

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 _____ перший (бакалаврський)

Система автоматизації суддівства у спортивній гімнастиці за допомогою
автоматів станів та двонапрямних рекурентних мереж
(тема)

Виконав:
здобувач _____ четвертого _____ року навчання,
групи _____ ІТШ-22-4

_____ Ігор ДИШУК
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)

Тип програми _____ освітньо-професійна

Освітня програма _____ Штучний інтелект

(повна назва освітньої програми)

Керівник _____ ас. Сергій ГРИНЬОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ

(підпис)

_____ Лариса ЧАЛА
(власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____

Кафедра _____ Штучного інтелекту _____

Рівень вищої освіти _____ перший (бакалаврський) _____

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

Тип програми _____ освітньо-професійна _____

Освітня програма _____ Штучний інтелект _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

« _____ » _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Дишуку Ігорю Вадимовичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Система автоматизації суддівства у спортивній гімнастиці за допомогою автоматів станів та двонапрямних рекурентних мереж _____

затверджена наказом університету від 5 червня 2026 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 8 липня 2026 р.

3. Вихідні дані до роботи Науково-технічні публікації з комп'ютерного зору, аналізу скелетних даних, автоматів станів та рекурентних нейронних мереж; правила суддівства FIG зі спортивної гімнастики; матеріали щодо існуючих систем відеоаналізу та підтримки суддівства; документація Python, OpenCV, MediaPipe, PyTorch, NumPy, Tkinter; відеозаписи гімнастичних вправ і сформований набір даних для навчання та тестування системи.

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі та постановка задачі

2) Математичні основи та архітектура інтелектуальної системи

3) Програмна реалізація модулів аналізу та класифікації

4) Інтерфейс користувача та аналіз роботи системи

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.06.2026	виконано
2	Аналіз предметної галузі та особливостей суддівства у спортивній гімнастиці	08.06.2026	виконано
3	Огляд існуючих рішень для відеоаналізу та підтримки суддівства	10.06.2026	виконано
4	Розробка математичної моделі та архітектури інтелектуальної системи	13.06.2026	виконано
5	Реалізація алгоритму отримання скелетної моделі та сегментації руху	16.06.2026	виконано
6	Формування набору даних і підготовка скелетних послідовностей	18.06.2026	виконано
7	Навчання та тестування моделі Bi-GRU для класифікації гімнастичних елементів	21.06.2026	виконано
8	Реалізація алгоритму розрахунку оцінки складності D-score	23.06.2026	виконано
9	Розробка графічного інтерфейсу користувача та демонстрація роботи системи	25.06.2026	виконано
10	Написання пояснювальної записки та оформлення додатків	28.06.2026	виконано
11	Перевірка роботи, виправлення зауважень і підготовка до нормоконтролю	30.06.2026	виконано
12	Підготовка презентації та доповіді до захисту	01.07.2026	виконано
13	Попередній захист кваліфікаційної роботи	03.07.2026	виконано
14	Захист перед екзаменаційною комісією	08.07.2026	

Дата видачі завдання 8 червня 2026 р.

Здобувач 
(підпис)

Керівник роботи 
(підпис)

ас. Сергій ГРИНЬОВ
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 99 с., 10 рис., 2 дод., 21 джерело.

АВТОМАТИ СТАНІВ, ВІДЕОАНАЛІЗ, ДВОНАПРЯМНІ РЕКУРЕНТНІ МЕРЕЖІ, КОМП'ЮТЕРНИЙ ЗІР, СПОРТИВНА ГІМНАСТИКА, СУДДІВСТВО, BI-GRU, D-SCORE, MEDIAPIPE.

Об'єкт дослідження – процес відеоаналізу та оцінювання гімнастичних вправ.

Предмет дослідження – методи екстракції скелетних моделей, автомати станів та рекурентні нейронні мережі для класифікації гімнастичних елементів.

Мета роботи – розробка інтелектуальної системи автоматизованого відеоаналізу гімнастичних вправ для об'єктивного розрахунку оцінки складності D-score.

Методи дослідження – теоретичний аналіз правил FIG, підходів до відеоаналізу рухів та рекурентних нейронних мереж; експериментальний метод, що включає програмну реалізацію системи, формування набору даних, навчання та тестування моделі. Методи розробки базуються на Python, OpenCV, MediaPipe та PyTorch.

У результаті роботи проведено теоретичний аналіз існуючих підходів до автоматизованого відеоаналізу гімнастичних вправ, методів екстракції скелетних моделей, алгоритмів сегментації рухів та архітектур рекурентних нейронних мереж. Для навчання й тестування моделі було сформовано власний набір даних з анотованими послідовностями скелетних поз. Отримані дані використано для навчання Bi-GRU моделі, класифікації гімнастичних елементів і розрахунку D-score. За результатами тестування підтверджено працездатність запропонованого підходу для класифікації гімнастичних елементів.

ABSTRACT

Bachelor's thesis contains: 99 pp., 10 fig., 2 ann., 21 references.

ARTISTIC GYMNASTICS, BI-GRU, BIDIRECTIONAL RECURRENT NEURAL NETWORKS, COMPUTER VISION, D-SCORE, FINITE STATE MACHINES, JUDGING, MEDIAPIPE, VIDEO ANALYSIS.

The object of research – the process of video analysis and evaluation of gymnastics exercises.

The subject of research – methods of skeletal model extraction, finite state machines, and recurrent neural networks for the classification of gymnastics elements.

The purpose of the work – the development of an intelligent system for automated video analysis of gymnastics exercises for objective calculation of the difficulty score, namely D-score.

Research methods – theoretical analysis of FIG rules, approaches to motion video analysis, and recurrent neural networks; experimental method, including software implementation of the system, dataset formation, model training, and testing. Development methods are based on Python, OpenCV, MediaPipe, and PyTorch.

As a result of the work, a theoretical analysis of existing approaches to automated video analysis of gymnastics exercises, methods of skeletal model extraction, motion segmentation algorithms, and recurrent neural network architectures was carried out. A custom dataset with annotated sequences of skeletal poses was formed for model training and testing. The obtained data were used to train the Bi-GRU model, classify gymnastics elements, and calculate the D-score. The testing results confirmed the applicability of the proposed approach for the classification of gymnastics elements.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	8
Вступ.....	9
1 Аналіз предметної галузі та постановка задачі.....	12
1.1 Особливості оцінювання вправ у спортивній гімнастиці	12
1.2 Огляд існуючих рішень	14
1.2.1 Системи відеоповторів IRCOS	15
1.2.2 Система підтримки суддівства Fujitsu	16
1.2.3 Програмні засоби для мануального відеоаналізу	17
1.2.4 Порівняльний аналіз існуючих рішень.....	18
1.3 Проблематика існуючих методів оцінювання	18
1.4 Мета роботи та постановка задачі.....	21
2 Математичні основи та архітектура інтелектуальної системи.....	23
2.1 Аналіз сучасних наукових підходів до розпізнавання дій на основі скелетних даних	23
2.2 Науково-технічне обґрунтування гібридної архітектури системи	25
2.3 Математична формалізація задачі інтелектуального оцінювання.....	28
2.4 Попередня обробка та просторово-часова нормалізація кінематичних ознак.....	31
2.5 Алгоритмічна сегментація відеопотоку на основі детермінованих автоматів станів	35
2.6 Архітектура та математична модель двонапрямної рекурентної мережі Bi-GRU	41
3 Програмна реалізація модулів аналізу та класифікації.....	48
3.1 Використані технології та програмні засоби	48
3.2 Структура програмного проєкту	50
3.3 Формування та підготовка набору даних	52
3.4 Реалізація обробки відео та екстракції скелетної моделі.....	57
3.5 Реалізація сегментації гімнастичних елементів.....	61

3.6 Реалізація та навчання моделі Bi-GRU	65
3.7 Реалізація алгоритму розрахунку D-score	68
4 Інтерфейс користувача та аналіз роботи системи.....	71
4.1 Структура графічного інтерфейсу користувача.....	71
4.2 Сценарій роботи користувача із системою	74
4.3 Демонстрація результатів обробки відео.....	75
4.4 Обмеження розробленої системи	78
4.5 Перспективи подальшого розвитку.....	80
Висновки	83
Перелік джерел посилання	85
Додаток А Програмний код основного pipeline обробки відео.....	88
Додаток Б Відомість кваліфікаційної роботи.....	99

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Гімнастичний елемент – окремий технічний рух спортсмена, що має визначену назву, групу та базову складність;

Ключова точка – точка скелетної моделі, що відповідає певному суглобу або частині тіла людини;

Руховий сегмент – часовий проміжок відеозапису, який відповідає потенційному гімнастичному елементу;

Скелетна модель – подання тіла людини у вигляді набору ключових точок;

ШІ – штучний інтелект;

Bi-GRU – Bidirectional Gated Recurrent Unit – двонапрямна рекурентна нейронна мережа;

Code of Points – правила суддівства у спортивній гімнастиці;

D-score – Difficulty score – оцінка складності гімнастичної вправи;

E-score – Execution score – оцінка виконання гімнастичної вправи;

FIG – Fédération Internationale de Gymnastique – Міжнародна федерація гімнастики;

FSM – Finite State Machine – автомат станів;

GRU – Gated Recurrent Unit – рекурентний блок із керованими вентилями;

MediaPipe Pose – інструмент комп'ютерного зору для визначення ключових точок тіла людини;

MediaPipe Tasks – набір інструментів MediaPipe для виконання задач комп'ютерного зору;

OpenCV – Open Source Computer Vision Library – бібліотека комп'ютерного зору для обробки зображень і відео;

Tkinter – стандартна бібліотека Python для створення графічного інтерфейсу користувача.

ВСТУП

Сучасний етап розвитку спорту вищих досягнень характеризується високою складністю технічних елементів, зростанням інтенсивності змагальної діяльності та підвищеними вимогами до точності оцінювання. У спортивній гімнастиці, де підсумковий результат часто залежить від десятих і сотих часток бала, об'єктивність суддівства є одним із ключових чинників справедливого визначення результату. Чинна система оцінювання, регламентована Міжнародною федерацією гімнастики (FIG), значною мірою базується на візуальному аналізі рухів спортсмена суддівською бригадою. Проте такий підхід має природні обмеження, пов'язані з людським фактором, високою швидкістю виконання вправ, складністю одночасного контролю багатьох параметрів руху та можливим зниженням концентрації суддів під час тривалих змагань.

Історично суддівство у спортивній гімнастиці базувалося переважно на безпосередньому спостереженні за виконанням вправи. Надалі для уточнення спірних моментів почали використовуватися відеоповтори, які дали змогу переглядати окремі епізоди виступу в уповільненому режимі. Однак навіть у такому випадку остаточне рішення залишається за суддею, а отже, повністю усунути суб'єктивність оцінювання неможливо. У зв'язку з розвитком комп'ютерного зору, машинного навчання та методів аналізу рухів людини з'явилися передумови для створення інтелектуальних систем підтримки спортивного суддівства.

На сьогодні існують окремі технологічні рішення для автоматизації аналізу гімнастичних вправ. Найбільш відомими є системи, що використовують спеціалізовані 3D-сенсори та лазерне сканування тіла спортсмена. Такі рішення забезпечують високу точність аналізу, однак потребують дороговартісного обладнання, складного налаштування та здебільшого доступні лише на змаганнях найвищого рівня. У цьому контексті більш доступним і перспективним напрямом є використання

звичайного відеопотоку, алгоритмів комп'ютерного зору та моделей глибокого навчання для автоматизованого розпізнавання гімнастичних елементів.

Актуальність роботи зумовлена потребою у створенні доступного програмного рішення, здатного автоматизувати процес аналізу гімнастичних комбінацій та допомогти у розрахунку оцінки складності D-score. Особливо важливим це є в умовах, коли суддя має одночасно контролювати технічний зміст вправи, визначати групи складності елементів та враховувати вимоги правил FIG. Автоматизація частини цього процесу дає змогу зменшити когнітивне навантаження на суддів, прискорити обробку виступу та підвищити повторюваність результатів оцінювання.

Метою кваліфікаційної роботи є розробка та дослідження інтелектуальної системи автоматизованого відеоаналізу гімнастичних вправ для розпізнавання елементів і розрахунку оцінки складності D-score. Для досягнення цієї мети в роботі передбачено аналіз предметної області та існуючих рішень, формування набору даних зі скелетними послідовностями рухів, використання MediaPipe Pose для екстракції ключових точок тіла, застосування автоматів станів для сегментації вправи, навчання двонапрямної рекурентної нейронної мережі Bi-GRU для класифікації елементів та реалізацію алгоритму підрахунку D-score.

Об'єктом дослідження є процес відеоаналізу та оцінювання гімнастичних вправ. Предметом дослідження є методи екстракції скелетних моделей, автомати станів та рекурентні нейронні мережі для класифікації гімнастичних елементів.

Практичне значення роботи полягає у можливості використання розробленої системи як допоміжного інструменту для суддів, тренерів і спортсменів. Результати роботи можуть бути застосовані у системах підтримки прийняття рішень у спортивному суддівстві, тренувальних комплексах, програмних засобах відеоаналізу та навчальних проєктах,

пов'язаних із комп'ютерним зором і аналізом рухів людини. Запропонований підхід є доступнішою альтернативою дороговартісним апаратним рішенням, оскільки базується на аналізі стандартного відеопотоку та програмних методах обробки даних.

Окрему увагу в роботі приділено саме оцінці складності вправи, тобто D-score, оскільки цей компонент потребує точного розпізнавання виконаних елементів, їхньої послідовності та відповідності вимогам правил FIG. На відміну від оцінки виконання, яка значною мірою пов'язана з аналізом технічних помилок, розрахунок складності може бути формалізований через визначення елементів, їх груп, вартості та спеціальних вимог. Саме тому автоматизація цього етапу є доцільним напрямом для застосування методів комп'ютерного зору та нейронних мереж.

Таким чином, робота спрямована на поєднання спортивної експертизи та сучасних методів штучного інтелекту для створення доступного інструменту аналізу гімнастичних комбінацій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Особливості оцінювання вправ у спортивній гімнастиці

Спортивна гімнастика належить до переліку найскладніших та найбільш динамічних олімпійських видів спорту. Сучасні вправи на гімнастичних приладах характеризуються надзвичайно високою швидкістю виконання акробатичних елементів та складною біомеханікою рухів. Оцінювання виступів атлетів на міжнародних та національних змаганнях регламентується «Правилами суддівства» (Code of Points), які розробляються та оновлюються Міжнародною федерацією гімнастики (FIG) кожні чотири роки відповідно до нового олімпійського циклу [1]. Ці правила є основним документом, що визначає вартість кожного елемента, вимоги до композиції вправи та критерії нарахування штрафів за технічні помилки.

Основою сучасної системи оцінювання у спортивній гімнастиці є розділення фінального результату на дві незалежні оцінки, які виставляються різними групами експертів:

- оцінка за складність (D-score – Difficulty score), яка базується на технічному вмісті вправи. Вона включає вартість десяти найбільш складних елементів, спеціальні вимоги до груп елементів та надбавки за складні сполучення рухів;

- оцінка за виконання (E-score – Execution score), яка відображає чистоту та техніку виконання. Вона починається з 10,0 бала, з яких судді віднімають штрафи за кожен технічний огріх: зігнуті кінцівки, зайві кроки, падіння тощо [1].

У межах цієї роботи основна увага приділяється саме оцінці за складність, тобто D-score, оскільки вона безпосередньо пов'язана з розпізнаванням виконаних елементів та визначенням їхньої вартості. На відміну від E-score, який залежить від якості виконання, чистоти рухів і візуального визначення помилок, D-score має більш формалізований

характер. Для його розрахунку необхідно встановити, які саме елементи були виконані, до яких груп вони належать, яку складність мають та чи відповідають структурним вимогам.

Згідно з міжнародними правилами FIG, для забезпечення максимальної об'єктивності на змаганнях високого рівня формуються дві окремі суддівські колегії. Перша колегія фокусується виключно на ідентифікації елементів та розрахунку D-score, тоді як друга – на фіксації помилок для виведення E-score. Проте в реаліях українського спорту, особливо на змаганнях регіонального рівня, часто спостерігається дефіцит кваліфікованих кадрів. У таких умовах один спеціаліст нерідко вимушений поєднувати ролі та паралельно заповнювати обидва протоколи оцінювання.

Процес роботи судді є поетапним і вимагає високої концентрації. Під час виступу гімнаста, який триває від 30 до 90 секунд, суддя повинен у реальному часі виконувати графічний запис вправи за допомогою спеціальної стенографії – символів FIG. Кожен виконаний елемент має свій унікальний символ та відповідну групу складності: від «А» – 0,1 бала, до «J» – 1,0 бала.

Після завершення виступу суддя розшифровує стенограму, обирає найдорожчі елементи, перевіряє виконання структурних вимог та вираховує підсумковий бал за складність.

Визначення D-score ускладнене тим, що гімнастична комбінація є безперервною послідовністю рухів, у якій окремі елементи швидко переходять один в інший і можуть поєднуватися маховими або підготовчими фазами. Тому судді необхідно не лише розпізнати сам елемент, а й правильно визначити його початок, завершення та ключові технічні положення. Будь-яка неточність у фіксації цих моментів може призвести до неправильного визначення вартості елемента або його неврахування під час розрахунку складності.

Важливим аспектом є те, що кожен гімнастичний елемент має чітко визначені технічні ознаки, за якими він ідентифікується як конкретний рух

із таблиці правил. До таких ознак належать положення тіла, напрям обертання, амплітуда руху, фаза фіксації та критичні кути відхилення. Наприклад, виконання повороту на жердині вимагає точного визначення моменту початку та завершення оберту відносно вертикальної осі приладу.

Через необхідність точного визначення технічних ознак кожного елемента процес підрахунку D-score часто потребує додаткового часу після завершення виступу. Суддя має перевірити правильність запису комбінації, уточнити спірні моменти, визначити вартість елементів та врахувати структурні вимоги. Це може призводити до тривалих пауз у ході змагань, особливо коли один фахівець самостійно опрацьовує дані за двома критеріями.

Використання засобів автоматизації дозволяє прискорити технічну частину підрахунку балів і отримати результат за складність швидше, ніж при традиційному візуальному аналізі та ручному стенографуванні.

Таким чином, процес суддівства у гімнастиці є складним аналітичним завданням, що вимагає одночасного розпізнавання швидких рухів атлета, визначення часових меж елементів, аналізу їхніх технічних ознак та класифікації згідно з правилами. Висока інтенсивність виконання вправ і необхідність ручного підрахунку балів створюють значне навантаження на експертів. Це зумовлює актуальність дослідження підходів до автоматизації сегментації вправи на фази та класифікації елементів для об'єктивного розрахунку D-score.

1.2 Огляд існуючих рішень

На сьогодні у спортивній гімнастиці існують певні технологічні рішення для підтримки суддівства та тренувального процесу, проте вони суттєво відрізняються за рівнем автоматизації та доступністю. До таких рішень можна віднести системи відеоповторів, автоматизовані комплекси з

використанням 3D-сенсорів, а також програмне забезпечення для мануального відеоаналізу техніки рухів.

1.2.1 Системи відеоповторів IRCOS

Одним із найбільш поширених методів контролю на офіційних змаганнях є використання систем відеодопомоги, таких як IRCOS (Instant Replay and Control System) [2]. Ця система базується на багатокамерній зйомці виступу з різних ракурсів, що дозволяє суддям переглядати вправу в уповільненому режимі або кадр за кадром.

Використання IRCOS є критично важливим під час подання протестів на оцінку, оскільки дозволяє уточнити технічний зміст програми. Проте головним недоліком системи залишається її повна залежність від людського фактора: відео лише надає чіткіше зображення, але ідентифікацію елементів та розрахунок балів все одно здійснює суддя вручну на основі власного візуального аналізу. На рисунку 1.1 наведено приклад робочого інтерфейсу системи відеоповторів.



Рисунок 1.1 – Інтерфейс системи миттєвого відеоповтору IRCOS

1.2.2 Система підтримки суддівства Fujitsu

Одним із найбільш технологічно розвинених автоматизованих рішень є система підтримки суддівства (Judging Support System), розроблена компанією Fujitsu у співпраці з FIG [3]. Технологія використовує 3D-лазерні сенсори (LIDAR), які зчитують тисячі точок на тілі гімнаста та створюють точну цифрову копію його рухів у тривимірному просторі.

Система здатна автоматично вимірювати кути розведення ніг, амплітуду махів та чіткість фіксації положень, що зменшує вплив суб'єктивного фактора при ідентифікації складних елементів. Однак таке рішення потребує високоякісного спеціалізованого обладнання та складного калібрування сенсорів навколо кожного приладу. Через закритий комерційний статус і високу ціну такі системи доступні переважно для змагань найвищого рівня, залишаючись недосяжними для масового спорту та повсякденної підготовки атлетів в Україні. На рисунку 1.2 наведено вигляд інтерфейсу такої системи на базі лазерного сканування.



Рисунок 1.2 – Інтерфейс системи 3D-аналізу на базі лазерних сенсорів

1.2.3 Програмні засоби для мануального відеоаналізу

У тренувальній практиці часто застосовується спеціалізоване програмне забезпечення для відеоаналізу техніки, наприклад, Dartfish або Kinovea [4], [5]. Ці інструменти дозволяють тренерам вручну накладати графічні маркери на відео, вимірювати траєкторії руху спортсмена та порівнювати два виконання в одному кадрі.

Хоча ці засоби є корисними для методичної роботи, вони не містять алгоритмів автоматичного розпізнавання елементів згідно з правилами FIG. Сегментація вправи та підрахунок балів у таких програмах потребують значного часу й участі експерта, тому їх складно використовувати для швидкого отримання D-score під час виконання комбінації.

На рисунку 1.3 наведено процес аналізу техніки рухів за допомогою графічних маркерів у спеціалізованому ПЗ.



Рисунок 1.3 – Програмне забезпечення для мануального відеоаналізу рухів гімнаста

1.2.4 Порівняльний аналіз існуючих рішень

Таким чином, на ринку та в науковій сфері спостерігається розрив між дорогими лазерними системами та доступними, але переважно ручними методами відеоаналізу. Системи відеоповторів допомагають суддям переглядати спірні моменти, але не виконують автоматичну ідентифікацію елементів. Лазерні комплекси забезпечують високий рівень точності, проте потребують спеціального обладнання та складного налаштування. Програмні засоби для тренувального аналізу є доступнішими, але потребують ручної роботи тренера або експерта.

Це свідчить про необхідність пошуку доступного програмного підходу до автоматизованого аналізу гімнастичних вправ. Такий підхід має поєднувати можливість обробки стандартного відеопотоку, автоматизоване виділення рухових ознак та подальшу класифікацію елементів без використання високовартісного спеціалізованого обладнання.

1.3 Проблематика існуючих методів оцінювання

Незважаючи на детальну регламентацію правил та наявність окремих технологічних рішень для підтримки суддівства, процес оцінювання у спортивній гімнастиці все ще має низку обмежень. Вони пов'язані не лише з особливостями людського сприйняття, а й з недостатнім рівнем автоматизації доступних систем, високою вартістю спеціалізованих апаратних комплексів та складністю аналізу швидких динамічних рухів у реальному часі.

Однією з головних проблем традиційного суддівства залишається залежність від людського фактору. Хоча досвідчені судді володіють необхідними навичками для ідентифікації гімнастичних елементів, висока інтенсивність сучасних вправ значно підвищує ймовірність помилки. За короткий проміжок часу суддя має не лише впізнати складний рух, а й

оцінити його технічні параметри: положення тіла, амплітуду, кут відхилення, момент початку й завершення елемента. У такому темпі навіть кваліфікований фахівець може помилитися через необхідність одночасного контролю багатьох деталей. На рисунку 1.4 наведено схему ідентифікації критичного кута відхилення.



Рисунок 1.4 – Схема ідентифікації критичного кута відхилення

Окрім складності візуальної фіксації рухів, вагомим чинником є значне когнітивне навантаження на експертів. Суддівство великих турнірів зазвичай триває протягом багатьох годин, що неминуче викликає втому та зниження концентрації уваги. Це створює умови для появи випадкових помилок, особливо наприкінці змагального дня. Крім того, на оцінку може підсвідомо впливати суб'єктивний фактор, такий як авторитет атлета або його попередня репутація, що ускладнює повну реалізацію принципу об'єктивності та рівності учасників.

Окремою проблемою, яка є критичною для української спортивної практики, є необхідність роботи в режимі багатозадачності. Коли один спеціаліст змушений одночасно фіксувати технічний зміст програми для розрахунку D-score та для виведення E-score відстежувати дрібні технічні

огріхи, імовірність похибки зростає в кілька разів. Людина просто не може рівномірно розподілити увагу між ідентифікацією складних рухів у просторі та фіксацією чистоти їхнього виконання, що нерідко призводить до пропуску окремих елементів або неправильного трактування їхньої вартості.

Важливо також враховувати проблему обмеженого ракурсу спостереження. Суддя знаходиться у фіксованій точці, з якої окремі частини тіла гімнаста можуть бути перекриті приладом або самим спортсменом. Через це виникають так звані сліпі зони, у яких технічна помилка або важливий момент виконання залишаються непоміченими. Системи відеоповторів частково зменшують цю проблему, однак вони не усувають потребу в ручному аналізі, оскільки остаточне рішення все одно приймає суддя.

Наявні автоматизовані рішення також мають обмеження. Високоточні комплекси на основі 3D-сенсорів можуть забезпечувати детальний аналіз рухів [3], проте вони потребують високовартісного обладнання, складного налаштування та спеціальних умов використання. Програмні засоби для тренувального відеоаналізу є доступнішими, але здебільшого працюють як допоміжні інструменти для ручного розбору техніки й не забезпечують автоматичного розпізнавання гімнастичних елементів відповідно до правил FIG [1].

Отже, проблема оцінювання у спортивній гімнастиці полягає не лише у суб'єктивності традиційного суддівства, а й у відсутності доступного програмного інструменту, здатного автоматично аналізувати відеопотік, виділяти ключові параметри руху та класифікувати гімнастичні елементи для подальшого розрахунку D-score.

Це зумовлює потребу у розробці підходу, який поєднає доступність звичайного відеоаналізу з можливостями інтелектуальної обробки рухових даних.

1.4 Мета роботи та постановка задачі

З урахуванням особливостей оцінювання вправ у спортивній гімнастиці, а також обмежень існуючих методів суддівства та наявних технологічних рішень, у роботі постає задача розробки інтелектуальної системи для автоматизованого аналізу гімнастичних комбінацій за відеопотоком.

Основна ідея кваліфікаційної роботи полягає в автоматизованій ідентифікації гімнастичних елементів, визначенні часових меж та етапів виконання вправи, а також розрахунку підсумкової оцінки за складність, тобто D-score, згідно з міжнародними критеріями [1].

Метою даної роботи є розробка доступного та об'єктивного програмного рішення, яке дозволило б зменшити вплив людського фактору у спортивному суддівстві шляхом переходу від виключно візуального спостереження до автоматизованого аналізу рухів гімнаста.

Обґрунтування обраного підходу полягає в тому, що використання високовартісних апаратних комплексів, зокрема систем тривимірного лазерного сканування та активних 3D-сенсорів, є малодоступним для масової спортивної практики. Тому доцільним є застосування технологій, що базуються на аналізі стандартного відеопотоку без необхідності використання спеціалізованого обладнання.

Для цього передбачається виділення скелетної моделі тіла гімнаста, подальша сегментація вправи на окремі часові сегменти та класифікація елементів за послідовностями рухів.

Такий підхід дозволяє поєднати доступність звичайного відеоаналізу з можливостями інтелектуальної обробки рухових даних. На відміну від систем відеоповторів, які лише допомагають судді переглядати виступ, запропонований підхід передбачає автоматизоване виділення рухових ознак і подальше використання цих даних для розрахунку D-score.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- проаналізувати актуальні правила Міжнародної федерації гімнастики (FIG) [1] та виділити технічні критерії для класифікації елементів за групами складності та їхньою базовою вартістю;
- сформувати спеціалізований анотований набір даних, що містить послідовності координат ключових точок тіла гімнаста, а також параметри складності та груп елементів;
- налаштувати алгоритм комп'ютерного зору для стабільного виділення скелетної моделі тіла гімнаста за відеопотоком;
- розробити логіку темпоральної сегментації вправи для визначення часових меж окремих елементів;
- побудувати та навчити класифікаційну модель для ідентифікації гімнастичних елементів у виділених часових сегментах;
- розробити алгоритм автоматичного підрахунку D-score, здатний обирати найбільш складні елементи комбінації та застосовувати до них відповідні коефіцієнти згідно з правилами FIG;
- створити інтерфейс користувача для візуалізації результатів обробки відео, ідентифікованих елементів та розрахованої оцінки за складність.

Таким чином, основним завданням роботи є створення комплексного підходу до автоматизованого аналізу гімнастичних вправ, який поєднує обробку відеопотоку, виділення рухових ознак, сегментацію вправи, класифікацію елементів та розрахунок D-score.

2 МАТЕМАТИЧНІ ОСНОВИ ТА АРХІТЕКТУРА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

2.1 Аналіз сучасних наукових підходів до розпізнавання дій на основі скелетних даних

Розпізнавання дій людини за відеоданими є одним із важливих напрямів комп'ютерного зору та машинного навчання. Традиційні підходи до аналізу відео базувалися переважно на обробці послідовності зображень, де модель повинна була самостійно виділяти корисні ознаки з пікселів кадру. Однак для задач, пов'язаних з аналізом рухів людини, більш ефективним підходом є використання скелетних даних, тобто координат ключових точок тіла. У такому випадку дія людини подається не як набір зображень, а як часова послідовність положень суглобів. Це дозволяє зменшити вплив фону, освітлення, кольору одягу та інших другорядних факторів, зосередившись саме на динаміці руху [6], [7].

Напрямок розпізнавання дій на основі скелетних даних відомий як *Skeleton-based Human Action Recognition*. У межах цього напрямку відеозапис спочатку перетворюється на послідовність скелетних поз, після чого модель аналізує зміну положення ключових точок у часі [6], [7]. Такий підхід особливо актуальний для спортивної гімнастики, оскільки гімнастичні елементи відрізняються не лише кінцевою позою, а й характером переходу між положеннями тіла, амплітудою руху, напрямом обертання та часовою структурою виконання. Тому для класифікації елементів важливо враховувати не окремий кадр, а повну послідовність руху.

Одним із сучасних напрямів у цій галузі є використання просторово-часових графових згорткових мереж, або ST-GCN [6]. У таких моделях тіло людини розглядається як граф, де вершинами є ключові точки скелета, а ребрами – анатомічні зв'язки між суглобами. Це дозволяє моделі одночасно

враховувати просторову структуру тіла та часову зміну положень у послідовності кадрів. Перевагою ST-GCN є здатність ефективно аналізувати складні взаємозв'язки між різними частинами тіла, що робить ці моделі досить потужними для задач розпізнавання дій. Проте такі архітектури зазвичай потребують значних обчислювальних ресурсів, великих анотованих наборів даних і ретельного налаштування параметрів, що може бути проблемою для прикладних систем у спортивній практиці.

Іншим поширеним підходом є поєднання згорткових і рекурентних нейронних мереж, зокрема архітектур типу CNN + LSTM. У такому випадку згорткова частина використовується для виділення просторових ознак із кадрів або скелетних представлень, а рекурентна частина аналізує їхню зміну в часі. LSTM-мережі добре підходять для роботи з послідовностями, оскільки здатні враховувати попередній контекст і зберігати інформацію про важливі етапи руху [8]. Такий підхід є ефективним для багатьох задач розпізнавання дій, однак він також може потребувати значних обчислювальних ресурсів, особливо якщо на вхід подаються повні зображення або великі багатовимірні ознаки.

Окрему групу становлять рекурентні архітектури, які працюють безпосередньо з послідовностями координат ключових точок. До них належать RNN, LSTM та GRU-моделі [8], [9], [10]. Їхньою перевагою є природна придатність до аналізу часових рядів, оскільки кожен гімнастичний елемент можна подати як послідовність скелетних поз. У порівнянні з LSTM, GRU має простішу внутрішню структуру та меншу кількість параметрів, що дає змогу зменшити обчислювальні витрати й прискорити навчання моделі. Для задачі класифікації гімнастичних елементів це є важливою перевагою, оскільки система має працювати з обмеженим набором навчальних даних і бути придатною для практичного використання без складної апаратної інфраструктури.

Таким чином, сучасні підходи до розпізнавання дій на основі скелетних даних можна поділити на кілька основних груп: графові

просторово-часові моделі, комбіновані CNN + LSTM архітектури та рекурентні моделі для аналізу послідовностей координат. Найбільш потужні методи, такі як ST-GCN, забезпечують глибокий аналіз структури руху, однак потребують великих обсягів даних і значних обчислювальних ресурсів. У межах даної роботи доцільним є використання більш компактного підходу, орієнтованого на аналіз скелетних послідовностей за допомогою рекурентної моделі, оскільки він краще відповідає умовам обмеженого набору даних, прикладному характеру задачі та потребі у доступній реалізації системи аналізу відео.

2.2 Науково-технічне обґрунтування гібридної архітектури системи

Після аналізу сучасних підходів до розпізнавання дій на основі скелетних даних можна зробити висновок, що найбільш потужні архітектури, зокрема ST-GCN та CNN-LSTM [6], [8], мають високий потенціал для задач класифікації рухів людини. Вони дозволяють враховувати як просторову структуру тіла, так і часову динаміку виконання руху. Проте для задачі автоматизованого аналізу гімнастичних вправ важливо враховувати не лише точність моделі, а й практичні умови її використання: обмежений обсяг навчального набору даних, необхідність роботи зі звичайним потоком відеоданих, відсутність спеціалізованого обладнання та бажану можливість запуску системи на стандартних обчислювальних ресурсах.

Використання просторово-часових мереж ST-GCN є доцільним у задачах, де доступні великі анотовані набори даних і потужні графічні процесори для навчання моделей. Такі архітектури добре працюють із графовим поданням скелета людини [6], однак їх застосування потребує значних обчислювальних ресурсів і великої кількості якісно розмічених прикладів. У межах цієї роботи така умова є обмеженням, оскільки формування великого набору даних для спортивної гімнастики є складним

процесом. Кожен приклад потребує точного виділення елемента, його коректного підпису, визначення часових меж та відповідності правилам FIG [1]. Тому повністю залежати від складної моделі, яка потребує великої кількості навчальних прикладів, у цьому випадку недоцільно.

Архітектури типу CNN-LSTM також можуть застосовуватися для розпізнавання дій у відео, оскільки вони поєднують обробку просторових ознак і аналіз часової послідовності. Однак у випадку аналізу повних відеокадрів такий підхід може бути досить вимогливим до ресурсів. Крім того, гімнастичні елементи часто відрізняються не фоном або зовнішнім виглядом кадру, а саме характером руху тіла, положенням суглобів, амплітудою та послідовністю фаз. Через це більш доцільним є перехід від аналізу повного зображення до аналізу скелетної моделі [7], [11], [12], яка містить ключову інформацію про рух гімнаста.

У межах цієї роботи обрано гібридну архітектуру, яка поєднує MediaPipe Pose для екстракції скелетної моделі [11], [12], детермінований автомат станів для сегментації вправи та двонапрямну рекурентну мережу Bi-GRU для класифікації виділених елементів [9], [10]. Такий підхід дозволяє розділити задачу на кілька логічних етапів: спочатку з відео виділяються координати ключових точок тіла, потім безперервна комбінація поділяється на окремі рухові сегменти, після чого кожен сегмент класифікується моделлю, орієнтованою на аналіз часових послідовностей. Окремі положення цього підходу були представлені у науковій роботі, присвяченій інтелектуальному аналізу складності гімнастичних комбінацій засобами комп'ютерного зору [13].

Використання MediaPipe Pose дозволяє отримати формалізований опис положення тіла гімнаста у вигляді координат ключових точок. Завдяки цьому система працює не з усім зображенням, а з компактним набором числових ознак, які описують рух. Це зменшує залежність від фону, освітлення, кольору форми спортсмена та інших другорядних факторів. Крім того, такий підхід знижує обсяг даних, які потрібно обробляти на

наступних етапах, що є важливим для забезпечення швидкої роботи системи.

Детермінований автомат станів використовується для сегментації гімнастичної вправи на окремі елементи. Його перевага полягає в інтерпретованому описі переходів між характерними станами тіла гімнаста, що дозволяє пояснити визначення початку або завершення певного елемента. Це важливо для спортивної гімнастики, оскільки суддівська логіка також базується на визначенні ключових положень, фаз руху та моментів переходу між ними.

Для класифікації виділених сегментів обрано двонапрямну рекурентну нейронну мережу Bi-GRU. Рекурентні мережі добре підходять для аналізу часових послідовностей, оскільки враховують порядок зміни даних у часі. У задачі розпізнавання гімнастичних елементів це є важливим, оскільки значення руху залежить не лише від окремих положень тіла, а й від послідовності їх виконання. Двонапрямність Bi-GRU дозволяє аналізувати рух з урахуванням як попередніх, так і наступних положень тіла, що підвищує якість розпізнавання динамічних елементів.

Порівняно з LSTM, модель GRU має простішу структуру та меншу кількість параметрів [9]. Це робить її придатною для умов обмеженого навчального набору даних і потреби у швидкій роботі системи. У межах цієї роботи Bi-GRU є компромісом між здатністю аналізувати часову динаміку руху та вимогами до обчислювальної ефективності.

З науково-технічної точки зору, вибір гібридної архітектури MediaPipe Pose + FSM + Bi-GRU є обґрунтованим. На відміну від складних моделей, які потребують великих обсягів даних і потужних графічних процесорів, запропонований підхід орієнтований на доступну реалізацію та інтерпретовану обробку рухів. Поєднання скелетної моделі, автомата станів і рекурентної мережі забезпечує зв'язок між сегментацією, класифікацією елементів і розрахунком D-score.

2.3 Математична формалізація задачі інтелектуального оцінювання

Для подальшого проєктування системи необхідно формально описати задачу автоматизованого оцінювання гімнастичної вправи. У межах даної роботи відеозапис виступу розглядається як впорядкована послідовність кадрів, що надходять на вхід системи та містять інформацію про положення тіла гімнаста у часі.

Нехай вхідний відеопотік гімнастичної комбінації задається у вигляді:

$$V = \{f_1, f_2, \dots, f_N\}, \quad (2.1)$$

де f_t – окремий кадр відеопотоку в момент часу t ;

N – загальна кількість кадрів у відеозаписі.

Задача системи полягає у побудові оператора відображення:

$$F: V \rightarrow D, \quad (2.2)$$

де D – підсумкова оцінка складності вправи.

D -score формується на основі розпізнаних гімнастичних елементів, їхньої базової вартості та належності до відповідних груп складності [1].

Оскільки безпосередній аналіз повного зображення є надмірним для поставленої задачі, відеопотік спочатку перетворюється на послідовність скелетних моделей тіла гімнаста. Для кожного кадру f_t формується набір ключових точок:

$$P_t = \{p_1^t, p_2^t, \dots, p_{33}^t\}, \quad (2.3)$$

де p_i^t – ключова i -та точка тіла у кадрі t ;

x_i^t, y_i^t, z_i^t – координати i -ї ключової точки тіла у кадрі t ;

v_i^t – показник видимості i -ї ключової точки.

Таким чином, замість повного зображення система аналізує послідовність положень суглобів і частин тіла гімнаста:

$$P = \{P_1, P_2, \dots, P_N\}. \quad (2.4)$$

На наступному етапі задача зводиться до поділу послідовності P на окремі часові сегменти, які відповідають потенційним гімнастичним елементам:

$$S = \{s_1, s_2, \dots, s_M\}, \quad (2.5)$$

де $s_j = [t_{\text{поч}}, t_{\text{кін}}]$ – часовий інтервал j -го елемента;

$t_{\text{поч}}$ – кадр початку елемента;

$t_{\text{кін}}$ – кадр його завершення;

M – кількість виділених сегментів.

Такий поділ є необхідним, оскільки гімнастична комбінація є безперервною послідовністю рухів, а для розрахунку D-score потрібно працювати не з усім відео одразу, а з окремими завершеними елементами.

Для кожного виділеного сегмента виконується класифікація. Класифікатор отримує на вхід послідовність скелетних координат у межах цього сегмента та визначає клас відповідного гімнастичного елемента. Оскільки один руховий сегмент може містити складну композицію (наприклад, перехід та безпосередньо сам елемент), оператор класифікації може повертати підмножину розпізнаних елементів:

$$C(s_j) = \{e_{j1}, e_{j2}, \dots, e_{jk_j}\}, \quad (2.6)$$

де e_{ji} – i -й розпізнаний базовий елемент у межах j -го часового сегмента;

k_j – кількість базових елементів, розпізнаних у межах j -го сегмента.

Крім назви елемента, для подальшого розрахунку використовується довідкова інформація про його базову складність та групу. У загальному вигляді кожному розпізнаному базовому елементу ставиться у відповідність набір параметрів:

$$e_{ji} \rightarrow (d_{ji}, g_{ji}), \quad (2.7)$$

де d_{ji} – базова вартість елемента;

g_{ji} – група елемента.

Після класифікації всіх сегментів формується множина розпізнаних елементів:

$$E = \{e_1, e_2, \dots, e_Q\}, \quad (2.8)$$

де Q – загальна кількість розпізнаних базових елементів після обробки всіх сегментів.

На її основі виконується розрахунок підсумкової оцінки складності D -score. У межах розроблюваної системи ця оцінка складається з базової складності розпізнаних елементів та додаткових складових, пов'язаних із виконанням спеціальних вимог до груп елементів [1]:

$$D = D_{\text{елем}} + D_{\text{спец}}, \quad (2.9)$$

де $D_{\text{елем}}$ – сума базової вартості врахованих елементів;

$D_{\text{спец}}$ – додаткові бали за виконання спеціальних вимог.

Таким чином, загальний оператор оцінювання можна подати як послідовність перетворень:

$$V \rightarrow P \rightarrow S \rightarrow E \rightarrow D. \quad (2.10)$$

Це означає, що вхідне відео спочатку перетворюється на послідовність скелетних координат, потім поділяється на окремі рухові сегменти, після чого кожен сегмент класифікується як певний гімнастичний елемент, а отримані результати використовуються для розрахунку підсумкового D-score.

Під час формалізації задачі також необхідно враховувати можливі похибки, які виникають на різних етапах обробки. До них належать неточне визначення координат ключових точок, часткове перекриття тіла гімнаста приладом або самим спортсменом, помилки сегментації та неправильна класифікація елементів. Тому задача системи полягає не лише у побудові оператора $F: V \rightarrow D$, а й у мінімізації похибки розпізнавання елементів та забезпеченні стійкості до геометричних оклюзій.

Отже, математично задача інтелектуального оцінювання гімнастичної вправи полягає у побудові такого оператора F , який за вхідним відеопотоком гімнастичної комбінації визначає послідовність виконаних елементів і формує підсумкову оцінку складності D-score. Така формалізація дозволяє розглядати подальші модулі системи як окремі етапи єдиного процесу: екстракцію скелетної моделі, просторову нормалізацію та часове приведення послідовностей, сегментацію рухів, класифікацію елементів та розрахунок оцінки складності.

2.4 Попередня обробка та просторово-часова нормалізація кінематичних ознак

У розробленій системі обробка кінематичних координат виконується не через прогнозування положення точок, а через фільтрацію ненадійних значень, геометричну нормалізацію скелетної моделі та приведення часових послідовностей до єдиної довжини. Такий підхід відповідає поточній реалізації системи та дозволяє підготувати дані MediaPipe Pose до подальшої сегментації й класифікації гімнастичних елементів.

Після обробки відеопотоку алгоритм MediaPipe Pose формує для кожного кадру набір ключових точок тіла гімнаста [11], [12]. Кожна точка описується тривимірними координатами:

$$p_i = (x_i, y_i, z_i), \quad (2.11)$$

де i – номер ключової точки скелетної моделі.

Окрім координат, для кожної точки визначається показник видимості v_i , який характеризує надійність її виявлення у поточному кадрі [11], [12]. Для зменшення впливу помилково визначених або частково перекритих точок у системі використовується порогова фільтрація за показником видимості. Точка вважається надійною лише тоді, коли виконується умова:

$$v_i \geq v_{min}, \quad (2.12)$$

де $v_{min} = 0,2$ мінімально допустиме значення показника видимості.

Якщо значення видимості точки є нижчим за заданий поріг, така точка не використовується під час перевірки геометричних умов і визначення станів тіла. У випадках, коли поза гімнаста не була виявлена на кадрі, координати та показники видимості заповнюються нульовими значеннями. Це дозволяє зберігати однакову структуру даних для всіх кадрів послідовності кадрів.

Наступним етапом є геометрична нормалізація скелетної моделі [7], [10]. Вона необхідна для того, щоб зменшити вплив положення гімнаста в кадрі, масштабу зображення та відстані до камери, оскільки ці фактори можуть змінювати числові значення координат навіть для однакових рухів. Для цього координати ключових точок центруються відносно середньої точки таза, яка використовується як умовний центр тіла. Координати центра таза визначаються як середнє значення координат лівого та правого тазостегнових суглобів:

$$c_t = \frac{p_{\{23,t\}} + p_{\{24,t\}}}{2}, \quad (2.13)$$

де $p_{23,t}$ та $p_{24,t}$ – координати лівого та правого тазостегнових суглобів у момент часу t .

Після цього визначається масштабний коефіцієнт, який дорівнює евклідовій відстані між центром плечей та центром таза:

$$s_t = \|c_{\{shoulders,t\}} - c_t\|, \quad (2.14)$$

де $c_{shoulders,t}$ – середня точка між лівим і правим плечем у момент часу t .

Нормалізовані координати кожної ключової точки обчислюються за формулою:

$$\hat{p}_{i,t} = \frac{p_{i,t} - c_t}{s_t}. \quad (2.15)$$

Завдяки такому перетворенню система аналізує не абсолютне положення гімнаста у кадрі, а відносну конфігурацію його тіла. Це важливо для класифікації елементів, оскільки один і той самий рух може бути виконаний у різних частинах кадру або з різною відстанню до камери.

Оскільки виділені гімнастичні елементи можуть мати різну тривалість, перед поданням даних на вхід нейронної мережі виконується часове узгодження послідовностей. Для цього кожна послідовність координат приводиться до фіксованої довжини L . Це необхідно для того, щоб усі приклади мали однаковий формат і могли правильно оброблятися однією моделлю класифікації. Крім того, таке перетворення дозволяє зберегти загальну динаміку руху незалежно від того, скільки кадрів тривав конкретний елемент у відеозаписі. У системі використовується лінійна інтерполяція, яка дозволяє перетворити послідовність довільної довжини на послідовність із наперед заданою кількістю часових кроків [10].

Нехай початкова послідовність має вигляд:

$$X = (x_1, x_2, \dots, x_T), \quad (2.16)$$

де T – початкова кількість кадрів у сегменті.

Після часового узгодження формується нова послідовність:

$$\hat{X} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_L), \quad (2.17)$$

де L – фіксована довжина вхідної послідовності для класифікаційної моделі.

Таке перетворення дозволяє подавати на вхід Bi-GRU послідовності однакового розміру незалежно від фактичної тривалості виконання елемента. Крім того, інтерполяція частково зменшує вплив нерівномірності руху та різної кількості кадрів у сегментах відео.

Додатково у системі передбачено стабілізацію сегментації руху. Вона базується на аналізі зміни положення ключових точок між сусідніми кадрами. Для цього може бути використане середнє зміщення вибраних точок тіла:

$$m_t = \frac{1}{K} \sum_{i=1}^K \|p_{i,t} - p_{i,t-1}\|, \quad (2.18)$$

де K – кількість ключових точок, які використовуються для аналізу руху.

Якщо зміна положення точок є занадто малою або, навпаки, не відповідає очікуваній логіці руху, це враховується під час визначення меж елемента. Для уникнення випадкового спрацювання алгоритму через окремі помилкові кадри використовуються лічильники послідовного підтвердження станів. Стан вважається зафіксованим лише тоді, коли відповідні геометричні умови виконуються протягом кількох кадрів поспіль.

Окремо виконується уточнення меж виділеного сегмента за активністю руху. Якщо у знайденому сегменті є ділянки з недостатнім рухом, вони можуть бути відкинуті, а межі сегмента зміщуються ближче до активної частини елемента. Це дозволяє передавати на класифікацію не весь проміжок між двома станами, а саме ту частину відеопослідовності, яка містить основний рух гімнастичного елемента.

Таким чином, математична обробка кінематичних координат у розробленій системі включає фільтрацію точок за показником видимості, геометричну нормалізацію скелетної моделі, часове узгодження послідовностей та стабілізацію сегментації за параметрами руху. Сукупність цих етапів дозволяє підготувати координати MediaPipe Pose до подальшого аналізу та зменшити вплив масштабу, положення спортсмена в кадрі, короткочасних втрат точок і різної тривалості виконання елементів.

2.5 Алгоритмічна сегментація відеопотоку на основі детермінованих автоматів станів

Після екстракції, фільтрації та нормалізації скелетних координат наступним етапом роботи системи є поділ безперервної гімнастичної комбінації на окремі рухові сегменти. Це завдання є важливим, оскільки класифікаційна модель повинна отримувати на вхід не весь відеозапис одразу, а окремі сегменти, які відповідають потенційним гімнастичним елементам. Для цього у системі використовується алгоритмічна сегментація на основі детермінованого автомата станів.

Детермінований автомат станів дає змогу описати вправу як послідовність логічних станів тіла гімнаста та переходів між ними. У межах розробленої системи основна ідея полягає в тому, що окремий гімнастичний елемент розглядається як руховий сегмент між відносно стабільними положеннями. Тому система аналізує не лише координати ключових точок

у кожному кадрі, а й зміну цих координат у часі, швидкість руху та виконання геометричних умов для характерних положень тіла.

У системі виділяються три основні опорні положення, які використовуються для визначення меж рухових сегментів: Handstand, Angle та Feet-stand. Вони розглядаються як стабільні стани тіла гімнаста, поява або зникнення яких керує переходами детермінованого автомата між режимом стабільного положення та активною фазою руху. Саме ці положення виконують роль контрольних точок, за якими система визначає початок або завершення потенційного елемента. Стан Handstand відповідає положенню тіла у стійці на руках, у якому тіло гімнаста має вертикальну орієнтацію, а гомілковостопні суглоби розташовані вище плечей. Стан Angle описує положення «кут», для якого характерне зігнуте положення тіла з винесенням ніг уперед. Стан Feet-stand відповідає положенню стоячи на ногах, яке може використовуватися для фіксації завершення комбінації або окремого рухового сегмента.

Візуальні приклади стабільних станів тіла гімнаста, які використовуються в алгоритмі сегментації, наведено на рисунку 2.1.



Рисунок 2.1 – Приклади стабільних станів тіла гімнаста для сегментації руху

Для визначення станів використовуються координати ключових точок скелетної моделі та показники їхньої видимості. У програмній реалізації мінімальний поріг видимості ключової точки становить 0,2. Якщо видимість певної точки є нижчою від цього порога, така точка не враховується у відповідному геометричному критерії. Це дає змогу зменшити вплив часткових перекриттів тіла гімнаста приладом або неточного визначення окремих точок у кадрі.

Одним із базових геометричних критеріїв є розрахунок кута між трьома точками скелетної моделі. Для точок a , b та c кут у точці b визначається через скалярний добуток векторів:

$$\alpha = \arccos \frac{(a - b) \cdot (c - b)}{\|a - b\| \|c - b\|}, \quad (2.19)$$

де α – кут між відповідними частинами тіла;

a, b, c – координати ключових точок скелетної моделі;

$\|\cdot\|$ – норма вектора.

У програмній реалізації цей критерій використовується для перевірки положень у тазостегнових, колінних та інших суглобах. Наприклад, для стану Angle важливими є кут у тазостегновому суглобі, кут у коліні, положення плечей відносно таза та напрям розташування ніг. Для стану Handstand перевіряється вертикальність тулуба, положення гомілковостопних суглобів відносно плечей, а також орієнтація тіла відносно камери. Для стану Feet-stand аналізується положення плечей, таза, колін і гомілковостопних суглобів відносно оціненої лінії опори.

Перехід у стан Handstand, Angle або Feet-stand відбувається лише тоді, коли відповідні кутові та позиційні умови виконуються протягом заданої кількості послідовних кадрів. Це дає змогу уникнути випадкового переходу між станами через одиничні помилки визначення координат або короточасні втрати окремих ключових точок.

Крім геометричних умов, у системі використовується критерій швидкості руху ключових точок. Він потрібний для того, щоб відрізнити стабільні положення тіла від активної фази виконання елемента. Для оцінювання інтенсивності руху використовується середнє міжкадрове зміщення вибраних ключових точок скелетної моделі, яке визначається за зміною їхніх координат між сусідніми кадрами:

$$v_t = \frac{1}{K} \sum_{i=1}^K \sqrt{(x_i^t - x_i^{t-1})^2 + (y_i^t - y_i^{t-1})^2} \cdot s \quad (2.20)$$

де v_t – середня швидкість руху ключових точок у кадрі t ;

K – кількість ключових точок, що використовуються для оцінювання руху;

x_i^t, y_i^t – координати i -ї ключової точки у кадрі t ;

s – масштабний коефіцієнт, пов'язаний з розміром кадру.

Якщо середня швидкість не перевищує заданий поріг, положення може розглядатися як достатньо стабільне для перевірки одного з опорних станів. Якщо ж стабільний стан зникає протягом певної кількості кадрів, система вважає, що гімнаст перейшов до активної фази руху, і починає формувати сегмент потенційного елемента.

Логіка сегментації будується на змінних, які відображають поточний стан системи: наявність стабільного положення, активність елемента та кількість послідовних кадрів, у яких виконується або не виконується певна умова. Для цього використовуються лічильники послідовного підтвердження станів Handstand, Angle і Feet-stand, а також лічильник кадрів, у яких жоден із характерних станів не розпізнається. Такий підхід дає змогу уникнути випадкового перемикання станів через короткочасний шум у координатах.

У загальному вигляді перехід до активного рухового сегмента можна описати так: якщо система перебувала у стабільному стані, але протягом

заданої кількості кадрів жоден із характерних станів не підтверджується, то починається сегмент елемента. Навпаки, якщо після активної фази знову виявляється стабільний стан, система завершує поточний сегмент. Отже, сегмент гімнастичного елемента можна подати як часовий інтервал:

$$s_j = [t_{start}^j, t_{end}^j], \quad (2.21)$$

де s_j – j -й виділений сегмент;

t_{start}^j – кадр початку сегмента;

t_{end}^j – кадр завершення сегмента.

Для запобігання появі надто коротких сегментів у системі використовується обмеження на мінімальну тривалість сегмента. У розробленому алгоритмі мінімальна тривалість елемента становить:

$$T_{min} = 0,5 \text{ с.} \quad (2.22)$$

Це означає, що сегмент враховується лише тоді, коли його тривалість не менша за встановлений часовий поріг. У кадрах це обмеження залежить від частоти відеозапису:

$$N_{min} = \max(35, ; \text{round}(T_{min} \cdot FPS)), \quad (2.23)$$

де N_{min} – мінімальна кількість кадрів у сегменті;

FPS – частота кадрів відеозапису;

T_{min} – мінімальна тривалість сегмента у секундах.

Значення «35» у формулі використовується як додаткове мінімальне обмеження на кількість кадрів сегмента, тому руховий сегмент приймається лише тоді, коли його тривалість не менша за встановлений часовий поріг і водночас він містить не менше 35 кадрів.

Додатково після попереднього виділення сегментів виконується уточнення їхніх меж за інтенсивністю руху. Для цього всередині знайденого сегмента аналізуються значення швидкості ключових точок. Якщо активна частина руху займає лише частину сегмента, межі можуть бути скориговані так, щоб краще відповідати реальному початку та завершенню елемента. Це дає змогу зменшити вплив зайвих підготовчих або завершальних кадрів на подальшу класифікацію та передавати моделі більш точну послідовність руху.

Загальну схему роботи детермінованого автомата станів, який використовується для виділення рухових сегментів, наведено на рисунку 2.2.

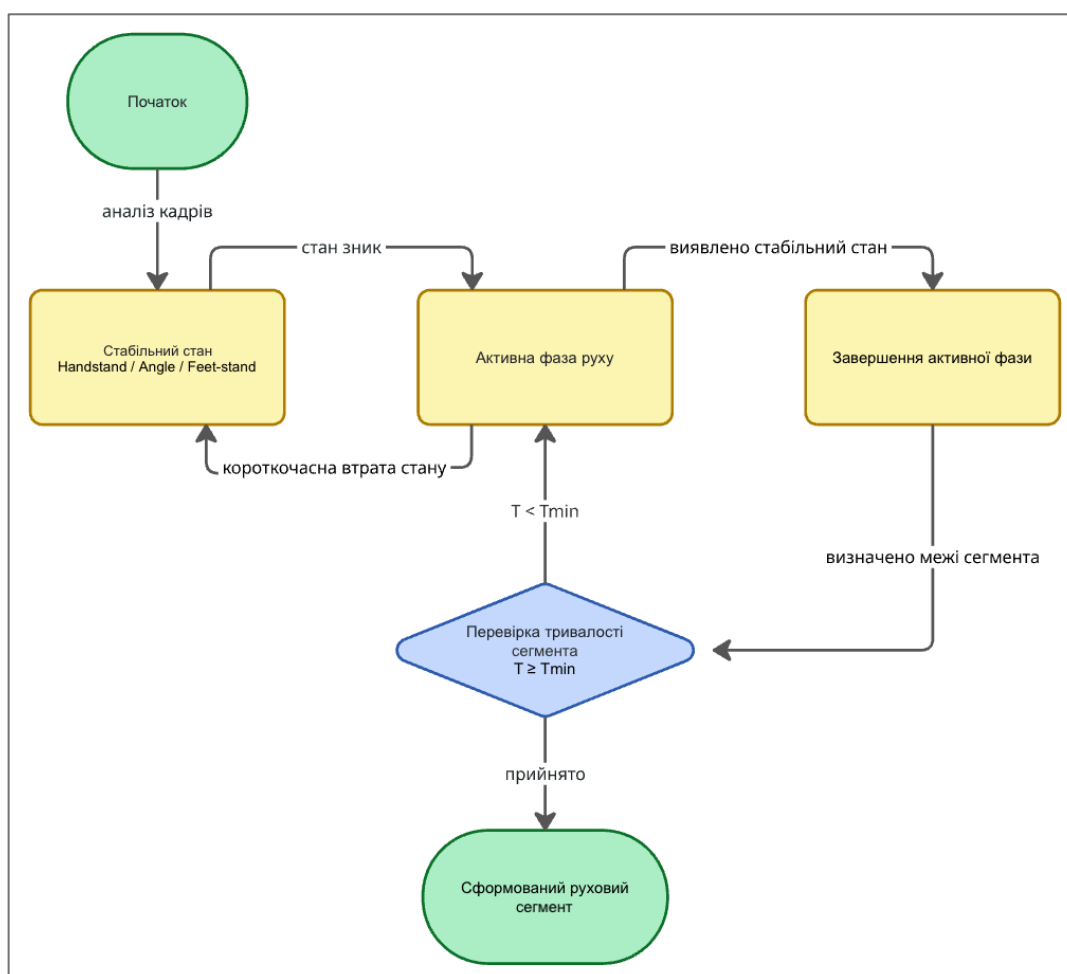


Рисунок 2.2 – Схема роботи детермінованого автомата станів для сегментації руху

Таким чином, алгоритмічна сегментація на основі детермінованого автомата станів забезпечує поділ безперервного відеопотоку на окремі логічно завершені рухові сегменти. Використання геометричних критеріїв, швидкості руху, лічильників послідовного підтвердження станів та мінімальної тривалості сегмента дає змогу підготувати дані для подальшої класифікації гімнастичних елементів.

2.6 Архітектура та математична модель двонапрямної рекурентної мережі Bi-GRU

Після виділення окремих рухових сегментів наступним етапом роботи системи є класифікація гімнастичних елементів. На цьому етапі кожен сегмент подається на вхід моделі, яка повинна визначити, якому гімнастичному елементу відповідає послідовність рухів. Оскільки гімнастичний елемент характеризується не лише окремим положенням тіла, а й динамікою зміни цих положень у часі, для розв'язання задачі класифікації доцільно використовувати рекурентну нейронну мережу [8], [9], [10].

У межах розробленої системи для класифікації гімнастичних елементів використовується двонапрямна рекурентна нейронна мережа з комірками GRU, тобто Bi-GRU [9], [10]. Така архітектура дає змогу аналізувати послідовність скелетних координат у двох часових напрямках: від початку сегмента до його завершення та у зворотному напрямі. Це є важливим для аналізу гімнастичних рухів, оскільки належність руху до певного класу часто визначається не лише початковою фазою, а й завершальним положенням тіла та характером переходу між ними.

На вхід класифікаційної моделі подається послідовність скелетних ознак фіксованої довжини. Для кожного часового кроку використовується набір координат 33 ключових точок тіла та показник видимості кожної точки. Оскільки для однієї ключової точки зберігаються координати x , y , z

та значення *visibility*, загальна кількість ознак на одному часовому кроці становить:

$$d = 33 \cdot 4 = 132. \quad (2.24)$$

Перед формуванням вхідного вектора координати ключових точок нормалізуються: за центр тіла приймається середина між лівим і правим тазостегновими суглобами, а масштабування виконується з використанням відстані між центром плечей і центром таза. Це дозволяє зменшити вплив положення спортсмена в кадрі та відносного масштабу фігури на результат класифікації.

Отже, вхідний сегмент можна подати у вигляді матриці:

$$X = (x_1, x_2, \dots, x_L), \quad x_t \in \mathbb{R}^{132}, \quad (2.25)$$

де X – вхідна послідовність ознак для одного рухового сегмента;

x_t – вектор ознак у момент часу t ;

L – фіксована довжина послідовності після часового узгодження;

132 – кількість ознак, що описують скелетну модель у кожному кадрі.

У поточній реалізації всі сегменти перед поданням до нейронної мережі приводяться до однакової довжини за допомогою лінійної інтерполяції. За замовчуванням, використовується довжина 64 послідовності L , тобто кожен руховий сегмент подається моделі як послідовність із 64 часових кроків. Це необхідно тому, що різні гімнастичні елементи можуть мати різну тривалість у відеозаписі, тоді як нейронна мережа повинна отримувати вхідні дані однакового розміру.

Комірка GRU є різновидом рекурентної нейронної мережі, яка використовує механізми керування потоком інформації [9].

На відміну від класичної RNN, GRU краще зберігає важливу інформацію з попередніх часових кроків і зменшує проблему зникнення

градієнта під час навчання [9], [10]. Водночас GRU має простішу структуру, ніж LSTM, оскільки використовує меншу кількість внутрішніх параметрів. Це робить її придатною для задач, у яких необхідно поєднати здатність аналізувати часову динаміку та достатню обчислювальну ефективність.

Теоретично роботу GRU-комірки можна описати через два основні механізми: вентиль оновлення та вентиль скидання. Вентиль оновлення визначає, наскільки сильно кандидат нового прихованого стану впливає на формування поточного стану, а вентиль скидання визначає, яку частину попередньої інформації слід враховувати під час формування нового стану. Для моменту часу t ці величини можна подати так:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \quad (2.26)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \quad (2.27)$$

де z_t – вентиль оновлення;

r_t – вентиль скидання;

x_t – вхідний вектор ознак у момент часу t ;

h_{t-1} – прихований стан на попередньому часовому кроці;

W_z, W_r, U_z, U_r – матриці ваг;

b_z, b_r – вектори зміщення;

σ – сигмоїдна функція активації.

Проміжний кандидат прихованого стану визначається з урахуванням вентиля скидання:

$$\tilde{h}_t = \tanh(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h), \quad (2.28)$$

де $\tilde{h}_t$ – кандидат нового прихованого стану;

$\odot$ – поелементне множення;

$\tanh$ – гіперболічний тангенс;

W_h, U_h – матриці ваг;

b_h – вектор зміщення.

Після цього поточний прихований стан формується як комбінація попереднього стану та нового кандидата:

$$h_t = (1 - z_t) \odot p_{t-1} + z_t \odot \tilde{h}_t. \quad (2.29)$$

У розробленій системі ці обчислення не програмуються вручну, а виконуються стандартним модулем `torch.nn.GRU` бібліотеки PyTorch [14], [15]. Тоді як програмна реалізація використовує готовий оптимізований механізм рекурентного шару, наведені формули пояснюють математичну основу роботи GRU-комірки.

Особливістю використаної моделі є двонапрямність. Це означає, що одна GRU-мережа обробляє послідовність у прямому напрямі, а інша – у зворотному. Для прямого напрямку приховані стани можна подати як:

$$\vec{h}_t = GRU_f(x_t, \vec{h}_{t-1}), \quad (2.30)$$

а для зворотного напрямку:

$$\overleftarrow{h}_t = GRU_b(x_t, \overleftarrow{h}_{t+1}), \quad (2.31)$$

де $\vec{h}_t$ – прихований стан прямого проходу;

$\overleftarrow{h}_t$ – прихований стан зворотного проходу;

GRU_f – рекурентний шар прямого напрямку;

GRU_b – рекурентний шар зворотного напрямку.

Після обробки послідовності результати прямого та зворотного напрямів об'єднуються в один вектор ознак:

$$h = [\vec{h}_L; \overleftarrow{h}_1], \quad (2.32)$$

де h – узагальнений вектор ознак рухового сегмента;

$\vec{h}_L$ – останній прихований стан прямого напрямку;

$\overleftarrow{h}_1$ – останній прихований стан зворотного напрямку;

$[\cdot; \cdot]$ – операція конкатенації векторів.

У програмній реалізації це відповідає об'єднанню останніх прихованих станів прямого та зворотного потоків. Отриманий вектор передається до повнозв'язного шару, який формує вихідні значення для кожного класу гімнастичного елемента:

$$o = W_o h + b_o, \quad (2.33)$$

де o – вектор вихідних значень моделі;

W_o – матриця ваг вихідного шару;

b_o – вектор зміщення;

h – об'єднаний вектор прихованих станів Bi-GRU.

Кількість вихідних нейронів відповідає кількості класів гімнастичних елементів, що використовуються у навчальному наборі даних. Під час інференсу до вихідного вектора застосовується функція softmax, яка перетворює значення моделі на ймовірності належності сегмента до кожного класу:

$$p_i = \frac{e^{o_i}}{\sum_{k=1}^C e^{o_k}}, \quad (2.34)$$

де p_i – ймовірність належності сегмента до i -го класу;

o_i – вихідне значення моделі для i -го класу;

C – загальна кількість класів.

Класом розпізнаного елемента вважається той, для якого модель отримала найбільше значення ймовірності:

$$\hat{y} = \arg \max_i p_i. \quad (2.35)$$

Під час навчання моделі використовується Cross-Entropy – функція втрат, яка є стандартною для задач багатокласової класифікації [10], [16]. У програмній реалізації вона застосовується до вихідних значень моделі без явного попереднього застосування softmax, оскільки відповідна операція враховується всередині функції CrossEntropyLoss [16]. Для пакета навчальних прикладів функцію втрат можна подати так:

$$\mathcal{L} = -\frac{1}{B} \sum_{j=1}^B \sum_{i=1}^C y_{ji} \log(\hat{y}_{ji}), \quad (2.36)$$

де B – розмір пакета навчальних прикладів;

C – кількість класів гімнастичних елементів;

y_{ji} – істинна мітка i -го класу для j -го прикладу;

$\hat{y}_{ji}$ – прогнозована ймовірність належності j -го прикладу до i -го класу.

У поточній реалізації під час навчання за замовчуванням використовуються ваги класів, що дозволяє частково компенсувати нерівномірність кількості прикладів різних гімнастичних елементів у навчальному наборі. За потреби цей механізм може бути вимкнений під час запуску процесу навчання моделі. Це є важливим для прикладної задачі, оскільки окремі елементи можуть бути представлені меншою кількістю навчальних прикладів, ніж інші.

Під час валідації та тестування якості роботи моделі оцінюється без урахування ваг класів, що дає змогу отримати більш нейтральну оцінку результатів класифікації.

Схему роботи двонапрямної рекурентної мережі Bi-GRU наведено на рисунку 2.6.

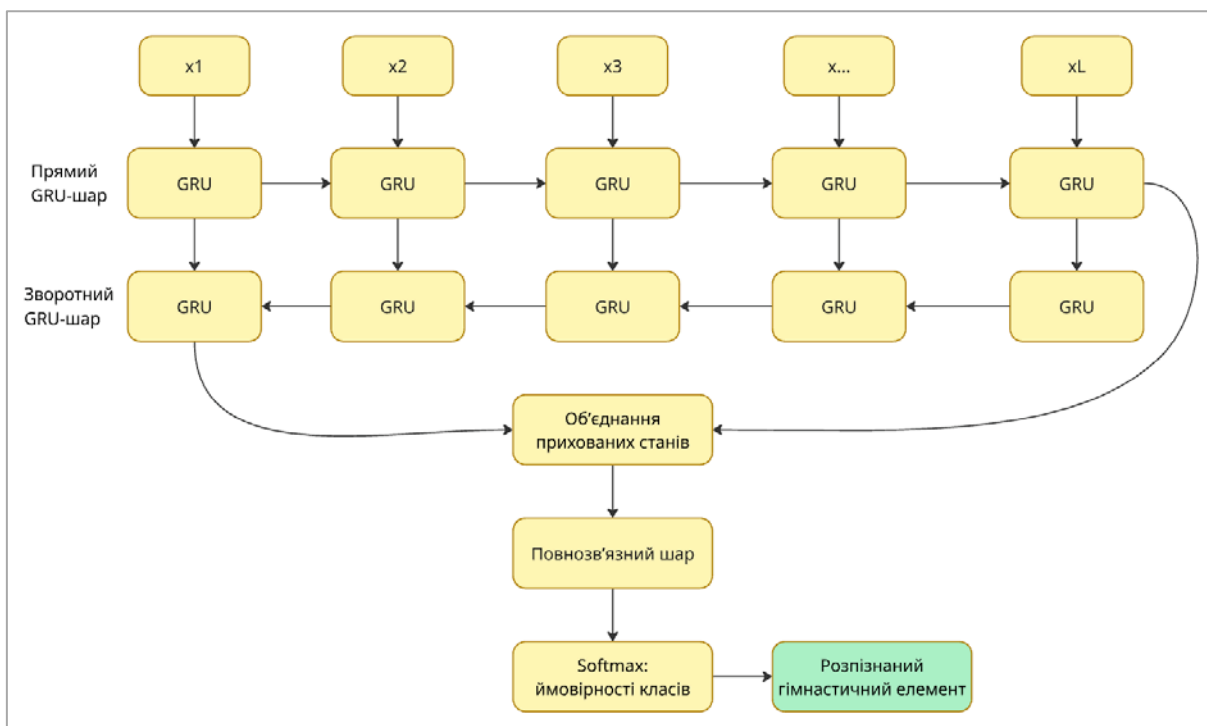


Рисунок 2.6 – Розгорнута в часі схема двонапрямної рекурентної мережі Bi-GRU

Таким чином, використання Bi-GRU дозволяє класифікувати гімнастичні елементи на основі часової динаміки скелетних координат. Двонапрямна обробка послідовності дає змогу враховувати як початкові, так і завершальні фази руху, а використання GRU-комірок забезпечує баланс між здатністю аналізувати часові залежності та обчислювальною ефективністю. У структурі розробленого програмного комплексу зазначена модель функціонує як базовий класифікаційний модуль, що забезпечує відображення ізольованих рухових сегментів у множину дискретних класів гімнастичних елементів. Отримані на цьому етапі предикативні мітки є константним базисом для подальшого алгоритмічного обчислення оцінки складності комбінації (D-score).

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛІВ АНАЛІЗУ ТА КЛАСИФІКАЦІЇ

3.1 Використані технології та програмні засоби

Програмна реалізація розробленої системи виконана з використанням засобів комп'ютерного зору, машинного навчання, обробки числових даних та створення графічного інтерфейсу користувача. Основною метою програмної частини є побудова послідовного процесу обробки гімнастичної вправи: від завантаження відеофайлу до отримання скелетної моделі спортсмена, сегментації руху, класифікації гімнастичних елементів і подальшого розрахунку оцінки складності за закладеними у системі правилами, наближеної до D-score. Розрахунок оцінки виконується на основі складності розпізнаних елементів, їхніх груп і реалізованих у системі правил нарахування додаткових балів.

Основною мовою програмування у проєкті є Python. Її використання зумовлене наявністю широкого набору бібліотек для обробки відеоданих, роботи з масивами, побудови нейронних мереж і створення графічного інтерфейсу користувача. Python також забезпечує зручну інтеграцію різних модулів системи в єдиний програмний ланцюг обробки.

Для роботи з відеопотоком у системі використано бібліотеку OpenCV [17]. Вона застосовується для зчитування відеофайлів, отримання параметрів відео, покадрової обробки, візуалізації проміжних результатів та формування вихідного відео з накладеними результатами аналізу. Завдяки OpenCV система може працювати зі звичайними відеозаписами гімнастичних вправ без необхідності використання спеціалізованих сенсорів або маркерів під час запуску основного програмного комплексу.

Для отримання скелетної моделі спортсмена використано MediaPipe Pose [11], [12]. Цей інструмент дає змогу визначати координати ключових точок тіла людини на кожному кадрі відео. У межах проєкту

використовуються просторові координати точок x , y , z , а також показник видимості (*visibility*), що характеризує надійність виявлення відповідної точки. У реалізації передбачено використання як класичного підходу через `mp.solutions.pose`, так і сучаснішого варіанту MediaPipe Tasks із моделлю `pose_landmarker_full.task`.

Для обробки числових даних використано бібліотеку NumPy [18]. Вона застосовується під час нормалізації скелетних координат, формування ознак, приведення сегментів руху до єдиної довжини та підготовки вхідних даних для класифікаційної моделі. Використання NumPy дає змогу ефективно працювати з багатовимірними масивами, які описують часові послідовності координат ключових точок тіла.

Сегментація гімнастичної вправи в розробленій системі реалізована евристичним способом. Вона базується на аналізі опорних поз спортсмена, швидкості руху та тривалості окремих фаз. Такий підхід дає змогу поділяти безперервний відеопотік на окремі сегменти, які надалі можуть бути подані на вхід класифікаційної моделі. У межах цієї роботи сегментація не є універсальною для всіх можливих гімнастичних вправ, а орієнтована на визначені типи рухових станів і переходів між ними.

Для реалізації класифікатора гімнастичних елементів використано бібліотеку PyTorch [14]. У проєкті побудовано модель на основі двонапрявленої рекурентної нейронної мережі Bi-GRU [15]. Така архітектура дає змогу аналізувати часову послідовність руху як у прямому, так і у зворотному напрямі, що є важливим для розпізнавання гімнастичних елементів, структура яких залежить від послідовності положень тіла у часі. PyTorch використано як для навчання моделі в окремому тренувальному скрипті, так і для подальшого інференсу в основному процесі обробки.

Для зберігання метаданих у проєкті використано JSON-файли [21]. Зокрема, вони містять опис гімнастичних елементів, відповідність між індексами моделі та назвами класів, а також параметри навчання моделі. На основі цих даних система після класифікації елементів виконує підрахунок

складності комбінації за внутрішніми правилами. Такий підхід дає змогу відокремити логіку програмної обробки від довідкової інформації про елементи та спростити подальше розширення системи.

Графічний інтерфейс користувача реалізовано за допомогою Tkinter у поєднанні з бібліотекою Pillow [19], [20]. Tkinter використано для створення основних елементів інтерфейсу, зокрема вікон, кнопок, полів вибору файлів і відображення результатів роботи системи. Pillow застосовується для роботи із зображеннями в інтерфейсі. Наявність графічного інтерфейсу дає змогу користувачу запускати аналіз відео без безпосередньої роботи з програмним кодом.

Отже, програмна реалізація системи поєднує інструменти комп'ютерного зору, аналізу скелетних даних, евристичної сегментації руху, нейромережевої класифікації та графічного інтерфейсу користувача. Використання Python, OpenCV, MediaPipe, NumPy, PyTorch, Tkinter і Pillow забезпечило побудову цілісної системи для аналізу відеозаписів гімнастичних вправ та автоматизованого визначення їхніх складових елементів.

3.2 Структура програмного проєкту

Програмний проєкт побудовано за модульним принципом. Його структура відображає основні етапи роботи системи: обробку відео, отримання скелетної моделі, сегментацію руху, класифікацію гімнастичних елементів, розрахунок оцінки складності та взаємодію з користувачем через графічний інтерфейс. Такий підхід дає змогу розділити функціональність системи на окремі компоненти та спростити подальше розширення програмного комплексу.

Компонентна структура розробленого програмного забезпечення локалізована в кореневому каталозі src, де акумульовано базові функціональні модулі системи. Роль головного координатора

обчислювальних процесів виконує диспетчерський сценарій `src/main_pipeline.py`. Він виступає архітектурним осердям системи, що забезпечує наскрізну консолідацію та синхронізацію послідовних етапів потокового відеоаналізу:

- завантаження відеофайлу;
- виклик функцій отримання скелетних точок і сегментації;
- передавання виділених сегментів до класифікатора;
- обробку результатів класифікації;
- підрахунок оцінки складності за внутрішніми правилами системи.

Модуль `src/element_segmenter.py` відповідає за обробку відеопотоку, отримання скелетної моделі спортсмена та поділ руху на окремі сегменти. У межах цього модуля використовується MediaPipe Pose [11], [12], а також може використовуватися `pose_landmarker_full.task` – файл моделі MediaPipe Tasks для отримання скелетних орієнтирів тіла. Модуль `src/element_classifier.py` забезпечує класифікацію виділених сегментів за допомогою навченої моделі Bi-GRU.

Окремо в проєкті виділено каталог `training`, який містить скрипти для навчання класифікаційної моделі. Зокрема, файл `training/train_element_classifier.py` використовується для завантаження даних, навчання моделі та збереження результатів навчання. Каталог `dataset` призначений для зберігання навчальних даних і метаданих: `dataset/manifest.csv` та `dataset/poses` використовуються під час навчання моделі, а `dataset/elements.json` містить опис гімнастичних елементів, їх складність і групи для подальшого підрахунку оцінки після класифікації.

У каталозі `models` зберігаються навчена модель `element_classifier.pt`, файл відповідності між індексами та назвами класів `label_map.json`, а також конфігурація навчання `train_config.json`. Каталог `processed_videos` використовується для збереження результатів роботи системи, зокрема оброблених відеофайлів з накладеними результатами аналізу. Графічний інтерфейс користувача реалізовано у файлі `ui_app.py`, через який користувач

може обрати відеофайл, запустити аналіз і переглянути отримані результати.

Взаємодія модулів відбувається послідовно. Користувач обирає відео через графічний інтерфейс або запускає основний сценарій обробки. Далі система отримує скелетні точки спортсмена, виконує сегментацію руху, передає кожен сегмент до класифікатора та на основі JSON-метаданих розраховує оцінку складності за внутрішніми правилами. Після завершення аналізу формуються результати роботи системи та, за потреби, зберігається оброблене відео.

Отже, структура програмного проєкту є логічною та відповідає основним етапам аналізу гімнастичної вправи. Модульна організація коду забезпечує зрозумілий розподіл відповідальності між компонентами, спрощує супровід системи та створює можливості для її подальшого вдосконалення. Повний програмний код основного pipeline обробки відео наведено у додатку А.

3.3 Формування та підготовка набору даних

Для навчання класифікаційної моделі було сформовано спеціалізований набір даних із прикладами окремих гімнастичних елементів у вигляді послідовностей скелетних координат. Використання власного набору даних зумовлене тим, що відкриті відеонабори не відповідають переліку елементів, потрібних для подальшого розрахунку оцінки складності, наближеної до D-score. Тому підготовка даних виконувалася безпосередньо в межах програмного проєкту.

Початковими даними були відеозаписи гімнастичних вправ, що містили окремі елементи або їх послідовності. Оскільки класифікаційна модель повинна отримувати на вхід не цілий відеозапис, а окремий руховий сегмент із відповідною міткою класу, на першому етапі з відео виділялися ділянки, які відповідають конкретним гімнастичним елементам.

Для цього використовувалися допоміжні сценарії проєкту, які дозволяли попередньо виділяти потенційні рухові сегменти, переглядати їх і зберігати підтверджені відеосегменти. Після цього окремий сценарій додавав підтверджений сегмент до набору даних: виконував екстракцію скелетних координат, зберігав їх у форматі .prz та пов'язував отриманий приклад із відповідною міткою класу. Такий підхід поєднує автоматизоване попереднє виділення рухових ділянок із ручним контролем правильності розмітки.

Після підтвердження сегмента виконувалася екстракція скелетної моделі тіла гімнаста засобами MediaPipe [12]. Для кожного кадру формувався набір ключових точок, що описують положення основних частин тіла. Кожна ключова точка зберігалася у вигляді координат x, y, z та показника visibility. Таким чином, кожен приклад у наборі даних подавався не як послідовність зображень, а як послідовність числових ознак, що описують зміну положення тіла гімнаста в часі.

Програмний код для збереження прикладу гімнастичного елемента до набору даних наведено у лістингу 3.1.

Лістинг 3.1 – Програмний код для збереження прикладу гімнастичного елемента до набору даних

```
elements = _load_elements(elements_path)
if label in elements:
    existing = elements[label]
    old_difficulty = float(existing["difficulty"])
    old_group = int(existing["group"])
    difficulty_changed =
    abs(old_difficulty - difficulty) > 1e-6
    group_changed = old_group != group
    if difficulty_changed or group_changed:
        raise ValueError(
            f"Label '{label}' вже є в elements.json "
            f"з іншими даними: {existing}"
```

Продовження лістингу 3.1

```

        )
    else:
        elements[label] = {
            "difficulty": float(difficulty),
            "group": int(group),
        }
        _save_elements(elements_path, elements)

xyz, vis, fps = extract_landmarks(video_path)

poses_path = poses_dir / f"{elem_id}.npz"
np.savez_compressed(
    poses_path,
    xyz=xyz,
    vis=vis,
    fps=fps,
)

```

Як видно з лістингу 3.1, перед додаванням нового прикладу система перевіряє, чи існує відповідна мітка в `elements.json`. Якщо елемент уже наявний, його складність і група мають збігатися з раніше збереженими даними. Після цього з відео витягуються масиви `xyz`, `vis` і значення `fps`, які зберігаються у стисненому форматі `.npz`. Запис мітки класу, шляху до файлу зі скелетними координатами та службових параметрів до `manifest.csv` виконується далі в цьому ж сценарії.

Скелетні дані зберігаються у каталозі `dataset/poses`. Кожен файл `.npz` містить масив `xyz`, масив `vis` та значення `fps`. Масив `xyz` відповідає координатам ключових точок, `vis` – показникам їхньої видимості, а `fps` – частоті кадрів вихідного відеозапису. Такий формат дозволяє відокремити етап обробки відео від етапу навчання моделі, оскільки під час навчання використовуються вже підготовлені скелетні послідовності.

Опис прикладів набору даних зберігається у файлі `dataset/manifest.csv`. Він містить ідентифікатор прикладу, шлях до файлу зі скелетними координатами, мітку класу, поле `split`, частоту кадрів і кількість кадрів у сегменті. У поточній версії всі приклади позначені як `train`, тому поділ на навчальну, валідаційну та тестову вибірки виконується автоматично під час запуску скрипта навчання.

Довідкова інформація про гімнастичні елементи зберігається у файлі `dataset/elements.json`. Для кожного елемента вказується його базова складність і належність до групи. Ці дані потрібні не для навчання моделі, а для інтерпретації розпізнаних класів і подальшого розрахунку оцінки складності.

Підготовка даних до навчання реалізована у файлі `training/train_element_classifier.py`. Під час завантаження прикладів координати ключових точок нормалізуються для зменшення впливу положення спортсмена в кадрі, масштабу зображення та відстані до камери. Для роботи з масивами використовується NumPy [18], а для формування тензорів і навчання моделі – PyTorch [14]. Після нормалізації координати x , y , z об'єднуються з показником `visibility`. Оскільки система використовує 33 ключові точки й 4 ознаки для кожної з них, один часовий крок описується 132 ознаками.

Програмний код для формування вхідних ознак для навчання моделі Bi-GRU наведено у лістингу 3.2.

Лістинг 3.2 – Програмний код для формування вхідних ознак для навчання моделі Bi-GRU

```
def __getitem__(self, idx: int):
    rec = self.records[idx]
    data = np.load(rec.poses_path)
    xyz = data["xyz"].astype(np.float32)
    vis = data["vis"].astype(np.float32)
    xyz = normalize_xyz(xyz)
```

Продовження лістингу 3.2

```

        features =
        np.concatenate([xyz, vis[..., None]], axis=2)
        features = features.reshape(features.shape[0], -1)
        features =
        resample_sequence(features, self.seq_len)
        x = torch.from_numpy(features)
        y =
        torch.tensor(self.label_map[rec.label],
dtype=torch.long)
        return x, y

```

У лістингу 3.2 показано, що під час завантаження прикладу з файлу .prz окремо зчитуються координати x, y, z та показники видимості vis. Після нормалізації координат ці дані об'єднуються в єдиний масив ознак розмірності [T, 33, 4], де T – початкова кількість кадрів у сегменті. Далі масив перетворюється у формат [T, 132] і приводиться до фіксованої довжини послідовності.

Оскільки гімнастичні елементи можуть мати різну тривалість, перед поданням до нейронної мережі всі послідовності приводяться до однакової довжини. У програмній реалізації це виконується шляхом ресемплінгу, тобто перетворення початкової послідовності кадрів на послідовність із фіксованою кількістю часових кроків. У поточній конфігурації довжина послідовності становить 64 часові кроки. Це дозволяє подавати всі приклади на вхід моделі Bi-GRU незалежно від фактичної тривалості елемента у відеозаписі.

У поточному наборі даних використовується 142 приклади, які належать до 14 класів гімнастичних елементів. До цих класів належать cast, diamidov, diamidov_540, double_forward, double_pike, felge, felge_turn, healy, honma, makuts, press, stutzkehr, turn та uprise. Кількість прикладів для різних класів є нерівномірною, що є типовою проблемою для прикладних наборів даних, сформованих на основі обмеженої кількості відеозаписів.

Нерівномірність розподілу класів може впливати на якість навчання моделі, оскільки класи з більшою кількістю прикладів можуть розпізнаватися стабільніше, ніж рідкісні класи. Для часткової компенсації цієї проблеми у скрипті навчання передбачено використання ваг класів. За замовчуванням цей механізм увімкнений, якщо його не вимкнено окремим параметром запуску.

Таким чином, формування та підготовка набору даних у розробленій системі включає виділення рухових сегментів із відеозаписів, призначення міток класів, збереження скелетних координат у форматі .prz, опис прикладів у файлі manifest.csv, нормалізацію координат, об'єднання координат із показниками visibility та приведення послідовностей до сталої довжини. Така структура даних забезпечує можливість навчання Bi-GRU моделі на послідовностях скелетних ознак і подальшого використання результатів класифікації для розрахунку оцінки складності, наближеної до D-score.

3.4 Реалізація обробки відео та екстракції скелетної моделі

Обробка відео та екстракція скелетної моделі є першим основним обчислювальним етапом роботи системи після завантаження вхідного відеофайлу. На цьому етапі відеозапис гімнастичної вправи перетворюється з послідовності зображень на послідовність числових ознак, які описують положення тіла гімнаста в кожному кадрі. Надалі ці дані використовуються для сегментації вправи на окремі рухові сегменти та класифікації гімнастичних елементів.

У загальному pipeline системи цей етап викликається у файлі src/main_pipeline.py через функцію extract_landmarks. Після її виконання система отримує масиви скелетних координат і показників видимості, а також основні параметри відео. Далі отримані дані передаються до модуля

сегментації, який визначає часові межі потенційних гімнастичних елементів.

Основна реалізація екстракції скелетної моделі розміщена у файлі `src/element_segmenter.py`. Функція `extract_landmarks` приймає шлях до відеофайлу, відкриває його засобами OpenCV [17] та перевіряє, чи файл може бути коректно зчитаний. Після відкриття відео визначаються його базові параметри: частота кадрів `fps`, ширина кадру `frame_w` та висота кадру `frame_h`. Ці параметри використовуються на наступних етапах для коректної обробки часових інтервалів і візуалізації результатів.

Для екстракції пози гімнаста у системі передбачено два варіанти використання MediaPipe [12]. Перший варіант базується на стандартному модулі `mp.solutions.pose`, який виконує покадрове визначення ключових точок тіла. Другий варіант використовує MediaPipe Tasks і модель `pose_landmarker_full.task`. Перед використанням цього варіанта система перевіряє наявність відповідного файлу моделі. Якщо файл відсутній, генерується повідомлення про помилку та вказується необхідність розмістити модель поруч зі скриптом або в кореневому каталозі проєкту. Під час обробки відео кожен кадр зчитується послідовно. Оскільки OpenCV завантажує кадри у форматі BGR [17], перед поданням до MediaPipe [12] виконується перетворення зображення у формат RGB. Це необхідно тому, що моделі MediaPipe очікують саме RGB-представлення зображення. Після цього кадр передається до моделі визначення пози, яка повертає координати ключових точок тіла.

У розробленій системі для кожного кадру формується скелетна модель, що містить 33 ключові точки тіла [11], [12]. Для кожної точки зберігаються координати x , y , z та показник `visibility` [11], [12]. Координати x і y описують положення точки на зображенні, координата z відображає відносну глибину, а `visibility` характеризує надійність виявлення точки у поточному кадрі. Такий формат даних дозволяє використовувати не лише

геометричне положення частин тіла, а й оцінку достовірності визначених координат.

Якщо у певному кадрі позу гімнаста не виявлено, система не перериває обробку відео. У такому випадку для відповідного кадру формуються нульові масиви координат і показників видимості. Це рішення дозволяє зберегти узгоджену структуру даних для всього відеозапису, оскільки кожному кадру відповідає один запис у масивах `xyz` і `vis`. Завдяки цьому подальші етапи обробки можуть працювати з послідовністю сталої структури навіть за наявності окремих кадрів, у яких поза була втрачена.

Після завершення покадрової обробки всі координати об'єднуються у підсумковий масив `xyz_all`, а показники видимості – у масив `vis_all`. Разом із ними функція повертає частоту кадрів, ширину та висоту відео. Отримані дані використовуються для подальшої сегментації руху та класифікації виділених сегментів.

Окремо у файлі `src/main_pipeline.py` реалізовано візуальний контроль результатів екстракції скелетної моделі. Для цього координати ключових точок перетворюються з відносних координат кадру в піксельні, після чого на відео відображаються точки та з'єднання між ними. Це дає змогу перевірити коректність визначення положення тіла гімнаста у відеопотоці.

Після екстракції скелетної моделі подальша обробка виконується вже не на основі пікселів кадру, а на основі координат ключових точок і показників їхньої видимості. Це зменшує обсяг даних і дозволяє зосередитися саме на русі тіла гімнаста, а не на другорядних особливостях кадру.

Програмний код фрагменту гілки реалізації на основі `mp.solutions.pose` для покадрової екстракції скелетної моделі з відео наведено у лістингу 3.3. Він демонструє основну логіку покадрової обробки: зчитування кадру, перетворення зображення з формату BGR у RGB, виклик моделі MediaPipe, формування масивів `xyz` і `vis`, а також обробку випадку, коли позу на кадрі не знайдено.

Лістинг 3.3 – Покадрова екстракція скелетної моделі з відео

```

xyz_frames = []
vis_frames = []

with mp_pose.Pose(
    static_image_mode=False,
    model_complexity=2,
    enable_segmentation=False,
    min_detection_confidence=0.25,
    min_tracking_confidence=0.25,
) as pose:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = pose.process(image)

    if results.pose_landmarks:
        lms = results.pose_landmarks.landmark
        xyz = np.array(
            [[lm.x, lm.y, lm.z] for lm in lms],
            dtype=np.float32,
        )
        vis = np.array(
            [getattr(lm, "visibility", 0.0) for lm in lms],
            dtype=np.float32,
        )
    else:
        xyz = np.zeros((33, 3), dtype=np.float32)
        vis = np.zeros((33,), dtype=np.float32)

    xyz_frames.append(xyz)
    vis_frames.append(vis)

```

Повна функція `extract_landmarks` додатково містить відкриття відеофайлу, зчитування параметрів відео, альтернативну гілку реалізації на основі `MediaPipe Tasks` та об'єднання покадрових результатів у підсумкові масиви `NumPy` [18].

Таким чином, реалізований етап обробки відео перетворює вхідний відеопотік на структуроване подання руху гімнаста у вигляді послідовності скелетних координат. Це дозволяє відокремити подальший аналіз від вихідного зображення кадру та працювати з компактними числовими ознаками, які безпосередньо описують положення тіла спортсмена в часі.

3.5 Реалізація сегментації гімнастичних елементів

Після екстракції скелетної моделі наступним етапом роботи системи є сегментація гімнастичної вправи, тобто поділ безперервної послідовності руху на окремі часові сегменти. Це необхідно тому, що класифікаційна модель `Bi-GRU` не обробляє весь відеозапис як один приклад, а отримує на вхід окремі рухові сегменти, кожен з яких потенційно відповідає певному гімнастичному елементу.

Основна реалізація сегментації розміщена у файлі `src/element_segmenter.py` у функції `compute_segments_from_landmarks`. На вхід цієї функції подаються масиви `xyz_all` і `vis_all`, отримані після екстракції скелетної моделі, а також параметри відеозапису: частота кадрів `fps`, ширина кадру `frame_w` та висота кадру `frame_h`. Результатом роботи функції є список пар кадрів (`start`, `end`), які визначають початок і завершення знайдених рухових сегментів.

У розробленій системі сегментація виконується не нейронною мережею, а евристичним алгоритмом на основі детермінованої логіки станів. Використання станів і переходів між ними відповідає загальній ідеї автоматного опису процесів, коли поведінка системи визначається поточним станом і умовами переходу до наступного стану. Такий підхід дає

змогу інтерпретувати процес визначення меж елемента, оскільки початок і завершення сегмента пов'язуються з переходами між характерними положеннями тіла гімнаста. У межах реалізації аналізуються три основні опорні стани: стійка на руках, кутова поза та стійке положення на ногах.

Для визначення стійки на руках використовується функція `_is_handstand_candidate`. Вона перевіряє взаємне розташування плечей, таза та гомілковостопних суглобів, вертикальність тулуба, орієнтацію спортсмена відносно камери та стабільність положення за швидкістю руху ключових точок. Кутова поза визначається функцією `_is_angle_candidate`, у якій аналізуються положення плеча, таза, коліна, гомілковостопного суглоба та кисті, а також кути в тазостегновому й колінному суглобах. Стійке положення на ногах визначається функцією `_is_feet_stand_candidate`, яка враховує положення плечей, таза, колін і гомілковостопних суглобів, оцінену лінію опори та низьку швидкість руху.

Додатково для перевірки стабільності положень використовується допоміжна функція `_speed_ok`, яка оцінює швидкість зміни положення ключових точок між сусідніми кадрами. Це дозволяє відрізнити відносно статичні опорні стани від активної фази руху, у якій координати точок змінюються значно інтенсивніше.

Під час обробки кожного кадру система окремо перевіряє, чи відповідає поточна скелетна поза одному з опорних станів. Для цього обчислюються логічні ознаки `handstand_candidate`, `angle_candidate` та `feet_candidate`. Якщо хоча б одна з них має значення `True`, кадр вважається таким, що належить до стабільного стану. Якщо ж жоден із характерних станів не підтверджується, система розглядає це як можливий вихід з опорного положення та початок активного руху.

Для підвищення стійкості алгоритму використовуються лічильники послідовного підтвердження станів: `stand_true_streak`, `angle_true_streak`, `feet_true_streak` і `false_streak`. Вони фіксують, скільки кадрів поспіль підтверджується відповідний опорний стан або, навпаки, не

підтверджується жоден із них. Завдяки цьому система не реагує на одиничні помилки визначення ключових точок.

Логіка переходів між опорним станом та активною фазою руху реалізована через змінні `in_stand` і `element_active`. Якщо один із лічильників досягає встановленого порога, система фіксує стабільний стан. Якщо після цього протягом заданої кількості кадрів опорні стани не підтверджуються, кадри починають позначатися як частина рухового сегмента.

Також у реалізації передбачено змінну `final_stand_locked`, яка використовується для обробки фінальної стійки наприкінці вправи та запобігає помилковому відкриттю нового сегмента після завершення комбінації.

Програмний код для визначення переходу від опорного стану до активного рухового сегмента наведено у лістингу 3.4.

Лістинг 3.4 – Програмний код для визначення переходу від опорного стану до активного рухового сегмента

```
if not in_stand:
    if feet_true_streak >= feet_min_frames:
        start_frame = i - feet_min_frames + 1
        in_stand = True

    if element_active:
        element_active = False
        set_element_state_from(start_frame, False)

    if near_end:
        final_stand_locked = True

elif angle_true_streak >= angle_min_frames:
    start_frame = i - angle_min_frames + 1
    in_stand = True
    if element_active:
```

Продовження лістингу 3.4

```

        element_active = False
        set_element_state_from(start_frame, False)
    elif stand_true_streak >= stand_min_frames:
        start_frame = i - stand_min_frames + 1
        in_stand = True

        if element_active:
            element_active = False

        set_element_state_from(start_frame, False)

    elif feet_true_streak >= feet_min_frames and near_end:
        final_stand_locked = True

        if element_active:
            element_active = False
            set_element_state_from(i, False)

    if (
        in_stand
        and not final_stand_locked
        and false_streak >= stand_end_frames
    ):
        end_frame = i - stand_end_frames + 1
        in_stand = False
        element_active = True
        set_element_state_from(end_frame, True)

```

У лістингу 3.4 показано основну логіку перемикання між стабільним станом і активною частиною руху. Якщо підтверджується стійке положення на ногах, кутова поза або стійка на руках, система встановлює стан `in_stand`. Коли після цього жоден з опорних станів не підтверджується протягом

заданої кількості кадрів, змінна `element_active` набуває значення `True`, а відповідні кадри позначаються як частина гімнастичного елемента.

Після проходження всієї послідовності кадрів система формує список сегментів на основі логічного масиву `element_flags`. Початком сегмента вважається перший кадр із позначкою `True`, а завершенням – останній кадр перед поверненням до `False`. Надто короткі сегменти відсіюються: мінімальна довжина визначається як максимум між тривалістю 0,5 с і обмеженням у 35 кадрів.

Після первинного формування сегментів виконується уточнення їхніх меж за інтенсивністю руху за допомогою функції `_trim_segment_by_motion`. Вона зміщує межі сегмента ближче до ділянки з реальною руховою активністю, що дозволяє зменшити кількість зайвих підготовчих або завершальних кадрів.

Отримані межі повертаються у вигляді пар (`start`, `end`) і надалі використовуються в основному `pipeline` для вирізання послідовностей `xyz` і `vis`, які передаються до моделі `Bi-GRU`. Отже, реалізований алгоритм сегментації поєднує геометричні критерії, перевірку видимості ключових точок, оцінювання швидкості руху та лічильники підтвердження станів.

3.6 Реалізація та навчання моделі `Bi-GRU`

Після сегментації гімнастичної вправи кожен виділений руховий сегмент передається до класифікаційної моделі. Її завдання полягає у визначенні того, якому гімнастичному елементу відповідає послідовність скелетних координат. У розробленій системі для цього використовується модель `Bi-GRU`, реалізована засобами бібліотеки `PyTorch` [14].

Основна реалізація навчання моделі розміщена у файлі `training/train_element_classifier.py`, а використання навченої моделі під час інференсу – у файлі `src/element_classifier.py`.

На вхід моделі подається послідовність скелетних ознак, у якій один часовий крок містить 132 ознаки: 33 ключові точки тіла та 4 параметри для кожної точки – x , y , z і $visibility$ [11], [12]. Перед поданням до моделі координати нормалізуються, об'єднуються з показниками видимості та приводяться до фіксованої довжини. У поточній конфігурації довжина послідовності становить 64 часові кроки.

Архітектура моделі складається з двонапрямого GRU-шару та повнозв'язного шару Linear. GRU є різновидом рекурентної нейронної мережі, призначеним для обробки послідовних даних [9]. У програмній реалізації внутрішні обчислення GRU не реалізуються вручну, а виконуються стандартним модулем `torch.nn.GRU` [15].

Програмний код для реалізації моделі Bi-GRU наведено у лістингу 3.5.

Лістинг 3.5 – Програмний код для реалізації моделі Bi-GRU

```
class GRUClassifier(nn.Module):
    def __init__(
        self,
        input_dim,
        hidden_dim,
        num_layers,
        num_classes,
    ):
        super().__init__()
        self.gru = nn.GRU(
            input_dim,
            hidden_dim,
            num_layers=num_layers,
            batch_first=True,
            bidirectional=True,
            dropout=(
                0.2 if num_layers > 1 else 0.0
            ),
        )
```

Продовження лістингу 3.5

```

        self.fc = nn.Linear(
            hidden_dim * 2,
            num_classes,
        )
    def forward(self, x):
        _, h = self.gru(x)
        h = torch.cat(
            [h[-2], h[-1]],
            dim=1,
        )
        return self.fc(h)

```

Як видно з лістингу 3.5, параметр `bidirectional=True` задає двонаправну обробку послідовності, а `batch_first=True` визначає формат вхідного тензора як `[batch, seq_len, features]`. Якщо кількість рекурентних шарів більша за один, між ними застосовується `dropout` [15]. У методі `forward` фінальні приховані стани прямого та зворотного напрямів об'єднуються за допомогою `torch.cat` і передаються до повнозв'язного шару.

Навчання моделі виконується на підготовленому наборі скелетних послідовностей. Якщо у файлі `manifest.csv` не задано окремі навчальну, валідаційну та тестову вибірки, скрипт навчання виконує поділ автоматично відповідно до параметрів `val_frac` і `test_frac`. Дані подаються до моделі пакетами через `DataLoader`. Навчання виконується на GPU, якщо такий пристрій доступний, або на CPU в іншому випадку.

Для навчання використовується функція втрат `CrossEntropyLoss`, а оптимізація параметрів виконується алгоритмом `Adam`. Використання `CrossEntropyLoss` відповідає задачі багатокласової класифікації, у якій модель має обрати один клас із заданого набору гімнастичних елементів [16]. Оскільки набір даних має нерівномірний розподіл класів, у навчанні передбачено використання ваг класів. Найкраща модель зберігається за значенням точності на валідаційній вибірці `val_acc`.

Основні гіперпараметри навчання зберігаються у файлі `models/train_config.json`. У поточній конфігурації довжина вхідної послідовності становить 64 часові кроки, розмір прихованого стану GRU – 128, кількість рекурентних шарів – 2, розмір пакета – 16, кількість епох навчання – 40, швидкість навчання – 0,001. Для автоматичного поділу даних використовуються частки 0,1 для валідаційної вибірки та 0,1 для тестової вибірки, а для відтворюваності результатів задано `seed = 42`.

Після завершення навчання у каталозі `models` зберігаються файл ваг моделі `element_classifier.pt`, мапа класів `label_map.json` та конфігурація навчання `train_config.json`. Під час роботи основної системи ці файли використовуються для відтворення архітектури моделі, завантаження її ваг і перетворення числового індексу класу на назву гімнастичного елемента.

На етапі інференсу для кожного виділеного сегмента виконується така сама підготовка ознак, як і під час навчання. Після проходження через модель вихідні значення перетворюються на ймовірності за допомогою `softmax` [10]. Система вибирає клас із найбільшою ймовірністю та повертає назву елемента разом із числовим значенням впевненості. Отриманий результат далі використовується для розрахунку оцінки складності, наближеної до D-score.

Таким чином, модель Bi-GRU виконує роль класифікаційного модуля, який перетворює виділені рухові сегменти на розпізнані назви гімнастичних елементів.

3.7 Реалізація алгоритму розрахунку D-score

Після сегментації відеозапису та класифікації виділених рухових сегментів система переходить до розрахунку підсумкової оцінки складності. На цьому етапі вже відомо, які гімнастичні елементи були розпізнані, тому подальша задача полягає у перетворенні результатів класифікації на

числову оцінку D-score, що в спортивній гімнастиці використовується для відображення складності комбінації [1].

Реалізація цього етапу розміщена у файлі `src/main_pipeline.py` у функції `score_video`. Метадані елементів зберігаються у файлі `dataset/elements.json`, де назва елемента є ключем JSON-об'єкта, а його складність і група – відповідними параметрами. Після класифікації система використовує ці дані для визначення складності та групи розпізнаного елемента. Такий підхід дозволяє відокремити сам алгоритм обробки відео від довідкової інформації про гімнастичні елементи.

Базова частина оцінки формується як сума складностей усіх розпізнаних елементів. Для цього під час проходження по сегментах система отримує передбачену назву елемента, знаходить його складність у `elements.json` і додає її до змінної `base_difficulty`. Одночасно з цим для кожної представленої групи елементів зберігається найбільша складність серед розпізнаних елементів цієї групи. Це потрібно для подальшого нарахування додаткових балів за групи.

Окремою особливістю реалізації є обробка складених випадків. У функції `_expand_elements` елемент `honma` розгортається на `arc+honma`, а елемент `cast` – на `cast+angle`. Це зроблено для того, щоб система могла враховувати окремі складові таких рухів під час підрахунку складності та аналізу груп елементів. Завдяки цьому складений рух не розглядається лише як один клас, а може бути врахований через кілька пов'язаних компонентів.

Після формування базової суми система розраховує бонуси за представлені групи елементів. Для кожної такої групи визначається найскладніший розпізнаний елемент. Якщо його складність становить 0,3 або більше, до оцінки додається 0,5 бала, інакше – 0,3 бала. Підсумкова оцінка визначається як сума базової складності елементів і бонусів за групи.

Програмний код для обчислення бонусів за групи та підсумкової оцінки наведено у лістингу 3.6.

Лістинг 3.6 – Програмний код для обчислення підсумкової оцінки складності

```
group_bonus = 0.0
group_bonus_map = {}

for group, best_diff in group_best_diff.items():
    bonus = (
        0.5 if best_diff >= 0.3 else 0.3
    )
    group_bonus_map[group] = bonus
    group_bonus += bonus

total_difficulty = (
    base_difficulty + group_bonus
)
```

У лістингу 3.6 показано, що підсумкова оцінка складається із суми складностей розпізнаних елементів `base_difficulty` та бонусів за представлені групи `group_bonus`. Значення бонусів для окремих груп зберігаються у `group_bonus_map`, а підсумковий результат – у `total_difficulty`.

Отримане значення використовується як оцінка складності гімнастичної комбінації. У межах розробленої системи алгоритм забезпечує автоматизований перехід від розпізнаних елементів до D-score на основі складності елементів, їхніх груп і правил нарахування додаткових балів [1].

Таким чином, у третьому розділі розглянуто програмну реалізацію основних модулів системи: використані технології, структуру проєкту, формування набору даних, екстракцію скелетної моделі, сегментацію елементів, навчання моделі Bi-GRU та алгоритм розрахунку D-score. Реалізовані модулі утворюють єдиний pipeline, який забезпечує послідовну обробку відеозапису від отримання скелетних координат до розпізнавання елементів і формування підсумкової оцінки складності.

4 ІНТЕРФЕЙС КОРИСТУВАЧА ТА АНАЛІЗ РОБОТИ СИСТЕМИ

4.1 Структура графічного інтерфейсу користувача

Для практичного використання розробленої системи було реалізовано графічний інтерфейс користувача. Його основне призначення полягає у тому, щоб забезпечити зручну взаємодію користувача із системою без необхідності безпосереднього запуску програмного коду або роботи з командним рядком. Через інтерфейс користувач може обрати вхідний відеофайл, запустити аналіз гімнастичної вправи, переглянути результати розпізнавання елементів і, за потреби, зберегти оброблене відео з накладеними результатами.

Графічний інтерфейс реалізовано у файлі `ui_app.py` з використанням бібліотеки Tkinter [19]. Для роботи із зображеннями у вікні попереднього перегляду використовується бібліотека Pillow [20]. Головне вікно програми має назву «Gymnastics Difficulty Scoring» і поділене на дві основні частини. Ліва частина призначена для вибору відеофайлу, запуску обробки, відображення числових і табличних результатів, а права частина використовується для попереднього перегляду результатів обробки.

У верхній частині інтерфейсу розміщено поле Input video, текстове поле для шляху до файлу та кнопку Browse. За допомогою цієї кнопки користувач відкриває стандартне діалогове вікно вибору файлу та обирає відеозапис гімнастичної вправи. У реалізації передбачено вибір поширених відеоформатів, зокрема `mp4`, `mov`, `avi` та `mkv`. Після вибору файлу шлях до нього відображається у відповідному текстовому полі.

Під полем вибору відео розміщено дві основні кнопки керування: Run code та Save video. Кнопка Run code запускає аналіз обраного відеофайлу. Після її натискання з графічного інтерфейсу викликається основний pipeline обробки, реалізований у `src/main_pipeline.py`. У процесі обробки система виконує отримання скелетної моделі засобами MediaPipe [11], [12],

сегментацію руху на рухові сегменти, класифікацію гімнастичних елементів за допомогою навченої моделі та розрахунок підсумкової оцінки складності. Кнопка **Save video** використовується для збереження обробленого відео. Вона стає доступною після попереднього запуску аналізу, оскільки для збереження потрібно спочатку сформувати результати обробки.

Головне вікно графічного інтерфейсу користувача представлено на рисунку 4.1.

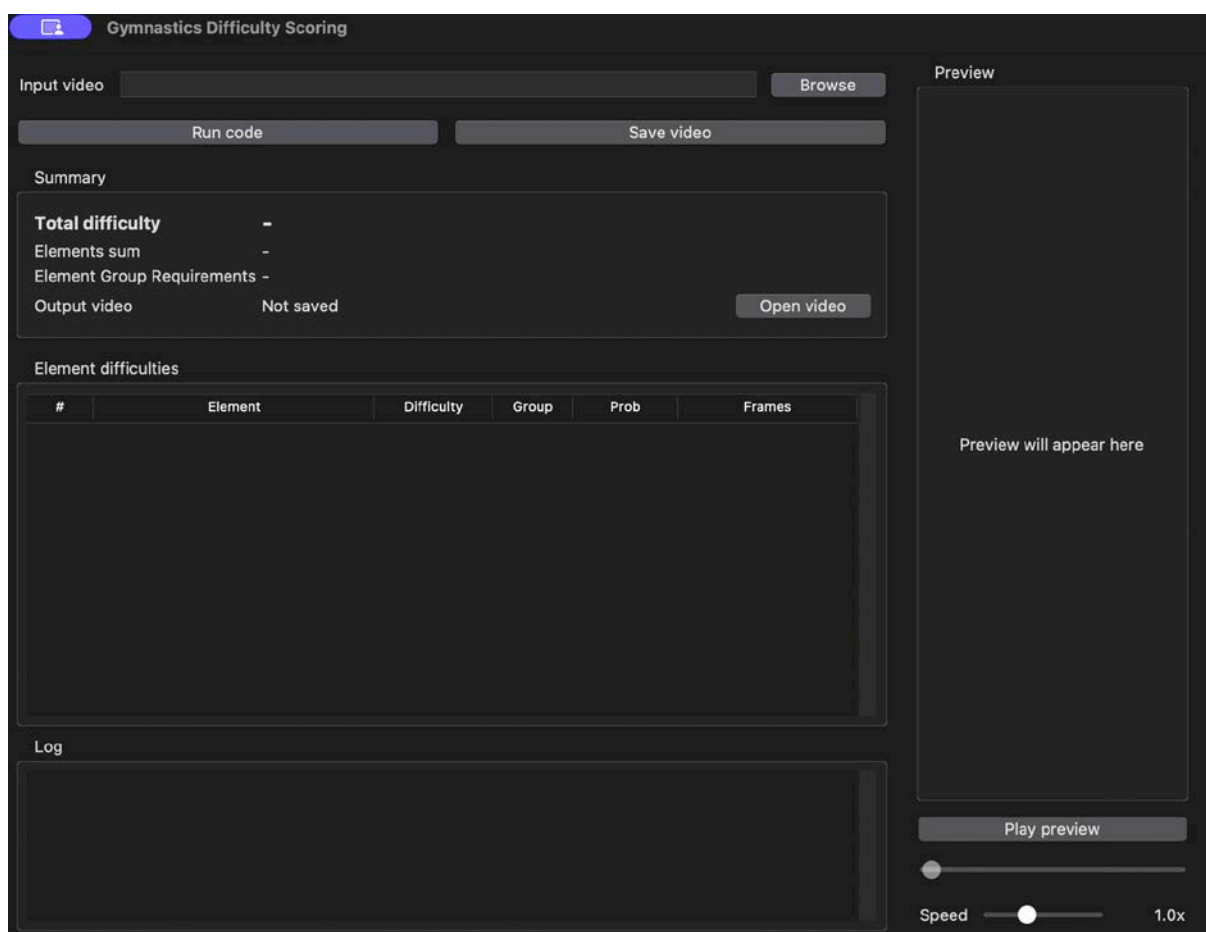


Рисунок 4.1 – Головне вікно графічного інтерфейсу користувача

Окремий блок **Summary** призначений для відображення підсумкових результатів аналізу. У ньому наведено значення **Total difficulty**, тобто підсумкову оцінку складності, **Elements sum** – суму складностей розпізнаних елементів, а також **Element Group Requirements** – додаткову

складову, пов'язану з урахуванням груп елементів. Також у цьому блоці відображається шлях до збереженого вихідного відеофайлу. Якщо оброблене відео було збережене, стає доступною кнопка Open video, яка дозволяє відкрити результат безпосередньо з інтерфейсу.

Нижче розміщено таблицю Element difficulties, у якій подаються розпізнані системою гімнастичні елементи. Для кожного елемента в таблиці відображається його порядковий номер, назва, складність, група, імовірність розпізнавання та межі відповідного рухового сегмента у кадрах. Таке подання дозволяє користувачу не лише побачити підсумкову оцінку, а й перевірити, з яких саме елементів вона була сформована.

У нижній частині лівої панелі розміщено блок Log. Він використовується для виведення службових повідомлень про стан роботи системи. У цьому блоці користувач може бачити повідомлення про вибраний вхідний файл, запуск обробки, завершення аналізу або можливі помилки. Наявність журналу повідомлень спрощує контроль за процесом обробки відео.

Права частина головного вікна містить блок Preview, призначений для попереднього перегляду результатів обробки. Після виконання аналізу в ньому відображається послідовність кадрів із накладеними результатами, а для керування переглядом передбачено кнопку Play preview, повзунок перемотування та регулятор Speed, що дає змогу переглядати матеріал у зручному темпі.

Таким чином, графічний інтерфейс розробленої системи забезпечує повний базовий сценарій взаємодії користувача з програмою: вибір відеозапису, запуск аналізу, перегляд підсумкової оцінки, аналіз таблиці розпізнаних елементів, попередній перегляд результатів обробки та збереження обробленого відео. Інтерфейс не містить окремих налаштувань моделі або параметрів навчання, оскільки його призначення полягає саме у практичному запуску вже реалізованого pipeline аналізу гімнастичної вправи.

4.2 Сценарій роботи користувача із системою

Робота користувача із системою починається із запуску графічного інтерфейсу, реалізованого у файлі `ui_app.py`. Після запуску програми відкривається головне вікно, у якому користувач може обрати вхідний відеозапис, запустити аналіз, переглянути результати обробки та зберегти оброблене відео з накладеними результатами. Такий підхід дає змогу використовувати систему без роботи з командним рядком і без безпосереднього редагування програмного коду.

Першим етапом роботи є вибір відеофайлу. Для цього користувач натискає кнопку `Browse` у блоці `Input video` та обирає відеозапис гімнастичної вправи. Після вибору шлях до файлу відображається у відповідному текстовому полі. Якщо файл не було обрано або вказаний шлях не відповідає існуючому відеофайлу, система не запускає обробку та виводить повідомлення про помилку.

Після вибору відео користувач натискає кнопку `Run code`. Перед початком аналізу система очищає попередні результати, перевіряє коректність вхідного файлу та запускає основний `pipeline` обробки. Аналіз виконується у фоновому потоці, що дає змогу не блокувати роботу графічного інтерфейсу під час обробки відеозапису. У процесі роботи в блоці `Log` відображаються службові повідомлення про стан виконання аналізу.

Під час обробки відео система послідовно виконує отримання скелетної моделі спортсмена, сегментацію руху на окремі рухові сегменти, класифікацію гімнастичних елементів та розрахунок оцінки складності. Користувач на цьому етапі очікує завершення обробки, після чого результати автоматично відображаються у відповідних полях інтерфейсу.

Після завершення аналізу в блоці `Summary` оновлюються підсумкові значення: `Total difficulty`, `Elements sum` та `Element Group Requirements`. Вони відображають загальну оцінку складності, суму складностей розпізнаних

елементів і додаткову складову за групи елементів. Детальні результати подаються у таблиці *Element difficulties*, де для кожного розпізнаного елемента наведено його назву, складність, групу, імовірність розпізнавання та межі відповідного рухового сегмента у кадрах.

Окремо користувач може переглянути результати обробки у блоці *Preview*. У цьому блоці відображається послідовність кадрів із накладеними результатами аналізу. Для керування переглядом передбачено кнопку *Play preview*, повзунок перемотування та регулятор *Speed*. Це дозволяє перевірити, у яких моментах відеозапису система визначила окремі рухові сегменти та розпізнані гімнастичні елементи.

Після попереднього запуску аналізу користувач може зберегти оброблене відео. Для цього використовується кнопка *Save video*, після натискання якої відкривається діалогове вікно вибору шляху збереження. Після формування вихідного відеофайлу його шлях відображається у блоці *Summary*. За потреби користувач може відкрити збережене відео безпосередньо з інтерфейсу за допомогою кнопки *Open video*.

Отже, сценарій роботи користувача із системою є послідовним і складається з кількох основних етапів: запуск програми, вибір відеозапису, запуск аналізу, очікування завершення обробки, перегляд результатів, попередній перегляд кадрів із накладеними результатами та збереження вихідного відео. Така організація взаємодії робить систему зручною для практичного використання під час аналізу гімнастичних вправ.

4.3 Демонстрація результатів обробки відео

Після завершення аналізу відеозапису система формує результати, які відображаються у графічному інтерфейсі користувача. Основна мета цього етапу полягає у тому, щоб надати користувачу не лише підсумкову оцінку складності, наближену до *D-score*, а й деталізовану інформацію про розпізнані гімнастичні елементи. Результати подаються у числовій,

табличній та візуальній формах, що спрощує перевірку роботи системи та аналіз окремих моментів відеозапису.

Після виконання обробки у блоці Summary відображаються основні числові показники. Поле Total difficulty містить підсумкову оцінку складності комбінації, яка формується на основі складностей розпізнаних елементів і додаткових балів за групи. Поле Elements sum показує суму базових складностей елементів, а Element Group Requirements відображає додаткову складову, пов'язану з урахуванням груп елементів. Таким чином, користувач може бачити не лише загальний результат, а й його основні складові.

Результати аналізу гімнастичної вправи в інтерфейсі користувача представлено на рисунку 4.2.

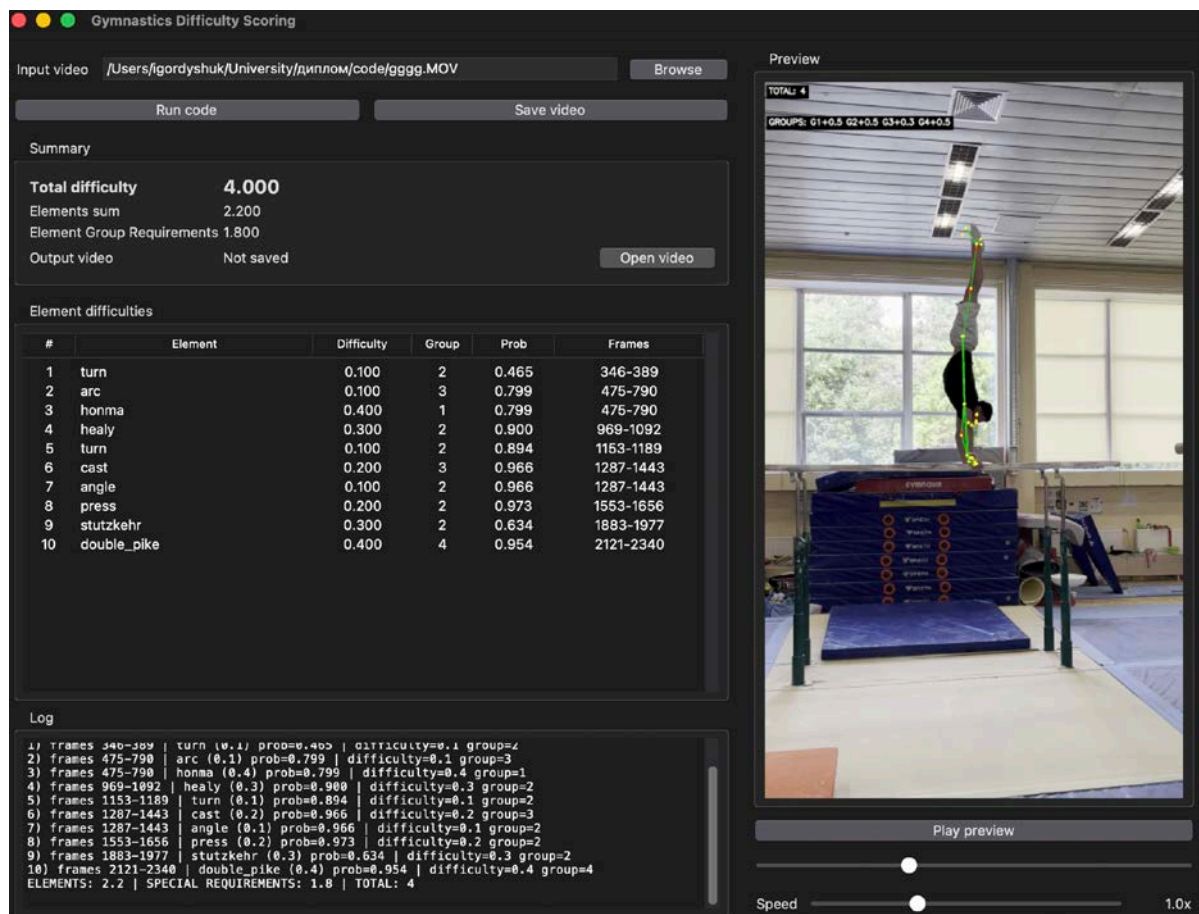


Рисунок 4.2 – Результати аналізу гімнастичної вправи в інтерфейсі користувача

Детальні результати класифікації подаються у таблиці Element difficulties. У ній для кожного розпізнаного елемента відображається порядковий номер, назва елемента, його складність, група, імовірність розпізнавання та межі відповідного рухового сегмента у кадрах. Таке подання дозволяє простежити, які саме елементи були враховані під час формування підсумкової оцінки складності. Крім того, наявність меж сегмента дає змогу зіставити результат класифікації з конкретною частиною відеозапису.

Окремо результати демонструються у блоці Preview. У цьому режимі користувач може переглядати послідовність кадрів із накладеними результатами обробки. На кадрах відображається скелетна модель спортсмена, а також службова інформація про розпізнаний елемент, підсумкову оцінку складності та групові бонуси. Завдяки цьому користувач може візуально перевірити, чи відповідають результати розпізнавання фактичному руху гімнаста у відеозаписі.

Кадр відео з накладеними результатами обробки представлено на рисунку 4.3.

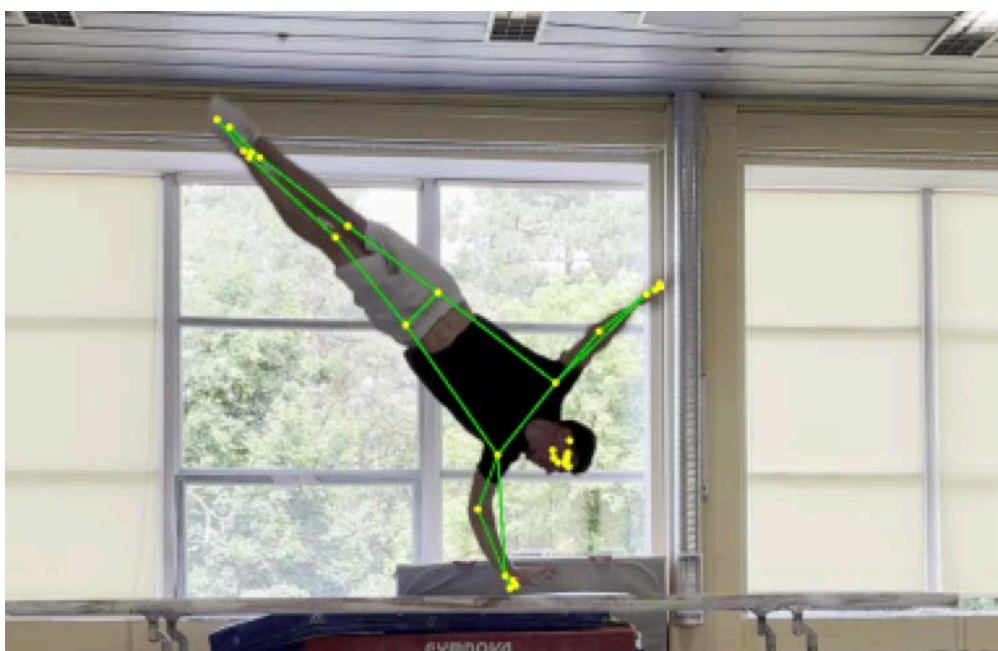


Рисунок 4.3 – Кадр відео з накладеними результатами обробки

За потреби користувач може зберегти оброблене відео у файл. У такому випадку система формує вихідний відеозапис, у якому результати аналізу накладаються безпосередньо на кадри. Для роботи з відеоданими та формування вихідного відео використовується бібліотека OpenCV [17]. Після збереження шлях до відеофайлу відображається у блоці Summary, а користувач може відкрити його за допомогою кнопки Open video. Це дозволяє переглянути результат поза межами інтерфейсу програми або використати його для подальшого аналізу.

Отже, демонстрація результатів обробки відео в розробленій системі охоплює кілька рівнів подання інформації. Числові показники дають змогу оцінити загальну складність комбінації, таблиця розпізнаних елементів деталізує внесок кожного рухового сегмента, а попередній перегляд і збережене відео дозволяють візуально перевірити результати аналізу. Такий підхід робить роботу системи більш зрозумілою для користувача та дає змогу використовувати її як допоміжний інструмент для аналізу гімнастичних вправ.

4.4 Обмеження розробленої системи

Розроблена система забезпечує автоматизований аналіз відеозапису гімнастичної вправи, однак її робота має низку обмежень, пов'язаних як з умовами зйомки, так і з особливостями використаних алгоритмів. Ці обмеження необхідно враховувати під час інтерпретації отриманих результатів, оскільки система призначена для допоміжного аналізу, а не для повної заміни офіційного суддівства.

Одним із основних обмежень є залежність від якості вхідного відео. Якщо відеозапис має низьку роздільну здатність, розмиття, недостатнє освітлення або різкі зміни положення камери, якість отримання скелетної моделі може знижуватися. У таких випадках ключові точки тіла спортсмена

можуть визначатися нестабільно, що впливає на подальшу сегментацію руху та класифікацію гімнастичних елементів.

Важливим фактором також є ракурс зйомки та видимість тіла гімнаста в кадрі. Система працює коректніше за стабільного та інформативного ракурсу, близького до умов, для яких побудовані евристичні сегментації. Якщо спортсмен не повністю потрапляє в кадр або основні частини тіла перекриваються приладом, іншими об'єктами чи самим спортсменом, це може призводити до помилкового визначення ключових точок і неточного виділення рухових сегментів.

Окреме обмеження пов'язане з використанням MediaPipe Pose або MediaPipe Tasks для отримання скелетної моделі [11], [12]. Робота наступних етапів системи безпосередньо залежить від того, наскільки стабільно ці інструменти визначають положення ключових точок у кожному кадрі. Якщо поза спортсмена не визначається або окремі точки мають низьку видимість, система може отримати неповні чи неточні дані для аналізу. Крім того, поточна реалізація орієнтована на аналіз одного спортсмена в кадрі, а не на складні багатолюдні сцени.

Ще одним обмеженням є склад розпізнаваних класів. Класифікатор працює лише з тими гімнастичними елементами, які були представлені у навчальному наборі даних і задані у мапі класів. Тому система не може коректно розпізнавати всі можливі елементи спортивної гімнастики. Якщо у відеозаписі присутній елемент, якого немає серед навчальних класів, модель може віднести його до найбільш схожого відомого класу або сформувати помилковий результат.

Якість класифікації також обмежується обсягом і збалансованістю навчального набору даних. У поточній реалізації набір містить обмежену кількість прикладів, а кількість зразків для різних класів є нерівномірною. Через це елементи, які мають більше навчальних прикладів, можуть розпізнаватися стабільніше, ніж рідкісні елементи. Це знижує узагальнювальну здатність моделі під час обробки нових відеозаписів,

особливо якщо вони відрізняються за ракурсом, якістю зйомки або манерою виконання руху.

Також слід враховувати, що реалізований розрахунок оцінки складності є спрощеним і базується на внутрішніх правилах системи. Оцінка формується на основі розпізнаних елементів, їх складності та групових бонусів, однак не охоплює всі деталі офіційного регламенту FIG [1]. Зокрема, система не аналізує повний перелік суддівських вимог, не оцінює якість виконання елементів і не розраховує E-score.

Отже, розроблена система може використовуватися як допоміжний інструмент для автоматизованого відеоаналізу та попереднього розрахунку оцінки складності, наближеної до D-score. Водночас її результати потребують експертної перевірки, особливо у випадках складних ракурсів, низької якості відео, нестабільного визначення скелетної моделі або наявності елементів, які не входять до навчального набору даних.

4.5 Перспективи подальшого розвитку

Розроблена система демонструє можливість автоматизованого аналізу відеозапису гімнастичної вправи, виділення рухових сегментів, класифікації елементів і розрахунку оцінки складності, наближеної до D-score. Водночас поточна реалізація має демонстраційний характер, тому її подальший розвиток може здійснюватися у кількох напрямках.

Одним із основних напрямів удосконалення є розширення переліку розпізнаваних гімнастичних елементів. У поточній реалізації класифікатор працює лише з тими класами, які були представлені у навчальному наборі даних і задані у мапі класів. У подальшому систему можна доповнити новими елементами, розширити файл метаданих `elements.json` та повторно навчити модель на більшій кількості прикладів. Це дозволить наблизити систему до практичного використання для аналізу ширшого набору гімнастичних комбінацій.

Важливою перспективою є збільшення та балансування навчального набору даних. Для підвищення якості класифікації доцільно зібрати більше прикладів для кожного елемента, залучити відеозаписи різних спортсменів, різні ракурси зйомки, темп виконання та умови освітлення. Більш різноманітний набір даних може підвищити здатність моделі узагальнювати результати на нові відеозаписи та зменшити залежність від окремих умов зйомки.

Окремим напрямом розвитку є підтримка інших снарядів і типів гімнастичних вправ. Поточна система орієнтована на конкретний набір елементів і відповідні правила сегментації руху. Для адаптації до інших снарядів необхідно розширити перелік класів, уточнити метадані елементів, а також змінити або доповнити алгоритми сегментації відповідно до характерних положень тіла та фаз руху на кожному снаряді.

Подальше вдосконалення також може бути пов'язане з точнішим урахуванням правил FIG [1]. У поточній реалізації підсумкова оцінка формується за спрощеною схемою на основі складностей розпізнаних елементів і групових бонусів. У майбутньому доцільно реалізувати повнішу модель розрахунку D-score, яка враховуватиме більшу кількість спеціальних вимог, обмежень, повторень елементів, правил зарахування та особливостей конкретного гімнастичного снаряда.

Для підвищення точності розпізнавання можна вдосконалити класифікаційну модель. Це може включати добір інших архітектур нейронних мереж, налаштування гіперпараметрів, використання більш якісних ознак, а також поєднання скелетних координат із додатковими візуальними характеристиками кадру [10], [14]. Такі зміни можуть підвищити стабільність класифікації складних або схожих між собою гімнастичних елементів.

Перспективним напрямом є перехід від обробки готового відеофайлу до роботи в реальному часі. У такому режимі система могла б отримувати відеопотік безпосередньо з камери, покадрово будувати скелетну модель,

визначати рухові сегменти, класифікувати елементи та оновлювати результати в інтерфейсі під час виконання вправи. Це розширило б можливості використання системи у тренувальному процесі та під час оперативного аналізу виступів.

Ще одним важливим напрямом подальшого розвитку є додавання модуля оцінювання виконання, тобто E-score. Поточна система зосереджена переважно на визначенні складності комбінації, тоді як оцінка виконання потребує аналізу технічних помилок, положення тіла, амплітуди рухів, стабільності приземлення, відхилень від необхідних кутів і форми елементів. Реалізація такого модуля дозволила б перейти від аналізу лише складності до більш комплексної підтримки суддівського оцінювання.

Отже, подальший розвиток системи може бути спрямований на розширення набору елементів, збільшення навчального набору даних, підтримку інших снарядів, уточнення правил розрахунку складності, підвищення точності моделі, реалізацію роботи в реальному часі та додавання модуля оцінювання E-score. Такі вдосконалення дадуть змогу зробити систему більш універсальною, стабільною та придатною для практичного використання у спортивній гімнастиці.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено інтелектуальну систему автоматизованого відеоаналізу гімнастичних вправ для розпізнавання елементів і розрахунку оцінки складності, наближеної до D-score. Поставлене завдання виконано в повному обсязі. Проаналізовано предметну галузь, досліджено існуючі рішення, обґрунтовано архітектуру системи, сформовано навчальний набір даних, реалізовано програмні модулі обробки відео, сегментації руху, класифікації елементів, розрахунку оцінки складності та графічний інтерфейс користувача.

Проведений аналіз показав, що традиційне суддівство у спортивній гімнастиці залежить від людського фактору, а наявні системи відеоповторів не виконують автоматичного розпізнавання елементів. Високоточні рішення на основі 3D-сенсорів потребують спеціального обладнання, тому їх використання є обмеженим. У зв'язку з цим розробка доступної програмної системи, яка працює зі звичайним відеозаписом, є актуальною та практично доцільною.

У роботі обґрунтовано використання гібридної архітектури, що поєднує комп'ютерний зір, автомати станів і двонапрямну рекурентну нейронну мережу Bi-GRU. Відеозапис перетворюється на послідовність скелетних координат, після чого система виділяє рухові сегменти, класифікує їх як гімнастичні елементи та формує підсумкову оцінку складності. Такий підхід дозволяє аналізувати не все зображення кадру, а саме положення тіла спортсмена та динаміку руху.

Практичну реалізацію виконано мовою Python із використанням OpenCV, MediaPipe, NumPy, PyTorch, Tkinter і Pillow. Система включає модулі отримання скелетної моделі, сегментації руху, класифікації елементів за допомогою Bi-GRU, розрахунку оцінки складності та графічний інтерфейс користувача. Для навчання моделі сформовано власний набір даних зі скелетними послідовностями гімнастичних

елементів. У поточній реалізації він містить приклади для 14 класів, а кожен руховий сегмент подається моделі як послідовність із 64 часових кроків і 132 ознак на кожному кроці.

Порівняно з існуючими аналогами розроблена система має перевагу в доступності, оскільки не потребує спеціального обладнання та може працювати зі звичайним відеозаписом. На відміну від систем відеоповторів, вона виконує автоматизовану обробку рухових даних і формує попередню оцінку складності. Водночас система не є повною заміною офіційного суддівства, оскільки розрахунок оцінки реалізовано у спрощеному вигляді та не охоплює всі вимоги правил FIG.

Практичне значення роботи полягає у можливості використання розробленої системи як допоміжного інструменту для аналізу гімнастичних вправ у тренувальному, навчальному або дослідницькому процесі. Матеріали роботи також можуть бути використані під час вивчення комп'ютерного зору, машинного навчання, аналізу часових послідовностей і розробки прикладних інтелектуальних систем.

Основними обмеженнями системи є залежність від якості відеозапису, ракурсу зйомки, видимості тіла спортсмена, стабільності роботи MediaPipe Pose або MediaPipe Tasks, а також від повноти й збалансованості навчального набору даних. Крім того, система не розраховує E-score і не оцінює якість виконання елементів, тому її результати потребують експертної перевірки.

Отже, результати кваліфікаційної роботи відповідають поставленій меті та завданням. Розроблена система підтверджує можливість автоматизованого аналізу гімнастичних вправ за відеозаписом і формування попередньої оцінки складності, наближеної до D-score. Подальший розвиток доцільно спрямувати на розширення переліку елементів, збільшення навчального набору даних, підтримку інших снарядів, точніше врахування правил FIG, підвищення точності моделі, реалізацію роботи в реальному часі та додавання модуля оцінювання E-score.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Code of points: men's artistic gymnastics 2025-2028. *World Gymnastics*. URL: https://www.gymnastics.sport/publicdir/rules/files/en_1.1%20-%20MAG%20Code%20of%20Points%202025-2028.pdf (date of access: 21.05.2026).
2. Instant replay and control system for sports judging. *World Gymnastics*. URL: https://www.gymnastics.sport/publicdir/rules/files/en_1.2%20-%20Appendix%20to%20the%20Code%20of%20Points%202025-2028.pdf (date of access: 21.05.2026).
3. About Fujitsu. *World Gymnastics*. URL: <https://www.gymnastics.sport/site/pages/legal-fujitsu.php> (date of access: 21.05.2026).
4. Dartfish | Video analysis solutions to improve teams' & athletes' performance. *Dartfish*. URL: <https://www.dartfish.com/> (date of access: 21.05.2026).
5. Video analysis software. *Kinovea*. URL: <https://www.kinovea.org/> (date of access: 21.05.2026).
6. Yan S., Xiong Y., Lin D. Spatial temporal graph convolutional networks for skeleton-based action recognition. *Proceedings of the AAAI conference on artificial intelligence*. 2018. Vol. 32, no. 1. URL: <https://doi.org/10.1609/aaai.v32i1.12328> (date of access: 02.06.2026).
7. Yong Du, Wang W., Wang L. Hierarchical recurrent neural network for skeleton based action recognition. *2015 IEEE conference on computer vision and pattern recognition (CVPR)*, Boston, MA, USA, 7–12 June 2015. 2015. URL: <https://doi.org/10.1109/cvpr.2015.7298714> (date of access: 02.06.2026).
8. Hochreiter S., Schmidhuber J. Long Short-Term Memory. *Neural computation*. 1997. Vol. 9, no. 8. P. 1735–1780. URL: <https://doi.org/10.1162/neco.1997.9.8.1735> (date of access: 02.06.2026).

9. Learning phrase representations using RNN encoder–decoder for statistical machine translation / K. Cho et al. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar. Stroudsburg, PA, USA, 2014. URL: <https://doi.org/10.3115/v1/d14-1179> (date of access: 02.06.2026).
10. Goodfellow I., Bengio Y., Courville A. *Deep Learning*. Cambridge: MIT Press, 2016. URL: <https://www.deeplearningbook.org/> (date of access: 02.06.2026).
11. BlazePose: on-device real-time body pose tracking / V. Bazarevsky et al. *arXiv*. 2020. URL: <https://doi.org/10.48550/arXiv.2006.10204> (date of access: 02.06.2026).
12. MediaPipe: a framework for building perception pipelines / C. Lugaresi et al. *arXiv*. 2019. URL: <https://doi.org/10.48550/arXiv.1906.08172> (date of access: 02.06.2026).
13. Дишук І.В., Гриньов С.А. Інтелектуальний аналіз складності гімнастичних комбінацій засобами комп'ютерного зору. *Радіoeлектроніка та молодь у XXI столітті* : матеріали 30-го Міжнар. молодіж. форуму, 22–24 квітня 2026 р. Харків : ХНУРЕ, 2026. Т. 6. С.40–42.
14. PyTorch: an imperative style, high-performance deep learning library / A. Paszke et al. *arXiv*. 2019. URL: <https://doi.org/10.48550/arXiv.1912.01703> (date of access: 02.06.2026).
15. PyTorch documentation. GRU. *PyTorch*. URL: <https://pytorch.org/docs/stable/generated/torch.nn.GRU.html> (date of access: 02.06.2026).
16. PyTorch documentation. CrossEntropyLoss. *PyTorch*. URL: <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html> (date of access: 02.06.2026).
17. OpenCV - open computer vision library. *OpenCV*. URL: <https://opencv.org/> (date of access: 05.06.2026).

18. Array programming with NumPy / C. R. Harris et al. *Nature*. 2020. Vol. 585, no. 7825. P. 357–362. URL: <https://doi.org/10.1038/s41586-020-2649-2> (date of access: 05.06.2026).
19. Python software foundation. tkinter – Python interface to Tcl/Tk. *Python Documentation*. URL: <https://docs.python.org/3/library/tkinter.html> (date of access: 05.06.2026).
20. Pillow documentation. *Pillow*. URL: <https://pillow.readthedocs.io/> (date of access: 05.06.2026).
21. json – JSON encoder and decoder. *Python Documentation*. URL: <https://docs.python.org/3/library/json.html> (date of access: 05.06.2026).