Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Використання глибинної нейронної мережі для покращення якості відео тенісних матчів
(тема)

Виконав:
студент 2 курсу, групи СШІм-20-2
Асландуков М. М.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Бодянський Є. В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Асландукову Матвію Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Використання глибинної нейронної мережі для покращення якості відео тенісних матчів

затверджена наказом університету від 24 березня 20 22 р. № 414 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 16 травня 20 22 р.

3. Вихідні дані до роботи глибинна нейронна мережа, алгоритм для відновлення неперервної траєкторії, набір даних для тренування та тестування системи, мова програмування Python, фреймворки tensorflow, keras, numpy, matplotlib

4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз предметної галазі і постановка задачі, огляд метрик оцінювання якості, аналіз підходів до вирішення задачі детекції м'яча та генерації неперервної траєкторії, розробка глибинної нейронної мережа та її навчання, розробка алгоритму синхронізації результатів нейронної мережі у траєкторію, експериментальна оцінка ефективності методів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) діаграма кількості вболівальників найпопулярніших видів спорту, архітектура запропонованої нейронної мережі, графіки залежності функції втрат та метрики якості від кількості пройдених епох навчання, графік залежності метрики BIoU від порогового значення, графік розподілу похибки створеної системи, приклад відновлення траєкторії за наявними координатами

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	31.03.2022	виконано
2	Огляд існуючих метрик оцінювання якості	03.04.2022	виконано
3	Аналіз підходів вирішення задачі детекції	05.04.2022	виконано
4	Створення архітектури нейронної мережі	09.04.2022	виконано
5	Проведення експериментальних досліджень	14.04.2022	виконано
6	Написання пояснювальної записки	23.04.2022	виконано
7	Нормоконтроль	30.04.2022	виконано
8	Перевірка на академічний плагіат	04.05.2022	виконано
9	Підготовка презентації та доповіді	10.05.2022	виконано
10	Попередній захист	11.05.2022	виконано
11	Рецензування	13.05.2022	виконано
12	Захист перед ЕК	16.05.2022	

Дата видачі завдання 28 березня 20 22 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 68 с., 20 рис., 3 табл., 22 формули, 23 джерела.

ВИБІРКА, ГЛИБИННА НЕЙРОННА МЕРЕЖА, ВІДЕО, ТЕНІС, ТРАЄКТОРІЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА.

Об’єкт дослідження – задача покращення якості відео тенісних матчів.

Предмет дослідження – використання глибинної нейронної мережі для покращення якості відео.

Мета роботи – створення системи для генерації відео з підвищеною якістю, шляхом створення архітектури нейронної мережі, її навчання, та подальшого використання отриманих результатів для генерації.

Методи дослідження – теоретичний (збір та структуризація теоретичного матеріалу), експериментальний (програмна реалізація нейронної мережі та її навчання). Методи розробки базуються на технологіях Python з фреймворком tensorflow.

У результаті роботи проведено теоретичний аналіз архітектур нейронних мереж, методи їх оптимізації, а також альтернативні методи вирішення поставленої задачі та метрики якості для їх оцінювання. Для оптимізації параметрів нейронної мережі були проаналізовані й використані існуючі набори даних, що складаються з близько 20000 кадрів відео реальних тенісних матчів. Аналогічні набори даних були використані й для порівняння результативності й продуктивності розробленої системи з альтернативними методами розв’язання задачі.

ABSTRACT

Explanatory note: 68 p., 20 figures, 3 tables, 22 formulas, 23 sources.

DATASET, DEEP NEURAL NETWORK, VIDEO, TENNIS, TRAJECTORY, CONVOLUTIONAL NEURAL NETWORK

The object of the research is the problem of improving quality of videos from tennis matches.

The subject of the research is using of deep neural network for improving quality of such videos.

The research methods are based on theoretical approaches (collection and structuring of theoretical materials) and experimental ones (implementation of neural network and its training). Development methods are based on Python technologies with tensorflow framework.

As a result, the theoretical analysis of neural network architectures, methods of their optimization, as well as alternative methods of solving the problem and quality metrics for their evaluation were conducted. To optimize the neural network parameters, existing data sets consisting of about 20,000 frames of video from real tennis matches were used. Similar datasets were used to compare the quality and efficiency of the developed system with alternative methods of solving the problem.

ЗМІСТ

ВСТУП

Сьогодні в життя людей все більш і більш проникають електронні пристрої: телефони, комп'ютери, фотоапарати, відеокамери, та багато інших. Усі вони породжують надзвичайно велику кількість різноманітної інформації: тексти, документи, зображення, мультимедіа, відео, та безліч інших поширених або вузько спеціалізованих форматів.

Без перебільшення можна сказати, що саме графічна інформація впевнено посідає перше місце за необхідним об'ємом для зберігання. Наприклад, художній фільм може займати близько 10 гігабайтів, що в тисячі разів більше декількох мегабайтів, необхідних для зберігання першоджерельної текстової книги. В той же час, існує легка можливість зниження якості як окремих зображень, так і одразу всіх кадрів відео шляхом пропорційного зменшення їх розміру. При цьому загальна структура зображення залишається майже незмінною для людського ока, що викликає велику спокусу трохи пожертвувати якістю заради значного зменшення необхідного об'єму для зберігання даних. Саме тому досить часто при послідовній передачі графічних файлів, накопичується зниження якості, що може призводити до втрати дрібних, але у деяких випадках дуже важливих деталей. Існує також і людський фактор не дуже професійного володіння електронними пристроями для зйомки графічних матеріалів, що може привести до матеріалів низької якості ще на першому етапі їх створення. Не варто забувати й про наявність архівної інформації низької якості через недостатньо розвинуту техніку для її створення. Аналізуючи усі ці фактори, виникає природне питання: чи можливо підвищити якість початково поганого зображення та, відповідно, відео?

В останні роки ця задача активно досліджувалась, й були створені вражаючі системи, засновані на технологіях глибинних згорткових нейронних мереж, здатні покращувати роздільну здатність зображень у 4-8 разів. Але основним обмеженням цих систем є необхідність опрацьовувати

усе зображення повністю, що по-перше, потребує використання значних часових ресурсів, а по-друге, у результаті формується гарне зображення при поверхневому перегляді, але якісь важливі дрібні деталі можуть так і не відновитися. Саме завдяки цьому виникає можливість створення більш ефективних систем для обробки зображень конкретної предметної галузі з початково відомими важливими дрібними деталями, якість яких має бути підвищена в першу чергу.

Однією з таких предметних галузей є відео, отримані з трансляцій спортивних змагань. Доволі у багатьох видів спорту використовуються дрібні предмети для гри, які при великій швидкості руху та низькій якості відео може бути доволі важко відстежувати. До таких видів спорту можна віднести, наприклад, бадмінтон, хокей, великий та настільний теніс. У кожному з них основними предметами гри є дрібні предмети такі як волани, шайби, м’ячі, швидкість яких може досягати близько 500 кілометрів на годину [1].

У цій роботі було вирішено зосередитися саме на покращенні якості тенісних відео з декількох причин:

– проблема поганого бачення м’яча є найбільш критичною саме у великому тенісі: цьому сприяють і співвідношення розмірів м’яча до розміру повного поля гри (тенісного корту), і велика швидкість польоту;

– зйомка відео тенісних матчів використовується за допомогою статичної камери, яка охоплює одразу весь корт, що дає можливість не тільки відстежувати місцезнаходження м’яча на кожному окремому кадрі, а й відстежувати одразу всю траєкторію польоту та відображати її на відео-результаті.

Метою даної роботи є розробка системи, основними двома компонентами якої є глибинна нейронна мережа для визначення місцезнаходження м’яча на кожному окремому кадрі відео, а також спеціальний алгоритм для синхронізації отриманих результатів нейронної мережі у безперервну траєкторію польоту м’яча. Будуть проаналізовані

різні можливі варіанти архітектури глибинної нейронної мережі, а також їх комбінації з другою частиною системи. Для навчання нейронної мережі, а також подальшого тестування ефективності системи, будуть зібрані репрезентативні датасети, що охоплюють як різноманітні покриття тенісних кортів, так і усі можливі ситуації місцезнаходження м’яча. Створена нейронна мережа дозволить не тільки покращувати якість відео, а й дасть можливість її незалежного використання, наприклад для автоматичного отримання тієї чи іншої статистики про виконання ударів тенісистів. А спеціальний алгоритм для синхронізації отриманих результатів нейронної мережі у безперервну траєкторію польоту дозволить ще сильніше покращити точність передбачення місцезнаходження м’яча нейронною мережею завдяки використанню одразу усієї інформації з відео, а не тільки окремих кадрів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Актуальність роботи

На сьогоднішній день теніс є дуже популярним видом спорту. Лише за останній рік на професійному рівні було проведено 386 чоловічих та 373 жіночих змагання, а загальна кількість проведених матчів досягає 34000 [2]. Оскільки відеозаписи усіх цих професійних матчів зберігаються, то при середній тривалості матчу півтори години, загальну кількість річного відеоматеріалу можна оцінити як 51000 годин, а необхідні об'єми пам'яті для їх зберігання як 10^{13} байт.

В той же час кількість присутніх вболівальників лише на одному матчі може досягати 40000, а загальна кількість прихильників цього виду спорту оцінюється в мільярд людей [3]. Загалом можна сказати, що теніс впевнено входить до 10 найбільш популярних видів спорту, а по оцінкам 2022 року так взагалі займає 5 місце з мільярдом фанатів (див. рис. 1.1).

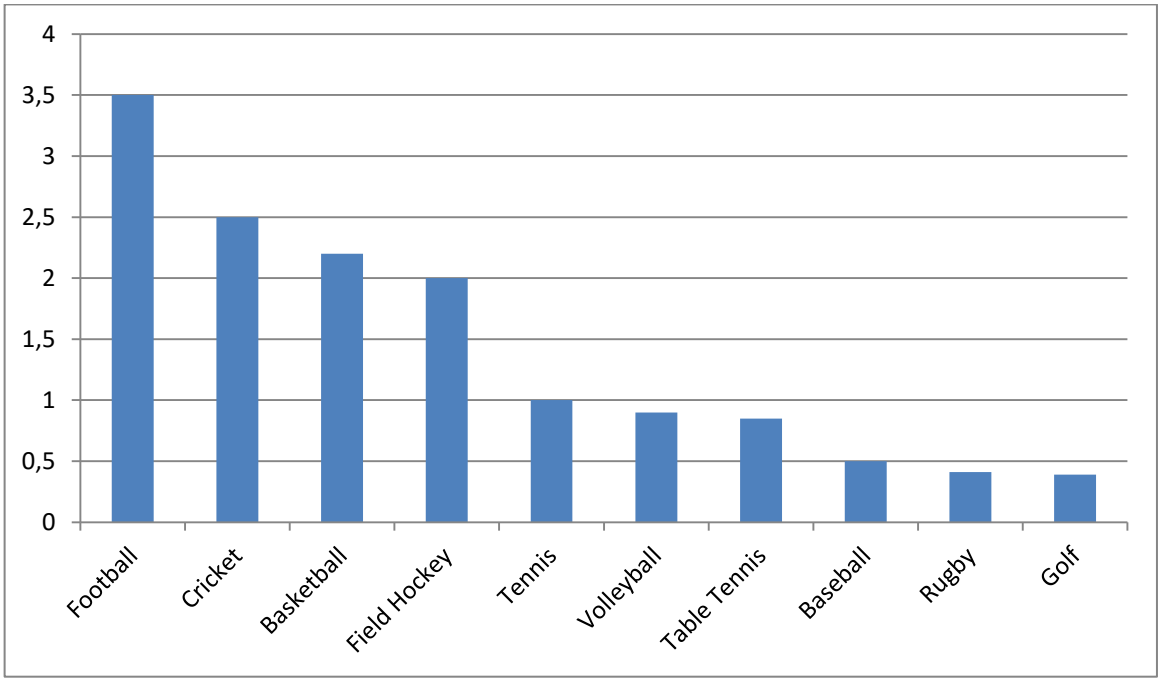

Рисунок 1.1 – Кількість вболівальників топових видів спорту, млрд

Через надзвичайно широку географію проведення турнірів, тільки найбільш запеклі вболівальники мають можливість постійно подорожувати та дивитись матчі на реальних стадіонах, але більшість людей все ж таки спостерігають за матчами за допомогою відео-трансляцій, чи оглядів найбільш цікавіших моментів після завершення самих поєдинків. Більш того, в останні роки обмеження, пов’язані з COVID 19, ще сильніше сприяли переходу вболівальників до онлайн режиму.

Саме тому якість відео-трансляцій надзвичайно важлива на сьогоднішній день для можливості легко та однозначно відстежувати траєкторію польоту м’яча. Більш того, автоматичне визначення траєкторії польоту є першим кроком у задачі побудови різноманітної статистики, а також автоматичного коментування матчів, а тому вирішення цієї задачі дозволить у майбутньому частково або повністю замінити професійних тенісних коментаторів.

1.2 Огляд структури задач обробки зображень

Загалом обробка зображень є величезним спектром задач комп’ютерного зору. Їх можна поділити за типом вихідного результату:

– задачі класифікації – зазвичай вихідним результатом у цих задачах є один клас, який відповідає за те, що знаходиться на зображенні, або ж одразу розподіл ймовірностей на декілька класів у випадку, коли задача передбачає можливість появи на зображенні одразу декількох класів;

– задачі сегментації – в якомусь сенсі їх також можна вважати задачами класифікації, але на відміну від попередніх, при сегментації встановлюється клас кожного пікселя початкового зображення, а тому вихідний результат до них має таку ж просторову розмірність як і вхідний; практичними прикладами таких задач можуть бути сегментація клітин у медичних зображеннях, виділення об’єктів на супутникових знімках,

сегментація зображень з відеокамер безпілотних автомобілів, та багато інших;

– задача детекції – розширена версія задачі класифікації, у якій окрім самих класів, що знаходяться на зображенні, необхідно також вказати їх місцезнаходження, зазвичай за допомогою прямокутної чи кругової обмежувальної коробки (bounding box); найбільш відомими практичними прикладами є детекція обличь у соціальних мережах та автоматична анотація зображень;

– задача трекінгу – розширена версія задачі детекції, у якій окрім визначення місцезнаходження об’єктів необхідно також встановити відповідність між знайденими об’єктами на різних кадрах; особливістю цієї задачі є і те, що вхідними даними до неї є одразу послідовність зображень, яка може бути використана для покращення результату детекції на окремих зображеннях;

– задача генерації – вихідним результатом до цих задач є повноцінне зображення, у той час як входом може бути, наприклад, зображення меншої роздільної здатності, чи карта сегментації з попередньої задачі;

– існують же такі задачі, які стоять на межі комп’ютерного зору та інших областей, наприклад задача створення опису зображень (image captioning), у якій вихідним результатом є речення людської мовою.

У нашому випадку для виявлення траєкторії польоту м’яча необхідно розв’язати задачу трекінгу. Але оскільки для гри в теніс використовується лише один м’яч, то задача спрощується та зводиться до задачі детекції на кожному окремому кадрі відео.

1.3 Огляд методів вирішення задачі трекінгу

Одним з найпростіших підходів до розв’язання задачі трекінгу є використання background subtraction [4]. Основною ідеєю цього підходу є можливість визначення статичного фону на основі декількох перших

кадрів відео, отриманих зі статичної камери. Після цього можна визначати наявні на зображенні об’єкти шляхом аналізу різниці самого зображення та статичного фону: у місцях, де знаходяться рухомі об’єкти, різниця буде суттєвою, в той час як у більшості частин поточне зображення буде збігатися зі статичним фоном, а тому різниця буде майже нульовою.

Більш просунуті підходи беруть за основу різноманітні методи розв’язання задачі детекції окремих кадрів відео. Одним з найпоширеніших з них є YOLO, запропонована у 2016 році [5]. Цій метод моделює детекцію як задачу регресії. А саме, на першому кроці вхідне зображення поділяється на сітку розміру $S \times S$. Кожна комірка цієї сітки відповідає за детекцію об’єкта, якщо його центр знаходиться усередині відповідної комірки. Далі зображення проходить через згорткову нейронну мережу та на виході отримується тензор розміру $S \times S \times (B * 5 + C)$: для кожної комірки сітки передбачується B обмежувальних коробок, кожна з яких задається п’ятьма числами (координатами та розміром коробок та впевненістю наскільки ця коробка є правильною), а також розподілом ймовірностей на тип одного з C класів, що знаходиться у цій комірці. Заключним етапом детекції є відбір з усіх $S \times S \times B$ коробок найбільш ймовірних таким чином, щоб для кожного об’єкта була обрана лише одна обмежувальна коробка.

1.4 Аналіз сучасного стану згорткових нейронних мереж

На сьогоднішній день існує багато різних методів для вирішення задачі детекції, але найбільшу ефективність показують саме методи, що використовують усередині згороткові нейронні мережі (CNN, Convolutional Neural Networks).

В даній роботі необхідно навчитися розв’язувати задачу обробки саме зображень, але «згортка» – універсальна операція, тому CNN можна застосувати для будь-якого сигналу, наприклад даних з датчиків, чи

аудіосигналу, а не тільки картинки. Завдання аналізу зображення полягає в розумінні того, що саме на ньому знаходиться, наприклад якась тварина, або будинок. Людина легко справляється зі цією задачею і розпізнає те, що бачить перед собою, за великою кількістю дрібних ознак. Згорткова нейронна мережа за рахунок застосування спеціальної операції – власне згортки, дозволяє водночас зменшити кількість інформації, що зберігається в пам'яті мережі, за рахунок чого краще справляється з картинками більш високої роздільної здатності, і виділити опорні ознаки зображення, такі як ребра, контури або грані. На наступному рівні обробки з цих ребер і граней можна розпізнати повторювані фрагменти текстур, які далі можуть скластися в фрагменти зображення [6].

Згорткові нейронні мережі застосовуються досить широко і в різних областях. Першим і, по суті, найбільш тривіальним завданням, яке навчилися вирішувати за допомогою нейронних мереж, стала класифікація зображень. Класифікація за допомогою CNN активно застосовуються в медицині, наприклад можна навчити нейронну мережу класифікації хвороб або симптомів. У системах «розумного будинку», а також системах відеоспостереження CNN використовуються для відстеження пересування об'єктів.

Згорткові (конволюційні) нейронні мережі дуже схожі на найпростіші звичайні повнозв'язні нейронні мережі. Вони складаються з шарів нейронів, у яких кожен нейрон відповідного шару зв'язаний з кожним нейроном попереднього шару певним ваговим коефіцієнтом, значення якого підлягає навчанню. Таким чином кожен нейрон отримує значення від своїх входів, після чого виконується лінійне перетворення і слідує нелінійна функція активації. Вся мережа все ще виражає єдину диференційовану функцію оцінок: від пікселів зображення на вході до векторного представлення об'єкту на виході.

Основним недоліком повнозв'язних нейронних мереж є надзвичайна кількість зв'язків між нейронами. Дійсно, якщо розміри двох послідовних

шарів такої мережі дорівнюють n та m відповідно, то кількість зв'язків між лише цими двома шарами буде дорівнювати $n \times m$, що робить розмір нейронної мережі дуже великим навіть при відносно невеликих значеннях n, m.

На відміну від повнозв'язних нейронних мереж, архітектури згорткових мереж припускають, що вхід в мережу – це зображення, що дозволяє задати певні особливості в архітектурі. А саме, у кожному шарі згорткової мережі зберігається просторове відношення нейронів. Кожен згортковий шар має три виміри: ширину, висоту та глибину зображення. Перший шар мережі приймає вхідне зображення, зазвичай з вимірами $h \times w \times 3$, що позначають матрицю пікселів кожного з каналів зображення, а нейрони з наступних шарів зв'язуються тільки з локальною областю входу, що визначається архітектурою мережі та задається розміром фільтру. Прикладом типової архітектури згорткової нейронної мережі є архітектура VGG-net (див. рис. 1.2).

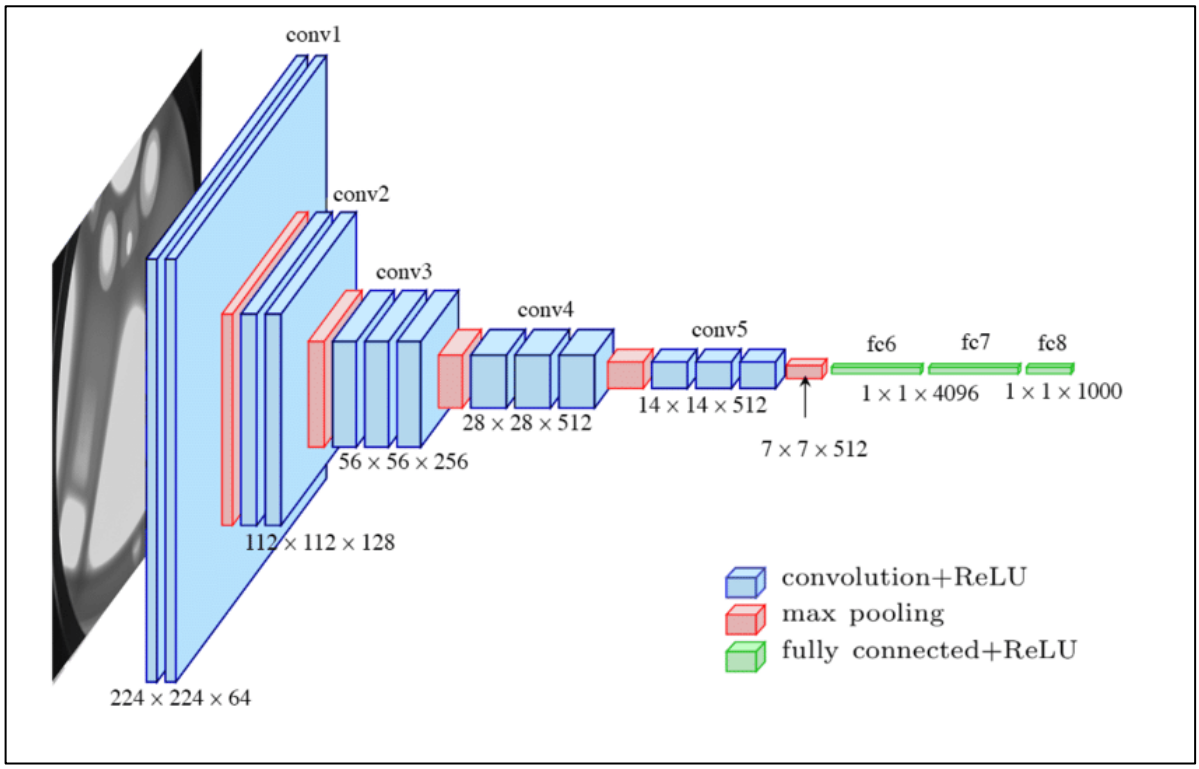

Рисунок 1.2 – Архітектура згорткової нейронної мережі VGG-net

У цій архітектурі спочатку йдуть певна кількість згорткових шарів, кожен з яких зменшує розмір зображення та збільшує кількість фільтрів. Також у кінці кожного шару виконується нелінійна функція активації ReLU, яку можна задати наступним чином (див. формулу 1.1):

$$f(x) = \max(x, 0). \tag{1.1}$$

Між деякими згортковими шарами використовуються max-pooling шари. У них в кожній підматриці розміру 2×2 залишається тільки нейрон з максимальним значенням. Такі шари виконують три основні функції:

– зменшення розміру зображення для того, щоб наступні згорткові шари працювали з більшою областю початкового зображення;

– збільшення інваріантності виходу мережі по відношенню до малого переносу входу;

– прискорення роботи наступних шарів, оскільки кількість нейронів для подальшої роботи зменшується в чотири рази.

В залежності від задачі наприкінці просторова інформація може все ж таки перетворюватися у звичайну непросторову, та повнозв’язні шари дозволяють отримати на виході векторне представлення вхідного зображення. Але у задачах, де вихідним результатом є інше зображення, зокрема задачах сегментації, просторова інформація має бути збережена та відновлена для початкових розмірів. Для цього використовуються up-pooling шари, які розповсюджують значення кожної окремої комірки на підматрицю розміру 2×2, тим самим збільшуючи кількість нейронів у 4 рази.

1.5 Аналіз основних підходів навчання нейронних мереж

Сама по собі архітектура будь-якої нейронної мережі нічого не варта, оскільки при заповненні параметрів моделі випадковими значеннями, вона

буде видавати абсолютно непридатний результат. Тому як тільки була створена архітектура мережі, настає етап її навчання. Щоб почати цей процес, початкові ваги зазвичай вибираються випадковим чином, після чого відбувається етап навчання. Існує три основних підходи навчання нейронних мереж:

– з вчителем;

– без вчителя;

– з підкріпленням.

Навчання з вчителем передбачає окрім вхідних тренувальних даних відповідних правильних вихідних. Найбільш відомим прикладом навчання з вчителем є задача класифікації, де у якості вхідних даних можуть бути, наприклад, зображення з кішками та собаками з попередньо визначеними класами для кожного зображення. Навчання без вчителя не передбачає наявність правильних вихідних даних. Найбільш відомим прикладом навчання без вчителя є задача кластерування. Повертаючись до прикладу з зображенням кішок та собак, необхідно розподілити усі зображення на дві папки, в одній з яких будуть знаходитись зображення з кішками, а в іншій з собаками. Базою для навчання з підкріпленням є прийняття відповідних дій для максимізації винагороди в конкретній ситуації.

Для вирішення задачі детекції буде використовуватись навчання з вчителем. В такому разі у якості тренувальних даних будуть наявні окремі кадри з відео тенісних матчів з відповідною інформацією про місцезнаходження м’яча на них.

Нейронна мережа складається з нейронів, пов'язаних один з одним певним зв’язком. Кожний зв’язок нейронної мережі має певну вагу, що позначає важливість цього зв’язку при множенні на вхідне значення [7]. Саме ці ваги усіх зв’язків і підлягають навчанню. Окрім безпосередніх зв’язків, кожен нейрон також має функцію активації, яка визначає вихід нейрона. Функція активації використовується для введення нелінійності в можливості моделювання мережі. Існує величезна кількість різноманітних

функцій активації, починаючи від найбільш простих, таких як ReLU, закінчуючи складними параметризованими функціями, параметри яких можуть також підлягати навчанню [8].

Процес навчання нейронної мережі, тобто визначення значень вагових коефіцієнтів, полягає в оптимізації функції на наборі тренувальних даних. Для цього використовуються градієнтні методи оптимізації, які складаються з двох основних кроків (див. рис. 1.3).

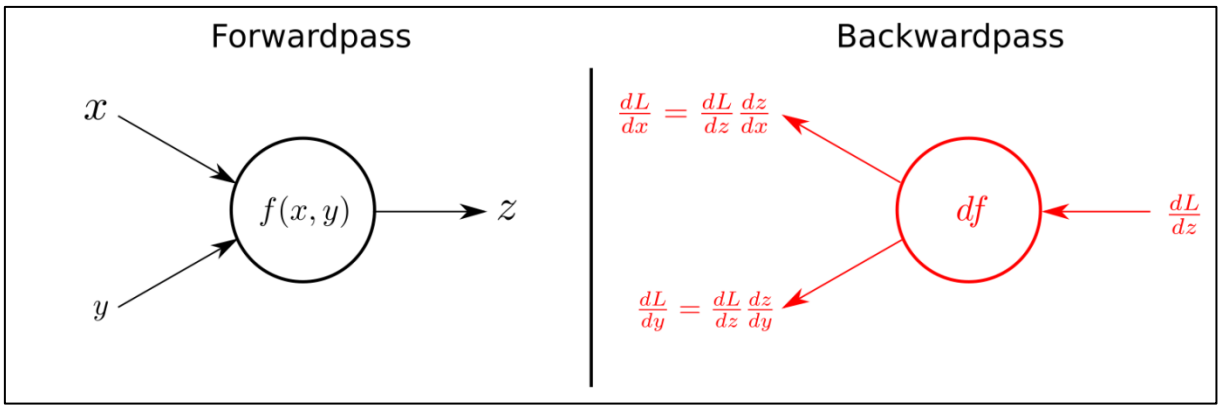

Рисунок 1.3 – Два кроки градієнтного спуску

А саме, на першому кроці відбувається пряме проходження вхідних даних по нейронній мережі, тобто визначення значення функції для усіх тренувальних вхідних даних. Після розрахунку функції втрат, виконується так зване зворотне поширення помилки. Починаючи з вихідного шару, ця інформація про втрати поширюється на всі нейрони в прихованому шарі, які вносять безпосередній внесок у вихід. Проте, нейрони прихованого шару отримують тільки частину загального сигналу про втрату, ґрунтуючись на відносному внеску, який кожен нейрон дав в вихідний результат. Цей процес повторюється шар за шаром до тих пір, поки всі нейрони в мережі не отримають сигнал про втрати, який описує їх відносний внесок в загальну втрату. Математично це виконується за допомогою підрахунку часткових похідних вихідної функції по всім

схованим параметрам моделі. Після цього підрахунку, усі коефіцієнти змінюються у сторону, протилежну градієнту, тим самим мінімізуючи значення функції, що оптимізуються. Цей процес повторюється певну кількість ітерацій, поки значення цільової функції продовжує зменшуватися.

1.6 Постановка задачі

На основі проведеного аналізу предметної галузі, можна сформулювати задачі, які необхідно вирішити в ході роботи. Результатом роботи має стати система, яка буде автоматично покращувати якість відео тенісних матчів за рахунок відстеження та подальшого перемальовування основної складової цих відео, а саме траєкторії польоту м’яча. Для цього необхідно:

– розглянути існуючі аналоги такої системи;

– розробити архітектуру глибинної нейронної мережі для задачі детекції місцезнаходження м’яча;

– розглянути існуючі набори даних та підготувати їх для використання шляхом розширення за допомогою аугментації даних та поділення на тренувальну, валідаційну та тестову частини;

– навчити створену нейронну мережу на заздалегідь підготованому наборі даних;

– розробити спеціальний алгоритм для об’єднання окремих результатів нейронної мережі у безперервну траєкторію польоту;

– обрати об’єктивні метрики оцінки якості системи, та провести оцінювання якості та ефективності розробленої системи на заздалегідь підготованому наборі даних.

2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1 Метрики оцінювання якості

Оскільки у даній задачі розглядається детекція конкретного об’єкта, а саме м’яча, то на відміну від загального випадку, результатом є тільки його місцезнаходження (x, y) без форми об’єкта. Для оцінки якості визначення місцезнаходження використовуються різні метрики, але всі вони базуються на значенні похибки, яку можна визначити за допомогою формули 2.1:

$$\varepsilon_k = \sqrt{\left(x_{pred}{}_k - x_{real}{}_k\right)^2 + \left(y_{pred}{}_k - y_{real}{}_k\right)^2}, \tag{2.1}$$

де k позначає номер кадру відео, а $\left(x_{pred_k}, y_{pred_k}\right), (x_{real_k}, y_{real_k})$ – координати передбачуваного та реального місцезнаходження м’яча на цьому кадрі відповідно.

Найпростіші метрики аналізують різноманітні характеристики розподілу ймовірностей на значеннях ε_k, зокрема математичне сподівання, медіану, квантиль. Але усі ці статистики досить погано передають реальну якість передбачення у випадку наявності одиничних кадрів з великим значенням похибки. Наприклад, математичне сподівання, яке у дискретному випадку являє собою середнє арифметичне, при 99% значень $\varepsilon_k = 0$ та 1% значень $\varepsilon_k = 1000$, буде дорівнювати 10, що є досить суттєвим значенням похибки, хоч і більшість передбачень є абсолютно точними.

Для подолання цих недоліків вводиться значення максимально допустимої похибки ε', що дає можливість знайти кількість правильно («true positive») та хибно («false positive») визначених положень (див. формулу 2.2). Також можна знайти кількість

правильно («true negative») та хибно («false negative») визначених кадрів з відсутністю м’яча.

$$\begin{cases} TP = \sum_{k \in SP} [\varepsilon_k \leq \varepsilon'] \\ FP = \sum_{k \in SP} [\varepsilon_k > \varepsilon'] \end{cases}, \tag{2.2}$$

де SP – множина усіх кадрів з наявністю м’яча на них.

В свою чергу знайдені значення TP, FP, TN, FN дозволяють використовувати стандартні метрики влучності, повноти та їх комбінації – гармонічного середнього $F_1\, score$ [9]. Саме ці метрики, які можна визначити формулою 2.3, у сукупності й дають найбільш об’єктивну оцінку показника якості методів визначення місцезнаходження.

$$\begin{cases} Precision = \dfrac{TP}{TP + FP} \\ Recall = \dfrac{TP}{TP + FN} \\ F = 2 \cdot \dfrac{Precision \cdot Recall}{Precision + Recall} \end{cases}. \tag{2.3}$$

$Precision$ (влучність моделі) показує долю правильно визначених позицій серед усіх передбачень, в той час як $Recall$ (повнота моделі) показує відносну кількість правильно визначених позицій серед загальної кількості кадрів з наявним м’ячем. Кожна з цих метрик приймає значення від 0 до 1, де більш близькі значення до 1 позначають більш якісну модель.

Окрім якості існує ще один дуже важливий показник, а саме швидкість роботи системи, яку можна визначити як кількість кадрів, що оброблюються за одну секунду (див. формулу 2.4).

$$FPS = \frac{Frames}{ProcessingTime}, \tag{2.4}$$

де $Frames$ – загальна кількість кадрів в відео;

$ProcessingTime$ – загальний час обробки відео, с.

2.2 Аналіз підходів до вирішення задачі детекції м’яча

2.2.1 Тривіальні та евристичні методи

Найбільш відомим класичним методом вважається евристичний підхід, запропонований у публікації у 2015 році [10]. Загалом цей метод складається з декількох послідовних етапів. На першому етапі визначається статичний фон відео за допомогою усереднення схожих частин усіх окремих кадрів. Далі виконується незалежна обробка кожного кадру відео, першим кроком якої є визначення бінарної маски зображення з метою визначення тих пікселів, у яких спостерігається наявність руху. Для цього знаходиться різниця поточного кадру з фоном та попереднім кадром, після чого отримані різниці перетворюються у бінарні значення завдяки відповідній фільтрації, а саме порівнянням із пороговим значенням t. Завершує перший крок операція бінарного «і» над двома знайденими відфільтрованими різницями, яка і формує остаточну бінарну маску зображення. Більш формально знаходження бінарної маски C можна записати за допомогою формули 2.5:

$$\begin{cases} A_k(x,y) = \begin{cases} 1, |Frame_k(x,y) - Background(x,y)| > t \\ 0, otherwise \end{cases} \\ B_k(x,y) = \begin{cases} 1, |Frame_k(x,y) - Frame_{k-1}(x,y)| > t \\ 0, otherwise \end{cases}, \\ C_k = A_k(x,y) \wedge B_k(x,y), 1 \le x \le w, 1 \le y \le h \end{cases} \tag{2.5}$$

де $Frame_k(x, y)$ – відповідний піксель k-го фрейму відео;

$Background(x, y)$ – відповідний піксель знайденого статичного фону;

t – порогове значення.

Після виконання першого кроку одиничні значення бінарної маски C позначають ті пікселі, які одночасно суттєво відрізняються від фону, а також від попереднього кадру відео. У публікації робиться припущення, що у більшості випадків завдяки великій швидкості м'яча і відмінності його кольору від фону, майже уся його область буде мати одиничні значення.

На другому кроці виконується пошук компонент зв'язності з одиничними значенням у масці C, кожна з якої являє собою кандидат на шукане місцезнаходження. Для знаходження остаточного кандидату спочатку виконуються фільтрація усіх знайдених об'єктів за розміром, після чого обирається кандидат з найбільш схожою формою та відношенням висоти до ширини на відповідні параметри м'яча.

Основною перевагою цього методу є простота реалізації та швидкість виконання. Разом з тим наявна й велика кількість недоліків, серед яких є необхідность встановлювати гіперпараметри евристичним шляхом для кожного окремого відео, оскільки в залежності від місцезнаходження камери, еталонні значення можуть змінюватись. Також для цього методу притаманно наявність відносно великої кількості помилок у другому етапі, оскільки порівняння кандидатів між собою відбувається за невеликим переліком простих ознак. Загалом автори відмічають $Precision = 90.48\%$, та $FPS = 60 \frac{\text{кадрів}}{\text{с}}$ на невеликому наборі даних розміром 1950 кадрів, що підтверджує велику швидкість обробки і відносно велику кількість неточних передбачень про місцезнаходження м'яча.

2.2.2 Нейромережевий метод

Більш сучасний альтернативний підхід був запропонований у 2019 році у публікації «A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications» [11]. Як і попередній евристичний метод, детекція відбувається на бінарній карті зображення, але основною відмінністю є саме спосіб побудови цієї карти. Замість детермінованого алгоритму, заснованому на знаходженні різниць поточного кадру відео з попереднім та фоном і подальшій «бінарізації» за допомогою порівнянь з певним пороговим значенням, використовується глибинна нейронна мережа, яка повністю автоматично генерує теплову карту. При цьому для кожного пікселя знаходиться ймовірність того, що він відноситься до шуканого м'яча. У запропонованій архітектурі ця ймовірність дискретизується у значення від 0 до 255. В подальшому отримана теплова карта може бути перетворена у бінарну за допомогою аналогічної операції порівняння з пороговим значенням.

В ідеальному випадку ймовірність відношення до шуканого м'яча повинна мати нормальний розподіл. А саме, якщо центр справжнього м'яча знаходиться у позиції (x_0, y_0), то ідеальну теплову карту для цього зображення можна описати формулою 2.6:

$$G(x,y) = \left\lfloor \left(\frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}}\right)(2\pi\sigma^2 \cdot 255)\right\rfloor, \tag{2.6}$$

де x, y $(1 \le x \le w, 1 \le y \le h)$ позначають координати на карті.

У цій формулі перший множник відповідає нормальному розподілу стосовно відстані то центру, а другий – перетворює значення у дискретизовану ймовірність від 0 до 255. Приклад ідеальної теплової карти можна побачити на рисунку 2.1.

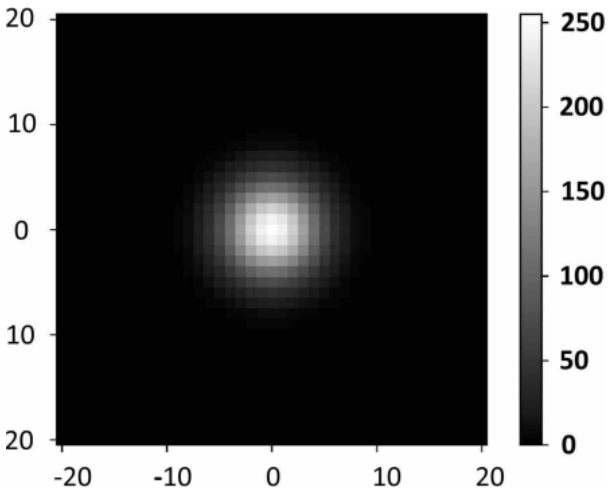

Рисунок 2.1 – Ідеальна теплова карта

Ця теплова карта відповідає координатам центру $(x_0 = 0, y_0 = 0)$, а також $\sigma^2 = 10$. Саме таке значення σ використовувалось при генерації правильних вихідних карт для тренувальних зображень, оскільки воно відповідає середньому радіуса м'яча в 5 пікселів.

Для генерації теплової карти використовується типова архітектура сегментаційної нейронної мережі, що складається з двох частин: шляху звуження і розширення (див. рис. 2.2).

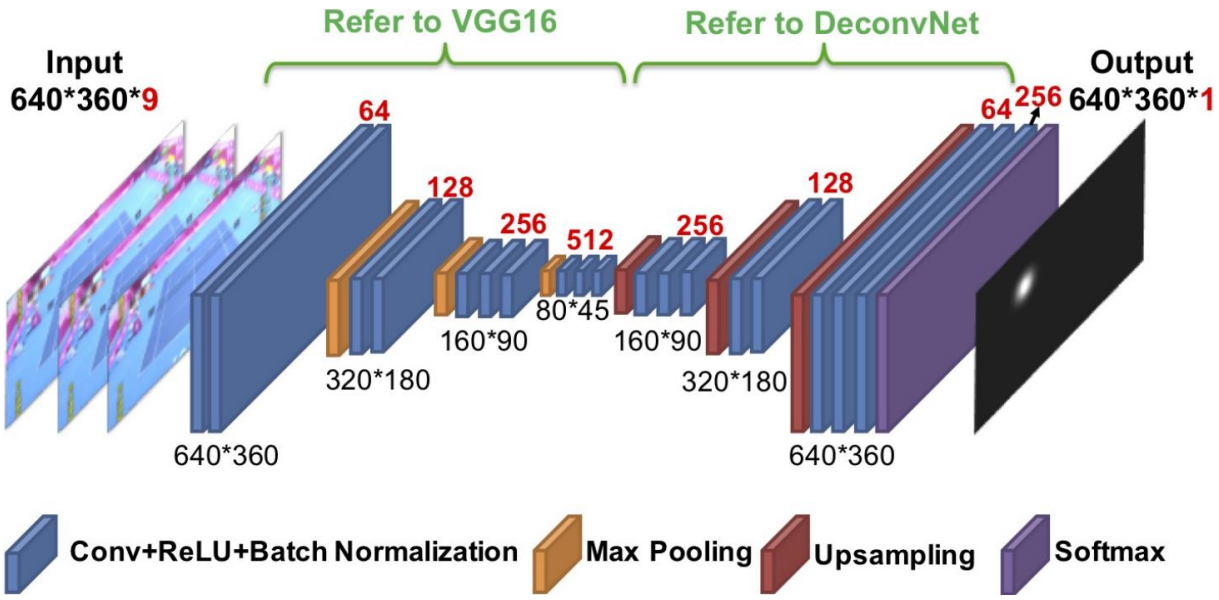

Рисунок 2.2 – Повна архітектура нейронної мережі

Шлях звуження являє собою перші 13 шарів відомої архітектури VGG-16 [12] для задачі класифікації, основними складовими якої є згорткові та пулінгові шари. Шлях розширення використовує 11 шарів архітектури DeconvNet [13], яка активно використовує up-pooling шари для відновлення просторової розмірності. В той же час архітектура має певні особливості. По-перше, вхідний шар приймає не одне зображення, а одразу три послідовних кадри, що дає можливість використовувати більшу кількість інформації, зокрема тенденцію траєкторії польоту, яка інколи дозволяє визначити точне місцезнаходження навіть у випадках, коли на останньому кадрі м'яча не видно. По-друге, передостанній вихідний шар має 256 каналів для кожного пікселя, які за допомогою softmax (див. формулу 2.7) перетворюються у розподіл ймовірностей кожного з пікселів, після чого шукане дискретизоване значення визначається як найбільш ймовірне (див. формулу 2.8):

$$P(x,y,k) = \frac{e^{L(x,y,k)}}{\sum_{l=0}^{255} e^{L(x,y,l)}}, \tag{2.7}$$

$$h(x,y) = \arg\max_{k} P(x,y,k), \tag{2.8}$$

де $L(x,y,k)$ позначає значення k-го каналу передостаннього шару у позиції (x,y).

Після отримання теплової карти відбувається її бінарізація шляхом порівняння з пороговим значенням. Після цього на бінарній карті знаходяться усі кругові об'єкти схожим способом попереднього методу. Але на відміну від нього, у випадку наявності декількох об'єктів, вони не порівнюються між собою, що призводить до відсутнього результату детекції. Це стає можливим завдяки тому, що запропонована глибинна нейронна мережа значно краще генерує бінарну карту, а тому у більшості випадків на ній наявний лише один круговий об'єкт.

Автори публікації відзначають, що запропонована модель на порядок випереджає існуючі аналоги, що робить подібні глибинні мережі цікавим предметом для дослідження у цій задачі. А саме, на тестовому наборі даних з 6250 кадрів показник метрик досягають $Precision = 95.3\%$, $Recall = 75.7\%$, $F = 84.3\%$, що на $10 - 20\%$ перевищує відповідні показники тривіальних алгоритмів. Також архітектура є досить гнучкою та зокрема дозволяє змінювати кількість вхідних кадрів. Експерименти показали, що використання одразу трьох послідовних кадрів значно підвищує якість моделі, що підтверджує здатність нейронної мережі враховувати особливості траєкторії польоту. В той же час можна відзначити один дуже суттєвий недолік моделі, який полягає в низькій швидкості обробки відео, а саме усього лиш $FPS = 2\frac{\text{кадри}}{\text{с}}$ на сучасних графічних процесорах. Така маленька швидкість хоч і дозволяє оброблювати окремі відео постфактум, але унеможливлює використання моделі для обробки відео в режимі реального часу.

2.3 Огляд підходу до синхронізації окремих результатів у траєкторію

Обидва описані в попередньому розділі підходи до вирішення задачі детекції м’яча на передостанньому кроці мають набір кандидатів на його місцезнаходження. І саме останній крок вибору правильного кандидату є слабким місцем обох методів та призводить до зниження показників $precision, recall$. Основним мотивом для використання додаткового алгоритму для синхронізації окремих результатів детекції у неперервну траєкторію є той факт, що місцезнаходження м’яча на кожному окремому кадрі відео не є випадковим, а навпаки являє собою дискретні точки послідовної траєкторії польоту, яка може бути описана певними фізичними законами. Саме цей факт дозволяє по-перше покращити відбір конкретних кандидатів з першої складової системи, а по-друге відновити саму траєкторію, що в подальшому може бути зображена на оновленому

результативному відео. Вперше доволі ефективний алгоритм генерації неперервної траєкторії був запропонований у публікації «A Novel Data Association Algorithm for Object Tracking in Clutter with Application to Tennis Video Analysis» [14].

Нехай перша складова системи для кожного кадру відео визначила множину кандидатів на місцезнаходження (див. формулу 2.9):

$$C_k = \{c_k^j\}_{j=1}^{m_k}, \tag{2.9}$$

де m_k – загальна кількість кандидатів у кадрі k;

c_k^j – j-й кандидат в відповідному кадрі, $\mathbb{R}^2$.

Відомо, що рух м’яча можна моделювати як рух об’єкту, кинутого під кутом до горизонту, який можна описати рівнянням руху з константним прискоренням a. Для знаходження цього рівняння достатньо знати місцезнаходження об’єкту у будь-які 3 дискретні моменти часу k_1, k_2, k_3 $(1 \le k_1 < k_2 < k_3)$, що належать одній траєкторії його польоту (див. формулу 2.10).

$$\begin{cases} a = 2\dfrac{(k_2 - k_1)(p_3 - p_2) - (k_3 - k_2)(p_2 - p_1)}{(k_2 - k_1)(k_3 - k_2)(k_3 - k_1)} \\ v_1 = \dfrac{p_2 - p_1}{(k_2 - k_1)} - \dfrac{(k_2 - k_1) \cdot a}{2} \\ p(k) = p_1 + (k - k_1) \cdot v_1 + \dfrac{(k - k_1)^2 a}{2} \end{cases}, \tag{2.10}$$

де $p_1, p_2, p_3 \in \mathbb{R}^2$ позначають місцезнаходження у моменти часу k_1, k_2, k_3 відповідно, a – константне прискорення, а v_1 – швидкість в перший момент часу.

Тоді задачею алгоритму є для кожного кадру знаходження правильного кандидату $c_k^{j_{k\_true}}$, а також розбиття усіх кадрів на

проміжки $[l_i; r_i]$, кожен з яких можна описати неперервною траєкторією $p_i(k)$.

Вирішення цієї задачі відбувається у два етапи: спочатку знаходяться кандидати на можливі проміжки $[l; r]$ з відповідними значеннями $c_k^{j_{k\_true}}$ $(l \le k \le r)$ та функцією траєкторії $p(k)$, після чого обирається підмножина цих проміжків, які загалом покривають усю довжину відео. Для знаходження можливих проміжків спочатку перебираються усі послідовні трійки кадрів з центром у момент часу i, а також відповідні кандидати на місцезнаходження $c_{i-1}^{j'}, c_i^j, c_{i+1}^{j''}$. При цьому розглядаються тільки ті трійки, в яких відстань між послідовними кандидатами не перевищує максимально можливої відстані R, яку можна подолати за один кадр (див. рис. 2.3).

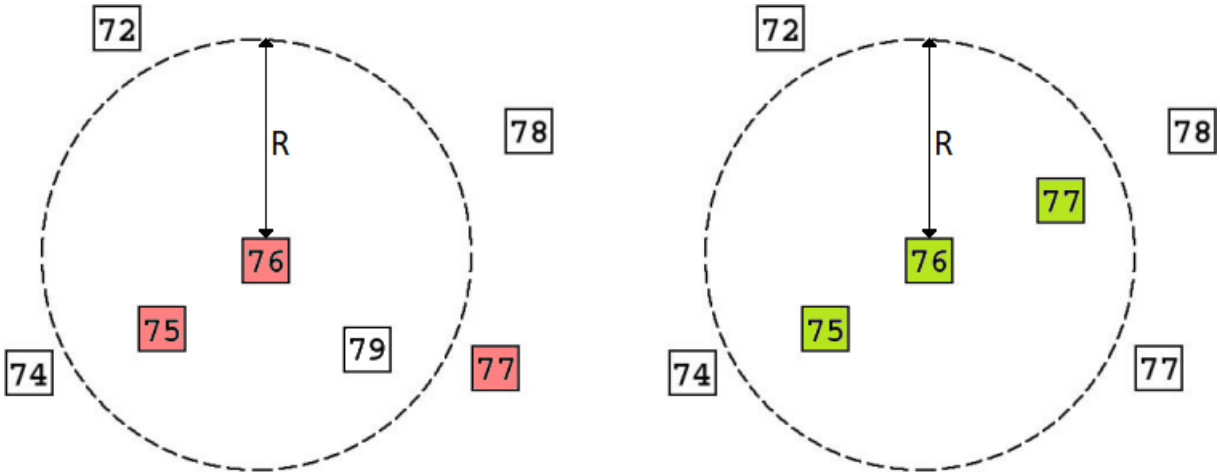

Рисунок 2.3 – Приклад відбору послідовних трійок

У цьому прикладі розглядаються послідовна трійка з центром $i = 76$, і тільки права трійка вважається правильною, оскільки в лівій відстань між кандидатами на 76 та 77 кадрах перевищує максимально можливу відстань R.

Кожна обрана трійка дає початкову наближену траєкторію $p(k)$. Але зазвичай таке наближення досить погано описує реальну траєкторію,

оскільки використовує дуже близькі моменти часи: чим далі від центрального моменту часу i, тим сильніше наближена траєкторії відхиляється від реальної. Це відбувається тому, що обрані три кандидати знайдені з певною похибкою, яка з більшим віддаленням проявляється все сильніше. В той же час при обчисленні рівняння траєкторії через місцезнаходження з більш віддаленими проміжками часу, їх похибка слабше відображається на правильності траєкторії. Саме тому після знаходження початкової наближеної траєкторії за трьома послідовними моментами часи відбувається її ітеративна оптимізація.

На кожній ітерації спочатку знаходиться підтримуюча множина S до $p(k)$, у яку входять усі кандидати c_k^j, що знаходяться на невеликій відстані від траєкторії $p(k)$. А саме, для кожного кадру $k \in [i-N; i+N]$ у вікні розміром N знаходиться найближчий кандидат до поточної траєкторії $p(k)$ та додається у множину S у випадку, коли відстань до нього не перевищує порогового значення d_{th}. Знайдена множина S дозволяє оновити рівняння траєкторії через 3 нові кандидати, що знаходяться далі один від одного (див. формулу 2.11):

$$\begin{cases} k_{min} = \min k \ \ \exists c_k^j \in S \\ k_{mid} = \arg\min\limits_k \left| |k_{min} - k| - |k_{max} - k| \right| \ \ \exists c_k^j \in S. \\ k_{max} = \max k \ \ \exists c_k^j \in S \end{cases} \quad (2.11)$$

Оновлення траєкторії $p(k)$ через найбільш віддаленні один від одного кандидати з кадрів $k_{min}, k_{mid}, k_{max}$ зазвичай дозволяє покращити її якість (див. рис. 2.4). На рисунку можна побачити приклад такої оптимізації, де траєкторія, визначена через послідовні кадри 75 – 77, оновлюється через кандидати з кадрів $k_{min} = 74, k_{mid} = 76, k_{max} = 78$ і більш точно наближує реальну траєкторію, що відмічена квадратами.

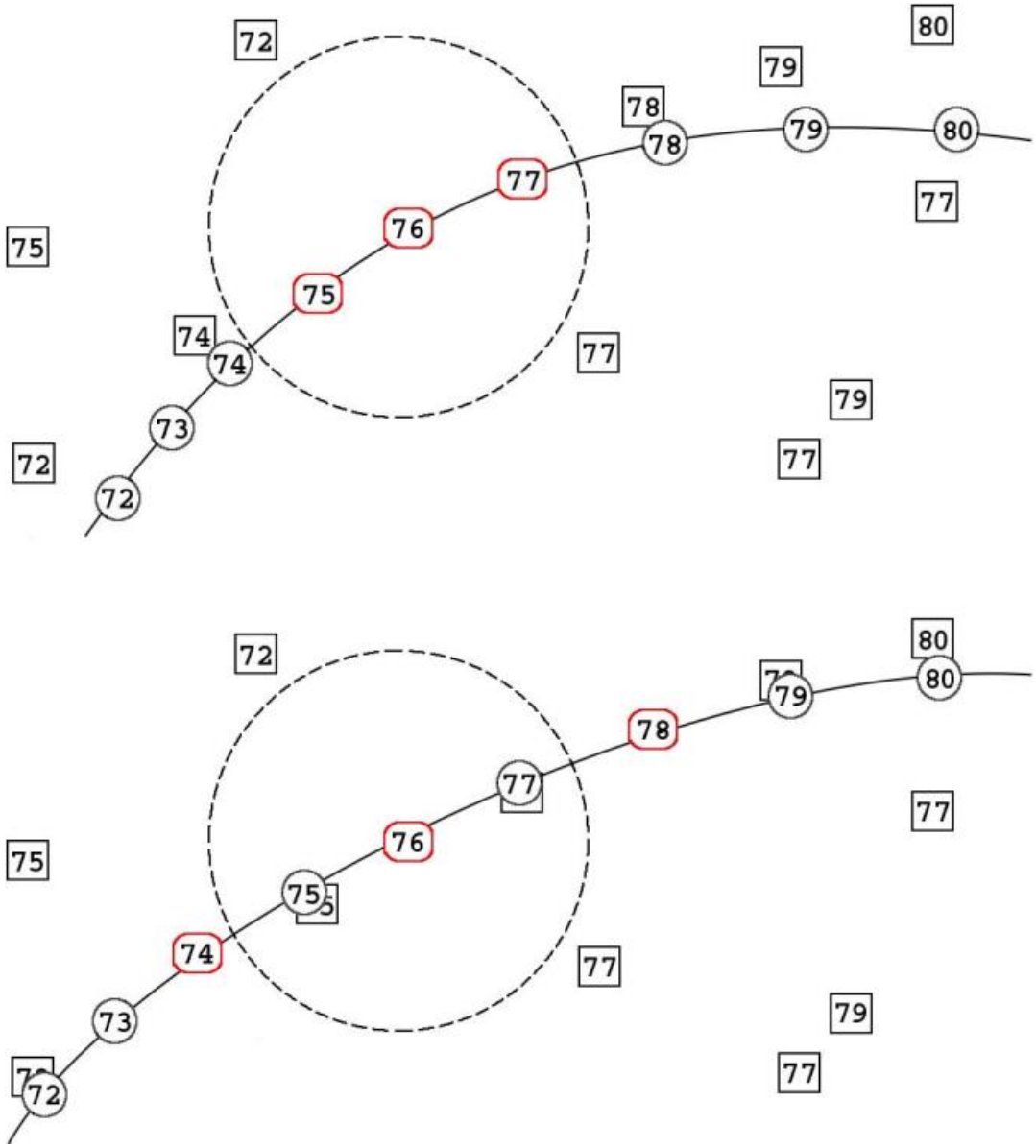

Рисунок 2.4 – Приклад однієї ітерації оптимізації траєкторії

Описана оптимізація початкової траєкторії $p(k)$ застосовується ітеративно, поки якість нової траєкторія є кращою. Для визначення якості траєкторії використовується метрика, яка підраховує сумарну відстань від дискретних позицій траєкторії до усіх наявних кандидатів на місцезнаходження у вікні (див. формулу 2.12).

$$\begin{cases} C(p) = \sum_{k=i-N}^{i+N} \sum_{j=1}^{m_k} \rho(p, c_k^j) \\ \rho\left(p, c_k^j\right) = \min(d^2\left(p(k), c_k^j\right), d_{th}^2) \end{cases}, \tag{2.12}$$

де $C(p)$ позначає якість траєкторії, менші значення якої є кращими, а d – Евклідову відстань між двома точками. Оптимізація завершується, коли якість оновленої траєкторії не є кращою, тобто $C(p') > C(p)$. Також для кожного центрального моменту часу i зберігається лише найкраща траєкторія разом з відповідною підтримуючою множиною, мінімальні та максимальні індекси кадрів якою дають інформацію про проміжок часу, на якому вона спостерігалася: $T_i = \{p_i(k), S_i, l_i = k_{min,i}, r_i = k_{max,i}\}$.

Кожна окрема знайдена траєкторія являє собою якийсь один неперервний проміжок польоту м’яча без взаємодії з іншими об’єктами такими як корт або гравець, а тому для генерації повної траєкторії польоту виникає необхідність у другому етапі алгоритму, який дозволяє обрати та об’єднати окремі T_i найбільш ефективним способом. Для цього розв’язується задача пошуку найкоротшого шляху у графі, де вершинами виступають усі T_i, а ваги орієнтованих ребер позначають степінь сумісності пар $\left(T_i, T_j\right)$ між собою. При цьому ребра існують лише між тими парами вершин $\left(T_i, T_j\right)$, для яких виконується нерівність 2.13:

$$l_i - r_j \leq k_{th}, \tag{2.13}$$

де k_{th} позначає максимальну кількість кадрів поспіль, у яких могло не бути правильних кандидатів. При цьому робиться припущення, що алгоритм детекції не міг допустити більш ніж k_{th} пропусків поспіль.

Основною складністю другого етапу є визначення ваг ребер. При цьому використовується трьохступенева функція відстані. А саме, для пари траєкторій T_i, T_j $(i < j)$ спочатку визначається перетинаються вони, чи ні,

шляхом порівняння їх відповідних проміжків часу $[l_i;\ r_i]$ та $[l_j;\ r_j]$. У позитивному випадку визначається, чи конфліктують їх підтримуючі множини, тобто чи існує момент часу $k \in [l_i; r_i]$, у який підтримуючі множини S_i та S_j мають різні підтримуючі об'єкти. У разі існування конфлікту вага вважається рівною нескінченності, що відповідає непроведенню ребра. В противному випадку вага вважається рівною нулю, оскільки траєкторії плавно переходять одна в іншу. У випадку відсутності перетину відповідних проміжків часу траєкторій, відстань визначається як мінімальна відстань у будь-який дискретний момент часу між ними (див. формулу 2.14).

$$D(T_i, T_j) = \min_{k=r_i \ldots l_j} d(p_i(k), p_j(k)). \tag{2.14}$$

Після побудови графу знаходиться найкоротший шлях між певною «гарною» стартовою траєкторією, яка своїм проміжком покриває перші кадри відео, до аналогічної фінішної вершини за допомогою алгоритму Дейкстри [15].

В експериментальній частині автори відмічають здатність алгоритму відбирати правильні кандидати навіть при наявності великої кількості, а саме 11.6 кандидатів на кожен кадр. Метрики $precision, recall$ при цьому не оцінюються, але наводиться графік розподілу похибки передбачень, з якого дуже добре видно, що основна частина передбачень має відхилення не більше 5 пікселів.

3 ОПИС СИСТЕМИ, ЩО ПРОПОНУЄТЬСЯ

3.1 Глибинна нейронна мережа для детекції м’яча

За основу першої складової системи була взята нейронна мережа [11], описана в попередньому розділі. При цьому були зроблені певні зміни в архітектурі, в першу чергу направлені на покращення швидкості моделі, оскільки саме цей показник був головним недоліком запропонованої моделі. Деякі з цих змін являють собою адаптацію підходу, запропонованого у публікації для трекінгу волану [16].

По-перше, було помічено, що більшість часу при проходженні через нейронну мережу займав саме останній згортковий шар перетворення 64 каналів розміру 640×360 на 256 каналів аналогічного розміру. Враховуючи розмір ядра згортки 3×3, загальна кількість операцій для обчислення сягає $64 \cdot 640 \cdot 360 \cdot 256 \cdot 3 \cdot 3 \approx 34 \cdot 10^9$, що є значно більшою кількістю в порівняння з усіма попередніми аналогічними операціями. А оскільки ці 256 каналів в подальшому все одно перетворювались на бінарне значення, було вирішено безпосередньо замінити кількість каналів в останньому згортковому шарі з 256 на 1 з подальшою сигмоїдальною функцією активації (див. формулу 3.1), яка також видає значення від 0 до 1 та може легко бути бінаризована.

$$S(x) = \frac{1}{1 + e^{-x}}. \tag{3.1}$$

По-друге, передбачення однієї теплової карти по одразу трьом вхідним кадрам було замінено на передбачення трьох відповідних теплових карт. Дійсно, с точки зору нейронної мережі немає різниці між передбаченням останньої теплової карти за двома попередніми кадрами, та першої за двома наступними. Саме тому виникає можливість змінити один

вихідний канал на три. Така оптимізація дозволяє збільшити швидкість майже у три рази, оскільки основна частина нейронної мережі залишається спільною для усіх трьох результативних теплових карт, а збільшення кількості каналів останнього шару у три рази майже не впливає на зменшення швидкості.

Ще однією зміною є використання ідеї з найбільш популярної архітектури для вирішення задачі сегментації U-net [17], а саме ідеї «skip connections». Ця ідея полягає в безпосередньому поєднанні двох частин звуження та розширення шляхом конкатенації відповідних ознакових карт, що дозволяє поєднати низько та високо-рівневі ознаки та покращити загальну якість моделі. Модифіковану архітектуру нейронної мережі можна побачити на рисунку нижче (див. рис. 3.1).

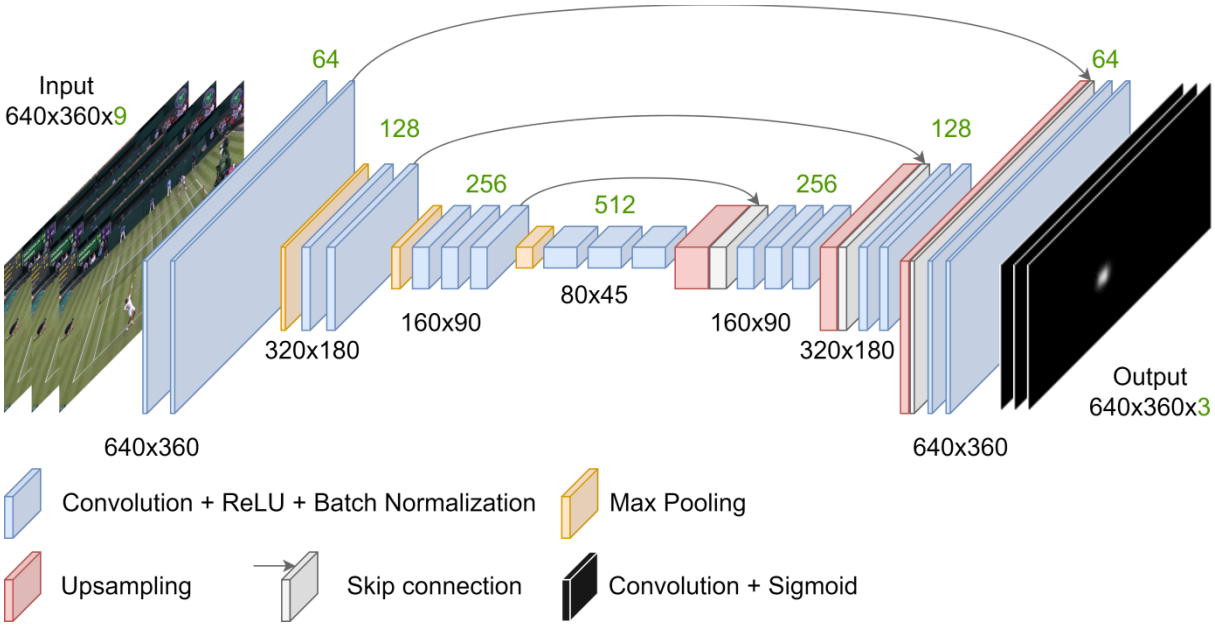

Рисунок 3.1 – Архітектура модифікованої нейронної мережі

В порівнянні з попередньою архітектурою кількість та параметри згорткових шарів залишаються незмінними, в той час як серед нових блоків можна помітити «сірі» блоки – конкатенацію високо-рівневих ознак з відповідними останніми для своєї роздільної здатності блоками низько-

рівневих, а також змінену кількість вихідних каналів з іншою функцією активації. Повний перелік усіх шарів з детальним описом їх параметрів наведено у таблиці 3.1.

Таблиця 3.1 – Перелік усіх шарів нейронної мережі

№	Назва шару	Деталі	Розмірність виходу
-	Input	3 послідовних кадри по 3 канали кожен	$640 \times 360 \times 9$
1	Conv1	$3 \times 3 \times 64$, ReLU, Batch Normalization	$640 \times 360 \times 64$
2	Conv2	$3 \times 3 \times 64$, ReLU, Batch Normalization	$640 \times 360 \times 64$
3	MaxPool1	$kernel\_size = 2 \times 2$	$320 \times 180 \times 64$
4	Conv3	$3 \times 3 \times 128$, ReLU, Batch Normalization	$320 \times 180 \times 128$
5	Conv4	$3 \times 3 \times 128$, ReLU, Batch Normalization	$320 \times 180 \times 128$
6	MaxPool2	$kernel\_size = 2 \times 2$	$160 \times 90 \times 128$
7	Conv5	$3 \times 3 \times 256$, ReLU, Batch Normalization	$160 \times 90 \times 256$
8	Conv6	$3 \times 3 \times 256$, ReLU, Batch Normalization	$160 \times 90 \times 256$
9	Conv7	$3 \times 3 \times 256$, ReLU, Batch Normalization	$160 \times 90 \times 256$
10	MaxPool3	$kernel\_size = 2 \times 2$	$80 \times 45 \times 256$
11	Conv8	$3 \times 3 \times 512$, ReLU, Batch Normalization	$80 \times 45 \times 512$
12	Conv9	$3 \times 3 \times 512$, ReLU, Batch Normalization	$80 \times 45 \times 512$
13	Conv10	$3 \times 3 \times 512$, ReLU, Batch Normalization	$80 \times 45 \times 512$
14	UpSample1	2×2, skip connection with Conv7	$160 \times 90 \times 768$
15	Conv11	$3 \times 3 \times 256$, ReLU, Batch Normalization	$160 \times 90 \times 256$
16	Conv12	$3 \times 3 \times 256$, ReLU, Batch Normalization	$160 \times 90 \times 256$
17	Conv13	$3 \times 3 \times 256$, ReLU, Batch Normalization	$160 \times 90 \times 256$
18	UpSample2	2×2, skip connection with Conv4	$320 \times 160 \times 384$
19	Conv14	$3 \times 3 \times 128$, ReLU, Batch Normalization	$320 \times 160 \times 128$
20	Conv15	$3 \times 3 \times 128$, ReLU, Batch Normalization	$320 \times 160 \times 128$
21	UpSample3	2×2, skip connection with Conv2	$640 \times 320 \times 192$
22	Conv16	$3 \times 3 \times 64$, ReLU, Batch Normalization	$640 \times 320 \times 64$
23	Conv17	$3 \times 3 \times 64$, ReLU, Batch Normalization	$640 \times 320 \times 64$
24	Conv18	$1 \times 1 \times 3$, Sigmoid	$640 \times 320 \times 3$

Загальна кількість шарів становить 24, серед яких 18 являють собою згорткові шари, що дає повне право називати цю нейронну мережу глибинною.

Для отримання шуканого місцезнаходження після проведення послідовних кадрів через нейронну мережу спочатку відбувається бінарізація отриманої теплової карти шляхом порівняння її елементів із пороговим значенням th. Після цього знаходяться усі одиничні компоненти зв'язності, центри яких передаються як кандидати на місцезнаходження до другої складової системи.

3.2 Алгоритм для комбінації результатів нейронної мережі

За основу алгоритму для комбінації результатів нейронної мережі у безперервну траєкторію польоту був взятий алгоритм, описаний у розділі 2.3. При цьому були зроблені певні зміни, націлені на покращення якості роботи.

По-перше, була змінена формула для пошуку k_{mid} під час ітерації оптимізації поточної траєкторії (див. формулу 3.2).

$$k_{mid} = \arg\min_{k} |k - i| \quad \exists c_k^j \in S, \tag{3.2}$$

де i – центральний момент часу;

S – підтримуюча множина знайденої траєкторії.

Така зміна призводить до більшої варіативності генерованих траєкторій, оскільки для різних i використовується різні k_{mid}. У сукупності з тим, що в подальшому з усієї множини згенерованих траєкторій обирається найбільш сумісна підмножина, більша варіативність додає гнучкості алгоритму.

По-друге, формула 2.12 для визначення якості траєкторії була вдосконалена шляхом підрахунку лише вибіркових мінімальних відстаней (див. формулу 3.3). Ця зміна робить функцію якості більш стабільною при наявності великої кількості кандидатів, оскільки

врахування до суми кожного з них робить більш якісну ту траєкторію, яка у середньому добре описує всіх кандидатів, в той час як насправді важливим є відстань саме до правильних місцезнаходжень.

$$\begin{cases} C(p) = \sum_{k=i-N}^{i+N} \min_{1 \le j \le m_k} \rho(p, c_k^j) \\ \rho\left(p, c_k^j\right) = \min(d^2\left(p(k), c_k^j\right), d_{th}^2) \end{cases} . \quad (3.3)$$

Адаптована функція $C(p)$ буде приймати нульове значення у випадку ідеальної траєкторії, в той час як попередня функція може ніколи не досягти ідеального значення через наявність помилкових кандидатів.

По-третє, був змінений спосіб моделювання траєкторії $p(k)$. У формулі 2.10 використовувалось стандартне рівняння руху об'єкта з константним прискоренням, яке може бути переписано у формі квадратичної залежності від часу. Для знаходження параметрів цього рівняння використовувалось лише 3 дискретні позиції у різні моменти часу. При цьому знайдене рівняння траєкторії $p(k)$ гарантувало ідеальне проходження через 3 обрані точки, але через можливу наявність похибки трьох обраних точок, отримана траєкторія могла гіршим чином проходити через інших кандидатів на місцезнаходження в усьому відповідному проміжку часу. Саме тому після завершення процесу оптимізації траєкторії, її рівняння перераховувалось через усі елементи відповідної підтримуючої множини. Для цього використовувався метод найменших квадратів оптимізації поліноміальної регресії другого ступеня [18], який дозволив знайти параметри $A, B, C \in \mathbb{R}^2$, що мінімізують сумарне відхилення траєкторії від відповідних елементів підтримуючої множини (див. формулу 3.4). Використання одразу усіх елементів підтримуючої множини для отримання рівняння траєкторії хоч і погіршило її відхилення від трьох попередніх опорних елементів, але дозволило

покращити загальне відхилення від усіх наявних елементів підтримуючої множини S.

$$\begin{cases} p(k) = Ak^2 + Bk + C \\ \sum_{c_k^j \in S} \left(p(k) - c_k^j\right)^2 \to min, \end{cases} \tag{3.4}$$

де A, B, C – параметри, що підлягають оптимізації, а c_k^j усі елементи підтримуючої множини S.

Також через відсутність опису конкретних характеристик, за якими відбувається пошук початкової та фінішної вершин у заключному етапі алгоритму пошуку найкоротшого шляху, була використана наступна метрика відбору (див. формулу 3.5).

$$\begin{cases} T_{start} = \underset{T_i \mid l_i - k_{start} \le k_{th}}{\arg\max} \; r_i - l_i + 1 \\ T_{finish} = \underset{T_i \mid k_{finish} - r_i \le k_{th}}{\arg\max} \; r_i - l_i + 1, \end{cases} \tag{3.5}$$

де k_{start}, k_{finish} позначають перший та останній кадри з наявними кандидатами на місцезнаходження, а k_{th} – максимальну кількість кадрів поспіль, у яких могло не бути правильних кандидатів.

Такий спосіб відбору початкової та фінішної вершин одночасно враховує той факт, що відповідні траєкторії не повинні сильно віддалятися від початку/кінця відео, а також мають бути доволі якісними, про що може свідчити їх довжина проміжку часу. Також була зроблена невелика модифікація для випадку, коли існують неперервні проміжки часу довжиною більше за k_{th} з відсутністю кандидатів на місцезнаходження. При цьому не існувало шляху між початковою та фінішною вершинами, а тому для виправлення проблеми усе відео було розділено на декілька сегментів, в кожному з яких шлях знаходився незалежно.

4 ОПИС ПРОВЕДЕНИХ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Обґрунтування вибору середовища та технологій реалізації

Для реалізації описаної системи та проведення подальших експериментів була обрана мова програмування Python, оскільки вона є високорівневою та надає велику кількість інструментів зі зручним функціоналом для роботи, а також дозволяє використовувати максимальну продуктивність сучасних операційних систем. Зокрема для створення та навчання нейронної мережі були використані бібліотеки sklearn, tensorflow, keras, а для обробки даних та їх візуалізації – numpy та matplotlib [19]. Для прискорення швидкості тренування та використання моделей було використане хмарне середовище Google Colab [20], яке надає доступ до сучасних потужних графічних процесорів.

4.2 Аналіз існуючих наборів даних

Оскільки для вирішення задачі детекції використовується навчання з вчителем, необхідно заздалегідь зібрати вибірку даних для тренування. У нашому випадку вхідні дані цієї вибірки повинні являти собою окремі кадри з відео тенісних матчів, а розмічені вихідні дані – інформацію про координати місцезнаходження м’яча. Загалом для ефективного навчання з метою подальшої коректної роботи моделі, можна виділити декілька основних вимог до цієї вибірки.

По-перше, кількість екземплярів вибірки повинна бути якомога більшою, оскільки в іншому випадку може спостерігатись недонавчання – ситуація, коли модель не може зрозуміти тенденцію, що лежить в основі даних. При цьому варто враховувати, що певна частина вибірки залишається для валідаційного та тестового наборів даних. Валідаційний набір потрібен для того, щоб уникати перенавчання, а тестовий – щоб

порівнювати ефективність якості роботи різних моделей між собою. Також варто зазначити, що великий розмір вибірки уповільнює навчання, а тому необхідно знаходити баланс у розмірі, щоб з одного боку його було достатньо для повного навчання моделі, а з іншого – час навчання був доступним з точки зору ресурсів.

По-друге, вибірка повинна складатися з різноманітних об’єктів, як за вхідними, так і вихідними параметрами. Звичайно, для виконання першої вимоги до вибірки про кількість екземплярів, можна наробити велику кількість однакових або майже однакових копій одного й того самого об’єкту, але це не буде мати жодного сенсу, оскільки не буде реалізована здатність нейронних мереж до узагальнення. В ідеалі, розподіл екземплярів тренувальної вибірки має збігатися з розподілом об’єктів, з якими в подальшому буде працювати натренована модель. В нашому випадку різноманітність екземплярів вибірки за вхідними параметрами має полягати в наступному:

а) повинні бути наявні відео матчів з усіма існуючими тенісними покриттями для корту, а саме з твердим (хардовим), ґрунтовим та трав’яним покриттями;

б) повинні бути наявні відео як жіночих, так і чоловічих матчів;

в) повинні бути наявні кадри відео з усіма можливими типами бачення м’яча, а саме:

1) кадри, де м’яч знаходиться поза межами бачення камери;

2) кадри, де м’яч не може бути легко ідентифікований у зв’язку з накладанням на інші елементу корту (такі як лінії, надписи, тощо);

3) кадри, де м’яч не видно у зв’язку з тим, що його бачення перекривається іншим об’єктом, зокрема гравцем;

4) кадри, де м’яч являє собою довгу лінію у зв’язку з великою швидкістю;

5) усі інші кадри, де м’яч може бути легко ідентифікований та являє собою звичайне коло.

В той же час, різноманітність екземплярів вибірки за вихідними параметрами має полягати в повному покритті корту місцезнаходженням м’яча, а також наявністю екземплярів з відсутністю м’яча (наприклад у випадку, коли він знаходиться високо у повітрі і не потрапляє у межі бачення камери).

Єдиний відомий на сьогоднішній день відкритий набір даних з розміченою інформацією про місцезнаходження м’яча є TrackNet Dataset [21]. Цей набір даних складається з 95 відео окремих розіграшів, кожен з яких належить одному з десяти обраних матчів. Кожне відео має роздільну здатність 1280×720 пікселей, частоту кадрів 30 fps, а їх загальна тривалість становить близько 11 хвилин. Усього цей датасет містить 19835 розмічених кадрів.

Аналізуючи різноманітність вхідних даних цього набору, можна сказати, що усі обрані десять матчів охоплюють як і усі можливі покриття тенісних кортів, так і мають в наявності поєдинки і чоловічих, і жіночих матчів (див. рис. 4.1).

Рисунок 4.1 – Різноманітні екземпляри набору даних TrackNet Dataset

Також варто відмітити, що на різних турнірах відносне положення камери для зйомки може трохи змінюватися, а тому місцеположення корту та, відповідно, м’яча теж відрізняються. Саме тому наявність кадрів з різних турнірів можете сприяти більш ефективному узагальненню моделі. Розмічені дані містять не тільки інформацію про місцезнаходження м’яча для кожного окремого кадру відео, а також і додаткову метаінформацію, яка дозволяє більш доцільно використовувати датасет. Приклад частини розмічених даних можна побачити на малюнку нижче (див. рис. 4.2).

```
file name,visibility,x-coordinate,y-coordinate,status
0068.jpg,1,518,322,2
0069.jpg,1,505,340,0
0070.jpg,1,489,363,0
0071.jpg,1,471,386,0
0072.jpg,1,457,409,0
0073.jpg,1,442,434,0
0074.jpg,0,,,
0075.jpg,0,,,
0076.jpg,0,,,
0077.jpg,0,,,
0078.jpg,1,375,439,0
0079.jpg,1,365,441,0
0080.jpg,2,354,444,0
0081.jpg,1,337,449,0
0082.jpg,1,327,452,1
0083.jpg,3,364,415,1
```

Рисунок 4.2 – Приклади частини розмічених даних TrackNet Dataset

На рисунку можна побачити, що у файлі розмітки для кожного кадру відео зберігається наступна інформація:

– filename – назва файлу з зображенням;

– x та y координати центра місцезнаходження м’яча;

– visibility – значення від 0 до 3, що характеризує якість бачення м’яча (0 позначає, що м’яч взагалі не видно на кадрі, 1 – м’яч може бути легко знайдено, 2 – м’яч знаходиться у кадрі, але не може бути легко знайденим, 3 – бачення м’яча перекрито іншими об’єктами в кадрі);

– status – значення від 0 до 2, що характеризує тип руху м’яча (0 позначає політ, 1 – момент удару, 2 – момент дотику до корту).

Для кращого розуміння, приклади якості бачення м’яча другого та третього типів наведені на малюнку нижче (див. рис. 4.3).

Рисунок 4.3 – Приклади бачення м’яча другого та третього типів

На верхній частині зображення можна побачити приклад поганого бачення м’яча другого типу у зв’язку зі збігом позиції м’яча з сіткою такого ж білого кольору. На нижній частині зображення можна побачити два послідовних кадра, серед яких у другому м’яч повністю «заховався» за гравцем. Але незважаючи на те, що м’яч не видно на одному зображенні, його позицію можна досить точно відновити, використовуючи попередні та наступні кадри. Саме це і є основною відмінністю третього типу бачення від нульового, де м’яч взагалі не знаходиться в полі бачення камери.

Для більш детального аналізу цього набору даних, за допомогою Python був реалізований допоміжний скрипт, який підрахував загальну

кількість кадрів кожного типу. Отриману інформацію можна побачити у таблиці 4.1.

Таблиця 4.1 – Кількість кадрів кожного з типів у набору

Тип кадру	Тип покриття корту			
	Хард	Ґрунт	Трава	Усього
Visibility 0	534	192	3	729
Visibility 1	9951	5562	2119	17632
Visibility 2	882	417	93	1392
Visibility 3	61	14	7	82
Status 0	10299	5677	2091	18067
Status 1	303	148	65	516
Status 2	292	168	63	523
Чоловічий	9855	4227	2222	16304
Жіночий	1573	1958	0	3531
Усього	33750	18363	6663	58776

Аналізуючи цю таблицю можна сказати, що розподіл кількості кадрів збігається з практичним розподілом ситуацій кожного з типів. Зокрема, найбільша кількість змагань проходить саме на хардових кортах, другим за популярністю покриттям є ґрунт, і останнім – трава. Те ж саме можна сказати й про статус польоту м’яча: більшість часу він знаходиться саме у польоті, а моментів удару та відскоку від корту значно менше.

Для аналізу різноманітності вихідних даних TrackNet Dataset, за допомогою Python був реалізований ще один скрипт, який зібрав інформацію щодо розподілу місцезнаходження м’яча, а саме для кожного розміченого кадру поставив відповідну точку на загальній карті розподілу (див. рис. 4.4).

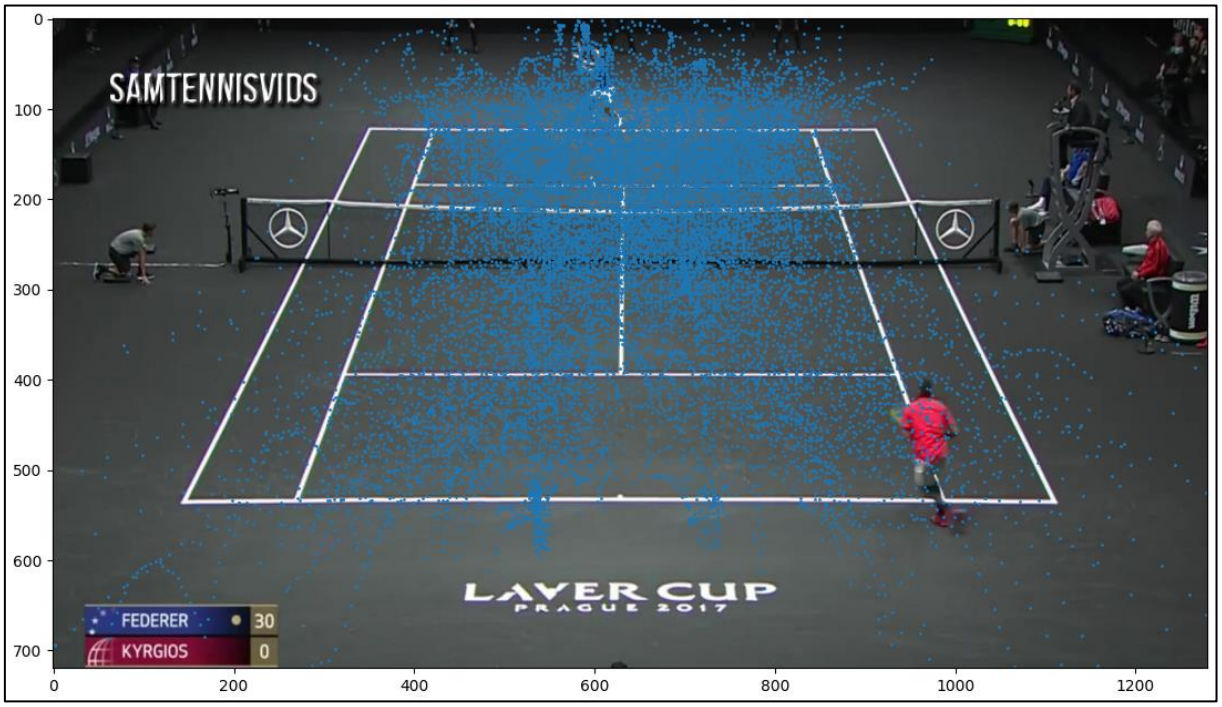

Рисунок 4.4 – Розподіл місцезнаходження м’яча TrackNet Dataset

На рисунку можна побачити, що в сукупності усі екземпляри вибірки охоплюють усі частини корту, що ще раз підкреслює її придатність до використання. Загалом можна сказати, що набір даних TrackNet Dataset відповідає усім вимогам якості, а тому його екземпляри можуть бути використані для тренування та тестування моделей.

4.3 Навчання глибинної нейронної мережі

Описаний у попередньому розділі набір даних був розділений на 3 частини: тренувальну, валідаційну та тестову. Для того, щоб у різних частинах не було дуже схожих між собою вхідних екземплярів, кадри з кожного з 95 розіграшів були повністю передані до однієї частини. При цьому до валідаційної частини потрапили по одному найкоротшому розіграшу з кожного матчу, до тестової – по одному найбільш довгому, а до тренувальної – решта. Такий спосіб розділення дозволив дотриматись необхідних пропорцій розмірів частин (тренувальна найбільш велика,

валідаційна найменша). Загалом до тренувальної вибірки потрапило 13841 кадрів, а до валідаційної та тестової – 771 та 5223 відповідно.

При тренуванні для досягнення більшої гнучкості моделі активно використовувалась аугментація даних, яка полягала в трьох перетвореннях:

– випадкове горизонтальне відображення кадрів;

– обрізка кадрів, при якій довжина та ширина зображення зменшується не більше ніж на 5%;

– використання одного і того ж кадру у одній випадковій з трьох можливих позицій знаходження серед трьох послідовних кадрів.

Описані перетворення дозволили суттєво збільшити варіативність вибірки і в той же час не погіршили якість та загальний вигляд оброблених даних.

У якості функції втрат використовувалась модифікована функція бінарної перехресної ентропії (див. формулу 4.1), яка була запропонована у 2018 році [22].

$$BFCE(\hat{y}, y) = -\sum_{i=1}^{n}((1-y_i)^{\gamma}\hat{y}_i \log y_i + y_i^{\gamma}(1-\hat{y}_i)\log(1-y_i)\,, \qquad (4.1)$$

де $\hat{y}$ – фундаментально істинні бінарні значення класів;

y – передбачені ймовірності;

n – загальна кількість передбачень;

γ – параметр функції.

При цьому використовувалося рекомендоване значення параметру $\gamma = 2$. На відміну від звичайної функції бінарної перехресної ентропії, яка може бути отримана при $\gamma = 0$, модифікована дозволяє додати більшу вагу для більш складних об’єктів для класифікації. В нашому випадку через надзвичайно маленький розмір м’яча, 99.9949% пікселів для класифікації

являють собою фон, а тому існує надзвичайно великий дисбаланс у кількості пікселів кожного з двох класів. Використання модифікованої функції втрат $BFCE$ дозволило вирішити цю проблему та змусити модель все ж таки знаходити одиничні значення, а не просто передбачувати усі ймовірності рівними нулю.

Для оптимізації ваг нейронної мережі використовувалася модифікація стохастичного градієнтного пошуку Adadelta. Загалом було виконано 300 епох навчання, кожна з яких охоплювала 10% повного тренувального набору даних. Для відстеження того, що оптимізація описаної раніше функції втрат дійсно призводить до навчання нейронної мережі правильно передбачувати теплову карту, була використана метрика якості binary intersection over union, яка є основною у задачах сегментації [23]. Цю метрику можна визначити формулою 4.2:

$$\begin{cases} BIoU = \dfrac{Intersecion}{Union} \\ Intersection = \sum_{x=1}^{w}\sum_{y=1}^{h}\left[heatmap_{x,y} > th\right] \wedge gt_{x,y}, \\ Union = \sum_{x=1}^{w}\sum_{y=1}^{h}\left[heatmap_{x,y} > th\right] \vee gt_{x,y} \end{cases} \quad (4.2)$$

де $heatmap$ – передбачена теплова карта;

gt – фундаментально істинна теплова карта;

th – порогове значення, за допомогою якого відбувається бінарізація.

Після виконання 300 епох, навчання було зупинено через відсутність значного покращення функції втрат та метрики BIoU на валідаційному наборі даних. Повний графік залежності функції втрат від кількості пройдених епох на тренувальному та валідаційному наборах даних можна побачити на рисунку нижче (див. рис. 4.5).

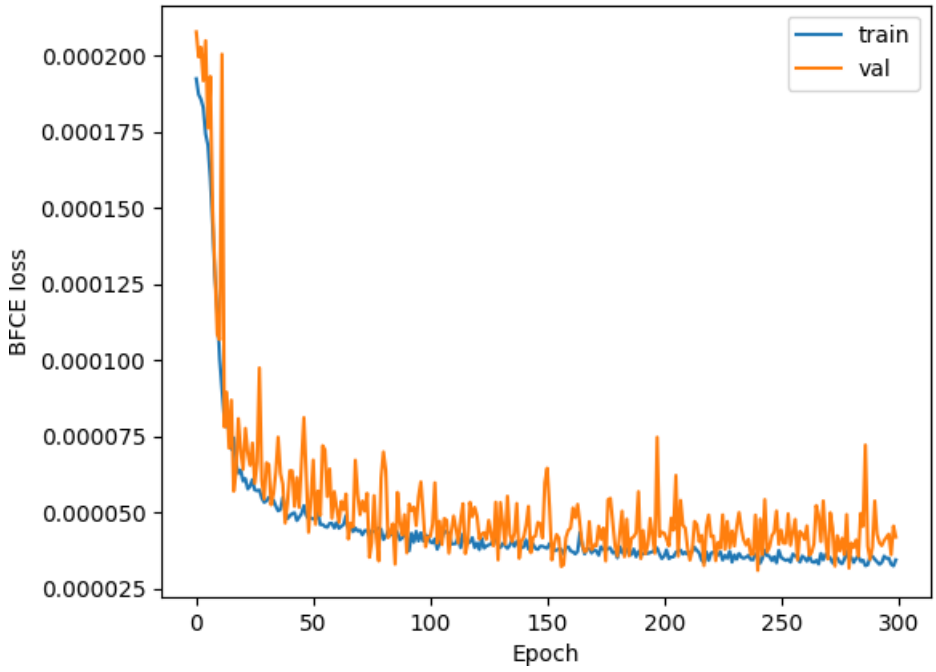

Рисунок 4.5 – Графік залежності функції втрат від кількості епох

Аналогічно графік залежності метрики BIoU від кількості пройдених епох можна побачити на рисунку 4.6. Під час навчання використовувався параметр цієї метрики – порогове значення $th = 0.5$.

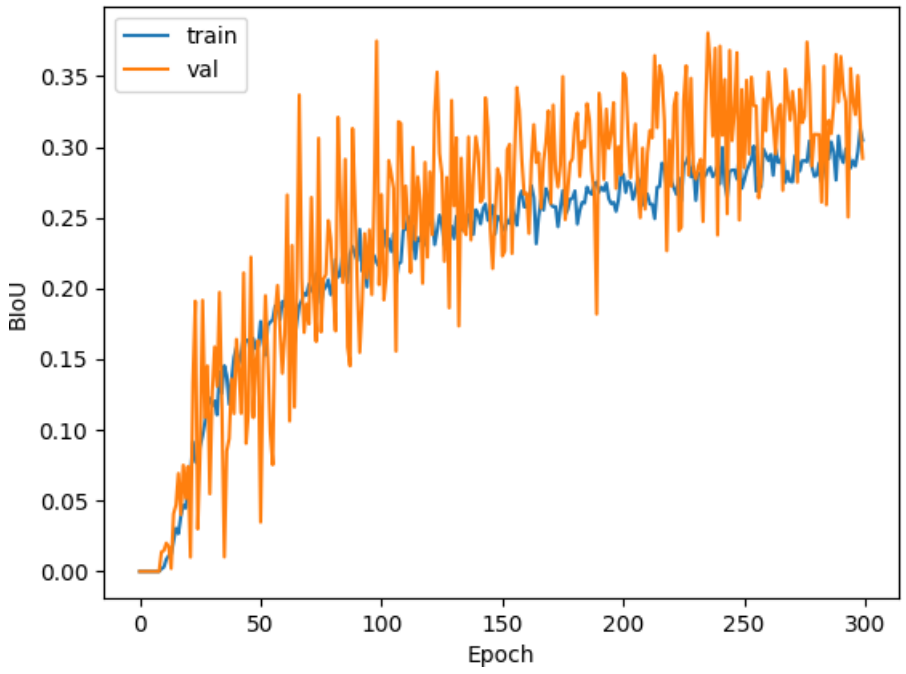

Рисунок 4.6 – Графік залежності метрики BIoU від кількості епох

Аналізуючи графік залежності метрики BIoU можна помітити, що протягом перших 8 епох значення метрики залишалося рівним нулю. Скоріш за все це було викликано надзвичайно великим дисбалансом у кількості пікселів кожного з класів, а тому спочатку модель дуже довго передбачувала більшість пікселів нульовим фоновим значенням. Але завдяки правильно обраній функції втрат нейронна мережа все ж таки змогла сконцентруватися на одиничних значеннях та досить швидко досягла значення BIoU = 0.15, після чого повільно покращувала його до 0.3. Після завершення навчання був побудований графік залежності метрики BIoU від порогового значення бінарізації th для пошуку оптимального значення th (див. рис. 4.7).

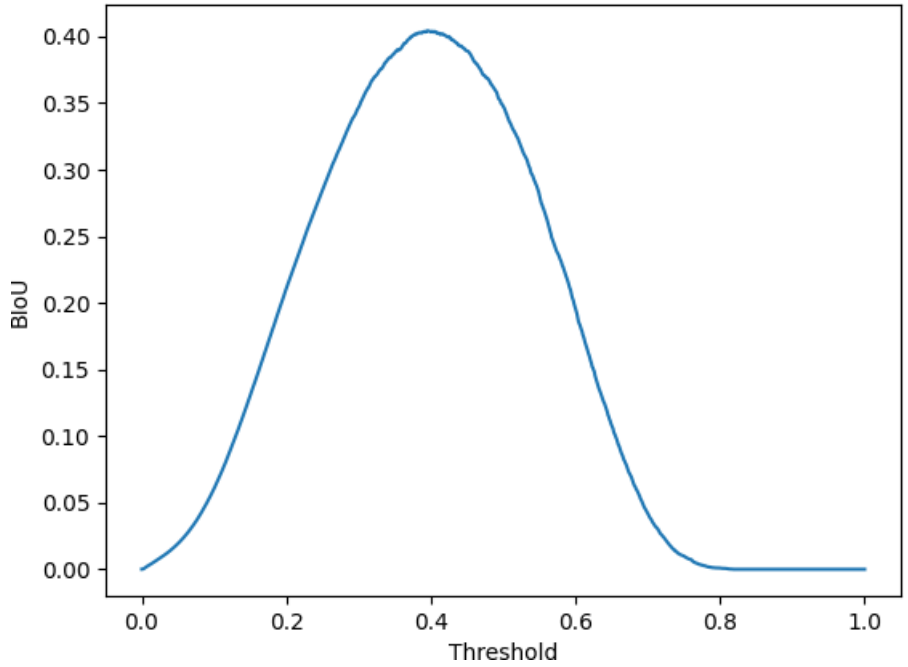

Рисунок 4.7 – Графік залежності BIoU від порогового значення

Побудований графік дозволив однозначно визначити найкраще порогове значення $th = 0.394$, при якому $BIoU = 0.404$, що приблизно на 10 відсотків перевищує відповідне значення метрики під час тренування, оскільки тоді використовувалося порогове значення 0.5.

4.4 Підбір гіперпараметрів алгоритму другої складової системи

Більшість параметрів були виставлені як рекомендовані значення у публікації [14]. Зокрема, розмір вікна $N = 15$, максимальна кількість кадрів поспіль з відсутністю правильних кандидатів на місцезнаходження $k_{th} = 15$, а $d_{th} = 7$ оскільки запропоноване у публікації значення 3 було для відео в понад два рази меншої роздільної здатності. Для визначення параметру R, що позначає максимальну відстань, яку м'яч може подолати за час одного кадру, був підрахований розподіл цієї відстані на тренувальному наборі даних (див. рис. 4.8).

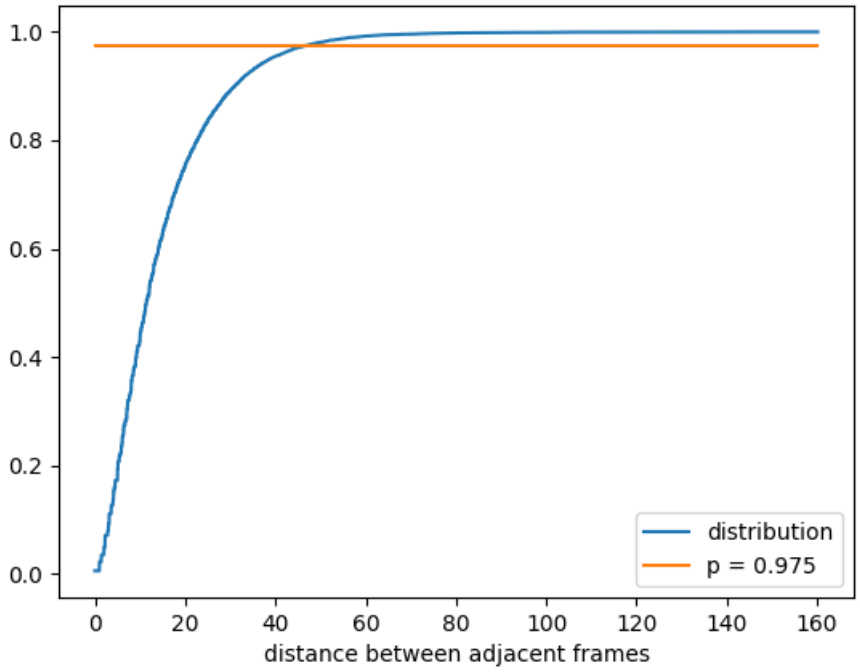

Рисунок 4.8 – Розподіл відстані між послідовними кадрами

На рисунку можна побачити функцію розподілу випадкової величини R. А саме, для кожної порогової відстані R' побачити ймовірність того, що відстань між позиціями м'яча у двох послідовних кадрах не буде перевищувати R'. Після аналізу цього розподілу було обрано значення $R = 47$ як 97.5% квантиль розподілу.

4.5 Оцінка ефективності створеної системи

Перед безпосередньою оцінкою різними метриками була проведена візуальна оцінка теплових карт, що генерує натренована нейронна мережа (див. рис. 4.9).

Рисунок 4.9 – Приклад згенерованої теплової карти

На наведеному прикладі можна побачити яскравий круг, який відповідає правильному місцезнаходженню м’яча. Якщо придивитися уважніше, то можна побачити більш бляклі плями, які знаходяться у зоні знаходження гравців. Незважаючи на їх наявність, в більшості випадків ці плями зникнуть при бінарізації теплових карт шляхом порівняння з пороговим значенням, а залишаться тільки яскраві області.

Для оцінки ефективності створеної системи використовувались метрики, описані у розділі 2.1. Оскільки описані метрики дозволяють оцінити якість детекції, то було також проведено порівняння результатів роботи окремої нейронної мережі, де передбачення місцезнаходження робиться тільки у випадку наявності одного кандидату, а також повної системи, де на другому етапі спеціальний алгоритм обирає правильних кандидатів з усіх наявних, а також за допомогою функції траєкторії дозволяє відновити деякі пропущені об'єкти. При цьому ефективність була досліджена окремо на тренувальному та тестовому наборах даних.

Оскільки усі досліджені метрики базуються на значенні похибки ε_k, яка показує відхилення передбаченого місцезнаходження від реального, то першим кроком був побудований графік її розподілу (див. рис. 4.10).

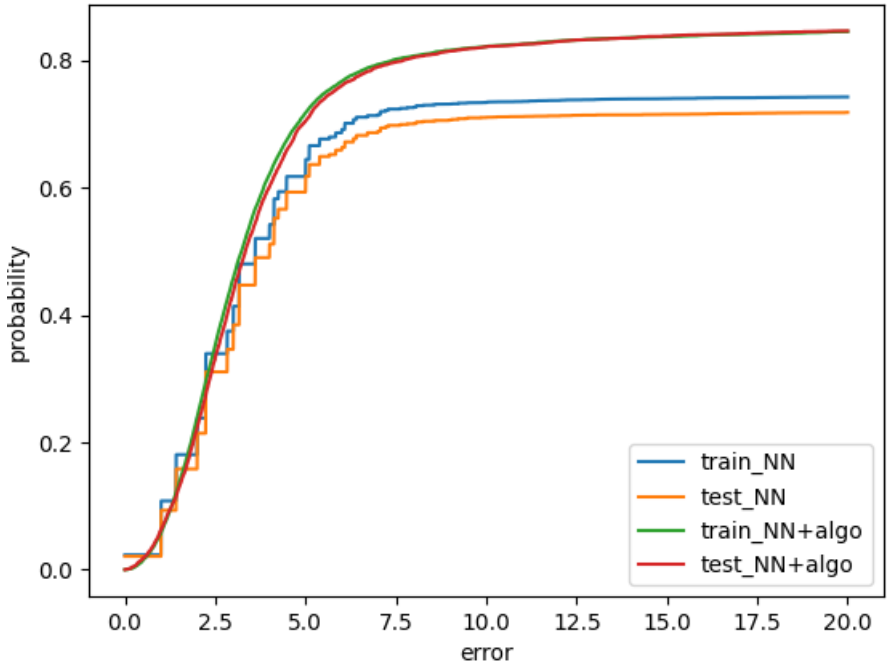

Рисунок 4.10 – Графік розподілу похибки

На рисунку можна помітити, що максимальне значення ймовірності не доходить до одиниці, оскільки враховувались лише ті кадри, для яких було зроблено передбачення. Можна побачити, що результати системи на

тренувальних даних трохи перевищують відповідні результати на тестових даних, що є повністю очікуваним. Також можна побачити, що функція розподілу похибки результатів окремої нейронної мережі (NN) є меншою за відповідну функцію повної системи (NN + algo). Це свідчить про те, що алгоритм з другої складової системи дозволяє обрати правильні кандидати в той час як окрема нейронна мережа просто ігнорує результати у випадку наявності декількох кандидатів.

Для знаходження показників $precision$, $recall$, $F_1\ score$ використовувалось порогове значення максимальної допустимої похибки $\varepsilon' = 7$, що збігається з параметром другої складової алгоритму d_{th}, який також відповідає максимально можливому відхиленню. Таке значення похибки візуально майже непомітно, а тому може бути використано для підрахунку метрик. Знайдені показники можна побачити у таблиці нижче (див. табл. 4.2).

Таблиця 4.2 – Метрики розробленої системи

Набір даних та алгоритм	$precision$	$recall$	$F_1\ score$
train_NN	94.08%	74.28%	83.02%
test_NN	93.31%	72.03%	81.3%
train_NN + algo	90.99%	86.12%	88.49%
test_NN + algo	89.63%	86.6%	88.09%

Отримані значень метрик дозволяють зробити два спостереження. По-перше, різниця між якістю системи на тренувальних та тестових даних є дуже незначною (відповідні показники відрізняються лише на 1-2%), що може бути зумовлено активною аугментацією даних при тренуванні, що не дало можливості нейронній мережі «запам’ятати» елементи тренувального набору даних. По-друге, алгоритм другої складової системи дозволяє в першу чергу суттєво покращити показник $recall$, але через більшу кількість передбачень трохи знижує їх влучність. Для більшої наочності

результатів роботи другої складової системи нижче наведено приклад відновлення траєкторії за наявними кандидатами (див. рис. 4.11). Для прикладу використовувалося найдовше тестове відео, що складалося з 900 кадрів.

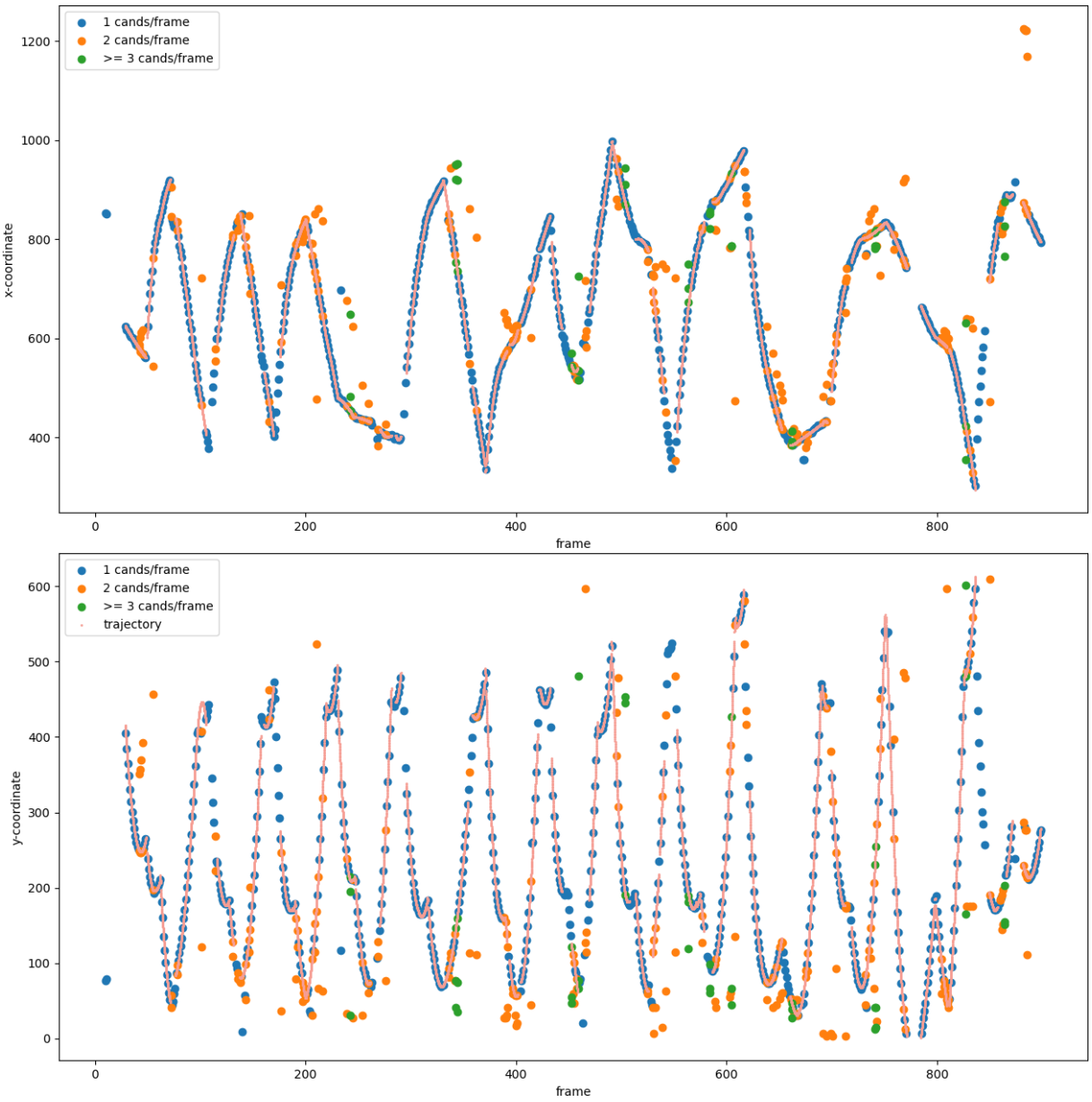

Рисунок 4.11 – Відновлення траєкторії за наявними кандидатами

На рисунку можна побачити дві окремі неперервні відновленні траєкторії по x та y координатам, зображені рожевим кольором. При цьому також окремими точками зображено існуючі дискретні кандидати,

знайдені першою складовою системи – нейронною мережею. В залежності від кількості знайдених кандидатів у кожному кадрі використовується різний колір для їх відображення. Можна побачити, що усі відновленні частини траєкторії проходять через правильні кандидати, що свідчить про можливість алгоритму якісно відбирати існуючі кандидати.

Окрім покрашення якості, знайдене рівняння неперервної траєкторії дозволяє відобразити її на перемальованому відео (див. рис. 4.12).

Рисунок 4.12 – Приклад кадру перемальованого відео

На рисунку можна побачити неперервну траєкторію польоту м’яча за останні декілька кадрів, що значно полегшує перегляд відео. Останнім показником ефективності є FPS. На сучасному GPU один прохід через нейронну мережу займає 137 мс та генерує теплову карту для одразу трьох послідовних кадрів, а тому $FPS = \frac{3}{0.137} \approx 22 \frac{\text{кадри}}{\text{с}}$.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було проведено детальний аналіз предметної галузі обраної тематики та показано актуальність роботи. Були розглянуті основні підходи до вирішення загальної задачі трекінгу, які дозволили визначити структуру системи для розв’язання задачі трекінгу польоту м’яча, основним компонентом якої є глибинна нейронна мережа. Також був проведений аналіз існуючих методів навчання нейронних мереж.

В ході проведення теоретичного аналізу були розглянуті та обрані об’єктивні метрики оцінки якості системи, а також існуючі підходи до вирішення поставленої задачі. Наступним кроком стала модифікація розглянутих аналогів шляхом знаходження та усунення їх слабких місць. Зокрема, була створена архітектура нейронної мережі, яка поєднує у собі елементи архітектур U-net і TrackNet, що дозволило одночасно покращити якість і швидкість роботи.

Заключним етапом роботи стало проведення експериментальних досліджень, де важливу роль відіграє набір даних. Був проведений детальний пошук датасетів, в результаті якого був знайдений існуючий набір різноманітних даних TrackNet Dataset. Під час навчання нейронної мережі активно використовувалась техніка аугментації даних, яка допомогла запобігти перенавчанню. Натренована система була протестована на знайденому наборі даних та досягла показників влучності та повноти 89.6% та 86.6% , що не поступається і навіть трохи перевершує відповідні показники існуючих аналогів. Також була досягнута швидкість обробки 22 кадри в секунду, що наближує систему для можливості модифікації відео в режимі реального часу.

В результаті виконання роботи були також написані тези доповіді до науково-технічної конференції, присвячену сучасним напрямкам розвитку інформаційно-комунікаційних засобів управління [24].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ashley J. The 5 Fastest Smashes in Badminton Ever. URL: https://www.triviasharp.com/the-5-fastest-smashes-in-badminton-ever/ (дата звернення 30.03.2022).

2. Renton. J. The 2021 ITF World Tennis Tour in Numbers. URL: https://www.itftennis.com/en/news-and-media/articles/8000-players-34000-matches-1500-titles-the-2021-itf-world-tennis-tour-in-numbers/ (дата звернення 01.04.2022).

3. Sourav. D. Most Popular Sports in The World 2022. URL: https://sportsbrowser.net/most-popular-sports/ (дата звернення 03.04.2022).

4. Aggarwal A., Biswas S., Singh S., Sural S., Majumdar A.. Object Tracking Using Background Subtraction and Motion Estimation in MPEG Videos, *Computer Vision – ACCV*. 2006. 11 p.

5. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. P. 779–788.

6. Згорткова нейронна мережа – просте пояснення CNN та її застосування. URL: https://evergreens.com.ua/ua/articles/cnn.html (дата звернення 03.04.2022).

7. C. Aggarwal C. Neural Networks and Deep Learning A Textbook / Charu C. Aggarwal., 2018. 497 p.

8. Bilonoh B., Bodyanskiy Y., Kolchygin B., Mashtalir S. Tunable Activation Functions for Deep Neural Networks. *Lecture Notes in Computational Intelligence and Decision Making. ISDMCI 2021. Lecture Notes on Data Engineering and Communications Technologies* / Eds. Babichev S., Lytvynenko V. Cham: Springer, 2021. Vol 77. P. 624–633.

9. Shung K. Accuracy, Precision, Recall or F1. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (дата звернення 12.04.2022).

10. Archana M., Geetha K. Object Detection and Tracking based on Trajectory in Broadcast Tennis Video. *Procedia Computer Science*. 2015, vol. 58, P. 225–232.

11. Huang Y., Liao I., Chen C., Peng W. TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications. *16th IEEE International Conference AVSS,* 2019. 12 p.

12. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. *The 3rd International Conference on Learning Representations.* 2015. 14 p.

13. Noh H., Hong S., Han B. Learning deconvolution network for semantic segmentation. *Proceedings of the IEEE International Conference on Computer Vision.* 2015, P. 1520–1528.

14. Yan F., Kostin A., Christmas W., Kittler J. A Novel Data Association Algorithm for Object Tracking in Clutter with Application to Tennis Video Analysis. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* 2006. Vol. 1. P. 634–641.

15. Dijkstra E. A note on two problems in connexion with graphs. *Numer. Math*. 1959. Vol. 1. P. 269–271.

16. Sun N., Lin Y., Chuang S., Hsu T., Yu D., Chung H. TrackNetV2: Efficient Shuttlecock Tracking Network. *International Conference on Pervasive Artificial Intelligence (ICPAI)*. 2020. P. 86–91.

17. Ronneberger O., Fischer P., Brox T. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI.* 2015. P. 234–241.

18. Fitting a polynomial using least squares method. URL: https://neutrium.net/mathematics/least-squares-fitting-of-a-polynomial/ (дата звернення 17.04.2022).

19. McGreggor D. Mastering Matplotlib. *Pact Publishing*. 2015. 294 p.

20. Bisong E. Google Colaboratory. Building Machine Learning and Deep Learning Models on Google Cloud Platform: a Comprehensive Guide for Beginners. *Apress, Berkeley, CA.* 2015. P. 59–64.

21. Huang Y. Dataset for Tracking High-speed and Tiny objects in Sport Applications. URL: https://nol.cs.nctu.edu.tw:234/open-source/TrackNet#dataset (дата звернення 19.04.2022).

22. Lin T., Goyal P., Girshick R., He K., Dollar P. Focal Loss for Dense Object Detection. *IEEE International Conference on Computer Vision (ICCV)*, 2017. P. 2999–3007.

23. Tiu E. Metrics to Evaluate your Semantic Segmentation Model. URL: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2 (дата звернення 20.04.2022).

24. Асландуков М. М. Використання глибинної нейронної мережі для покращення якості відео тенісних матчів. *Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління*: матеріали наук.-техн. конф. (м. Харків, 27 – 28 квіт. 2022 р.). Харків, 2022. Т. 2. С. 5.