Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:
«СИСТЕМА ВИЯВЛЕННЯ ФЕЙКОВОГО МУЛЬТИМЕДІЙНОГО КОНТЕНТУ»

Виконав: студент 2 курсу, групи 1БС-24м
спеціальності 125 Кібербезпека та захист інформації

_____ Назарій ЛЮДВА

Керівник: к. т. н., доцент каф. ЗІ
_____ Леонід КУПЕРШТЕЙН
«16» грудня 2025 р.

Опонент: к. т. н., доцент каф. ПЗ
_____ Наталя БАБЮК
«16» грудня 2025 р.

Допущено до захисту
В. о. зав. каф. ЗІ д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«16» грудня 2025 р.

Вінниця ВНТУ – 2025 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 125 «Кібербезпека та захист інформації»
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
В. о. зав. каф. ЗІ д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«24» 09 2025 року

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

Людві Назарію Вікторовичу

1. Тема роботи: «Система виявлення фейкового мультимедійного контенту» керівник роботи: Куперштейн Леонід Михайлович, к. т. н., доцент кафедри ЗІ, затверджені наказом №313 ректора ВНТУ від «24» вересня 2025 р.
2. Строк подання студентом роботи «16» грудня 2025 р.
3. Вихідні дані до роботи:

– формат додатку – локальний веб-застосунок (Streamlit) для аналізу відео та аудіо;

– архітектура – модульна мультимодальна система з паралельними гілками відео- та аудіоаналізу і шаром агрегування результатів (SAFE_OR);

– мова програмування – Python 3.11;

4. Зміст текстової частини: Вступ. 1. Аналіз предметної області. 2. Розробка архітектурних рішень. 3. Програмна реалізація системи. 4. Економічна частина. Висновки. Список використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу: Загальна модульна архітектура системи мультимодальної детекції діпфейків. Алгоритм попередньої обробки відеоданих для подачі у відеомодуль. Алгоритм попередньої обробки аудіосигналу. Алгоритм донавчання відеодетектора. Алгоритм роботи відеомодуля resnet50. Алгоритм донавчання аудіомодуля на базі wav2vec. Алгоритм роботи аудіомодуля в режимі застосування. Алгоритм роботи модуля агрегування результатів

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Леонід КУПЕРШТЕЙН, к.т.н., доц. каф. ЗІ	25.09.25	06.10.25
2	Леонід КУПЕРШТЕЙН, к.т.н., доц. каф. ЗІ	25.09.25	03.11.25
3	Леонід КУПЕРШТЕЙН, к.т.н., доц. каф. ЗІ	25.09.25	17.11.25
4	Олександр ЛЕСЬКО, к. е. н., доц., зав. каф. ЕПВМ	18.11.25	22.11.25

7. Дата видачі завдання «24» вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Приміт
1	Аналіз завдання. Вступ	24.09.2025 – 26.09.2025	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	27.09.2025 – 07.09.2025	
3	Науково-технічне обгрунтування	11.10.2025 – 22.10.2025	
4	Проєктування архітектури мультиагентної системи та специфікація компонентів	23.10.2025 – 26.10.2025	
5	Розроблення програмної структури системи виявлення фейкового мультимедійного контенту	27.10.2025 – 02.11.2025	
6	Інтеграція модулів, розширення функціоналу та реалізація вебінтерфейсу	03.11.2025 – 10.11.2025	
7	Тестування системи та підготовка результатів експериментів	10.11.2025 – 17.11.2025	
8	Розробка економічного розділу	18.11.2025 – 22.11.2025	
9	Оформлення пояснювальної записки	23.11.2025 – 29.11.2025	
10	Попередній захист та доопрацювання МКР	29.11.2025 – 11.12.2025	
11	Перевірка на наявність текстових запозичень	12.12.2025 – 15.12.2025	
12	Представлення МКР до захисту, рецензування	16.12.2025 – 19.12.2025	
13	Захист МКР	19.12.2025 – 23.12.2025	

Студент _____ Назарій ЛЮДВА

Керівник роботи _____ Леонід КУПЕРШТЕЙН

АНОТАЦІЯ

УДК 681.325.5

Людва Н. Система виявлення фейкового мультимедійного контенту. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2025. 106 с.

Укр. мовою. Бібліогр.: 62 назв; рис.: 38; табл.: 20.

Магістерська робота присвячена розробленню програмної системи для виявлення мультимедійних фейків у відеоконтенті шляхом поєднання відеомодуля та аудіомодуля. Відеомодуль виконує детекцію обличчя, відбір кадрів і класифікацію на базі ResNet50 з донавченою класифікаційною головою, а аудіомодуль здійснює сегментацію сигналу та класифікацію мовлення на базі Wav2Vec2 у задачі виявлення підробки голосу. Для підвищення надійності реалізовано агрегацію результатів на рівні файла з формуванням підсумкового вердикту, а також вебпрототип із візуалізацією і статистикою роботи модулів. Проведено валідаційне оцінювання, підбір порогів і порівняння з наявними інструментами, що підтвердило практичну придатність підходу та наявність складних випадків, зокрема помилкових спрацювань. У економічному розділі оцінено витрати на розроблення та обґрунтовано доцільність використання системи.

Ключові слова: діпфейк, мультимодальна детекція, відеоаналіз, аудіоаналіз, ResNet50, Wav2Vec2, детекція обличчя, класифікація, агрегація результатів, кібербезпека..

ABSTRACT

Liudva N. System for detecting fake multimedia content. Master's qualification work in specialty 125 – Cybersecurity, educational program – Security of information and communication systems. Vinnytsia: VNTU, 2025. 106 p.

In Ukrainian language. Bibliography: 62 titles; figures: 38; tables: 20

The master's thesis is devoted to the development of a software system for detecting multimedia fakes in video content by combining a video module and an audio module. The video module performs face detection, frame selection and classification based on ResNet50 with a pre-trained classification head, and the audio module performs signal segmentation and speech classification based on Wav2Vec2 in the task of detecting voice forgery. To increase reliability, aggregation of results at the file level with the formation of a final verdict was implemented, as well as a web prototype with visualization and statistics of the modules. Validation evaluation, threshold selection and comparison with existing tools were carried out, which confirmed the practical applicability of the approach and the presence of complex cases, in particular false positives. The economic section estimates the costs of development and justifies the feasibility of using the system.

Keywords: deepfake, multimodal detection, video analysis, audio analysis, ResNet50, Wav2Vec2, face detection, classification, aggregation of results, cybersecurity.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі мультимедійні платформи стали базовим способом комунікації, а відео зі звуком — головним носієм довіри. Паралельно зростає частка синтетичних і маніпульованих матеріалів, у яких одночасно підміняються візуальні й акустичні ознаки особи. Такі підробки пришвидшують шахрайські сценарії, підривають процедури ідентифікації та впливають на публічні рішення. В умовах війни аудіовізуальні фейки перетворюються на інструмент інформаційної боротьби, а отже потребують технічно відтворюваної експертизи на рівні сигналу, що підтверджується артефактами й проміжними метаданими.

Актуальність роботи полягає в побудові офлайн-конвеєра експертизи мультимедійних фейків, здатного стабільно працювати з матеріалами низької якості, несталим FPS, багаторазовим перекодуванням і реверберацією, підтримувати україномовний аудіоконтент і формувати звіти, придатні для технічної оцінки. На відміну від загальних підходів до реагування на інциденти безпеки, акцент переноситься на детальне виявлення мультимедійної підробки в змішаних відео–аудіо потоках з перехресними перевірками узгодженості між модальностями.

Об’єктом дослідження є процес виявлення фейкового мультимедійного контенту.

Предметом дослідження є методи і засоби виявлення фейкового мультимедійного контенту.

Метою магістерської роботи є покращення процесу виявлення фейкового мультимедійного контенту за рахунок аналізу відео та аудіо складової мультимедійного контенту.

Для досягнення мети необхідно розв’язати такі задачі:

- виконати огляд предметної області та визначити вимоги до системи;
- підготувати навчальні дані для відео й аудіо;
- обґрунтувати вибір базових моделей для відео та аудіо;
- виконати донавчання моделей для детекції підробок;
- реалізувати модулі аналізу відео й аудіо у програмній системі;

- визначити правила об’єднання результатів та формування підсумкового вердикту;
- провести експериментальну перевірку, підбір порогів і порівняння з аналогами;

Наукова новизна полягає у запропонованій модульній архітектурі виявлення мультимедійних фейків, де відео і аудіо аналізуються незалежними моделями, а підсумковий висновок формується узгодженими правилами агрегації на рівні файла. У роботі формалізовано порядок підготовки даних і донавчання ResNet50 для відео та Wav2Vec для аудіо в межах єдиного пайплайна, а також визначено процедуру підбору порогів рішень і структуру артефактів, що зберігаються для подальшої технічної перевірки результату.

Практична цінність полягає у створенні прототипу системи виявлення мультимедійних фейків, придатної для використання як інструмент первинної технічної перевірки відео і аудіоматеріалів із формуванням зрозумілого звіту та збереженням проміжних результатів аналізу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття та класифікація фейкового мультимедійного контенту

Мультимедія розуміється як узгоджене подання інформації кількома каналами – передусім зоровим і слуховим – які людина сприймає як єдину подію [1]. На практиці це короткі вертикальні ролики, стріми, відеодзвінки, подкасти з візуальним супроводом, інтерактивні презентації. До основних видів мультимедія відносимо статичні зображення та інфографіку, анімацію, відео з різною частотою кадрів, мовлення та музичний супровід, а також змішані формати з елементами взаємодії. Критичною є синхронізація модальностей, коли зображення губ, міміка і рухи синхронні з мовленням та інтонаціями, повідомлення сприймається переконливіше й запам’ятовується швидше [2]. У середовищі соціальних платформ це напряму впливає на перші секунди перегляду і рішення користувача «дивитися/ділитися», що визначає охоплення матеріалу. Узагальнену різницю в темпах охоплення аудиторії правдивими й неправдивими повідомленнями демонструє графік на рисунку 1.1 [3], де по горизонталі показано кількість користувачів, а по вертикалі — середній час, необхідний для досягнення такого охоплення.

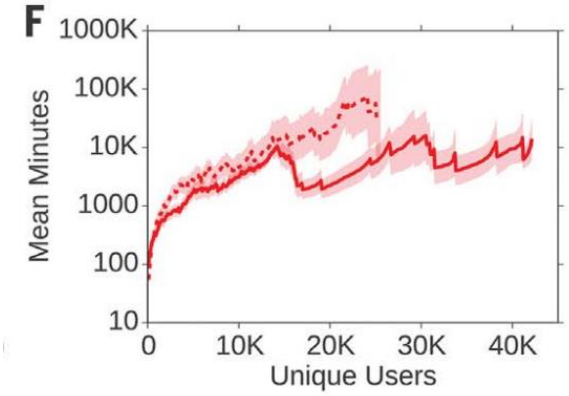


Рисунок 1.1 — Швидкість досягнення аудиторії неправдивими та правдивими повідомленнями, середній час до охоплення N користувачів

Панель F у Vosoughi–Roy–Aral показує, скільки середньо часу потрібно, щоб каскад досягнув певної кількості унікальних користувачів, по горизонталі кількість

користувачів, по вертикалі час у хвилинах на логарифмічній шкалі, дві криві відрізняють типи неправдивих новин (політичні проти інших), нижча крива означає швидше охоплення, вища повільніше, напівпрозора смуга відображає розкид оцінок

Мультимедійний контент розглядається не як ізольований файл, а як динамічний об’єкт із життєвим циклом, створення (або запис), редагування, первинна публікація, ремікси та репості на інших майданчиках, архівація чи видалення. На кожній фазі залишаються технічні сліди – параметри кодування, часові позначки, платформні метадані, характерні артефакти повторного перекодування. Саме ці сліди надалі використовуються для технічної перевірки достовірності. У повсякденних комунікаціях переважають короткі відеоформати, що споживаються з мобільних пристроїв, де автопрогравання й рекомендаційні алгоритми підсилюють видимість матеріалу в перші години після публікації. Практичний масштаб загрози для бізнес-середовища у 2022 і 2024 роках подано на рисунку 1.2 [4].

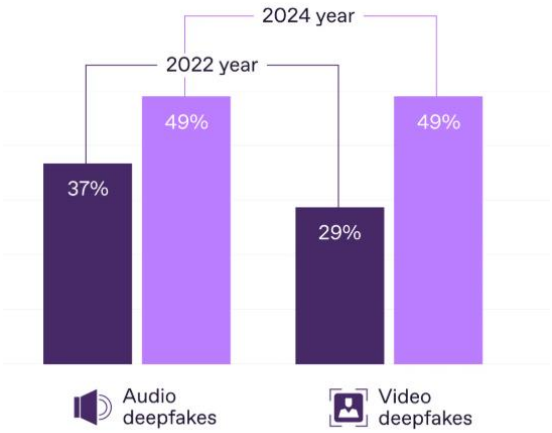


Рисунок 1.2 — Поширеність інцидентів із аудіо та відеодіпфейками серед компаній у 2022 проти 2024 років за даними Regula

Окремі загроз становлять атаки на біометричні та голосово-візуальні системи автентифікації . У сценаріях перевірки особи за обличчям і голосом застосовуються штучно згенеровані або перетворені зразки, які підвищують ймовірність хибного прийняття. Для систем розпізнавання мовлення та комп’ютерного зору характерні «adversarial» приклади — акустично чи візуально малопомітні збурення, здатні змусити модель помилитися у ключових словах, ідентифікації обличчя чи виявленні подій у

кадрі. У комплексних IoT-екосистемах і голосових інтерфейсах ризики посилюються тим, що відео і звук часто є єдиними каналами керування.

Фейковим мультимедійним контентом у роботі також є аудіовізуальний матеріал, у якому порушена хоча б одна з чотирьох опор правдивості: зміст, ідентичність, контекст або походження. Порушення змісту проявляється у підміні смислу через монтаж, виривання фраз або накладання «чужого» озвучення, це породжує стійкі хибні наративи і може провокувати панічні настрої в кризових умовах. Порушення ідентичності стосується невідповідності обличчя чи голосу реальній особі та безпосередньо загрожує фінансовою безпекою і біометричним процедурам, у відомих інцидентах клонований голос і підміна обличчя використовувалися для примусу до платежів або надання доступів. Оцінку середніх збитків від діпфейків подано на рисунку 1.3 [5].

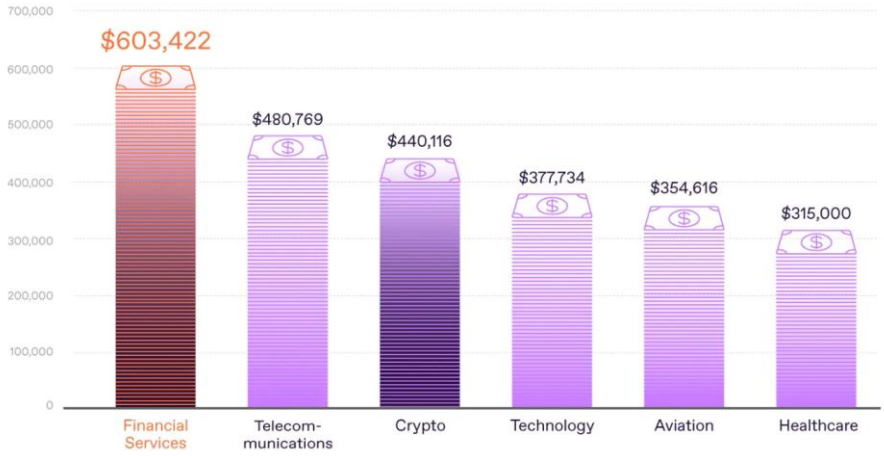


Рисунок 1.3 — Середні річні збитки від діпфейків у різних секторах

Порушення контексту виникає, коли місце, час або умови зйомки не збігаються із заявленими, під час війни такий прийом використовується для дискредитації оборонних рішень і деморалізації населення. Порушення походження означає відсутність прозорого ланцюга публікації, маніпуляції з первинним джерелом і масове дзеркалювання, через що спростування затягується, а шкода масштабується кросплатформним тиражуванням. Найвразливіші напрями застосування діпфейків у корпоративній практиці подано на наступному рисунку.

Механіка поширення підробок спирається на поєднання відкритих і закритих каналів. Короткі відеосервіси формують «вікно можливостей» завдяки ранньому залученню аудиторії, приватні чати і закриті групи прискорюють неформальне тиражування, а кросплатформні републікації дають матеріалу «друге дихання» вже після локальних спростувань. У воєнний час мультимедійні фейки стають інструментом інформаційної війн, синхронізовані відео й аудіо звернення від імені публічних осіб, постановні «зізнання», змонтовані «докази» подій використовуються для підриву довіри до інституцій, впливу на рішення у сфері безпеки та економіки, а також для дестабілізації громадської думки. Щоб зафіксувати, на що саме звертати увагу під час первинної кваліфікації, нижче на рисунку 1.4 подано узагальнення ключових ознак фейкового мультимедійного контенту [6].

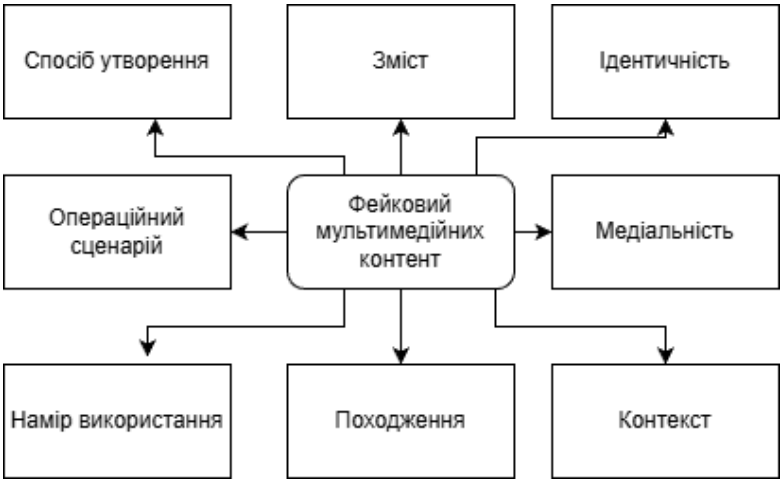


Рисунок 1.4 —Загальні ознаки фейкового мультимедійного контенту

Поняття змісту охоплює смислову узгодженість між тим, що показано у відеоряді, і тим, що відтворюється в аудіодоріжці, а також відповідність заявленій події, у цьому вимірі фіксуються монтажні підміни, виривання фраз, накладання чужого озвучення і невідповідність жестів мовленню. Далі природно переходить увага до ідентичності, де зіставляються обличчя та голос із референтними зразками конкретної особи і контролюються біометричні індикатори на кшталт мікроміміки, артикуляції, тембру та просодики, що дозволяє виявляти клонування або підміну. Наступним кроком перевіряється контекст, який включає місце, час і умови зйомки, характер освітлення та акустику середовища, тут звіряються метадані, погодні та геолокаційні маркери, фонові події у кадрі. Логічне завершення блоку опор становить походження, тобто історія

створення і публікації об’єкта з урахуванням першого джерела, ланцюга републікацій, цифрових маніфестів і відбитків записувального пристрою.

Після базових опор уточнюється матеріальність та технологія появи. Медійність визначає, з якими потоками працює система, лише відео, лише аудіо чи їх поєднання, і відповідно задає набір ознак та міжмодальні зв’язки, які потрібно врахувати. Спосіб утворення відрізняє повний синтез від маніпуляції реальним записом, комбінування фрагментів або реконтекстуалізації, причому кожен різновид залишає характерні сліди у просторових, часових та акустичних патернах.

Для коректної інтерпретації ознак береться до уваги операційний сценарій, що описує умови створення і доставки матеріалу в конкретних каналах з властивими їм бітрейтом, частотою кадрів, реверберацією та повторними перекодуваннями; саме ці параметри визначають стійкість обраних індикаторів та необхідні етапи нормалізації перед аналізом. Завершальним виміром виступає намір використання, який задає практичний сенс підробки від шахрайства, дискредитації та впливових операцій до випадків зі згодою або дослідницьких демонстрацій і тим самим впливає на пороги ухвалення рішень, формат звіту та пріоритизацію подальших дій. У такій послідовності кожна ознака доповнює попередні і формує єдину рамку для технічної верифікації мультимедійних матеріалів.

1.2 Огляд підходів до виявлення мультимедійного контенту

В огляді підходів до виявлення мультимедійних підробок доцільно відразу окреслити дві взаємодоповнювальні парадигми. Пасивна працює без вбудованих міток і спирається на природну статистику сигналів та біомеханіку сцени, виявляючи збої у текстурах кадру, мікромімиці, синхронності губ і фонем, спектрально-фазових характеристиках звуку та узгодженості з супровідним текстом [6]. Активна використовує маркери походження, цифрові маніфести, водяні знаки і записи ланцюга створення, що дає змогу оперативно підтверджувати автентичність, коли такі маркери збережені. У практиці обидва підходи поєднуються, адже за прозорого ланцюга створення рішення приймається швидко за активними індикаторами, а за їх відсутності

або пошкодженні пріоритет переходить до пасивного аналізу відео, аудіо та тексту з подальшим зіставленням результатів. Узагальнену послідовність поєднання підходів наведено на рисунку 1.5.

Рисунок 1.5 — Схема виявлення мультимедійних фейків

У відеоканалі первинні індикатори збираються на рівні окремих кадрів і в часовій динаміці. Для просторового аналізу беруть до уваги дрібномасштабні текстури шкіри та волосся, поведінку відблисків в очах і на емалі зубів, характер переходів на межах об’єктів, відповідність шумової структури типу сенсора, саме тут синтетика найчастіше «загладжує» або уніфікує деталі [7]. Часовий шар перевіряє природність моргання, мікровібрації голови, траєкторії губ і щелеп під час мовлення, а також узгодженість дрібних м’язових реакцій у часі. Коли обидві площини моделюються спільно, система стійкіше реагує на локальні вставки та гібридний монтаж, а додаткові тести фотометричної й геометричної консистентності (освітлення, тіні, глибина, оптичні віддзеркалення) допомагають виявляти невідповідності сцени реальній оптиці.

Аудіоканал додає незалежний шар доказів, чутливий до артефактів синтезу мовлення. Оцінюються спектральні та фазові властивості сигналу, мікроструктури періодичності, а також надсегментні ознаки просодики — темп, паузи, інтонаційні контури — у зіставленні з фізіологією мовця. Щоб не сплутати ефекти кодування з ознаками штучності, моделі навчають на вибірках із різними бітрейтами, реверберацією та побутовими шумами, окремо контролюється збереження індикаторів після перекодування й нормалізації рівнів.

Текстовий компонент охоплює як самостійні пости й публікації, так і пов’язані елементи — субтитри, описи, теги, автоматичні транскрипти. Тут вирішуються три

типові задачі, виявити штучно згенерований текст за стилометричними та статистичними профілями (перплексія, «burstiness», повторюваність конструкцій, нетипова відсутність «людських» помилок), відрізнити маніпулятивні наративи через виділення тверджень і їхню фактологічну перевірку за зовнішніми джерелами, а також перевірити узгодженість тексту із зображенням і звуком — чи відповідають підписи побаченим подіям, локаціям і акустичним подіям у момент часу, коли вони заявлені [7].

Коли всі три канали доступні одночасно, ключовою стає міжмодальна узгодженість. Система вирівнює артикуляцію з фонемною послідовністю, зіставляє інтонаційні акценти з мімікою, перевіряє, чи збігається акустичне середовище з видимим простором (характер реверберації, джерела шуму, напрямок звучання), і чи не суперечать текстові твердження тому, що реально відбувається у кадрі та в аудіо. Вимога одночасної правдоподібності в трьох потоках суттєво піднімає планку для зловмисника і знижує ризик хибних спрацьовувань.

Надійність рішень безпосередньо залежить від стійкості до спотворень каналу. Реальні платформи знижують бітрейт, змінюють частоту кадрів, вставляють фільтри, субтитри і стабілізацію, матеріали часто проходять через мультиперекодування. Тому датасети для навчання формуються з варіативними кодеками та рівнями якості, а моделі проходять доменну адаптацію і регуляризацію. Додатково застосовується тест перетворень, підозрілий об’єкт штучно перекодовують, ресемплюють і нормалізують, після чого перевіряють, чи зберігаються підозрілі патерни.

Оскільки противник намагається ухилятися від детекції «чистить» артефакти, варіює параметри генерації, маскує статистики або перефразовує текст у виробничих налаштуваннях доцільно поєднувати ансамблі різнорідних детекторів із рандомізованим препроцесингом, постійним моніторингом дрейфу даних і регулярним донавчанням на нових підробках. Прийняття рішень калібрується під конкретний сценарій, для відео орієнтуються на точність/повноту, для тексту на точність виявлення згенерованого або маніпулятивного вмісту та якість фактологічної верифікації. У потоках реального часу додаються часові вимоги затримка до рішення, частка об’єктів, що потребують ручного перегляду, і інтегральний ризик-бал, у якому зважено

поєднуються оцінки відео, аудіо, тексту та їхньої міжмодальної узгодженості [8-12]. У табл. 1.1 наведено порівняльну характеристику розглянутих інструментів і системи виявлення фейкового мультимедійного контенту.

Таблиця 1.1 - Порівняння інструментів виявлення мультимедійних фейків за модальністю, сильними сторонами та обмеженнями

Інструмент	Модальність	Сильні сторони	Обмеження
Reality Defender	Зображення, відео, аудіо, текст	Один сервіс для кількох типів медіа, швидка інтеграція	Потрібно передавати файли у хмару, обмежена пояснюваність
Hive Moderation (Deepfake Video)	Відео	Легке підключення для модерації, масштабування на потоки	Відвантаження даних, небагато деталей про причини вердикту
Sensity	Зображення, відео	Добре показує загальну картину фейків	Не інструмент для детального аналізу окремих файлів
Microsoft Video Authenticator	Фото, відео	Простий скринінг без складних налаштувань	Обмежена точність і покриття сценаріїв
ElevenLabs AI Speech Classifier	Аудіо	Швидко і точно в межах власної екосистеми	Не працює з усіма генераторами, потрібне завантаження
Одномодальна система детекції по відео (БДР)	Відео	Конфіденційність без хмари, придатність для технічної експертизи	Немає текстового каналу, потрібні локальні ресурси

Після методичного огляду доречно окреслити ландшафт готових інструментів. На ринку присутні хмарні детектори загального призначення, зокрема Reality Defender як мульти-модальний сервіс для зображень, відео, аудіо й тексту з веб-сканером, API та підтримкою C2PA/Content Credentials, а також Hive Moderation із окремим детектором

deepfake-відео. Свого часу Microsoft представила Video Authenticator для оцінювання ймовірності підробки у відео та фото з наголосом на обмеженнях і потребі комплексних заходів. Окремий сегмент формують платформи на кшталт Sensity, що фокусуються на моніторингу візуальних загроз і кампаній. В аудіосфері трапляються вузькоспеціальні класифікатори на кшталт ElevenLabs AI Speech Classifier, які визначають згенерованість звуку в межах конкретних моделей.

Поряд із алгоритмічними детекторами розвиваються засоби підтвердження походження. Стандарт C2PA та ініціатива Content Credentials дозволяють вбудовувати у медіа підписані маніфести з історією створення та редагувань, а водяні знаки, наприклад SynthID, маркують згенерований вміст і спрощують його ідентифікацію за умови збереженого маркера.

Порівнюючи підходи, насамперед важать умови використання у реальних робочих процесах. Хмарні API часто вимагають передавання файлів на сторонні сервери, що ускладнює роботу з чутливими матеріалами, а пояснюваність рішень і відтворюваність експертизи не завжди документовані на потрібному рівні. Активні маркери працюють швидко та надійно там, де ланцюг публікацій зберіг підпис, однак не допомагають у випадках повністю синтетичного контенту або втрати маркування в процесі перепублікацій.

1.3 Формалізація вимог та постановка задачі

У розділі сформовано цілісне уявлення про мультимедійні фейки як про клас сучасних кіберзагроз, що поєднує відео та аудіо і проявляється у вигляді інцидентів достовірності контенту з високою швидкістю поширення. Показано, що технологічна доступність генеративних і маніпулятивних інструментів, разом із соціальною інженерією та слабкими організаційними процедурами, створює умови для масштабованих атак проти користувачів і інституцій. Визначено, що ключовим фактором ризику є синхронізована підробка візуальних та акустичних ознак особи, яка ускладнює ручну експертизу й потребує автоматизованого перехресного аналізу.

Проведений аналіз сучасних кіберзагроз і підходів до виявлення підробок показує потребу у прикладному рішенні для мультимедійного детектування (відео + аудіо), здатному працювати в реальних каналах із втратами якості. Виявлені обмеження чинних рішень:

- Одномодальність або слабка перевірка міжмодальної узгодженості. Багато сервісів оцінюють лише відео або лише аудіо і не зіставляють артикуляцію з фонемами, міміку з інтонацією чи текст із подіями у кадрі. У гібридних підробках, де окремі модальності правдоподібні поодинці, відсутність перехресних тестів призводить до пропусків.
- Різке падіння точності під канальними спотвореннями. Соцплатформи знижують бітрейт, нестабілізують FPS, додають реверберацію і повторні перекодування, через що моделі, натреновані на відносно чистих даних, помиляються частіше. Без спеціальних аугментацій і доменної адаптації зростає як частка пропусків, так і хибних спрацьовувань.
- Нестійкість до replay та змішаних записів. Екранізації, повторне відтворення через побутові пристрої, вставки коротких синтетичних фрагментів у реальні ролики руйнують прості припущення детекторів. За відсутності покадрової або покусокової локалізації такі сценарії часто проходять непоміченими.
- Обмежена відтворюваність і брак прозорих індикаторів. Хмарні API нерідко повертають один узагальнений бал без проміжних доказів, карт підозри, часових інтервалів чи опису ознак. Це ускладнює внутрішню експертизу інцидентів, аудит рішень і підготовку матеріалів для службових висновків.
- Високі вимоги до ресурсів і складність інтеграції. Деякі моделі потребують значних обчислювальних ресурсів або обов’язкового завантаження даних у хмару, що суперечить вимогам конфіденційності та сповільнює впровадження. Інтеграція у наявні процеси моніторингу та офлайн-перевірки ускладнюється ліцензійними обмеженнями, квотами API і нестачею інструментів для звітування.

Метою магістерської роботи є покращення процесу виявлення фейкового мультимедійного контенту за рахунок аналізу відео та аудіо складової мультимедійного контенту.

Для досягнення мети необхідно розв’язати такі задачі:

- виконати огляд предметної області та визначити вимоги до системи;
- підготувати навчальні дані для відео й аудіо;
- обґрунтувати вибір базових моделей для відео та аудіо;
- виконати донавчання моделей для детекції підробок;
- реалізувати модулі аналізу відео й аудіо у програмній системі;
- визначити правила об’єднання результатів та формування підсумкового вердикту;
- провести експериментальну перевірку, підбір порогів і порівняння з аналогами;

Основні вимоги до розроблюваної системи:

Функціональні:

- Підтримка бінарної класифікації контенту на real / fake окремо для відео та аудіо.
- Аналіз відео з детекцією обличчя, відбором кадрів і класифікацією на базі ResNet50 з донавченою класифікаційною головою.
- Аналіз аудіо з розбиттям сигналу на короткі сегменти та класифікацією мовлення на базі Wav2Vec у задачі визначення підробки голосу.
- Послідовний запуск модулів відео та аудіо з формуванням підсумкового вердикту на рівні файла за пороговою логікою (агрегація результатів).
- Збереження результатів оцінювання у вигляді таблиць із прогнозами та підсумковими метриками для подальшого аналізу.
- Робота з локальними файлами без передавання даних у хмару.

Якісні:

- Можливість налаштування порогів рішення для відеомодуля та аудіомодуля на основі валідаційної вибірки.
- Відтворюваність експериментів через фіксовані моделі, збережені чекпойнти та однакові правила обчислення метрик

– Пояснюваність результату на рівні інтерфейсу через показ прикладу кадру з виділеним обличчям і виведення статистики класифікації (середні та максимальні оцінки).

Інтеграційні:

– Наявність вебінтерфейсу (Streamlit) для запуску аналізу та перегляду результатів на одному екрані.
– Можливість використовувати збережені чекпойнти моделей у середовищі застосунку через параметри конфігурації.
– Підтримка стандартного сценарію “завантажити файл і отримати висновок” для відео та аудіо.

Ресурсні та експлуаційні:

– Робота на доступному обчислювальному ресурсі без обов’язкової GPU, з можливістю запуску в локальному середовищі.
– Модульність рішення, де відео й аудіо модулі можуть донавчатися та оновлюватися незалежно, а агрегатор використовує їх вихідні оцінки.
– Локальне зберігання проміжних результатів і фінальних таблиць оцінювання без зовнішніх сервісів.
– Україномовний інтерфейс і повідомлення у вебзастосунку.

Відповідно до поставлених задач і вимог, у подальших розділах буде розроблено і описано прототип системи виявлення мультимедійних фейків, що працює локально з файлами без хмари та охоплює повну послідовність обробки відео й аудіо від підготовки даних до підсумкового звіту. Буде спроєктовано модулі попередньої обробки, відеодетектора на основі ResNet із донавчанням, аудіодетектора на основі Wav2Vec2.0 з тонким налаштуванням, шар перехресної перевірки узгодженості між потоками і вузол агрегування рішень з формуванням наочного звіту. Окремо буде підготовлено датасет і протокол донавчання, зафіксовано методи підвищення стійкості до канальних спотворень, визначено метрики та проведено випробування з аналізом отриманих результатів.

2 РОЗРОБКА АРХІТЕКТУРНИХ РІШЕНЬ

2.1 Загальна архітектура системи

У підрозділі 2.1 подано загальну архітектуру системи мультимодальної детекції діпфейків, яка виконує паралельний аналіз відео- та аудіокомпонент мультимедійного файла і формує підсумковий вердикт на рівні файла [13]. Інтеграція результатів модальностей реалізована за правилом SAFE_OR, яке відповідає консервативній стратегії прийняття рішення, файл позначається як імовірний діпфейк, якщо принаймні один із модулів (відео або аудіо) перевищує свій поріг та сигналізує про підробку. Такий підхід знижує ризик пропуску діпфейку у випадках, коли одна з модальностей має нижчу інформативність, містить шум або була частково спотворена. Загальна модульна архітектура системи мультимодальної детекції діпфейків зображена на рисунку 2.1

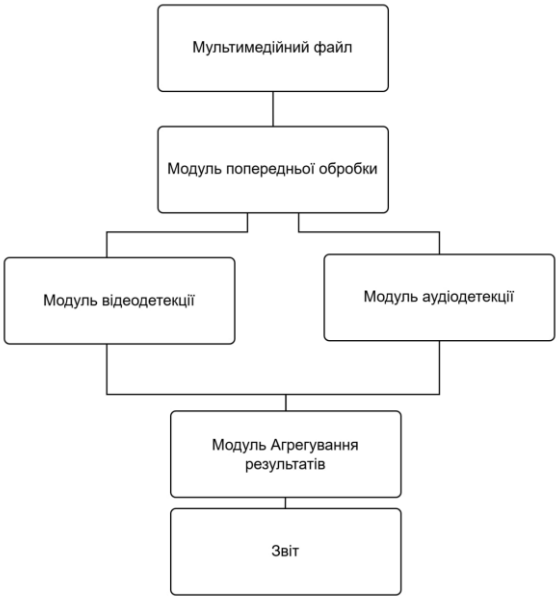


Рисунок 2.1 – Загальна модульна архітектура системи мультимодальної детекції діпфейків

Рисунок 2.1 відображає модульну структуру системи та взаємодію основних компонентів. На вході система приймає мультимедійний файл, після чого модуль

попередньої обробки виконує підготовку даних, нормалізацію контейнера, виділення відео- та аудіопотоків і фіксацію технічних метаданих (наприклад, кодеків, бітрейту, частоти кадрів і параметрів аудіо) [14]. Далі аналіз відбувається у двох паралельних гілках. Модуль відеодетекції виконує оцінювання автентичності відеоряду та формує статистики/оцінки для відеомодальності. Модуль аудіодетекції виконує аналогічне оцінювання звукової доріжки та формує результати для аудіомодальності. Після цього модуль агрегування результатів об'єднує виходи двох гілок за правилом SAFE_OR з урахуванням порогів та формує підсумковий висновок. На виході генерується звіт, який містить фінальний вердикт, узагальнені показники по кожній модальності та збережені артефакти аналізу. На рисунку 2.2 відображається загальна архітектура системи мультимодальної детекції діпфейків.

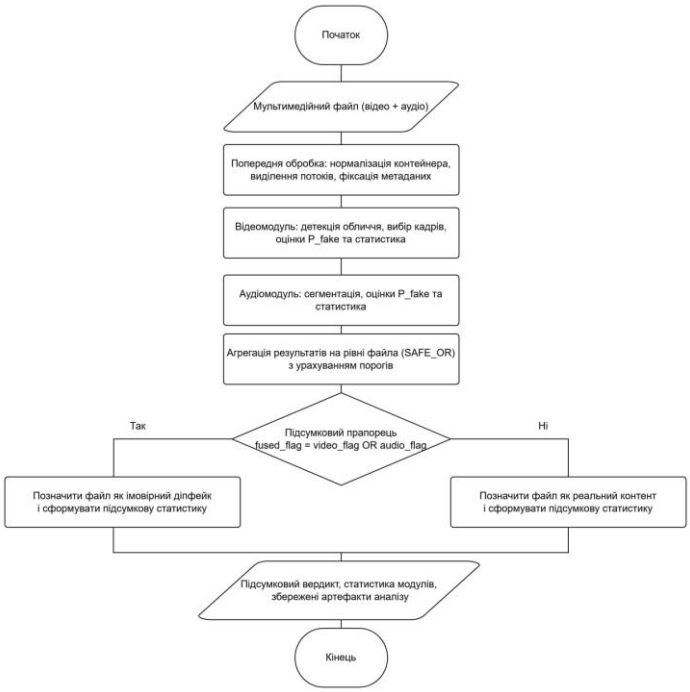


Рисунок 2.2 – Загальна архітектура системи мультимодальної детекції діпфейків

Рисунок 2.2 деталізує алгоритмічну послідовність роботи системи у форматі блок-схеми. Після отримання мультимедійного файла виконується попередня обробка, на якій готуються відео- та аудіопотоки й фіксується технічний контекст аналізу. Далі відеомодуль і аудіомодуль незалежно формують власні оцінки ризику підробки та статистику, після чого виконується агрегування результатів на рівні файла за правилом

SAFE_OR з урахуванням порогів. На підставі підсумкового прапорця fused_flag (логічне OR між рішеннями модальностей) система формує кінцевий висновок і створює звіт, у якому відображається фінальний вердикт, статистика роботи модулів і збережені артефакти аналізу.Узагальнене подання цього процесу у вигляді теоретико-множинної моделі наведено нижче.

Нехай:

V — множина кадрів відео (або кадрів із виділеною областю обличчя), що використовуються для аналізу;

A — множина аудіосегментів (фрагментів звукової доріжки), що використовуються для аналізу;

P — множина технічних параметрів та метаданих файла (кодек, FPS, бітрейт, параметри аудіо тощо);

M — множина попередньо оброблених даних, підготовлених до подачі в моделі;

X_v, X_a – множини ознак (представлень) для відео та аудіо;

R_v, R_a — результати аналізу відео- та аудіомодулів;

Θ – множина порогів прийняття рішень для модальностей;

R_f – результат інтеграції (ф’южну) на рівні файла;

U – множина збережених артефактів аналізу;

Z – підсумковий результат системи.

Тоді процес роботи системи описується послідовністю відображень:

$$D = \{V, A, P\} \to M \to \{X_v, X_a\} \to \{R_v, R_a\} \to Z$$

де:

D є вхідним описом мультимодальних даних (відео, аудіо та метадані), а перетворення відповідають основним етапам роботи системи.

Далі зафіксуємо відповідність етапів ланцюжка окремим відображенням:

$$f_{\{prep\}}: \{V, A, P\} \to M$$

$$f_{\{feat\}}: M \to \{X_v, X_a\}$$

$$f_{\{det\}}: \{X_v, X_a\} \to \{R_v, R_a\}$$

$$F_{\{SAFE_{OR}\}}: \{R_v, R_a\} \to Z$$

Тут f_prep відображає попередню обробку (нормалізацію контейнера, виділення потоків, підготовку даних і формування контексту P), f_feat виконує отримання ознак (представлень) для відео та аудіо, f_det реалізує оцінювання модальностей і формування вихідних статистик, F_SAFE_OR об'єднує результати модулів на рівні файла за правилом SAFE_OR з урахуванням порогів.

Результати модулів подамо у вигляді кортежів:

$$R_v = \left(p^{\{(v)\}\{mean\}}, p^{\{(v)\}\{max\}}, n_v, S_v, flag_v\right)$$

$$R_a = \left(p^{\{(a)\}\{mean\}}, p^{\{(a)\}\{max\}}, n_a, S_a, flag_a\right),$$

де:

p_mean та p_max – агреговані оцінки ймовірності підробки (наприклад, середнє та максимум по кадрах або сегментах), n_v і n_a – кількість використаних кадрів і сегментів, S_v і S_a – службові статистики та проміжні дані, flag_v і flag_a – бінарні рішення модулів.

Множину порогів для модальностей визначимо як:

$$\Theta = \{\tau_v, \tau_a\},$$

де:

τ_v – поріг для відеомодуля, τ_a – поріг для аудіомодуля.

Бінарні рішення модулів визначаються індикаторною функцією:

$$flag_v = I\left(p_{\{mean\}}^{\{(v)\}} \geq \tau_v\right)$$

$$flag_a = I\left(p_{\{mean\}}^{\{(a)\}} \geq \tau_a\right)$$

Інтеграція результатів на рівні файла виконується за правилом SAFE_OR:

$$flag_f = flag_v \vee flag_a$$

$$p_f = \max\left(p^{\{(v)\}\{mean\}}, p^{\{(a)\}\{mean\}}\right),$$

де:

flag_f – підсумковий прапорець (вердикт за правилом “або”), а p_f – інтегральна ризикова оцінка на рівні файла.

Результат ф’южну позначимо як:

$$R_f = (p_f, flag_f)$$

Фінальний набір результатів системи:

$$Z = \{R_v, R_a, R_f, \Theta, P, U\}$$

Фінальний набір містить результати відео- та аудіомодулів, інтегрований результат ф’южну, використані пороги, технічний контекст і метадані P, а також множину артефактів U (наприклад, проміжні списки оцінок, службові логи, візуальний вигляд або інші матеріали, що забезпечують відтворюваність та інтерпретацію аналізу).

2.2 Обґрунтування архітектурних рішень

Архітектуру системи підібрано з урахуванням реальних спотворень каналу та вимог прозорого аудиту, а саме коректна робота на матеріалах із низьким бітрейтом, несталим FPS, багаторазовим перекодуванням і реверберацією, підтримка україномовного аудіо, формування відтворюваних звітів [15]. Для відеогілки обрано ResNet — згорткову нейромережу з залишковими зв’язками, яка стабільно навчається у глибоких конфігураціях і надійно виявляє локальні просторові маркери підробок, згладжені ділянки шкіри, неприродні відблиски, порушені межі об’єктів, повторювані дрібні текстури [14]. Модель добре переноситься на ущільнені й «побиті» відео, легко адаптується під домен (нижні шари можна фіксувати, верхні — донавчати під типові спотворення), а карти активацій дають зрозумілі пояснення у звіті. Сучасні альтернативи на кшталт TimeSformer/ViT, SlowFast чи ConvNeXt розглядалися, однак

за обмежених вибірок і потреби прозорої перевірки саме ResNet забезпечує оптимальний баланс точності, стабільності та трудомісткості підтримки.

Логічним доповненням є аудіогілка на Wav2Vec2.0 — самонавченій моделі, що працює безпосередньо зі звуковою хвилею (наприклад, 16 кГц) і створює інформативні представлення без ручного проєктування спектральних ознак [16]. Вона поєднує робастність до шумів і компресії з чутливістю до тонких маркерів синтетичної мови, нетипової мікроперіодичності, фазових аномалій, характерних для TTS і конверсії голосу. Донавчання на україномовних та VoIP/телефонних записах підвищує переносимість на реальні канали. Варіанти на базі спектрограмних CNN із MFCC/CQCC, ECAPA-TDNN або AST також аналізувалися, проте зазвичай або вимагали більше даних і ресурсів, або гірше узагальнювалися на нові генератори, тому Wav2Vec2.0 виявилася практичнішим вибором.

Використання відкритих, добре описаних архітектур полегшує відтворюваність, аудит і повторне навчання, дозволяє зосередитись на прикладних аспектах доменно релевантних аугментаціях, правилах агрегування, перехресних перевірках між каналами та повному журналюванні.

Ключовою властивістю є природна сумісність обраних компонентів із шаром перехресних перевірок. Відеомодуль повертає покадрові і пофрагментні оцінки разом із тепловими картами, аудіомодуль позначає часові інтервали підвищеного ризику. Далі виконується зіставлення артикуляції з фонемною послідовністю та логіки акустичної сцени з видимим простором, що зменшує вплив хибних спрацьовувань окремих модальностей і підвищує доказову цінність інтегрального висновку.

Автономний режим обробки дає змогу застосовувати багатопрохідні процедури без обмежень затримки, що дозволяє виконувати стабілізацію, багатоваріантне кадрування облич і багатомасштабний аналіз, проводити контрольоване повторне кодування як стрес-тест, агрегувати рішення на довших часових вікнах та формувати розширений звіт з артефактами аналізу. У таких умовах доцільно обирати консервативні пороги, додавати запобіжні перевірки і розширювати журналювання, оскільки користувацький досвід не залежить від миттєвої реакції системи.

Ризики доменної невідповідності зменшуються завдяки аугментаціям під типові канальні спотворення, тестам стійкості до перетворень та частковому донавчанню на брудних вибірках. Ризик перенавчання контролюється розділенням мовців і сцен між підвибірками та заморожуванням ранніх шарів під час тонкого налаштування [17].

2.3 Модуль попередньої обробки

Перед тим як виконувати детекцію, моделі донавчаються під доменні особливості зібраних даних, оскільки попередньо навчені мережі не враховують специфіку діпфейк-артефактів у реальних умовах, повторні перекодування, нестабільний FPS, низький бітрейт, шум, різні мікрофони, компресію та монтаж. Донавчання дозволяє адаптувати класифікатор саме до ознак синтезу у відео та аудіо, підвищити стійкість до “побутових” спотворень і водночас зменшити ризик перенавчання завдяки контрольованій стратегії (коли основні шари зберігають базові узагальнені ознаки, а під задачу підлаштовується лише вихідна частина мережі) [18]. Алгоритм попередньої обробки відео наведено на рисунку 2.3.

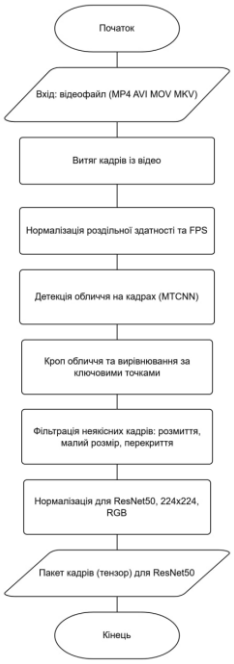


Рисунок 2.3 – Алгоритм попередньої обробки відеоданих для подачі у відеомодуль ResNet50

Вхідним є відеофайл у поширених контейнерах, після чого виконується витяг кадрів із відеопотоку. Далі застосовується нормалізація роздільної здатності та частоти кадрів, щоб привести відео до стабільного представлення і зробити різні записи порівнюваними між собою. На наступному кроці на кожному кадрі виконується детекція обличчя (MTCNN), що переводить аналіз у релевантну область, де найчастіше проявляються сліди підміни [19]. Після знаходження обличчя виконується кроп і вирівнювання за ключовими точками, щоб зменшити вплив поворотів голови, зсувів камери та зміни масштабу. Окремо передбачена фільтрація неякісних кадрів (розмиття, малий розмір, перекриття), оскільки такі кадри погіршують стабільність ознак і можуть викликати хибні спрацювання. Завершальним кроком є нормалізація під вхід ResNet50 (224 на 224, RGB, нормалізація значень) і формування пакета кадрів у вигляді тензора, який подається до відеомодуля. Алгоритм попередньої обробки аудіо наведено на рисунку 2.4.

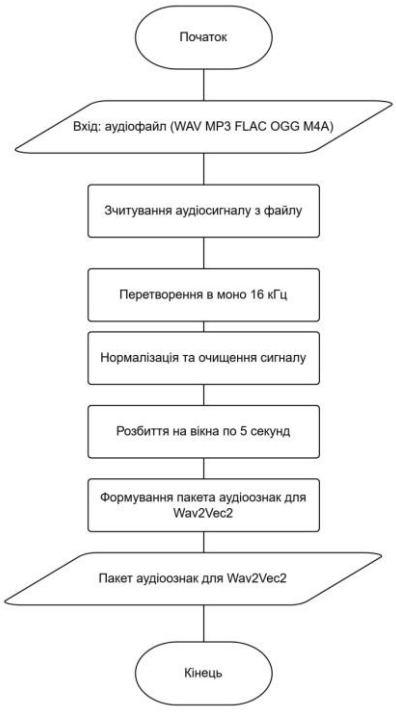


Рисунок 2.4 – Алгоритм попередньої обробки аудіосигналу для формування пакета сегментів і подачі в аудіомодуль Wav2Vec2

Вхідним є аудіофайл поширених форматів, після чого виконується зчитування аудіосигналу. Далі сигнал приводиться до моно 16 кГц. Частота 16 кГц обрана тому, що

цього діапазону достатньо для мовленнєвих ознак, а також це типовий формат, на який орієнтовані багато рішень класу Wav2Vec2 та сумісні з ними конвеєри крім того, таке зниження дискретизації порівняно з 44,1 або 48 кГц суттєво зменшує обчислювальні витрати без втрати ключової інформації для аналізу мовлення. Після ресемплінгу виконується нормалізація та очищення сигналу, щоб вирівняти рівні гучності та зменшити вплив шумових домішок на виділення ознак. Потім аудіо розбивається на вікна фіксованої тривалості по 5 секунд, що забезпечує однаковий контекст для моделі та коректне агрегування результатів на довших записах. На фінальному етапі формується пакет аудіосегментів у форматі, придатному для подачі в аудіомодуль (Wav2Vec2-сумісне представлення), після чого сегменти передаються на класифікацію.

Таким чином, модуль попередньої обробки забезпечує уніфікований вхід для двох модальностей і відокремлює вплив формату та якості даних від роботи детекторів, що напряму підвищує відтворюваність експериментів і коректність подальшого донавчання та інференсу.

2.4 Модуль відеодетекції на базі ResNet із донавчанням

Після етапу попередньої обробки відео формується пакет кадрів обличчя у вигляді тензора, який подається на вхід відеодетектора. Для адаптації моделі під задачу бінарної класифікації real або fake застосовано контрольоване донавчання лише класифікаційного блока ResNet50, тоді як основна частина мережі працює як фіксований екстрактор ознак [20]. Під основною частиною мережі, або backbone, у роботі розуміється набір згорткових шарів ResNet50, що витягають високорівневі ознаки з кадрів і використовують попередньо навчені ваги ImageNet [21]. Під класифікаційним блоком розуміється фінальна частина моделі, яка перетворює витягнуті ознаки на підсумкову ймовірність належності кадру до класу fake, тобто виконує кінцеве рішення для задачі real або fake. Схему донавчання відеодетектора наведено на рисунку 2.5.

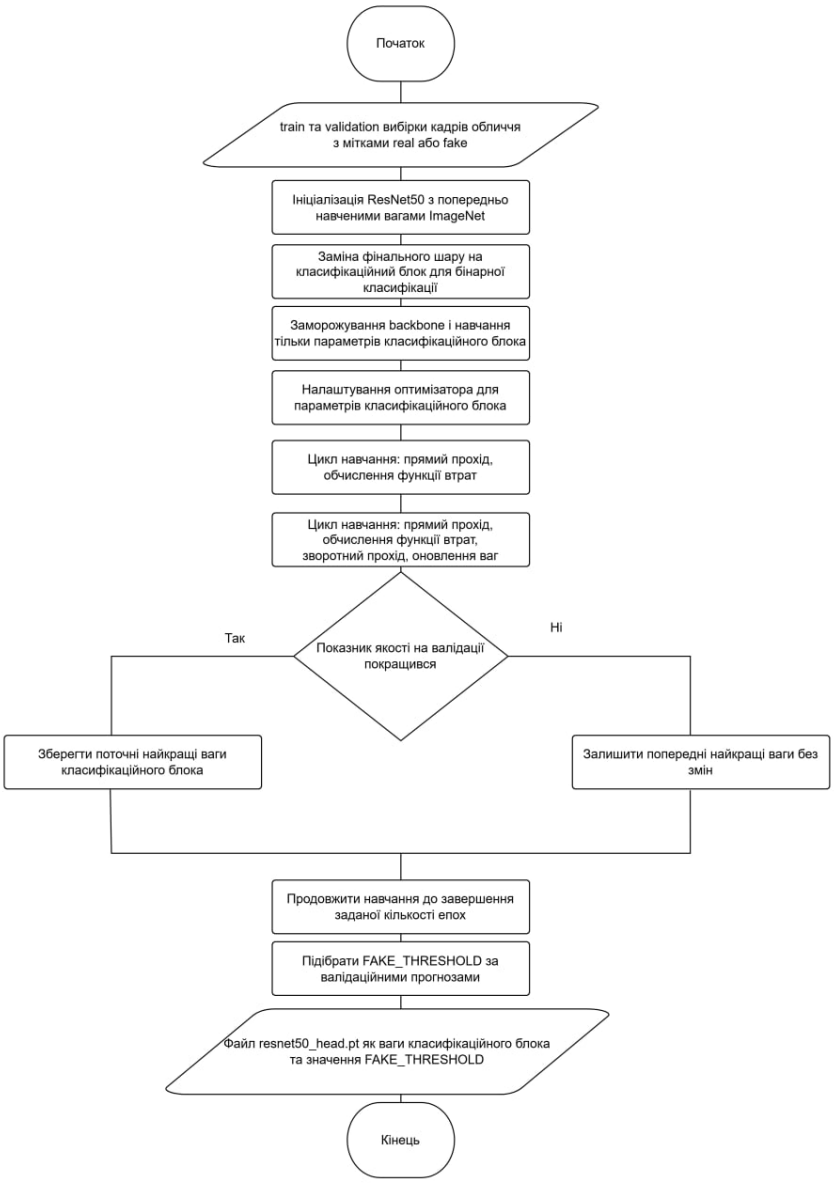


Рисунок 2.5 – Алгоритм донавчання відеодетектора

На вході етапу навчання використовуються навчальна і валідаційна вибірки, що містять тензори кадрів і відповідні мітки класів. Далі ініціалізується ResNet50 з попередньо навченими вагами, після чого стандартний вихідний шар замінюється на новий класифікаційний блок для двох класів [22]. Після заміни вихідного шару параметри backbone фіксуються, а оновлення ваг дозволяється лише для параметрів класифікаційного блока, що зменшує ризик перенавчання на відносно невеликому датасеті та прискорює збіжність навчання. У кожній епосі виконується прямий прохід, обчислення функції втрат, зворотне поширення похибки та оновлення ваг

класифікаційного блока, після чого проводиться оцінка якості на валідації. Якщо показник якості на валідації покращується, зберігаються поточні найкращі ваги класифікаційного блока, інакше зберігаються попередні найкращі ваги без змін. Навчання повторюється до завершення заданої кількості епох. Після завершення донавчання за валідаційними прогнозами підбирається порогове значення FAKE_THRESHOLD, яке надалі використовується під час інференсу для перетворення ймовірності моделі у фінальний вердикт відеомодуля. Чекпойнт, що зберігається у файлі resnet50_head.pt, у межах цієї роботи інтерпретується як ваги класифікаційного блока, а назва файлу використовується як технічне позначення артефакту з навчального скрипта.

Після завершення донавчання відеомодуль переходить у режим застосування і перетворює вхідний відеофайл на підсумковий вердикт про наявність ознак підробки. Алгоритм роботи відеомодуля на основі ResNet50 подано на рисунку 2.6.

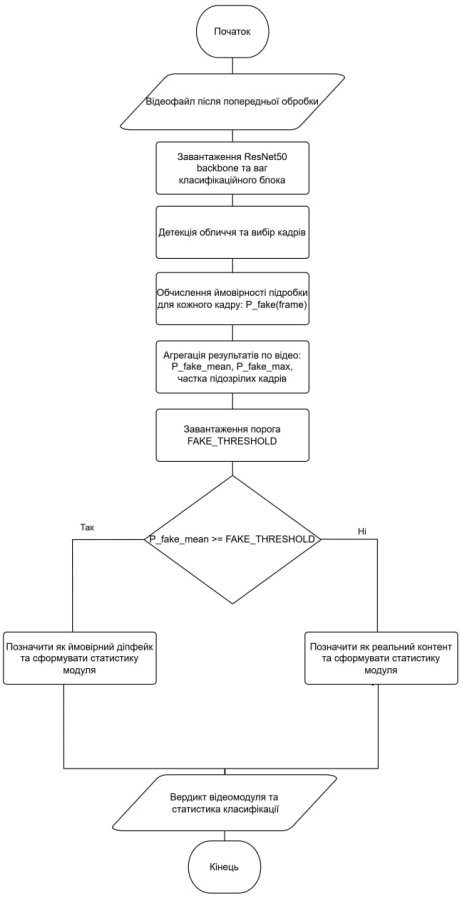


Рисунок 2.6 – Алгоритм роботи відеомодуля ResNet50

На вході модуль отримує відеофайл після попередньої обробки. Далі завантажуються ваги ResNet50, де основна частина мережі використовується як екстрактор ознак. Паралельно завантажуються ваги класифікаційного блока, який виконує фінальне рішення для задачі двокласової класифікації real або fake.

Після цього виконується детекція обличчя та вибір кадрів, що будуть проаналізовані. Кожен вибраний кадр подається в модель, і для нього обчислюється ймовірність підробки у вигляді значення P_fake(frame). Отримані покадрові оцінки узагальнюються на рівні всього відео, для чого обчислюються агреговані показники, зокрема середня ймовірність підробки, максимальна ймовірність підробки та частка підозрілих кадрів.

Далі модуль завантажує порогове значення FAKE_THRESHOLD і порівнює з ним середній показник ймовірності підробки. Якщо середнє значення не менше за поріг, формується гілка Так і відео позначається як ймовірний діпфейк із виведенням статистики класифікації. Якщо середнє значення менше за поріг, формується гілка Ні і контент позначається як реальний, також із відображенням статистики. На виході відеомодуль повертає вердикт і набір підсумкових показників, які використовуються для подальшого формування результату системи.

2.5 Модуль аудіодетекції на базі Wav2Vec із донавчанням

Після попередньої обробки аудіосигналу формується набір уніфікованих аудіосегментів, які використовуються для навчання аудіомодуля. Загальну логіку донавчання аудіомодуля на базі Wav2Vec2 показано на рисунку 2.7.

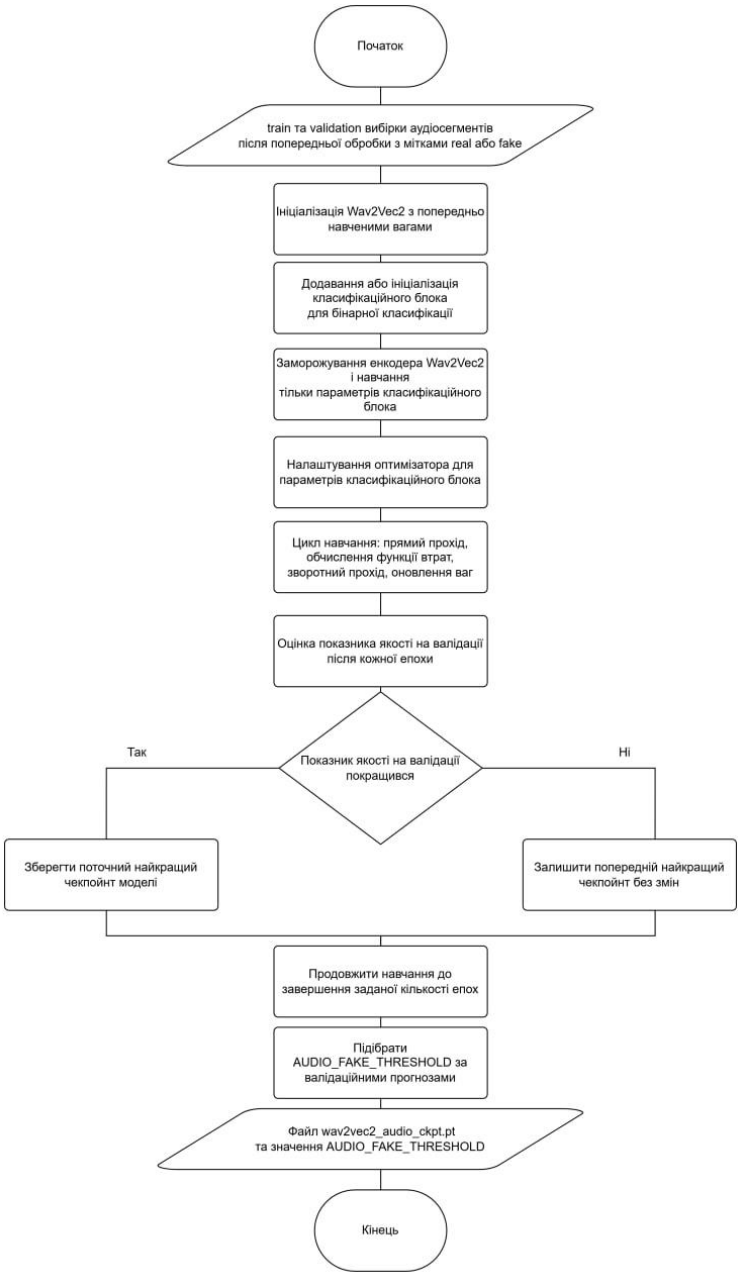


Рисунок 2.7 – Алгоритм донавчання аудіомодуля на базі Wav2Vec

На вході етапу донавчання використовуються train та validation вибірки аудіосегментів із мітками real або fake. Далі ініціалізується модель Wav2Vec2 з попередньо навченими вагами, після чого до неї додається або ініціалізується класифікаційний блок для бінарної класифікації [23]. Під класифікаційним блоком у цій роботі розуміється фінальна частина моделі, яка перетворює ознаки, отримані з Wav2Vec2, у підсумкову оцінку належності сегмента до класу fake або real. Водночас

енкодер Wav2Vec2 виконує роль екстрактора ознак мовлення, тобто формує компактне представлення аудіосегмента на основі акустичної структури сигналу.

Ключовий крок донавчання полягає у заморожуванні енкодера Wav2Vec2 і навчанні лише параметрів класифікаційного блока [24]. Це означає, що під час зворотного проходу оновлюються тільки ваги класифікаційного блока, а основна частина моделі не змінюється. Далі налаштовується оптимізатор саме для параметрів класифікаційного блока, що робить навчання більш керованим і зменшує ризик перенавчання на обмеженому доменному датасеті.

Навчання виконується у вигляді циклу епох. На кожній епосі аудіосегменти з train вибірки подаються на вхід моделі, виконується прямий прохід, обчислюється функція втрат, далі виконується зворотний прохід і оновлення ваг, але лише у класифікаційному блоці. Після завершення кожної епохи проводиться оцінка показника якості на валідації. Далі, відповідно до ромба на схемі, перевіряється умова, чи покращився показник якості на валідації. Якщо так, зберігається поточний найкращий чекпойнт моделі, якщо ні, попередній найкращий чекпойнт залишається без змін. Після цього навчання продовжується до завершення заданої кількості епох.

Окремим завершальним етапом, який також відображено на схемі, є підбір порогового значення AUDIO_FAKE_THRESHOLD за валідаційними прогнозами. Порогове значення потрібне для перетворення безперервної оцінки моделі у дискретний вердикт під час інференсу. У результаті донавчання зберігаються два основні артефакти, файл wav2vec2_audio_ckpt.pt і значення AUDIO_FAKE_THRESHOLD, які надалі використовуються у вебзастосунку для аналізу аудіодоріжки.

Алгоритм роботи аудіомодуля в режимі застосування показано на рисунку 2.8.

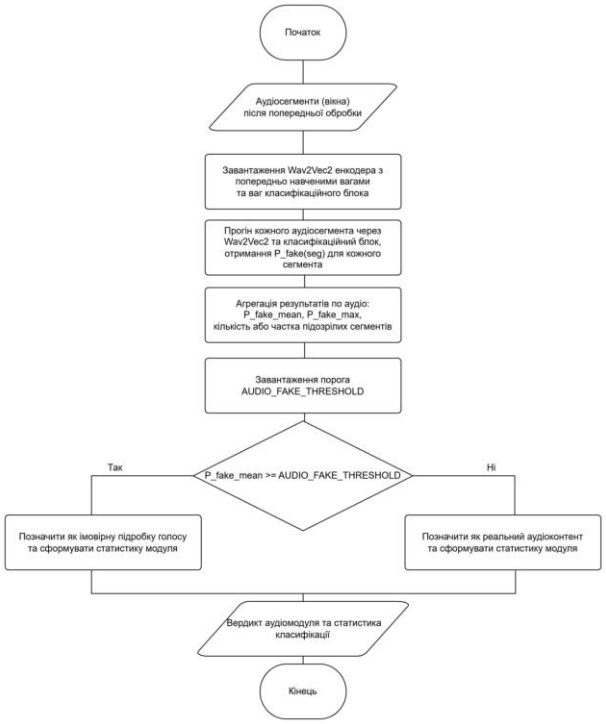


Рисунок 2.8 – Алгоритм роботи аудіомодуля в режимі застосування

Кожен аудіосегмент незалежно проходить через Wav2Vec2 та класифікаційний блок, у результаті для кожного вікна формується значення P_fake(seg). Такий підхід дозволяє отримати не лише одну оцінку для всього запису, а й набір оцінок по сегментах, що дає можливість бачити, чи є в аудіо локальні підозрілі фрагменти.

Після обробки всіх сегментів виконується агрегація результатів по аудіо. У модулі обчислюються узагальнені показники, зокрема середнє значення P_fake_mean та максимальне значення P_fake_max, а також може фіксуватися кількість або частка сегментів із підвищеною підозрілістю. Середнє значення використовується як більш стабільна оцінка для всього запису, тоді як максимум відображає можливі короткі піки ризику.

Далі завантажується порогове значення AUDIO_FAKE_THRESHOLD і в блоці умови перевіряється, чи перевищує P_fake_mean цей поріг. Якщо умова виконується, аудіозапис позначається як імовірна підробка голосу, і формується статистика модуля. Якщо умова не виконується, аудіозапис позначається як реальний аудіоконтент, після чого також формується статистика модуля. На виході повертаються вердикт

аудіомодуля та статистика класифікації, які надалі можуть використовуватися під час об’єднання результатів із відеомодулем.

2.6 Модуль агрегування результатів відео і аудіо

Після виконання відеодетекції та аудіодетекції система переходить до етапу агрегування, мета якого полягає у формуванні єдиного узгодженого вердикту на основі зведених оцінок двох модальностей. На вхід модуля агрегування надходять підсумкові показники відеомодуля Pvideo_mean і Pvideo_max, а також аудіомодуля Paudio_mean і Paudio_max, які відображають середню та максимальну ймовірність наявності ознак підробки в межах проаналізованого файлу. Структура роботи цього етапу показана на рисунку 2.9.

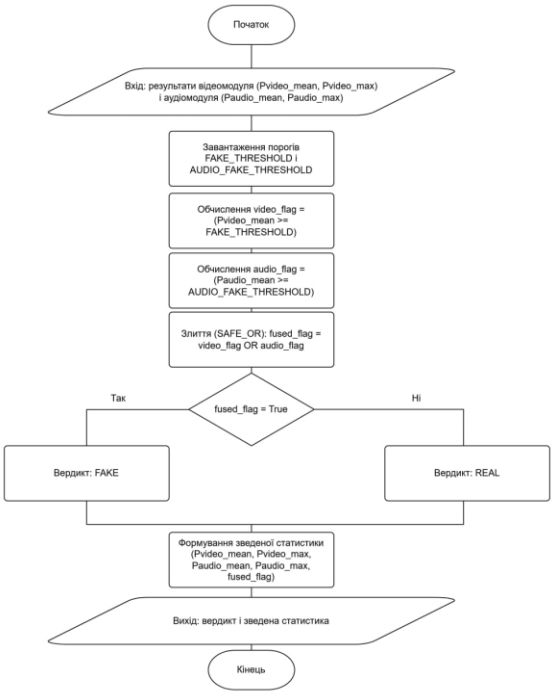


Рисунок 2.9 – Алгоритм роботи модуля агрегування результатів

На першому кроці модуль завантажує два незалежні пороги прийняття рішення, FAKE_THRESHOLD для відео та AUDIO_FAKE_THRESHOLD для аудіо. Розділення порогів є доцільним, оскільки відео- та аудіомоделі мають різні шкали впевненості й

різну чутливість до типових спотворень (компресія, шум, перекодування). Далі для кожної модальності обчислюється бінарний індикатор: video_flag визначається як істина, якщо Pvideo_mean більша або дорівнює FAKE_THRESHOLD, а audio_flag як істина, якщо Paudio_mean більша або дорівнює AUDIO_FAKE_THRESHOLD. Використання саме середнього значення як основи для порівняння з порогом дозволяє зменшити вплив одиничних випадкових сплесків оцінки та перейти до більш стійкого рішення на рівні всього файлу [25].

Після цього виконується злиття результатів за правилом SAFE_OR, де fused_flag обчислюється як логічне АБО між video_flag та audio_flag. Така стратегія означає, що фінальний сигнал ризику активується у випадку, якщо хоча б одна модальність демонструє стабільну ознаку підробки [26]. Практичний сенс SAFE_OR полягає у підвищенні чутливості системи до мультимедійних підробок, де атака може бути помітною лише в одному каналі (наприклад, синтетичний голос за відносно природного відео або навпаки). Водночас факт наявності Pvideo_max та Paudio_max у вихідній статистиці дає можливість додатково інтерпретувати, чи були короткі інтервали з високими значеннями ризику навіть тоді, коли середній рівень нижчий за поріг.

Фінальне рішення формується на основі перевірки fused_flag. Якщо fused_flag дорівнює True, модуль переходить по гілці Так і встановлює вердикт FAKE. Якщо fused_flag дорівнює False, тобто обидві модальності не перевищили свої пороги за середнім значенням, система переходить по гілці Ні та формує вердикт REAL. Таким чином, уся логіка рішення зводиться до двох послідовних порогових перевірок і одного правила злиття, що робить механізм прозорим, відтворюваним і простим для пояснення в межах опису системи.

Окрім самого вердикту, модуль агрегування формує зведену статистику, яка включає Pvideo_mean, Pvideo_max, Paudio_mean, Paudio_max та fused_flag. Також така статистика забезпечує основу для порівняння різних конфігурацій порогів і контролю поведінки системи на різних типах даних без необхідності повторного повного прогону моделей. Виходом роботи модуля агрегування є пара значень, підсумковий вердикт та зведена статистика, яка документує проміжні оцінки й прийняте рішення.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Технологічний стек і середовище виконання

У цьому підрозділі описано програмне та апаратне середовище, у якому реалізовано прототип системи виявлення фейкового мультимедійного контенту, а також основні бібліотеки і фреймворки, що формують технологічний стек. Засіб орієнтовано на локальне виконання без доступу до мережі Інтернет на персональних комп'ютерах під керуванням ОС Windows 10/11, а також Linux-дистрибутивів Ubuntu/Kali в обох випадках використовується однакова програмна конфігурація на базі Python та PyTorch [27],[28],[29],[30].

Такий підхід відповідає вимогам до конфіденційності обробки потенційно чутливих відео, аудіоматеріалів та забезпечує відтворюваність результатів у контрольованому середовищі.

Прототип реалізовано мовою програмування Python версії 3.11, що є де-факто стандартом для швидкої побудови дослідницьких прототипів з опорою на бібліотеки глибинного навчання та обробки сигналів [31]. Керування залежностями здійснюється через віртуальне середовище, всередині якого зафіксовані версії основних пакетів (torch 2.4.1+cpu, torchvision 0.19.1+cpu та супутні модулі) [32],[33],[34]. У коді передбачено автоматичне визначення доступного обчислювального пристрою, за наявності GPU з підтримкою CUDA обчислення можуть переноситися на графічний процесор, проте базовий сценарій магістерської роботи орієнтований на CPU-конфігурацію, достатню для офлайн-експертизи окремих роликів.

Ядром стеку для відеогілки є фреймворк PyTorch у поєднанні з бібліотекою timm, яка надає реалізацію ResNet50 та інструменти для завантаження попередньо навчених ваг [35]. Поверх базового ResNet побудовано та донавчено класифікаційну голову, що зберігається у вигляді файлу чекпоінта resnet50_head.pt і завантажується під час запуску системи. Для підготовки вхідних тензорів використовується модуль torchvision.transforms, кадри нормалізуються до розміру 224×224 пікселі, переводяться

у тензорний формат та масштабується яскравість за стандартними статистиками ImageNet, що забезпечує сумісність із попередньо навченим представленням ResNet [36].

Обробку відео на рівні кадрів реалізовано за допомогою бібліотеки OpenCV (cv2), яка відповідає за декодування контейнерів MP4/AVI/MOV/MKV, доступ до окремих кадрів за індексом і перетворення простору кольорів. Детекцію та початкове нормування обличчя виконує бібліотека facenet-pytorch із використанням детектора MTCNN, що повертає координати прямокутника обличчя та дозволяє виділити найбільше за площею обличчя у кадрі [37]. Далі реалізовано обтинання області обличчя з додатковим відступом (FACE_MARGIN) і масштабування, що забезпечує стабільні вхідні дані для ResNet навіть за наявності незначних зміщень камери або коливань масштабу. Уся логіка обробки організована так, щоб при аналізі відео враховувались лише вибрані кадри згідно з параметром FRAME_SAMPLE_RATE, а за потреби лише перші 10 секунд відео-файла.

Аудіогілка побудована на базі моделі Wav2Vec2.0, реалізованої у фреймворку HuggingFace Transformers [38], [39]. Для неї завантажується збережений стан класифікаційної моделі з файлу wav2vec2_audio_ckpt.pt, що містить параметри донавченої голови для задачі бінарної класифікації (real/fake). Попередню обробку звуку здійснює бібліотека librosa, вхідні WAV/MP3/FLAC/OGG/M4A-файли, а також аудіодоріжки, витягнуті з відео, ресемплюються до моноформату з частотою 16 кГц, нормалізується рівень гучності та формується масив відліків. Сигнал ділиться на послідовність фіксованих вікон тривалістю AUDIO_SEGMENT_SECONDS (типово 5 секунд), для кожного з яких обчислюється окрема оцінка ймовірності підробки, а потім усереднюється та агрегується у підсумковий показник [40].

Зв’язок між відео та аудіо забезпечується за рахунок бібліотеки MoviePy, яка використовується для витягування аудіодоріжки з відеофайлів у формат WAV із заданою частотою дискретизації [41]. Це дозволяє у мультимодальному режимі «Відео + Аудіо» виконувати послідовний аналіз, спочатку відеомодулем, потім аудіомодулем, після чого об’єднувати результати за правилом SAFE_OR. Робота з файловою системою

організована через модуль pathlib, усі тимчасові файли розміщуються в робочому каталозі застосунку, після завершення аналізу вони видаляються, що запобігає накопиченню проміжних даних і спрощує супровід.

Інтерфейс користувача реалізовано на базі фреймворку Streamlit, який надає легку у розгортанні веб-оболонку для локальних аналітичних інструментів [42]. Головна сторінка містить заголовок системи, короткий опис зі значеннями основних параметрів (тип пристрою, використовуваний backbone, поточні пороги FAKE_THRESHOLD та AUDIO_STRICT_THRESHOLD). Режим роботи обирається через горизонтальний перемикач st.radio з трьома вкладками «Відео», «Аудіо», «Відео + Аудіо», кожна з яких має власний блок завантаження файлів, додаткові параметри (наприклад, прапорець обрізання до перших 10 секунд) і зону виведення результатів. Для зручності експерта оригінальне відео та аудіо ховаються в елементі st.expander, що дозволяє за потреби програти матеріал без перевантаження основного екрану статистикою.

Отримані результати аналізу організуються у вигляді структурованих словників Python, які за потреби можуть бути у форматі JSON для подальшого зберігання або інтеграції з зовнішніми системами [43]. У кожному режимі зберігаються числові показники (кількість проаналізованих кадрів або сегментів, середня та максимальна ймовірність фейку, частка «високоризикових» кадрів тощо), а також проміжні артефакти, наприклад, кадр з виділеним обличчям. Така організація стеку дозволяє у наступних підрозділах безпосередньо описати підготовку даних, реалізацію окремих модулів і процедури експериментальної перевірки, не повертаючись до питань сумісності та розгортання середовища виконання.

3.2 Підготовка даних та структура датасету для донавчання детекторів

Для навчання та валідації мультимодальної системи було сформовано власний датасет, який містить як відеозаписи з обличчями, так і аудіодоріжки з мовленням. Підготовка даних включала етапи очищення, нормалізації формату та розподілу на

підмножини train для навчання та validation для кросвалідації окремо для відео та аудіо. Необхідність донавчання (fine-tuning) зумовлена тим, що базові попередньо натреновані моделі не враховують специфіку цільових даних проєкту, реальні умови зйомки, характерні артефакти компресії й монтажу, а також мовні та акустичні особливості українського мовлення і типові сценарії генерації TTS/voice-cloning. Тому донавчання на власних прикладах дозволяє адаптувати модель до домену застосування, підвищити стійкість до різних джерел контенту та забезпечити коректний вибір порогів рішення на валідаційній вибірці. Дані збиралися з відкритих джерел (соціальні мережі, відеохостинги, новинні ресурси) з урахуванням збалансованості класів real / fake.

Відеодатасет складається з коротких роликів тривалістю переважно від 3 до 10 секунд. Для забезпечення збалансованості навчання було відібрано 199 відео з реальними обличчями та 199 відео з діпфейками у навчальній підмножині. Для валідації виділено ще 40 реальних і 40 фейкових відео файлів. Таким чином, загальний обсяг відеодатасету становить 478 відеофрагментів загальним обсягом 790 МБ, порівну розподілених між класами «real» та «fake». Загальну структуру каталогів відеодатасету показано на рисунку 3.1.

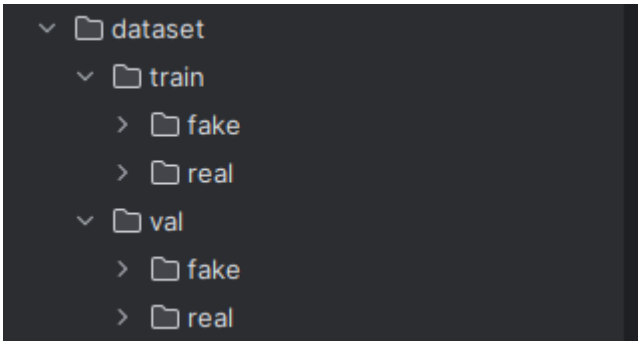


Рисунок 3.1 – Структура каталогів відеодатасету

Приклад вмісту директорії з фейковими відео навчальної підмножини (dataset/train/fake) наведено на рисунку 3.2.

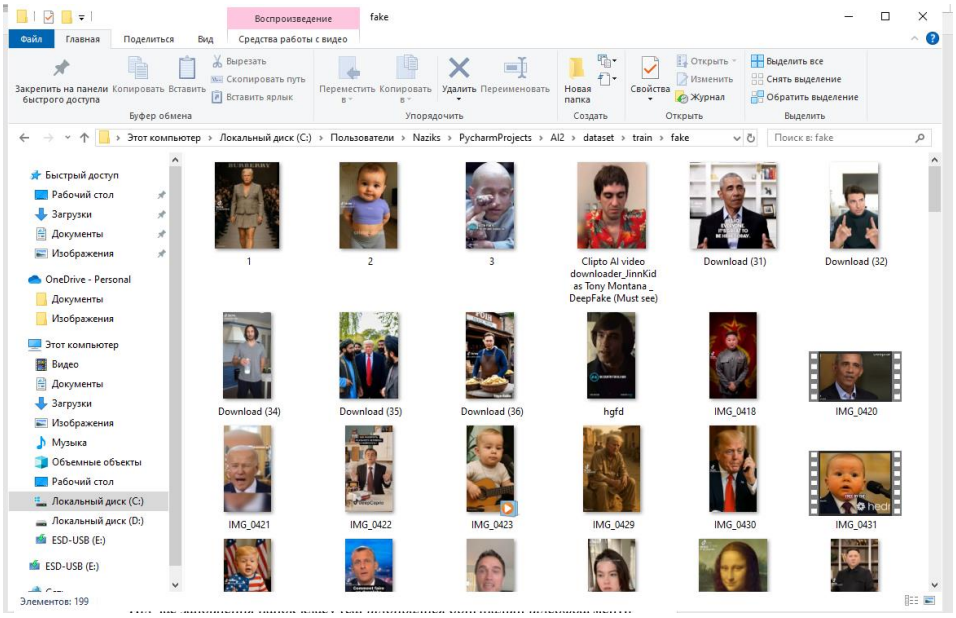


Рисунок 3.2 – Вміст директорії «dataset/train/fake»

Представлені різноманітні відео з політичними виступами, інтерв'ю, кліпами та побутовими сценами, згенерованими або зміненими засобами діпфейк-технологій. На рисунку 3.3 подано аналогічний скріншот для папки з реальними відео (dataset/train/real), що містить оригінальні записи без штучних модифікацій.

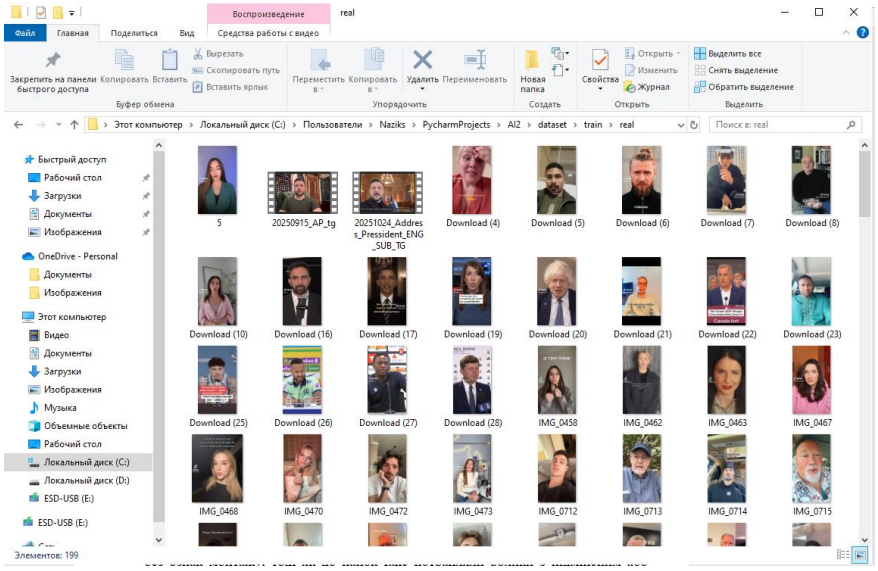


Рисунок 3.3 – Вміст директорії dataset/train/real

Окремо на рисунку 3.4 та на рисунку 3.5 ілюструється вміст валідаційних директорій (dataset /val/real та dataset /val/fake).

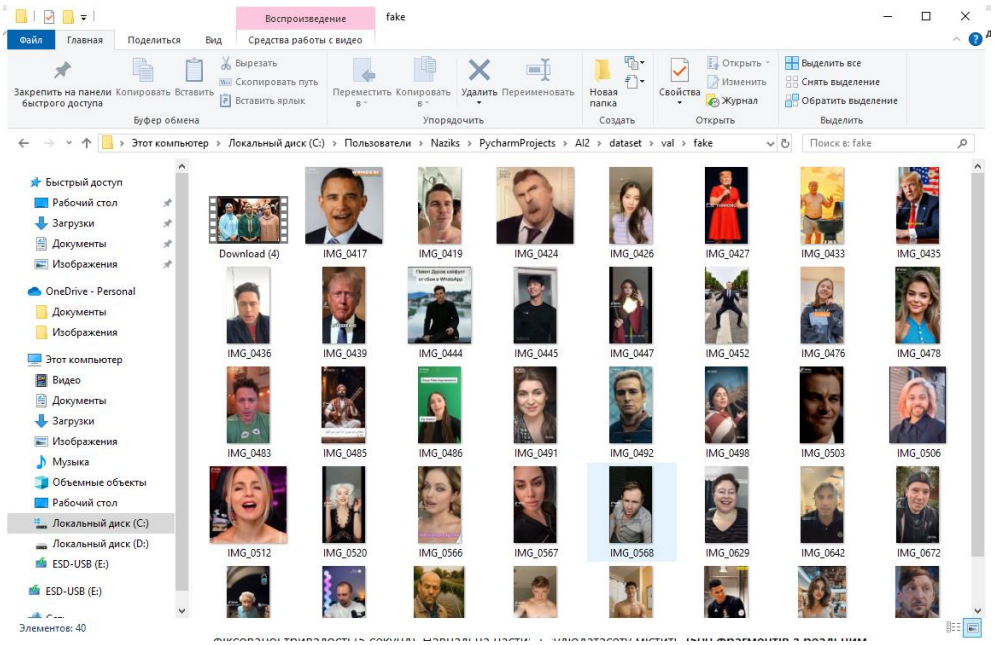


Рисунок 3.4 – Вміст директорії dataset/train/fake

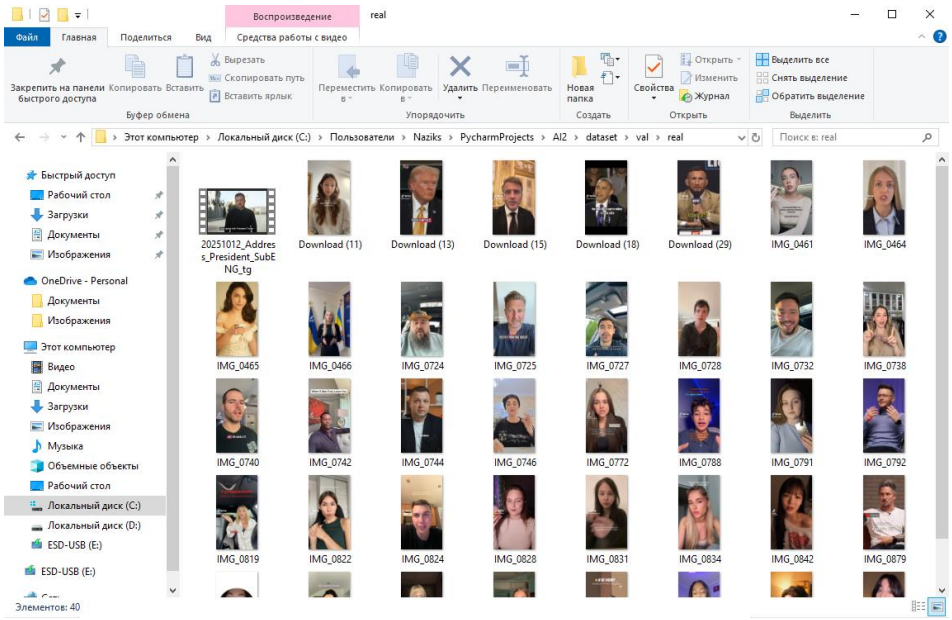


Рисунок 3.5 – Вміст директорії dataset/train/real

Аудіодатасет формувався як окрема підмножина, концептуально синхронна із відеоданими за задумом експерименту, але незалежна від них за способом підготовки. Усі аудіозаписи попередньо приведено до єдиного формату моно 16 кГц та розбито на фрагменти фіксованої тривалості 5 секунд. Синтетичні приклади створювалися двома джерелами генерації, (1) безкоштовним TTS-сервісом [44], де використано 6 різних

голосів, та (2) платним AI-сервісом [45], де використано 4 різних голоси, зокрема стилізації під Володимира Зеленського, Дональда Трампа, Камаллу Гарріс і королеву Великої Британії. Навчальна частина аудіодатасету містить 1500 фрагментів із реальним голосом та 1500 фрагментів із синтетичним мовленням (TTS/voice-cloning). Для валідації додатково сформовано 400 реальних і 400 фейкових аудіофрагментів. Загалом аудіодатасет включає 3800 сегментів. Загальний розмір катагологу «audio_dataset» 974 МБ. Загальну структуру каталогів аудіодатасету показано на рисунку 3.6.

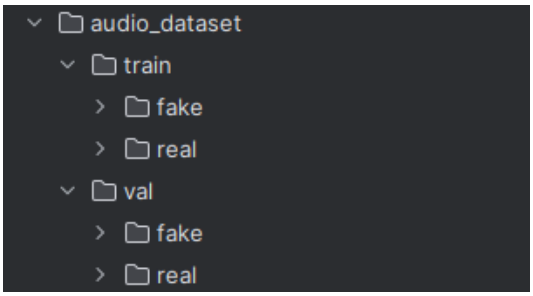


Рисунок 3.6 – Структура директорій аудіодатасету

На рисунку 3.7 наведено приклад вмісту папки з фейковими навчальними аудіозаписами (audio_dataset/train/fake) кожен файл відповідає окремому п’ятисекундному фрагменту синтетичної мови.

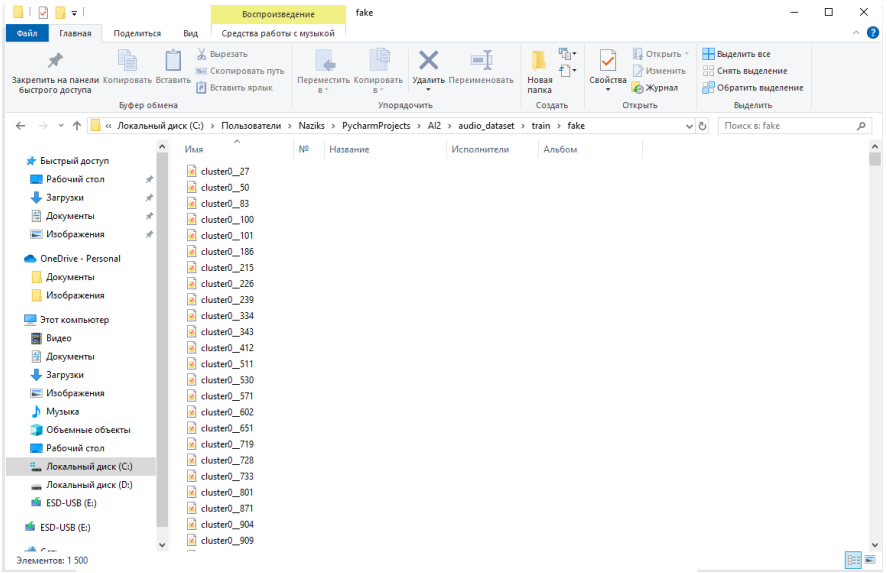


Рисунок 3.7 – Вміст директорії audio_dataset/train/fake

На рисунку 3.8 показано аналогічний скріншот для реальних навчальних записів (audio_dataset/train/real).

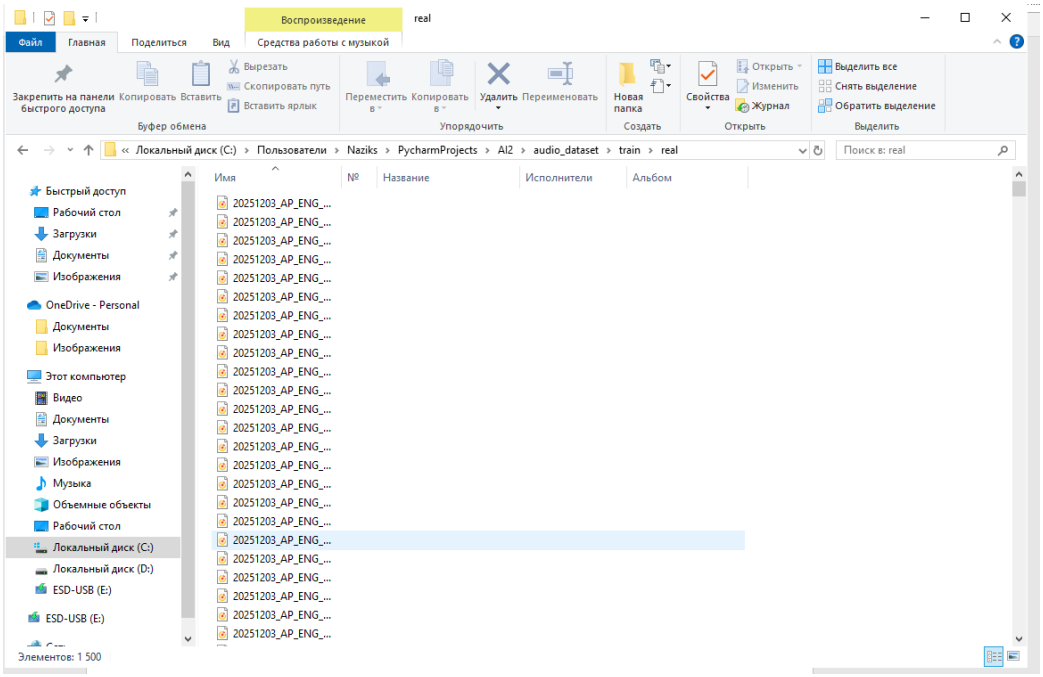


Рисунок 3.8 – Вміст директорії audio_dataset/train/real

Валідаційні аудіодані проілюстровані на рисунку 3.9 та на рисунку 3.10. Як і у випадку з відео, ці файли не використовуються при оновленні ваг моделі, вони служать «еталоном» для проміжної перевірки якості аудіомодуля на ще не бачених під час навчання прикладах та для підбору порога AUDIO_STRICT_THRESHOLD.

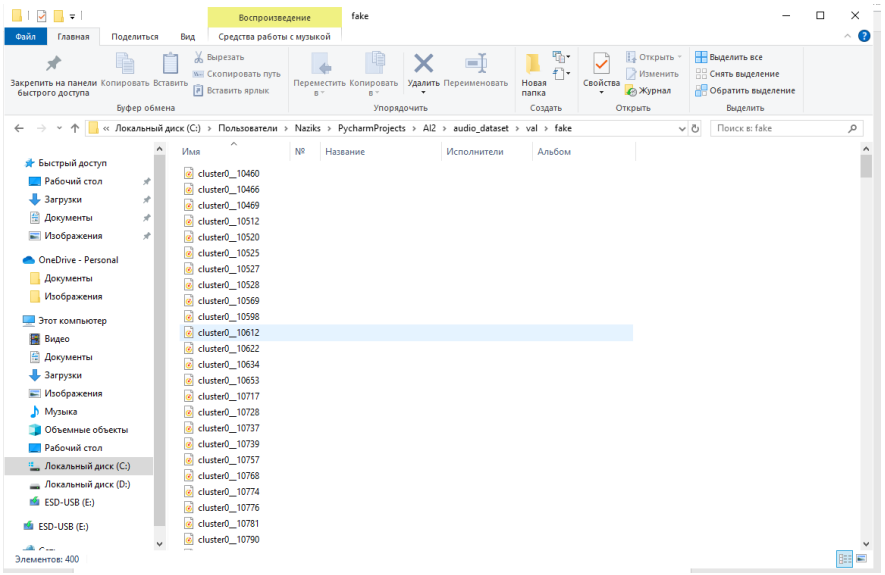


Рисунок 3.9 – Вміст директорії audio_dataset/val/fake

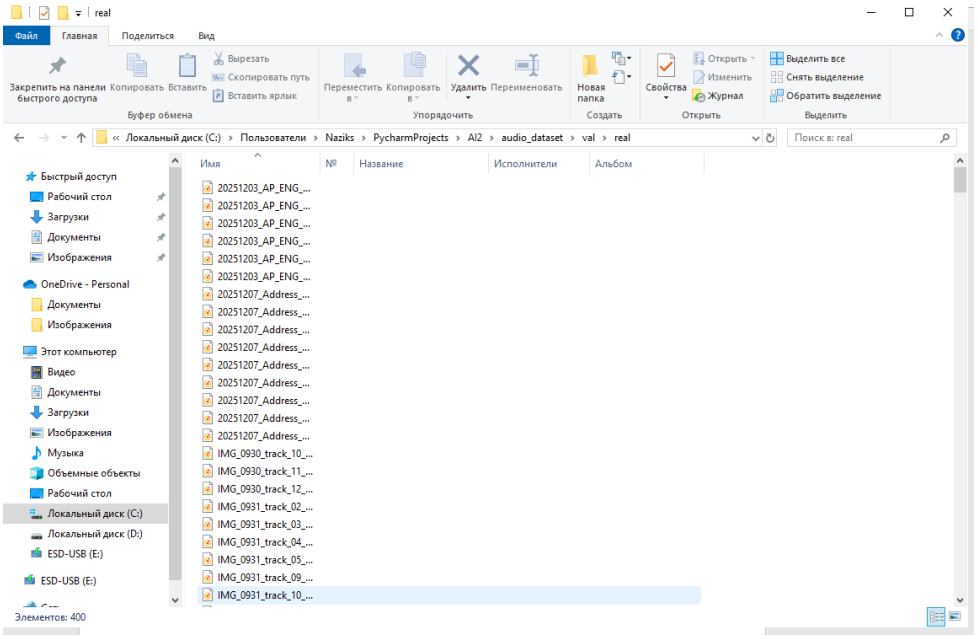


Рисунок 3.10 – Вміст директорії audio_dataset/val/real

Окремо сформовано директорію test, яка використовується для експериментальної перевірки роботи застосунку в усіх режимах. Директорія test зображена на рисунку 3.11.

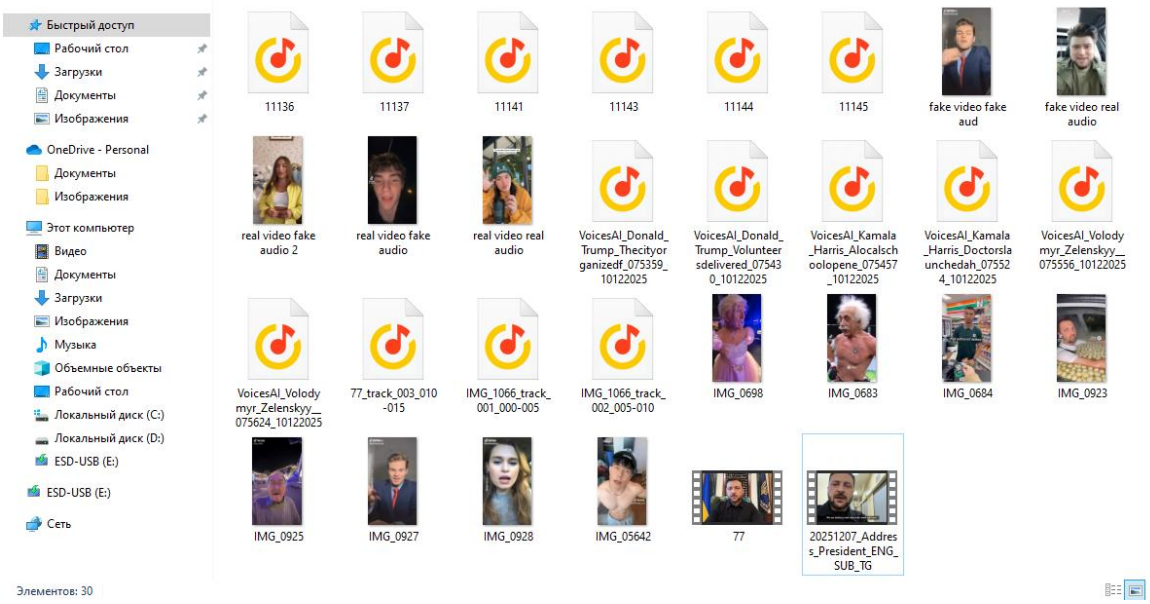


Рисунок 3.11 – Вміст директорії test

До неї включено: діпфейк-відео (fake video), реальні відео (real video), реальні аудіозаписи (real audio), синтетичні аудіозаписи (fake audio), а також змішані приклади

для перевірки вкладки «Відео + Аудіо» (комбінації fake video + fake audio, fake video + real audio, real video + fake audio, real video + real audio). Такий підхід дозволяє окремо тестувати кожну модальність і додатково перевіряти коректність інтегрованого рішення у мультимодальному режимі.

Загальний розподіл даних між підмножинами для відеодатасету становить, train — 78,35 % (398 прикладів), validation — 15,75 % (80 прикладів), test — 5,91 % (30 прикладів). Для аудіодатасету розподіл становить, train — 78,33 % (3000 сегментів), validation — 20,89 % (800 сегментів), test — 0,78 % (30 прикладів). Це забезпечує відокремлення навчання та підбору порогів на train і validation від фінального тестування на незалежних прикладах test.

Таким чином, обидві модальності — відео та аудіо — підготовлено у вигляді збалансованих і чітко структурованих наборів real/fake з розділенням на тренувальну, валідаційну та тестову вибірки, що дозволяє коректно навчати окремі модулі, порівнювати їхню якість, налаштовувати пороги рішення та інтегрувати результати в мультимодальну схему прийняття рішень, описану в наступних підрозділах.

3.3 Реалізація модуля відеодетекції

У цьому підрозділі розглянуто реалізацію відеомодуля, який виконує два основних завдання, донавчання класифікатора на базі ResNet50 за підготовленим відеодатасетом та подальший інференс моделі у веб-застосунку для аналізу окремих відеофайлів. Навчальна логіка зосереджена у скрипті train_resnet_head.py, тоді як безпосередній аналіз відео інтегровано в застосунок Streamlit у файлі deepfake_app_av.py.

Для формування навчальних прикладів використано клас FaceFrameDataset, що наслідує torch.utils.data.Dataset. Конструктор класу одразу отримує кореневу папку датасету, тип вибірки (train або val), кількість кадрів з кожного відео, розмір зображення та ширину відступу навколо обличчя. Усередині створюється екземпляр детектора MTCNN, збирається список відеофайлів із підпапок real і fake, а також налаштовуються

перетворення зображень, для train-вибірки використовується аугментація (горизонтальне віддзеркалення, легкі зміни яскравості й контрасту), для val-вибірки лише масштабування й нормалізація. Фрагмент коду конструктора наведено на рисунку 3.12.

```
class FaceFrameDataset(Dataset):  3 usages

    def __init__(self, root: str, split: str, frames_per_video: int, img_size: int, face_margin: int):
        assert split in ["train", "val"]
        self.root = root
        self.split = split
        self.frames_per_video = frames_per_video
        self.img_size = img_size
        self.face_margin = face_margin

        self.mtcnn = MTCNN(keep_all=False, device="cuda" if torch.cuda.is_available() else "cpu")
        self.items: List[Tuple[str, int]] = []
        mapping = {"real": 0, "fake": 1}
        for cls_name, cls_idx in mapping.items():
            d = Path(root) / split / cls_name
            if not d.exists():
                continue
            for ext in ("*.mp4", "*.avi", "*.mov", "*.mkv"):
                for p in d.glob(ext):
                    self.items.append((str(p), cls_idx))

        if split == "train":
            self.transform = T.Compose([
                T.Resize((img_size, img_size)),
                T.RandomHorizontalFlip(p=0.5),
                T.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.05, hue=0.02),
                T.ToTensor(),
                T.Normalize(mean=[0.485, 0.456, 0.406],
                            std=[0.229, 0.224, 0.225]),
            ])
        else:
            self.transform = T.Compose([
                T.Resize((img_size, img_size)),
                T.ToTensor(),
                T.Normalize(mean=[0.485, 0.456, 0.406],
                            std=[0.229, 0.224, 0.225]),
            ])
```

Рисунок 3.12 – Фрагмент коду конструктора класу FaceFrameDataset для формування кадрів обличчя з відео

Метод __len__ у цьому класі повертає загальну кількість тренувальних прикладів як добуток числа відео на кількість кадрів, що вибираються з кожного ролика. Основна логіка винесена в метод __getitem__, за індексом визначається, до якого відео належить кадр, відкривається відповідний файл, обчислюється загальна кількість кадрів, вибирається підмножина індексів, після чого читається потрібний кадр. Якщо відео не вдається прочитати, використовується заглушка-картинка нульової яскравості. Для коректних кадрів виконується детекція обличчя, обрізання області з відступом, конвертація в RGB та застосування підготовлених перетворень. На вихід повертається тензор кадру, мітка класу та ім’я відеофайла. Відповідний фрагмент показано на рисунку 3.13.

```
def __len__(self):
    return len(self.items) * self.frames_per_video

def __getitem__(self, idx):
    vid_id = idx // self.frames_per_video
    local_idx = idx % self.frames_per_video
    video_path, label = self.items[vid_id]

    cap = cv2.VideoCapture(video_path)
    total = read_total_frames(cap)
    if total == 0:
        cap.release()
        img = Image.fromarray(np.zeros( shape: (self.img_size, self.img_size, 3), dtype=np.uint8))
        return self.transform(img), label, Path(video_path).name

    idxs = sample_frame_indices(total, self.frames_per_video)
    frame_idx = idxs[min(local_idx, len(idxs) - 1)]
    cap.set(cv2.CAP_PROP_POS_FRAMES, frame_idx)
    ok, frame = cap.read()
    cap.release()

    if not ok or frame is None:
        img = Image.fromarray(np.zeros( shape: (self.img_size, self.img_size, 3), dtype=np.uint8))
        return self.transform(img), label, Path(video_path).name

    face_bgr = crop_face_bgr(frame, self.mtcnn, self.face_margin)
    face_rgb = cv2.cvtColor(face_bgr, cv2.COLOR_BGR2RGB)
    pil = Image.fromarray(face_rgb)
    x = self.transform(pil)
    return x, label, Path(video_path).name
```

Рисунок 3.13 – Фрагмент коду завантаження кадру, виділення обличчя та формування тензора у методі __getitem__ класу FaceFrameDataset

Архітектура класифікатора побудована на базі попередньо навченої мережі ResNet50. Функція build_resnet50_head створює модель із двома вихідними класами, завантажує ImageNet-ваги та за замовчуванням «заморожує» всі параметри, окрім фінального повнозв’язного шару fc. Таким чином на першому етапі донавчається лише «голова» мережі, що пришвидшує збіжність і зменшує ризик перенавчання на невеликому датасеті. Фрагмент реалізації показано на рисунку 3.14.

```
def build_resnet50_head(num_classes=2, freeze_backbone=True):  1 usage
    model = timm.create_model( model_name: "resnet50", pretrained=True, num_classes=num_classes)
    if freeze_backbone:
        for n, p in model.named_parameters():
            if "fc" not in n:
                p.requires_grad = False
    return model
```

Рисунок 3.14 – Фрагмент коду функції build_resnet50_head, що створює модель ResNet50 з донавчуваною «головою»

Навчання мережі організовано в окремій функції train_one_epoch, яка виконує одну епоху. Для кожної міні-партії виконується прямий прохід, обчислення крос-ентропійної функції втрат, зворотне поширення помилки та оновлення параметрів оптимізатором AdamW. Сумарна втрата акумулюється з урахуванням розміру батчу, а

в кінці повертається середнє значення втрат по всіх прикладах. Код наведено на рисунку 3.15.

```python
def train_one_epoch(model, loader, optim, device):  1 usage
    model.train()
    loss_meter, n = 0.0, 0
    for x, y, _ in tqdm(loader, desc="train", leave=False):
        x = x.to(device)
        y = y.to(device)
        logits = model(x)
        loss = F.cross_entropy(logits, y)
        optim.zero_grad()
        loss.backward()
        optim.step()
        loss_meter += float(loss.item()) * x.size(0)
        n += x.size(0)
    return loss_meter / max(n, 1)
```

Рисунок 3.15 – Фрагмент коду функції train_one_epoch, що реалізує одну епоху донавчання ResNet50-класифікатора

Оцінювання якості проводиться у функції validate, яка проходить по валідаційному завантажувачу, обчислює ймовірності класу «fake» для кожного кадру, переводить їх у бінарні передбачення і підраховує метрики точності та F1-міри як на рівні кадрів, так і на рівні відео (за усередненими ймовірностями по роликах). Додаткова функція evaluate_on_loader використовується для швидкої перевірки на train-вибірці, а print_split_stats друкує статистику по класах. Фрагмент коду з обчисленням метрик подано на рисунку 3.16.

```python
@torch.no_grad()  1 usage
def validate(model, loader, device) -> Dict[str, float]:
    model.eval()
    all_pred, all_true = [], []
    per_video_probs: Dict[str, List[float]] = {}
    per_video_labels: Dict[str, int] = {}

    for x, y, vidname in tqdm(loader, desc="val", leave=False):
        x = x.to(device)
        probs = torch.softmax(model(x), dim=1)[:, 1].detach().cpu().numpy()  # P(fake)
        pred = (probs >= 0.5).astype(np.int64)

        all_pred.extend(pred.tolist())
        all_true.extend(y.numpy().tolist())

        for i in range(len(vidname)):
            vn = vidname[i]
            per_video_probs.setdefault(vn, []).append(float(probs[i]))
            per_video_labels.setdefault(vn, int(y.numpy()[i]))

    acc = accuracy_score(all_true, all_pred) if len(all_true) else 0.0
    f1  = f1_score(all_true, all_pred) if len(set(all_true)) > 1 else 0.0

    v_preds, v_true = [], []
    for vn, ps in per_video_probs.items():
        v_preds.append(1 if (sum(ps)/len(ps)) >= 0.5 else 0)
        v_true.append(per_video_labels[vn])
    v_acc = accuracy_score(v_true, v_preds) if v_true else 0.0
    v_f1  = f1_score(v_true, v_preds) if len(set(v_true)) > 1 else 0.0

    return {"frame_acc": acc, "frame_f1": f1, "video_acc": v_acc, "video_f1": v_f1}
```

Рисунок 3.16 – Фрагмент коду обчислень кадрових і відеорівневих метрик точності та F1-міри у функції validate

Основний цикл навчання, реалізований у функції main, створює екземпляри FaceFrameDataset для train і val-підвибірок, обгортає їх у DataLoader, ініціалізує модель та оптимізатор, а далі у циклі по епохах викликає train_one_epoch та validate. Додатково зберігається найкраща модель за відео-F1 та за потреби розморожуються верхні шари ResNet для тонкого донавчання. Узагальнений фрагмент цього циклу наведено на рисунку 3.17.

```
train_ds = FaceFrameDataset(args.data_root, split="train", args.frames_per_video, args.img_size, args.face_margin)
val_ds   = FaceFrameDataset(args.data_root, split="val",   args.frames_per_video, args.img_size, args.face_margin)
if len(train_ds) == 0 or len(val_ds) == 0:
    print("Порожні train/val. Перевір структуру і файли.")
    return

print_split_stats(train_ds, name="TRAIN")
print_split_stats(val_ds,   name="VAL")

train_loader = DataLoader(train_ds, batch_size=args.batch, shuffle=True,
                          num_workers=args.workers, pin_memory=True)
val_loader   = DataLoader(val_ds,   batch_size=args.batch, shuffle=False,
                          num_workers=args.workers, pin_memory=True)


model = build_resnet50_head(num_classes=2, freeze_backbone=True).to(device)

# Оптимізатор для голови
head_params = [p for n, p in model.named_parameters() if p.requires_grad]
optim = torch.optim.AdamW(head_params, lr=args.lr, weight_decay=args.wd)

best_vf1 = -1.0
for ep in range(1, args.epochs + 1):
    t0 = time.time()

    tr_loss = train_one_epoch(model, train_loader, optim, device)
    tr_m = evaluate_on_loader(model, train_loader, device)  # sanity-check на train
    mets = validate(model, val_loader, device)

    dt = time.time() - t0
    print(
        f"Epoch {ep}/{args.epochs} | loss={tr_loss:.4f} | "
        f"train_acc={tr_m['acc']:.3f} train_f1={tr_m['f1']:.3f} | "
        f"val_frame_acc={mets['frame_acc']:.3f} val_frame_f1={mets['frame_f1']:.3f} | "
        f"val_video_acc={mets['video_acc']:.3f} val_video_f1={mets['video_f1']:.3f} | {dt:.1f}s"
```

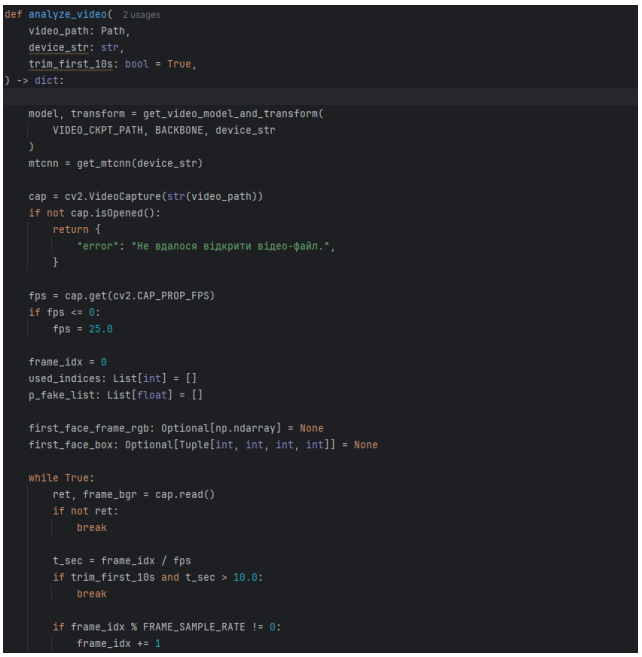
Рисунок 3.17 – Фрагмент коду основного циклу донавчання ResNet50-моделі та збереження найкращого чекпоінта

Після завершення навчання отриманий чекпоінт resnet50_head.pt використовується у веб-застосунку для аналізу довільних відеофайлів. Функція get_video_model_and_transform у файлі deepfake_app_av.py завантажує модель ResNet з тим самим backbone, підтягує збережені ваги класифікатора й формує ланцюжок перетворень (масштабування, перетворення в тензор, нормалізація) для інференсу. Фрагмент функції наведено на рисунку 3.18.

```python
def get_video_model_and_transform(
    ckpt_path: Path,
    backbone: str,
    device_str: str,
):

    dev = torch.device(device_str)

    model = timm.create_model(backbone, pretrained=True, num_classes=2)

    state = torch.load(str(ckpt_path), map_location=dev)
    head_state = state.get("state_dict", state)

    try:
        classifier = model.get_classifier()
        classifier.load_state_dict(head_state, strict=True)
        model.reset_classifier(num_classes=2)
    except Exception:
        model.load_state_dict(head_state, strict=False)

    model.to(dev)
    model.eval()

    transform = T.Compose(
        [
            T.Resize((224, 224)),
            T.ToTensor(),
            T.Normalize(
                mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225],
            ),
        ]
    )
    return model, transform
```

Рисунок 3.18 – Фрагмент коду функції get_video_model_and_transform, що завантажує навчений відеодетектор і перетворення

Безпосередній аналіз окремого відеофайла виконується функцією analyze_video. На вхід вона отримує шлях до відео та тип пристрою (cpu/cuda), завантажує модель і MTCNN, відкриває контейнер через cv2.VideoCapture та послідовно перебирає кадри. Кадри вибираються з кроком FRAME_SAMPLE_RATE, за потреби аналіз обмежується першими 10 секундами ролика. Для кожного відібраного кадру виконується детекція найбільшого обличчя, обрізання з відступом, формування тензора і передавання його до моделі; далі зберігається ймовірність класу «fake» для поточного кадру. Початок циклу обробки показано на рисунку 3.19.



```python
def analyze_video(  2 usages
    video_path: Path,
    device_str: str,
    trim_first_10s: bool = True,
) -> dict:

    model, transform = get_video_model_and_transform(
        VIDEO_CKPT_PATH, BACKBONE, device_str
    )
    mtcnn = get_mtcnn(device_str)

    cap = cv2.VideoCapture(str(video_path))
    if not cap.isOpened():
        return {
            "error": "Не вдалося відкрити відео-файл.",
        }

    fps = cap.get(cv2.CAP_PROP_FPS)
    if fps <= 0:
        fps = 25.0

    frame_idx = 0
    used_indices: List[int] = []
    p_fake_list: List[float] = []

    first_face_frame_rgb: Optional[np.ndarray] = None
    first_face_box: Optional[Tuple[int, int, int, int]] = None

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

        t_sec = frame_idx / fps
        if trim_first_10s and t_sec > 10.0:
            break

        if frame_idx % FRAME_SAMPLE_RATE != 0:
            frame_idx += 1
```

Рисунок 3.19 – Фрагмент коду ініціалізації аналізу відео та вибір кадрів для подальшої обробки у функції analyze_video

Після вибору кадру виконується детекція обличчя за допомогою MTCNN, розширення прямокутника на FACE_MARGIN пікселів, обрізання і формування вхідного тензора. Модель повертає логіти, які перетворюються в ймовірності за допомогою softmax; ймовірність класу «fake» накопичується у списку, а індекси використаних кадрів записуються окремо. Перший придатний кадр додатково зберігається як попередній перегляд для візуалізації. Заключна частина циклу і обчислення агрегованих показників наведені на рисунку 3.20.

```python
    if boxes is None or len(boxes) == 0:
        frame_idx += 1
        continue

    if isinstance(boxes, np.ndarray):
        if boxes.ndim == 2 and boxes.shape[0] > 1:
            areas = (boxes[:, 2] - boxes[:, 0]) * (boxes[:, 3] - boxes[:, 1])
            box = boxes[areas.argmax()]
        else:
            box = boxes[0]
    else:
        box = boxes[0]

    x1, y1, x2, y2 = box
    h, w, _ = frame_rgb.shape

    x1 = int(max(0, x1 - FACE_MARGIN))
    y1 = int(max(0, y1 - FACE_MARGIN))
    x2 = int(min(w, x2 + FACE_MARGIN))
    y2 = int(min(h, y2 + FACE_MARGIN))

    if x2 <= x1 or y2 <= y1:
        frame_idx += 1
        continue

    face_crop = frame_rgb[y1:y2, x1:x2, :]
    if face_crop.size == 0:
        frame_idx += 1
        continue

    if first_face_frame_rgb is None:
        first_face_frame_rgb = frame_rgb.copy()
        first_face_box = (x1, y1, x2, y2)
```

Рисунок 3.20 – Фрагмент коду обробки облич у кадрі, накопичення ймовірностей та формування узагальнених показників у функції analyze_video

У результаті функція analyze_video повертає набір діагностичних характеристик: кількість проаналізованих кадрів, середню й максимальну ймовірності фейку, частку кадрів із високою ймовірністю (frac_high), список окремих ймовірностей та прев'ю-кадр із рамкою навколо обличчя. Бінарний вердикт is_fake визначається порівнянням середньої ймовірності з поточним порогом FAKE_THRESHOLD. Детальний підбір значення цього порога та його роль у мультимодальному відео + аудіо буде розглянуто в наступному підрозділі, присвяченому реалізації модуля ф'южну та порогів рішень.

3.4 Реалізація модуля аудіодетекції

Аудіомодуль системи призначений для визначення ймовірності синтетичності голосу у вхідному аудіосигналі або в аудіодоріжці, витягнутій із відео. Реалізація включає дві частини, навчання класифікатора на базі Wav2Vec2.0 у скрипті train_audio_wav2vec2.py та інференс у застосунку Streamlit у файлі deepfake_app_av.py. Під час інференсу аудіосигнал приводиться до єдиної частоти дискретизації, ділиться на вікна фіксованої тривалості та кожне вікно класифікується нейромережею, після чого формується підсумкова оцінка за середнім значенням по сегментах. Фрагменти ключових налаштувань та аудіопайплайну наведено на рисунку 3.21.

```python
def get_env_float(name: str, default: float) -> float:  5 usages
    try:
        return float(os.getenv(name, default))
    except Exception:
        return default


def get_env_int(name: str, default: int) -> int:  4 usages
    try:
        return int(os.getenv(name, default))
    except Exception:
        return default


# Відео
BACKBONE = os.getenv("BACKBONE", "resnet50")
VIDEO_CKPT_PATH = Path(os.getenv("CHECKPOINT_PATH", ROOT_DIR / "resnet50_head.pt"))
FAKE_THRESHOLD = get_env_float( name: "FAKE_THRESHOLD",  default: 0.40)  # базовий поріг
FRAME_SAMPLE_RATE = get_env_int( name: "FRAME_SAMPLE_RATE",  default: 1)
FRAME_HIGH_THR = get_env_float( name: "FRAME_HIGH_THR",  default: 0.70)  # тільки для статистики
HIGH_FRAC_MIN = get_env_float( name: "HIGH_FRAC_MIN",  default: 0.30)    # тільки для статистики
FACE_MARGIN = get_env_int( name: "FACE_MARGIN",  default: 140)

# Аудіо
AUDIO_MODEL_CKPT = Path(os.getenv("AUDIO_MODEL_CKPT", ROOT_DIR / "wav2vec2_audio_ckpt.pt"))
AUDIO_STRICT_THRESHOLD = get_env_float( name: "AUDIO_STRICT_THRESHOLD",  default: 0.86)
AUDIO_SEGMENT_SECONDS = get_env_float( name: "AUDIO_SEGMENT_SECONDS",  default: 5.0)
AUDIO_MIN_WINDOWS = get_env_int( name: "AUDIO_MIN_WINDOWS",  default: 1)
AUDIO_FAKE_INDEX = get_env_int( name: "AUDIO_FAKE_INDEX",  default: 1)  # за замовчуванням fake = logit[1]
TARGET_SR = 16000

AUDIO_EXTS = {".wav", ".mp3", ".flac", ".ogg", ".m4a"}
```

Рисунок 3.21 – Фрагмент коду основних параметрів аудіомодуля та налаштування через змінні середовища

Підготовка навчальних прикладів для Wav2Vec2.0 організована через клас AudioSegmentDataset у train_audio_wav2vec2.py. Для кожного аудіофайлу сигнал завантажується у mono-форматі з цільовою частотою 16 кГц, а потім приводиться до фіксованої довжини сегмента, якщо запис довший, для train-вибірки береться випадковий фрагмент, а для val-вибірки центральний; якщо коротший, виконується доповнення нулями. Ключова логіка формування сегмента наведена на рисунку 3.22.

```python
class AudioSegmentDataset(Dataset):  2 usages
    def __init__(
        self,
        files,
        labels,
        segment_seconds: float = 5.0,
        target_sr: int = 16000,
        train: bool = True,
    ):
        assert len(files) == len(labels)
        self.files = list(files)
        self.labels = list(labels)
        self.segment_seconds = segment_seconds
        self.target_sr = target_sr
        self.segment_len = int(segment_seconds * target_sr)
        self.train = train

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

    def _load_wav(self, path: Path) -> torch.Tensor:  1 usage
        samples, sr = librosa.load(str(path), sr=self.target_sr, mono=True)
        return torch.from_numpy(samples.astype("float32"))

    def _crop_or_pad(self, x: torch.Tensor) -> torch.Tensor:  1 usage
        n = x.shape[0]
        if n >= self.segment_len:
            if self.train:
                max_start = max(n - self.segment_len, 0)
                start = random.randint( a: 0, max_start) if max_start > 0 else 0
            else:
                start = (n - self.segment_len) // 2
            return x[start:start + self.segment_len]
        else:
            pad_len = self.segment_len - n
            return torch.nn.functional.pad(x, (0, pad_len))
```

Рисунок 3.22 – Фрагмент коду логіки формування сегмента

Під час навчання батчі формуються через Wav2Vec2FeatureExtractor, який виконує паддінг та перетворює списки аудіфрагментів у тензори input_values та attention_mask. Далі модель Wav2Vec2ForSequenceClassification навчається оптимізатором AdamW, а найкращий чекпойнт зберігається за максимальним значенням F1 на val-вибірці. Фрагмент збереження найкращого чекпойнту наведено на рисунку 3.23.

```
    if val_metrics["f1"] > best_f1:
        best_f1 = val_metrics["f1"]
        state = {
            "model_name": args.model_name,
            "state_dict": model.state_dict(),
            "val_f1": best_f1,
            "config": vars(args),
        }
        torch.save(state, args.out)
        print(f"Saved best checkpoint to {args.out} (val_f1={best_f1:.3f})")

print("Готово.")
```

Рисунок 3.23 – Фрагмент коду збереження найкращого Wav2Vec2-чекпойнту за метрикою F1

У застосунку deepfake_app_av.py інференс аудіо реалізовано як послідовність, сегментація сигналу на вікна фіксованої тривалості та оцінювання кожного вікна моделлю Wav2Vec2.0. Сегментація формує лише повні вікна довжиною segment_seconds, а якщо аудіо коротше одного вікна, воно доповнюється нулями. Це пояснює, чому для деяких файлів кількість сегментів може бути 1. Фрагмент сегментації наведено на рисунку 3.24.

```
def split_audio_into_windows(samples: np.ndarray, sr: int, segment_seconds: float) -> List[np.ndarray]:  1 usage
    seg_len = int(segment_seconds * sr)
    n = samples.shape[0]
    if n <= seg_len:
        pad = seg_len - n
        seg = np.pad(samples,  pad_width: (0, pad))
        return [seg.astype("float32")]

    n_windows = n // seg_len
    windows = []
    for i in range(n_windows):
        start = i * seg_len
        end = start + seg_len
        seg = samples[start:end]
        windows.append(seg.astype("float32"))

    if not windows:
        pad = seg_len - n
        seg = np.pad(samples,  pad_width: (0, pad))
        windows = [seg.astype("float32")]

    return windows
```

Рисунок 3.24 – Фрагмент коду сегментації аудіосигналу на вікна фіксованої тривалості

Після сегментації кожне вікно подається у feature_extractor, модель повертає логіти, які перетворюються у ймовірність класу “fake”. Далі обчислюються агреговані значення p_mean та p_max, а підсумковий вердикт визначається порогом

AUDIO_STRICT_THRESHOLD за середнім значенням. Фрагмент інференсу наведено на рисунку 3.25.

```
for seg in windows:
    inputs = feature_extractor(
        raw_speech: [seg],
        sampling_rate=TARGET_SR,
        padding=True,
        return_tensors="pt",
    )
    input_values = inputs["input_values"].to(device)
    attention_mask = inputs.get("attention_mask")
    if attention_mask is not None:
        attention_mask = attention_mask.to(device)

    with torch.inference_mode():
        logits = model(
            input_values=input_values,
            attention_mask=attention_mask,
        ).logits
        probs = torch.softmax(logits, dim=-1)[0].cpu().numpy()
        p_fake = float(probs[AUDIO_FAKE_INDEX])
    p_fake_list.append(p_fake)

p_arr = np.array(p_fake_list, dtype=float)
p_mean = float(p_arr.mean())
p_max = float(p_arr.max())

is_fake = (p_mean >= AUDIO_STRICT_THRESHOLD)
```

Рисунок 3.25 – Фрагмент коду інференсу аудіомодуля та формування підсумкового рішення

Детальний підбір значення цього порога та його роль у мультимодальному «відео + аудіо» буде розглянуто в наступному підрозділі, присвяченому реалізації модуля «відео + аудіо» та порогів рішень.

3.5 Реалізація модуля агрегування результатів та порогів рішень

У цьому підрозділі описано реалізацію модуля інтеграції результатів відео- та аудіоаналізу у вкладці «Відео + Аудіо» веб-застосунку Streamlit. Рішення про синтетичність формується на основі порогів для кожної модальності, для відео використовується FAKE_THRESHOLD, для аудіо використовується AUDIO_STRICT_THRESHOLD. Значення цих порогів задаються через змінні середовища з дефолтами у коді (відео: 0.45, аудіо: 0.86), що показано на рисунку 3.26

```python
# Відео
BACKBONE = os.getenv("BACKBONE", "resnet50")
VIDEO_CKPT_PATH = Path(os.getenv("CHECKPOINT_PATH", ROOT_DIR / "resnet50_head.pt"))
FAKE_THRESHOLD = get_env_float( name: "FAKE_THRESHOLD",  default: 0.45)  # базовий поріг
FRAME_SAMPLE_RATE = get_env_int( name: "FRAME_SAMPLE_RATE",  default: 1)
FRAME_HIGH_THR = get_env_float( name: "FRAME_HIGH_THR",  default: 0.70)  # тільки для статистики
HIGH_FRAC_MIN = get_env_float( name: "HIGH_FRAC_MIN",  default: 0.30)    # тільки для статистики
FACE_MARGIN = get_env_int( name: "FACE_MARGIN",  default: 140)

# Аудіо
AUDIO_MODEL_CKPT = Path(os.getenv("AUDIO_MODEL_CKPT", ROOT_DIR / "wav2vec2_audio_ckpt.pt"))
AUDIO_STRICT_THRESHOLD = get_env_float( name: "AUDIO_STRICT_THRESHOLD",  default: 0.86)
AUDIO_SEGMENT_SECONDS = get_env_float( name: "AUDIO_SEGMENT_SECONDS",  default: 5.0)
AUDIO_MIN_WINDOWS = get_env_int( name: "AUDIO_MIN_WINDOWS",  default: 1)
AUDIO_FAKE_INDEX = get_env_int( name: "AUDIO_FAKE_INDEX",  default: 1)  # за замовчуванням fake = logit[1]
TARGET_SR = 16000

AUDIO_EXTS = {".wav", ".mp3", ".flac", ".ogg", ".m4a"}
```

Рисунок 3.26 – Фрагмент коду налаштування порогів і ключових параметрів мультимодального режиму

У межах мультимодального режиму підсумкове правило інтеграції реалізоване як SAFE_OR, якщо хоча б один модуль спрацював як «fake», фінальний вердикт також «fake» при цьому для відображення єдиної оцінки використовується максимальне значення ймовірності P(fake) між відео та аудіо. Фрагмент коду мультимодального обчислення наведено на рисунку 3.27.

```python
with col_right:
    st.markdown(
        "<div class='stats-card-title'>Об'єднана статистика</div>",
        unsafe_allow_html=True,
    )

    if video_result:
        v_p_mean = video_result["p_mean"]
        v_p_max = video_result["p_max"]
        v_frac_high = video_result["frac_high"]
        v_is_fake = video_result["is_fake"]
        v_n_frames = video_result["n_frames"]
    else:
        v_p_mean = v_p_max = v_frac_high = 0.0
        v_is_fake = False
        v_n_frames = 0

    if audio_result:
        a_p_mean = audio_result["p_mean"]
        a_p_max = audio_result["p_max"]
        a_is_fake = audio_result["is_fake"]
        n_segments = audio_result["n_segments"]
    else:
        a_p_mean = a_p_max = 0.0
        a_is_fake = False
        n_segments = 0
```

Рисунок 3.27 – Фрагмент коду мультимодального аналізу відео з витягуванням аудіо та правило SAFE_OR

Порогове значення для відеомодуля визначається окремим скриптом eval_resnet_val.py, він обчислює P(fake) для кожного відео на val-вибірці та перебирає набір кандидатних порогів у діапазоні 0.1–0.9, обираючи найкращий за метрикою F1. Після завершення скрипт виводить best_thr та підказку, які значення слід встановити у змінних середовища для застосунку (CHECKPOINT_PATH, BACKBONE, FAKE_THRESHOLD. Фрагмент коду підбір порога для відеодетектора на val-вибірці зображено на рисунку 3.28.

```
y_true = np.array([lab for _,lab,_ in rows], dtype=int)
pvals  = np.array([p   for _,_,p in rows], dtype=float)
best_thr, best_f1, best_acc = 0.5, -1.0, 0.0
for thr in np.linspace( start: 0.1,  stop: 0.9,  num: 17):
    y_pred = (pvals >= thr).astype(int)
    f1 = f1_score(y_true, y_pred) if len(set(y_true))>1 else 0.0
    acc = accuracy_score(y_true, y_pred)
    if f1 > best_f1:
        best_f1, best_thr, best_acc = f1, float(thr), float(acc)

print("\nBest threshold search (VAL):")
print(f"best_thr={best_thr:.2f} | video_f1={best_f1:.3f} | video_acc={best_acc:.3f}")
```

Рисунок 3.28 – Фрагмент коду підбіру порога для відеодетектора на val-вибірці

Результат роботи eval_resnet_val.py скрипта зображено на рисунку 3.29.

```
IMG_0629.MP4                                label=1  P_fake=0.739
IMG_0642.MP4                                label=1  P_fake=0.534
IMG_06701.MP4                               label=1  P_fake=0.736
IMG_0672.MP4                                label=1  P_fake=0.836
IMG_0681.MP4                                label=1  P_fake=0.647
IMG_0696.MP4                                label=1  P_fake=0.459
SaveTik.co_7540245845929807109.mp4          label=1  P_fake=0.701
IMG_05047.MOV                               label=1  P_fake=0.945

Best threshold search (VAL):
best_thr=0.45 | video_f1=0.809 | video_acc=0.787

Set in Streamlit env:
CHECKPOINT_PATH = C:\Users\Naziks\PycharmProjects\AI2\resnet50_head.pt
BACKBONE = resnet50
FAKE_THRESHOLD = 0.45
```

Рисунок 3.29 – Фрагмент коду визначення порогу скриптом eval_resnet_val.py

Для аудіомодуля у коді застосунку поріг рішення задається параметром AUDIO_STRICT_THRESHOLD (дефолт 0.86) і використовується під час формування ознаки is_fake як умова p_mean >= AUDIO_STRICT_THRESHOLD -L58. На валідаційному підборі отримано значення близьке до 0.86, однак у практичному режимі

застосунку обрано м’який поріг 0.60, щоб зменшити надмірну “жорсткість” рішення та уникати ситуацій, коли синтетичні фрагменти проходять як “real” через завищений поріг. Це значення встановлюється через змінну середовища AUDIO_STRICT_THRESHOLD, не змінюючи код (механізм get_env_float). Фрагмент коду застосування порога AUDIO_STRICT_THRESHOLD під час формування рішення аудіомодуля зображено на рисунку 3.30.

```
p_arr = np.array(p_fake_list, dtype=float)
p_mean = float(p_arr.mean())
p_max = float(p_arr.max())
frac_high = float((p_arr >= FRAME_HIGH_THR).mean())

is_fake = (p_mean >= FAKE_THRESHOLD)
```

Рисунок 3.30 – Фрагмент коду застосування порога AUDIO_STRICT_THRESHOLD під час формування рішення аудіомодуля

Результат роботи eval_audio_wav2vec2 скрипта зображено на рисунку 3.31.

```
cluster2__9499.wav                    label=1  P_fake=0.971
cluster2__9583.wav                    label=1  P_fake=0.961
cluster2__9641.wav                    label=1  P_fake=0.977
cluster2__9673.wav                    label=1  P_fake=0.963
cluster2__9684.wav                    label=1  P_fake=0.971
cluster2__9720.wav                    label=1  P_fake=0.956
cluster2__9790.wav                    label=1  P_fake=0.971
cluster2__9793.wav                    label=1  P_fake=0.982
cluster2__9794.wav                    label=1  P_fake=0.970
cluster2__9840.wav                    label=1  P_fake=0.973
cluster2__9857.wav                    label=1  P_fake=0.958
cluster2__9858.wav                    label=1  P_fake=0.980
cluster2__9891.wav                    label=1  P_fake=0.984
track_01_000-005.wav                  label=0  P_fake=0.177
track_02_005-010.wav                  label=0  P_fake=0.072
track_03_010-015.wav                  label=0  P_fake=0.044
track_04_015-020.wav                  label=0  P_fake=0.060

Best threshold search (VAL):
best_thr=0.86 | val_f1=0.800 | val_acc=0.876
```

Рисунок 3.31 – Фрагмент коду визначення порогу скриптом eval_audio_wav2vec2.py

Якщо у режимі «Відео + Аудіо» отримано v_p_mean = 0.47 при FAKE_THRESHOLD = 0.45, а також a_p_mean = 0.58 при AUDIO_STRICT_THRESHOLD = 0.60, тоді v_is_fake = True, a_is_fake = False, а фінальний вердикт combined_is_fake = True за правилом SAFE_OR, при цьому combined_p = max(0.47, 0.58) = 0.58.

3.6 Інтерфейс користувача та сценарії використання

Інтерфейс користувача системи реалізовано у вигляді вебзастосунку на базі Streamlit у файлі deepfake_app_av.py. На головному екрані застосунок відображає назву, коротке пояснення призначення та службові індикатори конфігурації у вигляді бейджів, зокрема пристрій виконання, активний backbone відеомодуля та встановлені пороги прийняття рішення для відео й аудіо. Нижче розміщено перемикач режимів аналізу з трьома опціями Відео, Аудіо, Відео + Аудіо, який задає подальший сценарій взаємодії користувача з системою. Загальний вигляд головного екрана та вибір режиму аналізу наведено на рисунку 3.32.

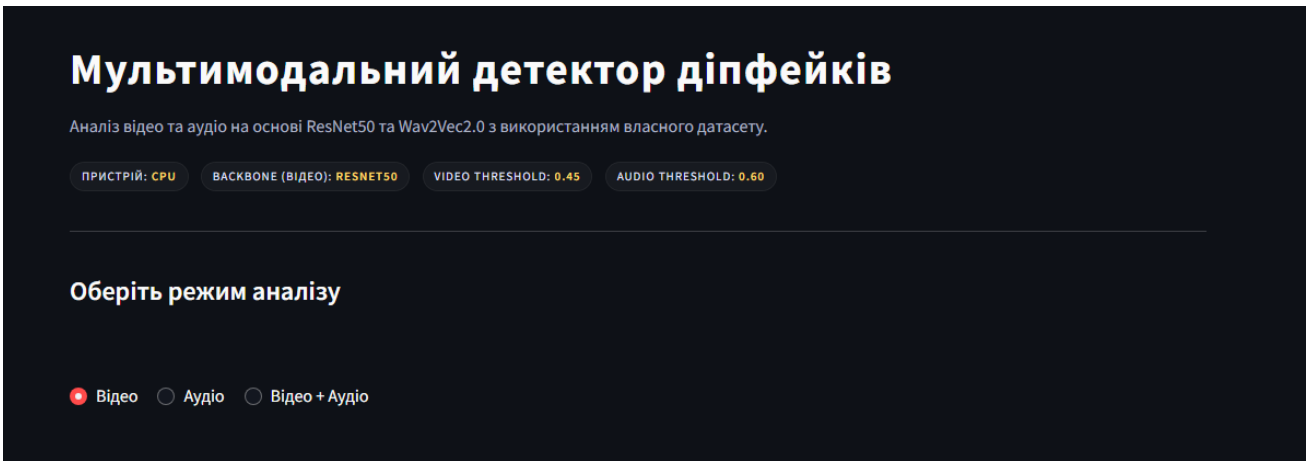


Рисунок 3.32 – Головний екран застосунку та вибір режиму аналізу

Фрагмент реалізації, що формує заголовок, блок бейджів параметрів та перемикач режимів у Streamlit, наведено на рисунку 3.33. Обране значення режиму зберігається у змінній mode та використовується для відображення відповідних секцій інтерфейсу (відеоаналізу, аудіоаналізу або мультимодального аналізу).

```python
st.title("Мультимодальний детектор діпфейків")
st.markdown(
    "<p class='subtitle'>Аналіз відео та аудіо на основі ResNet50 та Wav2Vec2.0 з використанням власного датасету.</p>",
    unsafe_allow_html=True,
)

st.markdown(
    f"""
    <div class="badges-row">
        <span>Пристрій: <strong>{device_str.upper()}</strong></span>
        <span>Backbone (відео): <strong>{BACKBONE}</strong></span>
        <span>VIDEO THRESHOLD: <strong>{FAKE_THRESHOLD:.2f}</strong></span>
        <span>AUDIO THRESHOLD: <strong>{AUDIO_STRICT_THRESHOLD:.2f}</strong></span>
    </div>
    """,
    unsafe_allow_html=True,
)

st.markdown("---")

st.markdown("#### Оберіть режим аналізу")
with st.container():
    st.markdown('<div class="mode-tabs">', unsafe_allow_html=True)
    mode = st.radio(
        "",
        options=["Відео", "Аудіо", "Відео + Аудіо"],
        horizontal=True,
    )
    st.markdown("</div>", unsafe_allow_html=True)
```

Рисунок 3.33 – Формування заголовка, бейджів параметрів та перемикача режимів у Streamlit

У режимі «Відео» користувачу надається форма запуску відеоаналізу, що включає інформаційні блоки з поясненням вхідних даних і майбутньої статистики, елемент завантаження відеофайлу, опцію обмеження аналізу першими 10 секундами, блок прев'ю оригінального відео та кнопку запуску. Після натискання кнопки застосунок переходить у стан виконання (spinner) і викликає процедуру аналізу відео з урахуванням параметра trim_first_10s. Загальний вигляд форми запуску відеоаналізу наведено на рисунку 3.34.

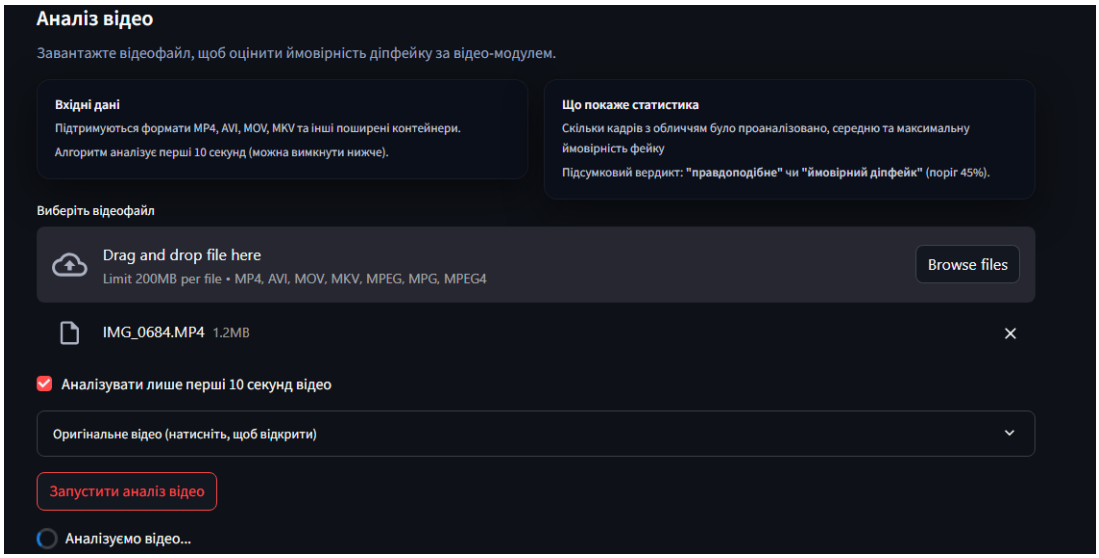


Рисунок 3.34 – Форма запуску відеоаналізу у режимі «Відео»

Фрагмент реалізації, що відповідає за завантаження відео, збереження файлу у тимчасовий шлях, відображення прев'ю у прихованому блоці та запуск обчислень кнопкою «Запустити аналіз відео», наведено на рисунку 3.35. Така організація інтерфейсу дозволяє користувачу виконати сценарій у кілька кроків, завантажити файл, за потреби увімкнути швидкий аналіз перших 10 секунд та запустити перевірку.

```
uploaded_video = st.file_uploader(
    "Виберіть відеофайл",
    type=["mp4", "avi", "mov", "mkv", "mpeg", "mpg"],
)

trim_first_10s = st.checkbox("Аналізувати лише перші 10 секунд відео", value=True)

if uploaded_video is not None:
    with st.expander("Оригінальне відео (натисніть, щоб відкрити)", expanded=False):
        st.video(uploaded_video)

    tmp_video_path = ROOT_DIR / ("tmp_video_" + uploaded_video.name)
    with open(tmp_video_path, "wb") as f:
        f.write(uploaded_video.getbuffer())

    if st.button("Запустити аналіз відео"):
        with st.spinner("Аналізуємо відео..."):
            video_result = analyze_video(tmp_video_path, device_str, trim_first_10s)
```

Рисунок 3.35 – Фрагмент коду завантаження відео, опція перших 10 секунд, прев'ю та запуск аналізу

Далі, після завершення обчислень, результати виводяться у дві колонки, зліва показується кадр-прев'ю з виділеним обличчям, справа формується блок статистики класифікації. У статистиці відображається середня ймовірність фейку (p_mean) як основний показник, а також допоміжні метрики, кількість проаналізованих кадрів, максимальна ймовірність (p_max) і частка кадрів із високою ймовірністю (frac_high). Фінальний текстовий вердикт у відеорежимі визначається порогом FAKE_THRESHOLD та умовою video_result["is_fake"]. На рисунку 3.36 показано інтерфейс відображення результатів відеомодуля після завершення аналізу, зокрема кадр-прев'ю та статистику класифікації.

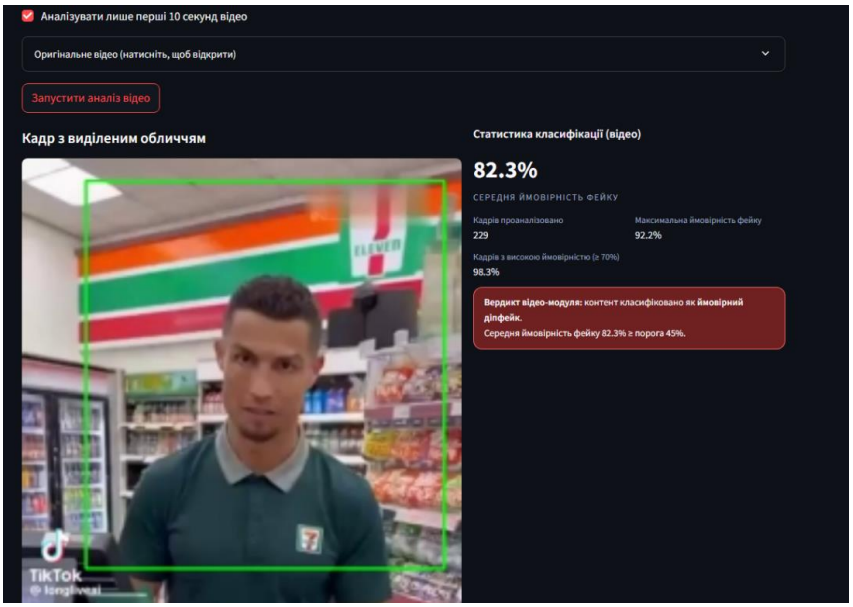


Рисунок 3.36 – Фрагмент відображення результатів відеоаналізу, попередній перегляд кадру та статистика класифікації

На рисунку 3.37 наведено відповідний фрагмент реалізації в deepfake_app_av.py, який формує ці елементи інтерфейсу та текстовий вердикт на основі порога FAKE_THRESHOLD.

```
else:
    col_left, col_right = st.columns([1.4, 1])

    with col_left:
        st.markdown("#### Кадр з виділеним обличчям")
        if video_result["preview_frame"] is not None:
            st.image(
                video_result["preview_frame"],
                caption="Перший знайдений кадр з обличчям",
                use_container_width=True,
            )
        else:
            st.info("Не вдалося побудувати прев'ю з обличчям.")

    with col_right:
        st.markdown(
            "<div class='stats-card-title'>Статистика класифікації (відео)</div>",
            unsafe_allow_html=True,
        )

        p_mean = video_result["p_mean"]
        p_max = video_result["p_max"]
        frac_high = video_result["frac_high"]

        st.markdown(
            f"""
            <div class="big-number">{p_mean * 100:.1f}%</div>
            <div class="big-number-label">СЕРЕДНЯ ЙМОВІРНІСТЬ ФЕЙКУ</div>
            """,
            unsafe_allow_html=True,
        )

        m1, m2 = st.columns(2)
        with m1:
            st.markdown(
                f"<div class='small-label'>Кадрів проаналізовано</div>"
                f"<div class='small-value'>{video_result['n_frames']}</div>",
                unsafe_allow_html=True,
            )
```

Рисунок 3.37 – Фрагмент виводу попереднього перегляду кадру, статистики та вердикту відеомодуля (Streamlit)

У режимі «Аудіо» користувачу надається форма запуску аудіоаналізу, інформаційні блоки з поясненням вхідних даних і статистики, завантаження

аудіофайлу, блок прослуховування оригінального сигналу та кнопка запуску обчислень. Загальний вигляд цієї форми наведено на рисунку 3.38.

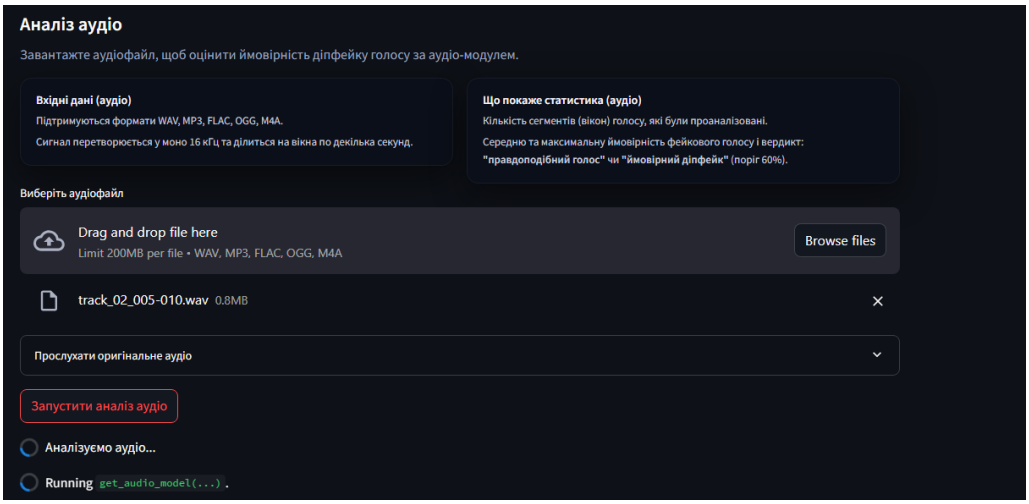


Рисунок 3.38 – Форма запуску аудіоаналізу у режимі «Аудіо»

Після натискання кнопки застосунок переходить у стан виконання (spinner) та викликає функцію analyze_audio, при цьому під час першого запуску може відображатися службовий статус ініціалізації моделі (Running get_audio_model), оскільки завантаження моделі кешується. Фрагмент реалізації форми завантаження і запуску аудіоаналізу наведено на рисунку 3.39.

```
elif mode == "Аудіо":
    st.markdown("<div class='section-title'>Аналіз аудіо</div>", unsafe_allow_html=True)
    st.markdown(
        "<p class='section-caption'>Завантажте аудіофайл, щоб оцінити ймовірність діпфейку голосу за аудіо-модулем.</p>",
        unsafe_allow_html=True,
    )

    # Інформаційні блоки для аудіо
    top_col1, top_col2 = st.columns(2)
    with top_col1:
        st.markdown(
            """
            <div class="info-card">
                <div class="info-card-title">Вхідні дані (аудіо)</div>
                <div>Підтримуються формати WAV, MP3, FLAC, OGG, M4A.</div>
                <div style="margin-top:4px;">Сигнал перетворюється у моно 16 кГц та ділиться на вікна по декілька секунд.</div>
            </div>
            """,
            unsafe_allow_html=True,
        )
    with top_col2:
        st.markdown(
            f"""
            <div class="info-card">
                <div class="info-card-title">Що покаже статистика (аудіо)</div>
                <div>Кількість сегментів (вікон) голосу, які були проаналізовані.</div>
                <div style="margin-top:4px;">Середню та максимальну ймовірність фейкового голосу і вердикт:
                <b>"правдоподібний голос"</b> чи <b>"ймовірний діпфейк"</b> (поріг {AUDIO_STRICT_THRESHOLD*100:.0f}%).</div>
            </div>
            """,
            unsafe_allow_html=True,
        )

    uploaded_audio = st.file_uploader(
        "Виберіть аудіофайл",
        type=["wav", "mp3", "flac", "ogg", "m4a"],
    )
```

Рисунок 3.39 – Код завантаження аудіо, прослуховування та запуск аналізу

Після завершення обчислень результати аудіомодуля виводяться у вигляді статистики класифікації, середня ймовірність фейку як основний показник, кількість

сегментів (вікон) та максимальна ймовірність серед сегментів. Далі формується текстовий вердикт шляхом порівняння середньої ймовірності p_mean із порогом AUDIO_STRICT_THRESHOLD. Приклад такого виводу результатів наведено на рисунку 3.40.

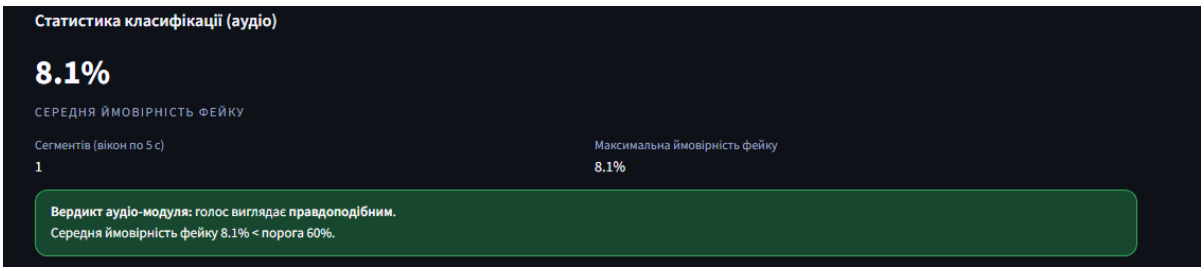


Рисунок 3.40 – Фрагмент відображення статистики класифікації аудіо та вердикту аудіомодуля

Відповідний фрагмент коду, що формує блок статистики та вердикт, наведено на рисунку 3.41.

```
p_mean = audio_result["p_mean"]
p_max = audio_result["p_max"]
n_segments = audio_result["n_segments"]

st.markdown(
    "<div class='stats-card-title'>Статистика класифікації (аудіо)</div>",
    unsafe_allow_html=True,
)

st.markdown(
    f"""
    <div class="big-number">{p_mean * 100:.1f}%</div>
    <div class="big-number-label">СЕРЕДНЯ ЙМОВІРНІСТЬ ФЕЙКУ</div>
    """,
    unsafe_allow_html=True,
)

m1, m2 = st.columns(2)
with m1:
    st.markdown(
        f"<div class='small-label'>Сегментів (вікон по {AUDIO_SEGMENT_SECONDS:.0f} с)</div>"
        f"<div class='small-value'>{n_segments}</div>",
        unsafe_allow_html=True,
    )
with m2:
    st.markdown(
        f"<div class='small-label'>Максимальна ймовірність фейку</div>"
        f"<div class='small-value'>{p_max * 100:.1f}%</div>",
        unsafe_allow_html=True,
    )

if audio_result["is_fake"]:
    st.markdown(
        f"""
        <div class="verdict-box-bad">
        <b>Вердикт аудіо-модуля:</b> голос класифіковано як <b>ймовірний діпфейк</b>.<br/>
        Середня ймовірність фейку {p_mean*100:.1f}% ≥ порога {AUDIO_STRICT_THRESHOLD*100:.0f}%.
```

Рисунок 3.41 – Код виведення вивід статистики (p_mean, p_max, n_segments) та вердикту аудіомодуля у Streamlit

У вкладці «Відео + Аудіо» реалізовано спільний аналіз двох модальностей у межах одного сценарію Streamlit, користувач завантажує відеофайл, після чого

застосунок послідовно запускає відеодетектор і аудіодетектор (аудіо витягується з того ж відео). Форма запуску мультимодального аналізу з пояснювальними блоками, завантаженням відео, опцією обмеження аналізу першими 10 секундами та кнопкою запуску наведена на рисунку 3.42.

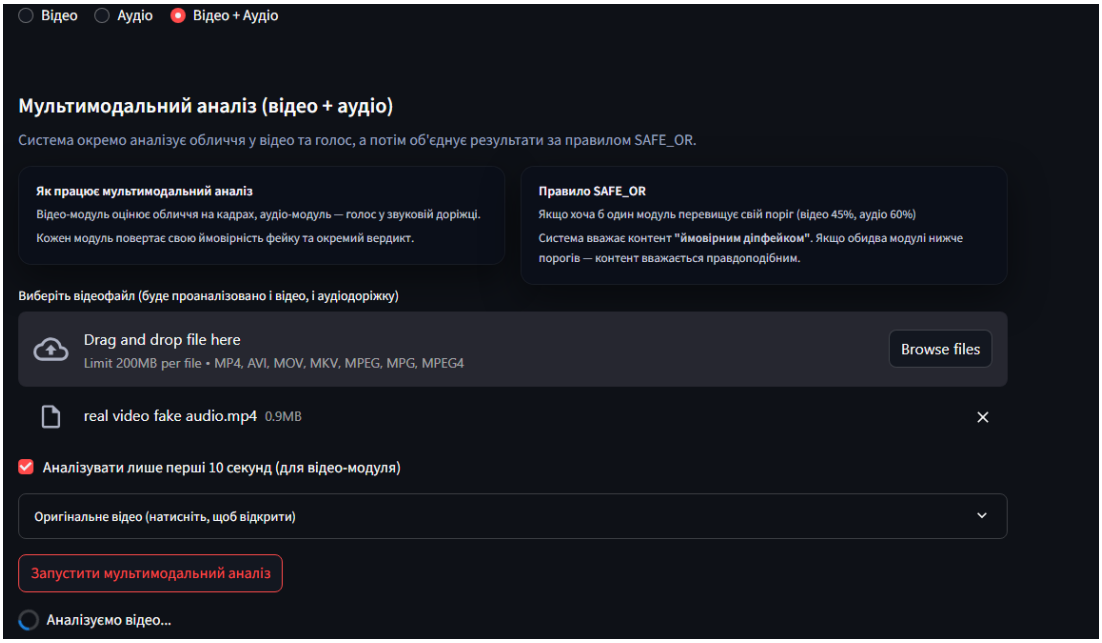


Рисунок 3.42 – Форма запуску мультимодального аналізу

Фрагмент коду, що реалізує завантаження відео, опцію trim_first_10s_both, витягування аудіодоріжки та запуск аналізу обох модулів, наведено на рисунку 3.43.

```
uploaded_video_both = st.file_uploader(
    "Виберіть відеофайл (буде проаналізовано і відео, і аудіодоріжку)",
    type=["mp4", "avi", "mov", "mkv", "mpeg", "mpg"],
)

trim_first_10s_both = st.checkbox("Аналізувати лише перші 10 секунд (для відео-модуля)", value=True)

if uploaded_video_both is not None:
    with st.expander("Оригінальне відео (натисніть, щоб відкрити)", expanded=False):
        st.video(uploaded_video_both)

    tmp_video_path = ROOT_DIR / ("tmp_both_video_" + uploaded_video_both.name)
    tmp_audio_from_video = ROOT_DIR / ("tmp_from_video_audio.wav")

    with open(tmp_video_path, "wb") as f:
        f.write(uploaded_video_both.getbuffer())

    if st.button("Запустити мультимодальний аналіз"):
        video_result = None
        audio_result = None

        try:
            with st.spinner("Аналізуємо відео..."):
                video_result = analyze_video(tmp_video_path, device_str, trim_first_10s_both)

            with st.spinner("Витягуємо аудіо з відео та аналізуємо голос..."):
                extract_audio_from_video(tmp_video_path, tmp_audio_from_video, TARGET_SR)
                audio_result = analyze_audio(tmp_audio_from_video, device_str)

            col_left, col_right = st.columns([1.4, 1])
```

Рисунок 3.43 – Фрагмент коду запуску мультимодального аналізу, завантаження відео, витягування аудіо та виклик відео- й аудіомодулів

Після завершення обчислень результати двох модулів інтегруються за правилом SAFE_OR, фінальний вердикт «ймовірний діпфейк», якщо хоча б один із модулів перевищив свій поріг для єдиної числової оцінки виводиться максимальне значення P(fake) між відео та аудіо. Відображення результатів мультимодального аналізу у вигляді прев'ю кадру з обличчям (з відеомодуля), об'єднаної статистики та підсумкового вердикту наведено на рисунку 3.44.

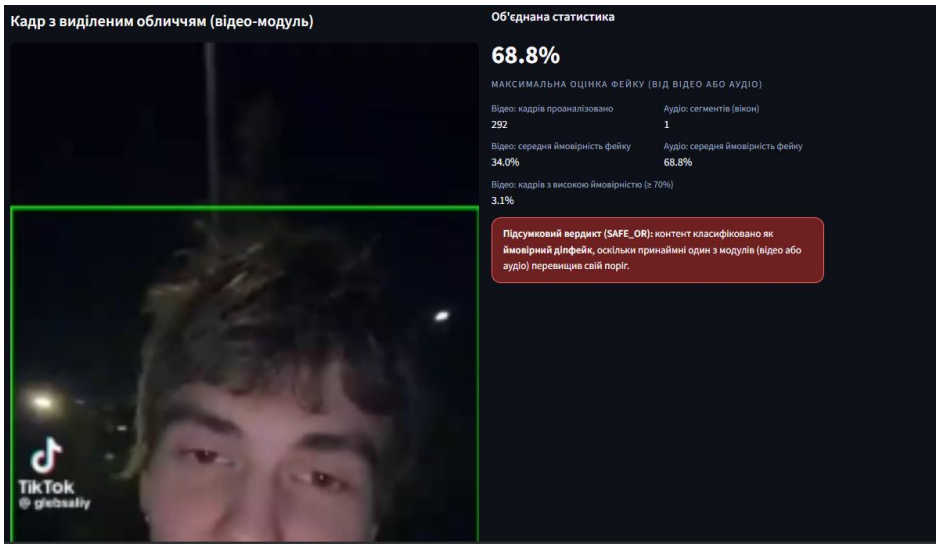


Рисунок 3.44 – Фрагмент відображення результатів мультимодального аналізу

Фрагмент коду обчислення combined_is_fake та combined_p і формування блоку «Об'єднана статистика» з вердиктом SAFE_OR наведено на рисунку 3.45.

```
col_left, col_right = st.columns([1.4, 1])

with col_left:
    st.markdown("##### Кадр з виділеним обличчям (відео-модуль)")
    if video_result and video_result.get("preview_frame") is not None:
        st.image(
            video_result["preview_frame"],
            caption="Перший знайдений кадр з обличчям",
            use_container_width=True,
        )
    else:
        st.info("Не вдалося побудувати прев'ю з обличчям.")

with col_right:
    st.markdown(
        "<div class='stats-card-title'>Об'єднана статистика</div>",
        unsafe_allow_html=True,
    )

    if video_result:
        v_p_mean = video_result["p_mean"]
        v_p_max = video_result["p_max"]
        v_frac_high = video_result["frac_high"]
        v_is_fake = video_result["is_fake"]
        v_n_frames = video_result["n_frames"]
    else:
        v_p_mean = v_p_max = v_frac_high = 0.0
        v_is_fake = False
        v_n_frames = 0

    if audio_result:
        a_p_mean = audio_result["p_mean"]
        a_p_max = audio_result["p_max"]
        a_is_fake = audio_result["is_fake"]
        n_segments = audio_result["n_segments"]
    else:
        a_p_mean = a_p_max = 0.0
        a_is_fake = False
```

Рисунок 3.45 – Фрагмент коду інтеграції результатів відео й аудіо за правилом SAFE_OR та формування об'єднаної статистики

Як на рисунку 3.44, якщо для відео v_p_mean = 0.34 при порозі 0.45, а для аудіо a_p_mean = 0.688 при порозі 0.60, тоді v_is_fake = False, a_is_fake = True, фінальний вердикт combined_is_fake = True за правилом SAFE_OR, а єдина оцінка combined_p = max(0.34, 0.688) = 0.688 (68.8 %).

3.7 Експериментальна перевірка і результати

У межах експериментальної перевірки відеомодуля виконано тестування на прикладах різної складності, від випадків із вираженими ознаками синтетичності до високоякісних генерацій, які можуть виглядати правдоподібно без знання контексту. Окрім фейкових зразків, додатково перевіряються й реальні відео з двох джерел (офіційний Telegram-канал та TikTok), що дає змогу порівняти поведінку системи на «fake» і «real» контенті в практичних сценаріях.

Кейс 1 (очевидний діпфейк у відео) демонструє ситуацію, коли система впевнено спрацьовує за рахунок великої кількості кадрів із вираженими ознаками синтетичності, на рисунку 3.46 середня ймовірність фейку становить 64,5%, що перевищує поріг 45%, при цьому проаналізовано 144 кадри, максимальна ймовірність сягає 97,1%, а частка кадрів із високою ймовірністю (≥70%) становить 41,0%, що узгоджується з фінальним вердиктом про «ймовірний діпфейк».

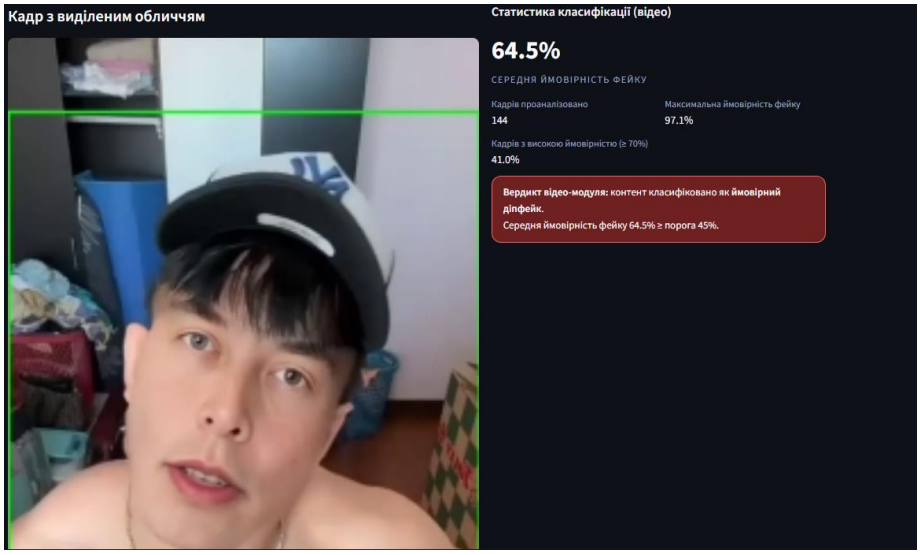


Рисунок 3.46 – Фрагмент очевидного діпфейку у відео

Переходячи до складніших прикладів, важливо враховувати, що у високоякісних генераціях ознаки підробки можуть проявлятися менш стабільно по всій послідовності кадрів, тому показники «піків» і частки високих кадрів часто знижуються, хоча середнє значення може залишатися вище порога.

Кейс 2 (складніший діпфейк високої якості) відображає саме такий сценарій На рисунку 3.47 відеомодуль також класифікує контент як «ймовірний діпфейк», оскільки середня ймовірність фейку становить 57,1% і перевищує поріг 45%, однак частка кадрів із високою ймовірністю (≥70%) є меншою (15,0%), що вказує на менш однорідний розподіл ознак синтетичності, водночас максимальна ймовірність фейку досягає 84,8%, і сукупність цих значень забезпечує спрацювання системи навіть у випадках, які без контексту можуть сприйматися як правдоподібні.

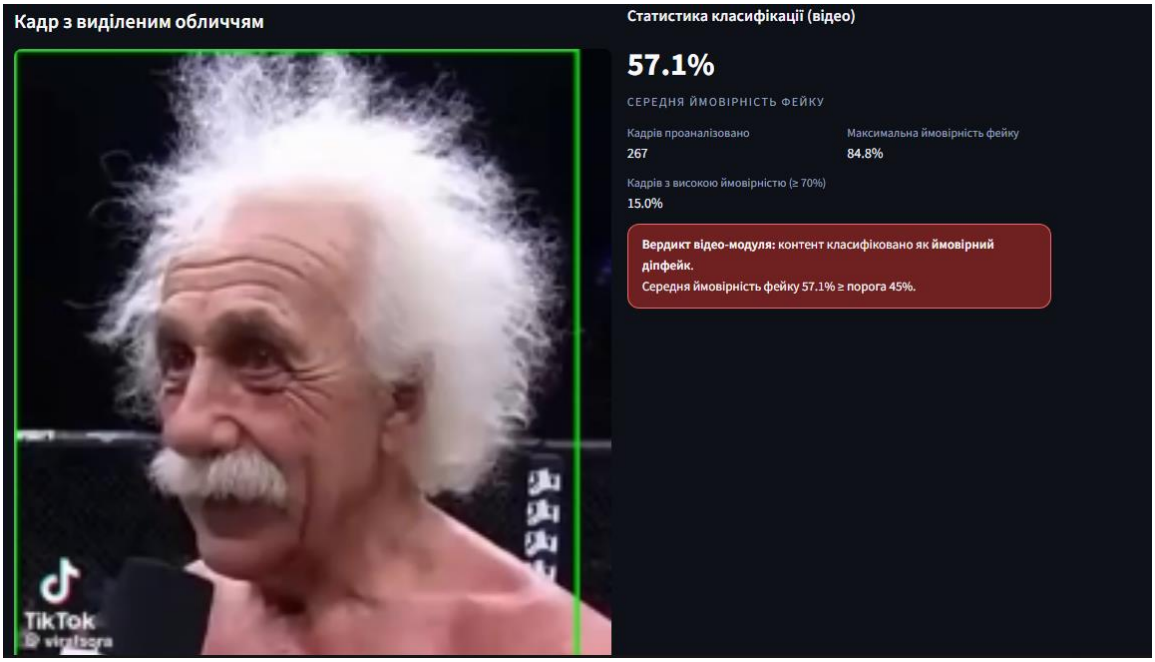


Рисунок 3.47 – Фрагмент діпфейку високої якості

На наступному етапі експериментальної перевірки виконано тестування відеомодуля на реальних відеоматеріалах із відкритих джерел, щоб оцінити, чи не виявляє система хибнопозитивних спрацювань у типових умовах зйомки та публікації контенту.

На рисунку 3.48 показано результат аналізу реального відео з TikTok. Відеомодуль сформував прев'ю кадру з виділеним обличчям та відобразив статистику

класифікації, середня ймовірність фейку становить 19,3%, що є нижчим за поріг 45%, тому фінальний вердикт визначено як «відео виглядає правдоподібним». Додатково показано кількість проаналізованих кадрів (300), максимальну ймовірність фейку (82,3%) та частку кадрів із високою ймовірністю (≥70%), яка становить 1,3%. Наявність поодиноких пікових значень при низькому середньому та малій частці "високих" кадрів відповідає очікуваній поведінці детектора на реальному контенті, де окремі кадри можуть містити артефакти освітлення, компресії або руху, але не формують стійкого сигналу синтетичності.

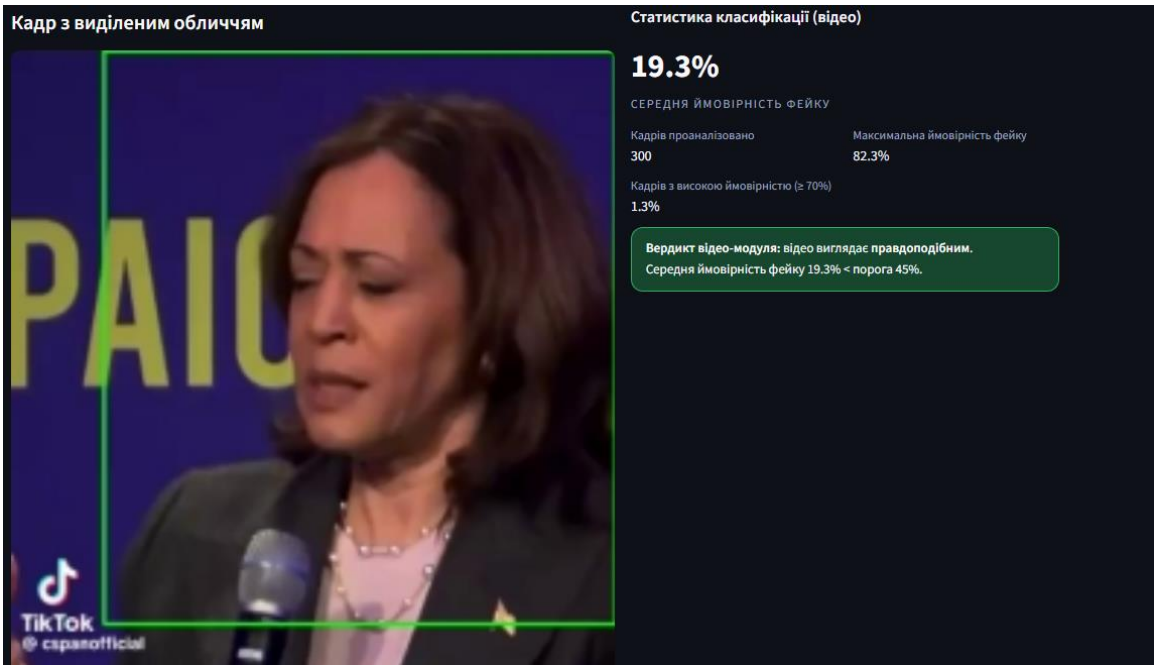


Рисунок 3.48 – Фрагмент результату відеоаналізу реального відео з TikTok у режимі «Відео»

На рисунку 3.49 наведено результат аналізу реального відео з офіційного Telegram-каналу. Система також відобразила прев'ю кадру з обличчям і статистику класифікації, де середня ймовірність фейку становить 39,3% при порозі 45%, тобто залишається нижче порогового значення, і фінальний вердикт сформовано як «відео виглядає правдоподібним». При цьому проаналізовано 251 кадр, максимальна ймовірність фейку досягає 85,8%, а частка кадрів із високою ймовірністю (≥70%) становить 9,2%. Такі значення свідчать, що в реальному відео можливі локальні "підозрілі" кадри (наприклад, через низьку освітленість, зернистість, шум або

агресивну компресію), однак середній показник не перетинає поріг, тому система не робить хибного висновку про діпфейк.

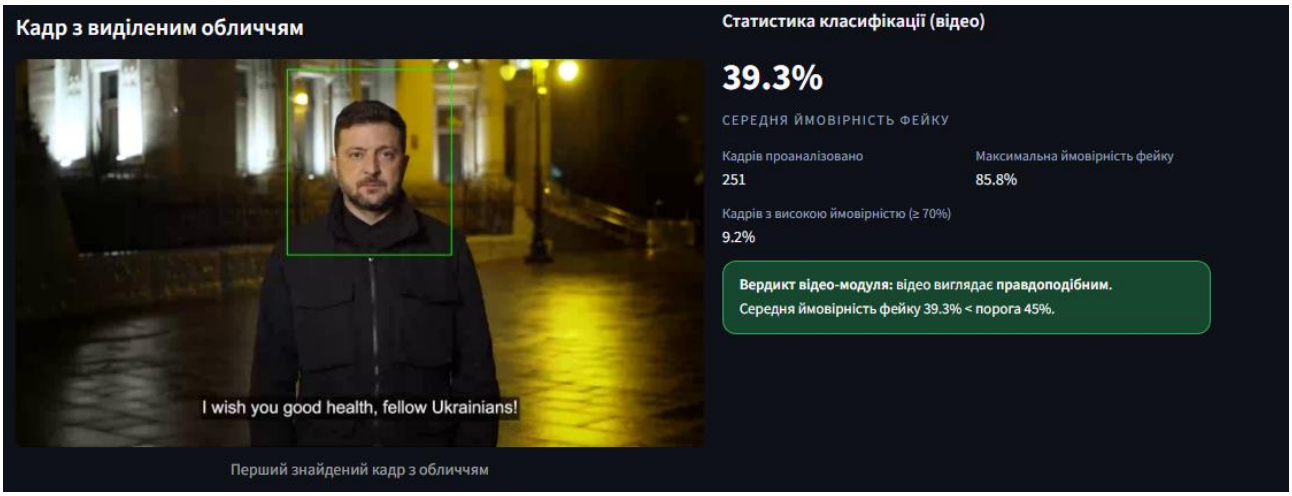


Рисунок 3.49 – Фрагмент результату відеоаналізу реального відео з офіційного Telegram-каналу у режимі «Відео»

У вкладці «Аудіо» виконано експериментальну перевірку на 8 зразках синтетичного мовлення, 6 аудіофрагментів, згенерованих безкоштовним українським TTS, а також 2 фрагменти, отримані за допомогою платного AI-сервісу. Для кожного прикладу система відображає середню та максимальну ймовірність фейку, кількість сегментів (вікон) і підсумковий вердикт за порогом 60%. Нижче наведено готові текстові вставки з місцями для рисунків і рекомендованими підписами (нумерацію рисунків підставте за порядком у вашому документі).

Перший приклад (безкоштовний TTS) демонструє впевнене спрацювання аудіомодуля, середня ймовірність фейку становить 93,2%, що суттєво перевищує поріг 60%, тому вердикт формується як «ймовірний діпфейк». Відповідний результат роботи модуля наведено на рисунку 3.50.

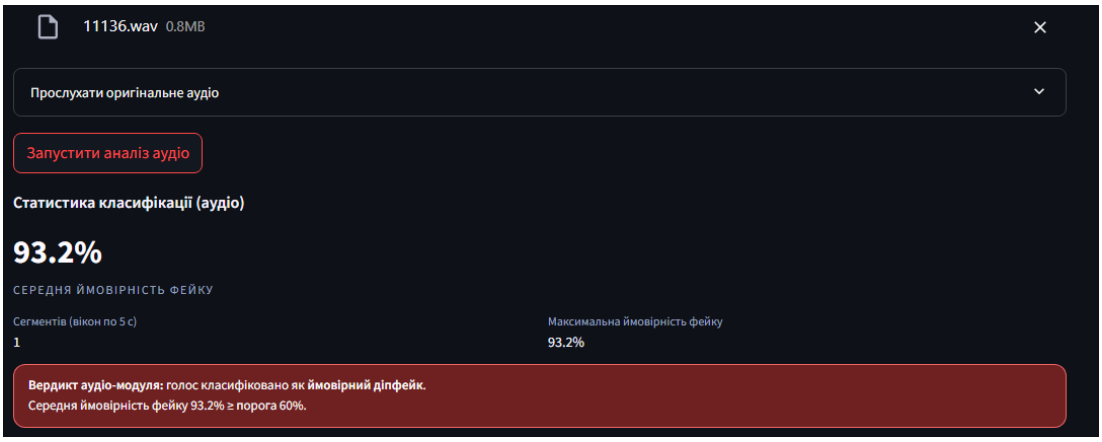


Рисунок 3.50 – Фрагмент результату аудіоаналізу синтетичного голосу (безкоштовний TTS) P(fake)mean = 93,2% > 60%

Другий приклад (безкоштовний TTS) також класифікується як синтетичний, отримано 80,6% середньої ймовірності фейку, що стабільно вище порога, отже система відносить голос до фейкового класу. Як саме це відображається в інтерфейсі застосунку показано на рисунку 3.51.

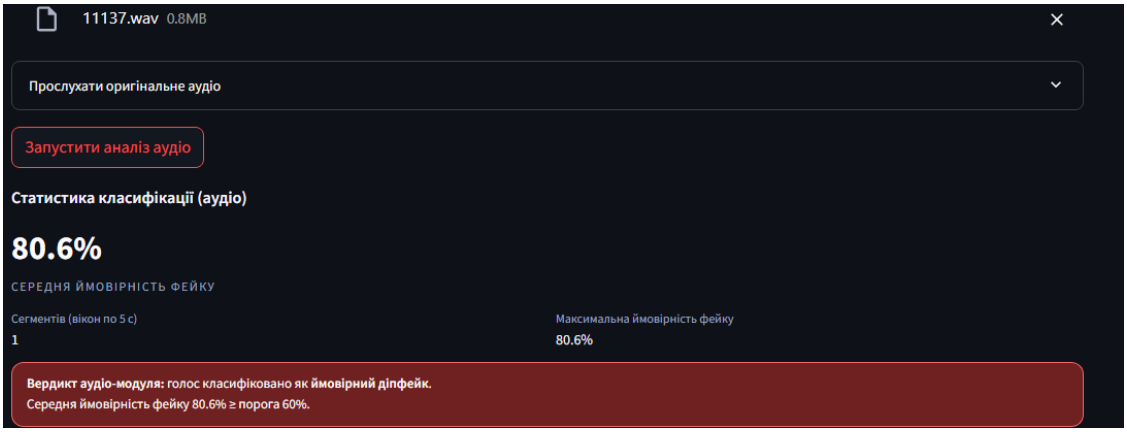


Рисунок 3.51 – Фрагмент результату аудіоаналізу синтетичного голосу (безкоштовний TTS) P(fake)mean = 80,6% > 60%

У третьому прикладі (безкоштовний TTS) спостерігається максимально виражена синтетичність за оцінкою моделі, P(fake)mean = 98,2%, що відповідає високій впевненості детектора та однозначному вердикту «ймовірний діпфейк». Приклад такого спрацювання наведено на рисунку 3.52.

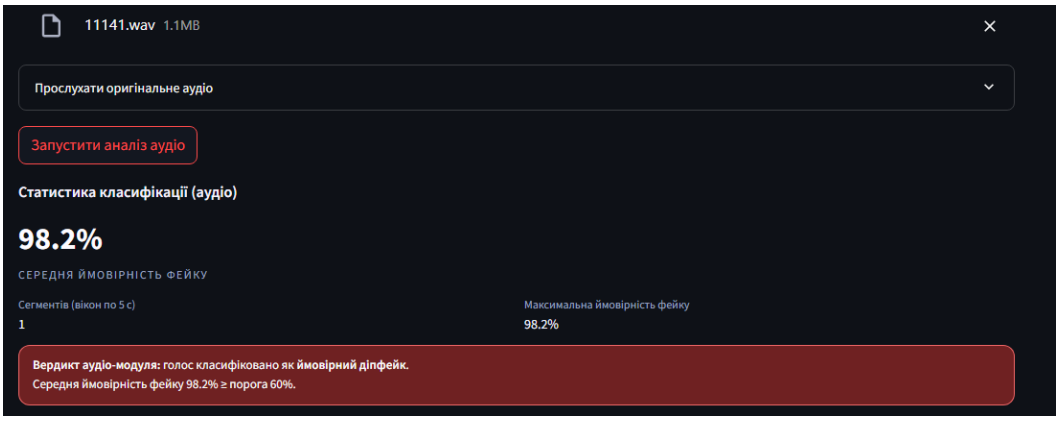


Рисунок 3.52 – Фрагмент результату аудіоаналізу синтетичного голосу (безкоштовний TTS) P(fake)mean = 98,2% > 60%

Четвертий приклад (безкоштовний TTS) підтверджує стійкість рішення аудіомодуля на різних голосах, при P(fake)mean = 92,3% система зберігає класифікацію «фейк» із суттєвим запасом відносно порога. Результат цього тесту подано на рисунку 3.53.

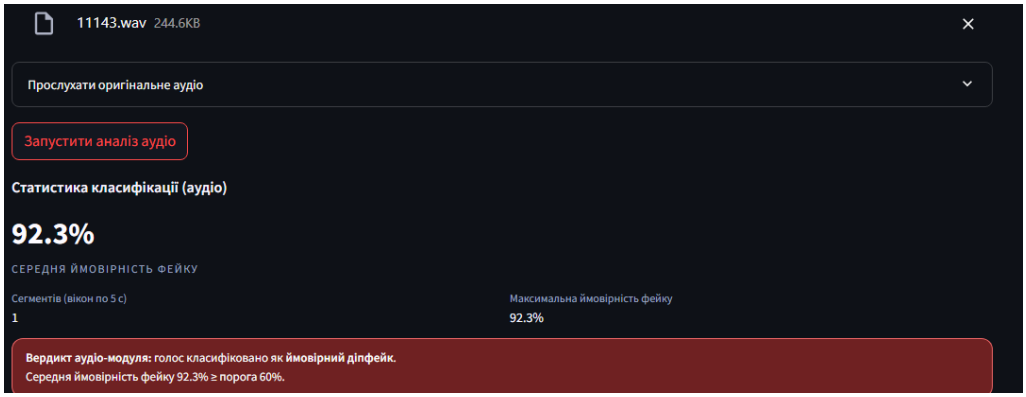


Рисунок 3.53 – Фрагмент результату аудіоаналізу синтетичного голосу (безкоштовний TTS) P(fake)mean = 92,3% > 60%

П’ятий приклад (безкоштовний TTS) має значення 98,3%, тобто модель виявляє характерні ознаки синтетичного голосу навіть на короткому фрагменті та формує вердикт «ймовірний діпфейк». Візуалізація статистики та вердикту наведена на рисунку 3.54.

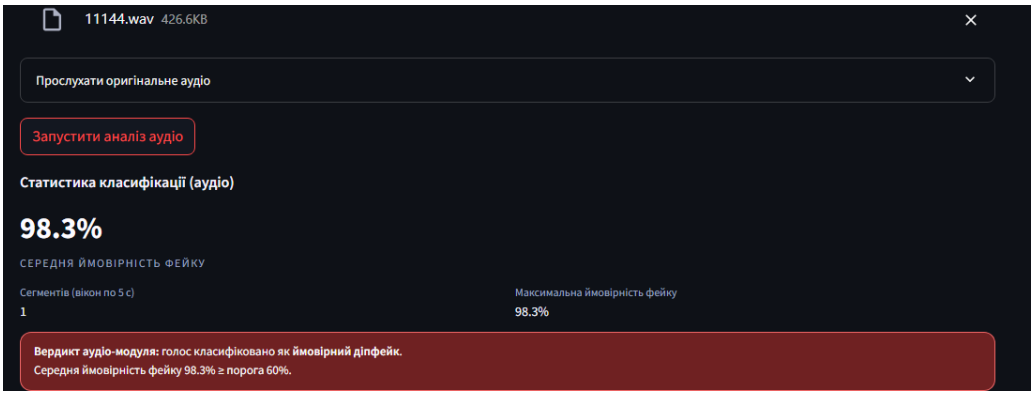


Рисунок 3.54 – Фрагмент результату аудіоаналізу синтетичного голосу (безкоштовний TTS) P(fake)mean = 98,3% > 60%

Шостий приклад (безкоштовний TTS) також впевнено перевищує порогове значення, P(fake)mean = 95,5%. Це означає, що для даного типу генерації аудіомодуль забезпечує високу чутливість до синтетичності голосу. Даний результат відображено на рисунку 3.55.

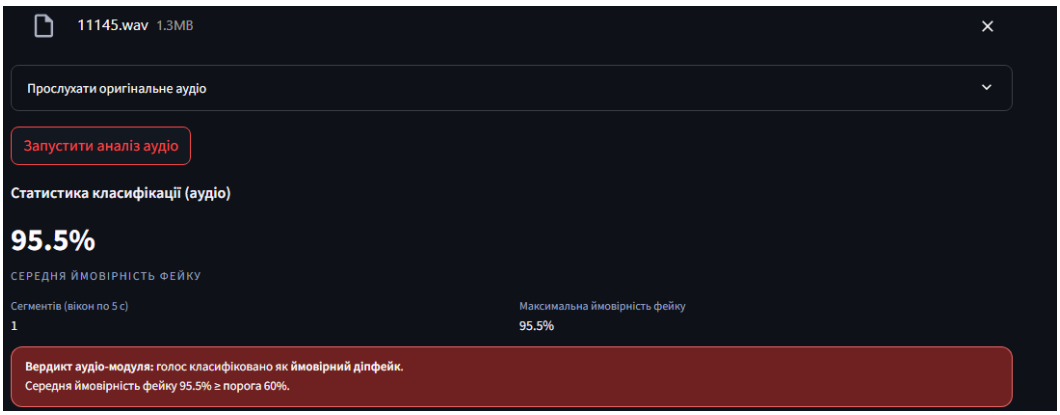


Рисунок 3.55 – Фрагмент результату аудіоаналізу синтетичного голосу (безкоштовний TTS) P(fake)mean = 95,5% > 60%

Далі наведено два приклади, згенеровані платним AI-сервісом, які в середньому мають більш «натуральне» звучання та можуть давати нижчі значення ймовірності фейку. У сьомому прикладі отримано P(fake)mean = 61,4%, значення лише незначно перевищує поріг 60%, але система все одно класифікує голос як «ймовірний діпфейк», що ілюструє прикордонний випадок для налаштованого порога рішення. Результат наведене на рисунку 3.56

Рисунок 3.56 – Фрагмент результату аудіоаналізу синтетичного голосу (платний AI-сервіс) P(fake)mean = 61,4% (поблизу порога 60%)

Восьмий приклад (платний AI-сервіс) демонструє вже впевнене спрацювання аудіомодуля, P(fake)mean = 87,4%, що суттєво перевищує поріг, тому підсумковий вердикт однозначно формується як «ймовірний діпфейк». Візуалізація статистики та вердикту наведена на рисунку 3.57.

Рисунок 3.57 – Фрагмент результату аудіоаналізу синтетичного голосу (платний AI-сервіс) P(fake)mean = 87,4% > 60%

У наведеному прикладі рисунок 3.58, результатів аудіоаналізу система відобразила 6 сегментів (вікон по 5 с), оскільки вхідний аудіофайл мав тривалість близько 30 секунд і під час обробки був автоматично поділений на рівні часові інтервали фіксованої довжини. Таким чином, аудіомодуль виконав класифікацію кожного з 6 вікон окремо, після чого сформував зведені показники у вигляді середньої та максимальної ймовірності синтетичності голосу, на основі яких виводиться підсумковий вердикт.

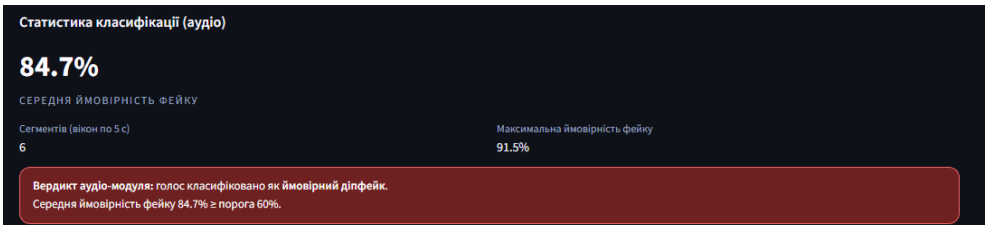


Рисунок 3.58 – Фрагмент результату аудіоаналізу синтетичного голосу (платний AI-сервіс) P(fake)mean = 84,7% > 60%

У наведеному прикладі на рисунку 3.59 показано результат перевірки першої аудіодоріжки, яку (за умовами експерименту) було витягнуто з відео Telegram-каналу та подано на вхід аудіомодуля. Оскільки тривалість файлу становить 5 секунд, система сформувала лише 1 сегмент (вікно по 5 с) і виконала класифікацію цього фрагмента як «правдоподібний голос», середня та максимальна ймовірність синтетичності збігаються і становлять P(fake)mean = 0,1%, що суттєво нижче порога 60%, тому підсумковий вердикт відображається у зеленому блоці як «реальне».

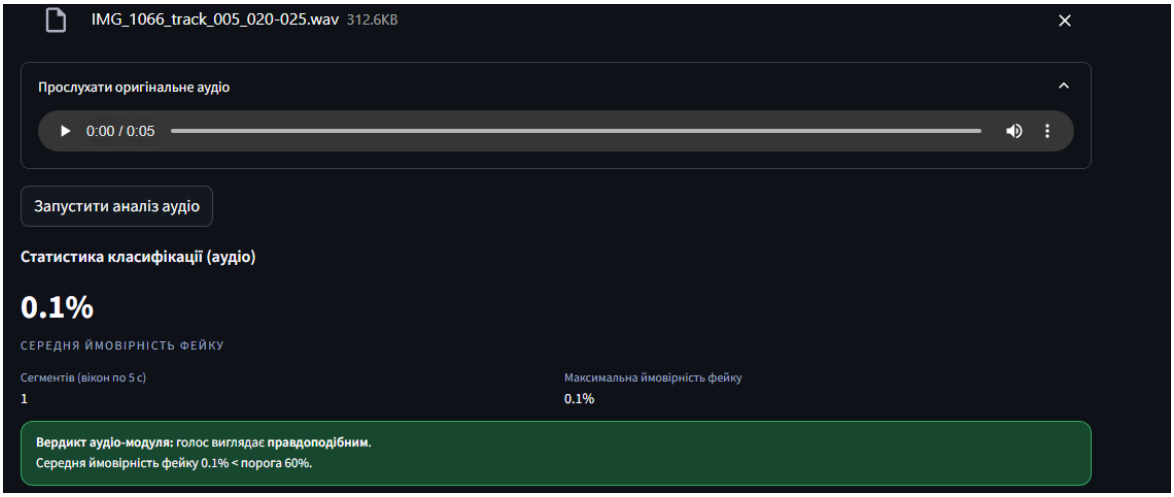


Рисунок 3.59 – Фрагмент результату аудіоаналізу реального голосу P(fake)mean = 0,1% < 60%

Додатково, для контролю відтворюваності результату на іншому фрагменті з того ж джерела, було перевірено другу аудіодоріжку, відповідний приклад наведено на рисунку 3.60. Аналогічно, через тривалість 5 секунд аудіо було проаналізовано як 1 сегмент (вікно 5 с), а підсумкова оцінка залишилась стабільно низькою, P(fake)mean =

1,5% (також P(fake)max = 1,5%), що менше за поріг 60% отже система класифікує голос як «правдоподібний» і повертає вердикт «реальне» для другої доріжки.

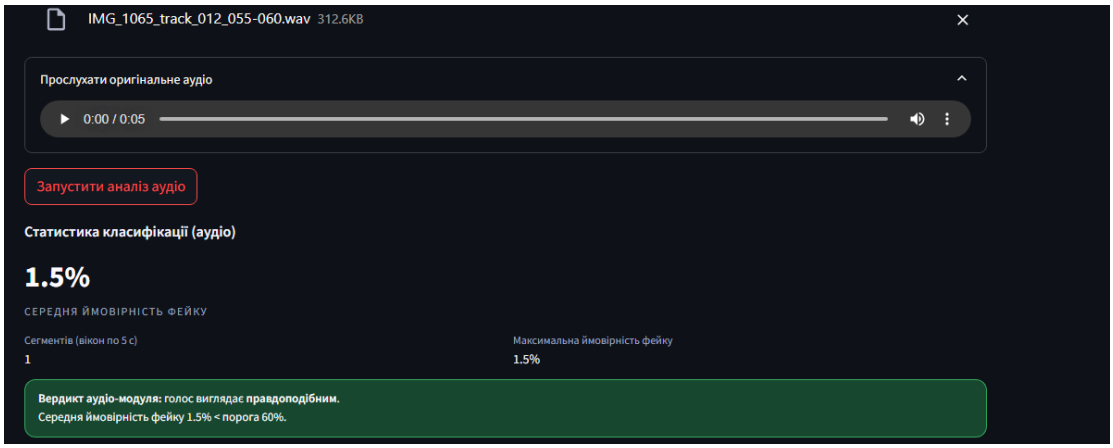


Рисунок 3.60 – Фрагмент результату аудіоаналізу реального голосу P(fake)mean = 1,5% < 60%

Перший приклад демонструє ситуацію, коли і відео-, і аудіокомпонента є синтетичними, система виводить підсумкову оцінку 85,7% як максимальне значення між модулями, при цьому для відео отримано середню P(fake) = 68,7% (301 кадр), а для аудіо P(fake) = 68,4% (3 сегменти). За правилом SAFE_OR фінальний вердикт формується як «імовірний діпфейк», оскільки обидва модулі перевищують свої пороги (відео 45%, аудіо 60%). Приклад такого результату наведено на рисунку 3.61.

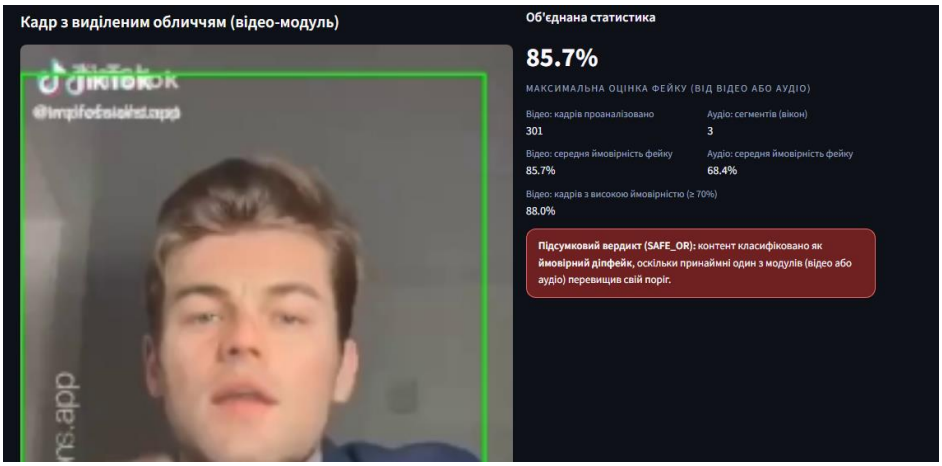


Рисунок 3.61 – Фрагмент результату мультимодального аналізу, фейкове відео та фейкове аудіо, підсумковий вердикт SAFE_OR «ймовірний діпфейк»

Другий приклад ілюструє комбінований випадок «реальне аудіо та фейкове відео», відеомодуль дає P(fake) = 58,3% (188 кадрів), що перевищує поріг 45%, тоді як аудіомодуль показує P(fake) = 41,6% (1 сегмент), тобто нижче 60%. Попри «правдоподібне» аудіо, SAFE_OR класифікує контент як «імовірний діпфейк» через спрацювання відеомодуля у зведеній статистиці відображається максимальна оцінка 58,3%. Цей результат показано на рисунку 3.62.

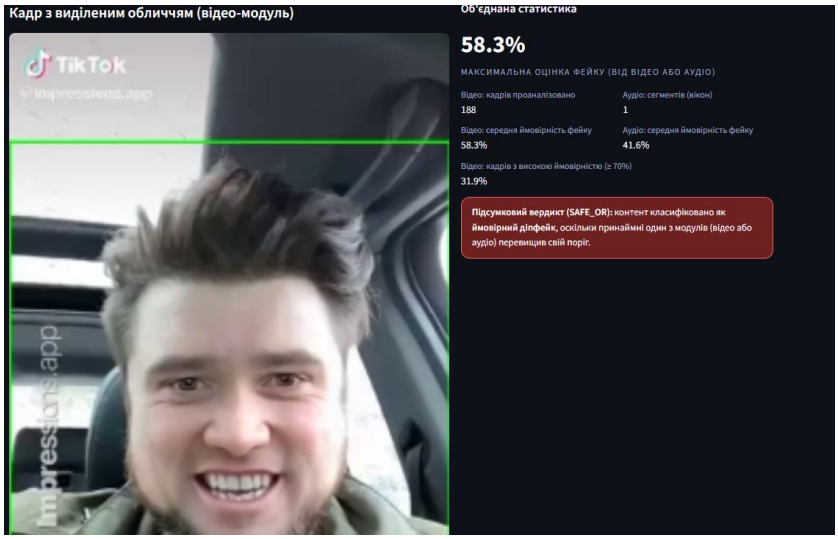


Рисунок 3.62 – Фрагмент результату мультимодального аналізу, фейкове відео та реальне аудіо, вердикт SAFE_OR «імовірний діпфейк» за рахунок відеомодуля

Третій приклад відображає протилежну ситуацію «фейкове аудіо та реальне відео», відеомодуль формує низьку середню ймовірність P(fake) = 12,5% (297 кадрів), тобто нижче порога 45%, однак аудіомодуль дає P(fake) = 74,5% (1 сегмент), що перевищує поріг 60%. У підсумку SAFE_OR повертає «імовірний діпфейк» через спрацювання аудіомодуля, а зведена оцінка дорівнює 74,5% як максимум між модальностями. Приклад наведено на рисунку 3.63.

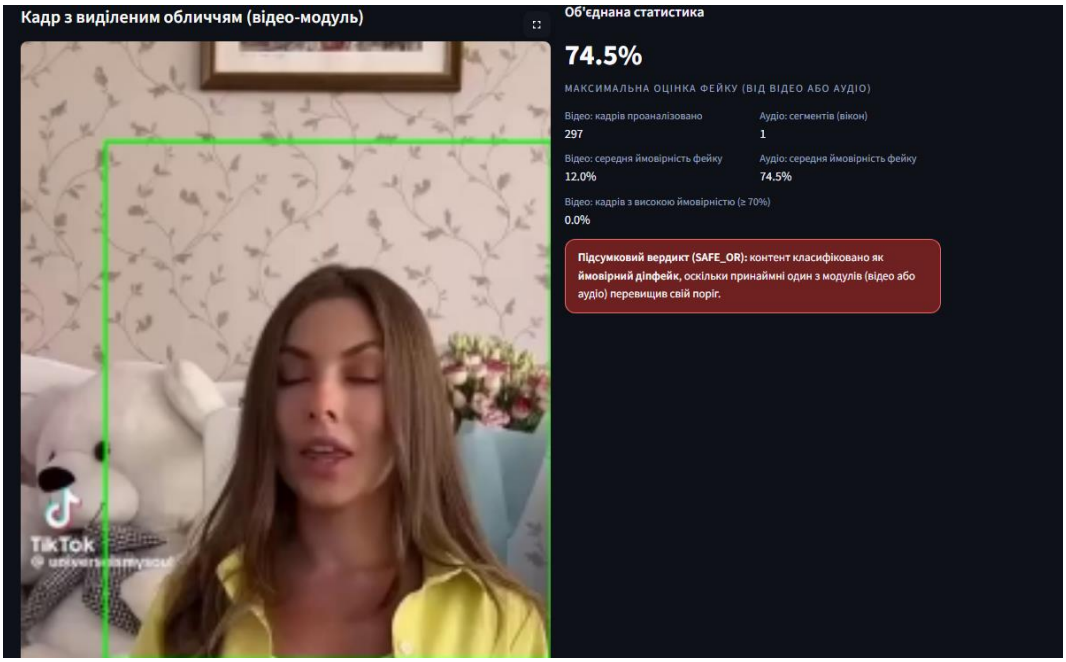


Рисунок 3.63 – Фрагмент результату мультимодального аналізу, реальне відео та фейкове аудіо, вердикт SAFE_OR «імовірний діпфейк» за рахунок аудіомодуля

Четвертий приклад демонструє випадок, коли обидві модальності є реальними, для відео отримано P(fake) = 20,0% (301 кадр), а для аудіо P(fake) = 6,2% (13 сегментів), тобто обидва значення нижчі за свої пороги (45% і 60%). За SAFE_OR фінальний вердикт відображається як «правдоподібний», а зведена оцінка становить 20,0% (максимум між відео та аудіо). Приклад такого результату подано на рисунку 3.64.

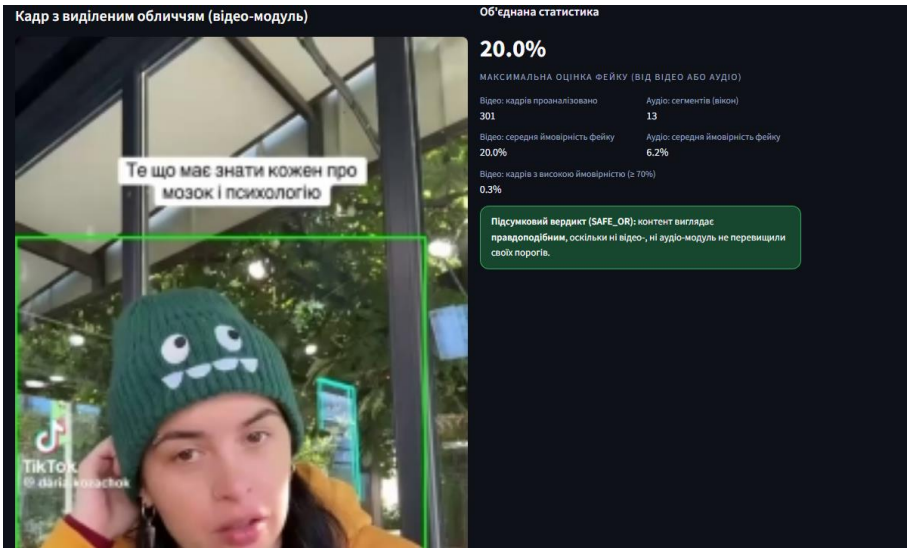


Рисунок 3.64 – Фрагмент результату мультимодального аналізу, реальне відео та реальне аудіо, підсумковий вердикт SAFE_OR «правдоподібний контент»

Для відеомодуля виконано підбір оптимального порогу класифікації на валідаційній вибірці (VAL) та обчислено основні метрики якості і матрицю помилок. Результат підбору порогу і підсумкові метрики наведено на рис. 3.65.

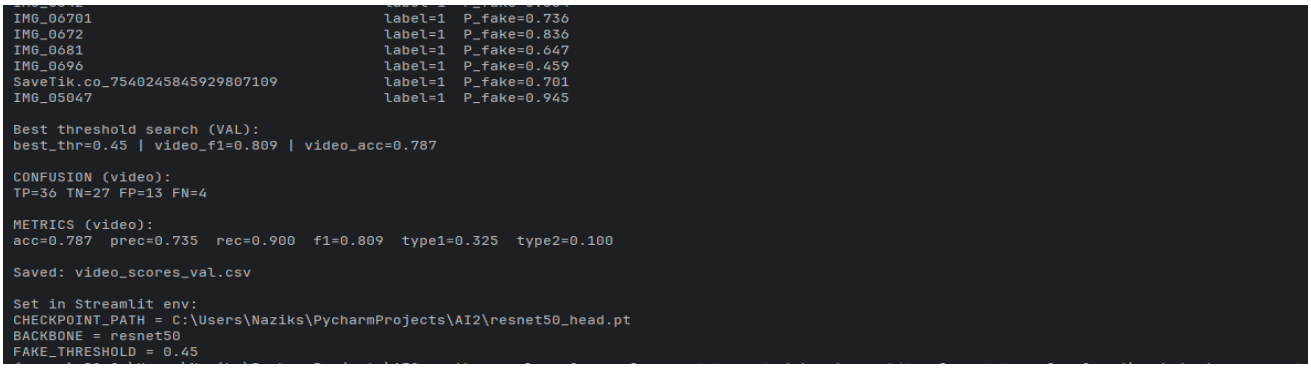


Рисунок 3.64 – Фрагмент результату пошуку оптимального порогу (best_thr) та метрики відеомодуля (ResNet50) на валідаційній вибірці (VAL)

Як видно з рис. 3.64, обраний поріг best_thr використовується для перетворення ймовірності P_fake у бінарне рішення (fake або real), після чого розраховуються acc, prec, rec, f1 та значення TP, TN, FP, FN. Аналогічно для аудіомодуля виконано підбір порогу та оцінку якості на VAL. Вихідні результати для аудіо наведено на рис. 3.65.

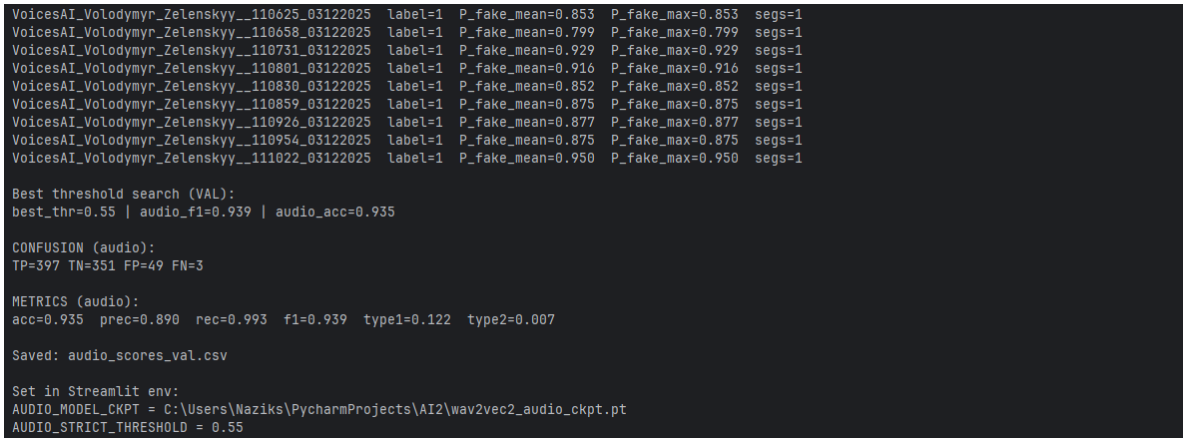


Рисунок 3.65 – Фрагмент результату пошуку оптимального порогу (best_thr) та метрики аудіомодуля (Wav2Vec) на валідаційній вибірці (VAL)

Для зручності подальшого використання у веб-застосунку Streamlit ключові параметри (best_thr) та узагальнені метрики для відео- й аудіомодуля зведено у табл.

3.1. У таблиці також зазначено назви збережених CSV-файлів із результатами оцінювання.

Таблиця 3.1 – Підібрані пороги класифікації та основні метрики якості для відео- й аудіомодуля на валідаційній вибірці (VAL)

Модуль	Чекпойнт	best_thr	acc	prec	rec	f1	type1	type2	CSV
Video (ResNet50)	resnet50_head.pt	0.45	0.7	0.73	0.90	0.80	0.32	0.10	video_scores_val.csv
Audio (Wav2Vec)	wav2vec2_audio_ckpt.pt	0.55	0.93	0.890	0.99	0.93	0.122	0.007	audio_scores_val.csv

Оскільки загальні метрики можуть приховувати структуру помилок класифікації, додатково наведено матрицю помилок, яка показує кількість істинно позитивних, істинно негативних, хибнопозитивних і хибнонегативних рішень. Відповідні значення подано у табл. 3.2.

Таблиця 3.2 – Матриця помилок (TP, TN, FP, FN) для відео- та аудіомодуля на валідаційній вибірці (VAL)

Модуль	TP	TN	FP	FN
Video (ResNet50)	36	27	13	4
Audio (Wav2Vec)	397	351	49	3

Під час апробації системи виявлено негативний кейс хибнопозитивного спрацювання відеомодуля. Відео, яке в межах дослідження розглядалося як реальне, було класифіковано системою як ймовірний діпфейк. Результат зображено на рисунку 3.66.

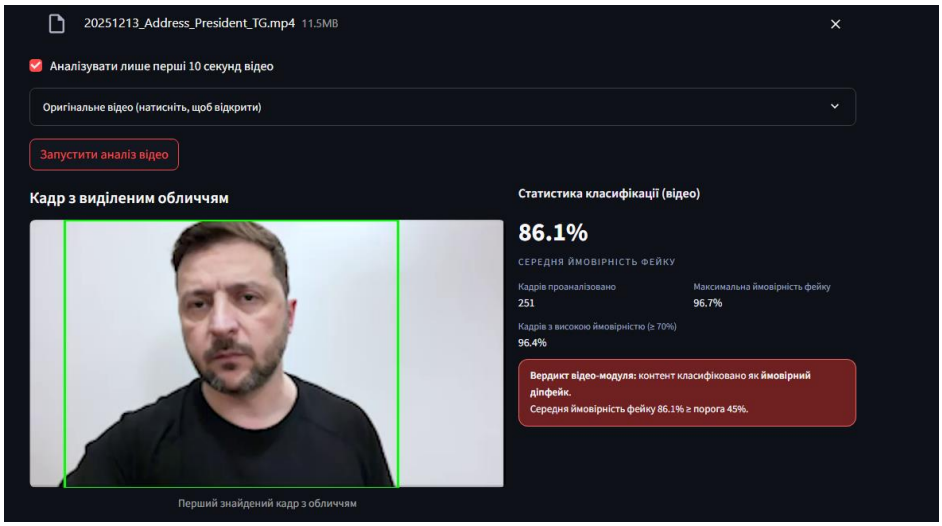


Рисунок 3.66 – Фрагмент результату дослідження реального відео

На екрані результатів видно, що модель сформувала високу оцінку ймовірності фейку та видало попередження, оскільки отримане значення перевищило встановлений поріг прийняття рішення. Цей випадок демонструє, що навіть для реального контенту можливі помилки класифікації, особливо за наявності відмінностей між умовами, на яких модель навчалася, і реальними умовами використання.

Ймовірні причини такого результату пов’язані не з самим фактом підробки, а з особливостями відеопотоку та доменним зсувом. Платформи й месенджери часто автоматично перекодовують відео, знижують бітрейт, змінюють різкість і додають компресійні артефакти, зокрема по контурах обличчя та на межах об’єктів. Для моделі такі артефакти можуть бути схожими на ознаки синтетичного походження, якщо в навчальних даних вони частіше траплялися у класі фейків або якщо набір даних загалом був недостатньо різноманітним. Додатковим фактором може бути нестабільність виділення обличчя в кадрах, коли рамка охоплює зайвий фон, частково обрізає обличчя або фіксує невдалий ракурс. У такому випадку класифікатор отримує менш інформативний вхід і може системно завищувати оцінку фейковості.

Для зменшення ймовірності хибнопозитивних спрацювань доцільно наблизити навчальні дані та умови інференсу до реального сценарію використання. Практичним підходом є доповнення класу реальних прикладів відео з тих самих джерел і з тією самою постобробкою, з якою система працює на практиці, зокрема з відео після

повторного стискання платформою. Важливо також збалансувати якість та рівень компресії між реальними і фейковими прикладами, щоб модель не використовувала якість відео як непрямий маркер класу. Додатково варто підсилити аугментації під типові артефакти перекодування, розмиття, шум і втрати деталей, щоб зробити модель менш чутливою до технічних спотворень. Окрему увагу потрібно приділити повній узгодженості препроцесингу між навчанням і застосунком, оскільки відмінності в нормалізації або кадруванні здатні зміщувати ймовірності. Також доцільно перекалібрувати поріг рішення на окремій вибірці реальних відео з цільових джерел, щоб зменшити ризик помилок саме в робочому домені. Якщо система використовує два канали, відео і аудіо, корисним є узгодження рішень між модулями, коли фінальний вердикт знижується за впевненістю у випадках, коли один канал демонструє високу ймовірність фейку, а інший її не підтверджує.

У межах експериментів додатково оцінено обчислювальні витрати навчання моделей і швидкодію під час аналізу файлів у застосунку. Для цього зафіксовано консольні журнали навчання моделей та емпіричний час інференсу на коротких фрагментах.

На рис. 3.67 наведено фрагмент журналу навчання відеомодуля (класифікаційна голова ResNet50), де відображено прогрес по епохах і час виконання кожної епохи. Підсумкова тривалість навчання відеомодуля за десять епох становила близько 6 год 57 хв 08 с.

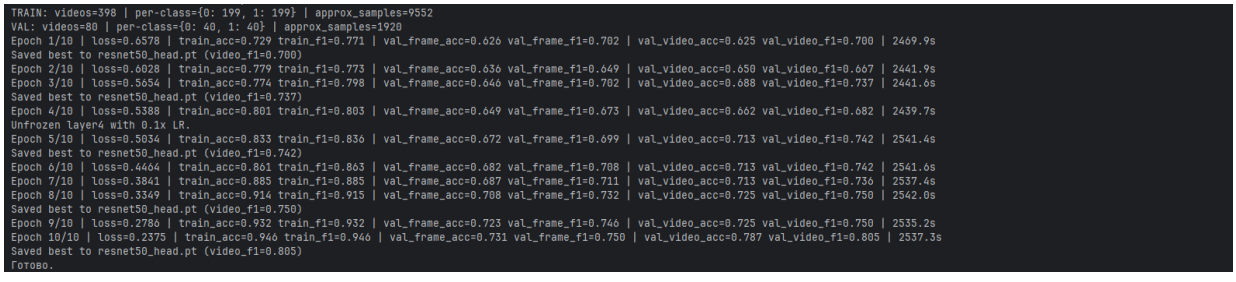


Рисунок 3.67 – Фрагмент журналу навчання відеомодуля

Після завершення навчання відеомодуля аналогічно було виконано навчання аудіомодуля на базі Wav2Vec2 із замороженим енкодером і донавчанням

класифікаційної голови. На рис. 3.68 наведено фрагмент журналу навчання аудіомодуля, де відображено метрики валідації та час виконання кожної епохи. Загальна тривалість навчання аудіомодуля за п’ять епох становила близько 4 год 42 хв 10 с.

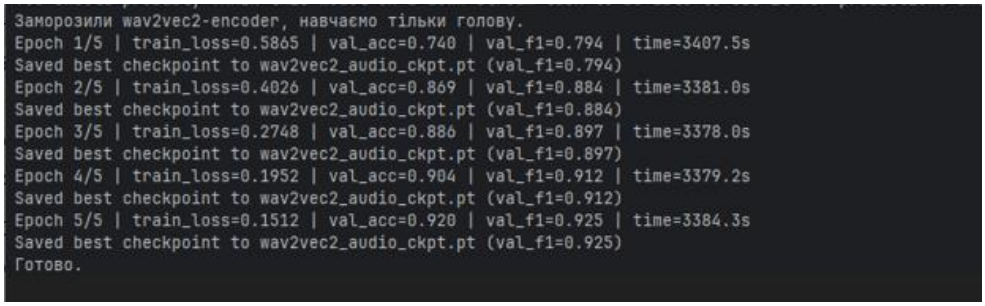


Рисунок 3.68 – Фрагмент журналу навчання аудіомодуля

Доцільно показати практичну швидкодію системи на етапі інференсу в застосунку. Експериментальне оцінювання швидкодії інференсу та тривалості навчання виконувалося на персональному комп’ютері з такими характеристиками: процесор AMD Ryzen 5 5600X (6 ядер), відеокарта NVIDIA GeForce RTX 5060 Ti, оперативна пам’ять 32 ГБ DDR4. У табл. 3.3 наведено час аналізу для відео та аудіо окремо, а також сумарний час для послідовного режиму «Відео + Аудіо», оскільки спочатку виконується відеодетекція, а далі аудіодетекція того самого файлу.

Таблиця 3.3 - Швидкодія детекції для відео, аудіо та режиму агрегації

Режим аналізу	Час для 10 с контенту	Оцінка часу для 3 хв контенту
Лише відео	31 с	Близько 19 хв 20 с
Лише аудіо	8 с	Близько 2 хв 24 с
Агрегація	39 с	Близько 21 хв 15 с

У межах порівняння з аналогами було виконано тестування на фейковому відеофрагменті. Результат роботи розробленої системи наведено на рисунку 3.69.

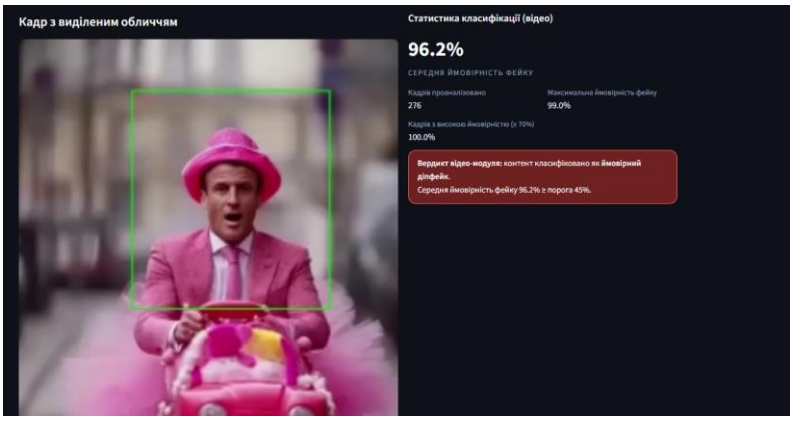


Рисунок 3.69 – Фрагмент результату аналізу фейкового відео у розробленому веб застосунку

Для зіставлення наведено результат перевірки того самого фейкового відео в сторонньому аналозі Deepware Scanner [46], що показано на рисунку 3.70.

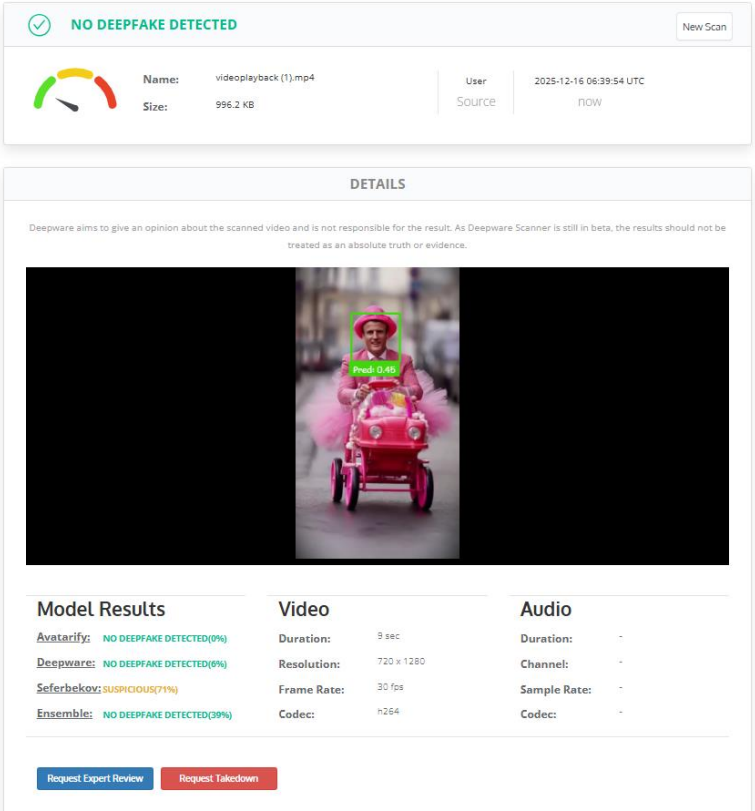


Рисунок 3.70 – Фрагмент результату аналізу фейкового відео в Deepware Scanner

За результатами порівняння видно, що розроблена система коректно ідентифікувала подані тестові зразки як фейкові, тоді як аналог у цьому кейсі не виявив

діпфейк. Це демонструє вищу чутливість і практичну ефективність реалізованого підходу для обраного типу маніпуляцій у рамках проведеного експерименту.

Було виконано порівняння результатів стороннього сервісу Hive AI's Deepfake Detection та розробленої системи на одному й тому самому мультимедійному файлі. Метою є показати, як відрізняється реакція систем на підробку в окремих модальностях і як це впливає на підсумковий вердикт.

За результатами Hive AI's Deepfake Detection (рисунок 3.71) відеокомпонент був позначений як імовірно згенерований/діпфейк, тоді як аудіокомпонент отримав оцінку «Not AI-Generated» із нульовим рівнем підозри [47]. Таким чином, сервіс фактично сформував суперечливий профіль, відео визначено як підробку, але аудіо як автентичне. У межах даного прикладу це є помилкою по аудіомодальності, оскільки у відео наявна підроблена звукова доріжка, тобто фактично має бути «fake» і для аудіо.

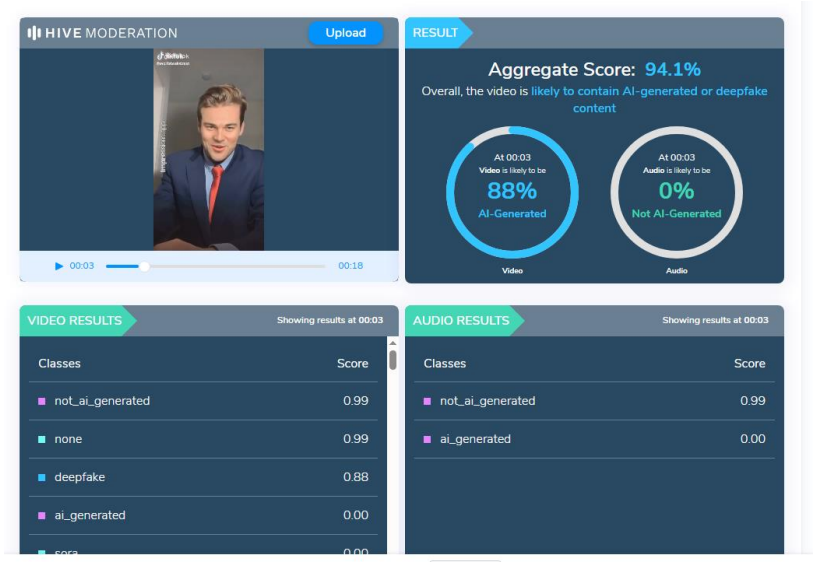


Рисунок 3.70 – Фрагмент результату аналізу фейкового відео та аудіо в Hive AI's Deepfake Detection

На відміну від цього, розроблена система (рисунок 3.71) коректно ідентифікувала підробку в обох модальностях. Відеомодуль зафіксував ознаки діпфейку у відеоряді, а аудіомодуль визначив звукову доріжку як синтетичну/підроблену. Підсумковий мультимодальний вердикт був сформований за правилом SAFE_OR, що означає, якщо хоча б один модуль перевищує поріг, файл позначається як імовірний діпфейк. У цьому

прикладі обидва модулі підтвердили підробку, тому інтегрований результат є однозначним та узгодженим із фактичним змістом файла.

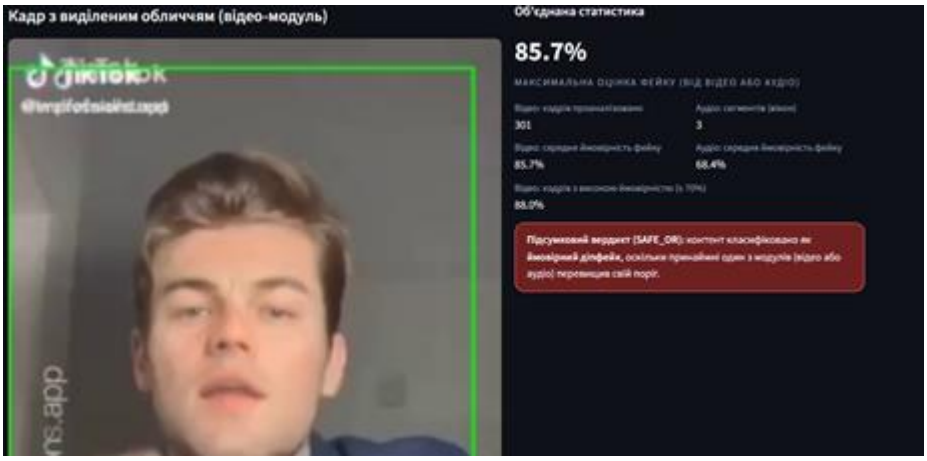


Рисунок 3.71 – Фрагмент результату аналізу фейкового відео та аудіо в розробленій системі

У межах експериментального порівняння додатково зіставлено результати детекції для одного й того самого відеофайла у системі, реалізованій у бакалаврській дипломній роботі (БДР), та в удосконаленій системі магістерської кваліфікаційної роботи (МКР). Метою такого порівняння є показати, як зміна підходу до обробки кадрів і детекції обличчя впливає на підсумковий вердикт.

За результатами системи БДР детекція виконувалась некоректно через проблему з виділенням обличчя на відео, зображено на рисунку 3.72.

Рисунок 3.72 – Фрагмент обробки відео системи БДР

Це призвело до того, що в аналіз потрапляли нерелевантні фрагменти кадру або взагалі не формувалася репрезентативна вибірка обличчя для класифікатора. У підсумку система БДР зафіксувала нульову кількість «fake frames» та середню імовірність фейку, близьку до нуля, і видала висновок, що відео є реальним, зображено на рисунку 3.73. Таким чином, помилка на етапі локалізації обличчя спричинила хибнонегативний результат на рівні всього файла.

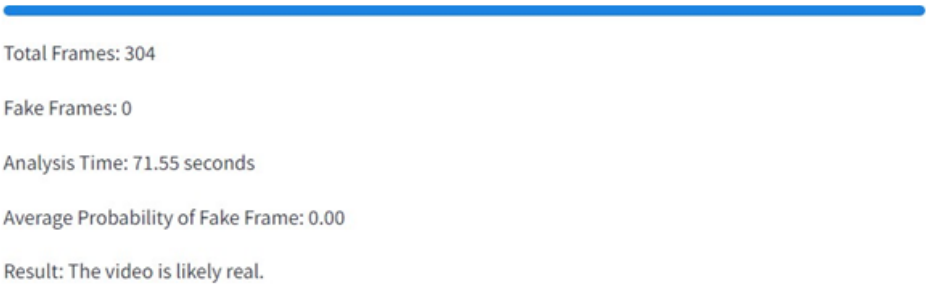


Рисунок 3.73 – Фрагмент результат відеодетекції системи БДР

На відміну від цього, система МКР продемонструвала коректну роботу етапу виділення обличчя та формування набору кадрів для аналізу, що підтверджується наявністю кадру з правильною локалізацією обличчя та достатньою кількістю проаналізованих кадрів, зображено на рисунку 3.74.

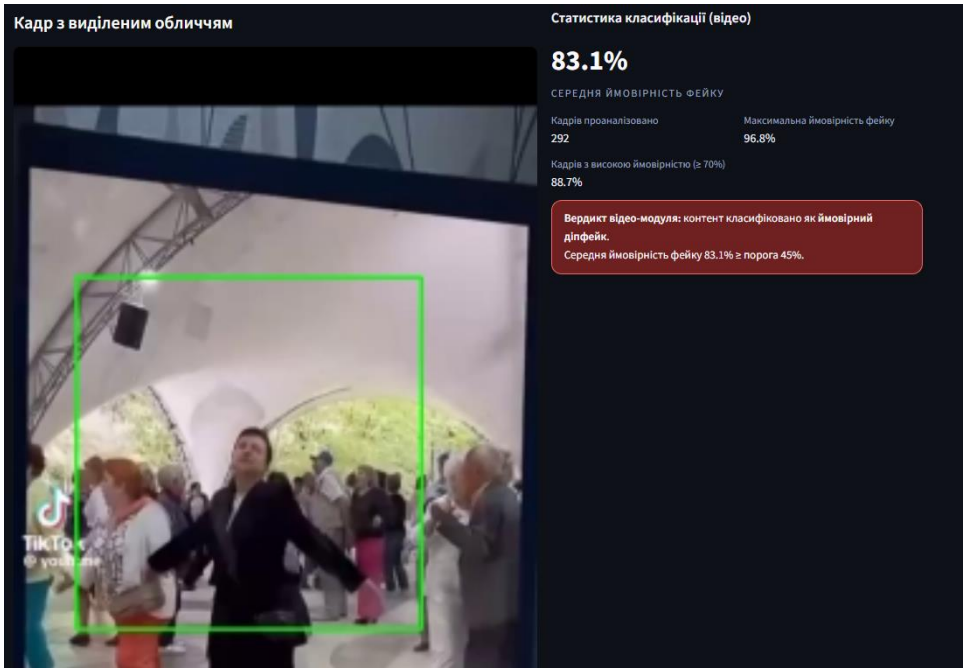


Рисунок 3.73 – Фрагмент результат відеодетекції системи МКР

За результатами відеомодуля МКР отримано високу середню імовірність фейку (83.1%) та максимальну імовірність (96.8%), а також сформовано вердикт відеомодуля про імовірний діпфейк із порівнянням із порогом (83.1% ≥ 45%). Отже, МКР забезпечила коректне виявлення підробки в ситуації, де попередня реалізація БДР помилково класифікувала контент як реальний.

Отримані результати підтверджують, що ключовим фактором підвищення точності у порівнюваному прикладі є стабільність та коректність етапу детекції/локалізації обличчя перед класифікацією. У БДР помилка на ранньому етапі призвела до “каскадного” ефекту й заниження оцінок P_fake, тоді як у МКР правильна локалізація забезпечила інформативні вхідні дані для моделі та, відповідно, правильний підсумковий вердикт.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на впровадження лише за умови відповідності вимогам науково-технічного прогресу та економічної доцільності. Саме тому результати дослідження повинні бути оцінені з позицій їх ефективності та потенціалу впровадження у практичну діяльність користувачів у сфері кібербезпеки.

Магістерська кваліфікаційна робота на тему «Система виявлення фейкового мультимедійного контенту» належить до науково-технічних розробок прикладного характеру, орієнтованих на подальше впровадження в підрозділах кібербезпеки, цифрової експертизи та медіамоніторингу. Для обґрунтування доцільності її комерціалізації необхідно оцінити науково-технічний рівень розробки, ступінь інноваційності та потенційні можливості виходу на ринок рішень для виявлення діпфейків.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

- проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;
- розраховано витрати на здійснення науково-технічної розробки;
- розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором;
- проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту є комплексна оцінка розробленої системи за сукупністю технічних та ринкових характеристик, що дозволяє зробити висновок про її конкурентоспроможність і перспективи подальших інвестицій у розвиток та промислове впровадження. Методичні підходи до такого аудиту

відповідають рекомендаціям, наведеним у методичних вказівках до економічної частини магістерських робіт.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші,	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі,
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної	Ринок малий, але має позитивну	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою

Продовження таблиці 4.1

7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуют ь ся у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуют ься у
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	3	2
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	4	4	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	4
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	2	2	3
8. Практична здійсненність (наявність фахівців)	2	3	2
9. Практична здійсненність (наявність фінансів)	2	2	1
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	38	39	35
Середньоарифметична сума балів $СБ_c$			37,33

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [48].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно з проведеними розрахунками середньоарифметична сума балів за результатами експертного оцінювання науково-технічного рівня та комерційного потенціалу розробки «Система виявлення мультимедійних фейків» становить 37,33 бала. Відповідно до шкали, наведеної в таблиці 4.3, це значення потрапляє до інтервалу 31…40 балів, що дає підстави віднести розробку до рівня з комерційним потенціалом вище середнього і свідчить про доцільність подальшого впровадження результатів магістерської роботи в практику.

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [48]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i , \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *і-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *і-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{нi}}{I_{ai}} , \tag{4.2}$$

де $I_{ні}$ та $I_{на}$ – чисельні значення конкретного *i*-го технічного показника якості відповідно для нової розробки та аналога;

для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}}; \tag{4.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований програмний засіб	Відношення параметрів нової розробки до аналога	Питома вага показника
Точність виявлення діпфейкових відео	%	85	91	1,07	0,25
Середній час аналізу 10-секундного відеофрагмента	с	12	8	1,50	0,20
Кількість підтримуваних типів вхідних даних (відео / аудіо / відео+аудіо)	од.	1	3	3,00	0,15
Частка коректних рішень на зашумлених або стиснених даних	%	80	90	1,12	0,20
Мінімальний обсяг оперативної пам’яті для запуску аналізу на CPU-пристрої	ГБ	8	4	2,00	0,20

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 0{,}25 \cdot 1{,}07 + 0{,}20 \cdot 1{,}50 + 0{,}15 \cdot 3{,}00 + 0{,}20 \cdot 1{,}12 + 0{,}20 \cdot 2{,}00 = 1{,}64.$$

Отже, за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,64 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи на тему «Система виявлення фейкового мультимедійного контенту», під час планування, обліку і калькулювання собівартості науково- дослідної роботи групуємо за відповідними статтями

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, робітникам, студентам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [48]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник науково-дослідної роботи (проєктний менеджер системи детекції діпфейків)	40 000,00	1 818,18	10	18 181,80

Інженер-дослідник з кібербезпеки (розробник архітектури мультимодального детектора діпфейків)	35 000,00	1 590,91	18	28 636,38
Інженер-програміст (розробник моделей глибинного навчання для аналізу відео та аудіо)	32 000,00	1 454,55	20	29 091,00
Інженер з тестування програмного забезпечення (QA- інженер системи детекції діпфейків)	28 000,00	1 272,73	16	20 363,68
Всього				96 272,86

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Система виявлення фейкового мультимедійного контенту» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \tag{4.5}$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \tag{4.6}$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [48];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 8000{,}00 \cdot 1{,}20 \cdot 1{,}15 / (22 \cdot 8) = 62{,}73$ *грн.*

$З_{р1} = 62{,}73 \cdot 6{,}00 = 376{,}38$ *грн.*

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість, год	Розряд	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Витрати, грн
Підготовка робочого місця та інфраструктури для обробки мультимедійних даних	5,0	2	1,20	70,57	352,84
Інсталяція та налаштування середовища Python, бібліотек PyTorch, Librosa, Streamlit	8,0	3	1,35	79,39	635,12
Збір і попередня обробка відеоданих (обрізання, конвертація, кадрування)	12,0	3	1,35	79,39	952,68
Формування і валідація відеодатасету (розподіл на train/val/test, перевірка якості)	10,0	3	1,35	79,39	793,90
Підготовка та розмітка аудіодатасету (розбиття на вікна, нормалізація, маркування real/fake)	12,0	3	1,35	79,39	952,68
Навчання та відлагодження відеомоделі ResNet50 на підготовленому відеодатасеті	15,0	4	1,50	88,21	1323,15
Навчання та відлагодження аудіомоделі Wav2Vec2.0 на аудіодатасеті	15,0	4	1,50	88,21	1323,15
Інтеграція відео- та аудіомодулів у Streamlit-додаток, проведення функціонального тестування	12,0	4	1,50	88,21	1058,52
Всього					7392,00

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = *(*96272,86 + 7392,00*) · 11 / 100%* = 11403,13 *грн.*

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

*Зн = (*96272,86 + 7392,00 + 11403,13*) · 22 / 100% = 25314,96 грн.*

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Система виявлення фейкового мультимедійного контенту».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}, \quad (4.9)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1 = 2,000 \cdot 185,00 \cdot 1,05 - 0 \cdot 0 = 388,50$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, од.	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Багатофункціональний офісний папір А4 (80 г/м², 500 арк./пачка)	190,00	2,000	0	0	380,00
Папір для друку звітів А4 підвищеної якості	220,00	1,000	0	0	220,00
Набір маркерів для дошки (4 кольори)	160,00	1,000	0	0	160,00
USB-флеш-накопичувач 128 ГБ для перенесення датасету	450,00	1,200	0	0	450,00
Зовнішній жорсткий диск 1 ТБ для резервного зберігання мультимедійних даних	2 600,00	1,000	0	0	2 600,00
Тонер-картридж для лазерного принтера	1 250,00	1,000	0	0	1 250,00
Всього					5 060,00

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Система виявлення фейкового мультимедійного контенту», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де $Н_j$ – кількість комплектуючих *j*-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих *j*-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*).

При цьому до комплектуючих віднесено елементи, необхідні для організації експериментального стенду мультимодального детектора діпфейків, графічний прискорювач для навчання та інференсу нейронних мереж, високошвидкісний твердотільний накопичувач для зберігання відео- та аудіодатасетів, а також периферійні пристрої (веб-камера, мікрофон, USB-хаб) для запису власних тестових матеріалів і гнучкого підключення обладнання. Проведені розрахунки зведено до таблиці 4.8.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Графічний прискорювач NVIDIA RTX-серії для навчання та інференсу моделей детекції діпфейків	1	18 000,00	18 000,00
Твердотільний накопичувач NVMe SSD 2 ТБ для зберігання відео- та аудіодатасетів	1	4 200,00	4 200,00
Веб-камера Full HD для запису тестових відеофрагментів	1	1800,00	1800,00
USB-мікрофон студійного типу для запису голосових доріжок	1	2500,00	2500,00
USB-хаб (7-портовий) для підключення периферійних пристроїв		700,00	700,00
Всього			27 200,00

4.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» відносять витрати на придбання спеціального наукового обладнання, а також витрати на його розробку, транспортування, монтаж і налаштування. Таке обладнання повинне бути необхідним саме для проведення експериментальних досліджень за обраною тематикою.

Спецустаткування спеціального призначення не використовується, і витрати за даною статтею не передбачаються**.**

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для

проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.11)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 450,00 \cdot 1 \cdot 1,05 = 472,50$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт.	Ціна за одиницю, грн	Коеф. KiK_iKi	Вартість, грн
Ліцензія ОС Windows 11 Pro для робочої станції розробника	1	450,00	1,05	472,50
Ліцензія на професійне середовище розробки (PyCharm Professional / аналогічний IDE)	1	7,000	1,00	7 000,00
Пакет офісних програм для оформлення звітної документації (MS Office / аналог)	1	4 500,00	1,00	4 500,00
Річна підписка на хмарний сервіс для навчання моделей з використанням GPU (обчислювальний клауд)	1	2 000,00	1,05	2 100,00
Підписка на хмарне сховище для зберігання відео- та аудіодатасету (≈2 ТБ)	1	3 000,00	1,05	3 150,00
Антивірусний комплекс / засіб захисту робочої станції	1	800,00	1,00	800,00
Всього				18 022,50

4.3.7 Амортизація обладнання, програмних засобів та приміщень

Амортизаційні відрахування по кожному виду обладнання, програмного забезпечення та приміщень, які використовувалися під час досліджень, розраховуємо за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.12)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (80\,000{,}00 \cdot 1) / (4 \cdot 12) = 1\,666{,}67$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання / ПЗ / інфраструктури	Балансова вартість, грн	Строк корисного використання, років	Термін використання під час НДР, міс	Амортизаційні відрахування, грн
Робоча станція інженера-розробника (навчання та інференс моделей відео/аудіо)	80 000,00	4	1	1 666,67
Ноутбук для тестування та демонстрації роботи системи	38 000,00	3	1	1 055,56
Зовнішній SSD 2TB для зберігання датасету та результатів експериментів	4 500,00	3	1	125,00
Маршрутизатор/фаєрвол для ізольованого тестового сегмента (мережеві експерименти)	9 000,00	4	1	187,50
Комутатор для лабораторного сегмента (1GbE/2.5GbE)	3 000,00	5	1	50,00

Моніторингова периферія (монітор, клавіатура, UPS)	15 000,00	5	1	250,00
Аудіообладнання для збору реальних зразків (мікрофон/аудіоінтерфейс)	6 000,00	5	1	100,00
Приміщення/робоче місце для проведення експериментів (орендована площа)	300 000,00	30	1	833,33
Всього				4 268,06

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.13)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Ц$_e$ = 12,56 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,38 \cdot 172,0 \cdot 12,56 \cdot 0,95 / 0,97 =$ 804,00 *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.11 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість роботи, год	$K_{вп i}$	η_i	Сума, грн
Робоча станція інженера-розробника (навчання/інференс ResNet50, Streamlit)	0,38	172,0	0,95	0,97	804,00
Ноутбук для тестування та демонстрації системи	0,10	172,0	0,90	0,97	200,44
Зовнішній SSD/накопичувач для зберігання відео та аудіодатасету	0,01	172,0	0,70	0,98	15,43
Комутатор 1GbE для лабораторного сегмента	0,03	172,0	0,95	0,98	62,83
Маршрутизатор/мережевий шлюз (доступ до інтернет-ресурсів)	0,2	172,0	0,95	0,98	41,88
Моніторингова периферія (монітор + UPS + клавіатура/миша)	0,12	172,0	0,90	0,97	253,89
Аудіоінтерфейс/мікрофон для запису та перевірки аудіоданих	0,01	40,0	0,85	0,95	4,50
Принтер для друку матеріалів та звітів	0,25	4.0	0,90	0,95	11,90
Всього					1 394,87

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Система виявлення фейкового мультимедійного контенту» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, залучених до виконання дослідження, відрядження, пов’язані з проведенням випробувань програмно-апаратних засобів, а також поїздки на наукові з’їзди, конференції, семінари та наради, безпосередньо пов’язані з виконанням даної роботи.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.14)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження»,

приймемо $Н_{св} = 0\%$.

*$B_{св}$ = (*96 272,86+ 7 392,00*) · 0 / 100% = 0,00 грн.*

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.15)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = *(*96 272,86+ 7 392,00*) · 30 / 100%* = 31 099,46 *грн.*

4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.16)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в$ = *(*96 272,86+ 7 392,00*) · 50 / 100%* = 51 832,43 *грн.*

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.17)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$B_{нзв}$ = (96 272,86 + 7 392,00) · 100 / 100% = 103 664,86 *грн.*

Витрати на проведення науково-дослідної роботи на тему «Система виявлення фейкового мультимедійного контенту» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + А_{обл} + B_е + B_{св} + B_{сп} + І_в + B_{нзв} \quad (4.18)$$

$B_{заг}$= 103 664,86 + 11 403,13 + 25 314,96 + 31 099,46 + 51 832,43 + 103 664,86 = 326 979,70 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 326 979,70 / 0,9 = 363 310,78 *грн.*

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

У ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої

науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Система виявлення фейкового мультимедійного контенту» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	300	700	900	600

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 200000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 5000,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 1000,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [49]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.20)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 40\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (1000 \cdot 20000 + 6000 \cdot 300) \cdot 0{,}8333 \cdot 0{,}4 \cdot (1 - 0{,}18) \approx 5958428{,}32 \text{ грн.}$$

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (1000 \cdot 20000 + 6000 \cdot 700) \cdot 0{,}8333 \cdot 0{,}4 \cdot (1 - 0{,}18) \approx 6614402{,}08 \text{ грн.}$$

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (1000 \cdot 20000 + 6000 \cdot 900) \cdot 0{,}8333 \cdot 0{,}4 \cdot (1 - 0{,}18) \approx 6942388{,}96 \text{ грн.}$$

Збільшення чистого прибутку 4-го року:

$$\Delta П_4 = (1000 \cdot 20000 + 6000 \cdot 600) \cdot 0{,}8333 \cdot 0{,}4 \cdot (1 - 0{,}18) \approx 6450408{,}64 \text{ грн.}$$

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{4.21}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}12$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

ПП ≈ 5320025,29 + 5272960,84 + 4941455,33 + 4099351,30 = 19633792,77 грн

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.22}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=1,5;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 363 310,78 грн.

$PV = k_{інв} \cdot ЗВ$ = 1,5 · 363 310,78 = 544 966,17 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.23}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 19 633 792,77 грн;

PV – теперішня вартість початкових інвестицій, 544 966,17 грн.

$E_{абс} = ПП - PV$ = 19 633 792,77 − 544 966,17 = 19 088 826,60 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.24}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 19 088 826,60 грн;

PV – теперішня вартість початкових інвестицій, 544 966,17 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_{в} = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+19\ 088\ 826{,}60 / 544\ 966{,}17)^{1/4} = 2{,}45.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.25}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,3.

$\tau_{мін}$ = 0,11+0,3 = 0,41 < 2,45 свідчить про те, що внутрішня економічна дохідність інвестицій $E_{в}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Система виявлення фейкового мультимедійного контенту» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_{в}}, \tag{4.26}$$

де $E_{в}$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 2,45 = 0,40 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

На підставі проведених розрахунків встановлено, що рівень комерційного потенціалу науково-технічної розробки за темою «Система виявлення фейкового мультимедійного контенту» є високим, що підтверджує доцільність її подальшого впровадження та виходу на ринок кібербезпекових рішень.

За результатами оцінювання за техніко-економічними параметрами розроблений програмний засіб має конкурентні переваги над одномодальними підходами за рахунок:

- мультимодального аналізу відео та аудіо (ResNet50 для відео та Wav2Vec2.0 для аудіо);
- об'єднання рішень за правилом SAFE_OR, що підвищує ймовірність виявлення фейку навіть при маскуванні в одному з каналів;
- зручної реалізації у вигляді Streamlit-застосунку, придатного для запуску на CPU, що спрощує використання та знижує бар'єр впровадження.

Економічні показники комерціалізації проєкту підтверджують його інвестиційну привабливість:

- приведена вартість зростання всіх чистих прибутків інвестора за 4 роки (ПП) становить 19 633 792,77 грн;
- теперішня вартість початкових інвестицій (PV) становить 544 966,17 грн;
- абсолютний економічний ефект становить 19 088 826,60 грн.
- внутрішня економічна дохідність істотно перевищує мінімально допустиму величину, що характеризує низькі інвестиційні ризики
- період окупності інвестицій є дуже малим: приблизно 0,09 року, що значно менше допустимого нормативу 3 роки

Отже, результати техніко-економічних розрахунків доводять, що розроблена система виявлення фейкового мультимедійного контенту має високу економічну ефективність, значний комерційний потенціал та є доцільною для впровадження і практичного застосування у сфері медіа-безпеки та протидії дезінформації

ВИСНОВКИ

У роботі розв’язано поставлене завдання розроблення програмної системи мультимодальної детекції діпфейків у мультимедійних матеріалах шляхом поєднання аналізу відеоряду та звукової доріжки з формуванням підсумкового вердикту на рівні файла. Досягнення мети підтверджується виконанням основних етапів: аналізом підходів до виявлення діпфейків, проєктуванням архітектури, реалізацією прототипу, навчанням моделей та експериментальною оцінкою результатів.

Запропонована система включає модуль попередньої обробки мультимедійного файла, відеомодуль і аудіомодуль, а також модуль інтеграції результатів. Відеомодуль виконує детекцію обличчя на кадрах та оцінює ймовірність підробки за допомогою моделі на базі ResNet50 з донавченою класифікаційною головою. Аудіомодуль здійснює сегментацію аудіодоріжки та класифікацію фрагментів із використанням моделі класу Wav2Vec2. Інтеграція результатів реалізована за правилом SAFE_OR з урахуванням порогів для кожної модальності, що підвищує надійність рішення у випадках, коли одна з модальностей має нижчу інформативність або зазнає спотворень.

Достовірність результатів обґрунтовано експериментальним оцінюванням на валідаційній вибірці з підбором порогів. Для відеомодуля отримано best_thr = 0.45, acc = 0.70, prec = 0.73, rec = 0.90, f1 = 0.80. Для аудіомодуля отримано best_thr = 0.55, acc = 0.93, prec = 0.89, rec = 0.99, f1 = 0.93. Це підтверджує, що аудіокласифікатор забезпечує вищі узагальнені метрики на VAL, тоді як відеомодель має високу повноту, що важливо для мінімізації пропусків підробок.

Практична значущість розробки підтверджується реалізацією повного циклу аналізу у веб-інтерфейсі, від завантаження файла до отримання вердикту, статистик по модальностях і збережених артефактів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Куперштейн Л. М., Людва Н. В., Прокопенко С. О. Система виявлення фейкового мультимедійного контенту: тези доповіді [Електронний ресурс] // Молодь в науці: дослідження, проблеми, перспективи (МН-2026): матеріали Міжнародної науково-практичної інтернет-конференції студентів, аспірантів та молодих науковців (20.10.2025-26.06.2026). Вінниця: Вінницький національний технічний університет, 2026. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26852 (дата звернення: 15.10.2025).
2. Mayer R. E. Multimedia Learning. 3rd ed. Cambridge University Press, 2020 [Електронний ресурс]. URL: https://www.cambridge.org/highereducation/books/multimedia-learning/FB7E79A165D24D47CEACEB4D2C426ECD (дата звернення: 15.10.2025).
3. Vosoughi S., Aral S. The spread of true and false news online. Science. 2018 [Електронний ресурс]. URL: https://pubmed.ncbi.nlm.nih.gov/29590045 (дата звернення: 16.10.2025).
4. Regula. Deepfake Trends 2024. 25.10.2024 [Електронний ресурс]. URL: https://static-content.regulaforensics.com/PDF-files/0831-Regula-Deepfake-Research-Report-Final-version.pdf (дата звернення: 25.10.2025).
5. Xu K. та ін. A Survey of Adversarial Examples in Computer Vision: Attack, Defense and Beyond. Wuhan University Journal of Natural Sciences. 2025 [Електронний ресурс]. URL: https://wujns.edpsciences.org/articles/wujns/full_html/2025/01/wujns-1007-1202-2025-01-0001-20/wujns-1007-1202-2025-01-0001-20.html (дата звернення: 05.11.2025).

6. Financial Times. Arup lost $25mn in Hong Kong deepfake video conference scam. 2024 [Електронний ресурс]. URL: https://www.ft.com/content/b977e8d4-664c-4ae4-8a8e-eb93bdf785ea (дата звернення: 05.11.2025).
7. Google DeepMind. SynthID - watermarking AI-generated content [Електронний ресурс]. URL: https://deepmind.google/technologies/synthid/ (дата звернення: 09.11.2025).
8. Reality Defender. Real-Time Deepfake Detection Platform [Електронний ресурс]. URL: https://www.realitydefender.com/ (дата звернення: 09.11.2025).
9. Hive Moderation. Deepfake Video Detector (API Docs) [Електронний ресурс]. URL: https://docs.thehive.ai/docs/deepfake-detector (дата звернення: 09.11.2025).
10. Microsoft. New technologies to combat disinformation (Video Authenticator). 01.09.2020 [Електронний ресурс]. URL: https://blogs.microsoft.com/on-the-issues/2020/09/01/disinformation-deepfakes-newsguard/ (дата звернення: 09.11.2025).
11. Sensity AI. Deepfake Detection & Threat Intelligence [Електронний ресурс]. URL: https://sensity.ai/deepfake-detection/ (дата звернення: 09.11.2025).
12. ElevenLabs. AI Speech Classifier [Електронний ресурс]. URL: https://elevenlabs.io/news/ai-speech-classifier/ (дата звернення: 09.11.2025).
13. Baltrusaitis T., Ahuja C., Morency L.-P. Multimodal Machine Learning: A Survey and Taxonomy. 2017 [Електронний ресурс]. URL: https://arxiv.org/abs/1705.09406 (дата звернення: 10.11.2025).
14. FFmpeg. ffprobe Documentation [Електронний ресурс]. URL: https://ffmpeg.org/ffprobe.html (дата звернення: 11.11.2025).
15. Міністерство цифрової трансформації України. Про затвердження Вимог до аудиторів інформаційної безпеки. 2023 [Електронний ресурс]. URL: https://zakon.rada.gov.ua/laws/show/z1065-23#Text (дата звернення: 12.11.2025).
16. Адміністрація Держспецзв’язку. Методичні рекомендації щодо реагування суб’єктами забезпечення кібербезпеки на різні види подій у кіберпросторі (наказ

від 03.07.2023 No. 570). 2023 [Електронний ресурс]. URL: https://cip.gov.ua/services/cm/api/attachment/download?id=64762 (дата звернення: 13.11.2025).

17. Baevski A., Zhou Y., Mohamed A., Auli M. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. 2020 [Електронний ресурс]. URL: https://arxiv.org/abs/2006.11477 (дата звернення: 14.11.2025).
18. PyTorch Tutorials. Transfer Learning for Computer Vision Tutorial [Електронний ресурс]. URL: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html (дата звернення: 15.11.2025).
19. timesler. facenet-pytorch: Pretrained PyTorch face recognition and MTCNN face detection. GitHub repository [Електронний ресурс]. URL: https://github.com/timesler/facenet-pytorch (дата звернення: 18.11.2025).
20. Torchvision Documentation. resnet50 (pretrained weights, transforms, ImageNet-1K variants) [Електронний ресурс]. URL: https://docs.pytorch.org/vision/stable/models/generated/torchvision.models.resnet50.html (дата звернення: 19.11.2025).
21. PyTorch Tutorials. Transfer Learning for Computer Vision Tutorial. Last Updated: Jan 27, 2025 [Електронний ресурс]. URL: https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html (дата звернення: 23.11.2025).
22. Konovalenko I., Fundytus S. Застосування згорткової нейромережі ResNet для автоматизованого виявлення дефектів на металоконструкціях за допомогою аналізу зображень. Наука і техніка сьогодні. 2025 [Електронний ресурс]. URL: https://perspectives.pp.ua/index.php/nts/article/view/21829 (дата звернення: 24.11.2025).
23. Barkovska O. Research into speech-to-text transformation using transformer neural network models. Information Technology and Security Systems (ITSSI). 2022. DOI:

10.30837/ITSSI.2022.22.005 [Електронний ресурс]. URL: https://itssi-journal.org/index.php/itssi/article/view/292 (дата звернення: 25.11.2025).

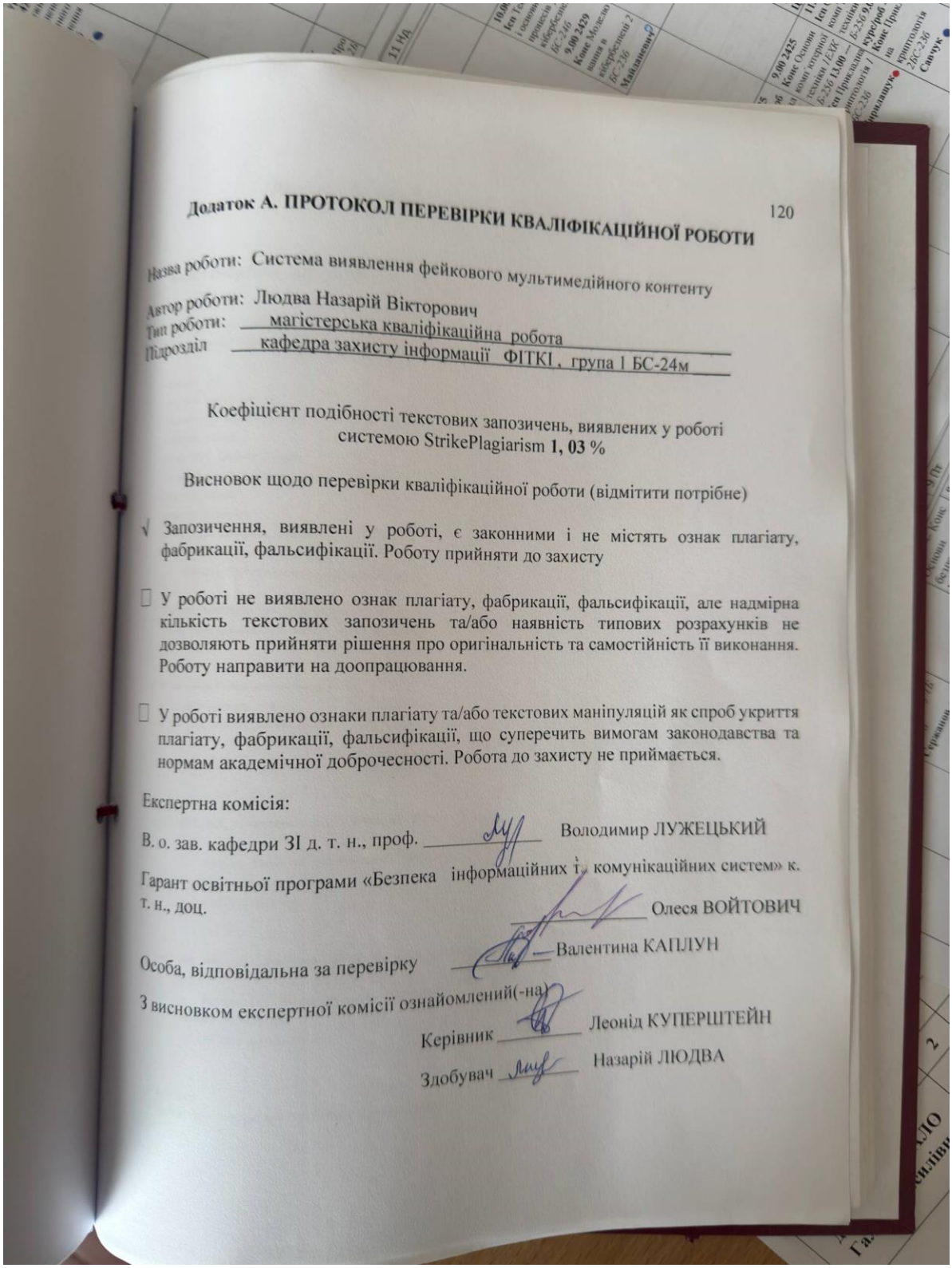
24. PyTorch Forums. Passing to the optimizers frozen parameters [Електронний ресурс]. URL:https://discuss.pytorch.org/t/passing-to-the-optimizers-frozen-parameters/83358 (дата звернення: 26.11.2025).
25. scikit-learn. Tuning the decision threshold for cost-sensitive learning [Електронний ресурс].URL:https://scikitlearn.org/stable/auto_examples/model_selection/plot_cost_sensitive_learning.html (дата звернення: 27.11.2025).
26. Google for Developers. ROC та AUC (Machine Learning Crash Course, українською) [Електронний ресурс]. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=uk (дата звернення: 27.11.2025).
27. Microsoft. Windows 10 Home and Pro - Product Lifecycle [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/lifecycle/products/windows-10-home-and-pro (дата звернення: 27.11.2025).
28. Microsoft. Windows 11 Home and Pro - Product Lifecycle [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/lifecycle/products/windows-11-home-and-pro (дата звернення: 27.11.2025).
29. Ubuntu. Ubuntu Server Documentation [Електронний ресурс]. URL: https://ubuntu.com/server/docs (дата звернення: 27.11.2025).
30. Kali Linux. Kali Tools Documentation [Електронний ресурс]. URL: https://www.kali.org/docs/tools/kali-tools/ (дата звернення: 27.11.2025).
31. Python Software Foundation. Python 3.11 Documentation [Електронний ресурс]. URL: https://docs.python.org/3.11/ (дата звернення: 28.11.2025).
32. Python Software Foundation. venv - Creation of virtual environments [Електронний ресурс]. URL: https://docs.python.org/3.11/library/venv.html (дата звернення: 28.11.2025).

33. PyTorch. PyTorch Documentation (stable) [Електронний ресурс]. URL: https://pytorch.org/docs/stable/index.html (дата звернення: 28.11.2025).
34. PyTorch. Torchvision Documentation (stable) [Електронний ресурс]. URL: https://pytorch.org/vision/stable/index.html (дата звернення: 28.11.2025).
35. Wightman R. (timm). pytorch-image-models (timm): PyTorch Image Models. GitHub repository [Електронний ресурс]. URL: https://github.com/huggingface/pytorch-image-models (дата звернення: 02.12.2025).
36. PyTorch. Torchvision Transforms [Електронний ресурс]. URL: https://pytorch.org/vision/stable/transforms.html (дата звернення: 02.12.2025).
37. OpenCV. OpenCV Documentation (4.x) [Електронний ресурс]. URL: https://docs.opencv.org/4.x/ (дата звернення: 04.12.2025).
38. timesler. facenet-pytorch: Pretrained PyTorch face recognition and MTCNN face detection. GitHub repository [Електронний ресурс]. URL: https://github.com/timesler/facenet-pytorch (дата звернення: 04.12.2025).
39. Hugging Face. Transformers Documentation [Електронний ресурс]. URL: https://huggingface.co/docs/transformers/index (дата звернення: 05.12.2025).
40. Hugging Face. Wav2Vec2 model documentation (Transformers) [Електронний ресурс]. URL: https://huggingface.co/docs/transformers/model_doc/wav2vec2 (дата звернення: 05.12.2025).
41. librosa. librosa Documentation [Електронний ресурс]. URL: https://librosa.org/doc/latest/index.html (дата звернення: 05.12.2025).
42. MoviePy. MoviePy Documentation [Електронний ресурс]. URL: https://zulko.github.io/moviepy/ (дата звернення: 07.12.2025).
43. Streamlit. Streamlit Documentation [Електронний ресурс]. URL: https://docs.streamlit.io/ (дата звернення: 10.12.2025).
44. Hugging Face. Ukrainian TTS (robinhad/ukrainian-tts), demo space [Електронний ресурс]. URL: https://huggingface.co/spaces/robinhad/ukrainian-tts (дата звернення: 11.12.2025).

45. Google Play. Voices AI: Voice Changer (com.leonfiedler.voiceai) [Електронний ресурс]. URL: https://play.google.com/store/apps/details?hl=uk&id=com.leonfiedler.voiceai (дата звернення: 12.12.2025).
46. Deepware. Deepware Scanner [Електронний ресурс]. URL: https://scanner.deepware.ai/ (дата звернення: 12.12.2025).
47. The Hive. Deepfake Detection (API documentation) [Електронний ресурс]. URL: https://docs.thehive.ai/reference/deepfake-detection (дата звернення: 13.12.2025).
48. Козловський В. О. , Лесько О. Й., Кавецький В. В.Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця. ВНТУ. 2021. 42 с. (дата звернення: 14.12.2025).
49. Кавецький В. В., Козловський В. О., Причепа І. В. Економічне обґрунтування інноваційних рішень: практикум. Вінниця. ВНТУ. 2016. 113 с. (дата звернення: 14.12.2025).

ДОДАТКИ

Додаток А. ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ



Додаток А. ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ 120

Назва роботи: Система виявлення фейкового мультимедійного контенту

Автор роботи: Людва Назарій Вікторович
Тип роботи: магістерська кваліфікаційна робота
Підрозділ кафедра захисту інформації ФІТКІ, група І БС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **1, 03** %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

В. о. зав. кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Безпека інформаційних і комунікаційних систем» к. т. н., доц. _____ Олеся ВОЙТОВИЧ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Леонід КУПЕРШТЕЙН

Здобувач _____ Назарій ЛЮДВА

Додаток Б. ТЕКСТ ПРОГРАМНОГО ЗАСТОСУНКУ

Файл deepfake_app_av.py

```
import os
from pathlib import Path
from typing import List, Tuple, Optional

import cv2
import numpy as np
import streamlit as st
import torch
from PIL import Image
from facenet_pytorch import MTCNN
import timm
import librosa
from transformers import (
    Wav2Vec2FeatureExtractor,
    Wav2Vec2ForSequenceClassification,
)
import torchvision.transforms as T
from moviepy.editor import VideoFileClip


# ==========================
# Налаштування / ENV
# ==========================

ROOT_DIR = Path(__file__).resolve().parent


def get_env_float(name: str, default: float) -> float:
    try:
        return float(os.getenv(name, default))
    except Exception:
        return default


def get_env_int(name: str, default: int) -> int:
    try:
        return int(os.getenv(name, default))
    except Exception:
        return default


# Відео
BACKBONE = os.getenv("BACKBONE", "resnet50")
```

```
VIDEO_CKPT_PATH = Path(os.getenv("CHECKPOINT_PATH", ROOT_DIR / "resnet50_head.pt"))
FAKE_THRESHOLD = get_env_float("FAKE_THRESHOLD", 0.45)  # базовий поріг
FRAME_SAMPLE_RATE = get_env_int("FRAME_SAMPLE_RATE", 1)
FRAME_HIGH_THR = get_env_float("FRAME_HIGH_THR", 0.70)  # тільки для статистики
HIGH_FRAC_MIN = get_env_float("HIGH_FRAC_MIN", 0.30)    # тільки для статистики
FACE_MARGIN = get_env_int("FACE_MARGIN", 140)

# Аудіо
AUDIO_MODEL_CKPT = Path(os.getenv("AUDIO_MODEL_CKPT", ROOT_DIR / "wav2vec2_audio_ckpt.pt"))
AUDIO_STRICT_THRESHOLD = get_env_float("AUDIO_STRICT_THRESHOLD", 0.86)
AUDIO_SEGMENT_SECONDS = get_env_float("AUDIO_SEGMENT_SECONDS", 5.0)
AUDIO_MIN_WINDOWS = get_env_int("AUDIO_MIN_WINDOWS", 1)
AUDIO_FAKE_INDEX = get_env_int("AUDIO_FAKE_INDEX", 1)  # за замовчуванням fake = logit[1]
TARGET_SR = 16000

AUDIO_EXTS = {".wav", ".mp3", ".flac", ".ogg", ".m4a"}


# ==========================
# Streamlit базові налаштування
# ==========================

st.set_page_config(
    page_title="Мультимодальний детектор діпфейків",
    page_icon="🎭",
    layout="wide",
    initial_sidebar_state="collapsed",
)

device_str = "cuda" if torch.cuda.is_available() else "cpu"
device = torch.device(device_str)

# ==========================
# Глобальні стилі
# ==========================

st.markdown(
    """
    <style>
    .block-container {
        padding-top: 2.5rem;
        padding-bottom: 2.5rem;
        max-width: 1200px;
    }
    body {
        background: radial-gradient(circle at top left, #1b2436 0, #050814 45%, #020308 100%);
        color: #f5f7ff;
        font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", sans-serif;
    }
```

```
h1 {
    font-size: 40px !important;
    font-weight: 700 !important;
    letter-spacing: 0.03em;
}
.subtitle {
    font-size: 14px;
    color: #adb3c4;
    margin-top: -8px;
    margin-bottom: 18px;
}
.badges-row span {
    display: inline-block;
    padding: 4px 10px;
    margin-right: 8px;
    margin-bottom: 4px;
    border-radius: 999px;
    font-size: 11px;
    text-transform: uppercase;
    letter-spacing: 0.09em;
    background: rgba(255,255,255,0.04);
    border: 1px solid rgba(255,255,255,0.06);
    color: #e4e6ff;
}
.badges-row span strong {
    color: #ffd86b;
}
.mode-tabs .stRadio > div {
    flex-direction: row;
    justify-content: flex-start;
}
.mode-tabs label {
    padding: 6px 16px;
    border-radius: 999px;
    border: 1px solid rgba(255,255,255,0.12);
    background: radial-gradient(circle at top, rgba(255,255,255,0.06), rgba(0,0,0,0.8));
    font-size: 13px;
    font-weight: 600;
    text-transform: uppercase;
    letter-spacing: 0.08em;
    color: #e5e8ff;
}
.mode-tabs input[type="radio"]:checked + div > span {
    background: linear-gradient(90deg, #ffcf71, #ff7a5c);
    color: #000 !important;
}
.info-card {
    padding: 14px 18px 16px 18px;
    border-radius: 14px;
    background: rgba(11,15,25,0.9);
```

```
    border: 1px solid rgba(255,255,255,0.06);
    box-shadow: 0 14px 40px rgba(0,0,0,0.45);
    font-size: 13px;
    color: #c8ccdd;
}
.info-card-title {
    font-size: 14px;
    font-weight: 600;
    margin-bottom: 4px;
    color: #f7f3ff;
}
.stats-card-title {
    font-size: 16px;
    font-weight: 600;
    margin-bottom: 10px;
    color: #f7f3ff;
}
.big-number {
    font-size: 32px;
    font-weight: 700;
    margin-bottom: 2px;
    color: #fdfbff;
}
.big-number-label {
    font-size: 12px;
    text-transform: uppercase;
    letter-spacing: 0.12em;
    color: #9fa6c0;
    margin-bottom: 12px;
}
.small-label {
    font-size: 12px;
    color: #9fa6c0;
}
.small-value {
    font-size: 14px;
    color: #f2f4ff;
    font-weight: 500;
}
.verdict-box-ok {
    margin-top: 10px;
    padding: 10px 14px;
    border-radius: 12px;
    background: rgba(26, 78, 51, 0.9);
    border: 1px solid rgba(56, 183, 100, 0.9);
    font-size: 13px;
}
.verdict-box-bad {
    margin-top: 10px;
    padding: 10px 14px;
```

```
        border-radius: 12px;
        background: rgba(122, 35, 35, 0.9);
        border: 1px solid rgba(255, 112, 112, 0.9);
        font-size: 13px;
    }
    .section-title {
        font-size: 22px;
        font-weight: 600;
        margin-top: 26px;
        margin-bottom: 6px;
    }
    .section-caption {
        font-size: 13px;
        color: #9fa6c0;
        margin-bottom: 14px;
    }
    </style>
    """,
    unsafe_allow_html=True,
)


# ==========================
# Моделі (кешуються)
# ==========================

@st.cache_resource
def get_video_model_and_transform(
    ckpt_path: Path,
    backbone: str,
    device_str: str,
):

    dev = torch.device(device_str)

    model = timm.create_model(backbone, pretrained=True, num_classes=2)

    state = torch.load(str(ckpt_path), map_location=dev)
    head_state = state.get("state_dict", state)

    try:
        classifier = model.get_classifier()
        classifier.load_state_dict(head_state, strict=True)
        model.reset_classifier(num_classes=2)
    except Exception:
        model.load_state_dict(head_state, strict=False)

    model.to(dev)
    model.eval()
```

```
    transform = T.Compose(
        [
            T.Resize((224, 224)),
            T.ToTensor(),
            T.Normalize(
                mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225],
            ),
        ]
    )
    return model, transform


@st.cache_resource
def get_mtcnn(device_str: str):
    dev = torch.device(device_str)
    mtcnn = MTCNN(
        image_size=224,
        margin=0,
        keep_all=False,
        post_process=False,
        device=dev,
    )
    return mtcnn


@st.cache_resource
def get_audio_model(ckpt_path: Path, device_str: str):
    dev = torch.device(device_str)

    state = torch.load(str(ckpt_path), map_location=dev)
    model_name = state.get("model_name", "facebook/wav2vec2-base")

    feature_extractor = Wav2Vec2FeatureExtractor.from_pretrained(model_name)
    model = Wav2Vec2ForSequenceClassification.from_pretrained(
        model_name,
        num_labels=2,
        problem_type="single_label_classification",
    )

    model.load_state_dict(state["state_dict"], strict=True)
    model.to(dev)
    model.eval()
    return feature_extractor, model


# ========================
# Відео-пайплайн
# ========================
```

```
def analyze_video(
    video_path: Path,
    device_str: str,
    trim_first_10s: bool = True,
) -> dict:

    model, transform = get_video_model_and_transform(
        VIDEO_CKPT_PATH, BACKBONE, device_str
    )
    mtcnn = get_mtcnn(device_str)

    cap = cv2.VideoCapture(str(video_path))
    if not cap.isOpened():
        return {
            "error": "Не вдалося відкрити відео-файл.",
        }

    fps = cap.get(cv2.CAP_PROP_FPS)
    if fps <= 0:
        fps = 25.0

    frame_idx = 0
    used_indices: List[int] = []
    p_fake_list: List[float] = []

    first_face_frame_rgb: Optional[np.ndarray] = None
    first_face_box: Optional[Tuple[int, int, int, int]] = None

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

        t_sec = frame_idx / fps
        if trim_first_10s and t_sec > 10.0:
            break

        if frame_idx % FRAME_SAMPLE_RATE != 0:
            frame_idx += 1
            continue

        frame_rgb = cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2RGB)

        boxes, _ = mtcnn.detect(frame_rgb)
        if boxes is None or len(boxes) == 0:
            frame_idx += 1
            continue

        if isinstance(boxes, np.ndarray):
            if boxes.ndim == 2 and boxes.shape[0] > 1:
```

```
                areas = (boxes[:, 2] - boxes[:, 0]) * (boxes[:, 3] - boxes[:, 1])
                box = boxes[areas.argmax()]
            else:
                box = boxes[0]
        else:
            box = boxes[0]

        x1, y1, x2, y2 = box
        h, w, _ = frame_rgb.shape

        x1 = int(max(0, x1 - FACE_MARGIN))
        y1 = int(max(0, y1 - FACE_MARGIN))
        x2 = int(min(w, x2 + FACE_MARGIN))
        y2 = int(min(h, y2 + FACE_MARGIN))

        if x2 <= x1 or y2 <= y1:
            frame_idx += 1
            continue

        face_crop = frame_rgb[y1:y2, x1:x2, :]
        if face_crop.size == 0:
            frame_idx += 1
            continue

        if first_face_frame_rgb is None:
            first_face_frame_rgb = frame_rgb.copy()
            first_face_box = (x1, y1, x2, y2)

        face_pil = Image.fromarray(face_crop)
        inp = transform(face_pil).unsqueeze(0).to(device)

        with torch.inference_mode():
            logits = model(inp)
            probs = torch.softmax(logits, dim=1)[0].cpu().numpy()
            p_fake = float(probs[1])

        used_indices.append(frame_idx)
        p_fake_list.append(p_fake)

        frame_idx += 1

cap.release()

if not p_fake_list:
    return {
        "n_frames": 0,
        "p_mean": 0.0,
        "p_max": 0.0,
        "frac_high": 0.0,
        "frame_indices": [],
```

```
            "p_list": [],
            "preview_frame": None,
            "is_fake": False,
            "reason": "Не виявлено придатних лицевих кадрів.",
        }

    p_arr = np.array(p_fake_list, dtype=float)
    p_mean = float(p_arr.mean())
    p_max = float(p_arr.max())
    frac_high = float((p_arr >= FRAME_HIGH_THR).mean())

    is_fake = (p_mean >= FAKE_THRESHOLD)

    preview_frame_with_box = None
    if first_face_frame_rgb is not None and first_face_box is not None:
        x1, y1, x2, y2 = first_face_box
        preview_frame_with_box = first_face_frame_rgb.copy()
        cv2.rectangle(
            preview_frame_with_box,
            (x1, y1),
            (x2, y2),
            (0, 255, 0),
            2,
        )

        ph, pw, _ = preview_frame_with_box.shape
        max_h = 380
        if ph > max_h:
            scale = max_h / ph
            new_w = int(pw * scale)
            preview_frame_with_box = cv2.resize(
                preview_frame_with_box, (new_w, max_h), interpolation=cv2.INTER_AREA
            )

    return {
        "n_frames": len(p_fake_list),
        "p_mean": p_mean,
        "p_max": p_max,
        "frac_high": frac_high,
        "frame_indices": used_indices,
        "p_list": p_fake_list,
        "preview_frame": preview_frame_with_box,
        "is_fake": is_fake,
    }


# =========================
# Аудіо-пайплайн
# =========================
```

```
def split_audio_into_windows(samples: np.ndarray, sr: int, segment_seconds: float) -> List[np.ndarray]:
    seg_len = int(segment_seconds * sr)
    n = samples.shape[0]
    if n <= seg_len:
        pad = seg_len - n
        seg = np.pad(samples, (0, pad))
        return [seg.astype("float32")]

    n_windows = n // seg_len
    windows = []
    for i in range(n_windows):
        start = i * seg_len
        end = start + seg_len
        seg = samples[start:end]
        windows.append(seg.astype("float32"))

    if not windows:
        pad = seg_len - n
        seg = np.pad(samples, (0, pad))
        windows = [seg.astype("float32")]

    return windows


def analyze_audio(
    audio_path: Path,
    device_str: str,
) -> dict:
    feature_extractor, model = get_audio_model(AUDIO_MODEL_CKPT, device_str)

    samples, sr = librosa.load(str(audio_path), sr=TARGET_SR, mono=True)
    windows = split_audio_into_windows(samples, TARGET_SR, AUDIO_SEGMENT_SECONDS)


    p_fake_list: List[float] = []

    for seg in windows:
        inputs = feature_extractor(
            [seg],
            sampling_rate=TARGET_SR,
            padding=True,
            return_tensors="pt",
        )
        input_values = inputs["input_values"].to(device)
        attention_mask = inputs.get("attention_mask")
        if attention_mask is not None:
            attention_mask = attention_mask.to(device)

        with torch.inference_mode():
            logits = model(
```

```
                input_values=input_values,
                attention_mask=attention_mask,
            ).logits
            probs = torch.softmax(logits, dim=-1)[0].cpu().numpy()
            p_fake = float(probs[AUDIO_FAKE_INDEX])
        p_fake_list.append(p_fake)

    p_arr = np.array(p_fake_list, dtype=float)
    p_mean = float(p_arr.mean())
    p_max = float(p_arr.max())

    is_fake = (p_mean >= AUDIO_STRICT_THRESHOLD)

    return {
        "n_segments": len(p_fake_list),
        "p_mean": p_mean,
        "p_max": p_max,
        "p_list": p_fake_list,
        "is_fake": is_fake,
    }


def extract_audio_from_video(video_path: Path, audio_out_path: Path, target_sr: int = TARGET_SR):
    clip = VideoFileClip(str(video_path))
    if clip.audio is None:
        clip.close()
        raise RuntimeError("У відео відсутня аудіодоріжка.")
    clip.audio.write_audiofile(
        str(audio_out_path),
        fps=target_sr,
        nbytes=2,
        codec="pcm_s16le",
        verbose=False,
        logger=None,
    )
    clip.close()


# ==========================
# UI
# ==========================

st.title("Мультимодальний детектор діпфейків")
st.markdown(
    "<p class='subtitle'>Аналіз відео та аудіо на основі ResNet50 та Wav2Vec2.0 з використанням
власного датасету.</p>",
    unsafe_allow_html=True,
)

st.markdown(
```

```
    f"""
    <div class="badges-row">
        <span>Пристрій: <strong>{device_str.upper()}</strong></span>
        <span>Backbone (відео): <strong>{BACKBONE}</strong></span>
        <span>VIDEO THRESHOLD: <strong>{FAKE_THRESHOLD:.2f}</strong></span>
        <span>AUDIO THRESHOLD: <strong>{AUDIO_STRICT_THRESHOLD:.2f}</strong></span>
    </div>
    """,
    unsafe_allow_html=True,
)

st.markdown("---")

st.markdown("#### Оберіть режим аналізу")
with st.container():
    st.markdown('<div class="mode-tabs">', unsafe_allow_html=True)
    mode = st.radio(
        "",
        options=["Відео", "Аудіо", "Відео + Аудіо"],
        horizontal=True,
    )
    st.markdown("</div>", unsafe_allow_html=True)


# ----------------------------
# Режим ВІДЕО
# ----------------------------

if mode == "Відео":
    st.markdown("<div class='section-title'>Аналіз відео</div>", unsafe_allow_html=True)
    st.markdown(
        "<p class='section-caption'>Завантажте відеофайл, щоб оцінити ймовірність діпфейку за відео-
модулем.</p>",
        unsafe_allow_html=True,
    )

    top_col1, top_col2 = st.columns(2)
    with top_col1:
        st.markdown(
            """
            <div class="info-card">
                <div class="info-card-title">Вхідні дані</div>
                <div>Підтримуються формати MP4, AVI, MOV, MKV та інші поширені контейнери.</div>
                <div style="margin-top:4px;">Алгоритм аналізує перші 10 секунд (можна вимкнути
нижче).</div>
            </div>
            """,
            unsafe_allow_html=True,
        )
    with top_col2:
```

```
        st.markdown(
            f"""
            <div class="info-card">
                <div class="info-card-title">Що покаже статистика</div>
                <div>Скільки кадрів з обличчям було проаналізовано, середню та максимальну ймовірність
фейку</div>
                <div style="margin-top:4px;">Підсумковий вердикт: <b>"правдоподібне"</b> чи
<b>"ймовірний діпфейк"</b> (поріг {FAKE_THRESHOLD*100:.0f}%).</div>
            </div>
            """,
            unsafe_allow_html=True,
        )

    uploaded_video = st.file_uploader(
        "Виберіть відеофайл",
        type=["mp4", "avi", "mov", "mkv", "mpeg", "mpg"],
    )

    trim_first_10s = st.checkbox("Аналізувати лише перші 10 секунд відео", value=True)

    if uploaded_video is not None:
        with st.expander("Оригінальне відео (натисніть, щоб відкрити)", expanded=False):
            st.video(uploaded_video)

        tmp_video_path = ROOT_DIR / ("tmp_video_" + uploaded_video.name)
        with open(tmp_video_path, "wb") as f:
            f.write(uploaded_video.getbuffer())

        if st.button("Запустити аналіз відео"):
            with st.spinner("Аналізуємо відео..."):
                video_result = analyze_video(tmp_video_path, device_str, trim_first_10s)

            if "error" in video_result:
                st.error(video_result["error"])
            else:
                col_left, col_right = st.columns([1.4, 1])

                with col_left:
                    st.markdown("##### Кадр з виділеним обличчям")
                    if video_result["preview_frame"] is not None:
                        st.image(
                            video_result["preview_frame"],
                            caption="Перший знайдений кадр з обличчям",
                            use_container_width=True,
                        )
                    else:
                        st.info("Не вдалося побудувати прев’ю з обличчям.")

                with col_right:
                    st.markdown(
```

```
                    "<div class='stats-card-title'>Статистика класифікації (відео)</div>",
                    unsafe_allow_html=True,
                )

                p_mean = video_result["p_mean"]
                p_max = video_result["p_max"]
                frac_high = video_result["frac_high"]

                st.markdown(
                    f"""
                    <div class="big-number">{p_mean * 100:.1f}%</div>
                    <div class="big-number-label">СЕРЕДНЯ ЙМОВІРНІСТЬ ФЕЙКУ</div>
                    """,
                    unsafe_allow_html=True,
                )

                m1, m2 = st.columns(2)
                with m1:
                    st.markdown(
                        f"<div class='small-label'>Кадрів проаналізовано</div>"
                        f"<div class='small-value'>{video_result['n_frames']}</div>",
                        unsafe_allow_html=True,
                    )
                with m2:
                    st.markdown(
                        f"<div class='small-label'>Максимальна ймовірність фейку</div>"
                        f"<div class='small-value'>{p_max * 100:.1f}%</div>",
                        unsafe_allow_html=True,
                    )

                st.markdown(
                    f"<div class='small-label' style='margin-top:8px;'>Кадрів з високою ймовірністю
(≥ {FRAME_HIGH_THR*100:.0f}%)</div>"
                    f"<div class='small-value'>{frac_high * 100:.1f}%</div>",
                    unsafe_allow_html=True,
                )

                if video_result["is_fake"]:
                    st.markdown(
                        f"""
                        <div class="verdict-box-bad">
                        <b>Вердикт відео-модуля:</b> контент класифіковано як <b>ймовірний
діпфейк</b>.<br/>
                        Середня ймовірність фейку {p_mean*100:.1f}% ≥ порога
{FAKE_THRESHOLD*100:.0f}%.
                        </div>
                        """,
                        unsafe_allow_html=True,
                    )
                else:
```

```
                        st.markdown(
                            f"""
                            <div class="verdict-box-ok">
                            <b>Вердикт відео-модуля:</b> відео виглядає <b>правдоподібним</b>.<br/>
                            Середня ймовірність фейку {p_mean*100:.1f}% < порога
{FAKE_THRESHOLD*100:.0f}%.
                            </div>
                            """,
                            unsafe_allow_html=True,
                        )

        try:
            if tmp_video_path.exists():
                tmp_video_path.unlink()
        except Exception:
            pass


# ----------------------------
# Режим АУДІО
# ----------------------------

elif mode == "Аудіо":
    st.markdown("<div class='section-title'>Аналіз аудіо</div>", unsafe_allow_html=True)
    st.markdown(
        "<p class='section-caption'>Завантажте аудіофайл, щоб оцінити ймовірність діпфейку голосу за
аудіо-модулем.</p>",
        unsafe_allow_html=True,
    )

    # Інформаційні блоки для аудіо
    top_col1, top_col2 = st.columns(2)
    with top_col1:
        st.markdown(
            """
            <div class="info-card">
                <div class="info-card-title">Вхідні дані (аудіо)</div>
                <div>Підтримуються формати WAV, MP3, FLAC, OGG, M4A.</div>
                <div style="margin-top:4px;">Сигнал перетворюється у моно 16 кГц та ділиться на вікна
по декілька секунд.</div>
            </div>
            """,
            unsafe_allow_html=True,
        )
    with top_col2:
        st.markdown(
            f"""
            <div class="info-card">
                <div class="info-card-title">Що покаже статистика (аудіо)</div>
                <div>Кількість сегментів (вікон) голосу, які були проаналізовані.</div>
```

```
                <div style="margin-top:4px;">Середню та максимальну ймовірність фейкового голосу і
вердикт:
                <b>"правдоподібний голос"</b> чи <b>"ймовірний діпфейк"</b> (поріг
{AUDIO_STRICT_THRESHOLD*100:.0f}%).</div>
            </div>
            """,
            unsafe_allow_html=True,
        )

    uploaded_audio = st.file_uploader(
        "Виберіть аудіофайл",
        type=["wav", "mp3", "flac", "ogg", "m4a"],
    )

    if uploaded_audio is not None:
        with st.expander("Прослухати оригінальне аудіо", expanded=False):
            st.audio(uploaded_audio)

        tmp_audio_path = ROOT_DIR / ("tmp_audio_" + uploaded_audio.name)
        with open(tmp_audio_path, "wb") as f:
            f.write(uploaded_audio.getbuffer())

        if st.button("Запустити аналіз аудіо"):
            with st.spinner("Аналізуємо аудіо..."):
                audio_result = analyze_audio(tmp_audio_path, device_str)

            p_mean = audio_result["p_mean"]
            p_max = audio_result["p_max"]
            n_segments = audio_result["n_segments"]

            st.markdown(
                "<div class='stats-card-title'>Статистика класифікації (аудіо)</div>",
                unsafe_allow_html=True,
            )

            st.markdown(
                f"""
                <div class="big-number">{p_mean * 100:.1f}%</div>
                <div class="big-number-label">СЕРЕДНЯ ЙМОВІРНІСТЬ ФЕЙКУ</div>
                """,
                unsafe_allow_html=True,
            )

            m1, m2 = st.columns(2)
            with m1:
                st.markdown(
                    f"<div class='small-label'>Сегментів (вікон по {AUDIO_SEGMENT_SECONDS:.0f}
с)</div>"
                    f"<div class='small-value'>{n_segments}</div>",
                    unsafe_allow_html=True,
```

```
                )
            with m2:
                st.markdown(
                    f"<div class='small-label'>Максимальна ймовірність фейку</div>"
                    f"<div class='small-value'>{p_max * 100:.1f}%</div>",
                    unsafe_allow_html=True,
                )

            if audio_result["is_fake"]:
                st.markdown(
                    f"""
                    <div class="verdict-box-bad">
                    <b>Вердикт аудіо-модуля:</b> голос класифіковано як <b>ймовірний діпфейк</b>.<br/>
                    Середня ймовірність фейку {p_mean*100:.1f}% ≥ порога
{AUDIO_STRICT_THRESHOLD*100:.0f}%.
                    </div>
                    """,
                    unsafe_allow_html=True,
                )
            else:
                st.markdown(
                    f"""
                    <div class="verdict-box-ok">
                    <b>Вердикт аудіо-модуля:</b> голос виглядає <b>правдоподібним</b>.<br/>
                    Середня ймовірність фейку {p_mean*100:.1f}% < порога
{AUDIO_STRICT_THRESHOLD*100:.0f}%.
                    </div>
                    """,
                    unsafe_allow_html=True,
                )

        try:
            if tmp_audio_path.exists():
                tmp_audio_path.unlink()
        except Exception:
            pass


# ---------------------------
# Режим ВІДЕО + АУДІО (SAFE_OR)
# ---------------------------

elif mode == "Відео + Аудіо":
    st.markdown("<div class='section-title'>Мультимодальний аналіз (відео + аудіо)</div>",
unsafe_allow_html=True)
    st.markdown(
        "<p class='section-caption'>Система окремо аналізує обличчя у відео та голос, а потім об'єднує
результати за правилом SAFE_OR.</p>",
        unsafe_allow_html=True,
    )
```

```
    # Інформаційні блоки для мультимодального режиму
    top_col1, top_col2 = st.columns(2)
    with top_col1:
        st.markdown(
            """
            <div class="info-card">
                <div class="info-card-title">Як працює мультимодальний аналіз</div>
                <div>Відео-модуль оцінює обличчя на кадрах, аудіо-модуль — голос у звуковій
доріжці.</div>
                <div style="margin-top:4px;">Кожен модуль повертає свою ймовірність фейку та окремий
вердикт.</div>
            </div>
            """,
            unsafe_allow_html=True,
        )
    with top_col2:
        st.markdown(
            f"""
            <div class="info-card">
                <div class="info-card-title">Правило SAFE_OR</div>
                <div>Якщо хоча б один модуль перевищує свій поріг
(відео {FAKE_THRESHOLD*100:.0f}%, аудіо {AUDIO_STRICT_THRESHOLD*100:.0f}%)</div>
                <div style="margin-top:4px;">Система вважає контент <b>"ймовірним діпфейком"</b>. Якщо
обидва модулі нижче порогів — контент вважається правдоподібним.</div>
            </div>
            """,
            unsafe_allow_html=True,
        )

    uploaded_video_both = st.file_uploader(
        "Виберіть відеофайл (буде проаналізовано і відео, і аудіодоріжку)",
        type=["mp4", "avi", "mov", "mkv", "mpeg", "mpg"],
    )

    trim_first_10s_both = st.checkbox("Аналізувати лише перші 10 секунд (для відео-модуля)",
value=True)

    if uploaded_video_both is not None:
        with st.expander("Оригінальне відео (натисніть, щоб відкрити)", expanded=False):
            st.video(uploaded_video_both)

        tmp_video_path = ROOT_DIR / ("tmp_both_video_" + uploaded_video_both.name)
        tmp_audio_from_video = ROOT_DIR / ("tmp_from_video_audio.wav")

        with open(tmp_video_path, "wb") as f:
            f.write(uploaded_video_both.getbuffer())

        if st.button("Запустити мультимодальний аналіз"):
            video_result = None
```

```
audio_result = None

try:
    with st.spinner("Аналізуємо відео..."):
        video_result = analyze_video(tmp_video_path, device_str, trim_first_10s_both)

    with st.spinner("Витягуємо аудіо з відео та аналізуємо голос..."):
        extract_audio_from_video(tmp_video_path, tmp_audio_from_video, TARGET_SR)
        audio_result = analyze_audio(tmp_audio_from_video, device_str)

    col_left, col_right = st.columns([1.4, 1])

    with col_left:
        st.markdown("##### Кадр з виділеним обличчям (відео-модуль)")
        if video_result and video_result.get("preview_frame") is not None:
            st.image(
                video_result["preview_frame"],
                caption="Перший знайдений кадр з обличчям",
                use_container_width=True,
            )
        else:
            st.info("Не вдалося побудувати прев’ю з обличчям.")

    with col_right:
        st.markdown(
            "<div class='stats-card-title'>Об'єднана статистика</div>",
            unsafe_allow_html=True,
        )

        if video_result:
            v_p_mean = video_result["p_mean"]
            v_p_max = video_result["p_max"]
            v_frac_high = video_result["frac_high"]
            v_is_fake = video_result["is_fake"]
            v_n_frames = video_result["n_frames"]
        else:
            v_p_mean = v_p_max = v_frac_high = 0.0
            v_is_fake = False
            v_n_frames = 0

        if audio_result:
            a_p_mean = audio_result["p_mean"]
            a_p_max = audio_result["p_max"]
            a_is_fake = audio_result["is_fake"]
            n_segments = audio_result["n_segments"]
        else:
            a_p_mean = a_p_max = 0.0
            a_is_fake = False
            n_segments = 0
```

```
                    combined_is_fake = (v_is_fake or a_is_fake)
                    combined_p = max(v_p_mean, a_p_mean)

                    st.markdown(
                        f"""
                        <div class="big-number">{combined_p * 100:.1f}%</div>
                        <div class="big-number-label">МАКСИМАЛЬНА ОЦІНКА ФЕЙКУ (ВІД ВІДЕО АБО
АУДІО)</div>
                        """,
                        unsafe_allow_html=True,
                    )

                    m1, m2 = st.columns(2)
                    with m1:
                        st.markdown(
                            f"<div class='small-label'>Відео: кадрів проаналізовано</div>"
                            f"<div class='small-value'>{v_n_frames}</div>",
                            unsafe_allow_html=True,
                        )
                        st.markdown(
                            f"<div class='small-label' style='margin-top:6px;'>Відео: середня
ймовірність фейку</div>"
                            f"<div class='small-value'>{v_p_mean * 100:.1f}%</div>",
                            unsafe_allow_html=True,
                        )
                    with m2:
                        st.markdown(
                            f"<div class='small-label'>Аудіо: сегментів (вікон)</div>"
                            f"<div class='small-value'>{n_segments}</div>",
                            unsafe_allow_html=True,
                        )
                        st.markdown(
                            f"<div class='small-label' style='margin-top:6px;'>Аудіо: середня
ймовірність фейку</div>"
                            f"<div class='small-value'>{a_p_mean * 100:.1f}%</div>",
                            unsafe_allow_html=True,
                        )

                    st.markdown(
                        f"<div class='small-label' style='margin-top:8px;'>Відео: кадрів з високою
ймовірністю (≥ {FRAME_HIGH_THR*100:.0f}%)</div>"
                        f"<div class='small-value'>{v_frac_high * 100:.1f}%</div>",
                        unsafe_allow_html=True,
                    )

                    if combined_is_fake:
                        st.markdown(
                            """
                            <div class="verdict-box-bad">
                            <b>Підсумковий вердикт (SAFE_OR):</b> контент класифіковано як <b>ймовірний
```

```
діпфейк</b>,
                            оскільки принаймні один з модулів (відео або аудіо) перевищив свій поріг.
                            </div>
                            """,
                            unsafe_allow_html=True,
                        )
                    else:
                        st.markdown(
                            """
                            <div class="verdict-box-ok">
                            <b>Підсумковий вердикт (SAFE_OR):</b> контент виглядає
<b>правдоподібним</b>,
                            оскільки ні відео-, ні аудіо-модуль не перевищили своїх порогів.
                            </div>
                            """,
                            unsafe_allow_html=True,
                        )

            except Exception as e:
                st.error(f"Помилка під час мультимодального аналізу: {e}")

            finally:
                try:
                    if tmp_video_path.exists():
                        tmp_video_path.unlink()
                except Exception:
                    pass
                try:
                    if tmp_audio_from_video.exists():
                        tmp_audio_from_video.unlink()
                except Exception:
                    pass
```

Файл train_resnet_head.py

```
import os
import time
import cv2
import random
import argparse
from pathlib import Path
from typing import List, Tuple, Dict

import torch
import numpy as np
from PIL import Image
from tqdm import tqdm
from sklearn.metrics import accuracy_score, f1_score

import timm
import torch.nn.functional as F
import torchvision.transforms as T
from facenet_pytorch import MTCNN
from torch.utils.data import Dataset, DataLoader

# ====================== DEFAULT CONFIG ======================
DEF_DATA_ROOT = "dataset"
```

```
DEF_FRAMES_PER_VIDEO = 16         # рівномірно вибрані кадри з кожного відео
DEF FACE MARGIN = 100
DEF_IMG_SIZE = 224                # для ResNet50
DEF_BATCH = 16
DEF_EPOCHS = 10                   # 4 епохи "голова", далі розморозка layer4
DEF_LR = 1e-3
DEF_WEIGHT_DECAY = 1e-4
DEF WORKERS = 0                   # Windows-safe; можеш збільшити після перевірки
DEF_SEED = 42
DEF_OUT = "resnet50_head.pt"

random.seed(DEF_SEED)
np.random.seed(DEF_SEED)
torch.manual seed(DEF SEED)

# ====================== UTILS ======================
def sample_frame_indices(num_frames: int, k: int) -> List[int]:
    if num_frames <= 0:
        return []
    if k >= num_frames:
        return list(range(num_frames))
    step = num_frames / k
    return [int(i * step) for i in range(k)]

def read total frames(cap: cv2.VideoCapture) -> int:
    n = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    return max(n, 0)

def crop_face_bgr(frame_bgr: np.ndarray, mtcnn: MTCNN, margin: int) -> np.ndarray:

    h, w = frame bgr.shape[:2]
    try:
        pil = Image.fromarray(cv2.cvtColor(frame bgr, cv2.COLOR BGR2RGB))
        boxes, _ = mtcnn.detect(pil)
        if boxes is not None and len(boxes) > 0:
            x1, y1, x2, y2 = boxes[0]
            x1 = max(int(x1) - margin, 0)
            y1 = max(int(y1) - margin, 0)
            x2 = min(int(x2) + margin, w)
            y2 = min(int(y2) + margin, h)
            face = frame_bgr[y1:y2, x1:x2]
            if face.size > 0:
                return face
    except Exception:
        pass
    # центр-кроп
    side = min(h, w)
    y0 = (h - side) // 2
    x0 = (w - side) // 2
    return frame bgr[y0:y0+side, x0:x0+side]

# ====================== DATASET ======================
class FaceFrameDataset(Dataset):

    def __init__(self, root: str, split: str, frames_per_video: int, img_size: int, face_margin: int):
        assert split in ["train", "val"]
        self.root = root
        self.split = split
        self.frames_per_video = frames_per_video
        self.img size = img size
        self.face_margin = face_margin

        self.mtcnn = MTCNN(keep_all=False, device="cuda" if torch.cuda.is_available() else "cpu")
        self.items: List[Tuple[str, int]] = []
        mapping = {"real": 0, "fake": 1}
        for cls name, cls idx in mapping.items():
            d = Path(root) / split / cls name
            if not d.exists():
                continue
            for ext in ("*.mp4", "*.avi", "*.mov", "*.mkv"):
                for p in d.glob(ext):
                    self.items.append((str(p), cls_idx))

        if split == "train":
```

```
            self.transform = T.Compose([
                T.Resize((img size, img size)),
                T.RandomHorizontalFlip(p=0.5),
                T.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.05, hue=0.02),
                T.ToTensor(),
                T.Normalize(mean=[0.485, 0.456, 0.406],
                            std=[0.229, 0.224, 0.225]),
            ])
        else:
            self.transform = T.Compose([
                T.Resize((img_size, img_size)),
                T.ToTensor(),
                T.Normalize(mean=[0.485, 0.456, 0.406],
                            std=[0.229, 0.224, 0.225]),
            ])

    def __len__(self):
        return len(self.items) * self.frames_per_video

    def __getitem__(self, idx):
        vid_id = idx // self.frames_per_video
        local_idx = idx % self.frames_per_video
        video_path, label = self.items[vid_id]

        cap = cv2.VideoCapture(video path)
        total = read_total_frames(cap)
        if total == 0:
            cap.release()
            img = Image.fromarray(np.zeros((self.img_size, self.img_size, 3), dtype=np.uint8))
            return self.transform(img), label, Path(video_path).name

        idxs = sample_frame_indices(total, self.frames_per_video)
        frame idx = idxs[min(local idx, len(idxs) - 1)]
        cap.set(cv2.CAP_PROP_POS_FRAMES, frame_idx)
        ok, frame = cap.read()
        cap.release()

        if not ok or frame is None:
            img = Image.fromarray(np.zeros((self.img size, self.img size, 3), dtype=np.uint8))
            return self.transform(img), label, Path(video_path).name

        face bgr = crop face bgr(frame, self.mtcnn, self.face margin)
        face_rgb = cv2.cvtColor(face_bgr, cv2.COLOR_BGR2RGB)
        pil = Image.fromarray(face rgb)
        x = self.transform(pil)
        return x, label, Path(video_path).name

# ====================== MODEL ======================
def build_resnet50_head(num_classes=2, freeze_backbone=True):
    model = timm.create model("resnet50", pretrained=True, num classes=num classes)
    if freeze_backbone:
        for n, p in model.named_parameters():
            if "fc" not in n:
                p.requires grad = False
    return model

# ====================== TRAIN / VAL ======================
def train_one_epoch(model, loader, optim, device):
    model.train()
    loss meter, n = 0.0, 0
    for x, y, _ in tqdm(loader, desc="train", leave=False):
        x = x.to(device)
        y = y.to(device)
        logits = model(x)
        loss = F.cross_entropy(logits, y)
        optim.zero grad()
        loss.backward()
        optim.step()
        loss_meter += float(loss.item()) * x.size(0)
        n += x.size(0)
    return loss meter / max(n, 1)

@torch.no grad()
def validate(model, loader, device) -> Dict[str, float]:
```

```
    model.eval()
    all_pred, all_true = [], []
    per_video_probs: Dict[str, List[float]] = {}
    per_video_labels: Dict[str, int] = {}

    for x, y, vidname in tqdm(loader, desc="val", leave=False):
        x = x.to(device)
        probs = torch.softmax(model(x), dim=1)[:, 1].detach().cpu().numpy()  # P(fake)
        pred = (probs >= 0.5).astype(np.int64)

        all_pred.extend(pred.tolist())
        all_true.extend(y.numpy().tolist())

        for i in range(len(vidname)):
            vn = vidname[i]
            per video probs.setdefault(vn, []).append(float(probs[i]))
            per_video_labels.setdefault(vn, int(y.numpy()[i]))

    acc = accuracy score(all true, all pred) if len(all true) else 0.0
    f1  = f1_score(all_true, all_pred) if len(set(all_true)) > 1 else 0.0

    v_preds, v_true = [], []
    for vn, ps in per_video_probs.items():
        v_preds.append(1 if (sum(ps)/len(ps)) >= 0.5 else 0)
        v true.append(per video labels[vn])
    v_acc = accuracy_score(v_true, v_preds) if v_true else 0.0
    v f1  = f1 score(v true, v preds) if len(set(v true)) > 1 else 0.0

    return {"frame_acc": acc, "frame_f1": f1, "video_acc": v_acc, "video_f1": v_f1}

@torch.no grad()
def evaluate_on_loader(model, loader, device) -> Dict[str, float]:

    model.eval()
    all_pred, all_true = [], []
    for x, y, _ in loader:
        x = x.to(device)
        probs = torch.softmax(model(x), dim=1)[:, 1].detach().cpu().numpy()
        pred = (probs >= 0.5).astype(np.int64)
        all_pred.extend(pred.tolist())
        all_true.extend(y.numpy().tolist())
    acc = accuracy score(all true, all pred) if len(all true) else 0.0
    f1  = f1_score(all_true, all_pred) if len(set(all_true)) > 1 else 0.0
    return {"acc": acc, "f1": f1}

def print_split_stats(ds: FaceFrameDataset, name: str):
    from collections import Counter
    c = Counter([lbl for  , lbl in ds.items])
    print(f"{name}: videos={len(ds.items)} | per-class={dict(c)} | approx_samples={len(ds)}")

# ======================= MAIN =======================
def main():
    ap = argparse.ArgumentParser()
    ap.add argument("--data root", type=str, default=DEF DATA ROOT)
    ap.add_argument("--frames_per_video", type=int, default=DEF_FRAMES_PER_VIDEO)
    ap.add argument("--face margin", type=int, default=DEF FACE MARGIN)
    ap.add_argument("--img_size", type=int, default=DEF_IMG_SIZE)
    ap.add_argument("--batch", type=int, default=DEF_BATCH)
    ap.add_argument("--epochs", type=int, default=DEF_EPOCHS)
    ap.add argument("--lr", type=float, default=DEF LR)
    ap.add_argument("--wd", type=float, default=DEF_WEIGHT_DECAY)
    ap.add argument("--workers", type=int, default=DEF WORKERS)
    ap.add_argument("--out", type=str, default=DEF_OUT)
    args = ap.parse_args()

    device = "cuda" if torch.cuda.is available() else "cpu"

    # Датасети
    train_ds = FaceFrameDataset(args.data_root, "train", args.frames_per_video, args.img_size,
args.face_margin)
    val ds   = FaceFrameDataset(args.data root, "val",   args.frames per video, args.img size,
args.face_margin)
    if len(train ds) == 0 or len(val ds) == 0:
        print("Порожні train/val. Перевір структуру і файли.")
```

```
        return

    print_split_stats(train_ds, "TRAIN")
    print_split_stats(val_ds,   "VAL")

    train_loader = DataLoader(train_ds, batch_size=args.batch, shuffle=True,
                              num_workers=args.workers, pin_memory=True)
    val loader   = DataLoader(val ds,   batch size=args.batch, shuffle=False,
                              num_workers=args.workers, pin_memory=True)


    model = build_resnet50_head(num_classes=2, freeze_backbone=True).to(device)

    # Оптимізатор для голови
    head_params = [p for n, p in model.named_parameters() if p.requires_grad]
    optim = torch.optim.AdamW(head params, lr=args.lr, weight decay=args.wd)

    best_vf1 = -1.0
    for ep in range(1, args.epochs + 1):
        t0 = time.time()

        tr_loss = train_one_epoch(model, train_loader, optim, device)
        tr_m = evaluate_on_loader(model, train_loader, device)  # sanity-check на train
        mets = validate(model, val_loader, device)

        dt = time.time() - t0
        print(
            f"Epoch {ep}/{args.epochs} | loss={tr_loss:.4f} | "
            f"train_acc={tr_m['acc']:.3f} train_f1={tr_m['f1']:.3f} | "
            f"val_frame_acc={mets['frame_acc']:.3f} val_frame_f1={mets['frame_f1']:.3f} | "
            f"val video acc={mets['video acc']:.3f} val video f1={mets['video f1']:.3f} | {dt:.1f}s"
        )


        if ep == 4:
            for n, p in model.named_parameters():
                if n.startswith("layer4."):
                    p.requires_grad = True

            params = []
            for n, p in model.named_parameters():
                if p.requires grad:
                    lr = args.lr * 0.1 if n.startswith("layer4.") else args.lr
                    params.append({"params": [p], "lr": lr})
            optim = torch.optim.AdamW(params, weight_decay=args.wd)
            print("Unfrozen layer4 with 0.1x LR.")

        if mets["video f1"] > best vf1:
            best_vf1 = mets["video_f1"]
            torch.save(
                {"state_dict": model.state_dict(),
                 "config": vars(args),
                 "best_video_f1": best_vf1},
                args.out
            )
            print(f"Saved best to {args.out} (video f1={best vf1:.3f})")

    print("Готово.")

if   name   == "  main  ":
    main()
```

Файл train_audio_wav2vec2.py

```
import argparse

import random

import time

from pathlib import Path
```

```
import numpy as np
import torch
import librosa
from torch.utils.data import Dataset, DataLoader
from sklearn.metrics import accuracy_score, f1_score
from transformers import Wav2Vec2FeatureExtractor, Wav2Vec2ForSequenceClassification


RANDOM_SEED = 42
random.seed(RANDOM_SEED)
np.random.seed(RANDOM_SEED)
torch.manual_seed(RANDOM_SEED)

AUDIO_EXTS = {".wav", ".mp3", ".flac", ".ogg", ".m4a"}


def list_audio_files(root: Path):
    files = []
    labels = []
    mapping = {"real": 0, "fake": 1}
    for cls_name, lab in mapping.items():
        d = root / cls_name
        if not d.exists():
            continue
        for p in d.iterdir():
            if p.is_file() and p.suffix.lower() in AUDIO_EXTS:
                files.append(p)
                labels.append(lab)
    return files, labels


class AudioSegmentDataset(Dataset):
    def __init__(
        self,
        files,
        labels,
        segment_seconds: float = 5.0,
        target_sr: int = 16000,
        train: bool = True,
    ):
        assert len(files) == len(labels)
        self.files = list(files)
        self.labels = list(labels)
        self.segment_seconds = segment_seconds
        self.target_sr = target_sr
        self.segment_len = int(segment_seconds * target_sr)
        self.train = train

    def __len__(self):
        return len(self.files)
```

```
    def _load_wav(self, path: Path) -> torch.Tensor:
        samples, sr = librosa.load(str(path), sr=self.target_sr, mono=True)
        return torch.from_numpy(samples.astype("float32"))

    def _crop_or_pad(self, x: torch.Tensor) -> torch.Tensor:
        n = x.shape[0]
        if n >= self.segment_len:
            if self.train:
                max_start = max(n - self.segment_len, 0)
                start = random.randint(0, max_start) if max_start > 0 else 0
            else:
                start = (n - self.segment_len) // 2
            return x[start:start + self.segment_len]
        else:
            pad_len = self.segment_len - n
            return torch.nn.functional.pad(x, (0, pad_len))

    def __getitem__(self, idx):
        path = self.files[idx]
        label = self.labels[idx]
        wav = self._load_wav(path)
        wav = self._crop_or_pad(wav)
        return wav, label


def make_collate_fn(feature_extractor):
    def collate(batch):
        wavs, labels = zip(*batch)
        arrays = [w.numpy() for w in wavs]
        inputs = feature_extractor(
            arrays,
            sampling_rate=16000,
            padding=True,
            return_tensors="pt",
        )
        input_values = inputs["input_values"]
        attention_mask = inputs.get("attention_mask")
        labels_t = torch.tensor(labels, dtype=torch.long)
        return input_values, attention_mask, labels_t

    return collate


def train_epoch(model, loader, optimizer, device):
    model.train()
    total_loss = 0.0
    n_examples = 0

    for input_values, attention_mask, labels in loader:
```

```
            input_values = input_values.to(device)
            if attention_mask is not None:
                attention_mask = attention_mask.to(device)
            labels = labels.to(device)

            outputs = model(
                input_values=input_values,
                attention_mask=attention_mask,
                labels=labels,
            )
            loss = outputs.loss

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

            bs = labels.size(0)
            total_loss += float(loss.item()) * bs
            n_examples += bs

    return total_loss / max(1, n_examples)


@torch.no_grad()
def eval_epoch(model, loader, device):
    model.eval()
    all_labels = []
    all_probs = []

    for input_values, attention_mask, labels in loader:
        input_values = input_values.to(device)
        if attention_mask is not None:
            attention_mask = attention_mask.to(device)
        labels = labels.to(device)

        outputs = model(
            input_values=input_values,
            attention_mask=attention_mask,
        )
        logits = outputs.logits
        probs = torch.softmax(logits, dim=-1)[:, 1]

        all_labels.extend(labels.cpu().numpy().tolist())
        all_probs.extend(probs.cpu().numpy().tolist())

    if not all_labels:
        return {"acc": 0.0, "f1": 0.0}

    y_true = np.array(all_labels, dtype=int)
    pvals = np.array(all_probs, dtype=float)
```

```
    y_pred = (pvals >= 0.5).astype(int)

    acc = accuracy_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred) if len(set(y_true)) > 1 else 0.0

    return {"acc": float(acc), "f1": float(f1)}


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("--data_root", type=str, default="audio_dataset")
    parser.add_argument("--model_name", type=str, default="facebook/wav2vec2-base")
    parser.add_argument("--segment_seconds", type=float, default=5.0)
    parser.add_argument("--batch_size", type=int, default=4)
    parser.add_argument("--epochs", type=int, default=5)
    parser.add_argument("--lr", type=float, default=1e-5)
    parser.add_argument("--out", type=str, default="wav2vec2_audio_ckpt.pt")
    parser.add_argument("--freeze_encoder", action="store_true")
    args = parser.parse_args()

    device = "cuda" if torch.cuda.is_available() else "cpu"

    data_root = Path(args.data_root)
    train_files, train_labels = list_audio_files(data_root / "train")
    val_files, val_labels = list_audio_files(data_root / "val")

    if not train_files or not val_files:
        print("Порожній train або val. Перевір структуру audio_dataset/train|val/real|fake.")
        return

    print(f"TRAIN: {len(train_files)} аудіофайлів")
    print(f"VAL:   {len(val_files)} аудіофайлів")

    feature_extractor = Wav2Vec2FeatureExtractor.from_pretrained(args.model_name)

    train_ds = AudioSegmentDataset(
        train_files,
        train_labels,
        segment_seconds=args.segment_seconds,
        target_sr=16000,
        train=True,
    )
    val_ds = AudioSegmentDataset(
        val_files,
        val_labels,
        segment_seconds=args.segment_seconds,
        target_sr=16000,
        train=False,
    )
```

```
collate = make_collate_fn(feature_extractor)

train_loader = DataLoader(
    train_ds,
    batch_size=args.batch_size,
    shuffle=True,
    collate_fn=collate,
)
val_loader = DataLoader(
    val_ds,
    batch_size=args.batch_size,
    shuffle=False,
    collate_fn=collate,
)

model = Wav2Vec2ForSequenceClassification.from_pretrained(
    args.model_name,
    num_labels=2,
    problem_type="single_label_classification",
)

if args.freeze_encoder:
    for p in model.wav2vec2.parameters():
        p.requires_grad = False
    print("Заморозили wav2vec2-encoder, навчаємо тільки голову.")

model.to(device)

optimizer = torch.optim.AdamW(
    [p for p in model.parameters() if p.requires_grad],
    lr=args.lr,
)

best_f1 = -1.0

for epoch in range(1, args.epochs + 1):
    t0 = time.time()
    train_loss = train_epoch(model, train_loader, optimizer, device)
    val_metrics = eval_epoch(model, val_loader, device)
    dt = time.time() - t0

    print(
        f"Epoch {epoch}/{args.epochs} | "
        f"train_loss={train_loss:.4f} | "
        f"val_acc={val_metrics['acc']:.3f} | "
        f"val_f1={val_metrics['f1']:.3f} | "
        f"time={dt:.1f}s"
    )

    if val_metrics["f1"] > best_f1:
```

```
            best_f1 = val_metrics["f1"]
            state = {
                "model_name": args.model_name,
                "state_dict": model.state_dict(),
                "val_f1": best_f1,
                "config": vars(args),
            }
            torch.save(state, args.out)
            print(f"Saved best checkpoint to {args.out} (val_f1={best_f1:.3f})")

    print("Готово.")


if __name__ == "__main__":
    main()
```

Файл evel_resnet_val.py

```
import os
import cv2
import argparse
from pathlib import Path
from typing import List

import torch
import numpy as np
from PIL import Image
from facenet_pytorch import MTCNN
import timm
import torchvision.transforms as T
import torch.nn.functional as F
from sklearn.metrics import f1_score, accuracy_score

from metrics_utils import confusion_from_scores, metrics_from_confusion, save_scores_csv

DEVICE = "cuda" if torch.cuda.is_available() else "cpu"


def list_videos(dirp: Path) -> List[str]:
    vids = []
    for ext in ("*.mp4", "*.avi", "*.mov", "*.mkv"):
        vids += [str(p) for p in dirp.glob(ext)]
    return vids


def read_total_frames(cap):
    return max(int(cap.get(cv2.CAP_PROP_FRAME_COUNT)), 0)


def sample_indices(n, k):
```

```
    if n <= 0:
        return []
    if k >= n:
        return list(range(n))
    step = n / k
    return [int(i * step) for i in range(k)]


def crop_face_bgr(frame_bgr, mtcnn: MTCNN, margin=100):
    h, w = frame_bgr.shape[:2]
    try:
        pil = Image.fromarray(cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2RGB))
        boxes, _ = mtcnn.detect(pil)
        if boxes is not None and len(boxes) > 0:
            x1, y1, x2, y2 = boxes[0]
            x1 = max(int(x1) - margin, 0)
            y1 = max(int(y1) - margin, 0)
            x2 = min(int(x2) + margin, w)
            y2 = min(int(y2) + margin, h)
            face = frame_bgr[y1:y2, x1:x2]
            if face.size > 0:
                return face
    except Exception:
        pass

    side = min(h, w)
    y0 = (h - side) // 2
    x0 = (w - side) // 2
    return frame_bgr[y0:y0 + side, x0:x0 + side]


def build_model(img_size=224, num_classes=2):
    model = timm.create_model("resnet50", pretrained=False, num_classes=num_classes)
    model.eval().to(DEVICE)
    transform = T.Compose([
        T.Resize((img_size, img_size)),
        T.ToTensor(),
        T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ])
    return model, transform


def load_checkpoint(model, ckpt_path: str):
    state = torch.load(ckpt_path, map_location=DEVICE)
    sd = state.get("state_dict", state)
    sd = {k.replace("module.", "", 1): v for k, v in sd.items()}
    model.load_state_dict(sd, strict=False)
    return model
```

```
def score_video(video_path: str, mtcnn: MTCNN, model, transform, frames_per_video=16) -> float:
    cap = cv2.VideoCapture(video_path)
    n = read_total_frames(cap)
    if n == 0:
        return 0.0

    idxs = sample_indices(n, frames_per_video)
    probs = []

    for idx in idxs:
        cap.set(cv2.CAP_PROP_POS_FRAMES, idx)
        ok, frame = cap.read()
        if not ok or frame is None:
            continue

        face = crop_face_bgr(frame, mtcnn, margin=120)
        rgb = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
        x = transform(Image.fromarray(rgb)).unsqueeze(0).to(DEVICE)

        with torch.inference_mode():
            p = F.softmax(model(x), dim=1).detach().cpu().numpy()[0][1]  # P(fake)

        probs.append(float(p))

    cap.release()
    return float(np.mean(probs)) if probs else 0.0


def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--data_root", type=str, default="dataset")
    ap.add_argument("--split", type=str, default="val", choices=["train", "val", "test"])
    ap.add_argument("--ckpt", type=str, required=True)
    ap.add_argument("--frames_per_video", type=int, default=16)
    ap.add_argument("--out_csv", type=str, default="video_scores_val.csv")
    args = ap.parse_args()

    mtcnn = MTCNN(keep_all=False, device=DEVICE)
    model, transform = build_model()
    load_checkpoint(model, args.ckpt)

    split_root = Path(args.data_root) / args.split
    classes = {"real": 0, "fake": 1}

    rows = []  # (key, label, p_fake_mean)
    for cname, lab in classes.items():
        for vp in list_videos(split_root / cname):
            p = score_video(vp, mtcnn, model, transform, args.frames_per_video)
            key = Path(vp).stem  # щоб було зручно зводити з аудіо по імені
            rows.append((key, lab, p))
```

```
    if len(rows) == 0:
        print(f"{args.split.upper()} порожній")
        return

    print(f"\nPer-video probabilities on {args.split.upper()}:")
    for key, lab, p in rows:
        print(f"{key:40s}  label={lab}  P_fake={p:.3f}")

    y_true = np.array([lab for _, lab, _ in rows], dtype=int)
    pvals = np.array([p for _, _, p in rows], dtype=float)

    best_thr, best_f1, best_acc = 0.5, -1.0, 0.0
    for thr in np.linspace(0.1, 0.9, 17):
        y_pred = (pvals >= thr).astype(int)
        f1 = f1_score(y_true, y_pred) if len(set(y_true)) > 1 else 0.0
        acc = accuracy_score(y_true, y_pred)
        if f1 > best_f1:
            best_f1, best_thr, best_acc = float(f1), float(thr), float(acc)

    print(f"\nBest threshold search ({args.split.upper()}):")
    print(f"best_thr={best_thr:.2f} | video_f1={best_f1:.3f} | video_acc={best_acc:.3f}")

    c = confusion_from_scores(y_true.tolist(), pvals.tolist(), best_thr)
    m = metrics_from_confusion(c)

    print("\nCONFUSION (video):")
    print(f"TP={c.tp} TN={c.tn} FP={c.fp} FN={c.fn}")

    print("\nMETRICS (video):")
    print(
        f"acc={m.accuracy:.3f}  prec={m.precision:.3f}  rec={m.recall:.3f}  f1={m.f1:.3f}  "
        f"type1={m.err_type_1:.3f}  type2={m.err_type_2:.3f}"
    )

    save_scores_csv(args.out_csv, rows)
    print(f"\nSaved: {args.out_csv}")

    if args.split == "val":
        print("\nSet in Streamlit env:")
        print(f"CHECKPOINT_PATH = {Path(args.ckpt).resolve()}")
        print("BACKBONE = resnet50")
        print(f"FAKE_THRESHOLD = {best_thr:.2f}")


if __name__ == "__main__":
    main()
```

Файл evel_audio_wav2vec2.py

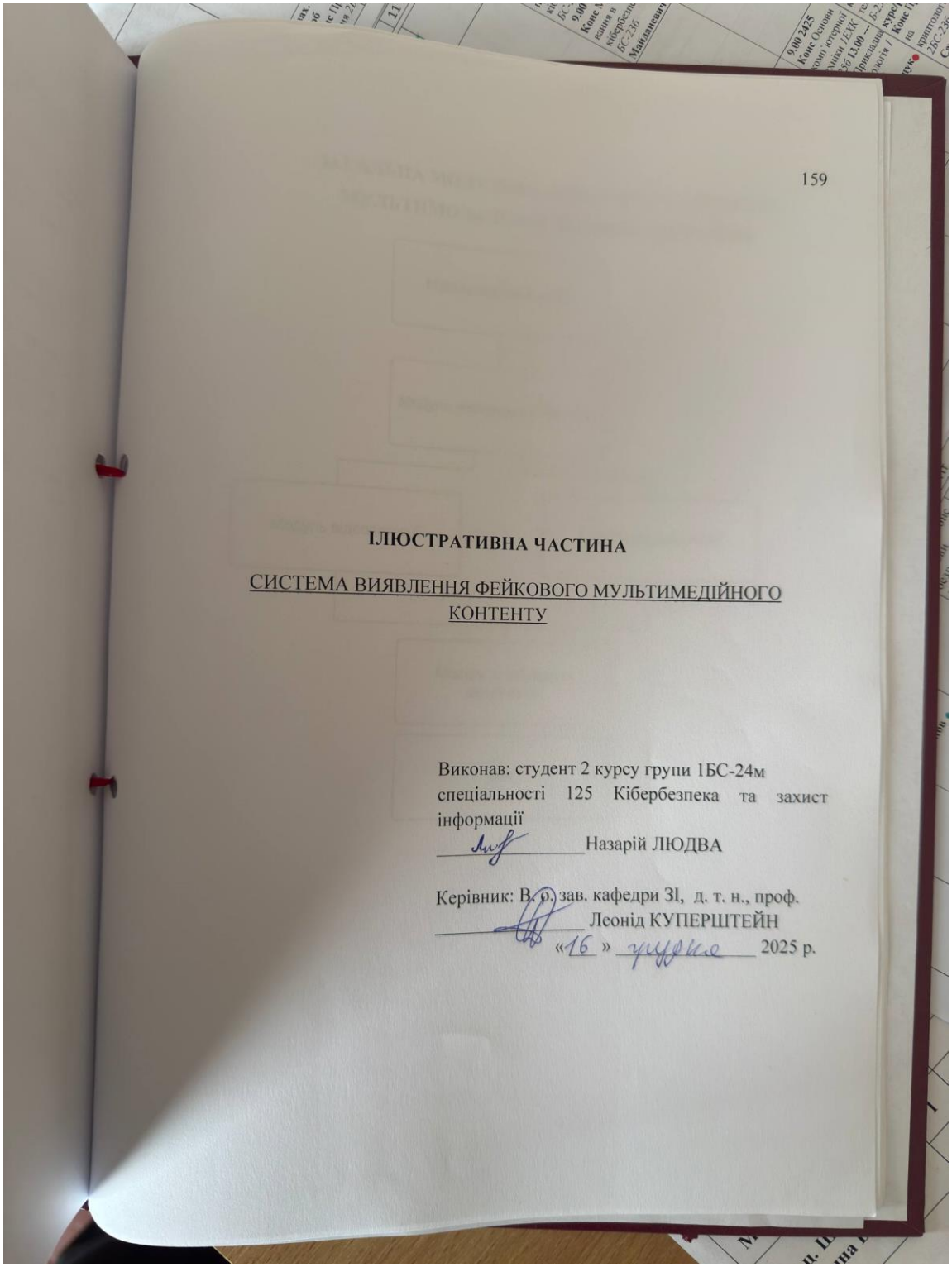
```
import argparse
from pathlib import Path
from typing import List, Tuple

import numpy as np
import librosa
import torch
from transformers import Wav2Vec2FeatureExtractor, Wav2Vec2ForSequenceClassification
from sklearn.metrics import f1_score, accuracy_score

from metrics_utils import confusion_from_scores, metrics_from_confusion, save_scores_csv

DEVICE = "cuda" if torch.cuda.is_available() else "cpu"


def list_audios(dirp: Path) -> List[str]:
    exts = ("*.wav", "*.mp3", "*.flac", "*.ogg", "*.m4a")
    out = []
    for e in exts:
        out += [str(p) for p in dirp.glob(e)]
    return out


def split_audio_into_segments(y: np.ndarray, sr: int, seg_sec: float) -> List[np.ndarray]:
    seg_len = int(seg_sec * sr)
    if seg_len <= 0:
        return [y.astype("float32")]

    if len(y) <= seg_len:
        pad = seg_len - len(y)
        if pad > 0:
            y = np.pad(y, (0, pad))
        return [y.astype("float32")]

    segments = []
    for start in range(0, len(y) - seg_len + 1, seg_len):
        seg = y[start:start + seg_len]
        segments.append(seg.astype("float32"))

    return segments if segments else [y.astype("float32")]


def load_audio_model(ckpt_path: str):
    state = torch.load(ckpt_path, map_location=DEVICE)
    model_name = state.get("model_name", "facebook/wav2vec2-base")

    fe = Wav2Vec2FeatureExtractor.from_pretrained(model_name)
    model = Wav2Vec2ForSequenceClassification.from_pretrained(
        model_name,
        num_labels=2,
```

```
        problem_type="single_label_classification",
    )
    model.load_state_dict(state["state_dict"], strict=True)
    model.to(DEVICE).eval()
    return fe, model


def score_audio_file(
    audio_path: str,
    feature_extractor,
    model,
    target_sr: int,
    segment_seconds: float,
    fake_index: int,
) -> Tuple[float, float, int]:
    y, sr = librosa.load(audio_path, sr=target_sr, mono=True)
    segments = split_audio_into_segments(y, target_sr, segment_seconds)

    scores = []
    for seg in segments:
        inputs = feature_extractor(
            [seg],
            sampling_rate=target_sr,
            padding=True,
            return_tensors="pt",
        )
        input_values = inputs["input_values"].to(DEVICE)
        attention_mask = inputs.get("attention_mask")
        if attention_mask is not None:
            attention_mask = attention_mask.to(DEVICE)

        with torch.inference_mode():
            logits = model(input_values=input_values, attention_mask=attention_mask).logits
            probs = torch.softmax(logits, dim=-1)[0].detach().cpu().numpy()
            p_fake = float(probs[fake_index])

        scores.append(p_fake)

    arr = np.array(scores, dtype=np.float32)
    return float(arr.mean()), float(arr.max()), int(len(segments))


def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--data_root", type=str, default="dataset")
    ap.add_argument("--split", type=str, default="val", choices=["train", "val", "test"])
    ap.add_argument("--ckpt", type=str, required=True)
    ap.add_argument("--target_sr", type=int, default=16000)
    ap.add_argument("--segment_seconds", type=float, default=5.0)
    ap.add_argument("--fake_index", type=int, default=1)
```

```
ap.add_argument("--out_csv", type=str, default="audio_scores_val.csv")
args = ap.parse_args()

feature_extractor, model = load_audio_model(args.ckpt)

split_root = Path(args.data_root) / args.split
classes = {"real": 0, "fake": 1}

rows = []  # (key, label, p_fake_mean)
aux = []   # для виводу p_max і кількості сегментів

for cname, lab in classes.items():
    for apath in list_audios(split_root / cname):
        p_mean, p_max, nseg = score_audio_file(
            apath,
            feature_extractor,
            model,
            target_sr=args.target_sr,
            segment_seconds=args.segment_seconds,
            fake_index=args.fake_index,
        )
        key = Path(apath).stem
        rows.append((key, lab, p_mean))
        aux.append((key, lab, p_mean, p_max, nseg))

if len(rows) == 0:
    print(f"{args.split.upper()} порожній")
    return

print(f"\nPer-audio probabilities on {args.split.upper()}:")
for key, lab, p_mean, p_max, nseg in aux:
    print(f"{key:40s}  label={lab}  P_fake_mean={p_mean:.3f}  P_fake_max={p_max:.3f}  segs={nseg}")

y_true = np.array([lab for _, lab, _ in rows], dtype=int)
pvals = np.array([p for _, _, p in rows], dtype=float)

best_thr, best_f1, best_acc = 0.5, -1.0, 0.0
for thr in np.linspace(0.1, 0.9, 17):
    y_pred = (pvals >= thr).astype(int)
    f1 = f1_score(y_true, y_pred) if len(set(y_true)) > 1 else 0.0
    acc = accuracy_score(y_true, y_pred)
    if f1 > best_f1:
        best_f1, best_thr, best_acc = float(f1), float(thr), float(acc)

print(f"\nBest threshold search ({args.split.upper()}):")
print(f"best_thr={best_thr:.2f} | audio_f1={best_f1:.3f} | audio_acc={best_acc:.3f}")

c = confusion_from_scores(y_true.tolist(), pvals.tolist(), best_thr)
m = metrics_from_confusion(c)
```

```
    print("\nCONFUSION (audio):")
    print(f"TP={c.tp} TN={c.tn} FP={c.fp} FN={c.fn}")

    print("\nMETRICS (audio):")
    print(
        f"acc={m.accuracy:.3f}  prec={m.precision:.3f}  rec={m.recall:.3f}  f1={m.f1:.3f}  "
        f"type1={m.err_type_1:.3f}  type2={m.err_type_2:.3f}"
    )

    save_scores_csv(args.out_csv, rows)
    print(f"\nSaved: {args.out_csv}")

    if args.split == "val":
        print("\nSet in Streamlit env:")
        print(f"AUDIO_MODEL_CKPT = {Path(args.ckpt).resolve()}")
        print(f"AUDIO_STRICT_THRESHOLD = {best_thr:.2f}")


if __name__ == "__main__":
    main()
```

Додаток В. ІЛЮСТРАТИВНА ЧАСТИНА



159

ІЛЮСТРАТИВНА ЧАСТИНА

СИСТЕМА ВИЯВЛЕННЯ ФЕЙКОВОГО МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

Виконав: студент 2 курсу групи 1БС-24м спеціальності 125 Кібербезпека та захист інформації

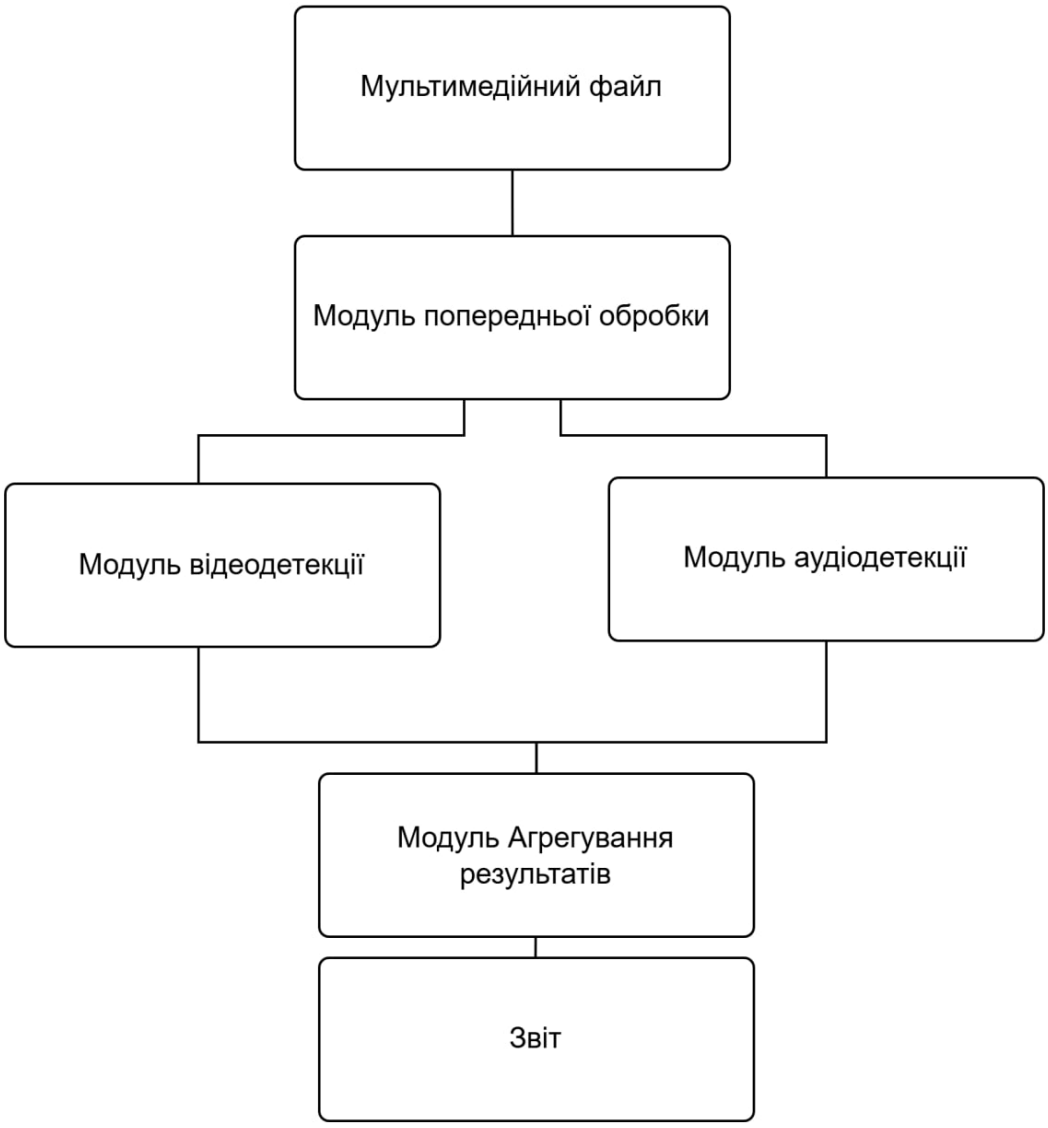
_____Назарій ЛЮДВА

Керівник: В. о. зав. кафедри ЗІ, д. т. н., проф.

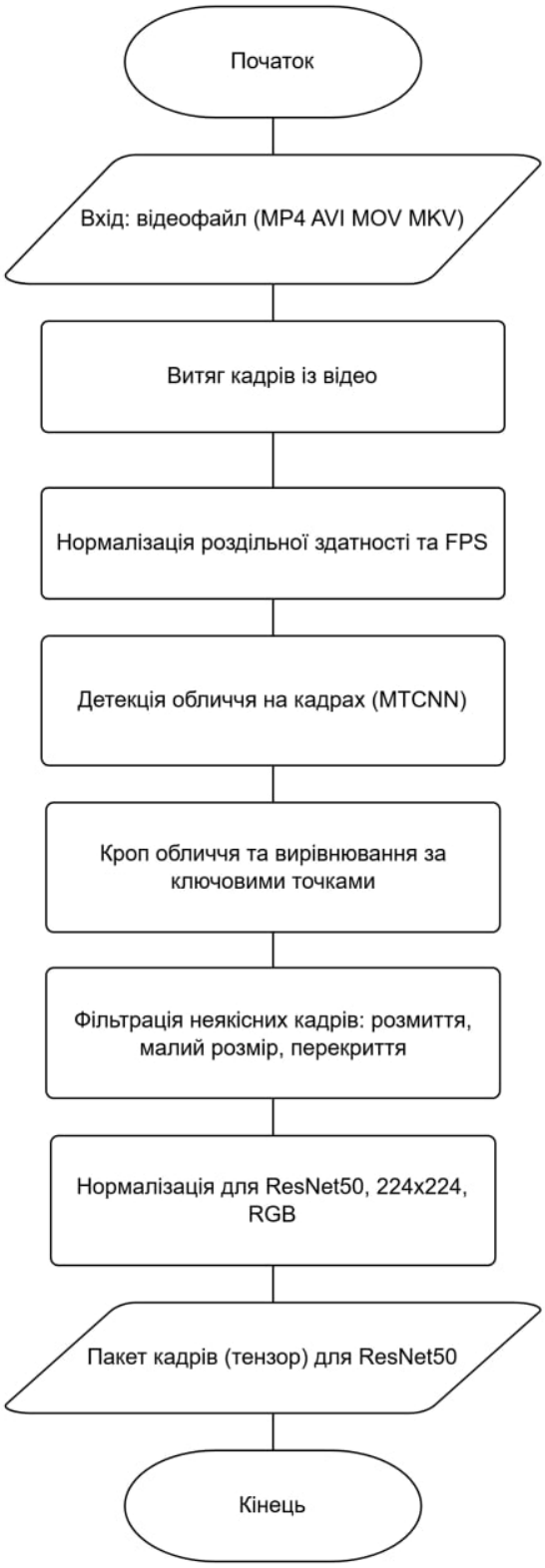
_____ Леонід КУПЕРШТЕЙН

«16» грудня 2025 р.

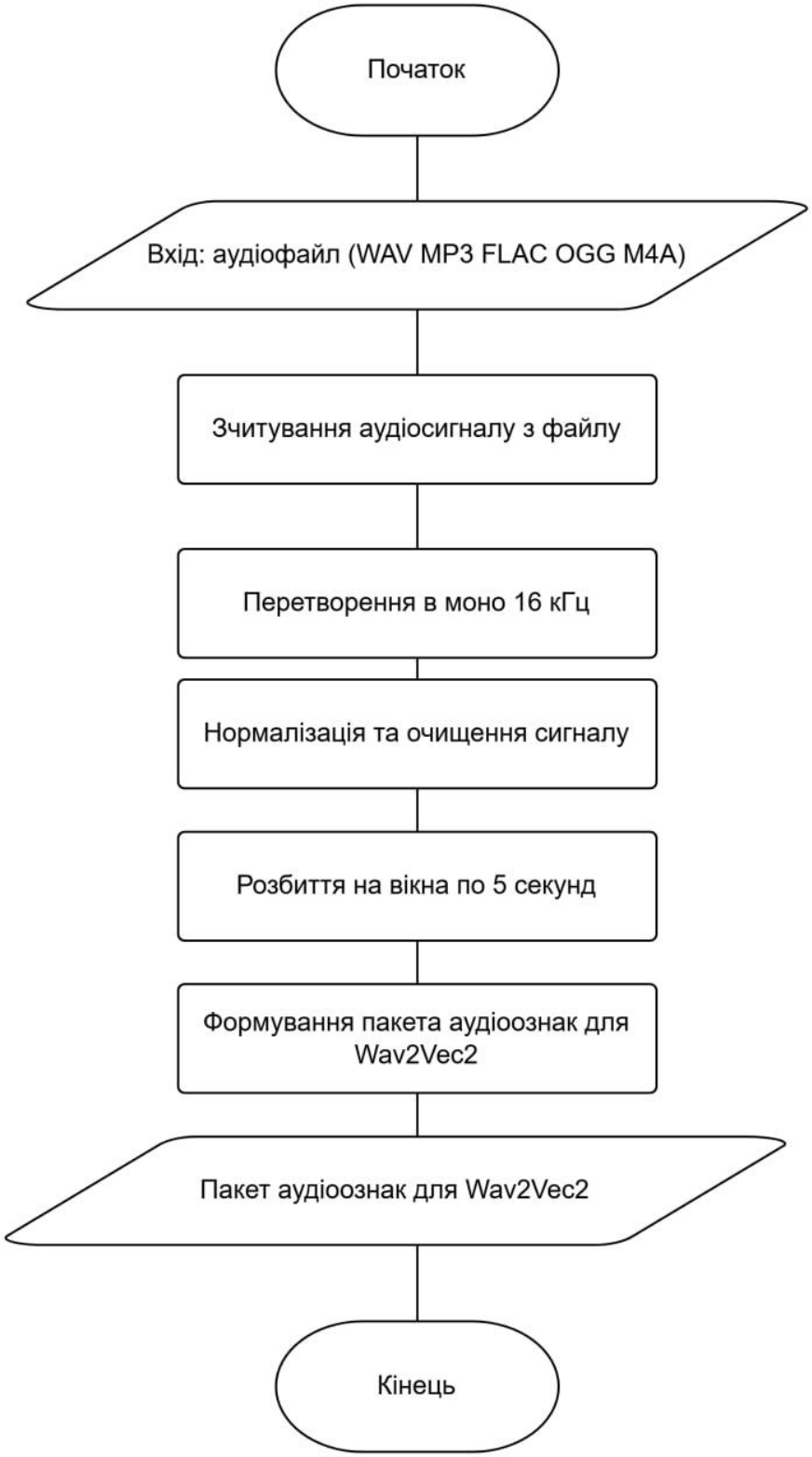
ЗАГАЛЬНА МОДУЛЬНА АРХІТЕКТУРА СИСТЕМИ МУЛЬТИМОДАЛЬНОЇ ДЕТЕКЦІЇ ДІПФЕЙКІВ



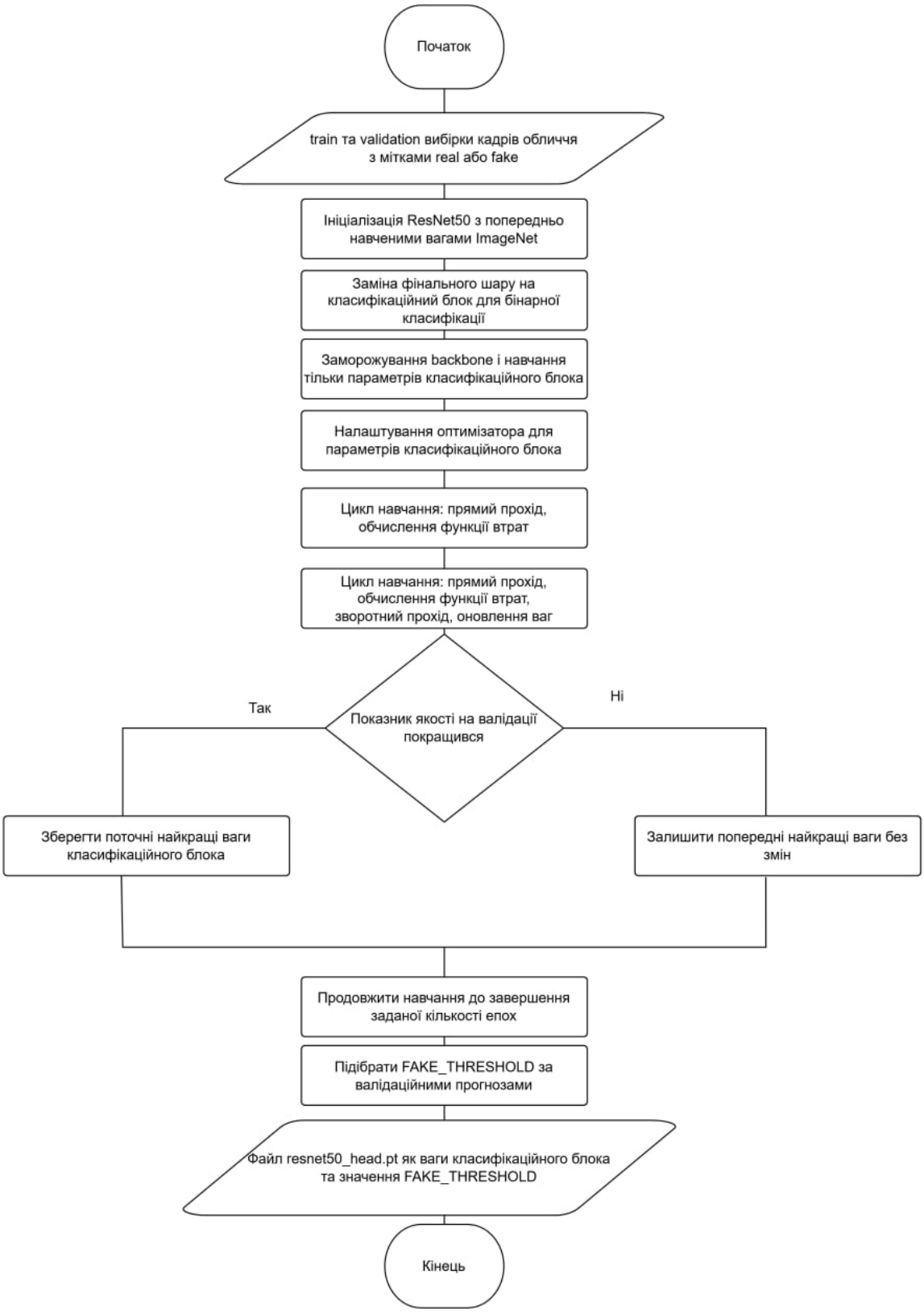
АЛГОРИТМ ПОПЕРЕДНЬОЇ ОБРОБКИ ВІДЕОДАНИХ ДЛЯ ПОДАЧІ У ВІДЕОМОДУЛЬ



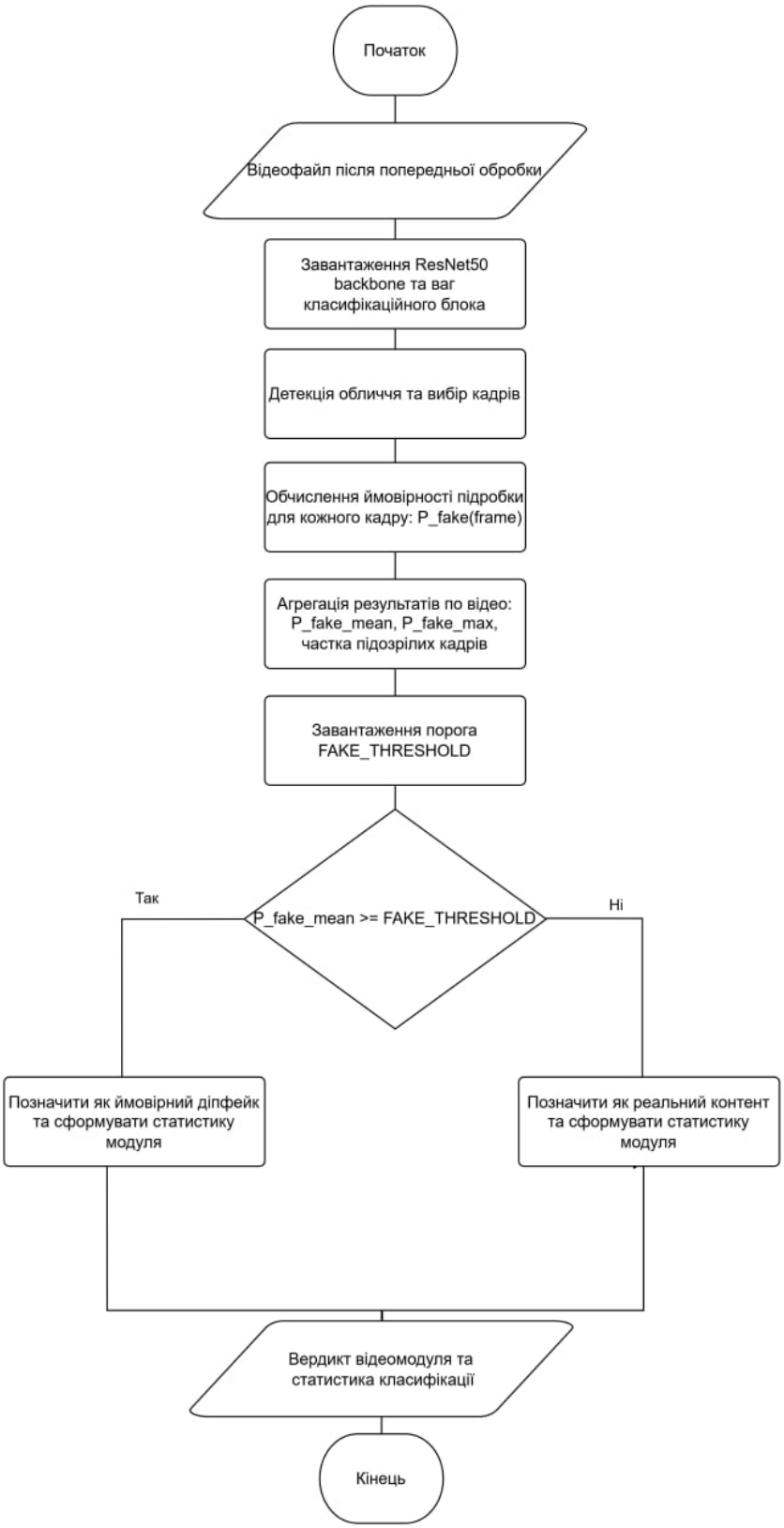
АЛГОРИТМ ПОПЕРЕДНЬОЇ ОБРОБКИ АУДІОСИГНАЛУ



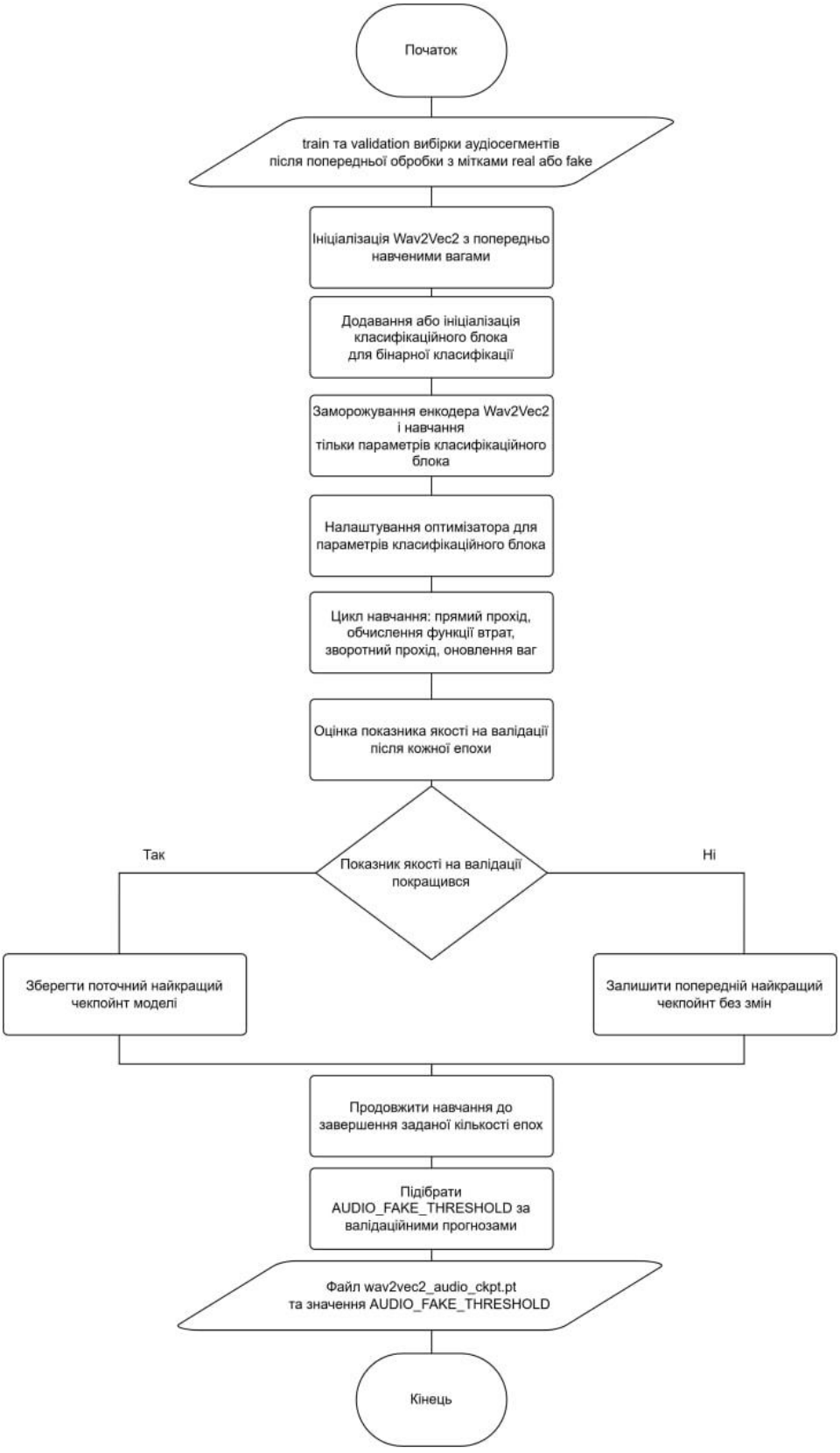
АЛГОРИТМ ДОНАВЧАННЯ ВІДЕОДЕТЕКТОРА



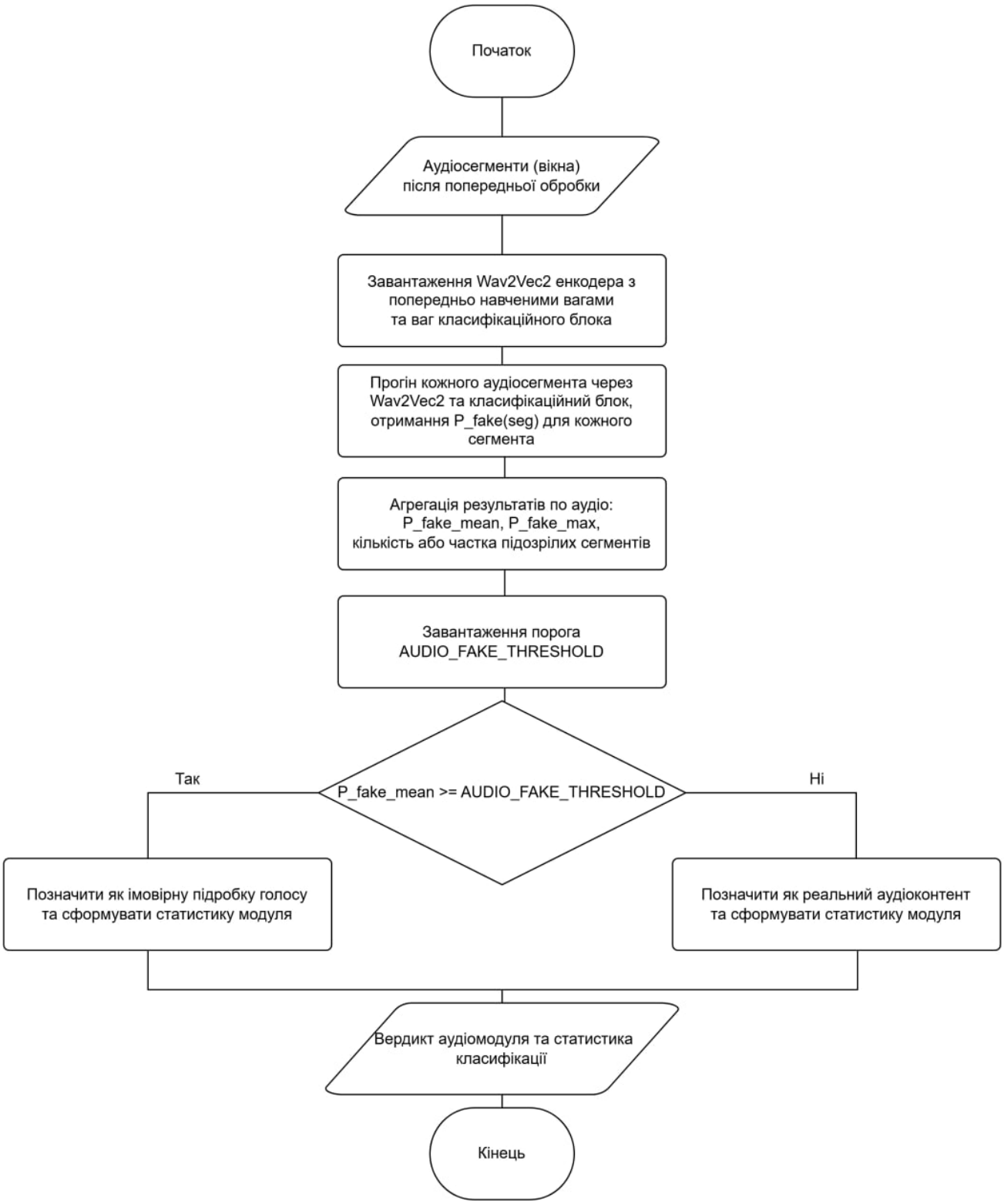
АЛГОРИТМ РОБОТИ ВІДЕОМОДУЛЯ RESNET50



АЛГОРИТМ ДОНАВЧАННЯ АУДІОМОДУЛЯ НА БАЗІ WAV2VEC



АЛГОРИТМ РОБОТИ АУДІОМОДУЛЯ В РЕЖИМІ ЗАСТОСУВАННЯ



АЛГОРИТМ РОБОТИ МОДУЛЯ АГРЕГУВАННЯ РЕЗУЛЬТАТІВ