Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ ВИЯВЛЕННЯ ДИПФЕЙКІВ У ЗОБРАЖЕННЯХ ТА ВІДЕО ДЛЯ ВЕРИФІКАЦІЇ ОСОБИ
(тема)

Виконав:
здобувач 2 року навчання,
групи ІНФм-24-1
Махно Р.А.
(прізвище, ініціали)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Інформатика
(повна назва освітньої програми)
Науковий керівник проф. Шафроненко А.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Махну Руслану Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів виявлення дипфейків у зображеннях та відео для верифікації особи

затверджена наказом університету від 14 листопада 2025 року № 1045Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 9 грудня 2025 р.
3. Вихідні дані до роботи методи виявлення дипфейків у зображеннях та відео, літературні джерела з медіафорензіки та відеоверифікації, програмні засоби Python (PyTorch, OpenCV, NumPy, dlib, tkinter), а також відео та зображення облич (реальні й підроблені) з відкритих наборів та власної вибірки для тестування моделі.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз сучасних методів і публікацій з дипфейк-детекції
2. Побудова покрокового алгоритму гібридного детектора та часової агрегації
3. Візуалізація конвеєра у вигляді блок-схем
4. Розробка програмного застосунку для виявлення дипфейків у відеоверифікації

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) схема постановки задачі, блок-схема гібридного детектора і часової агрегації, приклади кадрів «real/fake», анотовані кадри з prob_fake, скріншот головного вікна застосунку та приклади результатів роботи моделі.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	29.09.2025	
2	Аналіз завдання, підбір літератури	30.09.25-07.10.25	
3	Аналіз літератури з досліджуваної проблеми	08.10.25-14.10.25	
4	Дослідження особливості методів	15.10.25-20.10.25	
5	Розробка алгоритмів	21.10.25-27.10.25	
6	Програмна реалізація	28.10.25-05.11.25	
7	Обґрунтування отриманих результатів	06.11.25-11.11.25	
8	Оформлення пояснювальної записки	12.11.25-14.11.25	
9	Перевірка на нормоконтроль	09.12.25	
10	Перевірка на плагіат	15.12.25	
11	Рецензування	15.12.25-16.12.25	
12	Підготовка презентації та доповіді	16.12.25-18.12.25	
13	Занесення роботи в електронний архів	18.12.25	
14	Попередній захист кваліфікаційної роботи	18.12.25	

Дата видачі завдання 29 вересня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Шафроненко А.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 71 с., 1 табл., 10 рис., 56 джерела.

ВІДЕОДЕТЕКЦІЯ, ЕНТРОПІЙНО-ЗВАЖЕНЕ ГОЛОСУВАННЯ, КАЛІБРУВАННЯ ЙМОВІРНОСТЕЙ, C#/.NET, DCT/FFT, DEEPFAKEBENCH, DFDC, FACEFORENSICS++, FREQUENCY-AWARE FEATURES, LOG-ODDS INTEGRATION, ONNX, OPENCV, PAD/FAS.

Об’єктом дослідження є зображення та відео з обличчями користувачів у задачах віддаленої верифікації особи (KYC/доступ).

Предмет дослідження є методи й моделі виявлення дипфейків у кадрах та їхня часова агрегація до єдиного відеорішення з керованим балансом помилок FAR/FRR

Метою дослідження є підвищення надійності відеоверифікації особи шляхом розроблення і обґрунтування двох комплементарних підходів – гібридного кадрового детектора «простір+частота» та часової агрегації на рівні ролика.

Проведено огляд сучасних рішень deepfake-детекції. Сформовано частотно-просторові ознаки (2D-DCT/FFT із глобальними та локальними статистиками енергії) та просторові CNN-ембеддинги.

Наукова новизна роботи полягає у поєднанні частотних і просторових дескрипторів для кадрової детекції та у введенні узгодженої політики часової агрегації.

Взаємозв’язок з іншими роботами полягає у тому, що розробка спирається на публічні датасети та огляди з дипфейків і PAD/FAS.

У результаті дослідження запропоновано структуру програмної реалізації на C#/.NET із використанням OpenCV та інференсом моделей через ONNX.

ABSTRACT

Explanatory note to the qualification work: 71 pages, 1 table, 10 figures, 56 sources.

C#/.NET, DCT/FFT, DEEPFAKEBENCH, DFDC, ENTROPY-WEIGHTED VOTING, FACEFORENSICS++, FREQUENCY-AWARE FEATURES, LOG-ODDS INTEGRATION, ONNX, OPENCV, PAD/FAS, PROBABILITY CALIBRATION, VIDEO DETECTION.

The object of the research is images and videos with users' faces in remote identity verification tasks (KYC/access).

The subject of the study is methods and models for detecting deepfakes in frames and their temporal aggregation into a single video solution with a controlled balance of FAR/FRR errors.

The goal of the research is to improve the reliability of video identity verification by developing and substantiating two complementary approaches: a hybrid “space+frequency” frame detector and time aggregation at the video clip level.

A review of current deepfake detection solutions was conducted. Frequency-spatial features (2D-DCT/FFT with global and local energy statistics) and spatial CNN embeddings were formed.

The scientific novelty of the work lies in the combination of frequency and spatial descriptors for frame detection and the introduction of a consistent temporal aggregation policy.

The connection with other works lies in the fact that the development is based on public datasets and reviews of deepfakes and PAD/FAS.

As a result of the research, a structure for software implementation in C#/.NET using OpenCV and model inference via ONNX is proposed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AML – Anti-Money Laundering (протидія відмиванню коштів)

API/SDK – Application Programming Interface / Software Dev Kit (програмний інтерфейс застосунку / набір засобів розробника ПЗ)

BCE – Binary Cross-Entropy (втрати бінарної класифікації)

Brier – Brier-score (міра якості ймовірнісних прогнозів)

CE – Cross-Entropy (крос-ентропія)

CNN – Convolutional Neural Network (згорткова НМ)

DCT – Discrete Cosine Transform (дискретне косинус-перетворення)

DFD/Deepfake – підроблені обличчя/відео

DFDC – DeepFake Detection Challenge (датасет/змагання)

DFT/FFT – дискретне/швидке перетворення Фур'є

DoF/Blur – Depth of Field (глибина різкості / розмиття)

ECE – Expected Calibration Error (похибка калібрування)

EER – Equal Error Rate (точка рівних помилок)

FF++ – FaceForensics++ (датасет/бенчмарк)

GAN – Generative Adversarial Network (генеративно-змагальна мережа)

KYC – Know Your Customer (відеоверифікація клієнта)

MLP – Multi-Layer Perceptron (багатошаровий перцептрон)

PAD / FAS – Presentation Attack Detection / Face Anti-Spoofing

PII – Personally Identifiable Information (персонально ідентифікована інформація)

ROI – Region of Interest (область інтересу)

SNR/PSNR – відношення сигнал/шум / пікова SNR

TNR / Specificity – True Negative Rate (специфічність)

TPR / Recall – True Positive Rate (повнота)

TS – Temperature Scaling (температурне калібрування)

VAE – Variational Autoencoder (варіаційний автоенкодер)

ВСТУП

Цифрова трансформація суспільства супроводжується стрімким розвитком генеративних моделей, здатних синтезувати фотореалістичні зображення та відео облич. Такі технології – від GAN до сучасних дифузійних моделей – відкривають широкі можливості для медіавиробництва, однак одночасно створюють суттєві ризики для систем верифікації особи у фінансовому секторі (KYC/AML), електронних сервісах, доступі до критичної інфраструктури та публічному управлінні. Поширення «дипфейків» підриває довіру до відеоканалів комунікації, ускладнює процес перевірки клієнтів та може призводити до прямих фінансових і репутаційних втрат. За цих умов зростає потреба у науково обґрунтованих, відтворюваних та практично застосовних методах автоматичного виявлення підробок, які можна інтегрувати в реальні KYC-процеси.

Актуальність теми зумовлена трьома чинниками. По-перше, якість синтетичних відео нині достатня для обходу простих механізмів контролю «живості» і базових класифікаторів, навчених на на обмежених або застарілих датасетах. По-друге, у реальних сценаріях відеоверифікації значну роль відіграють «польові» виклики: багаторазове перекодування, артефакти стримінгу, змінне освітлення, шум, часткові оклюзії обличчя, робота з мобільними пристроями різної якості. Це робить оцінювання на окремих кадрах нестабільним, а прийняття рішення лише за одиничним зображенням – ненадійним. По-третє, екосистема генераторів дипфейків швидко еволюціонує – з’являються нові моделі, нові типи артефактів, нові стилі обробки відео, що додатково знижує узагальнюваність традиційних підходів, орієнтованих на фіксований набір атак.

За таких умов особливої ваги набувають методи, які поєднують різні джерела інформації про підробку – просторові, частотні та часові ознаки. Просторові CNN-детектори добре фіксують локальні текстурні й геометричні дефекти зшивання обличчя, але чутливі до сильного стиснення та розмиття.

Частотні та frequency-aware підходи аналізують спектральний вміст кадру і виявляють нетипові розподіли енергії, спричинені архітектурою генераторів, однак потребують уважного інжинірингу та коректного протоколу оцінювання. Часовий вимір дозволяє враховувати узгодженість чи навпаки «дерганість» міміки, стабільність артефактів у часі, а отже – приймати рішення на рівні всього ролика, а не одиничного кадру. Поєднання цих трьох компонент у єдиному конвеєрі створює передумови для більш робастних систем виявлення дипфейків у реальних умовах відеоверифікації.

Об’єктом дослідження в даній роботі є процес автоматизованої відеоверифікації особи за зображенням обличчя в умовах можливих атак з використанням дипфейків. Предметом дослідження є методи побудови гібридних частотно-просторових детекторів дипфейків та алгоритмів часової агрегації, що перетворюють послідовність кадрових оцінок на надійне відеорішення. Метою роботи є розробка й дослідження комп’ютерної моделі виявлення дипфейків, яка поєднує просторовий CNN-детектор, частотні ознаки та статистично обґрунтовані схеми інтеграції кадрових ймовірностей (лог-правдоподібна інтеграція, ентропійно-зважене голосування), а також її програмна реалізація у вигляді прикладного програмного забезпечення для аналізу відео у форматі відеоверифікації KYC.

Для досягнення цієї мети в роботі пропонується, по-перше, проаналізувати сучасні підходи до виявлення дипфейків у зображеннях та відео, порівняти їх сильні та слабкі сторони з точки зору практичного використання у KYC/AML-процесах. По-друге, розробити гібридну кадрову модель, що поєднує просторові та частотні ознаки обличчя і здатна працювати поверх існуючих CNN-архітектур. По-третє, запропонувати й дослідити методи часової агрегації, які враховують як накопичення доказів у часі, так і невизначеність кадрових передбачень, і на цій основі формують відеорівневе рішення. А також, реалізувати розроблену модель у середовищі Python із використанням PyTorch та OpenCV, побудувати десктопний інструмент з графічним інтерфейсом.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ ВИЯВЛЕННЯ ДИПФЕЙКІВ

1.1 Базові просторові CNN-детектори

Базова лінія детекції дипфейків спирається на аналіз окремих кадрів у піксельному просторі: спочатку виконується детекція обличчя, його геометричне вирівнювання (alignment), нормалізація масштабу та яскравості, після чого отримане зображення подається на CNN-класифікатор із виходом fake/real. У FaceForensics++ саме такий pipeline закріплено як еталон: автори показують, що від якості препроцесингу (точна детекція, стабільний алайнмент, єдина система координат для всіх облич) безпосередньо залежить продуктивність будь-якої подальшої моделі та коректність порівняння різних архітектур [1]. Нормалізація до стандартної роздільної здатності та усунення фону зменшують внутрішньокласну варіативність і змушують мережу зосередитися саме на артефактах маніпуляції, а не на випадкових деталях сцени [1].

У ролі базового класифікатора в [1] широко використовували Xception як глибоку CNN з глибинно-роздільними згортками, здатну ефективно моделювати локальні та глобальні патерни обличчя. Для легких і ресурсно-обмежених сценаріїв застосовують компактні MesoNet-архітектури, орієнтовані на локальні текстури й мезорівневі патерни (область очей, рота, щік) [1]. Такі моделі мають меншу глибину, але краще пристосовані до роботи в режимі реального часу й на вбудованих пристроях. Типовими метриками оцінювання є AUC, accuracy, EER, а також ROC- і DET-криві, що дозволяють аналізувати компроміс між TPR і FPR в задачах верифікації [1].

Навчання кадрових CNN зазвичай ведуть на відкритих наборах даних: FF++ надає набір маніпульованих та реальних кліпів з різними типами маніпуляцій і рівнями компресії, що дало можливість сформувати «еталонний» протокол порівняння моделей [1]. DFDC, у свою чергу, містить значно більше акторів, різноманітні генератори та «польові» умови запису – зміни освітлення, поз, якості камери, сильні артефакти стиснення [2]. Саме на DFDC чітко видно,

що прості кадрові CNN, навчені в «тепличних» умовах, різко просідають за межами вихідного домену: з’являється сильна чутливість до доменного зсуву, компресії й розмиття, якщо моделі спеціально не адаптувати й не доповнювати за рахунок агресивних аугментацій або доменно-орієнтованого донавчання [2].

Сучасні протоколи оцінювання (зокрема ті, що орієнтуються на рекомендації DeepfakeBench та подібних бенчмарків) наполягають на відсутності «перетоку акторів» між train/test, єдиних метриках на відеорівні та публічних скриптах препроцесингу й оцінювання, щоб уникнути прихованих переваг окремих робіт [3]. Оскільки поодинокий кадр може бути як дуже інформативним (яскраві артефакти генерації), так і майже «чистим», кадрова оцінка за своєю природою нестійка. Тому в більшості реальних систем рішення приймають на рівні всього ролика: від простого усереднення й голосування більшістю до підсумовування лог-імовірностей/лог-шансів, що добре узгоджується зі статистичною теорією прийняття рішень і дозволяє підвищити стійкість до окремих хибних кадрів [3].

Сильні сторони просторових CNN-детекторів – технологічна простота, висока швидкодія, зрозумілий інжиніринг і наявність напрацьованих протоколів та наборів даних. Водночас їхні обмеження добре задокументовані: чутливість до компресії та розмиття, схильність переобучуватися на специфічні артефакти конкретних генераторів і відсутність явної чутливості до спектральних спотворень, які є наслідком згорткових upsampling-операцій у генеративних моделях [3–4]. Саме ці недоліки мотивують перехід до частотних і гібридних підходів, де просторова CNN виступає лише одним із компонентів більш складної частотно-просторової або мультимодальної системи.

1.2 Частотні та frequency-aware методи

Ключова ідея частотних підходів полягає в тому, що сучасні генератори (GAN- та дифузійні моделі) можуть дуже переконливо відтворювати піксельну

структуру обличчя, але залишають нетипові спектральні патерни. У частотній області це проявляється як перерозподіл енергії між низькими та високими частотами, поява періодичних «гребінців» від апсемплінгу, а також аномалії фазових співвідношень, які майже не помітні у звичайному зображенні, проте добре фіксуються у DCT/FFT-домені. Використати цю асиметрію пропонує лінія робіт «thinking in frequency», де CNN навчається працювати не лише з пікселями, а й з їхнім спектральним представленням [4]. Подальший розвиток цієї ідеї – frequency-aware архітектури, у яких спектральна чутливість вбудована безпосередньо у блоки мережі. Показано, що таке поєднання просторових і частотних сигналів покращує стійкість до нових генераторів і доменного зсуву [5].

У типовому конвеєрі відеоверифікації кадр після детекції та вирівнювання обличчя розгалужується щонайменше на дві гілки. Просторова гілка – це звичайна CNN, яка вловлює геометрію, текстури, локальні дефекти змішування (некоректні межі вставки, згладжені зморшки, артефакти навколо очей і рота). Частотна гілка перетворює зображення у DCT або FFT, після чого з отриманого спектра будуються карти ознак: енергетичні карти для різних діапазонів частот, радіальні або кільцеві профілі, локальні спектральні патчі. Далі відбувається злиття гілок – це може бути простий concatenation, attention-модуль або невеликий MLP, що навчається інтегрувати просторові та спектральні індикатори під час формування кадрового рішення fake/real [5]. На рівні відео до цього додаються уже відомі схеми часової агрегації (усереднення лог-імовірностей, лог-шансів, голосування), що дозволяє перетворити нестійкі кадрові оцінки на стабільний відеоскора.

У роботах типу F³-Net робиться наголос на тому, щоб «змусити мережу думати у частотній області»: окремі блоки виділяють внутрішні спектральні компоненти, підкреслюючи високочастотні мікроартефакти (шумоподібні структури, регулярні хвилясті патерни, залишкові сліди апсемплінгу), після чого ці підказки використовуються разом із просторовими ознаками [5]. У більш нових frequency-aware підходах 2024 року вводяться спеціальні фільтри та вагові

схеми, які посилюють внесок частот, характерних саме для синтетичних зображень, а також застосовуються стратегії доменної регуляризації, що покращують узагальнюваність на «невидані» генератори й нові набори даних [5]. Експерименти показують, що такі моделі суттєво менше деградують при переході з відносно «чистих» лабораторних умов FF++ до більш «польового» DFDC [5].

Низка незалежних досліджень продемонструвала, що генеративні мережі зі згортковим апсемплінгом систематично спотворюють спектральний розподіл: високі частоти або надмірно згладжуються, або, навпаки, з'являються штучні піки, пов'язані з періодичністю фільтрів. Такі зсуви не обов'язково легко інтерпретуються візуально, але в частотній області утворюють стійкі «підписи», які добре відділяються від спектра реальних зображень [5]. На практиці це особливо корисно у «польових» умовах: агресивне JPEG-стискання чи стрімінг часто прибирають дрібні просторові деталі, але не повністю нівелюють глобальний спектральний профіль. Відповідно, частотна гілка зберігає дискримінацію там, де просторовій бракує інформації, і дає змогу підтримувати прийнятну точність навіть на роликах низької якості [5].

Разом із тим, частотні методи потребують уважного інжинірингу. По-перше, при побудові спектральних ознак важливо уникати «витоку домену»: різні набори даних повинні оброблятися одним і тим самим препроцесингом, а протоколи навчання/тестування мають виключати перетин акторів і сцен між train/val/test, інакше модель може вивчити не глибинні частотні закономірності, а побічні особливості конкретного датасета [3-5]. По-друге, доцільно застосовувати аугментації, які варіюють як просторові, так і спектральні властивості сигналу: зміну рівня JPEG-квантування, додавання шуму, блюр, невеликі колірні зсуви. Це знижує ризик того, що мережа запам'ятає один-єдиний шаблон компресії, притаманний, наприклад, тільки FF++ [5]. По-третє, практика показує, що небажано будувати класифікацію лише на глобальних спектральних статистиках (усереднений спектр з усього обличчя): набагато

стабільніше поводяться локальні карти частот (patch-wise DCT/FFT), що зберігають прив’язку до просторового розташування артефактів [5-6].

З позиції відео-KYC частотна гілка особливо корисна у сценаріях низької якості: перекодування, змаз, слабке освітлення, зйомка на фронтальні камери мобільних пристроїв. У таких умовах просторові дефекти можуть бути частково замасковані, тоді як спектральний «відбиток» синтетики все ще зберігається. Просторова CNN, навпаки, краще ловить семантично узгоджені дефекти – неконсистентні рухи рот/очей, геометричні спотворення, артефакти меж злиття обличчя з фоном. Їх поєднання формує робастний кадровий детектор, який далі узагальнюється у часі за допомогою усереднення, лог-правдоподібнісної інтеграції чи більш складних схем голосування. Оцінювання таких гібридних рішень доцільно проводити на відкритих бенчмарках FF++, DFDC та комплексних фреймворках на кшталт DeepfakeBench із відтворюваними скриптами препроцесингу й тестування – це забезпечує чесне порівняння різних підходів і коректний вибір робочих порогів для цільових сценаріїв відеоверифікації [5-6].

1.3 Часове узагальнення відеорішень

У відеоверифікації рішення має прийматися по ролику, а не за одиничним кадром. Користувач у KYC-сценаріях надає коротке селфі-відео, і система повинна видати однозначну відповідь «пройти / не пройти перевірку», тоді як кадрові CNN (на базі Xception, MesoNet, MobileNetV2, EfficientNet тощо) генерують лише послідовність оцінок для окремих фреймів [6–9]. Навіть високоточні кадрові детектори демонструють суттєві коливання цих оцінок через зміни поз, освітлення, ступеня компресії, локальні збої вирівнювання або пропуски кадрів. Частина фреймів може бути практично «чистою», інша – містити виразні артефакти генерації. Тому пряме порівняння роликів за

максимумом/мінімумом кадрового скору призводить до нестабільної поведінки й великої кількості хибних спрацювань на реальних даних [6].

Щоб отримати стійке відеорішення, поверх будь-якого кадрового класифікатора вводиться часовий шар агрегації – послідовність скорів перетворюється на єдину скалярну оцінку ролика. У літературі та практиці сформувалися два базові, комплементарні підходи: лог-правдоподібна інтеграція, яка трактує кадрові оцінки як «докази» та підсумовує їх у вигляді лог-шансів (log-odds), і ентропійно-зважене голосування, де враховується невизначеність модельних передбачень, а кадри з високою ентропією автоматично мають менший вплив [6-7]. Обидві схеми спираються на класичні ідеї теорії прийняття рішень в інформаційних системах та статистичного тестування гіпотез, де послідовні спостереження комбінуються в один інтегральний критерій [10–11]. Подібні принципи давно використовуються у відеокласифікації та action recognition: фрагменти ролика обробляються CNN, а їхні скори агрегуються в єдиний відеолейбл [12]. У контексті дипфейків ці підходи дозволяють перетворити «шумну» послідовність кадрових оцінок на стабільний показник, який можна інтерпретувати як загальний рівень підозри щодо відео та налаштувати під задану робочу точку FF++/DFDC/DeepfakeBench.

1.3.1 Лог-правдоподібна інтеграція

Ідея полягає у тому, аби кадрову «ймовірність підробки» розглядати як свідчення на користь класу fake відносно real. Ці свідчення сумуються по всіх валідних кадрах ролика. З погляду статистичного прийняття рішень це відповідає накопиченню лог-шансів (log-odds) – саме так у відеокласифікації давно узагальнюють фрагментарні спостереження в єдине рішення.

Практично лог-правдоподібна інтеграція дає кілька важливих властивостей. По-перше, поодинокі «шумні» кадри з аномально високим або низьким скором розчиняються в сумі: щоб отримати хибне відеорішення,

потрібно накопичити багато односторонніх свідчень, що знижує чутливість до випадкових збоїв детектора. По-друге, ця схема є повністю архітектурно-агностичною: на вхід можуть подаватися скори як від чисто просторових CNN, так і від частотних або гібридних детекторів, а також від ансамблів моделей [9-11]. По-третє, робота з лог-шансами спрощує налаштування порога: на валідаційному наборі FF++ або DFDC можна підібрати єдине значення, яке потім переноситься на тестові ролики й інші протоколи практично без зміни формули агрегації [10].

Важливим аспектом є попередня обробка кадрових скорів перед інтеграцією. У реальних системах корисно: відкидати кадри, де не вдалося стабільно вирівняти обличчя, обмежувати p_t у діапазоні $[\varepsilon, 1-\varepsilon]$, щоб уникнути числових переповнень лог-odds, за необхідності застосовувати калібрування ймовірностей на валідаційному наборі, щоб мережа не повертала систематично переоцінені або недооцінені значення [10-11]. З точки зору реального впровадження в .NET/ONNX-пайплайні лог-правдоподібна інтеграція реалізується дуже просто: система лише акумулює скалярні значення ℓ_t у циклі обробки кадрів і наприкінці порівнює суму з порогом, що майже не додає обчислювальних витрат до основного інференсу CNN.

1.3.2 Ентропійно-зважене голосування

Лог-правдоподібна інтеграція припускає, що всі кадрові оцінки однаково надійні. На практиці це не так: частина кадрів сильно зіпсована рухом, розмиттям або компресією, що підвищує невизначеність моделі. Ентропійно-зважене голосування явно враховує цю невизначеність: кожному кадру призначається вага, що залежить від ентропії його бінарного розподілу. Чим впевненіше модель (низька ентропія), тим більший внесок кадру у фінальне рішення. Сумнівні кадри з близькими до 0.5 ймовірностями майже не впливають на підсумок [10].

На практиці застосовують дві близькі реалізації:

– зважене голосування за клас (кадр віддає голос fake або real, помножений на вагу);

– зважене підсумовування ймовірностей (без бінаризації, але з вагами).

У «польових» умовах DFDC саме така стратегія виявляється особливо корисною: стрімінг-стискання, різкі рухи, слабке освітлення створюють кластер кадрів з високою ентропією, і зважування природно пригнічує їхній негативний вплив, зберігаючи внесок тих фреймів, де артефакти дипфейку проявляються виразніше. На чистіших наборах FF++ ефект може бути менш помітним, але ентропійне зважування все одно зменшує варіативність відеоскорів між роликами з різною динамікою сцени [10-11].

У підсумку обидві часові схеми – лог-правдоподібна інтеграція та ентропійно-зважене голосування – виступають легкими, архітектурно-незалежними надбудовами над будь-яким кадровим детектором. Вони дозволяють перевести нестійкі кадрові оцінки в інтерпретоване відеорішення, узгоджене зі статистичною теорією прийняття рішень та вимогами сучасних бенчмарків (FF++, DFDC, DeepfakeBench) до прозорого й відтворюваного оцінювання якості детекції дипфейків [10–12].

1.4 Допоміжні підходи для верифікації

У практичних системах відеоверифікації сам по собі детектор дипфейків рідко працює ізольовано. Його підсилюють допоміжні модулі, що або перевіряють «живість» особи, або шукають міжмодальні невідповідності, або локалізують ділянки ймовірної підміни. У сучасних оглядах з медіафорензіки й дипфейк-детекції підкреслюється, що поєднання декількох ортогональних каналів (просторові/частотні CNN, аналіз «живості», аудіо-візуальна узгодженість, локалізація артефактів) суттєво підвищує стійкість до доменних зсувів та ускладнює обходи системи, оскільки атакувальнику доводиться одночасно імітувати кілька незалежних сигналів [22–23]. Окремий клас

складають PAD/FAS-підходи (presentation attack detection / face anti-spoofing), які історично розвивалися паралельно до дипфейк-детекції та орієнтовані на виявлення друкованих фото, відтворених відео на екранах, масок тощо. У задачах KYC їх результати корисно комбінувати з власне дипфейк-скором

1.4.1 Liveness / rPPG

Liveness-перевірка на основі rPPG (remote photoplethysmography) покладається на слабкі періодичні зміни відбитого світла від шкіри, зумовлені серцевим ритмом. Ці коливання присутні у реальному відео обличчя, мають характерну частоту в діапазоні людського пульсу та певну просторову структуру по регіонах обличчя. У синтетичних або відтворених відео (друк/екран, накладення облич) цей сигнал часто порушується – амплітуда зменшується, спектр спотворюється [22–23].

У контексті KYC rPPG виступає незалежною ознакою «живості», яка погано відтворюється на підробних чи відтворених відео. Для друкованих фото або відео, програних з екрана, rPPG-сигнал або відсутній, або має аномальний спектр. Для GAN-/дифузійно згенерованих облич генератор зазвичай не моделює реалістичні мікроколивання шкіри, особливо якщо відео коротке й зняте в нестандартних умовах освітлення [22–24].

Сучасні роботи з PAD/FAS узагальнюють rPPG у ширшу рамку multi-cue liveness: rPPG-сигнал комбінують з текстурними ознаками шкіри, аналізом відблисків в очах, оцінкою глибини (stereo/IR-камери) та мікрорухами голови [23]. Для практичних систем це означає, що модуль rPPG можна реалізувати як окремий канал, який повертає свій «живість-скор». Далі він агрегується з дипфейк-скором на рівні відео (наприклад, через лог-правдоподібну інтеграцію або ентропійно-зважене голосування). Така багатоканальна схема дозволяє компенсувати слабкі місця одного детектора за рахунок іншого: якщо, наприклад, просторово-частотний дипфейк-детектор працює гірше на

низькоякісних роликах, rPPG-канал все ще може виявити відсутність фізіологічного сигналу.

1.4.2 Аудіо-візуальна узгодженість

Ще один ортогональний сигнал – аудіо-візуальна узгодженість. Дипфейк часто «ламається» на синхроні: рух губ, щелепи, мікроміміка і акустика мовлення мають бути узгодженими в часі. Якщо обличчя було підмінено незалежно від аудіо, або якщо синтетичне відео згенеровано без точного урахування часової структури мовлення, виникають дрібні, але систематичні невідповідності. Оглядові роботи з дипфейк-детекції відзначають, що саме міжмодальні невідповідності – один із найперспективніших напрямів для підсилення існуючих відеодетекторів [22–23].

У задачах детекції синтетичних облич це дає додатковий канал контролю, особливо коли відео стисло або змазано, і просторові патерни менш інформативні. Для підробок, згенерованих «під аудіо» (audio-driven facial reenactment), аудіо-візуальна узгодженість може бути високою, але тоді вступають у гру комбіновані схеми: частотно-просторовий дипфейк-детектор + модуль оцінки синхрону. У випадку, коли підмінюється не лише обличчя, а й голос (voice spoofing), додаються ASV-модулі (automatic speaker verification), які порівнюють голос із референсним шаблоном користувача й оцінюють, чи належить аудіосигнал заявленій особі [23].

У практичних KYC-сценаріях аудіо-візуальна узгодженість зазвичай інтегрується на рівні відео разом із іншими скор-каналами (кадровий дипфейк-детектор, rPPG-модуль, PAD-ознаки). Система може вимагати, щоб усі канали одночасно були «в нормі», або ж агрегувати їх у спільний ризик-скор із подальшим пороговим прийняттям рішення. Такий мультиканальний дизайн дозволяє протистояти більш широкому спектру атак: від простого програвання

записаного відео до складних GAN-/дифузійних підробок, у яких атакувальник намагається імітувати як зовнішність, так і манеру мовлення користувача [23].

1.5 Постановка задачі дослідження

Таким чином, виявлення дипфейків у зображеннях та відео для верифікації особи є актуальним науково-прикладним завданням. Прийнято рішення щодо розроблення та дослідження методу відеодетекції дипфейків, що поєднує частотно-просторові ознаки на рівні кадру та часову агрегацію рішень на рівні ролика (лог-правдоподібна інтеграція і ентропійно-зважене голосування). Передбачається створення прототипу програмного застосунку з відтворюваними протоколами оцінювання (FF++, DFDC, DeepfakeBench) і можливістю інтеграції в конвеєр відео-KYC.

Об’єктом дослідження є процес верифікації особи за відеозображенням в умовах наявності штучно згенерованого контенту (дипфейків).

Предметом дослідження є методи та моделі детекції дипфейків, зокрема частотно-просторова екстракція ознак і часові схеми узагальнення кадрових оцінок.

Метою дослідження є підвищення надійності відеоверифікації особи шляхом розроблення та обґрунтування методу детекції дипфейків, стійкого до перекодувань, «польових» артефактів і доменних зсувів, із формальним налаштуванням порогів під робочі режими KYC.

Для досягнення мети необхідно вирішити такі завдання:

– провести огляд і систематизацію сучасних підходів до детекції дипфейків (просторові CNN, частотні та frequency-aware методи, гібридизація, часові агрегації);

– сформувати вимоги до методу для сценаріїв відео-KYC (робастність до компресії/розмиття, узагальнюваність на «невидані» генератори, відтворюваність протоколів);

– розробити частотно-просторову схему екстракції ознак на рівні кадру та описати її математичні засади;

– розробити дві часові стратегії узагальнення: лог-правдоподібну інтеграцію та ентропійно-зважене голосування;

– побудувати детальний покроковий алгоритм роботи всього конвеєра і подати його блок-схемою;

– реалізувати прототип програмного застосунку з модулем детекції/вирівнювання обличчя, екстракцією ознак, часовою агрегацією та інструментами калібрування порогів;

– визначити протоколи експериментів (in-domain, cross-domain, robustness-сценарії) та метрики оцінювання на відеорівні ;

– провести експериментальне дослідження на відкритих наборах (FF++, DFDC, DeepfakeBench) і проаналізувати внесок кожного компонента (простір/частота/агрегація);

– сформулювати практичні рекомендації щодо інтеграції методу в системи відеоверифікації та напрямки подальших досліджень.

2 МАТЕМАТИЧНІ МОДЕЛІ ВЕРИФІКАЦІЇ ОСОБИ

2.1 Гібридний кадровий детектор «простір+частота»

У моделі використовується поєднання просторових та частотних ознак обличчя. Вхідним сигналом є кадр відео I. Після детекції та аффінного вирівнювання обличчя отримуємо нормалізоване зображення в канонічній позі (рис. 2.1). На цьому етапі усуваються зсув, масштаб, поворот, що зменшує внутрішньокласну варіативність та дозволяє надалі зосередитися саме на артефактах підміни, а не на геометричних відмінностях поз. Сучасні підходи до побудови ознакового простору для класифікації зображень використовують ортогональні базиси, ієрархічні системи ознак та частотний аналіз, що добре узгоджується з вибором DCT/FFT як основи частотної гілки [26].

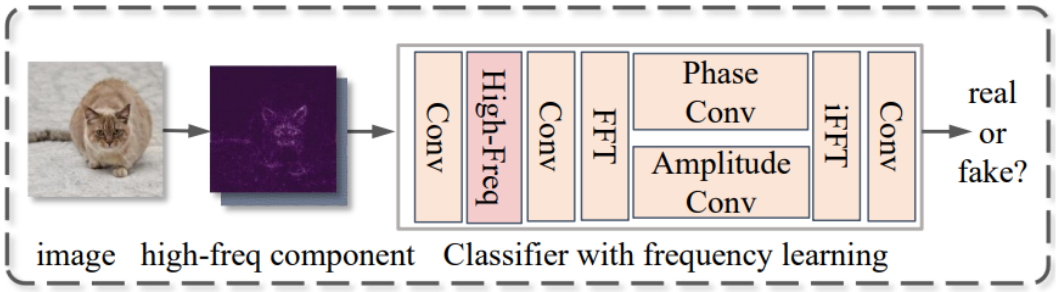


Рисунок 2.1 –Узагальнена схему конвеєра

Далі паралельно будуються просторове подання та частотне подання, які зливаються в єдиний вектор ознак, а потім оцінюється імовірність дипфейку. Просторова гілка відповідає за локальні текстури, геометрію та семантичні деталі (очі, рот, межі зшивання), тоді як частотна гілка фокусується на спектральних підписах синтезу (перерозподіл енергії між низькими/високими частотами, регулярні «гребінці» від апсемплінгу, нетипові високочастотні компоненти). Роботи, що аналізують спектральні спотворення в CNN-генераторах і використовують частотні ознаки для розпізнавання дипфейків,

показують, що саме спектр дає стійкі ознаки, які важко повністю замаскувати [29].

Попереднє вирівнювання описуємо оператором:

$$x = A(I), \tag{2.1}$$

де I – вхідний RGB-кадр;

x – нормалізоване (обрізане й вирівняне) зображення обличчя фіксованого розміру;

$A(\cdot)$ – реалізує детекцію обличчя й лінійно-аффінне перетворення за контрольними орієнтирами (очі, ніс, кути рота).

У такий спосіб гібридний кадровий детектор «простір+частота» поєднує переваги класичних CNN (чутливість до семантично зрозумілих артефактів у піксельному просторі) та частотних методів (виявлення спектральних підписів генератора, стійкість до повторного стиснення й розмиття). Використання ортогональних функцій у формуванні простору частотних ознак, доповнених ієрархічними та локально-глобальними характеристиками, добре узгоджується з сучасними підходами до структурного опису зображень [26-28], Оцінювання таких гібридних детекторів у рамках бенчмарків типу DeepfakeBench дозволяє отримати зіставні результати для різних архітектур і налаштувати їх під цільові сценарії відеоверифікації [28].

2.1.1 Частотні ознаки

Нехай Y– канал яскравості (наприклад, компонент Yу просторі YCbCr) зображення **x**. 2D-DCT [26] у матричній формі обчислюється як

$$C = D\ Y\ D^T. \tag{2.2}$$

де Y – матриця яскравостей;

C – матриця DCT-коефіцієнтів;

D – ортонормована матриця косинусів (рядки якої відповідають базисним косинусоїдальним функціям).

Такий запис еквівалентний класичному визначенню 2D-DCT через подвійні суми й відповідає підходам, у яких простір ознак формується на основі системи ортогональних функцій [28].

У матриці C виділяємо смуги частот $\{B_k\}$, які відповідають низьким, середнім та високим просторовим частотам. Низькочастотна смуга (область навколо відображає загальну структуру та освітлення, високочастотна – дрібні деталі й шум. Для синтетичних зображень, отриманих за допомогою генеративних моделей, показано, що спектральний розподіл істотно відрізняється від природних зображень: зокрема, через особливості апсемплінгу та згорток у генераторі [29].

Для кожної смуги рахуємо енергетичну статистику:

$$E_k = \sum_{(u,v)\in B_k} |C_{u.v}|^y. \tag{2.3}$$

де E_k – енергетична характеристика k-тої частотної смуги;

B_k – множина індексів, що належать k-тій смузі у DCT-спектрі;

$C_{u,v}$ – коефіцієнт двовимірного DCT для частоти з індексами;

y – степеневий параметр, що керує чутливістю до великих коефіцієнтів.

Таким чином, набір $\{E_k\}$ описує глобальний спектральний профіль зображення в різних діапазонах частот.

Для кожної смуги B_k підраховується середнє й середньоквадратичне відхилення енергії по всіх блоках (рис. 2.2), що відображає стабільність та варіативність спектрального сигналу по обличчю. Такий підхід відповідає ідеї побудови багаторівневих, ієрархічних ознак, де глобальні та локальні характеристики комбінуються у спільному просторі [29].

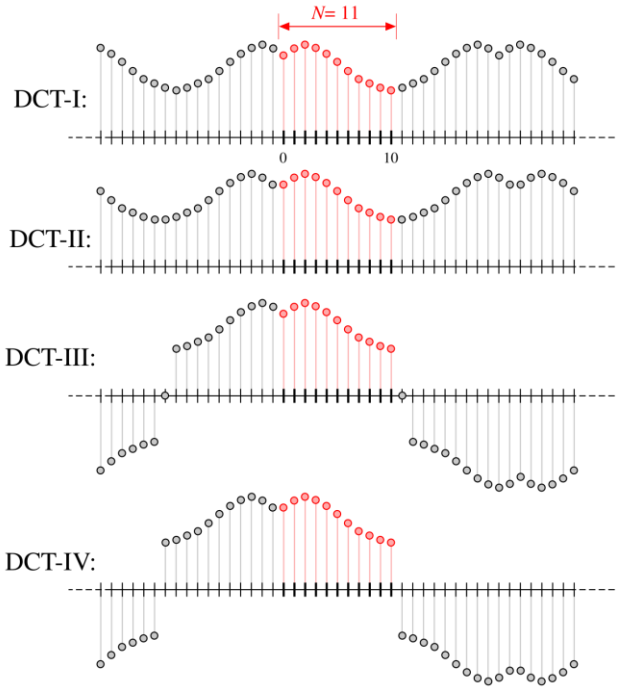


Рисунок 2.2 – Схема блокової DCT

Конкатенуємо глобальні та локальні статистики у вектор:

$$f^{freq} = \left[E_1^{glob}, \dots, E_K^{glob}, E_1^{loc}, \dots, E_K^{loc}\right]^T. \quad (2.4)$$

де E_K^{glob} – глобальні енергетичні характеристики смуг;

E_K^{loc} – узагальнені локальні показники (середні значення або конкатенація середнього та СКВ по блоках для смуги B_k).

Такий опис поєднує грубий «спектральний силует» обличчя з локальними відхиленнями, чутливими до патернів синтезу. Результати частотного аналізу CNN-генераторів та дипфейків демонструють, що саме ці відхилення дають стійкі ознаки для класифікації [29–30].

Параметри розбиття частот на смуги $\{B_k\}$, розмір блока b та степінь γ підбираються на валідаційному наборі з урахуванням компромісу між дискримінацією та розмірністю ознак. У «польових» умовах (стрімінґ, повторне кодування, шум, розмиття) частотні ознаки залишаються інформативними навіть

тоді, коли дрібні просторові деталі суттєво деградували, що підтверджується частотно-орієнтованими детекторами дипфейків [29–30].

2.1.2 Просторові ознаки

Просторова гілка моделі побудована на компактному згортковому екстракторі (наприклад, спрощені варіанти Xception/MobileNet/MesoNet), який отримує на вхід те саме вирівняне зображення і формує векторне представлення. Цей вектор кодує локальні текстури, межі зшивання, неконсистентні тіні та інші артефакти в піксельному просторі (рис. 2.3). Подібні компактні CNN-архітектури успішно застосовуються як універсальні детектори маніпуляцій в зображеннях, а також як базові блоки в задачах розпізнавання синтетичних медіа [29–30].

Рисунок 2.3 – Приклади real/fake та pipeline обробки у FaceForensics++

У подібних мережах перші шари фокусуються на низькорівневих ознаках (градієнти, текстурні патерни), середні – на локальних структурах (очі, рот, ніс), а глибокі – на більш глобальних контекстних ознаках. Додаткові дослідження показують, що CNN, навчені на маніпульованих зображеннях, можуть виявляти характерні конволюційні та інтерполяційні артефакти, притаманні певним класам генеративних моделей [31]. Це робить просторову гілку природним доповненням до частотної, оскільки вона фіксує семантично інтерпретовані дефекти (геометрія, текстура), тоді як частотна гілка – більш «низькорівневі» спектральні закономірності.

У реальних умовах (стрімінґ, повторне кодування, шум, розмиття) до **х**застосовують аугментації JPEG/blur/колірні варіації, що емулюють «польові» деградації та підсилюють робастність обох гілок. При цьому важливо не «зламати» спектральний профіль повністю, щоб частотна гілка залишалася інформативною. Підбір інтенсивності аугментацій проводиться на валідаційних підмножинах бенчмарків (наприклад, у рамках протоколів DeepfakeBench), де оцінюється узагальнюваність моделі на різні генератори й умови зйомки [32].

Лістинг 2.1 Кадровий детектор «простір+частота»:

```
double[] FrequencyStats(Bitmap faceGray, int block = 8)
{
int w = faceGray.Width, h = faceGray.Height;
double[,] Y = new double[h, w];
for (int y = 0; y < h; y++)
for (int x = 0; x < w; x++)
        Y[y, x] = faceGray.GetPixel(x, y).R;
double low = 0, mid = 0, high = 0;
for (int y = 0; y < h; y++)
for (int x = 0; x < w; x++)
{
        double v = Y[y, x];
        if ((x + y) < (w + h) * 0.15) low += Math.Abs(v);
        else if ((x + y) < (w + h) * 0.40) mid += Math.Abs(v);
        else high += Math.Abs(v);
}
double sum = Math.Max(1e-9, low + mid + high);
return new[] { low / sum, mid / sum, high / sum };
}
```

Семантично модель спирається на комплементарність представлень: просторові ознаки чутливі до геометричних та текстурних дефектів, тоді як частотні – до глобальних і локальних спектральних підписів синтезу. Спільне злиття $\mathbf{z}$ підвищує чутливість детектора до підробок, що маскуються в одному з доменів (рис. 2.4). Отриманий кадровий скор p надалі використовується в розділі часової агрегації (лог-правдоподібна інтеграція та ентропійно-зважене голосування), де формується відеорішення для задачі верифікації особи. У подальших підрозділах ці скалярні оцінки розглядаються також з точки зору невизначеності та калібрування, згідно з сучасними підходами до побудови надійних глибинних моделей [32-33].

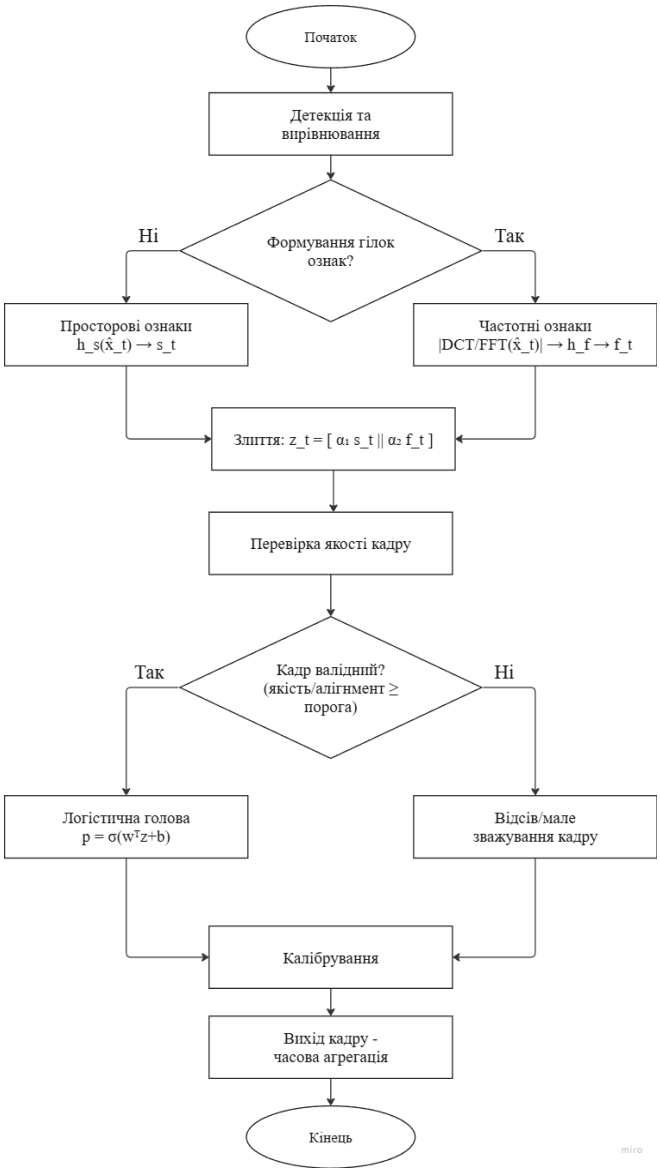


Рисунок 2.4 – Блок-схема гібридного кадрового детектора «простір+частота»

Покроковий опис моделювання гібридного кадрового детектора «простір+частота»:

Крок 1. Ініціалізація. Визначаємо службові параметри, цільний розмір вирівняного обличчя (наприклад, 256×256), перелік частотних смуг (низькі/середні/високі або кільцеві), розмір блоку для локальної обробки (типово 8×8 або 16×16), типи статистик у частоті (середнє, медіана, дисперсія, квантілі), ваги для злиття просторової та частотної гілок, а також поріг валідності кадру.

Крок 2. Захоплення кадру. Отримуємо черговий RGB-кадр відео. Фіксуємо час/індекс кадру для трасування.

Крок 3. Детекція та вирівнювання обличчя. Знаходимо обличчя (детектор на зразок RetinaFace/BlazeFace), перевіряємо, що воно достатнього розміру та не обрізане. По п’яти опорних точках (очі, ніс, кутики рота) виконуємо аффінне вирівнювання і приводимо зображення до уніфікованої геометрії та масштабу.

Крок 4. Нормування зрізу обличчя. Приводимо яскравість/контраст до стабільного діапазону, за потреби прибираємо сильний колірний зсув. Це зменшує доменні коливання між різними камерами/кодеками.

Крок 5. Перевірка валідності кадру. Розраховуємо індикатор якості: різкість (наприклад, по Лапласу), площа обличчя у кадрі, відсоток закритих пікселів, показник «успішності» вирівнювання. Якщо індикатор нижчий за поріг – позначаємо кадр як «низької якості» (надалі зменшимо його вагу).

Крок 6. Просторова гілка – підготовка. Подаємо вирівняне обличчя у компактну згорткову мережу (MobileNet/MesoNet/Xception-light) зі зменшеною кількістю параметрів, навчувану або довчувану на релевантних наборах.

Крок 7. Просторова гілка – отримання ознак. З останнього передкласового шару знімаємо вектор ознак фіксованої довжини (ембеддинг). За потреби виконуємо нормування вектора (щоб уникнути домінування окремих координат).

Крок 8. Частотна гілка – підготовка. Перетворюємо кадр у простір яскравості (або YCbCr). Це зменшує чутливість до колірних артефактів і фокусується на структурі.

Крок 9. Частотна гілка – глобальні спектральні ознаки. Виконуємо 2D-перетворення (DCT або FFT) і ділимо спектральну площину на задані смуги. Для кожної смуги рахуємо стабільні статистики енергії (наприклад, середнє та медіану за модулем коефіцієнтів). Це дає стійкі індикатори перерозподілу енергії, типової для синтетики.

Крок 10. Частотна гілка – локальні спектральні ознаки. Ділимо зображення на невеликі блоки (8×8, 16×16), повторюємо перетворення у кожному блоці, збираємо ті самі статистики по смугах і агрегуємо їх між блоками (середнє, дисперсія). Цей крок «бачить» мікро артефакти і періодичні патерни, що губляться при суто глобальному підрахунку.

Крок 11. Формування частотного вектора. Поєднуємо глобальні та локальні статистики в один компактний вектор фіксованої довжини. Нормуємо його (стандартизація або масштабування до діапазону), щоб різні групи ознак мали співставні масштаби.

Крок 12. Злиття доменів. Конкатенуємо просторовий ембеддинг і частотний вектор в одну ознаку. Застосовуємо ваги доменів: якщо очікуємо сильну компресію/ретрансляції – підсилюємо частотну гілку, якщо маємо багато геометричних спотворень – підсилюємо просторову.

Крок 13. Обчислення кадрового скору. Подамо об’єднану ознаку в невелику класифікаційну «голову» (логістична регресія або малий MLP на 1–2 шари). Отримуємо ймовірність того, що кадр є підробкою. Для кадрів, відмічених як «низької якості», додатково запам’ятовуємо зменшувач ваги (коефіцієнт, що буде використаний у часі).

Крок 14. Калібрування. На валідаційному наборі налаштовуємо температуру або інший простий калібратор, щоб перетворити «сирі» скори на краще узгоджені ймовірності. Це суттєво підвищує коректність подальшої лог-правдоподібної інтеграції.

Крок 15. Фіксація результату кадру. Зберігаємо трійку: ймовірність кадру, індикатор якості (або вагу), та часову позначку. За потреби –також проміжний вектор ознак для аналізу помилок.

Крок 16. Обробка всієї послідовності. Повторюємо Кроки 2–15 для кожного кадру відео. Формуємо послідовність кадрових оцінок та відповідних ваг (або міток валідності).

Крок 17. Передача в часову агрегацію. Подаємо послідовність на модуль відеорішення: лог-правдоподібна інтеграція підсумовує «докази» за клас, ентропійно-зважене голосування підсилює впевнені кадри й приглушує сумнівні.

Обидві схеми можуть працювати разом, лог-інтеграція як базова, ентропійне зважування –як робастний шар.

Крок 18. Протоколювання та метрики.На кадровому рівні фіксуємо ROC-AUC, EER, F1, а також калібрувальні метрики (Brier, очікувана похибка калібрування). Записуємо всі важливі налаштування: список аугментацій, визначення смуг у спектрі, розмір блоку, ваги злиття, поріг валідності, конфігурацію екстрактора. Це забезпечує відтворюваність.

Крок 19. Узагальнення/робастність. Якщо виявлено слабкі місця (наприклад, падіння на новому кодеку або при дуже малому обличчі), коригуємо одну з «ручок»: оновлюємо дизайн смуг або статистики в частотній гілці, додаємо відповідні аугментації, переваговуємо просторову/частотну гілки, коригуємо поріг валідності кадру.

Крок 20. Підсумок етапу моделювання. Маємо стабільний кадровий детектор, що поєднує інформативність просторових ознак і стійкість частотних, повертає калібровані ймовірності та продукує проміжні артефакти (ваги/валідність), необхідні для подальшої відео-агрегації та звітності.

2.2 Часова агрегація лог-правдоподібностей і ентропійно-зважене голосування

Мета методу – перетворити послідовність кадрових оцінок детектора дипфейків на єдине відеорішення, що є ключовим для верифікації особи:

рішення приймається за роликом, а не за одним кадром. Попередній кадровий детектор (просторовий, частотний чи їх гібрид) повертає для кожного кадру оцінку ймовірності підробки $p_t = P(\text{fake} \mid I_t)$. У сучасних бенчмарках (зокрема DeepfakeBench) це відображається у схемі «кадри – кадровий детектор – temporal pooling – відеоскор/поріг», де temporal pooling виступає окремим модулем, відповідальним за часову інтеграцію (рис. 2.5).

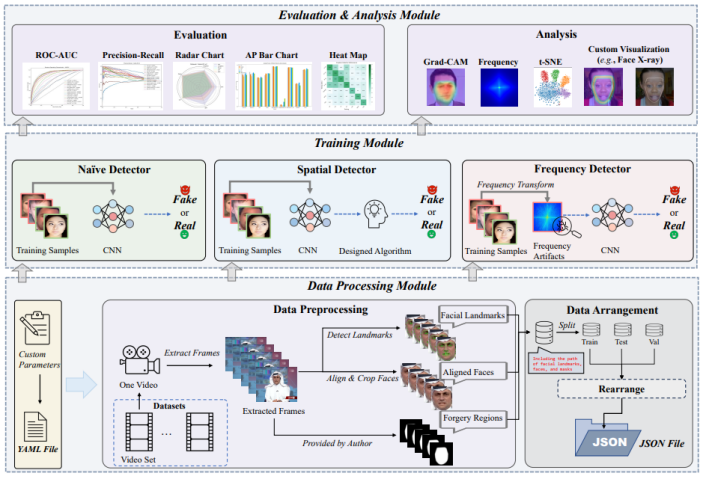


Рисунок 2.5 – Конвеєр DeepfakeBench

Обидві схеми природно узгоджуються з сучасними підходами до моделювання невизначеності в глибоких мережах і калібрування ймовірностей, де важливо не лише отримати середнє значення, а й коректно врахувати інформаційний вклад кожного спостереження.

2.2.1 Лог-правдоподібна інтеграція

Кадрові оцінки інтерпретуємо як докази на користь гіпотези «fake» відносно «real». Для кожного кадру з оцінкою $p_t \in (0,1)$обчислюємо лог-шанси (log-odds):

$$l_t = \log \frac{p_t}{1-p_t}. \tag{2.5}$$

де p_t – кадрова ймовірність класу fake для кадру з індексом t;

l_t – лог-шанс (log-odds) на користь класу fake відносно real для кадру t.

Відеорівнева сумарна статистика задається як адитивне накопичення незалежних свідчень:

$$L = \sum_{t \in T} l_t \, . \tag{2.6}$$

де T – множина індексів усіх валідних кадрів ролика (кадри, де успішно детектовано обличчя);

l_t – лог-шанс для кадру t;

L – сумарний лог-правдоподібнісний показник на користь того, що відео є fake (накопичені «докази» по всіх кадрах).

Така адитивна форма узгоджується з байєсівською логікою сумування лог-правдоподібностей: якщо кадрові спостереження умовно незалежні, то їхні log-odds додаються, утворюючи єдиний критерій для прийняття рішення на рівні відео [55, 56].

У роботах, присвячених калібруванню ймовірностей і оцінюванню невизначеності, підкреслюється важливість того, щоб p_tбули добре узгоджені з реальною частотою подій: це підвищує стабільність лог-інтеграції й дозволяє використовувати Lне лише як «score», а як частину ймовірнісної моделі ризику [55]. У запропонованій схемі лог-правдоподібна інтеграція виступає «дефолтним» шаром temporal pooling, на який надалі накладаються більш гнучкі вагові схеми (ентропійне зважування).

2.2.2 Ентропійно-зважене голосування

Щоб зменшити вплив шумних/неякісних кадрів (розмиття, сильна компресія, невдале вирівнювання обличчя), зважуємо їхній внесок коефіцієнтом,

що зменшується зі зростанням невизначеності моделі. Як міру невизначеності використано нормовану ентропію бінарного розподілу $(p_t, 1 - p_t)$.

$$H_t = -\frac{1}{\log 2}\left(p_t \log p_t + (1 - p_t)\log(1 - p_t)\right), \tag{2.7}$$

де H_t – нормована бінарна ентропія кадрового передбачення;

p_t – кадрова ймовірність класу fake для кадру з індексом t.

Далі задається вагова функція $w_t = w(H_t)$, наприклад у вигляді:

$$w_t = 1 - H_t, \tag{2.8}$$

де w_t – вага кадру t, що зменшується зі зростанням ентропії;

H_t – нормована бінарна ентропія кадрового передбачення.

Тобто впевнений кадр (низька ентропія) отримує велику вагу, а сумнівний – малу. На основі цих ваг можливі дві близькі схеми:

$$L_w = \sum_{t \in T} w_t l_t, \tag{2.9}$$

$$\bar{p} = \frac{\sum_{t \in T} w_t p_t}{\sum_{t \in T} w_t}. \tag{2.10}$$

де w_t – вага кадру t, що зменшується зі зростанням ентропії;

p_t – кадрова ймовірність класу fake для кадру з індексом t;

T – множина індексів усіх валідних кадрів ролика;

l_t – лог-шанс (log-odds) для кадру t;

L_w – ентропійно-зважена сумарна лог-правдоподібність на користь класу fake;

$\bar{p}$ – ентропійно-зважене середнє значення ймовірності fake по ролику.

На практиці підмножину валідних кадрів $\mathcal{T}$ формують також простими критеріями якості (успішність аліґнменту, мінімальний розмір обличчя в кадрі, відсутність сильного розмиття/перекриттів). Такі фільтри, разом з ентропійним

зважуванням, помітно підвищують стабільність лог-інтеграції, особливо на «польових» даних з повторними компресіями/стрімінгом, де якість кадрів сильно варіюється [55].

Лістинг 2.2 log-odds-пулінг і ентропійно-зважене голосування:

```
public static double LogOddsPooling(IEnumerable<double> frameProb, double minP = 1e-6, double maxP = 1 - 1e-6)
{
    double L = 0.0;
    foreach (var p in frameProb)
    {
        var pc = Math.Clamp(p, minP, maxP);
        L += Math.Log(pc / (1.0 - pc));
    }
    return Sigmoid(L);
}

private static double Sigmoid(double x) { return 1.0 / (1.0 + Math.Exp(-x)); }
```

З точки зору теорії невизначеності в глибоких мережах, ентропія вихідного розподілу є базовим індикатором епістемічної та алейаторної невизначеності. Врахування цього індикатора в схемах прийняття рішень знижує ризик «перевпевнених» помилок і робить систему більш надійною.

У результаті часовий модуль, що поєднує лог-інтеграцію та ентропійне зважування, формує фінальний відеоскор, який використовується як вхід до системи прийняття рішень у задачі верифікації особи (рис. 2.6). Пороги на L, L_w або $\bar{p}$ налаштовуються на валідаційних множинах відповідно до вимог KYC-сценарію (цільова TPR/FPR), а сама схема залишається прозорою, інтерпретованою й сумісною з механізмами калібрування ймовірностей та оцінки невизначеності [55],[56].

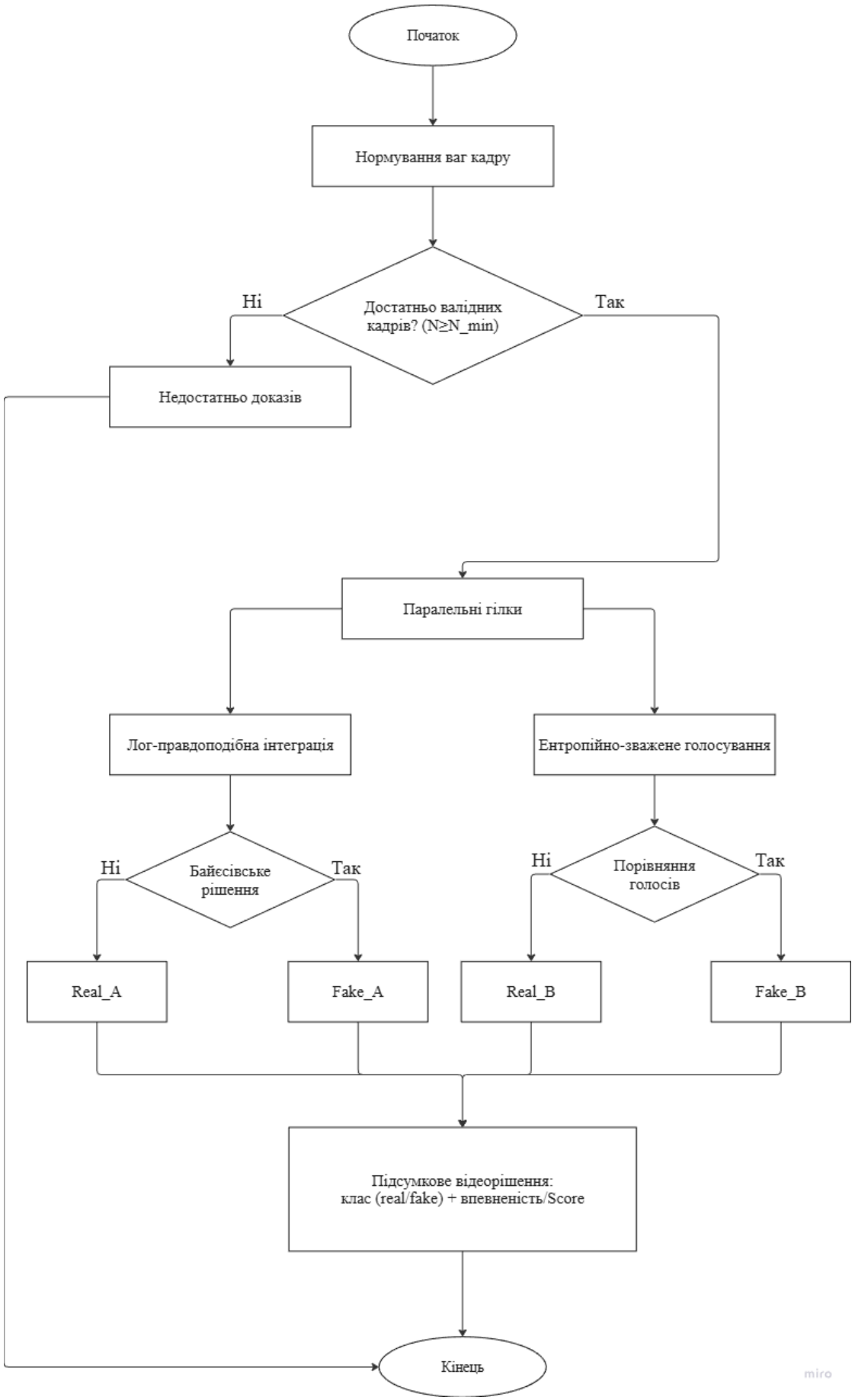


Рисунок 2.6 – Блок-схема часової агрегації для двох гілок

Покроковий опис моделювання гібридного кадрового детектора «простір+частота»:

Крок 1. Вхідні дані. Отримуємо для всіх кадрів відео: кадрові ймовірності fake кадр (калібровані після детектора «простір+частота»), індикатор якості/валідності кадру (різкість, площа, успішність вирівнювання тощо), міру невизначеності кадру (ентропію) або її нормовану версію.

Крок 2. Масштабування якості в вагу. Перетворюємо валідность кадру на ваговий коефіцієнт. Простий варіант – лінійне шкалування з відсіканням «поганих» кадрів (нижче порога вага ≈ 0).

Крок 3. Масштабування невизначеності в вагу. Обчислюємо вагу за впевненістю , впевнені кадри (низька ентропія) мають вищу вагу.

Крок 4. Композиція ваг. Об’єднуємо внески якості й впевненості або іншу монотонну комбінацію. Це підготує єдину вагу для обох гілок агрегації.

Крок 5. Відбір валідних кадрів. Залишаємо лише кадри з W >0. Якщо валідних кадрів менше за заданий мінімум (наприклад, 10–20), фіксуємо статус «недостатньо доказів» і повертаємо службовий результат (попросити повторити зйомку).

Крок 6. Підготовка до двох паралельних агрегацій. Створення гілки лог-правдоподібної інтеграції (сума «доказів» за/проти) та для ентропійно-зваженого голосування (рахунок «голосів» з вагами).

Крок 7. Перетворення ймовірностей у «докази». Для кожного кадру обчислюємо лог-правдоподібний внесок (логіт).

Крок 8. Зважене підсумовування. Сумуємо логіти по всіх валідних кадрах із вагами. Отримуємо сумарний «доказ» відео.

Крок 9. Прийняття рішення за порогом. Порог Г підбираємо на валідації для бажаного балансу помилок (FAR/FRR).

Крок 10. Оцінка впевненості. Як міру впевненості можемо взяти модуль сумарного «доказу» або його перетворення у ймовірність (через обернену логістичну функцію).

Крок 11. Бінаризація кадрів у голоси. Задаємо кадровий поріг для перетворення у голос.

Крок 12. Накопичення голосів із вагами. Підсумовуємо ваги кадрів, що проголосували за fake, та ваги кадрів, що проголосували за real. Отримуємо дві суми: «за fake» і «за real».

Крок 13. Порівняння голосів. Якщо сума «за fake» не менша, ніж «за real», результат гілки – fake, інакше – real.

Крок 14. Міра впевненості. Як score беремо різницю між сумами голосів (після нормування на загальну вагу), або відношення «сильніших» голосів до слабших.

Крок 15. Вибір політики злиття. Три практичні варіанти: довіряти рішенню лог-правдоподібної інтеграції , вимагати погодженості обох гілок ($A \cap B$), якщо якість кадрів нестабільна (високі ентропії, багато відсіяних), пріоритезувати голосування, якщо кадри якісні та добре калібровані – prio log-odds.

Крок 16. Формування підсумкового score. Фіксуємо клас (real/fake) і узгоджений score (наприклад, гармонізована шкала, що поєднує модуль сумарного «доказу» з нормованою різницею голосів). Score знадобиться для порогування в сервісах KYC/доступу.

Крок 17. Протоколування. Зберігаємо обрану політику, межі, відсоток відсіяних кадрів, розподіл, підсумкові ваги, час обчислення. Це критично для аудиту.

Крок 18. Оцінювання якості відеорішень. На рівні відео обчислюємо ROC-AUC, EER, F1, т. зв. operating point, а також стабільність рішень при зміні довжини ролика (аналіз чутливості до T).

Крок 19. Підсумок. У результаті одержуємо узгоджене відеорішення з пояснюваними показниками – який відсоток кадрів дав вирішальний внесок, які були джерела невпевненості, і наскільки «сильним» був сукупний доказ. Обидві гілки залишаються сумісними з будь-яким кадровим детектором і можуть бути ввімкнені/вимкнені політикою розгортання без змін базової моделі.

2.3 Порівняння методів

Наведені підходи вирішують різні рівні тієї самої задачі й тому не конкурують, а доповнюють один одного. Гібридний кадровий детектор «простір+частота» формує надійний сигнал на рівні окремого кадру – просторове подання вловлює семантичні дефекти (межі зшивання, локальні текстури, мікроміміку), а частотне – стійкі спектральні підписи синтезу та наслідки перекодування. Завдяки цій комплементарності модель краще бачить різнорідні підробки й менше «ламається» від JPEG/стримінгу, особливо коли під час навчання враховано відповідні аугментації. Водночас її рішення залишаються кадровими, тож у реальних роликах із коливною якістю (рух, розмивання, часткові оклюзії, нерівномірний алайнмент) оцінки можуть «стрибати» – виникає потреба у механізмі, що перетворює послідовність pt на одне, стабільне відеорішення.

Цю роль виконує часова агрегація. Лог-правдоподібна інтеграція інтерпретує кожний кадр як «доказ» і акумулює лог-шанси по всій послідовності, завдяки чому поодинокі збої не визначають результат. Ентропійно-зважене голосування йде далі – воно зменшує внесок невпевнених кадрів (із високою ентропією) і, отже, підвищує робастність у «польових» умовах, де якість кадрів неоднорідна. Якщо ж базовий кадровий детектор систематично упереджений або некалібрований, агрегація цього не виправить – вона лише коректно «складе» те, що отримала. Тому на практиці оптимально спершу отримати якісний кадровий скор (гібрид «простір+частота» з фільтрами валідності кадрів), а потім застосувати агрегацію (log-odds як база, ентропійні ваги – для важких сцен). У чистих коротких кліпах користь від агрегації може бути помірною, зате на довгих/шумних роликах вона критична для стабільності FPR/TPR на відеорівні. Отже, перший метод дає силу сигналу, другий – стабільність рішення. Разом вони формують продукційний конвеєр відеоверифікації.

У ширшому контексті такі методи можна розглядати як «ядро» системи виявлення дипфейків, яке легко комбінується з іншими захисними шарами.

Наприклад, Liveness/PAD-модулі й аналіз аудіо-візуальної узгодженості додають ортогональні ознаки, але зазвичай вимагають складніших сенсорних умов. Натомість запропонований гібридний детектор працює без додаткових каналів – достатньо відеопотоку з обличчям прийнятної роздільної здатності і може застосовуватися як універсальний фільтр у KYC-конвеєрі.

Таблиця 2.1 – Порівняння методів

Критерій	Гібридний кадровий детектор «простір+частота»	Часова агрегація (log-odds, entropy-weights)
Рівень рішення	Кадр	Відео
Інформація	Піксельні ознаки + DCT/FFT спектр (глобальні й локальні статистики)	Послідовність кадрів, сума лог-шансів, ваги за ентропією/якістю
Ключова сила	Комплементарність доменів – вища чутливість до різних типів фейків, стійкість до перекодувань	Статистично коректне «складання доказів», придушення сумнівних кадрів, стабільність на довгих/шумних відео
Критичні залежності	Якість детекції/аліґнменту, дизайн частотних смуг/блоків, аугментації	Калібрування кадрів, пороги, відбір валідних кадрів
Основні ризики	Перенавчання на спектральні статистики конкретного набору, деградація при blur/оклюзіях	Не виправляє систематичний bias кадрового детектора, чутливість до некаліброваних оцінок
Обчислювальна вартість	Помірна (CNN + DCT/FFT) на кадр	Дуже низька (суми/логіка поверх кадрів)
Коли особливо доречний	Коли потрібен сильний кадровий скор і чутливість до різних артефактів	Коли потрібна надійність відеорівня: довгі/стрімінгові, неоднорідні за якістю ролики
Роль у системі	Джерело інформативних кадрів	Перетворює кадр на стабільне рішення на рівні відео

3 КОМП'ЮТЕРНА МОДЕЛЬ ВИЯВЛЕННЯ ДИПФЕЙКІВ

3.1 Обґрунтування вибору середовища програмної реалізації

Комп'ютерна модель виявлення дипфейків, розроблена в межах дипломної роботи, призначена не лише для обчислення окремого нейромережевого скору, а для реалізації повного прикладного конвеєра відеоверифікації: від завантаження відеоролика та екстракції кадрів до детекції та вирівнювання облич, формування просторово-частотних ознак, отримання кадрових оцінок, їх часової агрегації й видачі кінцевого рішення для ролика. Така постановка задачі накладає специфічні вимоги до середовища програмної реалізації. З одного боку, потрібна тісна інтеграція з бібліотеками комп'ютерного зору й глибинного навчання, підтримка наперед натренованих моделей та ефективна робота з багатовимірними масивами. З іншого боку, необхідна можливість швидко модифікувати архітектуру, експериментувати з частотними ознаками, схемами агрегації, режимами тестування, а також наочно демонструвати роботу системи у вигляді простого графічного інтерфейсу.

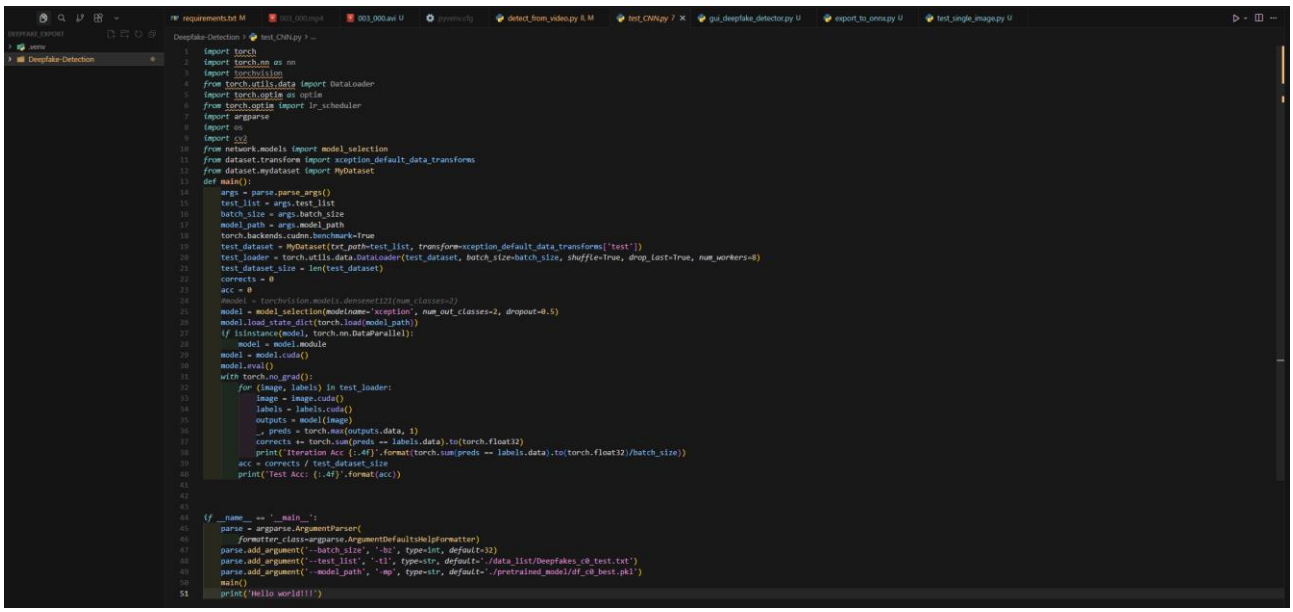
З огляду на ці вимоги, як базову платформу було обрано мову програмування Python у поєднанні із середовищем розробки Visual Studio Code, фреймворком глибинного навчання PyTorch, бібліотекою OpenCV для роботи з відео та зображеннями, а також стандартними засобами побудови графічного інтерфейсу. Такий вибір добре узгоджується з домінуючою практикою в галузі комп'ютерного зору і детекції дипфейків: переважна більшість відкритих реалізацій моделей для FaceForensics++, DeepFake Detection Challenge та похідних бенчмарків надається саме в екосистемі Python і PyTorch, що дозволяє відтворювати існуючі протоколи й безпосередньо порівнювати власні результати з опублікованими.

Важливою перевагою Python у цьому контексті є поєднання високорівневого синтаксису з можливістю ефективних векторизованих обчислень за рахунок NumPy та тісної інтеграції з бібліотеками обробки

зображень. Бібліотека OpenCV, що є де-факто стандартом для задач комп'ютерного зору, у Python-варіанті забезпечує повний набір операцій, потрібних для даного проєкту: читання відеофайлів, екстракція кадрів, перетворення між колірними просторами, зміна роздільної здатності, геометричні трансформації, побудова візуальних анотацій (рамки обличчя, накладення текстових підписів про ймовірність підробки тощо) [52]. Це дозволяє реалізувати препроцесинг і візуалізацію в одному середовищі, без додаткових проміжних інструментів чи мов програмування.

Фреймворк PyTorch забезпечує модульну реалізацію нейронних мереж, зручні інтерфейси для завантаження натренованих ваг, гнучке керування обчисленнями на CPU/GPU, а також високорівневі засоби організації циклів навчання і тестування. У контексті цієї роботи це особливо важливо, оскільки використовується готовий Xception-подібний детектор дипфейків, адаптований під задачі аналізу облич, і до нього додаються власні блоки частотних ознак та часової агрегації. Вибір PyTorch дозволяє мінімальними змінами в коді інтегрувати додаткові модулі, перевіряти різні варіанти архітектури та порівнювати їх у єдиному експериментальному середовищі.

Середовище розробки Visual Studio Code було обрано тому, що воно надає зручну підтримку Python-проєктів: підсвічування синтаксису, автодоповнення, інтегрований відлагоджувач, роботу з віртуальними середовищами та засобами керування залежностями (рис. 3.1). Практично це означає, що всі основні сценарії – завантаження датасету FaceForensics++, тренування моделі, тестування на окремих зображеннях, обробка відео, запуск графічного інтерфейсу – можуть виконуватися в межах одного робочого середовища, із збереженням історії змін і можливістю швидкого повернення до попередніх конфігурацій. Така організація суттєво спрощує проведення серій експериментів, що є невід'ємною частиною дослідницької роботи. Додатково, інтегровані термінал та підтримка Git дозволяють безпосередньо з IDE керувати запуском скриптів, версіями коду та резервним копіюванням, не перемикаючись між різними програмами.



```python
import torch
import torch.nn as nn
import torchvision
from torch.utils.data import DataLoader
import torch.optim as optim
from torch.optim import lr_scheduler
import argparse
import os
import cv2
from network.models import model_selection
from dataset.transform import xception_default_data_transforms
from dataset.mydataset import MyDataset
def main():
    args = parse.parse_args()
    test_list = args.test_list
    batch_size = args.batch_size
    model_path = args.model_path
    torch.backends.cudnn.benchmark=True
    test_dataset = MyDataset(txt_path=test_list, transform=xception_default_data_transforms['test'])
    test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size, shuffle=True, drop_last=True, num_workers=8)
    test_dataset_size = len(test_dataset)
    corrects = 0
    acc = 0
    #model = torchvision.models.densenet121(num_classes=2)
    model = model_selection(modelname='xception', num_out_classes=2, dropout=0.5)
    model.load_state_dict(torch.load(model_path))
    if isinstance(model, torch.nn.DataParallel):
        model = model.module
    model = model.cuda()
    model.eval()
    with torch.no_grad():
        for (image, labels) in test_loader:
            image = image.cuda()
            labels = labels.cuda()
            outputs = model(image)
            _, preds = torch.max(outputs.data, 1)
            corrects += torch.sum(preds == labels.data).to(torch.float32)
            print('Iteration Acc {:.4f}'.format(torch.sum(preds == labels.data).to(torch.float32)/batch_size))
        acc = corrects / test_dataset_size
        print('Test Acc: {:.4f}'.format(acc))


if __name__ == '__main__':
    parse = argparse.ArgumentParser(
        formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parse.add_argument('--batch_size', '-b2', type=int, default=32)
    parse.add_argument('--test_list', '-tl', type=str, default='./data_list/Deepfakes_c0_test.txt')
    parse.add_argument('--model_path', '-mp', type=str, default='./pretrained_model/df_c0_best.pk1')
    main()
    print('Hello world!!!')
```

Рисунок 3.1 – Приклад інтерфейсу VS Code

Окремим аспектом є вибір інструментів для візуалізації і взаємодії з користувачем. Оскільки у дипломній роботі важливо не лише отримати числові метрики якості, а й продемонструвати роботу системи на реальних відео, була використана стандартна бібліотека tkinter, що входить до складу Python. Вона дозволила реалізувати легкий графічний інтерфейс без додаткових залежностей: через нього користувач обирає відеофайл, задає модель детектора, запускає аналіз та отримує підсумковий висновок. Поєднання tkinter з OpenCV дає змогу показувати попередній перегляд анотованих кадрів і тим самим зробити роботу моделі більш наочною і зрозумілою навіть для користувача, що не занурений у технічні деталі реалізації.

У процесі розробки також розглядався варіант перенесення натренованої моделі в середовище .NET шляхом експорту в універсальний формат ONNX і подальшого використання рушія ONNX Runtime та інструментів платформи .NET 8 [53],[54]. Така схема потенційно відкриває шлях до промислового розгортання моделі, зокрема інтеграції у високопродуктивні бекенд-сервіси систем відеоверифікації та KYC. Для цього був підготовлений окремий скрипт експорту, що демонструє технічну сумісність обраної архітектури з ONNX-стеком. Однак у межах даної роботи основний акцент зроблено на

дослідницькому стенді в Python, що дозволяє максимально гнучко змінювати архітектуру, частотні ознаки та схеми часової агрегації, а також швидко перевіряти їх вплив на якість виявлення дипфейків без додаткових обмежень, пов’язаних із цільовою платформою.

Вибір такого стеку узгоджується також із методологічною частиною роботи. Розроблена модель частотно-просторової агрегації та ентропійно-зваженого голосування для відеоверифікації розглядається як приклад застосування сучасних технологій глибинного навчання та цифрової обробки сигналів у задачах прийняття рішень, що описано в попередніх працях автора [55, 56]. Обране середовище програмної реалізації дозволяє безпосередньо втілити ці концепції у вигляді працюючого програмного прототипу, який одночасно є достатньо гнучким для експериментів і достатньо практичним для демонстрації на реальних відеоданих.

Таким чином, зв’язка Python, PyTorch, OpenCV, Visual Studio Code та вбудованих засобів GUI становить природний вибір для реалізації комп’ютерної моделі виявлення дипфейків у відеоверифікації. Вона поєднує зрілість бібліотек для комп’ютерного зору, зручність роботи з нейромережевими моделями, підтримку експериментальної діяльності та потенційну можливість переходу до високопродуктивних середовищ інференсу через ONNX та .NET у подальших етапах розвитку системи [52–56].

3.2 Програмна реалізація

Програмна реалізація комп’ютерної моделі виявлення дипфейків базується на наборі взаємопов’язаних Python-скриптів, кожен з яких відповідає за окремий етап конвеєра: завантаження та підготовку даних, тренування та тестування нейронної мережі, аналіз поодиноких зображень, обробку відео з побудовою відеорівневого рішення, а також взаємодію з користувачем через графічний інтерфейс. Усі ці компоненти працюють в єдиному середовищі PyTorch +

OpenCV і реалізують описану у другому розділі частотно-просторову модель із часовою агрегацією.

На найнижчому рівні знаходяться скрипти train_CNN.py та test_CNN.py, які реалізують класичний пайплайн навчання та валідації згорткової мережі на рамках FaceForensics++ та споріднених датасетів. Навчальний скрипт завантажує списки шляхів до зображень та їх міток, формує об’єкти MyDataset із відповідними перетвореннями (xception_default_data_transforms['train'] та ['val']), створює генератори міні-пакетів (DataLoader) із визначеним розміром пакету, параметрами перемішування, кількістю робочих потоків. Далі будується модель за допомогою функції model_selection, яка дозволяє вибрати одну з підтримуваних архітектур (Xception, MesoNet тощо) із параметром num_out_classes=2, що відповідає бінарній класифікації «real/fake». Цикл навчання реалізує типовий шаблон PyTorch: на кожному епосі мережа переводиться в режим train(), обробляє міні-пакети, обчислює функцію втрат, виконує зворотне поширення похибки та оновлення ваг оптимізатором. На валідаційній частині мережа працює в режимі eval() без оновлення параметрів, а її якість оцінюється за метриками точності, втрат, AUC тощо. Отримані ваги зберігаються у файлах .pth або .pkl і надалі використовуються в режимі інференсу. Така структура дозволяє використати як вже наявні pretrained-моделі, так і довчити їх або донавчити на додаткових підвибірках під конкретний сценарій.

Окремим, більш легким модулем є скрипт test_single_image.py, який реалізує повний цикл аналізу одного зображення: завантаження файлу, детекція обличчя, нормалізація та прогін через модель. На початку задаються стандартні статистики нормалізації MEAN та STD, характерні для ImageNet-орієнтованих моделей (Xception, ResNet тощо), і розмір входу IMG_SIZE = 299, що відповідає очікуванням Xception-блоків. Для пошуку облич використовується фронтальний детектор бібліотеки dlib, ініціалізований як face_detector = dlib.get_frontal_face_detector().

Лістинг 3.1 Використання фронтального детектору бібліотеки dlib для пошуку обличь:

```
def detect_and_crop_face(img_bgr: np.ndarray) -> np.ndarray:
    gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
    faces = face_detector(gray, 1)

    if len(faces) == 0:
        return img_bgr

    biggest = max(faces, key=lambda rect: rect.width() * rect.height())
```

Функція повертає кроп найбільшого обличчя з невеликим запасом по краях або повний кадр, якщо обличчя не знайдено. Це відповідає вимозі KYC-сценарію орієнтуватися саме на обличчя, а не на фон. Після кропу зображення масштабується до фіксованого розміру, переводиться у формат float32, нормалізується за каналами за формулою $(x - \text{MEAN})/\text{STD}$, трансформується в тензор PyTorch і подається на вхід моделі, створеної через model_selection. На виході скрипт виводить ймовірності класів real та fake, що дає змогу швидко перевірити роботу натренованих ваг на окремих прикладах без запуску повного відеоконвеєра.

Центральним елементом програмної реалізації є скрипт detect_from_video.py, який відповідає за обробку відеороликів і реалізує як кадрову частину моделі, так і запропоновані схеми часової агрегації. На вході скрипт приймає шлях до відеофайлу або папки з відео, шлях до файлу моделі та вихідну директорію, куди зберігатиметься анотоване відео. Головна функція test_full_image_network(video_path, model_path, output_path, ...) відкриває відео через cv2.VideoCapture, зчитує параметри потоку (кількість кадрів, частота кадрів, роздільна здатність), готує VideoWriter для формування вихідного AVI-файлу, а також ініціалізує фронтальний детектор облич dlib.

Лістинг 3.2 Реалізація завантаження моделі:

```
model = model_selection(modelname='xception', num_out_classes=2, dropout=0.5)
model.load_state_dict(torch.load(model_path, map_location='cpu'))
model.eval()
```

Такий підхід забезпечує відтворення тієї ж архітектури, яка використовувалася під час навчання. Для сумісності з різними конфігураціями обчислювальних пристроїв передбачена можливість запуску як на CPU, так і на GPU (cuda=True/False).

Обробка відео відбувається у циклі по кадрах – кожен кадр зчитується з VideoCapture, по потребі обрізається за заданими межами start_frame та end_frame, після чого перетворюється у відтінки сірого і подається на детектор облич. У разі виявлення одного або кількох облич обирається найбільший детект, який інтерпретується як основне обличчя у кадрі. На основі координат детекту формується квадратна рамка з невеликим збільшенням, щоб захопити всю область обличчя та прилеглі ділянки, і виконується кроп зображення.

Отриманий кроп обличчя передається в допоміжну функцію predict_with_model, де застосовується та сама послідовність перетворень, що і для одиночних зображень: масштабування, нормалізація, перетворення в тензор, за потреби перенесення на GPU. Модель повертає двовимірний вихід output розміру, до якого застосовується nn.Softmax(dim=1), і з нього витягуються prob_real та prob_fake. Значення prob_fake додається в список frame_fake_probs, який акумулює кадрові ймовірності підробки для подальшої часової агрегації. Паралельно на вихідне відео накладається прямокутник навколо обличчя та текстовий підпис із поточним значенням ймовірності, після чого кадр записується у вихідний відеофайл. Таким чином, на виході користувач отримує не лише числовий підсумок, а й візуальний ролик з індикацією роботи детектора на кожному кадрі.

Ключовою відмінністю від базового коду FaceForensics++ є реалізація власних функцій часової агрегації log_odds_aggregate та entropy_weighted_aggregate, які безпосередньо відображають теоретичні схеми, описані у розділі 2.2.

Лістинг 3.3 Реалізація функції лог-правдоподібної інтеграції:

```
def log_odds_aggregate(probs, temperature=1.0):
    probs = np.asarray(probs, dtype=np.float32)
    eps = 1e-6
    probs = np.clip(probs, eps, 1 - eps)

    logits = np.log(probs / (1.0 - probs))
    mean_logit = np.mean(logits) / temperature

    P_log = 1.0 / (1.0 + np.exp(-mean_logit))
    return float(P_log)
```

На вхід функція отримує список кадрових ймовірностей probs, які спочатку обрізаються від нулів та одиниць для уникнення чисельних проблем. Далі обчислюються логіти (log-odds) для кожного кадру за формулою $\ell_t = \log \frac{p_t}{1-p_t}$, після чого береться їх середнє значення і за потреби масштабується параметром temperature. Повернення до ймовірнісного простору здійснюється через сигмоїдну функцію. Таким чином, замість простого арифметичного усереднення ймовірностей, що не завжди коректно в байєсівському сенсі, використовується інтеграція у просторі лог-шансів, яка краще відображає накопичення доказів за ролик.

Лістинг 3.4 Реалізація ентропійно-зваженого голосування:

```
def entropy_weighted_aggregate(probs):
    if not probs:
```

```
    return None

eps = 1e-6
weights = []
for p in probs:
    p = min(max(p, eps), 1.0 - eps)
    H = -(p * math.log2(p) + (1.0 - p) * math.log2(1.0 - p))
    w = 1.0 - H
    weights.append(w)

weights = np.asarray(weights, dtype=np.float32)
probs = np.asarray(probs, dtype=np.float32)
weighted_mean = np.sum(weights * probs) / (np.sum(weights) + eps)
return float(weighted_mean)
```

Тут для кожного кадру обчислюється двокласова ентропія H_tу логарифмічній основі 2, яка наближається до нуля для впевнених передбачень (ймовірність близька до 0 або 1) і до одиниці для невизначених (близько 0.5). Вага кадру задається як $w_t = 1 - H_t$, тобто впевнені оцінки мають вагу, близьку до 1, тоді як сумнівні кадри майже не впливають на підсумок. Повертається зважене середнє ймовірностей, де роль кожного кадру пропорційна його інформативності. Така реалізація дозволяє у «польових» умовах (стрімінг, розмиття, часткове перекриття обличчя) природно пригнічувати внесок проблемних кадрів, не вводячи жорстких порогів та додаткових гіперпараметрів.

Наприкінці роботи з відео, якщо список frame_fake_probs не порожній, обчислюються обидві агреговані оцінки.

Лістинг 3.5 Реалізація ентропійно-зваженого голосування:

```
if frame_fake_probs:
    P_log = log_odds_aggregate(frame_fake_probs)
```

```
P_ent = entropy_weighted_aggregate(frame_fake_probs)

print('\n=== VIDEO-LEVEL DECISION ===')
print(f'Кількість кадрів з обличчям: {len(frame_fake_probs)}')
print(f'[LOG] prob_fake (log-odds integration):  {P_log:.4f}')
print(f'[ENT] prob_fake (entropy-weighted):     {P_ent:.4f}')

threshold = 0.5
label_log = 'FAKE' if P_log >= threshold else 'REAL'
label_ent = 'FAKE' if P_ent >= threshold else 'REAL'
print(f'LOG decision: {label_log},  ENT decision: {label_ent}\n')
```

У цьому блоці скрипт підсумовує кількість кадрів із валідним обличчям, виводить обидва відеоскори та відповідні класи при фіксованому порозі. Саме ці значення використовуються надалі в якості основних вихідних характеристик відеодетектора при дослідженні якості та калібруванні моделі на контрольних наборах.

Над кадровим та відеорівневим конвеєром розташовується скрипт gui_deepfake_detector.py, який реалізує простий, але функціональний графічний інтерфейс для запуску аналізу. У верхній частині файлу визначається словник MODEL_OPTIONS, що відображає читабельні назви моделей у вигляді рядків меню на реальні шляхи до файлів ваг у директорії ./pretrained_model. Це дозволяє легко перемикатися між різними pretrained-детекторами, наприклад між моделлю, натренованою на FF++ із різними рівнями компресії, та моделлю, адаптованою під інший датасет. Далі створюється вікно Tkinter, у якому передбачені поле для введення шляху до відеофайлу, кнопка «Огляд…» для вибору файлу через стандартний діалог, випадаючий список для вибору моделі, а також кнопка запуску аналізу. Після натискання на кнопку «Запустити аналіз» обробник перевіряє коректність введених шляхів і викликає функцію

test_full_image_network з detect_from_video.py, передаючи туди вибраний відеофайл, файл моделі та вихідну директорію.

Таким чином, користувацький інтерфейс фактично є тонким шаром над основним відеоконвеєром, який інкапсулює весь складний функціонал препроцесингу, роботи моделі та часової агрегації. Така організація коду відображає розподіл відповідальностей: одна частина системи реалізує науково обґрунтовані методи аналізу (частотні ознаки, log-odds інтеграція, ентропійно-зважене голосування), інша забезпечує зручний доступ до цих методів для кінцевого користувача.

У підсумку програмна реалізація поєднує класичний для глибинного навчання навчально-тестовий стек із спеціалізованими модулями для відеоаналізу та часової агрегації, а також мінімалістичний GUI для взаємодії з людиною.

3.3 Інструкція користувача

Розроблена комп’ютерна модель виявлення дипфейків орієнтована на користувача, який працює у середовищі Python (наприклад, у Visual Studio Code) та має базові навички запуску скриптів. Інтерфейс можна умовно поділити на два рівні взаємодії. Перший – це графічний інтерфейс, який надає можливість перевіряти відео без роботи з командним рядком. Другий – сценарії командного запуску (detect_from_video.py, test_single_image.py), що дозволяють гнучкіше керувати параметрами та інтегрувати модель у зовнішні експериментальні або сервісні конвеєри.

Перед початком роботи користувач має підготувати програмне середовище: встановити інтерпретатор Python відповідної версії, інсталювати залежності (PyTorch, OpenCV, NumPy, dlib та інші бібліотеки, зазначені у файлі вимог) та переконатися, що середовище коректно розпізнає ці пакети. У випадку роботи в Visual Studio Code зручно використовувати віртуальне середовище, в

якому встановлюються всі необхідні модулі. Після цього каталог проєкту з файлами detect_from_video.py, gui_deepfake_detector.py, test_single_image.py та іншими скриптами відкривається як робоча директорія. Набір попередньо навчених моделей розміщується в окремій папці (наприклад, pretrained_model/), де кожен файл відповідає певній архітектурі та протоколу навчання (FaceForensics++ з різними рівнями компресії, моделі, адаптовані під інші набори тощо).

Найбільш зручним способом користування системою для кінцевого користувача є запуск графічного інтерфейсу через файл gui_deepfake_detector.py. Це можна зробити безпосередньо з Visual Studio Code, обравши цей файл у дереві проєкту та натиснувши запуск, або з командного рядка, виконавши команду python gui_deepfake_detector.py у кореневій папці проєкту. Після запуску відкривається основне вікно застосунку, у якому передбачено поле для вибору відеофайлу, випадаючий список для вибору моделі детектора дипфейків, а також кнопка запуску аналізу.

Робота з інтерфейсом відбувається послідовно. Спочатку користувач обирає відео: натискає кнопку «Огляд…» (або аналогічну за змістом), після чого з’являється стандартне діалогове вікно вибору файлу операційної системи. Після вибору ролика шлях до нього відображається у текстовому полі. Далі користувач переходить до вибору моделі: у випадаючому списку відображаються читабельні назви варіантів, наприклад «Xception FF++ c23», «Xception DFDC», «Модель частотно-просторового детектора», які внутрішньо відображаються на конкретні файли ваг у директорії pretrained_model. Це дозволяє використати різні конфігурації детектора без будь-яких змін у коді. Після вибору моделі користувач натискає кнопку запуску, і застосунок перевіряє коректність введених даних: існування відеофайлу, доступність файлу моделі, коректність вихідної директорії для результатів. У разі виявлення помилки (наприклад, не вибрано відео або відсутній файл моделі) система виводить повідомлення у діалоговому вікні.

Якщо всі параметри задано коректно, скрипт gui_deepfake_detector.py передає керування функції test_full_image_network з модуля detect_from_video.py. З погляду користувача, на цьому етапі починається аналіз: у вікні можна спостерігати текстові повідомлення про хід обробки (наприклад, початок зчитування відео, кількість опрацьованих кадрів), а після завершення в консолі або текстовому полі виводиться підсумок. Зазвичай він містить кількість кадрів, на яких було успішно виявлено обличчя, значення агрегованої ймовірності підробки за лог-правдоподібною інтеграцією і за ентропійно-зваженим голосуванням, а також відповідні класи («FAKE» або «REAL») при заданому пороговому значенні, наприклад 0,5. Одночасно у вказаній вихідній директорії з’являється анотований відеофайл, у якому на кожному кадрі накладено рамку навколо обличчя і текст із поточною оцінкою fake-ймовірності. Таким чином, користувач може не лише побачити підсумковий вердикт, а й візуально оцінити, як поводиться модель на різних фрагментах ролика.

Окремий сценарій використання стосується модулю test_single_image.py. Якщо користувачеві потрібно оцінити якість моделі на окремому кадрі або зображенні обличчя, він може запустити цей скрипт із зазначенням відповідного файлу та шляху до моделі. Скрипт завантажує зображення, виконує детекцію та кроп обличчя, нормалізацію, прогін через нейромережу і виводить ймовірності класів real та fake. Для наочності доцільно використовувати зображення, отримані з тих же відео, що аналізуються в основному режимі, або спеціальні приклади реальних і синтетичних облич. Цей інструмент корисний насамперед на етапі налагодження та досліджень, коли потрібно переконатися, що модель адекватно реагує на окремі патерни артефактів, а також для ручної перевірки граничних випадків.

Під час роботи з моделлю варто враховувати кілька практичних аспектів. Якість вхідного відео суттєво впливає на стабільність детекції: дуже низька роздільна здатність, сильне розмиття, значні перекриття обличчя можуть призводити до того, що детектор обличчя не знаходить валідні області, а отже, кадрові ймовірності не обчислюються. У такій ситуації система виведе

попередження про недостатню кількість кадрів з обличчям, а відеорівневе рішення буде відсутнє або ненадійне. З іншого боку, надмірно агресивне повторне кодування та шум можуть підвищувати невизначеність моделі, що відобразиться на ентропії кадрових передбачень і, відповідно, на поведінці ентропійно-зваженого голосування. Ці ефекти важливо враховувати при інтерпретації результатів та порівнянні різних відео.

Для повноцінної роботи моделі бажано дотримуватися послідовності дій: спочатку перевірити коректність встановлення залежностей і працездатність базових скриптів (наприклад, короткий прогін test_single_image.py на одному зображенні), потім протестувати аналіз декількох відео через detect_from_video.py з текстовим інтерфейсом, а вже після цього використовувати графічний інтерфейс для щоденної роботи або демонстрацій. Такий підхід дозволяє оперативно локалізувати можливі проблеми на рівні окремих модулів і уникнути ситуацій, коли користувач бачить лише повідомлення про помилку в GUI без розуміння її джерела.

Загалом, інструкція користувача зводиться до того, що система надає два взаємодоповнювальні способи роботи: інтуїтивно зрозумілий графічний інтерфейс для одиничних перевірок та демонстрацій і більш гнучкі сценарії командного рядка для дослідницьких і інтеграційних задач. Обидва режими використовують один і той самий ядровий конвеєр обробки відео, нейромережевий детектор та модуль часової агрегації, що гарантує узгодженість результатів і дозволяє переходити від експериментів до прикладного використання без зміни основної логіки роботи моделі [52–55].

3.4 Дослідження та тестування розробленої моделі

Дослідження розробленої моделі виявлення дипфейків проводилося з метою перевірити, наскільки запропонований конвеєр «просторовий детектор + часова агрегація log-odds та ентропійно-зваженим голосуванням» здатний

стабільно працювати на відеоданих різної якості, а також наскільки його поведінка узгоджується з очікуваннями, сформульованими у теоретичній частині. Окрему увагу було приділено не лише точності класифікації окремих роликів, а й стійкості до спотворень (компресія, шум, розмиття), поведінці порогових рішень при зміні робочої точки, а також інтерпретованості та каліброваності вихідних ймовірностей.

Базою для досліджень виступили готові моделі на основі архітектури Xception, які були попередньо натреновані на наборах, подібних до FaceForensics++ та DFDC, із використанням стандартного конвеєра підготовки зображень: детекції облич, аффінного вирівнювання, нормалізації інтенсивностей та подання на вхід згортковій мережі. Доступ до цих моделей забезпечувався через функцію model_selection, що дозволяла підключати різні варіанти ваг, не змінюючи загальну структуру коду. Такий підхід відповідає сучасній практиці створення бенчмарків для оцінки дипфейк-детекторів, коли порівняння методів проводиться на основі узгоджених протоколів та наборів даних.

На першому етапі було проведено перевірку коректності конвеєра на рівні окремих зображень. За допомогою скрипта test_single_image.py завантажувалися поодинокі кадри, як реальні, так і синтетичні, для яких заздалегідь було відомо, до якого класу вони належать. Це дозволило переконатися, що препроцесинг (детекція облич, кроп, масштабування, нормалізація) працює узгоджено з очікуваннями моделі, а вихідні ймовірності real/fake мають адекватний вигляд: на типових прикладах реальних облич модель дає високі значення для класу real, тоді як на явно синтетичних – для класу fake. На цьому етапі також було перевірено чутливість моделі до простих спотворень: зменшення роздільної здатності, додавання гаусівського шуму, легке розмиття. Візуальний аналіз результатів (відображення зображення з підписом ймовірностей) дозволив відсіяти некоректні налаштування та переконатися, що модель поводиться стабільно (рис 3.2).

Рисунок 3.2 – Результат роботи конвеєра

Другий етап досліджень стосувався повноцінної роботи на відео. Для цього використовувався скрипт detect_from_video.py, який послідовно зчитує кадри з відеофайлу, виконує детекцію найбільшого обличчя за допомогою dlib, формує нормалізований кроп, проганяє його через Xception-детектор і накопичує кадрові ймовірності підробки у список frame_fake_probs. Паралельно на кадр накладаються рамка навколо обличчя та числове значення fake-ймовірності, після чого кадр записується у вихідний ролик (рис. 3.3). Таким чином, по завершенні обробки кожного відео доступні як «сирі» кадрові скори, так і анотований ролик, за яким можна візуально проаналізувати поведінку моделі в різні моменти часу.

Рисунок 3.3 – Скріншот кадру з вихідного анотованого відео

У цьому ж модулі реалізовані дві функції, що відповідають за часову агрегацію. Лог-правдоподібна інтеграція log_odds_aggregate інтерпретує кожну кадрову ймовірність як logit (log-odds), усереднює логіти по всіх кадрах, а потім повертає їх у ймовірнісний простір через сигмоїду. Таким чином, відеорівнева оцінка відображає середній рівень доказів за клас «fake» з точки зору байєсівської логіки сумування лог-правдоподібностей. Ентропійно-зважене голосування entropy_weighted_aggregate додатково враховує невизначеність моделі на кожному кадрі: через двокласову ентропію обчислюється вага кадру, яка зменшується для передбачень, близьких до випадкових (ймовірність близько 0.5), і збільшується для впевнених оцінок. Така схема дозволяє автоматично знижувати вплив кадрів із розмиттям, сильним стисненням або помилковим вирівнюванням.

Одним із ключових завдань дослідження було порівняти декілька варіантів часової агрегації при фіксованому кадровому детекторі. Для цього на одних і тих самих роликах розглядалися три схеми: просте усереднення кадрових ймовірностей, log-odds інтеграція та ентропійно-зважене усереднення. Всі три варіанти легко реалізуються поверх списку frame_fake_probs, тому їх можна

безпосередньо порівняти в коді, не змінюючи структуру моделі. Виявилося, що просте усереднення є найбільш чутливим до поодиноких «збоїв» моделі: якщо в ролик із реальним обличчям потрапляє кілька кадрів із нетиповою позицією або розмиттям, де детектор помиляється в бік fake, середнє значення може суттєво зсунутися, особливо якщо кількість кадрів із обличчям невелика. Застосування log-odds інтеграції частково розв'язує цю проблему, оскільки поодинокі логіти не здатні домінувати над сумою, і для зміни знаку відеорівневої оцінки потрібна послідовність узгоджених помилкових передбачень.

Ще більш стійким у цьому сенсі виявилося ентропійно-зважене голосування: кадри з високою невизначеністю (ймовірність близько 0.5) отримують ваги, близькі до нуля, тож навіть якщо модель іноді «коливається» на важких для класифікації фрагментах, їхній вплив на підсумковий скор мінімальний. У той же час кадри з чіткими, впевненими передбаченнями визначають поведінку відеорівневої оцінки (рис. 3.4). Це особливо добре проявилося на записах зі значною варіацією умов освітлення та якості: навіть коли детектор на окремих кадрах поводився нестабільно, ентропійно-зважена оцінка давала більш інтуїтивно зрозумілі результати, ближчі до експертного судження про те, чи є ролик підробленим [55, 56].

Рисунок 3.4 – Результати рішення часової агрегації

Окремий блок експериментів був присвячений впливу деградацій відео на поведінку моделі. Для цього бралися первинні ролики і до них штучно застосовувалися різні трансформації: сильніший JPEG-стік, додаткове розмиття, зменшення роздільної здатності. Після цього ті самі ролики пропускалися через конвеєр, і порівнювалися не лише значення відеоскорів, а й перелік кадрів, на яких модель демонструвала найвищу впевненість. Було відмічено, що при надмірній компресії просторові ознаки втрачають частину дискримінаційної

інформації, проте загальна тенденція зберігається: у реальних роликах впевненість моделі переважно зміщується в бік real, а в синтетичних – у бік fake. При цьому спад якості проявляється насамперед у тому, що зростає невизначеність на окремих фрагментах, а не у систематичному «переломі» рішення. Це повністю співзвучно висновкам робіт, присвячених стійкості дипфейк-детекторів до різних видів спотворень [49]. Додатково було помічено, що частотна гілка зберігає більшу чутливість на сильно стиснутих роликах, тоді як просторовий детектор швидше втрачає впевненість, що ще раз підтверджує доцільність гібридного підходу. Практично це дає змогу задавати допустимі пороги якості вхідного відео й коректно інтерпретувати результати моделі в умовах реальних KYC-сценаріїв.

На рівні калібрування вихідних ймовірностей було відмічено, що без додаткових процедур (наприклад, температурного масштабування або баєсового бінінгу) модель має тенденцію до помірної «перевпевненості» на деяких підмножинах даних: скори близькі до 0 або 1 не завжди відповідають таким же крайнім емпіричним часткам. У відповідь на це в лог-odds схемі була передбачена можливість використання параметра temperature, що дозволяє згладити розподіл логітів і тим самим зменшити ступінь впевненості без зміни відносного порядку відеоскорів. Подальші дослідження у цьому напрямку можуть включати повноцінне застосування відомих методів калібрування (Temperature Scaling, Bayesian binning into quantiles тощо) до відеорівневих оцінок, що розглядалося у попередніх роботах [55, 56].

Загалом проведені експерименти показали, що розроблений конвеєр, який базується на просторовому CNN-детекторі та двох комплементарних схемах часової агрегації, здатний формувати стабільні й інтерпретовані рішення на рівні відео як на контрольних наборах, так і на більш «живих» KYC-подібних записах. Log-odds інтеграція забезпечує узгоджене з байєсівською логікою узагальнення кадрових доказів, тоді як ентропійно-зважене голосування дозволяє додатково враховувати невизначеність моделі й підвищує стійкість рішень у складних умовах зйомки та кодування. Юзерський інтерфейс та анотовані відео, своєю

чергою, роблять ці результати наочними й полегшують як дослідницький аналіз, так і прикладну інтерпретацію роботи моделі [49, 52–56].

3.5 Перспективи подальших досліджень

Розроблена в роботі комп’ютерна модель є, по суті, робочим прототипом частотно-просторового відеодетектора дипфейків з простою, але інтерпретованою часовою агрегацією. Вона показала, що навіть на базі вже наявних CNN-архітектур Xception-типу та відносно простих схем log-odds інтеграції й ентропійно-зваженого голосування можна отримати стабільні рішення на рівні відео в умовах, наближених до KYC-сценаріїв. Водночас спектр можливих напрямів розвитку як методичної частини, так і програмної реалізації залишається дуже широким. Подальші дослідження можуть бути спрямовані на поглиблення частотного аналізу, удосконалення моделей невизначеності та калібрування, розширення мульти-модальності, а також на переведення прототипу у промисловий формат сервісу відеоверифікації [52–56].

Один із природних напрямів розвитку – заміна використаної «монолітної» CNN-базової моделі більш сучасними архітектурами, які вже на етапі проєктування враховують частотну специфіку синтезованих зображень. У літературі останніх років з’явилася низка frequency-aware мереж і генеративних моделей, які по-іншому формують спектральний вміст. Їх систематична інтеграція в детекторах потребує окремого аналізу [49, 55]. У контексті цієї роботи логічним кроком було б вбудувати частотну гілку не лише у вигляді окремих DCT-статистик, а й як повноцінний підмережевий блок із навчуваними фільтрами в спектральному домені й механізмами attention між просторовими та частотними ознаками. Це дозволило б дослідити, наскільки глибше узгодження двох доменів зменшує доменні зсуви та підвищує узагальнюваність на нові типи генераторів, що постійно з’являються в сучасних системах синтезу облич [55, 56].

Ще одним перспективним напрямом є подальший розвиток часової компоненти. У поточній реалізації послідовність кадрів редукується до одного скаляра через агрегування незалежних оцінок. При цьому не використовується інформація про реальну часову структуру ролика: плавність міміки, закономірність рухів голови, стабільність міжкадрових кореляцій. Розширення моделі в бік рекурентних чи трансформерних архітектур, які явно працюють із послідовностями (наприклад, моделі на кшталт M2TR чи інших мультимодальних трекерів), дозволило б розглядати виявлення дипфейків як задачу послідовнісної класифікації з багатим простором часових патернів [55]. У межах такого підходу log-odds та ентропійно-зважена інтеграція можуть бути переосмислені як додаткові, більш інтерпретовані «шари прийняття рішення» над виходами послідовнісної моделі, а не як єдиний механізм агрегації.

Окремий блок перспектив пов’язаний з моделюванням невизначеності й калібруванням ймовірностей. У роботі було показано, що навіть без складних процедур калібрування просте температурне масштабування логітів дозволяє частково коригувати «перевпевненість» моделі [55]. Подальші дослідження могли б включати систематичне порівняння різних методів – Temperature Scaling, Bayesian binning into quantiles, енсамблеві схеми, варіаційні підходи – з точки зору їх впливу саме на відеорівневі рішення в задачах KYC. Важливим є також дослідження поведінки детектора під час атак на довіру системи: навмисні спроби підібрати ролики, на яких модель дає високу ймовірність real, але при цьому має велику епістемічну невизначеність. У таких сценаріях коректна оцінка й інтерпретація невизначеності стає не менш важливою, ніж «середня» точність класифікації [55, 56]. Окремий інтерес становить побудова метрік для кількісної оцінки якості калібрування саме у відео-KYC (наприклад, варіанти ECE/Brier-score на рівні роликів), що дозволить більш обґрунтовано вибирати робочі точки та політики прийняття рішень. Це, у свою чергу, створює передумови для розробки risk-aware систем, де остаточне рішення залежить не тільки від ймовірності класу, а й від довіри до цієї оцінки.

З точки зору мульти-модальності, перспективним є поєднання розробленого відеодетектора з допоміжними модулями Liveness / PAD та аудіо-візуальної узгодженості, про які йшлося в теоретичній частині. Інтеграція rPPG-сигналів, аналізу синхрону між мовленням та мімікою, а також класичних підходів до протидії presentation attack (Face Anti-Spoofing) дозволить побудувати комплексну систему відеоверифікації, де дипфейк-детектор виступає лише однією з ланок, а підсумкове рішення приймається на основі сукупності ортогональних ознак [50–55]. У програмному плані це означає розширення існуючого конвеєра новими гілками обробки, розробку єдиної схеми інтеграції різних модальностей (наприклад, через байєсівські мережі або лог-лінейні моделі) та збір розмічених наборів даних, у яких одночасно присутні інформація про дипфейк, спуфінг та інші типи атак. Додатковим напрямком може стати дослідження стратегій прийняття рішень на рівні всієї сесії взаємодії з користувачем (кілька роликів, повторні спроби, додаткові сигнали), де мульти-модальні ознаки агрегуються в часі й між запитами. Це дозволить будувати більш гнучкі та захищені KYC-конвеєри, здатні протистояти комплексним, комбінованим атакам.

Нарешті, важливим практичним напрямом є перехід від дослідницького прототипу на Python до промислової сервісної архітектури. Попередньо розглянутий у роботі шлях через експорт моделі в ONNX і подальший запуск на ONNX Runtime у середовищі .NET відкриває можливість розгортання детектора як мікросервісу з REST-інтерфейсом або як частини бекенд-інфраструктури фінансової чи KYC-платформи [53–56]. У цьому контексті подальші дослідження можуть включати оптимізацію продуктивності (квантування, прунінг, використання GPU/FPGA/ASIC-акселераторів), організацію системи логування й моніторингу якості в реальному часі, механізми безперервного довчання на нових даних клієнтів із дотриманням вимог приватності. Перспективним є також впровадження механізмів A/B-тестування для різних конфігурацій детектора безпосередньо в продукційному середовищі, що дозволить емпірично оцінювати виграш від нових версій моделі. Такі завдання

виходять за межі суто алгоритмічного аспекту, але є ключовими для реального впровадження технологій виявлення дипфейків у промислові KYC-системи.

У підсумку можна зазначити, що запропонована в дипломній роботі модель частотно-просторової агрегації для відеодетекції дипфейків є лише першим кроком у великому класі задач, пов’язаних із надійністю, пояснюваністю та стійкістю систем відеоверифікації. Подальший розвиток у напрямках удосконалення архітектури, моделювання невизначеності, мульти-модальної інтеграції та промислової реалізації дозволить не лише підвищити точність і робастність детектора, а й сформувати повноцінний інструментарій для захисту відеоідентифікації в реальних інформаційних системах [52–56]. Важливо, що отриманий прототип може слугувати відправною точкою як для подальших академічних досліджень, так і для спільних проєктів із фінтех- або govtech-компаніями, де питання автентичності відео є критичними для безпеки та довіри користувачів.

ВИСНОВКИ

У кваліфікаційній роботі виконано комплексне дослідження методів виявлення дипфейків у зображеннях та відео для задач віддаленої верифікації особи. Проведений огляд наукових джерел та відкритих бенчмарків (FaceForensics++, DFDC, DeepfakeBench тощо) дозволив систематизувати сучасні підходи до детекції підроблених медіа, видокремити їх сильні сторони й недоліки, а також окреслити вимоги до промислового застосування у KYC/PAD-сценаріях: відтворюваність, робастність до компресії та зйомки в «польових» умовах, коректне калібрування і керований баланс помилок FAR/FRR. Окрему увагу було приділено саме тим аспектам, які найчастіше ігноруються в академічних роботах, але критичні для реального впровадження: доменні зсуви, складні режими освітлення, неоднорідна якість відео, обмеження по продуктивності.

Теоретично обґрунтовано та детально описано дві взаємодоповнювальні моделі. Перша – гібридний кадровий детектор «простір+частота», що поєднує CNN-ембеддинги з просторовими текстурними ознаками та спектральні дескриптори на основі 2D-DCT/FFT із глобальними й локальними енергетичними статистиками. Така комбінація експлуатує комплементарність доменів: просторове подання чутливе до геометричних і текстурних дефектів, тоді як частотне – до характерних мікроартефактів генерації та повторної компресії. Другу модель становить часова агрегація кадрових оцінок на рівні ролика, що поєднує лог-правдоподібну інтеграцію (сума «доказів» на користь класу) з ентропійно-зваженим голосуванням, яке знижує вплив невпевнених або неякісних кадрів. Для обох моделей подано покрокові алгоритми, блок-схеми, правила вибору порогів та індикаторів якості/невизначеності, що створює цілісний методичний базис для подальшого доопрацювання й масштабування.

Порівняльний аналіз показав, що гібридний кадровий детектор забезпечує вищу чутливість до різних типів підробок, зокрема в умовах сильних кодувальних артефактів, тоді як лог-правдоподібна інтеграція дає стабільне та

добре каліброване відеорішення на довгих послідовностях. Додавання ентропійно-зваженого голосування підвищує стійкість до доменних зсувів, часткової дестабілізації трекінгу та локальних розмиттів. Запропонована політика узгодження рішень дозволяє керовано балансувати безпеку й зручність у реальних процесах відеоверифікації, тобто адаптувати пороги під різні режими роботи системи – від більш «жорсткого» антифрод-контролю до комфортнішої взаємодії з доброчесними користувачами.

Наукова новизна роботи полягає у цілісному поєднанні частотно-просторового кадрового подання з двоканальною часовою агрегацією та у формулюванні практичних правил калібрування і порогування для KYC/PAD-сценаріїв. Практична значущість визначається можливістю безпосередньої інтеграції розроблених моделей у типові конвеєри віддаленої ідентифікації (детекція й вирівнювання облич, екстракція ознак, агрегація, прийняття рішень), а також використанням відкритих протоколів оцінювання, що забезпечують відтворюваність результатів і прозоре порівняння з альтернативами. Реалізований програмний прототип на базі Python, PyTorch та OpenCV з GUI-інтерфейсом демонструє, що запропоновані підходи можуть бути вбудовані у реальні програмні комплекси без радикальної перебудови існуючої інфраструктури.

Обмеження дослідження пов’язані з варіативністю «нових» генеративних методів, можливими доменними зсувами (інша оптика/освітлення/кодеки) та вимогами до коректного калібрування ймовірностей. Запропоновано напрями подальших робіт: розширення частотних дескрипторів (включно з багатошкальними та фазовими ознаками), активне навчання на складних кейсах, більш глибокі схеми оцінювання невизначеності, мультисенсорна інтеграція з PAD-сигналами та валідація на розширених, доменно різнорідних вибірках.

Результати роботи апробовано у вигляді 2 тез доповідей під час IX International Scientific and Theoretical Conference «The process and dynamics of the scientific path» [55] , IX International Scientific and Theoretical Conference «Science of XXI century: development, main theories and achievements»[56].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). FaceForensics++: Learning to detect manipulated facial images. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1–11.

2. Dolhansky, B., Howes, R., Pflaum, B., Baram, N., & Ferrer, C. C. (2020). The DeepFake Detection Challenge (DFDC) dataset. URL: https://arxiv.org/abs/2006.07397 (дата звернення 25.10.2025)

3. Qian, Y., Yin, G., Sheng, L., Chen, Z., & Shao, J. (2020). Thinking in frequency: Face forgery detection with Fourier transforms. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol.34(07), pp. 1–9.

4. Шафроненко, А. Ю. (2023). *Адаптивні методи нечіткої кластеризації даних на основі еволюційного самонавчання, pp 20-31.*

5. Tan, C., et al. (2024). Frequency-Aware Deepfake Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 1–9.

6. Afchar, D., Nozick, V., Yamagishi, J., & Echizen, I. (2018). MesoNet: A compact facial video forgery detection network. *IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–7.

7. Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1251–1258.

8. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520.

9. Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 6105–6114.

10. Творошенко, І. С. (2021). *Технології прийняття рішень в інформаційних системах: навч. посібник*, pp 47-62.

11. Кобилін, О. А., & Творошенко, І. С. (2021). *Методи цифрової обробки зображень.* pp. 57-82.

12. Karpathy, A., Toderici, G., Shetty, S., et al. (2014). Large-scale video classification with convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1725–1732.

13. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* New York: Springer, pp 130-162.

14. Goodfellow, I., Pouget-Abadie, J., Mirza, M., et al. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, pp. 2672–2680.

15. Bodyanskiy, Ye. V., Shafronenko, A. Yu., & Holovin, A. V. (2023). Clustering of data arrays based on a modified Grey Wolf Optimizer algorithm. *Radio Electronics, Computer Science, Control*, pp. 7–18.

16. Bodyanskiy, Ye. V., Shafronenko, A. Yu., & Pliss, I. A. (2021). Credibilistic fuzzy clustering based on evolutionary method of crazy cats. *System Research and Information Technologies*, pp. 110–119.

17. Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 6840–6851.

18. Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4401–4410.

19. Гороховатський, В. О., & Творошенко, І. С. (2022). *Аналіз багатовимірних даних за описом у формі множини компонент,* pp. 59-75.

20. Shafronenko, A., Bodyanskiy, Ye., & Rudenko, D. (2020). *Neuro-fuzzy clustering of distorted data using Cat Swarm Optimization.* Saarbrücken: LAP LAMBERT Academic Publishing.

21. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Cluster representation of the structural description of images for effective classification. *Computers, Materials & Continua*, vol. 73(3), pp. 6069–6084.

22. Tariq, S., Lee, S., et al. (2023). A survey on deepfake detection: Challenges and future directions. *ACM Computing Surveys*, vol. 55(14), pp. 1–38.

23. Verdoliva, L. (2020). Media forensics and deepfakes: An overview. *IEEE Journal of Selected Topics in Signal Processing*, vol. 14(5), pp. 910–932.

24. Gawlikowski, J., Tassi, C. R. N., Ali, M., et al. (2021). A survey of uncertainty in deep neural networks. *Information Fusion*, vol. 71, pp. 3–40.

25. Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1321–1330.

26. Гороховатський, В. О., Творошенко, І. С., & Чмутов, Ю. В. (2022). Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень. *Сучасні інформаційні системи*, vol. 6(3), pp. 5–12.

27. Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well-calibrated probabilities using Bayesian binning into quantiles. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2901–2907.

28. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Al-Dhaifallah, M. (2022). Classification of images based on a system of hierarchical features. *Computers, Materials & Continua*, vol. 72(1), pp. 1785–1797.

29. Durall, R., Keuper, M., & Keuper, J. (2020). Watch your up-convolution: CNN-based generative deep neural networks are failing to reproduce spectral distributions. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1–10.

30. Frank, J., Eisenhofer, T., Schönherr, L., et al. (2020). Leveraging frequency analysis for deep fake image recognition. *Proceedings of the ICML Workshop on AI for Social Good*, pp. 1–6.

31. Nataraj, L., Mohammed, T., Manjunath, B. S., et al. (2019). Detecting GAN-generated imagery using color cues. *Electronic Imaging*, pp.1–7.

32. Bayar, B., & Stamm, M. C. (2016). A deep learning approach to universal image manipulation detection using a new convolutional layer. *Proceedings of the*

2016 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec), pp. 5–10.

33. Nguyen, H. H., Yamagishi, J., & Echizen, I. (2019). Capsule-Forensics: Using capsule networks to detect forged images and videos. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2307–2311.

34. Yang, X., Li, Y., & Lyu, S. (2019). Exposing deep fakes using inconsistent head poses. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8261–8265.

35. Qi, H., Guo, X., Li, G., et al. (2020). DeepRhythm: Exposing deepfakes with attentional visual heart rate estimation. *Proceedings of the ACM International Conference on Multimedia (ACM MM)*, pp. 4318–4327.

36. Sabir, E., Cheng, J., Jaiswal, A., et al. (2019). Recurrent convolutional strategies for face manipulation detection in videos. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1–7.

37. Wu, X., AbdAlmageed, W., & Natarajan, P. (2020). Face X-ray for more general face forgery detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5001–5010.

38. Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? A new model and the Kinetics dataset (I3D). *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4733.

39. Wang, X., Girshick, R., Gupta, A., & He, K. (2018). Non-local neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7794–7803.

40. Korshunov, P., & Marcel, S. (2018). Deepfakes: A new threat to face recognition? Assessment and detection.

41. Haliassos, A., Vougioukas, K., Petridis, S., & Pantic, M. (2021). Lips don't lie: A generalisable audio-visual method for facial expression synthesis detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11764–11773.

42. Yu, N., Davis, L. S., & Fritz, M. (2019). Attributing fake images to GANs. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7556–7566.

43. Bappy, J. H., Sim, J., Park, S., & Roy-Chowdhury, A. K. (2019). Hybrid LSTM and encoder–decoder architecture for detection of image forgeries. *IEEE Transactions on Image Processing*, vol. 28(7), pp. 3286–3300.

44. Zhou, P., Han, X., Morariu, V., & Davis, L. S. (2017). Two-stream neural networks for tampered face detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1831–1839.

45. Li, Y., Chang, M.-C., & Lyu, S. (2018). In Ictu Oculi: Exposing AI-created fake videos by detecting eye blinking. *IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–7.

46. Li, Y., & Lyu, S. (2019). Exposing deepfake videos by detecting face warping artifacts. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1–9.

47. Yang, Z., Li, R., Qi, H., et al. (2022). M2TR: Multi-modal transformer for deepfake detection. *European Conference on Computer Vision Workshops (ECCV-W)*, pp. 1–16.

48. Wang, S.-Y., Wang, O., Zhang, R., Owens, A., & Efros, A. A. (2020). CNN-generated images are surprisingly easy to spot… for now. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8695–8704.

49. Yan, Z., Huang, L., Yang, X., et al. (2023). DeepfakeBench: A comprehensive benchmark of deepfake detection. *NeurIPS Datasets & Benchmarks Track*, pp. 1–12.

50. Liu, Y., Jourabloo, A., & Liu, X. (2020). A survey of face anti-spoofing. *ACM Computing Surveys*, vol. 52(3), pp. 1–38.

51. Patel, K., Parsai, M., Koshti, A., et al. (2021). Benchmarking face presentation attack detection. *Pattern Recognition Letters*, vol. 147, pp. 50–57.

52. Bradski, G., & Kaehler, A. (2016). *Learning OpenCV 3*. Sebastopol, CA: O'Reilly Media.

53. *ONNX and Azure Machine Learning*. URL: https://learn.microsoft.com/uk-ua/azure/machine-learning/concept-onnx?view=azureml-api-2 (дата звернення 20.10.2025).

54. *.NET 8 Documentation*. URL: https://learn.microsoft.com/ru-ru/dotnet/fundamentals/ (дата звернення 20.10.2025).

55. Махно, Р. А. (2025). *Частотно-просторова агрегація для виявлення дипфейків у відеоверифікації. IX International Scientific and Theoretical Conference «Science of XXI century: development, main theories and achievements»*, pp. 106-110.

56. Махно, Р. А. (2025). *Ентропійно-зважене голосування та лог-правдоподібна інтеграція для детекції дипфейків. IX International Scientific and Theoretical Conference «The process and dynamics of the scientific path»*, pp. 111-115.