Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Програмна система для автоматичної сегментації динамічних фрагментів у відеозаписах за допомогою методів машинного навчання

Виконав: студент IV курсу, групи СП-41 спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Вонсович О.П.
(підпис) (прізвище та ініціали)

Керівник _____ Бойко І.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ Гром’як Р.С.
(підпис) (прізвище та ініціали)

Тернопіль 2025

АНОТАЦІЯ

Кваліфікаційна робота бакалавра, виконана Вонсович Олександром Петровичем, студентом групи СП-41 Тернопільського національного технічного університету, присвячена розробці програми для автоматичної сегментації динамічних фрагментів у відеозаписах за допомогою методів машинного навчання. Обсяг роботи становить 85 сторінок, містить 19 рисунків, 2 додатки та список використаних джерел з 22 позицій.

Метою дослідження є створення інструменту, який дозволяє автоматично виявляти та сегментувати динамічні моменти у відео з використанням сучасних архітектур нейронних мереж. У процесі роботи було сформовано датасет з відеофрагментів, виконано його фрагментацію та балансування. Для розв'язання поставленої задачі реалізовано та порівняно дві моделі: тривимірну згорткову нейронну мережу (3D CNN) та трансформер-архітектуру Timesformer.

Оцінку якості моделей проведено за допомогою метрик класифікації, серед яких confusion matrix, macro precision, macro recall, macro F1 score, balanced accuracy, ROC-AUC та precision-recall curve. Додатково здійснено візуальний аналіз результатів сегментації на тестових відео. За результатами порівняння встановлено, що трансформер-модель продемонструвала вищу точність класифікації, здатність адаптивно розпізнавати динамічні рухи навіть поза межами навчального набору даних, та потенціал для подальшого донавчання.

Отримані результати засвідчують доцільність подальшого розвитку системи шляхом розширення навчального датасету, оптимізації моделей та реалізації зручного графічного інтерфейсу для інтерактивного завантаження і обробки відеофайлів.

Ключові слова: відеокласифікація, машинне навчання, комп’ютерний зір, 3D CNN, трансформер, Timesformer, сегментація відео, аналіз динаміки, fine-tuning.

ABSTRACT

The bachelor’s qualification thesis, completed by Vonsovych Oleksandr Petrovych, a student of group SP-41 at Ternopil National Technical University, is dedicated to the development of a program for the automatic segmentation of dynamic fragments in video recordings using machine learning methods. The volume of the work comprises 85 pages, includes 19 figures, 2 appendices, and a list of references containing 22 sources.

The aim of the research is to create a tool that enables the automatic detection and segmentation of dynamic moments in video content by employing modern neural network architectures. In the course of the study, a dataset of video fragments was compiled, followed by its segmentation and balancing. To address the task, two models were implemented and compared: a three-dimensional convolutional neural network (3D CNN) and a transformer-based architecture, Timesformer.

The quality of the models was evaluated using classification metrics, including the confusion matrix, macro precision, macro recall, macro F1 score, balanced accuracy, ROC-AUC, and the precision-recall curve. Additionally, a visual analysis of segmentation results on test videos was conducted. The comparative results demonstrated that the transformer model achieved higher classification accuracy, an ability to adaptively recognize dynamic movements even beyond the boundaries of the training dataset, and promising potential for further fine-tuning.

The obtained results confirm the feasibility of further system development through the expansion of the training dataset, optimization of the models, and the implementation of a convenient graphical interface for interactive uploading and processing of video files.

Keywords: video classification, machine learning, computer vision, 3D CNN, transformer, Timesformer, video segmentation, motion analysis, fine-tuning.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CNN — Convolutional Neural Network, згорткова нейронна мережа, один із різновидів штучних нейронних мереж.

Fine-tuning — метод удосконалення попередньо навченої моделі на новому наборі даних для конкретного завдання. Полягає в частковому донавчанні моделі, адаптуючи її вагові коефіцієнти до нових умов, зазвичай зі збереженням знань, отриманих на базовому датасеті.

Трансформер — архітектура нейронних мереж, яка базується на механізмі самоуваги (self-attention) і дозволяє ефективно обробляти послідовності даних довільної довжини.

Датасет — упорядкована сукупність даних, що використовується для навчання, перевірки та тестування моделей машинного навчання.

ВСТУП

Відеоаналітика є одним із найдинамічніших напрямів сучасних інформаційних технологій, який активно впроваджується у сфери безпеки, спорту, транспорту, медіаіндустрії та побутового користування. Зокрема, поширення екшн-камер, таких як GoPro, зумовило стрімке зростання обсягів відеоматеріалів, що потребують аналізу, систематизації та швидкої обробки. Це актуалізує проблему розробки інструментів для автоматичного аналізу відео, зокрема виявлення найбільш динамічних або спокійних фрагментів, що дозволяє оптимізувати процес монтажу, створення відеозвітів, спортивної аналітики та стрімінгового контенту.

Незважаючи на значні успіхи у сфері комп'ютерного зору та машинного навчання, більшість існуючих систем орієнтовані переважно на детекцію об'єктів або відстеження руху конкретних елементів сцени. Натомість сегментація відео за динамічністю залишається менш розвиненим напрямом, а з поширенням побутових екшн-камер збільшується зростає потреба саме в сегментації відео за динамічністю. Відповідно до поняття «video content analysis», роль автоматичного поділу відеоматеріалів на умовно значущі сегменти є однією з основних функціональностей сучасних систем для розумного аналізу відео. Окрім класичних завдань, таких як виявлення руху чи об'єктів, дослідники відзначають, що поділ відео на динамічні й статичні фрагменти дозволяє ефективно створювати відеозвіти, а також оптимізувати ресурси при обробці великих обсягів даних, зокрема в умовах реального часу (real-time segmentation).

Нові підходи, що застосовуються у сегментації динамічних і статичних об'єктів, дедалі частіше поєднують глибинні нейронні мережі, зокрема 3D-CNN. Наприклад, за допомогою поєднання просторових і часових ознак у рамках 3D-згорток дослідники домоглися суттєвого підвищення точності сегментації динамічних об'єктів без необхідності ручного маркування.

Таким чином, автоматична сегментація відео фрагментів належить до одного з ключових напрямів сучасної відеоаналітики, має практичне значення в багатьох галузях (безпека, спорт, медіа) та відкриває нові можливості для інтеграції із технологіями моніторингу, стримінгу і автоматизованого монтажу.

Метою роботи є створення системи автоматичного аналізу відеозаписів для сегментації їх на динамічні та спокійні фрагменти на основі методів машинного навчання, що дозволить автоматизувати попередню обробку відеоматеріалів і спростити подальшу роботу з ними.

Для досягнення поставленої мети було проведено аналіз сучасних методів обробки відеоданих, визначено ефективні підходи до сегментації за динамічністю, зібрано та підготовлено навчальний набір відеофрагментів, реалізовано конвеєр їх попередньої обробки, розроблено модель згорткової тривимірної нейронної мережі для класифікації фрагментів відео за рівнем динамічності, а також виконано її навчання, тестування та оцінку якості роботи. Також було виконано порівняння цієї моделі з файнтюненим візуальним трансформером з метою визначення кращої архітектури для цієї заадчі. Окрему увагу приділено реалізації скрипта для автоматичного розбиття відео на сегменти з візуалізацією результатів та подальшим аналізом одержаних експериментальних даних.

Об’єктом дослідження є процес автоматичного аналізу відеозаписів, знятих за допомогою екшн-камер. Предметом дослідження виступають методи та програмні засоби сегментації відеозаписів за ознаками динамічності із застосуванням технологій глибокого машинного навчання.

У межах роботи використано методи аналізу літературних джерел та наукових публікацій з тематики відеоаналітики, методи попередньої обробки відеоданих, включно з ресемплінгом частоти кадрів та нормалізацією роздільної здатності, а також методи побудови, навчання та оцінювання моделей машинного навчання на основі 3D згорткових нейронних мереж та трансформерів. Оцінку якості проведено за загальноприйнятими метриками класифікації, а результати роботи моделі візуалізовано для полегшення подальшого аналізу.

Наукова новизна роботи полягає у застосуванні згорткових тривимірних нейронних мереж та моделей трансформерної архітектури для сегментації відеозаписів за рівнем динамічності без залучення сторонніх сенсорів (акселерометрів, GPS тощо) та створенні оригінального навчального набору даних із реальних побутових відеозаписів, виконаних за допомогою екшн-камер.

Практичне значення роботи полягає у можливості застосування розробленої системи для автоматичної попередньої обробки великих обсягів відеоматеріалів, що дозволяє суттєво скоротити час на їх перегляд і монтаж. Результати роботи можуть бути корисними блогерам, спортсменам, операторам екшн-камер та контентмейкерам, які працюють із відеозаписами екстремального або динамічного характеру. Крім того, запропонований програмний комплекс може використовуватися у навчальному процесі для демонстрації можливостей сучасних методів машинного навчання під час роботи з мультимедійними даними.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження сучасних технологій класифікації

Автоматична сегментація відео за рівнем динамічності полягає у виділенні з послідовності кадрів тих ділянок, де спостерігається висока активність руху або значущі події. Це важливо для задач відеосумаризації та створення «хайлайтів» – тобто коротких роликів з найцікавіших моментів. У контексті відеосумаризації переважно обирають ключові кадри чи фрагменти, що відповідають найважливішим подіям відео [3].

Для реалізації таких завдань широко використовують методи машинного навчання: згорткові нейронні мережі (CNN), рекурентні мережі (LSTM) та, що особливо актуально в останні роки, трансформерні архітектури [3]. Ці методи дозволяють моделювати просторово-часові залежності у відео й навчатися розрізняти статичні та динамічні сегменти.

Важливу роль у моделюванні просторово-часових ознак відео відіграють 3D-згорткові нейронні мережі. На відміну від традиційних 2D-CNN, які обробляють поодинокі кадри, 3D-CNN застосовують 3D-згортки, що охоплюють послідовність кадрів. Це дає змогу одночасно враховувати і просторову інформацію, і кінематику руху об’єктів у відео [4].

Наприклад, архітектура C3D (Tran et al., 2015) продемонструвала, що 3D-мережі ефективно вчаться одночасно моделювати «зовнішній вигляд та рух» відео [5].

У роботах з обробки спортивних відео показано, що використання 3D-CNN для виділення «гарячих» сцен (завершень атак, голів тощо) забезпечує вищі показники відновлення та точності в порівнянні з 2D-CNN.

Однак обчислювальна складність таких мереж велика: для глибоких 3D-CNN потрібно значно більше пам’яті та часу на тренування, а розмір моделей зростає квадратично порівняно з 2D-аналогами.

Поряд із згортковими мережами новим трендом стало застосування відеотрансформерів. На відміну від CNN, трансформери повністю базуються на механізмі self-attention, що дозволяє моделі захоплювати віддалені залежності у відеопослідовності. Наприклад, модель ViViT (Arnab et al., 2021) обробляє відео як послідовність спато-часових токенів і досягає кращих результатів на стандартних наборах даних (Kinetics, Something-Something тощо) у порівнянні з багатьма 3D-CNN [6].

Загалом було показано, що «суто-трансформерні» моделі відео здатні перевершувати традиційні 3D-згортки завдяки кращому моделюванню глобального контексту. Також запроваджено різні варіації відеотрансформерів (TimeSformer, Video Swin Transformer), які розділяють увагу за просторовою та часовою осями для підвищення ефективності. Окремі дослідницькі роботи використовують трансформери саме для завдань виділення динамічних сегментів. Наприклад, у проєкті MH-DETR використано DETR-подібний трансформер для одночасного локалізування «моментів» і оцінювання їх значущості [7]. Експерименти MH-DETR на різних наборах (QVAud, ActivityNet, TVSum тощо) показали, що такі моделі поступово вийшли на лідируючі позиції у завданнях виявлення хайлайтів. Таким чином, трансформери надають перспективний підхід до сегментації за динамікою завдяки своїй здатності аналізувати довгі відрізки відео з однією мережею.

Окрім згаданих методів, застосовують також гібридні підходи: наприклад, двопотокові моделі, що одночасно аналізують зображення та оптичний потік, чи поєднання CNN з LSTM/GRU, що моделюють часову послідовність кадрів. Такі методи дозволяють підсилити виявлення руху та врахувати довготривалі залежності, проте мають власні обмеження. Графові моделі та інші архітектури (наприклад, Graph Attention) також пропонувалися для ускладнених рухових патернів. Немає єдиного стандарту, і дослідники вибирають підхід залежно від доступності даних та специфіки застосування.

У патентній літературі та прикладних рішеннях також відображено розвиток цієї області. Наприклад, описано метод автоматичного генерування відеохайлайтів

зі спортивних трансляцій, який включає екстракцію ознак, сегментацію потоків кадрів і класифікацію подій [8]. Автори патенту пропонують розбивати вхідне відео на сегменти, ідентифікувати події в кожному сегменті та визначати, чи є подія «цікавою» (хайлайтом).

Щодо комерційних продуктів, компанія GoPro реалізувала у своєму редакторі Quik функцію автоматичної генерації роликів «Highlights». У пов'язаних документах згадується використання мережі з увагою в часі (temporal attention network) для автоматичного виявлення «важливих подій» у відеокліпі [9]. Таким чином, навіть прикладні системи, призначені для звичайних користувачів, опираються на алгоритми машинного навчання для пошуку динамічних сцен.

На основі проведеного аналізу формулюється науково-технічна задача: створити ефективний алгоритм (або модель) для автоматичної сегментації відео за рівнем динамічності. Це передбачає побудову відео-представлення та класифікатора, здатних точно виявляти активні чи «цікаві» ділянки у різнорідних відеопотоках. Невирішеними проблемами лишаються, зокрема, такі: висока розрахункова складність глибоких мереж проти обмежених ресурсів у реальному застосуванні; залежність результатів від жанру відео, умов зйомки та відношення користувача до «цікавості» фрагмента; брак розмічених даних для навчання універсальних моделей; а також суперечливі результати різних підходів (наприклад, 3D-CNN добре вловлюють рух, але можуть пропускати контекст, тоді як трансформери важче тренувати на малих наборах).

Метою дослідження є розробка і реалізація машинного алгоритму виділення динамічних сегментів у відео. Зокрема, планується вивчити модель машинного навчання яка б забезпечувала високу точність при помірних ресурсах обчислень. Для поставленої задачі необхідно обґрунтувати вибір вхідних даних (довжина кліпа, формат ознак) та характеристик моделі (топологія мережі, розмір та кількість шарів) на основі літературних даних та експериментальних оцінок. Таким чином, завдяки синтезу сучасних наукових результатів і практичних прикладів обґрунтовується реалізація в роботі нового алгоритму, спрямованого на точне виявлення динаміки у відео з використанням машинного навчання.

1.2 Обґрунтування напрямку дослідження

Після попереднього аналізу сучасних досліджень у галузі відеоаналітики, можна виокремити кілька основних напрямів, які найчастіше застосовуються для сегментації відео за рівнем динамічності.

Традиційно однією з найпростіших і найшвидших категорій методів залишаються класичні алгоритми комп'ютерного зору, зокрема оптичний потік, аналіз ключових кадрів або обчислення таких ручних ознак, як середня швидкість руху чи енергія потоку. Основними перевагами цих методів є їхня швидкість і невисокі вимоги до ресурсів. Водночас такі підходи демонструють нестабільність у випадках шумів або ривків камери, а також не здатні враховувати контекст сцени.

Другий напрям — це поєднання згорткових нейронних мереж (CNN) із рекурентними архітектурами, такими як LSTM (Long Short-Term Memory) чи GRU (Gated Recurrent Unit). Така комбінація дозволяє екстрагувати ознаки з окремих кадрів за допомогою CNN, а подальший аналіз послідовності виконується за допомогою рекурентної мережі. Подібні моделі мають перевагу у вигляді можливості використовувати попередньо навчені CNN, що спрощує процес адаптації до нових даних. Однак суттєвими обмеженнями залишаються уповільнене оброблення, складність навчання рекурентних компонентів і обмежене охоплення просторового контексту.

Особливий інтерес викликають 3D Convolutional Neural Networks, які за рахунок тривимірних згорток дозволяють одночасно аналізувати як просторові, так і часові компоненти відео. Це природно відповідає завданню сегментації за динамічністю, оскільки дає змогу враховувати рух у просторі й часі в межах одного фрагмента. Попри значні обчислювальні витрати та чутливість до головного руху камери, ці моделі забезпечують високу гнучкість завдяки використанню механізмів типу AdaptiveAvgPool3d і добре працюють на невеликих за обсягом датасетах середньої тривалості.

Сучасною альтернативою стають трансформерні архітектури, що застосовують self-attention для аналізу відео як послідовності просторово-часових токенів. Основною їхньою перевагою є здатність вловлювати глобальні залежності у відеопотоках без обмежень на порядок кадрів. Проте ці моделі є вкрай ресурсоємними, вимагають значних обсягів даних для повноцінного навчання та зазвичай потребують файнтюнінгу попередньо навченої моделі на специфічному датасеті для виконання поставленої задачі.

З огляду на обмеженість ресурсів і доступного обсягу даних, найбільш обґрунтованим рішенням є використання на першому етапі 3D CNN як перевіреного підходу, що показав високу ефективність у суміжних завданнях, та порівняння його продуктивності з відеотрансформерами, які становлять перспективний напрям подальших досліджень.

1.3 Методика вирішення задачі

Одним із ключових етапів є визначення формату даних, адже він істотно впливає на ефективність обраних моделей. Для 3D CNN зазвичай застосовують відеофрагменти фіксованого розміру з однаковою кількістю кадрів, тоді як трансформери здатні працювати з послідовностями токенів, сформованих з кадрів або їхніх патчів.

Наступним елементом методики є вибір архітектурних рішень. У межах проєкту планується реалізувати багатошарову конфігурацію 3D CNN, зокрема із поступовим нарощуванням кількості каналів і використанням адаптивного pooling. Для трансформерів перевага надається сучасним архітектурам, що використовують механізми self-attention для моделювання глобальних просторово-часових залежностей, уникаючи повного тренування моделі, надаючи перевагу fine-tuning.

Для об'єктивного порівняння продуктивності моделей буде визначено єдиний набір метрик, серед яких точність класифікації на тренувальному та валідаційному

датасеті, а також буде виконано суб’єктивну оцінку якості сегментації тестувальних відеороликів. Це дасть змогу забезпечити всебічну оцінку якості роботи кожного з підходів.

Окремо буде розроблено узгоджену процедуру тестування, яка включатиме попередню обробку відеоданих, формування вибірок для тренування, валідації та тестування, а також однаковий підхід до розподілу даних.

Завершальним етапом стане проведення порівняльних експериментів. Обидва типи моделей будуть навчатися та тестуватися на єдиному датасеті із застосуванням однакових процедур оцінювання. Порівняльний аналіз отриманих результатів дозволить обґрунтувати доцільність застосування кожного з підходів в умовах наявних обмежень і стане основою для рекомендацій щодо використання 3D CNN чи трансформерів для задачі сегментації відеофрагментів за динамічністю.

2 РОЗРОБКА ПРОГРАМНОГО РШЕННЯ

2.1 Збір та підготовка тренувальних даних

Для побудови системи автоматичного аналізу динамічності відеозаписів було сформовано власний тематичний датасет, що повністю відповідає специфіці поставленої задачі.

2.1.1 Метод збору тренувальних даних

Джерелом даних слугували особисті відеозаписи, зняті за допомогою екшн-камери GoPro Hero 11 під час реальних поїздок на гірському велосипеді. Такий вибір дозволив отримати матеріали зі стабільною якістю зображення, типовими для екшн-відео умовами зйомки та різноманітними сценаріями руху камери.

Зйомка здійснювалася у режимі HyperView з вертикальним кріпленням камери, що забезпечувало максимальний кут огляду та реалістичну передачу зміни ракурсу під час руху. Використання роздільної здатності 4K (2160×3840) та частоти 60 кадрів на секунду дозволяло зберегти необхідний рівень деталізації та плавності для подальшого аналізу динамічності сцени. Окрему увагу приділяли увімкненій стабілізації, адже саме її робота є одним з важливих факторів для тестування здатності моделей відрізняти реальний рух від компенсації тремтіння.

Всі відеозаписи виконувалися у лісистій місцевості у світлу пору доби, а деякі з них фіксували одну й ту ж локацію у різні сезони. Це дало змогу протестувати моделі на стійкість до зміни візуальних умов, таких як кольори листя, тіньове середовище та структура ландшафту.

В рамках роботи було визначено дві основні категорії для класифікації фрагментів: dynamic (динамічні) — сцен із активним рухом, зміною ракурсу та рельєфу місцевості, і calm (спокійні) — сцен з мінімальними коливаннями камери

або повільним плавним рухом. Анотація даних виконувалася вручну: було особисто переглянуто усі відеозаписи, розбито їх на фрагменти тривалістю від кількох секунд до кількох хвилин та присвоєно кожному фрагменту відповідну мітку. Особливої уваги потребували динамічні фрагменти, оскільки в окремих активних ділянках могли зустрічатися короткі періоди спокійного руху або зупинки. Такі ділянки видалялися, щоб забезпечити максимальну однорідність сцен за характером руху.

Після завершення процесу анотації всі відеофрагменти були експортовані у вертикальному форматі 1080×1920 пікселів зі зменшеним бітрейтом до 20 Мбіт/с для оптимізації обсягу даних, зберігаючи при цьому високу частоту кадрів — 60 fps. Це дозволило суттєво спростити зберігання та обробку даних без помітної втрати візуальної якості.

На даному етапі датасет містив 270 секунд динамічних фрагментів і 1099 секунд спокійних, що відображає реальний дисбаланс між тривалістю активних та спокійних моментів під час велопоїздок. Ця особливість обов’язково враховувалась на подальших етапах формування тренувальних та валідаційних вибірок.

Окремою перевагою сформованого датасету є його повна відкритість — всі відеоматеріали є авторськими та можуть бути використані у відкритому доступі для наукових чи навчальних цілей. Водночас, наявність єдиного джерела даних із фіксованими параметрами зйомки дозволяє уникнути варіативності, притаманної відкритим датасетам, що позитивно впливає на стабільність моделювання, але водночас обмежує різноманітність сценаріїв.

2.1.2 Фрагментація датасету

Після первинного збору та анотації відеоматеріалів постала необхідність у підготовці даних до навчання моделі машинного навчання. Основною метою цього етапу було забезпечення єдиної структури вхідних даних: послідовностей кадрів

однакової тривалості, а також зменшення дисбалансу між класами та підготовка вибірки до ефективного аналізу часових залежностей.

Процес фрагментації полягав у нарізанні довгих відеозаписів на фрагменти фіксованої тривалості. На початкових етапах було обрано довжину фрагмента в 64 кадри, пропускаючи кожен другий кадр для досягнення ефективної частоти 30 fps (що відповідало тривалості приблизно 2,14 секунди). Такий підхід дозволив зберегти загальну плавність сцени, зменшити обсяг даних та оптимізувати навантаження на обчислювальні ресурси без необхідності повного ресемплінгу відео.

Фрагментація здійснювалася програмним шляхом: відеофайли попередньо конвертувалися у послідовності окремих кадрів (зображень), після чого формувалися послідовності заданої довжини. Фрагменти, які не відповідали обраній тривалості (наприклад, залишки відео на кінцях роликів), відбраковувалися, оскільки неповні послідовності могли викликати проблеми при навчанні моделі та спотворювати результати.

З метою оптимізації подальшої роботи всі сформовані фрагменти зберігалися у вигляді шляхів до кадрів у вигляді списків разом із відповідними мітками класів.

2.1.3 Балансування датасету

Балансування датасету стало одним із критично важливих етапів підготовки даних, оскільки від цього безпосередньо залежала здатність моделі адекватно розпізнавати обидві категорії відеофрагментів. Після первинної анотації було виявлено суттєвий дисбаланс: кількість спокійних фрагментів більш ніж удвічі перевищувала кількість динамічних. Така ситуація створювала ризик навчання моделі на зміщеній вибірці, що могло призвести до упередженості у бік переважаючого класу. Модель в такому разі схилялася б до пріоритетного передбачення класу зі спокійними сценами, і навіть за достатньої точності на

загальному наборі даних не справлялася б з коректною класифікацією динамічних моментів, які в контексті поставленої задачі є більш інформативними.

На етапі вибору методу балансування було розглянуто кілька можливих підходів. Одним із найпростших варіантів є випадковий андерсемплінг (undersampling) — зменшення кількості фрагментів домінуючого класу шляхом випадкового видалення частини даних. Однак цей метод має суттєвий недолік: він призводить до втрати потенційно цінної інформації, адже частина наявних спокійних фрагментів просто не потрапляє до тренувальної вибірки. В умовах обмеженого обсягу зібраного датасету та потреби максимально використати доступні дані цей підхід виявився неприйнятним.

Альтернативним методом є апсемплінг мінорного класу шляхом дублювання існуючих динамічних фрагментів. Проте просте дублювання без внесення змін до даних збільшує ризик оверфіту (коли модель добре справляється на тренувальній вибірці, але погано на нових даних), оскільки модель може надмірно пристосуватися до повторюваних фрагментів і втратити здатність до узагальнення. До того ж дублювання не збагачує вибірку новими варіаціями, що особливо важливо для задачі аналізу відеодинаміки, де навіть незначні зміни у русі камери чи освітленні можуть суттєво впливати на сприйняття сцени.

З огляду на це було прийнято рішення використати метод аугментації (зміни) даних для синтетичного збільшення кількості динамічних фрагментів. Цей підхід передбачав застосування різноманітних випадкових перетворень до наявних динамічних сцен. Зокрема, використовувалися зміна яскравості (рисунок 2.1), випадкові повороти (рисунок 2.2) та горизонтальне дзеркальне відображення (рисунок 2.3). Важливим аспектом при виборі трансформацій було збереження характеру сцени — усі зміни не мали спотворювати змістовний компонент відеофрагмента чи змінювати його категорію. Аугментація дозволила значно урізноманітнити динамічні фрагменти, зробити вибірку більш різноманітною та водночас збалансувати кількість фрагментів у кожному класі без втрати цінної інформації.

Рисунок 2.1 — Випадкова аугментація яскравості

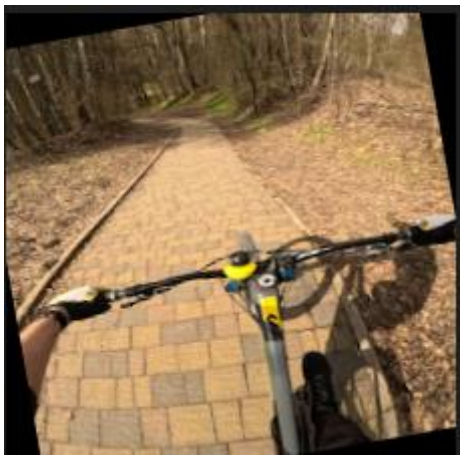


Рисунок 2.2 — Випадкова аугментація повороту

Рисунок 2.3 — Випадкова аугментація відображення

Завдяки цьому балансуванню було досягнуто без необхідності скорочення кількості спокійних фрагментів. Водночас тренувальна вибірка отримала нові унікальні варіації динамічних сцен, що сприяло покращенню стійкості моделі до незначних змін у відеоряді. Порівняно з андерсемплінгом, метод аугментації виявився значно ефективнішим, оскільки дозволив зберегти повний обсяг початкових даних та додатково розширити інформаційне різноманіття вибірки. А порівняно зі звичайним дублюванням, цей підхід мінімізував ризик оверфіту завдяки різноманітності згенерованих варіацій.

Отримана збалансована вибірка була використана для навчання і тестування моделей глибокого навчання. Це дозволило уникнути зміщеності моделі в бік домінуючого класу та забезпечити об’єктивну оцінку якості класифікації як для спокійних, так і для динамічних відеофрагментів.

2.1.4 Характеристика фінальних тренувальних даних

Після завершення процесів анотації, фрагментації та балансування був сформований фінальний датасет, придатний для навчання моделей глибокого навчання. Остаточна структура даних була організована у вигляді директорій, що відповідали класам відеофрагментів, що спрощувало процес завантаження даних та їх обробку під час тренування.

Датасет складався з двох основних категорій: “dynamic” та “calm”. Для кожного класу була створена окрема папка, в яку поміщалися відповідні відеофрагменти. Усі відеофрагменти зберігалися у вигляді послідовностей окремих зображень (кадрів) у форматі .png. Кожен відеофрагмент зберігався у власній папці, ім'я якої відповідало унікальному ідентифікатору кліпу, наприклад dynamic_0001, calm_0268 тощо.

Структура директорій мала такий вигляд як на рисунку 2.4:

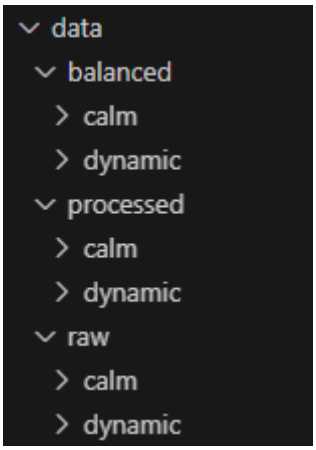


Рисунок 2.4 — Структура папок тренувальних даних

Розмір кожного зображення становив 224×224 пікселів, проте при тренування розмір зменшувався до 112×112 для збереження відеопам'яті. Такий формат даних відповідає типовій розмірності вхідних даних моделей і зберігало візуальну інформацію на кадрах без значної пікселяції. Детальні кількісні характеристики даних наведено на рисунку 2.5.

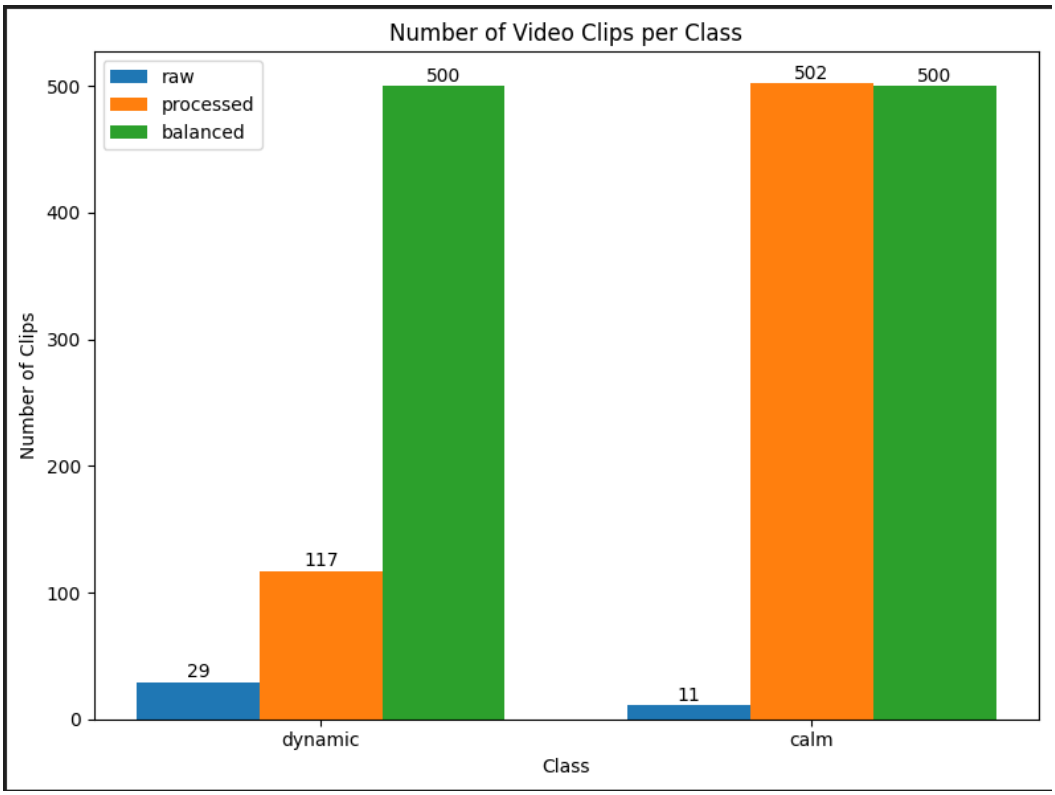


Рисунок 2.5 — Характеристики даних до і після обробки

Довжина кожного відеофрагмента була стандартизована та становила 64 кадри. При частоті 30 fps це відповідало приблизно 2,14 секунд відео. Таке значення було обрано як оптимальний компроміс між обсягом контексту для аналізу динамічності сцени та допустимим обсягом оперативної пам’яті під час навчання.

Для навчання моделі датасет був поділений на тренувальну, валідаційну та тестувальну вибірки у співвідношенні 80:10:10. Відповідно, тренувальна вибірка містила 800 фрагментів (400 у кожній категорії), а валідаційна та тестувальна по 100 фрагментів відповідно (50 у кожному класі), разом 1000 кліпів.

Для забезпечення швидкого доступу до даних під час навчання, було використано підготовлений скрипт на Python із застосуванням бібліотек torchvision та opencv, який завантажував послідовності кадрів, формував тривимірні тензори розміром [64, 3, 112, 112] для кожного фрагмента (кількість кадрів, колірні канали, висота, ширина) та передавав їх до моделі.

2.2 Тренування моделей

Тренування моделей відбувалося на особистому ПК з використанням GPU з виділеними 12ГБ відеопам’яті, чого вистачало для даної задачі без серйозних обмежень.

2.2.1 Тренування 3D CNN

В першу чергу для задачі класифікації відеофрагментів за динамічністю було обрано модель на основі 3D згорткової нейронної мережі (3D CNN). Головна перевага 3D CNN полягає в її здатності одночасно обробляти просторову

(всередині кадру) та часову (між кадрами) інформацію, що критично важливо для аналізу відео. Також згідно з аналізованих джерел, ці моделі дають хорошу якість класифікації з відносно невеликими вимогами до ресурсів та тренувальних даних.

Вхідні дані мали формат тензора розміру (C, T, H, W), де:

- C = 3 — кількість кольорових каналів (RGB)
- T = 64 — кількість послідовних кадрів у кліпі (в залежності від експерименту)
- H = 112, W = 112 — просторові розміри кадрів

Кожен відеофрагмент представляв собою короткий ролик з послідовних кадрів, отриманих із реального відео поїздок на гірському велосипеді.

Обрана архітектура 3DCNN має компактну структуру та дозволяє ефективно тренувати модель на середніх об’ємах даних без потреби у величезних обчислювальних ресурсах.

Архітектуру моделі описано у лістингу 2.6.

```
class Simple3DCNN(nn.Module):
    def __init__(self):
        super(Simple3DCNN, self).__init__()
        self.features = nn.Sequential(
            nn.Conv3d(3, 16, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool3d(kernel_size=2),

            nn.Conv3d(16, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool3d(kernel_size=2),

            nn.Conv3d(32, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.AdaptiveAvgPool3d((1, 1, 1))
        )
        self.classifier = nn.Linear(64, 2)

    def forward(self, x):
```

```
x = self.features(x)
x = x.view(x.size(0), -1)
x = self.classifier(x)
return x
```

Лістинг 2.6 — Архітектура 3D CNN моделі

Модель має низку характерних архітектурних особливостей, що визначають її здатність ефективно обробляти відеодані та здійснювати класифікацію на дві категорії: dynamic та calm. Основою мережі є три послідовно розташовані згорткові блоки, кожен з яких складається зі згорткового шару та функції активації. Кількість фільтрів у цих шарах поступово зростає: перший блок містив 16 фільтрів, другий — 32, а третій — 64. Це дозволяло на початкових рівнях захоплювати простіші локальні просторово-часові патерни, а на більш глибоких — формувати складніші узагальнені ознаки.

На завершальному етапі мережа містила лінійний класифікаційний шар, що виконув функцію бінарного передбачення. Він приймав на вхід агреговані ознаки з глобального пулінгу та здійснював поділ відеофрагментів на два класи — dynamic або calm, відповідно до характеру руху в кадрі.

Після тренування на 50 епох, графік loss (рисунок 2.7) та графік accuracy (рисунок 2.8) показують що модель активно навчалася приблизно до 30 епохи, після чого досягла плато на відмітці приблизно 87% точності.

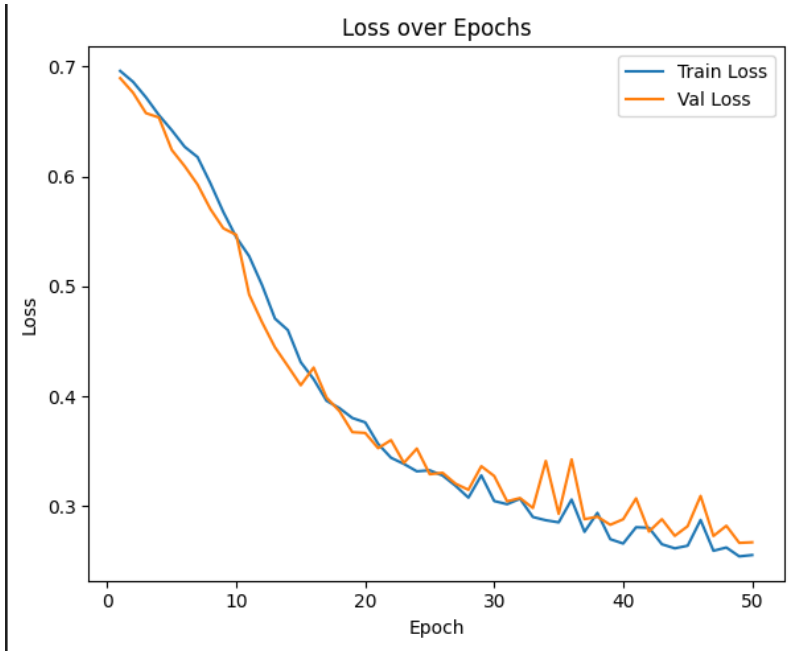


Рисунок 2.7 — Графік loss тренування 3DCNN

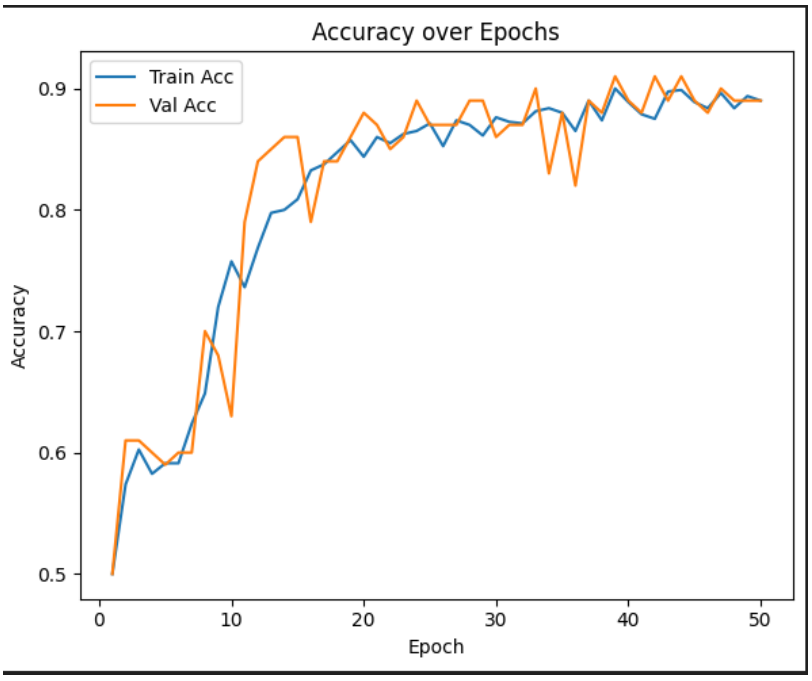


Рисунок 2.8 — Графік accuracy тренування 3DCNN

2.2.2 Тренування моделі трансформер-архітектури

Для розв’язання завдання класифікації за динамічністю було також обрано сучасну архітектуру Timesformer. Трансформер було обрано завдяки його здатності ефективно захоплювати глобальні просторово-часові залежності, що є ключовим для завдань відеоаналізу. Метою використання трансформера є порівняння його якісті з простішою згортковою моделлю, визначаючи чи наявний датасет є достатнім для більш складної архітектури.

На відміну від згорткових мереж, які працюють з локальними патернами, трансформер дозволяє моделювати взаємодії між кадрами та областями зображення на великих відстанях. Це є особливо цінним для аналізу велосипедних маршрутів, де рух об’єктів у кадрі міг мати різні масштаби та швидкість.

Також Timesformer не мав обмежень по довжині вхідної послідовності завдяки своїй архітектурі. Це дало змогу експериментувати з різними тривалостями кліпів без потреби переробки моделі. Проте більша довжина кліпів вимагала квадратично більше ресурсів відеопам’яті, що змусило використовувати 64 кадри, як для 3D CNN.

Було використано модель MCG-NJU/videomae-base, яка була попередньо навчена на великій кількості відеоданих. Це дозволило зменшити потребу у великих об’ємах власних даних для тренування та зосередитись на fine-tuning під конкретну задачу.

Було сформовано вхідні дані у вигляді тензорів розміру (B, C, T, H, W), де:

- B — розмір батчу (4 для оптимального завантаження GPU)
- C = 3 — кількість кольорових каналів (RGB)
- T = 64 — кількість кадрів у кліпі
- H = 112, W = 112 — розмір кадрів після масштабування

Для тонкого налаштування було обрано наступні параметри:

- кількість епох: 10
- batch size: 4 (для роботи на одній відеокарті з 12 ГБ VRAM)
- learning rate: 5e-5

- optimizer: AdamW
- weight decay: 0.01
- fp16: було увімкнено для пришвидшення обчислень на GPU та зниження споживання пам’яті

Згідно рисунка 2.9 з графіком loss та рисунка 2.10 з графіком accuracy, валідаційні метрики моделі після перших епох досягли плато в районі 90-95%.

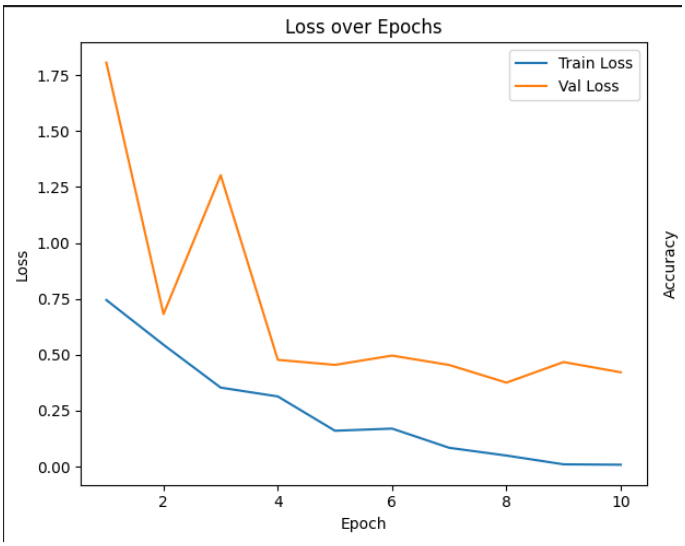


Рисунок 2.9 — Графік loss тренування Timesformer

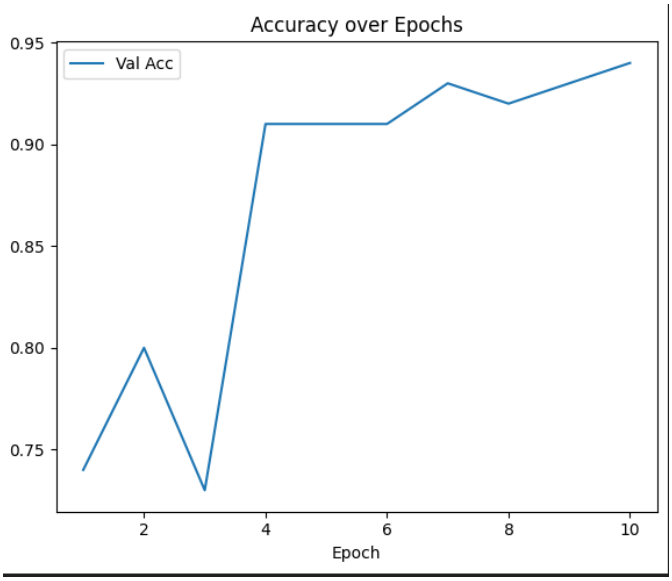


Рисунок 2.10 — Графік валідаційного accuracy тренування Timesformer

2.3 Евалюація моделей

Оцінювання якості моделей глибокого навчання для задач класифікації відеофрагментів не обмежується лише точністю (accuracy). Для збалансованого та коректного аналізу необхідно використовувати набір взаємодоповнюючих метрик, які дозволяють виявити як загальну продуктивність, так і особливості поведінки моделі на різних класах, враховувати дисбаланс даних та аналізувати її практичну корисність у реальних умовах.

2.3.1 Метрики якості моделей

Першою метрикою що була використана для евалюації є матриця неточностей (confusion matrix). Вона відображає кількість правильних і неправильних класифікацій для кожного класу, показуючи скільки відеофрагментів кожної категорії було віднесено до того чи іншого класу. Це дозволяє побачити не лише загальну кількість помилок, а й типи цих помилок — наприклад, наскільки часто спокійні фрагменти помилково класифікуються як динамічні.

На основі цієї матриці формується classification report, який об’єднує кілька ключових метрик для кожного класу: precision, recall, F1-score і support (кількість зразків цього класу в тестовій вибірці). Це дозволяє оцінити модель у розрізі кожної категорії, а не лише в середньому по всій вибірці.

В задачах з потенційно нерівномірним розподілом класів доцільно використовувати Macro Precision, Macro Recall та Macro F1 Score. На відміну від звичайних усереднених значень, які можуть бути зміщені на користь більш

численного класу, ці метрики обчислюють precision, recall і F1 для кожного класу окремо, а потім обчислюють їх середнє арифметичне без урахування ваг класів.

Ще однією метрикою, яка дозволяє контролювати баланс моделі в умовах потенційного дисбалансу чи різної складності класифікації класів, є Balanced Accuracy. Вона обчислюється як середнє значення recall для кожного класу. Ця метрика є більш надійною, ніж звичайна accuracy у випадках, коли одна з категорій може бути переоцінена моделлю, навіть за рівної кількості даних, через різну складність ознак, які характеризують спокійні та динамічні фрагменти.

Для оцінки поведінки моделі в умовах зміни порогу класифікації використовуються ROC-крива (Receiver Operating Characteristic curve) та precision-recall крива. ROC-крива дозволяє візуалізувати співвідношення між чутливістю (True Positive Rate) та специфічністю (False Positive Rate) для різних значень порогу. Це дає можливість визначити, наскільки добре модель відрізняє класи незалежно від вибраного порогу, що корисно, наприклад, коли важливіше контролювати кількість хибнопозитивних класифікацій динамічних фрагментів. Precision-recall крива, у свою чергу, особливо інформативна в умовах несиметричного класового балансу або коли позитивний клас є більш цінним. Для подальших застосувань системи (на зразок сегментації трейлових відео для пошуку небезпечних або активних ділянок) важливо, щоб модель максимально точно виявляла саме динамічні фрагменти, навіть ціною зниження recall для спокійних.

Окремо варто виділити суб’єктивний візуальний аналіз сегментованого відео. Це якісна метрика, яка не має числового вираження, але є критично важливою для відеозадач. Навіть при високих формальних показниках precision та recall модель може систематично помилятися на певних типах сцен або демонструвати нестабільну поведінку в послідовності класифікацій фрагментів. Візуальний перегляд сегментованого відео дозволяє оцінити, наскільки класифікаційні межі моделі узгоджуються з реальним сприйняттям відеоряду людиною, наскільки плавно і логічно змінюються категорії в часі, а також виявити потенційні слабкі місця моделі, які не проявляються через суто числові метрики.

Таким чином, використання комплексного набору метрик — від обчислюваних на основі матриці неточностей до кривих ROC та precision-recall — у поєднанні з візуальним аналізом дозволяє максимально повно оцінити якість моделі.

2.3.2 Метрики 3D CNN

Проведений аналіз метрик тривимірної згорткової нейронної мережі (3D CNN) для задачі бінарної класифікації відеофрагментів дозволив отримати низку кількісних показників, що відображають якість моделі на валідаційній вибірці.

Матриця неточностей на рисунку 2.11 показала, що модель правильно класифікувала 42 фрагменти класу calm і 46 фрагментів класу dynamic, водночас помилилася на 8 і 4 прикладах відповідно. Цей результат вказав на приблизно симетричний розподіл помилок між класами, що свідчить про відсутність значного зміщення моделі в бік одного з них. Помірно вища кількість помилок для класу calm підтвердила гіпотезу про неточності моделі при різких рухах головою, які впливають на класифікацію таких моментів як динамічних.

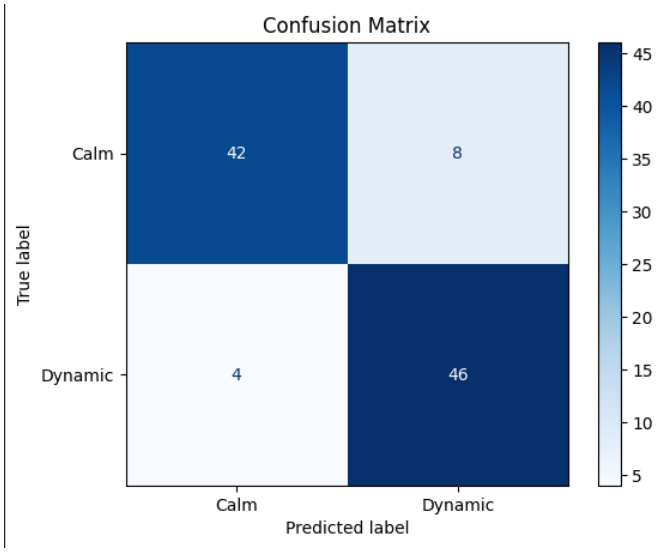
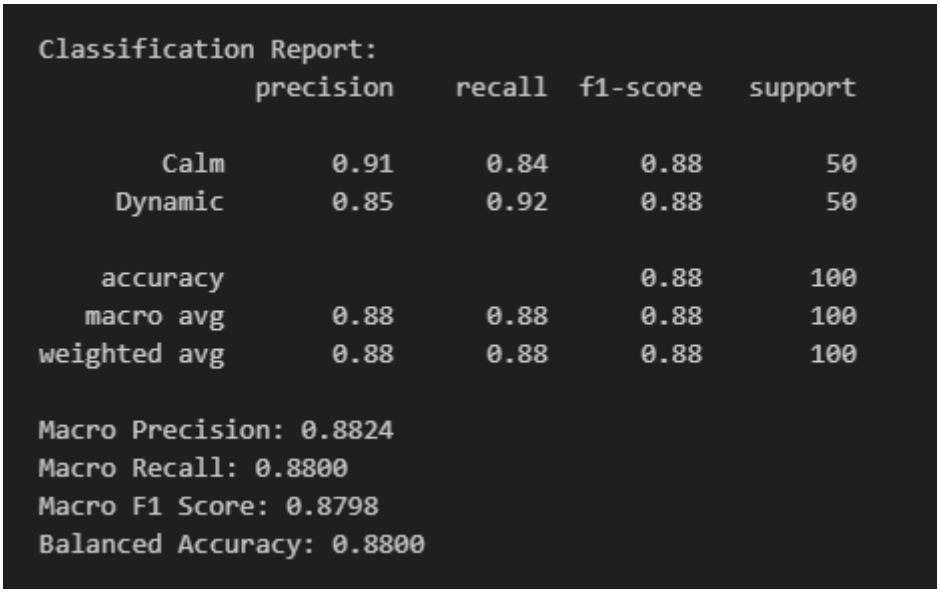


Рисунок 2.11 — Confussion matrix для 3D CNN на тестовому датасеті

Classification report на рисунку 2.12 дозволив деталізувати поведінку моделі для кожного класу окремо. Для класу calm, precision склав 0.91, що означало: серед усіх фрагментів, які модель віднесла до цього класу, 91% виявилися правильними. Recall на рівні 0.84 вказав, що з усіх наявних фрагментів класу calm модель правильно виявила 84%. F1-score становив 0.88 — середнє гармонійне між precision і recall — що свідчило про збалансовану роботу моделі стосовно цього класу. Для класу dynamic precision був дещо нижчим — 0.85, що означало гіршу здатність моделі уникати хибнопозитивних спрацьовувань у цьому класі. Recall дорівнював 0.92, що свідчило про трохи вищу, порівняно з calm, чутливість до істинних позитивних випадків. F1-score для dynamic склав 0.88, не поступаючись calm, що відображало стабільну роботу моделі на обох класах.



Classification Report:

	precision	recall	f1-score	support
Calm	0.91	0.84	0.88	50
Dynamic	0.85	0.92	0.88	50
accuracy			0.88	100
macro avg	0.88	0.88	0.88	100
weighted avg	0.88	0.88	0.88	100

Macro Precision: 0.8824
Macro Recall: 0.8800
Macro F1 Score: 0.8798
Balanced Accuracy: 0.8800

Рисунок 2.12 — Classification report для 3D CNN на тестовому датасеті

Macro Precision становив 0.88, що показало середнє арифметичне precision для обох класів без урахування їхньої кількості. Це дозволило оцінити, наскільки модель загалом уникає хибнопозитивних класифікацій для кожного класу. Macro Recall був на рівні 0.88 і продемонстрував середню здатність моделі коректно виявляти позитивні приклади обох класів. Macro F1 Score дорівнював 0.88, що

свідчило про рівномірно збалансовану якість класифікації для обох категорій без зміщення.

Balanced Accuracy, яка становила 0.8800, підтвердила, що модель однаково добре розпізнавала обидва класи незалежно від їх кількості. Цей показник є важливим у задачах, де навіть за збалансованих вибірок може існувати різна складність класифікації класів.

AUC Score дорівнював 0.95, що вказало на високу здатність моделі відрізняти між собою класи незалежно від обраного порогу. Значення вище 0.9 свідчило про те, що з ймовірністю 95% модель правильніше присвоювала вищу впевненість позитивним прикладам, ніж негативним. Це особливо важливо для моделей, що приймають рішення на основі безперервного значення, а не жорстко фіксованого порогу. Крива ROC AUC зображена на рисунку 2.13.

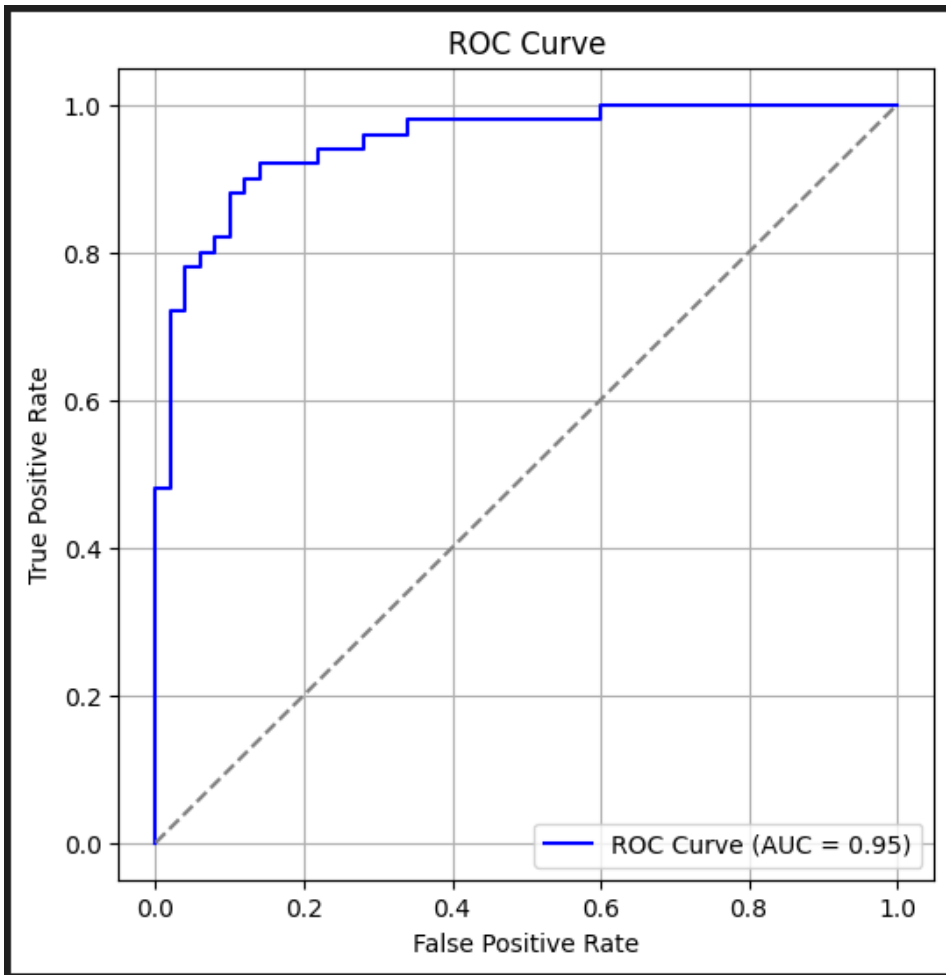


Рисунок 2.13 — ROC Curve для 3D CNN на тестовому датасеті

Нарешті, значення AP (average precision) для precision-recall curve склало 0.95. Це відобразило високу збалансованість precision і recall для всіх можливих порогів класифікації. Такий показник продемонстрував стабільність моделі при зміні порогу прийняття рішення і підтвердив її практичну корисність для задач, де критично важливий контроль співвідношення хибнопозитивних і хибнонегативних рішень. Крива precision-recall зображена на рисунку 2.14.

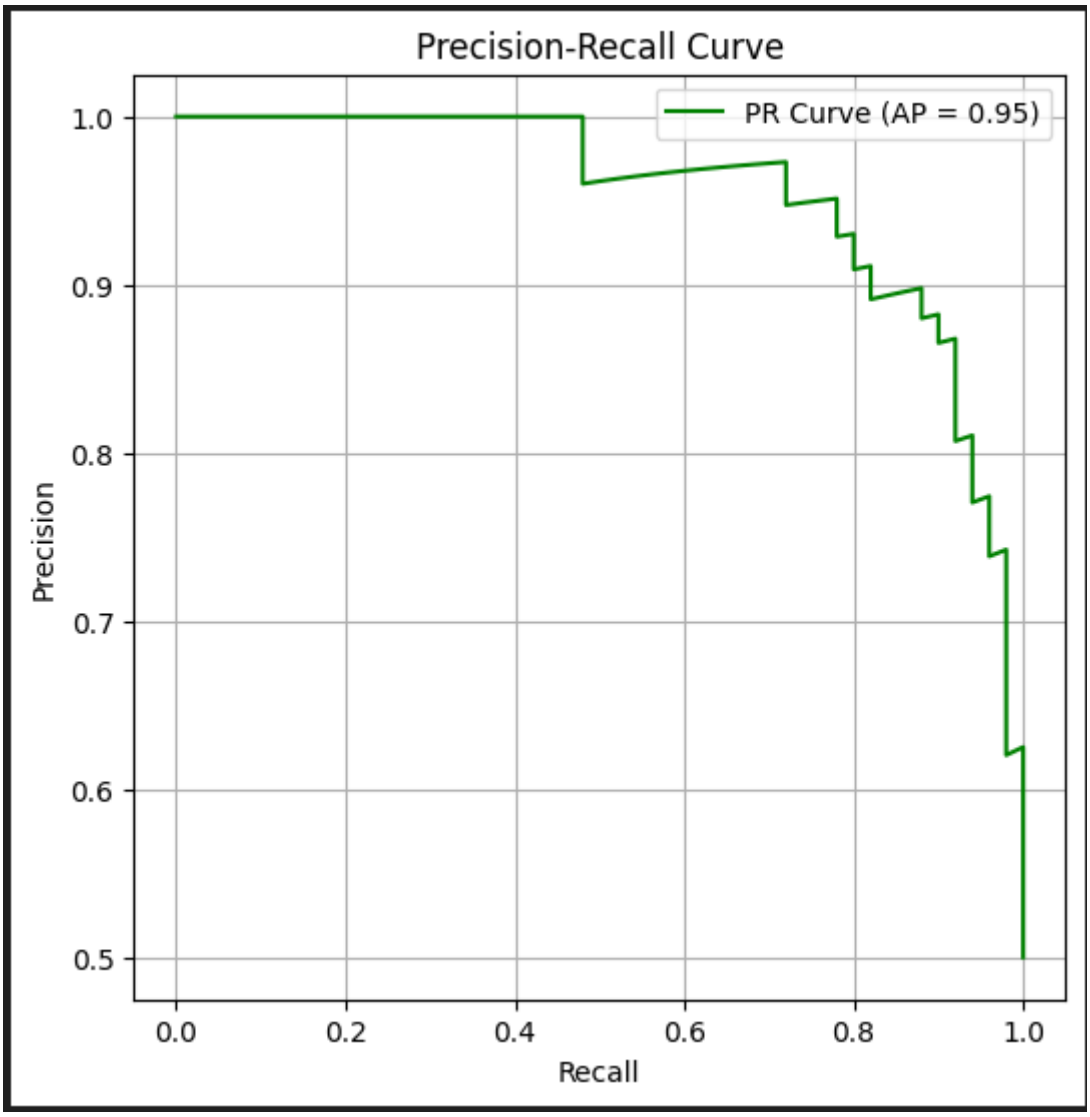


Рисунок 2.14 — Precision-recall Curve для 3D CNN на тестовому датасеті

У сукупності отримані результати засвідчили, що 3D CNN-модель досягла високих показників як загальної точності, так і балансованої роботи на окремих класах. Симетричність показників precision, recall і F1-score для обох категорій, а

також висока площа під ROC і precision-recall кривими вказали на те, що модель демонструвала не лише загальний високий рівень продуктивності, а й надійність у межах варіативності даних валідаційної вибірки.

Візуальний аналіз сегментованого відео наочно підтвердив загальні тенденції та слабкі сторони роботи моделі, які важко повністю зафіксувати лише числовими метриками. Під час перегляду класифікованих фрагментів стало помітно, що модель демонструє схильність до хибної класифікації певного типу сцен. Зокрема, динамічні моменти, де об’єктив камери рухається з високою швидкістю, але в полі зору наявна обмежена кількість помітних об’єктів чи текстурованих елементів, модель нерідко помилково відносила до категорії calm. Це пов’язано з тим, що основні ознаки, на які спирається модель при класифікації — локальні просторово-часові зміни контрастності та форми в кадрі. У випадку відсутності об’єктів чи орієнтирів, що могли б демонструвати швидкий рух (наприклад, стовбурів дерев, каменів, маркерів рельєфу), модель недостатньо виразно сприймала саму зміну положення камери як ознаку динаміки.

Особливо це проявлялося у сценах, де відео знімалося в умовах відкритого простору, наприклад, на стежках без виражених ландшафтних деталей або на ділянках, де камера спрямована переважно на небо чи рівномірну поверхню. За таких обставин модель не мала достатньої кількості орієнтирів для аналізу швидкості зміщення, оскільки зміни у відеоряді не супроводжувалися помітними переміщеннями текстурованих елементів, які б дозволили ідентифікувати динаміку.

Це обмеження підкреслює важливість збалансованості тренувального набору не лише за класами, а й за типами сцен усередині кожного класу. Зокрема, до категорії dynamic варто включати не лише кліпи з яскраво вираженими рухами об’єктів, але й відео з швидким рухом камери у візуально бідних або однорідних умовах.

2.3.3 Метрики Timesformer

Аналіз метрик трансформерної моделі дозволив отримати поглиблену оцінку її якості на валідаційній вибірці, де результати свідчать про значне поліпшення порівняно з попереднім експериментом.

Побудована confusion matrix (рисунок 2.15) показала, що для класу calm модель коректно класифікувала 45 з 50 прикладів, припустившись 5 хибних спрацьовувань, натомість для класу dynamic точність була ще вищою — правильно визначено 50 з 51 фрагмента, з однією помилкою. Ця матриця дозволила кількісно оцінити розподіл помилок за класами і підтвердила, що модель не лише забезпечує високу загальну точність, а й демонструє збалансованість у розпізнаванні обох категорій.

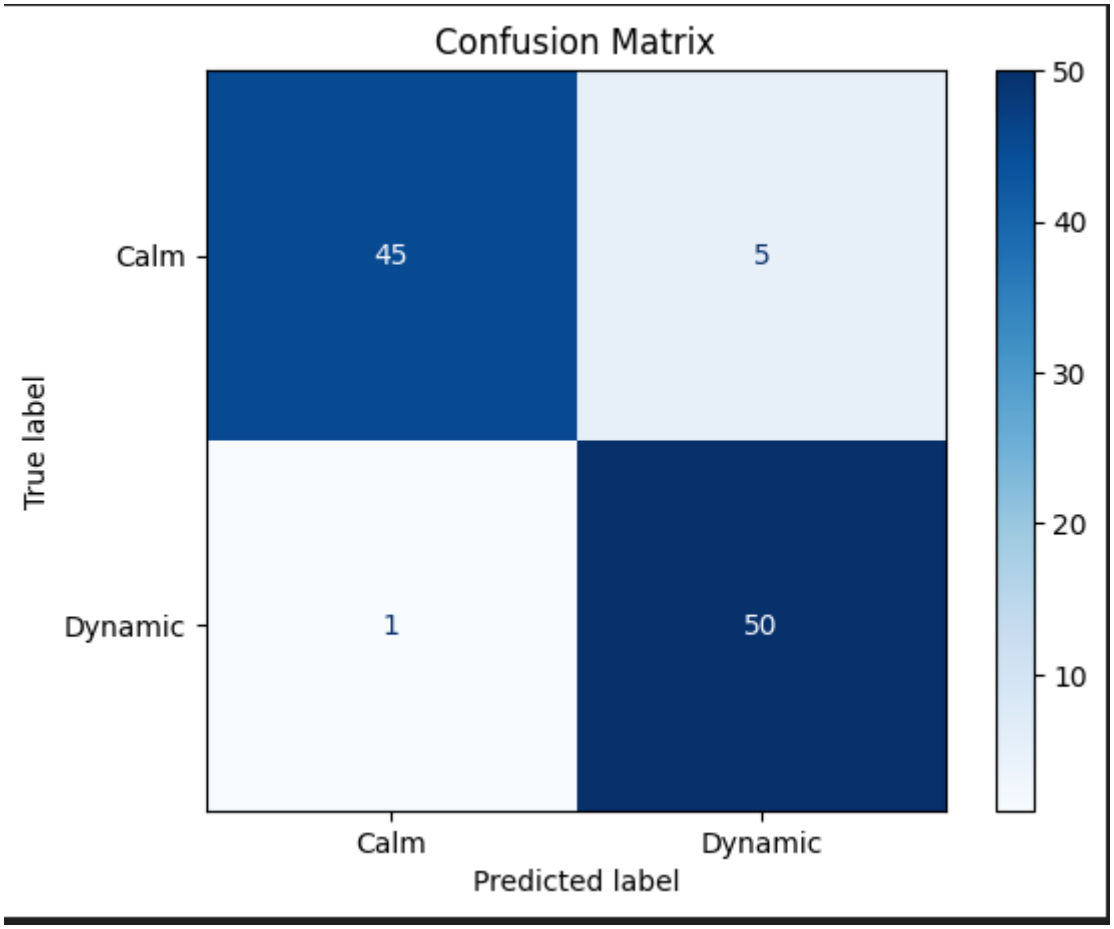


Рисунок 2.15 — Confussion matrix для Timesformer на тестовому датасеті

Classification Report (рисунок 2.16) деталізував результати за класами. Для класу calm було зафіксовано precision на рівні 0.98, що вказує, що з усіх фрагментів, які модель віднесла до calm, 98% були дійсно спокійними. Recall для цього класу становив 0.90, що означає, що з усіх наявних у валідаційній вибірці спокійних фрагментів модель правильно класифікувала 90%. F1-score об'єднав ці дві метрики в єдиний показник і склав 0.94, демонструючи високий баланс між точністю та повнотою класифікації цього класу. Для класу dynamic precision становив 0.91, recall — 0.98, а F1-score — також 0.94. Помітно, що для dynamic модель забезпечила вищу повноту, виявивши майже всі динамічні фрагменти, але трохи поступилася в precision.

```
Classification Report:
              precision    recall  f1-score   support

        Calm       0.98      0.90      0.94        50
     Dynamic       0.91      0.98      0.94        51

    accuracy                           0.94       101
   macro avg       0.94      0.94      0.94       101
weighted avg       0.94      0.94      0.94       101

Macro Precision: 0.9437
Macro Recall: 0.9402
Macro F1 Score: 0.9404
Balanced Accuracy: 0.9402
```

Рисунок 2.16 — Classification report для Timesformer на тестовому датасеті

Macro Precision склав 0.94, що є середнім арифметичним precision по обох класах без урахування їх частоти. Це дозволило оцінити середню точність класифікації для кожного класу незалежно від дисбалансу даних. Значення, близьке до 0.95, свідчить про стабільно високу точність моделі для кожного класу.

Macro Recall дорівнював 0.94, що аналогічно визначав середню повноту по класах, дозволяючи зрозуміти наскільки ефективно модель виявляла усі наявні

приклади кожного класу. Показник вище 0.94 підтвердив, що помилки пропуску були рідкісними.

Macro F1 Score, обчислений як середнє значення F1 по класах, становив 0.94. Це дозволило узагальнити баланс точності та повноти для обох класів. Значення цього показника є важливим при оцінюванні якості моделі на збалансованих задачах двокласової класифікації, як у даному випадку.

Balanced Accuracy була зафіксована на рівні 0.94. Ця метрика є середнім значенням recall для кожного класу і дозволяє коректно оцінювати якість моделі на збалансованих і незбалансованих вибірках. Її високе значення вказало на відсутність істотного перекосу у роботі моделі щодо одного з класів.

AUC Score склав 0.99, що відображає площу під ROC-кривою (рисунок 2.17). Цей показник продемонстрував, наскільки добре модель розділяє обидва класи по всьому діапазону порогових значень. Значення, наближене до 1, вказує на майже ідеальне відокремлення класів без значного перекриття.

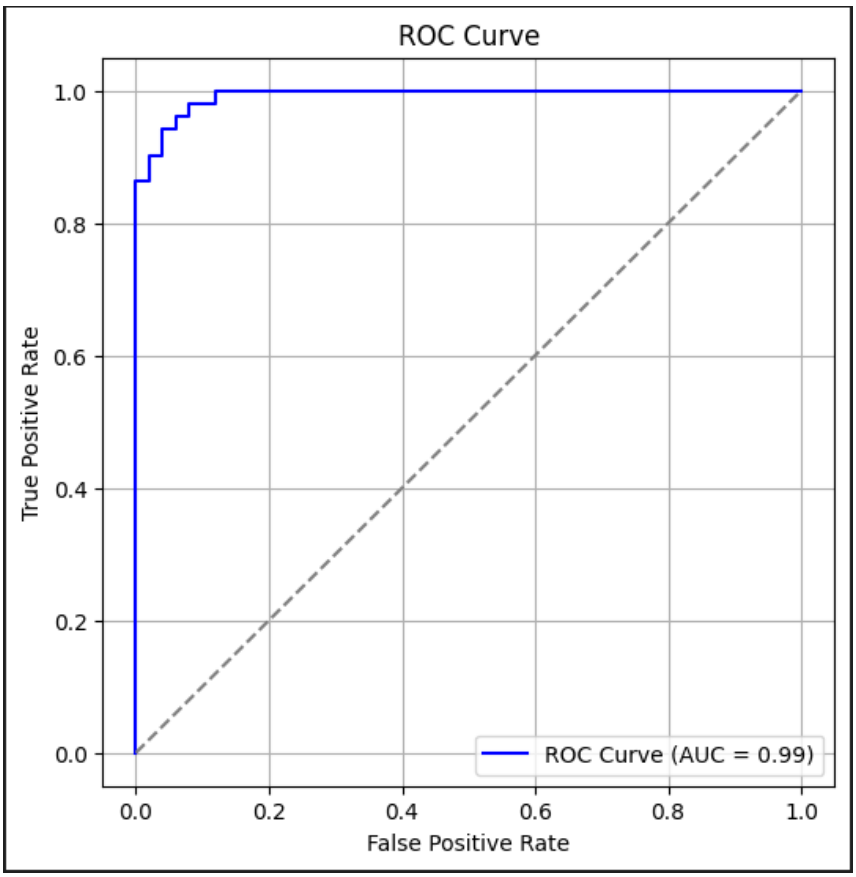


Рисунок 2.17 — ROC Curve для Timesformer на тестовому датасеті

Average Precision (AP) за PR-кривою (рисунок 2.18) також становив 0.99. Це підтвердило, що модель зберігає високу точність і повноту навіть при зміні порогу класифікації, що критично важливо для задач, де ці параметри можуть бути адаптовані під конкретні потреби користувача або системи.

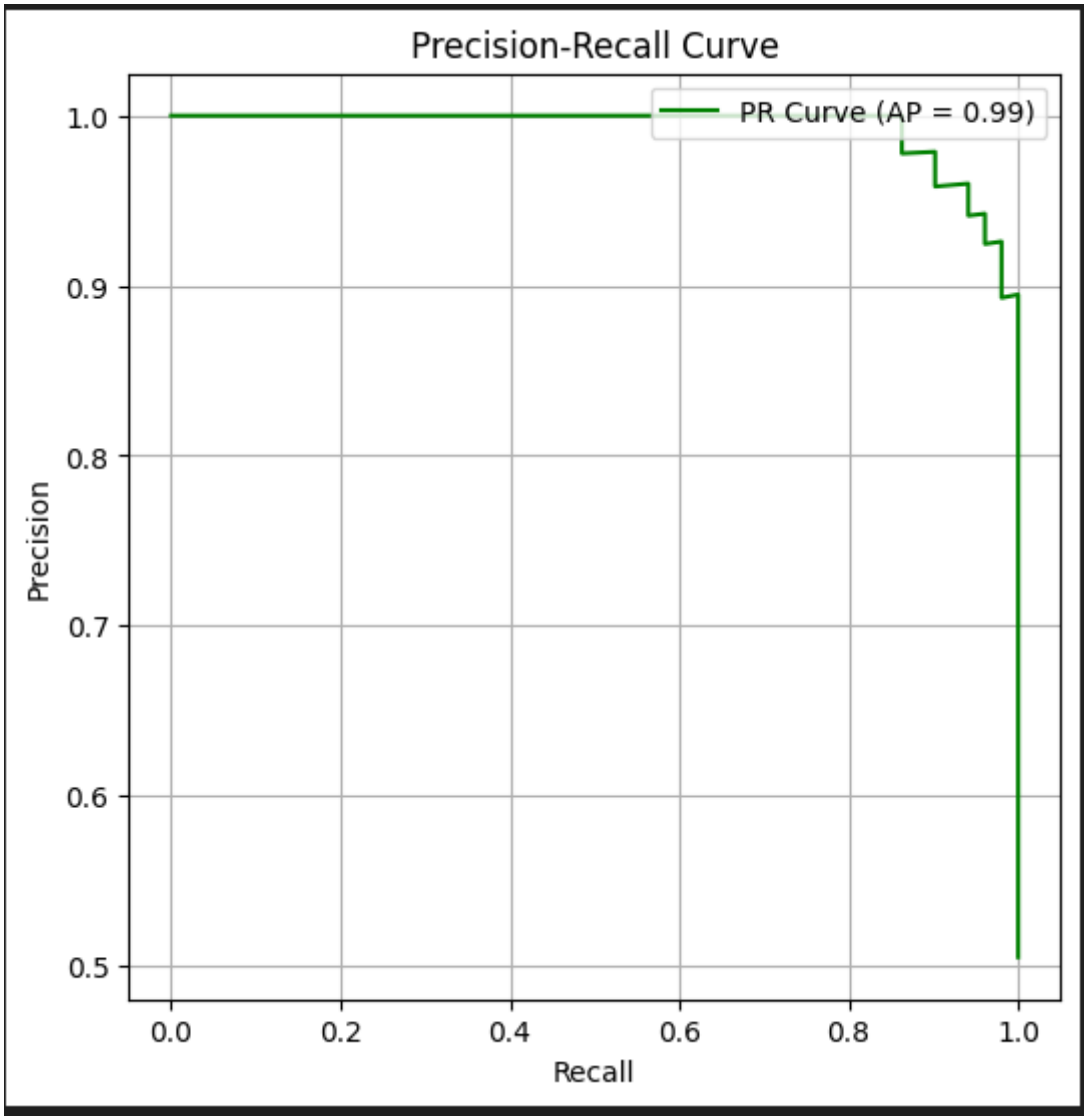


Рисунок 2.18 — Precision-recall Curve для 3D CNN на тестовому датасеті

Візуальний аналіз результатів сегментованого відео продемонстрував суттєве покращення поведінки трансформерної моделі порівняно з попередньою 3D CNN. Зокрема, було помічено, що модель значно краще розпізнає темп та швидкість подій на відео, коректніше інтерпретуючи контекст сцени навіть у випадках зі складними рухами камери. Якщо 3D CNN мала схильність

класифікувати різкі панорами чи зміщення кадру як ознаку динаміки незалежно від фактичної швидкості об’єктів у сцені, то трансформер демонстрував стійкість до подібних артефактів. Це дозволило уникнути помилкових класифікацій спокійних фрагментів із камерними ривками як динамічні, що є критично важливим для реальних застосувань у відеоаналітиці.

Ще одним важливим спостереженням стало те, що модель, на відміну від 3D CNN, почала розпізнавати моменти їзди велосипедом на низькій швидкості, пов’язані з виконанням трюків, як динамічні. Це особливо цінне, оскільки подібні сценарії не були представлені у тренувальних даних. Така здатність до узагальнення й адаптації до нових, раніше невідомих патернів поведінки вказує на високу гнучкість трансформерної архітектури. Фактично, модель навчилася враховувати не лише швидкість переміщення по екрану, а й характер рухів та їхню кінематичну структуру.

Це демонструє перспективу використання цієї моделі не лише для базової бінарної класифікації динаміки, а й для складніших відеозавдань — наприклад, детекції типів активності, відстеження складних технічних елементів, або ж адаптивної роботи в середовищах із високим рівнем візуального шуму. З огляду на те, що модель успішно класифікувала нові для неї патерни без додаткового навчання, можна стверджувати про наявність у неї потенціалу до подальшого розширення функціоналу й інтеграції в багатофункціональні системи відеоаналізу.

2.4 Порівняння натренованих моделей з аналогами

У сучасних системах аналізу відеоданих особливу увагу приділяють ефективності та адаптивності моделей для розпізнавання динамічних подій. З метою обґрунтування доцільності використання розроблених моделей було здійснено порівняльний аналіз їх результатів із наявними сервісами та рішеннями, що виконують класифікацію відео за різними ознаками.

2.4.1 Аналіз рішень у відкритому доступі

Існуючі сервіси для автоматичної класифікації відео, які доступні через хмарні API, демонструють потужний функціонал у загальних сценаріях відеоаналітики. Зокрема, Google Cloud Video Intelligence API здійснює аналіз відео на рівні кадрів або сегментів, працює із метаданими на рівні відео, сцен і кадрів, а також забезпечує можливості для кастомної класифікації через AutoML [10]. Цей сервіс дає змогу отримати швидкі результати, однак основною його метою є саме семантичне розпізнавання, а не оцінка рівня динамічності руху.

Ще одним гравцем цього класу є Amazon Rekognition Video, що дозволяє визначати об’єкти, сцени, особистості, містить функціонал для виявлення активностей, тексту, осіб та потенційно небезпечного контенту в відео, а також підтримує стрімінговий режим та інтерфейси для кастомізації моделей [11]. Подібно до Google API, сервіс орієнтований на загальні випадки відеоаналізу, а не на специфічну сегментацію динамічності.

Серед більш сучасних сервісів слід виокремити Twelve Labs Video Intelligence Platform [12], яка позиціонується як система для мультимодального аналізу відео з підтримкою класифікації, детекції та пошуку фрагментів за текстовими запитами. Сервіс надає зручний хмарний інтерфейс для завантаження відеофайлів, автоматичної обробки та аналізу контенту за допомогою API.

2.4.2 Тестування Twelve Labs Classification API

У межах порівняльного аналізу існуючих сервісів для автоматичної класифікації відеоконтенту було протестовано сервіс Twelve Labs. Ця платформа позиціонується як універсальне рішення для пошуку й класифікації фрагментів у відео за текстовими запитами без потреби в попередньому навчанні моделей.

Інтерфейс сервісу виявився зручним та інтуїтивно зрозумілим, а також підтримує інтеграцію через API, що дозволяє використовувати його в сторонніх застосунках.

Разом із тим, у процесі тестування було виявлено низку суттєвих обмежень. По-перше, система вимагає відео з роздільною здатністю не нижче 360p, що унеможливлює обробку низькоякісних або попередньо обрізаних відео для швидкого прототипування. По-друге, класифікація здійснюється виключно за текстовими запитами, які інтерпретуються мовною моделлю, а отже — результати є значною мірою контекстуальними та не завжди коректно відображають специфіку динамічних фрагментів.

Особливо критичною для даного дослідження виявилася низька точність передбачень у випадках із динамічними моментами, пов’язаними з рухом камери або об’єкта. Приклад аналізу відео зображено на рисунку 2.19

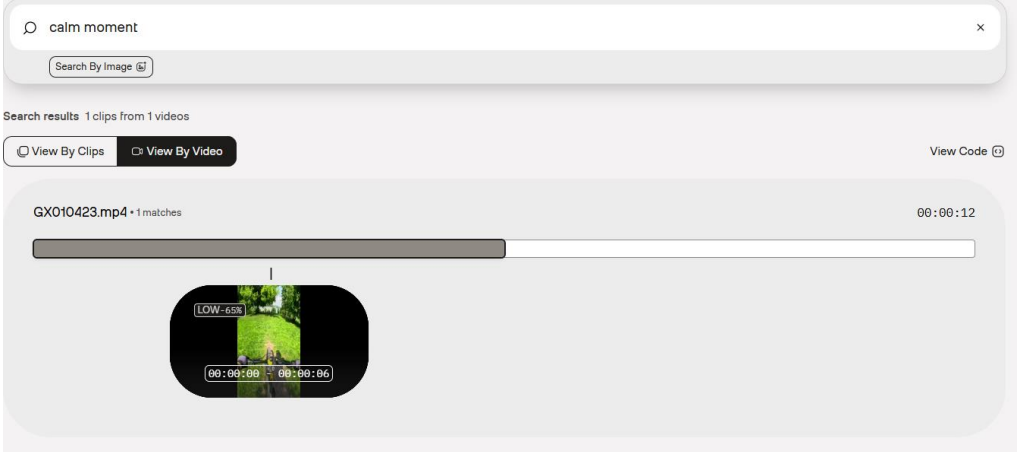


Рисунок 2.19 — Приклад роботи Twelve Labs

Попри ці недоліки, Twelve Labs має сильні стороні — підтримку REST API та простий у використанні веб-інтерфейс. Однак, для вирішення вузькоспеціалізованих завдань автоматичної сегментації динамічних фрагментів, його функціональності недостатньо.

2.5 Обмеження та перспективи

У межах виконаного дослідження було розроблено програмну систему для автоматичної сегментації динамічних фрагментів у відеозаписах за допомогою методів машинного навчання, зокрема архітектур 3D CNN та трансформерів. Проте отримані результати та експериментальний аналіз виявили низку обмежень, які потенційно впливають на якість і узагальнюваність побудованих моделей.

Одним із ключових обмежень є обмежений обсяг та одноманітність тренувального датасету. Для навчання моделей використовувалися відеофрагменти, які хоча й мали певну варіативність за умовами зйомки, проте були зняті в переважно схожих середовищах, що обмежувало можливості моделей адаптуватися до різноманітних сценаріїв, камерних рухів та стилів відео. Особливо це стало помітно в контексті класифікації POV-відео, де швидкість руху об’єктів не завжди корелює з динамічністю сцени. Залучення більш об’ємного та різноманітного датасету, який би включав відеозаписи з різних камер, ракурсів, умов освітлення, швидкостей та типів рухів, дозволило б суттєво підвищити узагальнюваність та стійкість моделей до нових даних.

Також важливим обмеженням слід вважати використання лише двох класів для класифікації фрагментів — "динамічний" та "спокійний". Така бінарна схема не враховує проміжні стани або специфічні види динаміки, як-от поступовий рух, фоновий рух без зміщення камери, або різкі зміни сцени без руху об’єкта зйомки. Розширення кількості класів або впровадження багатовимірної оцінки динамічності могло б забезпечити більш гнучке та реалістичне моделювання поведінки відео.

Ще одним фактором, що обмежував результати, є обрана архітектура моделей. Хоча 3D CNN продемонструвала базовий рівень ефективності, її здатність враховувати довгострокові темпоральні залежності виявилася обмеженою. Трансформерна модель, зокрема Timesformer, показала значно вищі результати, проте навіть вона потребує подальшого вдосконалення. Застосування сучасніших

архітектур, таких як Video Swin Transformer, MViT або комбінованих гібридних рішень, могло б дозволити ефективніше опрацьовувати просторово-часові зв’язки та адаптуватися до складних сцен.

Крім того, певним обмеженням є спосіб розмітки тренувальних даних. Ручна розмітка на основі суб’єктивного сприйняття динамічності вводить фактор людської помилки та варіативності, що може негативно впливати на якість навчання. Використання автоматизованих або напівавтоматизованих методів оцінки руху, наприклад, за допомогою оптичного потоку чи темпоральних градієнтів, могло б підвищити об’єктивність розмітки.

Окремим аспектом перспектив розвитку системи є інтеграція візуалізаційного веб-інтерфейсу для інтерактивного завантаження та обробки відео. Це дозволило б забезпечити зручний спосіб використання моделі не лише в експериментальному середовищі, а й в реальних прикладних задачах. Використання фреймворків на кшталт Streamlit могло б істотно покращити доступність системи для кінцевих користувачів та продемонструвати роботу моделей на практиці.

Загалом перспективними напрямами для подальших досліджень є збільшення обсягу та різноманітності навчальних даних, розширення кількості класів класифікації, використання прогресивніших архітектур глибоких нейронних мереж, а також розробка прикладних програмних інтерфейсів для інтеграції системи у сторонні сервіси та додатки.

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Домедична допомога при переломах

Для збору даних у цьому дослідженні використовувалася відеозйомка від першої особи під час катання на гірському велосипеді по пересіченій місцевості та спеціально облаштованих трасах. Під час одного з тренувальних заїздів виникла надзвичайна ситуація: на приземленні після стрибка інший спортсмен залишив свій велосипед на траєкторії, що стало причиною падіння через недостатню видимість перешкоди. У результаті стався сильний удар у зап’ястя лівої руки, що супроводжувався підозрою на перелом.

Згідно з визначенням, переломом називається часткове або повне порушення цілісності кісткової тканини, причому виділяють два основні типи — травматичні та патологічні [13]. Травматичні переломи виникають внаслідок дії зовнішніх механічних чинників, зокрема ударів, падінь або здавлення, тоді як патологічні є наслідком внутрішніх змін у кістковій тканині, спричинених, наприклад, метастазами пухлин, туберкульозом чи остеомієлітом. [14]

Крім того, переломи поділяються на відкриті та закриті [14]. Відкритий перелом супроводжується пошкодженням шкіри в місці травми, що створює ризик інфікування кісткових уламків і розвитку остеомієліту, тоді як при закритому переломі цілісність шкіри не порушується. Також, обидва типи можуть бути зі зміщенням кісткових уламків або без нього. За ступенем порушення цілісності тканини переломи бувають повні та неповні: при останніх утворюється тріщина, що не поширюється на всю товщину кістки [14].

Типові клінічні ознаки перелому включають різкий біль, деформацію ушкодженої ділянки, порушення її функції, набряк, крововилив у зоні травми, а також патологічну рухливість і крепітацію — характерне хрумтіння уламків при русі. У випадках відкритих переломів у рані можуть бути помітні уламки кістки [14]. Біль посилюється при зміні положення кінцівки і послаблюється у стані спокою. Крім місцевих проявів, переломи часто супроводжуються загальними

симптомами — порушенням сну, апетиту, підвищенням температури тіла, загальною слабкістю, а також можуть виникати ознаки травматичного шоку або гострої крововтрати [16].

Аналізуючи ситуацію, що виникла під час збору даних, можна стверджувати, що наявні ознаки — порушення функції руки, набряк, біль при зміні положення та різкий біль при пальпації — дозволяли припустити закритий перелом, що підтверджувала також відсутність кровотечі та відкритої рани.

Згідно з Порядком надання домедичної допомоги постраждалим при підозрі на перелом кісток кінцівок [18]:

- перед наданням допомоги переконатися у відсутності небезпеки для себе, оточуючих, постраждалого та за її відсутності перейти до наступного кроку;
- заспокоїти постраждалого та пояснити свої подальші дії;
- здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику;

Якщо у постраждалого ознаки відкритого перелому:

- розрізати одяг та оглянути рану;
- якщо є кровотеча з рани, діяти відповідно Порядку надання домедичної допомоги постраждалим при масивній зовнішній кровотечі, затвердженого наказом Міністерства охорони здоров’я України від 09 березня 2022 року № 441 [18];
- накласти стерильну, чисту пов’язку на рану;
- допомогти постраждалому прийняти зручне положення (таке, яке завдає найменше болю);
- іммобілізувати (знерухомити) пошкоджену кінцівку за допомогою стандартного обладнання (шин) чи підручних засобів: здійснювати іммобілізацію тільки за умови проходження відповідного навчання; визначити дистальний пульс на кінцівці до та після іммобілізації;

Якщо у постраждалого ознаки закритого перелому:

- допомогти постраждалому прийняти зручне положення (таке, яке завдає найменше болю);
- іммобілізувати (знерухомити) пошкоджену кінцівку за допомогою стандартного обладнання (шин) чи підручних засобів: здійснювати іммобілізацію тільки за умови проходження відповідного навчання; визначити дистальний пульс на кінцівці до та після іммобілізації;
- вкрити постраждалого термопокривалом/ковдрою;
- забезпечити постійний нагляд за постраждалим до приїзду бригади екстреної (швидкої) медичної допомоги;
- при погіршенні стану постраждалого до приїзду бригади екстреної (швидкої) медичної допомоги повторно здійснити виклик екстреної медичної допомоги;
- за можливості зібрати у постраждалого максимально можливу кількість інформації стосовно обставин травми та обставинах при її отримання. Всю отриману інформацію передати членам бригади екстреної (швидкої) медичної допомоги або диспетчеру прийому.
- Якщо до приїзду бригади екстреної (швидкої) медичної допомоги постраждалий втратив свідомість, слід перейти до Порядку надання домедичної допомоги дорослим при раптовій зупинці кровообігу або Порядку надання домедичної допомоги дітям при раптовій зупинці кровообігу, затверджених наказом Міністерства охорони здоров’я України від 09 березня 2022 року № 441 [18].

3.2 Охорона праці при роботі за персональним комп’ютером

Наступним етапом розробки системи став тривалий процес аналізу, розмітки та обробки відеофрагментів на персональному комп’ютері. Ця робота потребувала значної концентрації уваги, точності та витривалості, оскільки передбачала

багаторазовий перегляд записів, кадрування, а також позначення кліпів для подальшої роботи з ними. У зв’язку з цим особливої актуальності набуло питання організації оптимальних умов праці за комп’ютером, адже саме від них значною мірою залежить як якість виконання завдань, так і загальний фізичний та психоемоційний стан.

Важливість правильного планування робочого місця, згідно лекційних матеріалів [15], полягає у визначенні основних і допоміжних робочих рухів, оптимальному розміщенні обладнання, виборі відповідного інструменту, пристосувань та інвентарю, а також допоміжних пристроїв, які забезпечують безпеку праці. Особливу увагу при цьому приділяють положенню працюючого та його позі. Як зазначено у [15], зону основних робочих рухів необхідно визначати з урахуванням антропометричних даних людини: поля її зору, зросту, довжини рук і ніг. Робоча зона повинна розташовуватися в межах найзручнішого огляду та доступності, без потреби надмірного повороту голови (не більше 40° у горизонтальній та 30° у вертикальній площинах), корпуса (не більше 45° вліво або вправо), витягування рук (до 500 мм в сторони і 300 мм уперед) та ніг (до 350 мм уперед і 300 мм угору).

Крім ергономічних параметрів, на ефективність і комфорт роботи за комп’ютером впливають також освітлення, температура повітря, рівень шуму, кольорове оформлення приміщення та загальний естетичний стан робочого середовища [17]. Зокрема, освітлення робочого простору повинно здійснюватися за допомогою природного та штучного світла. У разі недостатності природного освітлення рекомендується застосовувати комбіноване — одночасне використання природного і штучного світла у денний час доби [19].

Відповідно до положень ДБН В.2.5-28:2018 «Природне і штучне освітлення» [20], освітленість на робочих місцях має відповідати характеру зорової роботи, забезпечувати рівномірний розподіл яскравості на поверхнях, уникати різких тіней, сліпучих відблисків і засліпленості, гарантувати сталість освітлення впродовж роботи, правильну передачу кольорів [20]. Таким чином, освітлення має

бути нейтральним і постійним, що дозволяє зменшити втому та підтримувати високу концентрацію під час тривалих завдань.

Окрему увагу слід приділяти естетичним чинникам, які, окрім безпосереднього впливу на здоров’я, формують позитивний психологічний настрій, що, своєю чергою, підвищує працездатність. Як підкреслюється у [17], важливими є художньо-конструкторські якості робочого місця, елементи декору, архітектурно-художнє оформлення інтер’єру, наявність рекреаційних зон тощо.

Науково доведено, що колірне оформлення приміщення також суттєво впливає на емоційний стан людини: одні кольори заспокоюють, інші — збуджують чи подразнюють. Наприклад, червоний асоціюється з енергією та стимулює до активності, тоді як зелений та світло-блакитний сприяють розслабленню і зменшенню нервового напруження [17]. Тому для тривалої монотонної роботи за комп’ютером рекомендується використовувати нейтральні відтінки — світло-зелений, блакитний, білий, які знижують зорове та емоційне навантаження, забезпечуючи комфортні умови праці [17].

Додатковим позитивним фактором є озеленення приміщень, що не лише прикрашає простір, а й покращує якість повітря, регулює мікроклімат, зменшує запиленість, а також має заспокійливий вплив на нервову систему [17]. Окрім цього, важливо враховувати емоційний вплив оформлення робочого місця особистими речами. Наявність невеликих сувенірів, фотографій або улюблених предметів сприяє створенню комфортного емоційного середовища та знижує психоемоційне напруження, що позитивно позначається на продуктивності та загальному стані [17].

ВИСНОВКИ

На основі аналізу сучасних підходів до обробки відеоданих та методів сегментації за динамічними ознаками було розроблено програмну систему для автоматичної сегментації динамічних фрагментів у відеозаписах із використанням методів машинного навчання. В межах роботи проведено огляд наукових джерел, патентних розробок та прикладних рішень, що дозволило обґрунтувати вибір нейронних архітектур та методик навчання. Зокрема, було реалізовано та порівняно дві моделі: тривимірну згорткову нейронну мережу (3D CNN) та трансформерну архітектуру Timesformer, адаптовані для задачі бінарної класифікації відеофрагментів за ознакою динамічності.

Розроблена система дозволяє автоматично здійснювати попередню обробку відео, розбивати його на фрагменти заданої тривалості, класифікувати кожен із них за допомогою натренованих моделей та формувати сегментовану розмітку за типом сцени. Застосування адаптивного глобального пулінгу та часових attention-механізмів дало змогу досягти високої точності класифікації навіть на обмежених обсягах тренувальних даних. Проведений аналіз метрик якості моделей показав, що трансформерна архітектура продемонструвала кращі результати за всіма показниками: загальна точність на валідаційній вибірці склала 94%, значення AUC досягло 0.99, а макро-усереднені precision, recall та F1-score становили понад 0.94. Модель 3D CNN показала прийнятний рівень якості, однак поступалася трансформеру в здатності виявляти складні темпоральні патерни.

Візуальний аналіз результатів сегментації підтвердив вищу адаптивність трансформерної моделі до нових типів сцен та рухів, зокрема розпізнавання низькошвидкісних трюків без явного зростання швидкості камери, що свідчить про перспективність використання даного підходу для більш широких задач відеоаналітики.

Результати роботи дозволяють автоматизувати задачі попереднього аналізу відеозаписів для виявлення динамічних моментів, що може бути корисним у

спортивній аналітиці, відеомоніторингу, робототехніці та інших сферах, де важливо виділяти активні фрагменти з тривалих відео.

Водночас проведене дослідження виявило низку обмежень, пов’язаних із обсягом та різноманітністю тренувального датасету, а також бінарною природою класифікації. Перспективним напрямом подальших досліджень є розширення об’єму навчальних даних, залучення різноманітних типів відео та рухів, збільшення кількості класів класифікації, а також інтеграція прогресивніших моделей глибокого навчання та великих мовних моделей. Додатково доцільною є розробка прикладного веб-інтерфейсу для інтерактивної взаємодії з системою, що дозволить перевіряти її працездатність та застосовувати в реальних умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методи глибокого навчання для сегментації екземплярів відео: опитування. Science Direct [Електронний ресурс]. — Режим доступу: URL: https://www.sciencedirect.com/science/article/pii/S0031320325004236 (дата звернення 17.06.2025)

2. Комплексний огляд застосування 3D згорткових нейронних мереж у медичній візуалізації. MDPI [Електронний ресурс]. — Режим доступу: URL: https://www.mdpi.com/2673-4591/59/1/3 (дата звернення 17.06.2025)

3. Методи відеорезюмування: комплексний огляд. Arxiv [Електронний ресурс]. — Режим доступу: URL: https://arxiv.org/html/2410.04449v1 (дата звернення 17.06.2025)

4. Qiu Zhaofan, Yao Ting, Mei Tao. Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks [Електронний ресурс]. — University of Science and Technology of China, Hefei, China; Microsoft Research, Beijing, China, 2017. — 9 с. — Режим доступу: URL: https://arxiv.org/abs/1711.10305 (дата звернення: 19.06.2025)

5. Tran Du, Bourdev Lubomir, Fergus Rob, Torresani Lorenzo, Paluri Manohar. Learning Spatiotemporal Features with 3D Convolutional Networks [Електронний ресурс]. — Facebook AI Research; Dartmouth College, 2017. — 9 с. — Режим доступу: URL: https://arxiv.org/abs/1412.0767 (дата звернення: 19.06.2025)

6. Arnab Anurag, Dehghani Mostafa, Heigold Georg, Sun Chen, Lucic Mario, Schmid Cordelia. ViViT: A Video Vision Transformer [Електронний ресурс]. — Google Research, 2021. — 11 с. — Режим доступу: URL: https://arxiv.org/abs/2103.15691 (дата звернення: 18.06.2025)

7. Xu Yifang, Sun Yunzhuo, Li Yang, Shi Yilei, Zhu Xiaoxiang, Du Sidan. MH-DETR: Video Moment and Highlight Detection with Cross-modal Transformer [Електронний ресурс]. — 2023. — 10 с. — Режим доступу: URL: https://arxiv.org/abs/2305.00355 (дата звернення: 18.06.2025)

8. Method and system for automatically generating video highlights : пат. CN109691124B Китай; заявл. 19.06.2017; опубл. 27.07.2021. — 13 с.

9. Systems and methods for automating video editing : пат. US11769528B2 США; заявл. 02.03.2021; опубл. 26.09.2023. — 47 с.

10. Платформа Google Cloud Intelligence API. Google Cloud [Електронний ресурс]. — Режим доступу: URL: https://cloud.google.com/video-intelligence (дата звернення 19.06.2025)

11. Платформа Amazon Rekognition. Amazon Web Services [Електронний ресурс]. — Режим доступу: URL: https://aws.amazon.com/rekognition/video-features (дата звернення 19.06.2025)

12. Як використовувати Twelve Labs API. Twelve Labs [Електронний ресурс]. — Режим доступу: URL: https://www.twelvelabs.io/blog/effortless-video-classifiers-with-twelve-labs-api-no-ml-training-required (дата звернення 19.06.2025)

13. РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ КРБ. ATutor [Електронний ресурс]. — Режим доступу: URL: https://dl.tntu.edu.ua/content.php?cid=514586 (дата звернення: 15.06.2025)

14. Долікарська допомога при переломах. ATutor [Електронний ресурс]. — Режим доступу: URL: https://dl.tntu.edu.ua/content.php?cid=299865 (дата звернення: 15.06.2025)

15. Організація робочих місць. ATutor [Електронний ресурс]. — Режим доступу: URL: https://dl.tntu.edu.ua/content.php?cid=289193 (дата звернення: 15.06.2025)

16. Мелех Л.В. Безпека життєдіяльності та охорона праці: навч. посіб. Львів: ЛДУ внутрішніх справ, 2022. 219 с

17. В. Г. Грибан., А. Є. Фоменко, Д. Г. Казначеєв. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОХОРОНА ПРАЦІ: підручник. Дніпро: Дніпроп. держ. ун-т внутр. справ, 2022. 388 с.

18. Про затвердження порядків надання домедичної допомоги особам при невідкладних станах: Наказ Міністерства охорони здоров’я України від 09.03.2022

№ 441. — Режим доступу: URL: https://zakon.rada.gov.ua/laws/show/z0356-22 (дата звернення: 15.06.2025).

19. Вимоги нормативних документів до систем виробничого освітлення. ATutor. Режим доступу: URL: https://dl.tntu.edu.ua/content.php?cid=289154 (дата звернення: 16.06.2025)

20. ДБН В.2.5-28-2018 "Природне і штучне освітлення". Чинний від 01.03.2019. — Режим доступу: URL: https://e-construction.gov.ua/laws_detail/3074958732556240833?doc_type=2

21. Дистанційний курс «Кваліфікаційні роботи бакалаврів» сайту дистанційного навчання ТНТУ [Електронний ресурс]. — Режим доступу: URL: https://dl.tntu.edu.ua/bounce.php?course=5329 (дата звернення: 19.06.2025)

22. Методичні вказівки до виконання дипломної роботи освітнього рівня “бакалавр” студентами усіх форм навчання для напряму підготовки 121 – “Інженерія програмного забезпечення” / уклад. : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. — Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016. — 28 с.

ДОДАТКИ

ДОДАТОК А — Лістинг коду програмної системи

```
import os
import torch
from torch.utils.data import Dataset
from torchvision import transforms
from PIL import Image
import random

class VideoClipsDataset(Dataset):
    def __init__(self, samples, clip_length=64, transform=None):
        self.samples = samples
        self.clip_length = clip_length
        self.transform = transform

    def __len__(self):
        return len(self.samples)

    def __getitem__(self, idx):
        clip_path, label = self.samples[idx]
        frame_files = sorted(os.listdir(clip_path))[:self.clip_length]

        frames = []
        for frame_file in frame_files:
            frame_path = os.path.join(clip_path, frame_file)
            image = Image.open(frame_path).convert("RGB")
            if self.transform:
                image = self.transform(image)
            frames.append(image)

        clip_tensor = torch.stack(frames).permute(1, 0, 2, 3)  # (C, T, H,
W)
        return clip_tensor, label
```

Лістинг А.1 — Клас тренувального датасету для моделей комп’ютерного бачення

```
import os
import cv2
import numpy as np
from tqdm import tqdm
import shutil
import random
import matplotlib.pyplot as plt

raw_data_dir = "./data/raw"
input_dirs = ['dynamic', 'calm'] # Input folders
output_dir = './data/processed' # Output folder
frame_size = (224, 224) # Frame size for to shrink
clip_length = 64 # Number of frames per clip
discard_incomplete = True # Whether to discard leftover frames
image_format = 'png'
frame_skip = 2 # Read every 2nd frame (60fps -> 30fps)

for label in input_dirs:
    os.makedirs(os.path.join(output_dir, label), exist_ok=True)

clip_counter = {'dynamic': 0, 'calm': 0}

for label in input_dirs:
    video_files = [f for f in os.listdir(f"{raw_data_dir}/{label}") if
f.endswith(('.mp4', '.mov', '.avi'))]
    print(f'Processing {len(video_files)} videos in {label}...')

    for video_file in tqdm(video_files):
        video_path = os.path.join(f"{raw_data_dir}/{label}", video_file)
        cap = cv2.VideoCapture(video_path)
        frames = []
        frame_idx = 0
        success, frame = cap.read()

        while success:
            if frame_idx % frame_skip == 0:
                frame = cv2.resize(frame, frame_size)
                frames.append(frame)
```

```
            frame_idx += 1
            success, frame = cap.read()

        cap.release()

        total_frames = len(frames)
        num_clips = total_frames // clip_length

        if num_clips == 0:
            continue  # skip short videos

        for i in range(num_clips):
            clip_frames = frames[i * clip_length: (i + 1) * clip_length]
            clip_id = clip_counter[label]
            clip_folder = os.path.join(output_dir, label,
f'{label}_{clip_id:04d}')
            os.makedirs(clip_folder, exist_ok=True)

            for j, frame in enumerate(clip_frames):
                frame_path = os.path.join(clip_folder,
f'frame_{j:04d}.{image_format}')
                cv2.imwrite(frame_path, frame)

            clip_counter[label] += 1

        if not discard_incomplete and total_frames % clip_length != 0:
            # Save remaining frames as one extra (shorter) clip
            clip_frames = frames[num_clips * clip_length:]
            clip_id = clip_counter[label]
            clip_folder = os.path.join(output_dir, label,
f'{label}_{clip_id:04d}')
            os.makedirs(clip_folder, exist_ok=True)

            for j, frame in enumerate(clip_frames):
                frame_path = os.path.join(clip_folder,
f'frame_{j:04d}.{image_format}')
                cv2.imwrite(frame_path, frame)
```

```
            clip_counter[label] += 1

# Data balancing
input_root = './data/processed'
output_root = './data/balanced'
target_count_per_class = 500
classes  = ['dynamic', 'calm']

for label in classes:
    os.makedirs(os.path.join(output_root, label), exist_ok=True)

# Copy existing calm clips
calm_clips = os.listdir(os.path.join(input_root, 'calm'))
random.shuffle(calm_clips)
calm_clips = calm_clips[:target_count_per_class]

for clip_name in tqdm(calm_clips, desc='Copying calm clips'):
    src_path = os.path.join(input_root, 'calm', clip_name)
    dst_path = os.path.join(output_root, 'calm', clip_name)
    shutil.copytree(src_path, dst_path)

def augment_clip(src_clip_path, dst_clip_path):
    os.makedirs(dst_clip_path, exist_ok=True)
    frame_files = sorted(os.listdir(src_clip_path))

    # Choose random augmentation type per clip
    aug_type = random.choice(['flip', 'brightness', 'rotate', 'none'])

    # Fix random factor per clip
    if aug_type == 'brightness':
        factor = random.uniform(0.7, 1.3)
    elif aug_type == 'rotate':
        angle = random.choice([-10, 10])

    for frame_file in frame_files:
        frame_path = os.path.join(src_clip_path, frame_file)
        frame = cv2.imread(frame_path)
```

```
        if aug_type == 'flip':
            frame = cv2.flip(frame, 1)

        elif aug_type == 'brightness':
            frame = np.clip(frame * factor, 0, 255).astype(np.uint8)

        elif aug_type == 'rotate':
            center = (frame.shape[1] // 2, frame.shape[0] // 2)
            M = cv2.getRotationMatrix2D(center, angle, 1.0)
            frame = cv2.warpAffine(frame, M, (frame.shape[1],
frame.shape[0]))

        frame_output_path = os.path.join(dst_clip_path, frame_file)
        cv2.imwrite(frame_output_path, frame)

# Copy existing dynamic clips
dynamic_clips = os.listdir(os.path.join(input_root, 'dynamic'))
existing_count = len(dynamic_clips)
additional_needed = target_count_per_class - existing_count

for clip_name in tqdm(dynamic_clips, desc='Copying existing dynamic
clips'):
    src_path = os.path.join(input_root, 'dynamic', clip_name)
    dst_path = os.path.join(output_root, 'dynamic', clip_name)
    shutil.copytree(src_path, dst_path)

# Duplicate and augment random dynamic clips
for i in tqdm(range(additional_needed), desc='Oversampling dynamic clips w/
augmentation'):
    src_clip = random.choice(dynamic_clips)
    src_path = os.path.join(input_root, 'dynamic', src_clip)
    new_clip_name = f'{src_clip}_aug{i}'
    dst_path = os.path.join(output_root, 'dynamic', new_clip_name)
    augment_clip(src_path, dst_path)

base_path = './data'
datasets = ['raw', 'processed', 'balanced']
classes = ['dynamic', 'calm']
```

```
counts = {dataset: {cls: 0 for cls in classes} for dataset in datasets}

# Count the number of files in each class subfolder for each dataset
for dataset in datasets:
    for cls in classes:
        folder_path = os.path.join(base_path, dataset, cls)
        if os.path.exists(folder_path):
            counts[dataset][cls] = len([f for f in
os.listdir(folder_path)])

# Prepare data for plotting
x = np.arange(len(classes))  # the label locations
width = 0.25  # the width of the bars

fig, ax = plt.subplots(figsize=(8, 6))

# Plot bars for each dataset
for i, dataset in enumerate(datasets):
    values = [counts[dataset][cls] for cls in classes]
    ax.bar(x + i * width, values, width, label=dataset)

# Customize the plot
ax.set_xlabel('Class')
ax.set_ylabel('Number of Clips')
ax.set_title('Number of Video Clips per Class')
ax.set_xticks(x + width)
ax.set_xticklabels(classes)
ax.legend()

# Show values on top of bars
for i, dataset in enumerate(datasets):
    values = [counts[dataset][cls] for cls in classes]
    for xi, yi in zip(x + i * width, values):
        ax.text(xi, yi + 0.5, str(yi), ha='center', va='bottom')

plt.tight_layout()
plt.show()
```

Лістинг А.2 — Скрипт для препроцесингу даних

```
import os
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import transforms
from dataset import VideoClipsDataset
from sklearn.metrics import accuracy_score
from tqdm import tqdm
from sklearn.model_selection import train_test_split

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)

# Training parameters
clip_length = 64
batch_size = 16
num_epochs = 50
learning_rate = 0.00005

transform = transforms.Compose([
    transforms.Resize((112, 112)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5],
                         std=[0.5, 0.5, 0.5])
])

# Load dataset paths
root_dir = "./data/balanced"
samples = []
for label in ["dynamic", "calm"]:
    label_dir = os.path.join(root_dir, label)
    for clip_folder in os.listdir(label_dir):
        clip_path = os.path.join(label_dir, clip_folder)
```

```
        samples.append((clip_path, 1 if label == "dynamic" else 0))

# Train / validation split
train_samples, temp = train_test_split(samples, test_size=0.2,
stratify=[label for _, label in samples], random_state=42)
val_samples, test_samples = train_test_split(temp, test_size=0.5,
stratify=[label for _, label in temp], random_state=42)

print(len(train_samples))
print(len(val_samples))
print(len(test_samples))

train_dataset = VideoClipsDataset(train_samples, clip_length=64,
transform=transform)
val_dataset = VideoClipsDataset(val_samples, clip_length=64,
transform=transform)

train_loader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=True, num_workers=4)
val_loader   = DataLoader(val_dataset, batch_size=batch_size,
shuffle=False, num_workers=4)

class Simple3DCNN(nn.Module):
    def __init__(self):
        super(Simple3DCNN, self).__init__()
        self.features = nn.Sequential(
            nn.Conv3d(3, 16, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool3d(kernel_size=2),

            nn.Conv3d(16, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool3d(kernel_size=2),

            nn.Conv3d(32, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.AdaptiveAvgPool3d((1, 1, 1))
        )
```

```
        self.classifier = nn.Linear(64, 2)

    def forward(self, x):
        x = self.features(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x

model = Simple3DCNN().to(device)

# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

train_loss_history = []
val_loss_history = []
train_acc_history = []
val_acc_history = []

# Training loop
for epoch in range(num_epochs):
    model.train()
    train_losses, train_preds, train_labels = [], [], []

    for clips, labels in tqdm(train_loader, desc=f"Epoch
{epoch+1}/{num_epochs} [Train]"):
        clips, labels = clips.to(device), labels.to(device)

        outputs = model(clips)
        loss = criterion(outputs, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        train_losses.append(loss.item())
        preds = torch.argmax(outputs, dim=1)
        train_preds.extend(preds.cpu().numpy())
```

```
            train_labels.extend(labels.cpu().numpy())

        train_acc = accuracy_score(train_labels, train_preds)
        avg_train_loss = sum(train_losses) / len(train_losses)

        # Validation loop
        model.eval()
        val_losses, val_preds, val_labels = [], [], []

        with torch.no_grad():
            for clips, labels in tqdm(val_loader, desc=f"Epoch
{epoch+1}/{num_epochs} [Val]"):
                clips, labels = clips.to(device), labels.to(device)

                outputs = model(clips)
                loss = criterion(outputs, labels)

                val_losses.append(loss.item())
                preds = torch.argmax(outputs, dim=1)
                val_preds.extend(preds.cpu().numpy())
                val_labels.extend(labels.cpu().numpy())

        val_acc = accuracy_score(val_labels, val_preds)
        avg_val_loss = sum(val_losses) / len(val_losses)

        # Save history
        train_loss_history.append(avg_train_loss)
        val_loss_history.append(avg_val_loss)
        train_acc_history.append(train_acc)
        val_acc_history.append(val_acc)

        print(f"Epoch {epoch+1}: Train Loss={avg_train_loss:.4f}, Train
Acc={train_acc:.4f}, "
              f"Val Loss={avg_val_loss:.4f}, Val Acc={val_acc:.4f}")

# Save model
torch.save(model.state_dict(), "3dcnn_dynamicity_classifier.pth")
print("Model saved as 3dcnn_dynamicity_classifier.pth")
```

```
import matplotlib.pyplot as plt

epochs = range(1, num_epochs + 1)

plt.figure(figsize=(12, 5))

# Loss plot
plt.subplot(1, 2, 1)
plt.plot(epochs, train_loss_history, label='Train Loss')
plt.plot(epochs, val_loss_history, label='Val Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss over Epochs')
plt.legend()

# Accuracy plot
plt.subplot(1, 2, 2)
plt.plot(epochs, train_acc_history, label='Train Acc')
plt.plot(epochs, val_acc_history, label='Val Acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Accuracy over Epochs')
plt.legend()

plt.tight_layout()
plt.show()
```

Лістинг А.3 — Скрипт тренування 3D CNN

```
import torch
from torchvision import transforms
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
accuracy_score
import matplotlib.pyplot as plt
from dataset import VideoClipsDataset
import numpy as np
import torch.nn as nn
```

```
from sklearn.model_selection import train_test_split
import os
from sklearn.metrics import (
    classification_report, precision_score, recall_score, f1_score,
    roc_auc_score, balanced_accuracy_score
)

from sklearn.metrics import roc_curve, auc, precision_recall_curve,
average_precision_score
import cv2

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
transform = transforms.Compose([
    transforms.Resize((112, 112)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5],
                         std=[0.5, 0.5, 0.5])
])

# Load dataset paths
root_dir = "./data/balanced"
samples = []
for label in ["dynamic", "calm"]:
    label_dir = os.path.join(root_dir, label)
    for clip_folder in os.listdir(label_dir):
        clip_path = os.path.join(label_dir, clip_folder)
        samples.   append((clip_path, 1 if label == "dynamic" else 0))

# Train / validation split
train_samples, temp = train_test_split(samples, test_size=0.2,
stratify=[label for _, label in samples], random_state=42)
val_samples, test_samples = train_test_split(temp, test_size=0.5,
stratify=[label for _, label in temp], random_state=42)

print(len(train_samples))
print(len(val_samples))
print(len(test_samples))
```

```
test_dataset = VideoClipsDataset(test_samples, clip_length=64,
transform=transform)

class Simple3DCNN(nn.Module):
    def __init__(self):
        super(Simple3DCNN, self).__init__()
        self.features = nn.Sequential(
            nn.Conv3d(3, 16, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool3d(kernel_size=2),

            nn.Conv3d(16, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool3d(kernel_size=2),

            nn.Conv3d(32, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.AdaptiveAvgPool3d((1, 1, 1))
        )
        self.classifier = nn.Linear(64, 2)

    def forward(self, x):
        x = self.features(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x

# Load trained model
model = Simple3DCNN().to(device)
model.load_state_dict(torch.load("3dcnn_dynamicity_classifier.pth"))
model.eval()

all_preds, all_labels, all_probs = [], [], []

with torch.no_grad():
    for pixels, label in test_dataset:
        pixel_values = pixels.permute(0, 1, 2, 3).unsqueeze(0).to(device)
```

```
        outputs = model(pixel_values)
        preds = torch.argmax(outputs, dim=1)
        probs = torch.softmax(outputs, dim=1)

        all_preds.append(preds.cpu().numpy()[0])
        all_probs.append(probs.cpu().numpy()[0])
        all_labels.append(label)

all_probs = np.array(all_probs)

# Confusion matrix
cm = confusion_matrix(all_labels, all_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Calm',
'Dynamic'])
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
plt.show()

acc = accuracy_score(all_labels, all_preds)
print(f"Validation Accuracy: {acc:.4f}")

# Classification report
print("\nClassification Report:")
print(classification_report(all_labels, all_preds, target_names=['Calm',
'Dynamic']))

# Precision, Recall, F1 separately (macro = avg over classes)
precision = precision_score(all_labels, all_preds, average='macro')
recall = recall_score(all_labels, all_preds, average='macro')
f1 = f1_score(all_labels, all_preds, average='macro')
balanced_acc = balanced_accuracy_score(all_labels, all_preds)

print(f"Macro Precision: {precision:.4f}")
print(f"Macro Recall: {recall:.4f}")
print(f"Macro F1 Score: {f1:.4f}")
print(f"Balanced Accuracy: {balanced_acc:.4f}")

# ROC Curve
```

```
fpr, tpr, _ = roc_curve(all_labels, all_probs[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.grid()
plt.show()

# Precision-Recall Curve
precision, recall, _ = precision_recall_curve(all_labels, all_probs[:, 1])
avg_prec = average_precision_score(all_labels, all_probs[:, 1])

plt.figure(figsize=(6, 6))
plt.plot(recall, precision, color='green', label=f'PR Curve (AP =
{avg_prec:.2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='upper right')
plt.grid()
plt.show()

# Video segmentaion
clip_length = 64
fps = 30
resize_dims = (112, 112)
labels = ['Calm', 'Dynamic']
colors = {'Calm': (0, 255, 0), 'Dynamic': (255, 0, 0)}

transform = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize(resize_dims),
    transforms.ToTensor(),
```

```
    transforms.Normalize(mean=[0.5, 0.5, 0.5],
                         std=[0.5, 0.5, 0.5])
])

def classify_clip(clip_frames):
    clip_tensor = torch.stack([transform(f) for f in clip_frames])  # [T,
C, H, W]
    clip_tensor = clip_tensor.permute(1, 0, 2, 3).unsqueeze(0).to(device)
# [1, C, T, H, W]
    with torch.no_grad():
        output = model(clip_tensor)
        pred = torch.argmax(output, dim=1).item()
    return labels[pred]

def segment_video(input_path, output_path):
    cap = cv2.VideoCapture(input_path)
    frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    fps = cap.get(cv2.CAP_PROP_FPS)  # capture original fps
    read_frames, clip_frames = 0, []
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = None

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

        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        clip_frames.append(frame_rgb)
        read_frames += 1

        if len(clip_frames) == clip_length:
            # Predict
            label = classify_clip(clip_frames)

            # Draw result on each frame
            for i in range(clip_length):
                color = colors[label]
```

```
                cv2.rectangle(clip_frames[i], (0, 0),
(clip_frames[i].shape[1]-1, clip_frames[i].shape[0]-1), color, 10)
                cv2.putText(clip_frames[i], label, (20, 60),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, color, 5,
cv2.LINE_AA)

            # Write output frames at original fps
            for f in clip_frames:
                f_bgr = cv2.cvtColor(f, cv2.COLOR_RGB2BGR)
                if out is None:
                    height, width, _ = f_bgr.shape
                    out = cv2.VideoWriter(output_path, fourcc, fps, (width,
height))  # original fps
                out.write(f_bgr)

            clip_frames = []

    cap.release()
    if out:
        out.release()

    print(f"Segmented video saved to {output_path}")

for video in os.listdir("./test_data"):
    if video.endswith(".mp4"):
        segment_video(f'./test_data/{video}',
f'./test_data/3dcnn/segmented_{video}')
```

Лістинг А.4 — Скрипт для евалюації 3D CNN

```
import os
import torch
from torchvision import transforms
from sklearn.model_selection import train_test_split
from transformers import TimesformerForVideoClassification,
TimesformerConfig, Trainer, TrainingArguments
from transformers import VideoMAEImageProcessor
```

```
from dataset import VideoClipsDataset
import numpy as np

# Check for GPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)

# Image processor for resizing/norm
processor = VideoMAEImageProcessor.from_pretrained("MCG-NJU/videomae-base")

transform = transforms.Compose([
    transforms.Resize((112, 112)),
    transforms.ToTensor(),
    transforms.Normalize(mean=processor.image_mean,
std=processor.image_std)
])

# Load dataset paths
root_dir = "./data/balanced"
samples = []
for label in ["dynamic", "calm"]:
    label_dir = os.path.join(root_dir, label)
    for clip_folder in os.listdir(label_dir):
        clip_path = os.path.join(label_dir, clip_folder)
        samples.append((clip_path, 1 if label == "dynamic" else 0))

# Train / validation split
train_samples, temp = train_test_split(samples, test_size=0.2,
stratify=[label for _, label in samples], random_state=42)
val_samples, test_samples = train_test_split(temp, test_size=0.5,
stratify=[label for _, label in temp], random_state=42)

print(len(train_samples))
print(len(val_samples))
print(len(test_samples))

# Datasets
```

```
train_dataset = VideoClipsDataset(train_samples, clip_length=64,
transform=transform)
val_dataset = VideoClipsDataset(val_samples, clip_length=64,
transform=transform)

# Hugging Face Datasets wrappers
def collate_fn(batch):
    pixel_values = torch.stack([item[0] for item in batch])
    labels = torch.tensor([item[1] for item in batch])
    return {"pixel_values": pixel_values.permute(0, 2, 1, 3, 4), "labels":
labels}

#  Model config and setup
config = TimesformerConfig.from_pretrained("MCG-NJU/videomae-base")
config.num_labels = 2

model = TimesformerForVideoClassification.from_pretrained("MCG-
NJU/videomae-base", config=config)
model.to(device)

# Training args
training_args = TrainingArguments(
    output_dir="./timesformer-finetune",
    eval_strategy="epoch",
    save_strategy="epoch",
    num_train_epochs=10,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    learning_rate=5e-5,
    weight_decay=0.01,
    logging_steps=10,
    save_total_limit=5,
    fp16=torch.cuda.is_available(),
    report_to="none",
    logging_strategy="epoch",
    logging_dir="./logs",
)
```

```
# Accuracy metric
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = np.argmax(logits, axis=1)
    accuracy = (preds == labels).mean()
    return {"accuracy": accuracy}

# Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    compute_metrics=compute_metrics,
    data_collator=collate_fn
)

# Train the model
trainer.train()

import pandas as pd
import matplotlib.pyplot as plt

# Extract log history
logs = trainer.state.log_history

# Convert to DataFrame
log_df = pd.DataFrame(logs)

# Show the log dataframe to see available columns
print(log_df.head())

loss = log_df[~log_df["loss"].isna()]["loss"]
eval_loss = log_df[~log_df["eval_loss"].isna()]["eval_loss"]
eval_accuracy = log_df[~log_df["eval_accuracy"].isna()]["eval_accuracy"]

epochs = range(1, 10 + 1)
```

```
plt.figure(figsize=(12, 5))

# Loss plot
plt.subplot(1, 2, 1)
plt.plot(epochs, loss, label='Train Loss')
plt.plot(epochs, eval_loss, label='Val Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss over Epochs')
plt.legend()

# Accuracy plot
plt.subplot(1, 2, 2)
plt.plot(epochs, eval_accuracy, label='Val Acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Accuracy over Epochs')
plt.legend()

plt.tight_layout()
plt.show()
```

Лістинг А.5 — Скрипт для тренування Timesformer

```
import cv2
import torch
from torchvision import transforms
from transformers import TimesformerForVideoClassification,
VideoMAEImageProcessor
from PIL import Image
import numpy as np
from dataset import VideoClipsDataset

import os
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
accuracy_score
import matplotlib.pyplot as plt
```

```
from sklearn.metrics import (
    classification_report, precision_score, recall_score, f1_score,
    roc_auc_score, balanced_accuracy_score
)

from sklearn.metrics import roc_curve, auc, precision_recall_curve,
average_precision_score

model_path = "./timesformer-finetune/checkpoint-2000"
clip_length = 64
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)

model = TimesformerForVideoClassification.from_pretrained(model_path)
model.to(device)
model.eval()

processor = VideoMAEImageProcessor.from_pretrained("MCG-NJU/videomae-base")

transform = transforms.Compose([
    transforms.Resize((112, 112)),
    transforms.ToTensor(),
    transforms.Normalize(mean=processor.image_mean,
std=processor.image_std)
])

# Load dataset paths
root_dir = "./data/balanced"
samples = []
for label in ["dynamic", "calm"]:
    label_dir = os.path.join(root_dir, label)
    for clip_folder in os.listdir(label_dir):
        clip_path = os.path.join(label_dir, clip_folder)
        samples.append((clip_path, 1 if label == "dynamic" else 0))

# Train / validation split
```

```
train_samples, temp = train_test_split(samples, test_size=0.2,
stratify=[label for _, label in samples], random_state=42)
val_samples, test_samples = train_test_split(temp, test_size=0.5,
stratify=[label for _, label in temp], random_state=42)

print(len(train_samples))
print(len(val_samples))
print(len(test_samples))

test_dataset = VideoClipsDataset(test_samples, clip_length=64,
transform=transform)

all_preds, all_labels, all_probs = [], [], []

with torch.no_grad():
    for pixels, label in test_dataset:
        pixel_values = pixels.permute(1, 0, 2, 3).unsqueeze(0).to(device)

        with torch.no_grad():
            outputs = model(pixel_values=pixel_values)
            logits = outputs.logits
            predicted_class = logits.argmax(dim=-1).item()
            probs = torch.softmax(logits, dim=1)

        all_preds.append(predicted_class)
        all_probs.append(probs.cpu().numpy()[0])
        all_labels.append(label)

all_probs = np.array(all_probs)

cm = confusion_matrix(all_labels, all_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Calm',
'Dynamic'])
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
plt.show()

# Print accuracy
```

```
acc = accuracy_score(all_labels, all_preds)
print(f"Validation Accuracy: {acc:.4f}")

# Classification report
print("\nClassification Report:")
print(classification_report(all_labels, all_preds, target_names=['Calm',
'Dynamic']))

# Precision, Recall, F1 separately (macro = avg over classes)
precision = precision_score(all_labels, all_preds, average='macro')
recall = recall_score(all_labels, all_preds, average='macro')
f1 = f1_score(all_labels, all_preds, average='macro')
balanced_acc = balanced_accuracy_score(all_labels, all_preds)

print(f"Macro Precision: {precision:.4f}")
print(f"Macro Recall: {recall:.4f}")
print(f"Macro F1 Score: {f1:.4f}")
print(f"Balanced Accuracy: {balanced_acc:.4f}")

# ROC Curve
fpr, tpr, _ = roc_curve(all_labels, all_probs[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.grid()
plt.show()

# Precision-Recall Curve
precision, recall, _ = precision_recall_curve(all_labels, all_probs[:, 1])
avg_prec = average_precision_score(all_labels, all_probs[:, 1])

plt.figure(figsize=(6, 6))
```

```
plt.plot(recall, precision, color='green', label=f'PR Curve (AP =
{avg_prec:.2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='upper right')
plt.grid()
plt.show()

colors = {
    0: (0, 255, 0),
    1: (255, 0, 0)
}
labels_map = {
    0: "Calm",
    1: "Dynamic"
}

def classify_clip(clip_frames):
    frames = [transform(Image.fromarray(f)) for f in clip_frames]
    clip_tensor = torch.stack(frames).permute(0, 1, 2,
3).unsqueeze(0).to(device)

    with torch.no_grad():
        outputs = model(pixel_values=clip_tensor)
        logits = outputs.logits
        predicted_class = logits.argmax(dim=-1).item()

    return predicted_class

def segment_video(input_path, output_path):
    cap = cv2.VideoCapture(input_path)
    frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    fps = cap.get(cv2.CAP_PROP_FPS)
    read_frames, clip_frames = 0, []
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = None
```

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

        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        clip_frames.append(frame_rgb)
        read_frames += 1

        if len(clip_frames) == clip_length:
            # Predict label for this clip
            label = classify_clip(clip_frames)

            # Draw annotation
            for i in range(clip_length):
                color = colors[label]
                cv2.rectangle(clip_frames[i], (0, 0),
(clip_frames[i].shape[1]-1, clip_frames[i].shape[0]-1), color, 8)
                cv2.putText(clip_frames[i], labels_map[label], (20, 60),
                            cv2.FONT_HERSHEY_SIMPLEX, 2, color, 5,
cv2.LINE_AA)

            # Write annotated frames to output video
            for f in clip_frames:
                f_bgr = cv2.cvtColor(f, cv2.COLOR_RGB2BGR)
                if out is None:
                    height, width, _ = f_bgr.shape
                    out = cv2.VideoWriter(output_path, fourcc, fps, (width,
height))
                out.write(f_bgr)

            clip_frames = []

    cap.release()
    if out:
        out.release()

    print(f"\nSegmented video saved to {output_path}")
```

```
for video in os.listdir("./test_data"):
    if video.endswith(".mp4"):
        segment_video(f'./test_data/{video}',
f'./test_data/transformer/segmented_{video}')
```

Лістинг А.6 — Скрипт для евалюації Timesformer

ДОДАТОК Б — Диск із кваліфікаційною роботою бакалавра