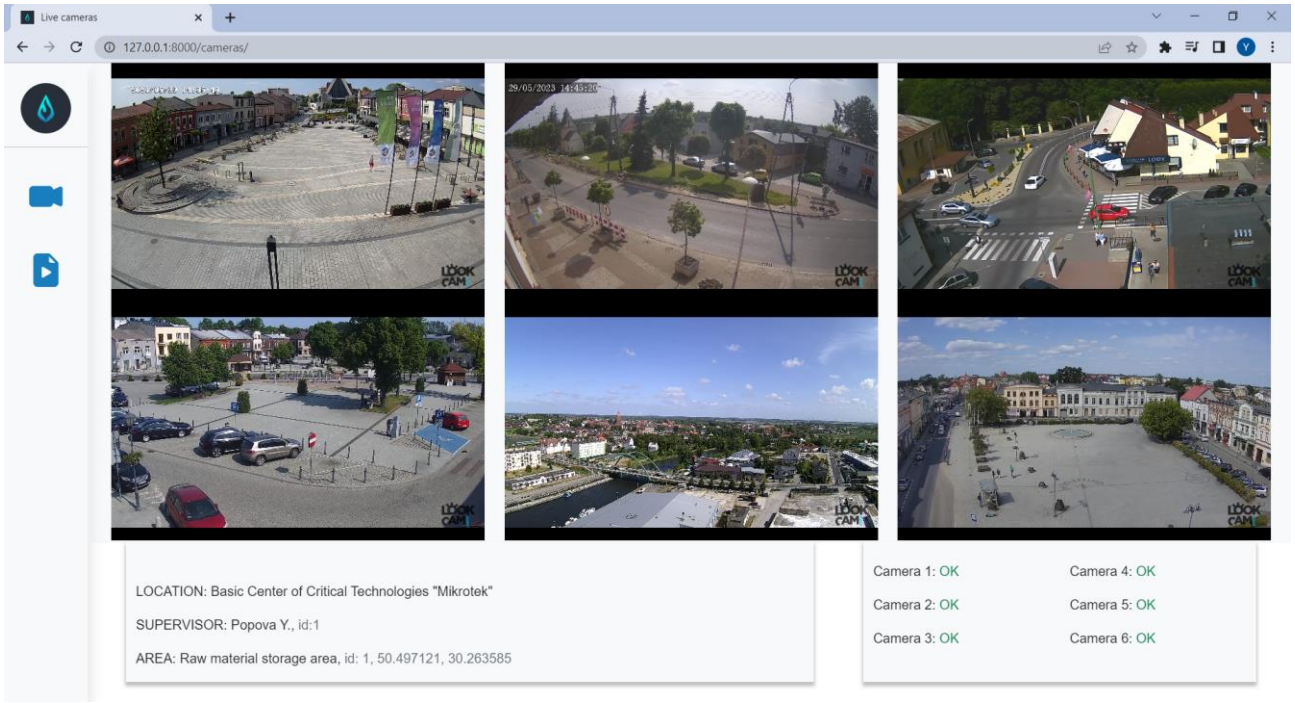

Рисунок 3.15 – Сторінка перегляду трансляцій

На цій сторінці в режимі реального часу відеопотік обробляється мережею та результати мають відображатись на самому відео потоці та на нижній панелі статусів.

З поточної сторінки користувач може потрапити на сторінку тестування системи зображену на рис.3.16, натиснувши другу згори кнопку на боковій панелі.

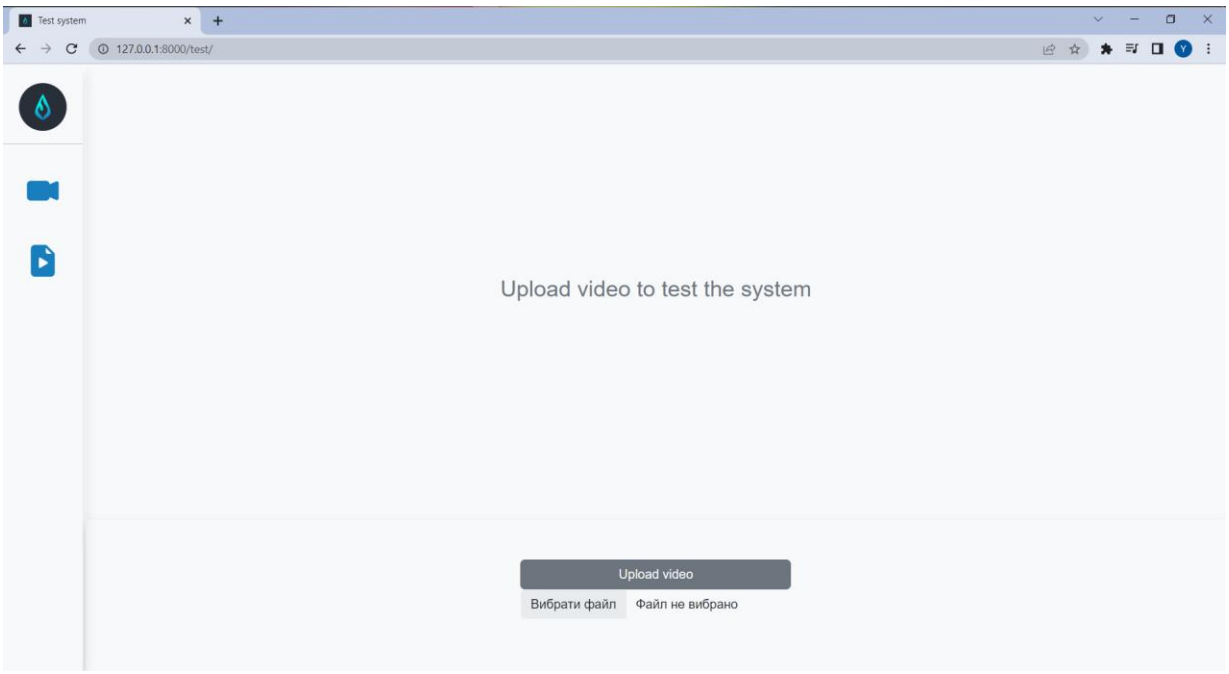

Рисунок 3.16 – Сторінка тестування СРОВП

Щоб розпочати тестування користувач повинен завантажити заздалегідь підготовлене відео обравши його та натиснувши на кнопку на нижній панелі. Цей процес зображено на рис.3.17. На рис.3.18 зображено попередній перегляд тестового відео.

Після завантаження відео запуститься модель нейронної мережі. Далі користувач повинен почекати поки все відео буде опрацьоване, після чого воно з’явиться на головній частині.

Результат опрацювання завантаженого відеоряду представлено на рис.3.19.

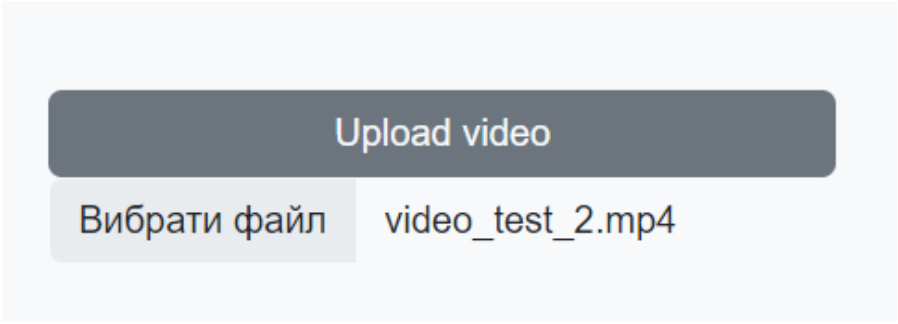

Рисунок 3.17 – Завантаження тестового відео з локального комп’ютера

Рисунок 3.18 – Попередній перегляд тестового відео

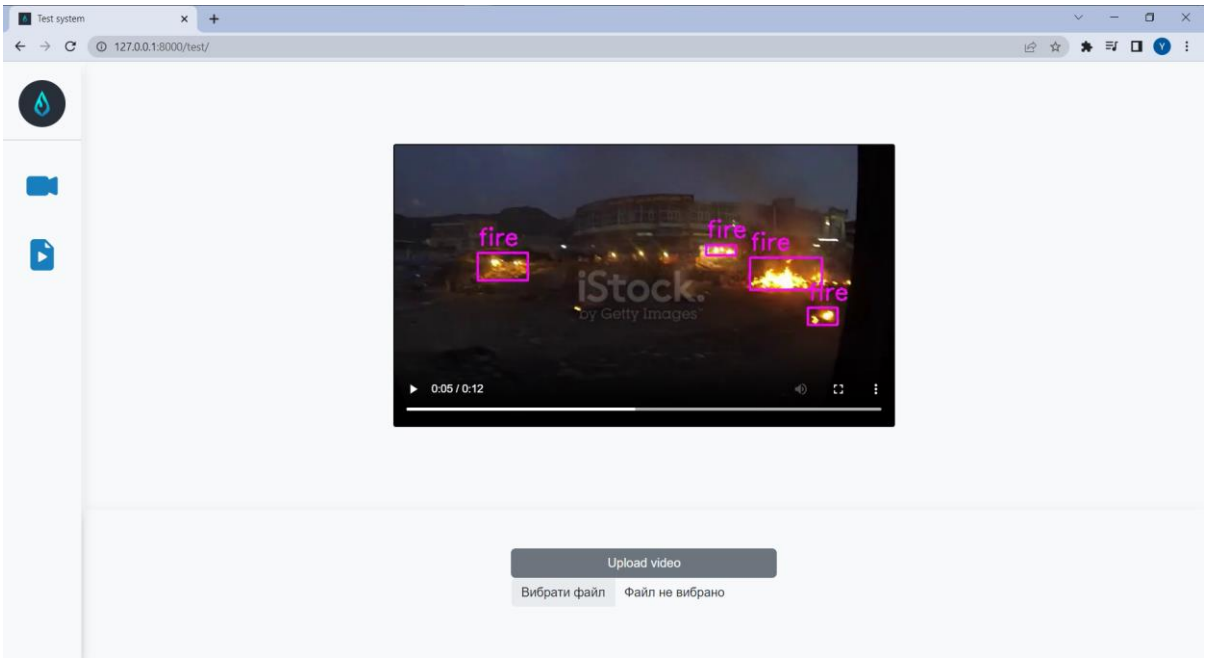

Рисунок 3.19 – Демонстрація опрацьованого НМ відео

Висновки

У ході виконання практичної частини було проведено ряд досліджень і було виявлено що нейронна мережа YOLOv5 навчена на 20 епохах на наборі переднавчених вагів найкраще підходить для поставленого завдання. Найкраще

співвідношення отриманої якості розпізнавання та затраченого на навчання часу має модель YOLOv5m.

Було розроблено програмний модуль, що відповідає розробленим функціональним вимогам, який приймає на вхід відеоряд, застосовує до нього навчену модель та зберігає результат у вигляді відеоряду з нанесеними позначками розташування та найменування знайдених об’єктів (fire).

ВИСНОВОК

У ході роботи було виконано поставлені завдання, а саме проаналізовано значущість проблеми пожеж на підприємствах, проаналізовано вже існуючі методи їх виявлення. Було проаналізовано наукові роботи в області використання ШІ для розпізнавання зображень, розглянуто різні архітектури нейронних мереж та проведено дослідження їх ефективності при різних параметрах навчання.

В результаті було виявлено, що мережа YOLOv5 дає найбільше ефективні результати.

Було спроектовано архітектуру застосунку, розроблено БД.

Результатом роботи є розроблений програмна система, що з використанням навченої нейронної мережі здатна розпізнавати ознаки пожежі на відеоряді з публічних камер відеоспостереження а також з відеозаписів.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Avazov, Kuldoshbay, et al. "Fire Detection Method in Smart City Environments Using a Deep-Learning-Based Approach." Electronics 11.1 (2021): 73. - Режим доступу: https://www.mdpi.com/2079-9292/11/1/73
2. Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., ... & Ghayvat, H. (2021), CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope – Режим доступу: CNN Variants for Computer Vision
3. Celik, Turgay, Hüseyin Özkaramanlı, and Hasan Demirel. (2007, April) Fire pixel classification using fuzzy logic and statistical color model. - Режим доступу: Fire-Pixel-Classification-using-Fuzzy-Logic-and-Statistical-Color-Model
4. Çelik, Turgay, Hüseyin Özkaramanlı, and Hasan Demirel. 2007 Fire and smoke detection without sensors: Image processing based approach. 15th European Signal Processing Conference. - Режим доступу: Fire and Smoke Detection without Sensors: Image Processing Based Approach
5. Hussain, Mahbub, Jordan J. Bird, and Diego R. Faria. UK Workshop on computational Intelligence. Springer, Cham, 2018. A study on cnn transfer learning for image classification. - Режим доступу: A-Study-on-CNN-Transfer-Learning-for-Image-Classification
6. Jonathan Hui mAP (mean Average Precision) for Object Detection [Електронний ресурс] - Режим доступу: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173
7. Michael Nielsen Neural Networks and Deep Learning 2019. [Електронний ресурс] - Режим доступу: http://neuralnetworksanddeeplearning.com/index.html

8. Mseddi, Wided Souidene, et al. "Fire detection and segmentation using YOLOv5 and U-net." 2021. - Режим доступу: https://eurasip.org/Proceedings/Eusipco/Eusipco2021/pdfs/0000741.pdf
9. Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation. - Режим доступу: https://arxiv.org/pdf/1505.04597.pdf%EF%BC%89
10. Park, S. S., Tran, V. T., & Lee, D. E. Application of various yolo models for computer vision-based real-time pothole detection. Applied Sciences, 2021 – Режим доступу: https://www.mdpi.com/2076-3417/11/23/11229
11. Qureshi, W. S., Ekpanyapong, M., Dailey, M. N., Rinsurongkawong, S., Malenichev, A., & Krasotkina, O. QuickBlaze: early fire detection using a combined video processing approach. Fire technology. (2016). - Режим доступу: Early-Fire-Detection-Using-a-Combined-Video-Processing-Approach
12. Thuan, Do. (2021) Evolution of Yolo algorithm and Yolov5: The State-of-the-Art object detection algorithm. Thuan, Do. (2021). - Режим доступу: https://www.theseus.fi/bitstream/handle/10024/452552/Do_Thuan.pdf?sequence=2&isAllowed=y
13. Umar, Malik Mohamed (2017) State of the art of smoke and fire detection using image processing. . - Режим доступу: State of the art of smoke and fire detection using image processing
14. Згорткові нейронні мережі 2022 - Режим доступу: https://drive.google.com/file/d/1fbpCMgySE_3RRHOqZ56UTgLGr59lXncr/view

ДОДАТКИ

Додаток А. Програмний код модуля НМ

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython import display

net = cv2.dnn.readNetFromONNX('yolov5-master/best.onnx')
classes = [ 'fire']
colors = [[255, 0, 255], [0, 255, 255]]

def drawObject(img, x, y, h, w, class_indedx):
    height, width = img.shape[0], img.shape[1]
    x_scale = width/640
    y_scale = height/640

    x1, y1 = int((x - w//2)*x_scale), int((y - h//2)*y_scale)
    x2, y2 = int((x + w//2)*x_scale), int((y + h//2)*y_scale)

    img = cv2.rectangle(
        img,
        (x1, y1), (x2, y2),
        color = colors[class_indedx],
        thickness = 2
    )

    font_scale = min(1, max(3,int(w/50)))
    font_thickness = min(2, max(10,int(w/50)))

    img = cv2.putText(
        img, classes[class_indedx],
        (x1+1, y1 - 10),
        cv2.FONT_HERSHEY_SIMPLEX,
        font_scale,
        colors[class_indedx],
        font_thickness
    )
    return img

def apply_yolo(img):

    height, width = img.shape[0], img.shape[1]

    object_boxes = []
    object_probas = []
    object_classes = []

    blob = cv2.dnn.blobFromImage(img, scalefactor=1/255, size=(640,640), mean=[0, 0, 0], swapRB=True)
    net.setInput(blob)
    detection = net.forward()[0]

    for res in detection:

        cx, cy, h, w = res[:4]
```

```
            probas = res[5:]

            class_indedx = np.argmax(probas)
            pred_class_prob = probas[class_indedx]
            if res[4] > 0.6:
                object_boxes.append([cx, cy, w, h])
                object_probas.append(pred_class_prob)
                object_classes.append(class_indedx)

        filtered_objects_ind = cv2.dnn.NMSBoxes(object_boxes, object_probas, 0.4, 0.4 )

        for index in filtered_objects_ind:
            cx, cy, h, w = object_boxes[index]
            img = drawObject(img, cx, cy, h, w, object_classes[index])

        return img

size = (640, 360)
codec = cv2.VideoWriter_fourcc(*'MJPG')
cap = cv2.VideoCapture('video_test_2.mp4')
writer = cv2.VideoWriter('video_test_output.avi', codec, 25, size)

while cap.isOpened():

    print('.', end='')
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv2.resize(frame, size)
    frame = apply_yolo(frame)

    writer.write(frame)

writer.release()
cap.release()

cap = cv2.VideoCapture('video_test_3.mp4')

while cap.isOpened():

    print('.', end='')
    ret, frame = cap.read()
    if not ret:
        break

    frame = apply_yolo(frame)

    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    plt.figure(figsize=(10,15))
    plt.imshow(frame)
    display.clear_output(wait=True)
    display.display(plt.gcf())

cap.release()
```

Додаток Б. Код навчання нейромережі

```
import os
import glob as glob
import cv2
import requests
import random
import numpy as np
import matplotlib.pyplot as plt
import zipfile


SEED = 42
np.random.seed(SEED)


TRAIN = True
EPOCHS = 50

if not os.path.exists('train'):

  with zipfile.ZipFile('Fire and Smoke Training.v1-fire-and-smoke-data-7-11-2022.yolov5pytorch.zip', 'r') as zip_ref:
     zip_ref.extractall()
  dirs = ['train', 'valid', 'test']

  for i, dir_name in enumerate(dirs):
    all_image_names = sorted(os.listdir(f"{dir_name}/images"))
    for j, image_name in enumerate(all_image_names):
      if (j % 2) == 0:
        file_name = image_name.split('.jpg')[0]
        os.remove(f"{dir_name}/images/{image_name}")
        os.remove(f"{dir_name}/labels/{file_name}.txt")

class_names = ['fire', 'smoke']
colors = np.random.uniform(0, 255, size=(len(class_names), 3))


def set_res_dir():
  # dir to dtore res
  res_dir_count = len(glob.glob('runs/train/*'))
  print(f"Current number of result directories: {res_dir_count}")
  if TRAIN:
    RES_DIR = f"results_{res_dir_count+1}"
    print(RES_DIR)
  else:
    RES_DIR = f"results_{res_dir_count}"
  return RES_DIR


!pip install -r requirements.txt


monitor_tensorboard()

RES_DIR = set_res_dir()
if TRAIN:
  !python train.py --data ../data.yaml --weights yolov5l.pt  --img 640 --epochs 30 --batch-size 16 --name {RES_DIR}
```

```
!zip -r lesult_100_16.zip runs/train/results_3

def show_valid_results(RES_DIR):
  !ls runs/train/{RES_DIR}
  EXP_PATH = f"runs/train/{RES_DIR}"
  validation_pred_images = glob.glob(f"{EXP_PATH}/*_pred.jpg")
  print(validation_pred_images)
  for pred_image in validation_pred_images:
      image = cv2.imread(pred_image)
      plt.figure(figsize=(19, 16))
      plt.imshow(image[:, :, ::-1])
      plt.axis('off')
      plt.show()

def inference(RES_DIR, data_path):
  # Directory to store inference results.\n",
  infer_dir_count = len(glob.glob('runs/detect/*'))
  print(f"Current number of inference detection directories: {infer_dir_count}")
  INFER_DIR = f"inference_{infer_dir_count+1}"
  print(INFER_DIR)
  # Inference on images.\n",
  !python detect.py --weights runs/train/{RES_DIR}/weights/best.pt \
  --source {data_path} --name {INFER_DIR}
  return INFER_DIR

rnd_array = np.random.randint(0, 50, size=10)

def visualize(INFER_DIR):
  INFER_PATH = f"runs/detect/{INFER_DIR}"
  infer_images = glob.glob(f"{INFER_PATH}/*.jpg")
  for i,pred_image in enumerate(infer_images):
    if i == 10:
      break
    image = cv2.imread(infer_images[rnd_array[i]])
    plt.figure(figsize=(12,9))
    plt.imshow(image[:,:,::-1])
    plt.axis('off')
    plt.show()

show_valid_results(RES_DIR)

IMAGE_INFER_DIR = inference(RES_DIR, '../test/images')

visualize(IMAGE_INFER_DIR)

!python detect.py --weights runs/train/results_3/weights/best.pt \
  --source ../test.png --name images_test
```