НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут прикладного системного аналізу

Кафедра штучного інтелекту

До захисту допущено:

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи і методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Система розпізнавання штучно згенерованих облич на основі згорткових нейронних мереж»

Виконав:
студент IV курсу, групи КІ-01
Богельський Богдан Ярославович _____

Керівник:
кандидат технічних наук, доцент,
Тимошенко Юрій Олександрович _____

Консультант з економічного розділу:
Доцент ЕК ФММ,
Шевчук О. А. _____

Консультант з нормоконтролю:
фахівець першої категорії кафедри ШІ,
Кравець П.В. _____

Рецензент:
Доктор технічних наук, професор
Дичка Іван Іванович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«31» січня 2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Богельський Богдан Ярославович

1. Тема роботи «Генерація штучних зображень облич у інтернеті та їх ідентифікація», керівник роботи Тимошенко Юрій Олександрович, кандидат технічних наук, доцент, затверджені наказом по університету від «31» травня 2024 р. №2240-с.
2. Термін подання студентом роботи «10» червня 2024 року.
3. Вихідні дані до роботи
4. Зміст роботи
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук Олена Анатоліївна, професор, д. е. н.		

7. Дата видачі завдання «05» лютого 2024 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми дипломної роботи (ДР)	15.04.2024 - 21.04.2024	Виконано
2	Ознайомлення зі структурою ДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ»	22.04.2024 - 28.04.2024	Виконано
3	Ознайомлення з ДСТУ 3008:2015 та стандартами ЄСПД	29.04.2024 - 03.05.2024	Виконано
4	Проведення дослідження за темою ДР	04.05.2024 - 12.05.2024	Виконано
5	Проведення роботи над теоретичною частиною ДР	12.05.2024 - 22.05.2024	Виконано
6	Проведення роботи над програмним продуктом	22.05.2024 - 03.06.2024	Виконано
7	Оформлення ДР та аналіз отриманих результатів	04.06.2024 – 10.06.2024	Виконано

Студент Богдан БОГЕЛЬСЬКИЙ

Керівник Юрій ТИМОШЕНКО

РЕФЕРАТ

Дипломна робота: 110 с., 32 рис, 11 табл., 44 посилання, 1 додаток.

ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ГЛИБИННЕ НАВЧАННЯ, АВТОМАТИЧНЕ РОЗПІЗНАВАННЯ ОБЛИЧ, НЕМАКСИМАЛЬНЕ СТИСНЕННЯ, БАГАТОЗАДАЧНІ ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ.

Об’єкт дослідження – методи детекції штучно згенерованих облич.

Предмет дослідження – системи детекції штучно згенерованих облич.

Мета роботи – побудувати адекватну модель на основі загорткових нейронних мереж з використанням не максимального стиснення для детекції штучно згенерованих облич.

ABSTRACT

Master's thesis: 110 p., 32 figures, 11 tables, 44 references, 1 appendix.

CONVOLUTIONAL NEURAL NETWORKS, DEEP LEARNING, AUTOMATIC FACE DETECTION, NON-MAXIMUM SUPPRESSION, MULTI-TASK CONVOLUTIONAL NEURAL NETWORKS.

The object of the study is methods of detection of artificially generated faces.

The subject of research is systems for detecting artificially generated faces.

The purpose of the work is to build an adequate model based on convolutional neural networks using non-maximal compression for detecting artificially generated faces.

Зміст

ВСТУП

У сучасному світі технології штучного інтелекту та машинного навчання швидко розвиваються, створюючи нові можливості та виклики. Одним із значущих напрямків є генерація штучних зображень облич, що стало можливим завдяки розвитку методів глибинного навчання, таких як Generative Adversarial Networks (GAN) та Variational Autoencoders (VAE). Ця проблема є особливо актуальною через зростання кількості так званих "Deepfake" зображень, які можуть використовуватися для шахрайства, маніпуляцій та поширення дезінформації. Технології Deepfake дозволяють створювати надзвичайно реалістичні підробки, які важко відрізнити від справжніх зображень або відео, що ставить під загрозу довіру до цифрового контенту.

Мета даної роботи полягає у розробці ефективних методів та алгоритмів для ідентифікації штучно згенерованих зображень облич. Реалізація поставленої мети здійснюватиметься через впровадження сучасних методів машинного навчання та алгоритмів глибинного навчання, таких як MobileNetV2, ResNet50 та VGG16, що дозволить отримати високоякісні результати з використанням ефективних програмних засобів.

Структура роботи включає кілька основних розділів:

1. Аналіз задачі розпізнавання штучно згенерованих облич: У цьому розділі буде розглянуто основні аспекти технології технічного зору, історія виникнення та розвитку штучних зображень, а також генеративні моделі штучного інтелекту, які використовуються для їх створення. Окрему увагу буде приділено етичним та безпековим аспектам пов'язаним з використанням штучно згенерованих зображень, а також їх потенційним небезпекам.
2. Розробка методів та алгоритмів для ідентифікації штучно згенерованих облич: Цей розділ присвячений огляду сучасних методів ідентифікації, налаштуванню та тренуванню моделей на базі

MobileNetV2, ResNet50 та VGG16. Будуть описані теоретичні аспекти цих алгоритмів, їх практичне застосування для ідентифікації штучних зображень, а також результати тестування та валідації моделей.

3. Реалізація системи ідентифікації штучно згенерованих облич: У цьому розділі буде представлено опис програмного забезпечення, яке реалізує запропоновані методи та алгоритми. Буде розглянуто процес підготовки даних, налаштування гіперпараметрів, а також результати тестування системи на різних наборах даних.
4. Функціонально-вартісний аналіз програмного продукту: Цей розділ присвячений економічному обґрунтуванню розробки, аналізу функціональних можливостей програмного продукту, а також оцінці варіантів реалізації з точки зору їх економічної ефективності.

Висновки до кожного розділу підсумовують основні результати та досягнення, отримані в ході виконання дипломної роботи. Завдання роботи, таким чином, полягає у створенні надійної системи для ідентифікації штучно згенерованих облич, яка може бути використана для підвищення безпеки та довіри до цифрового контенту.

Завдання дипломної роботи.

1. Аналіз підходів, методів та алгоритмів генерації та ідентифікації штучних зображень облич.
2. Аналіз програмних засобів для реалізації програмної системи.
3. Розробка програмного забезпечення для ідентифікації штучно згенерованих зображень облич.
4. Тестування розробленого програмного забезпечення.

РОЗДІЛ 1 АНАЛІЗ ЗАДАЧІ РОЗПІЗНАВАННЯ ШТУЧНО ЗГЕНЕРОВАНИХ ОБЛИЧ

Цей розділ присвячений детальному аналізу проблеми генерації та ідентифікації штучних зображень облич у сучасному цифровому середовищі. Швидкий розвиток технологій штучного інтелекту, зокрема в галузі технічного зору та глибинного навчання, призвів до виникнення нових методів створення штучних зображень, відомих як Deepfake. Такі зображення здатні створювати реалістичні підробки, які важко відрізнити від справжніх, що становить серйозну загрозу для інформаційної безпеки та довіри до цифрового контенту.

У цьому розділі розглядаються основні аспекти технології технічного зору, історія виникнення та розвитку штучних зображень, а також генеративні моделі штучного інтелекту, які використовуються для їх створення. Окрему увагу приділено етичним та безпековим аспектам, пов'язаним з використанням штучно згенерованих зображень, а також їх потенційним небезпекам. Нарешті, підводяться підсумки актуальності теми, що обґрунтовують необхідність подальших досліджень у цій галузі.

1.1 Вступ до технології технічного зору

Технологія технічного зору є одною з найбільш швидкозростаючих напрямків у сфері штучного інтелекту. Це дозволяє комп'ютерам отримувати, обробляти та аналізувати візуальну інформацію з навколишнього світу подібно до того, як це робить людина. Основними завданнями технічного зору є розпізнавання об'єктів, виявлення та відстеження руху, аналіз сцен, а також інтерпретація зображень та відео [1].

Основи технічного зору включають кілька ключових компонентів:

— захоплення зображень, процес отримання візуальних даних за допомогою камер або інших сенсорів;
— обробка зображень, первинна обробка, що включає корекцію зображень, видалення шуму, покращення контрасту та інших параметрів;
— аналіз зображень, витягування характеристик зображення, таких як контури, текстури, кольори, які використовуються для подальшого розпізнавання;
— розпізнавання об'єктів, ідентифікація та класифікація об'єктів на зображенні за допомогою алгоритмів машинного навчання;
— інтерпретація сцени, розуміння взаємодії між об'єктами в сцені та їхнього контексту [5].

Технічний зір знаходить застосування в різних галузях, включаючи промисловість, медицину, автомобільну індустрію, робототехніку та багато інших. Деякі з основних прикладів застосування включають:

— автономні транспортні засоби, розпізнавання доріг, знаків, пішоходів та інших транспортних засобів;
— медичні зображення, діагностика захворювань на основі аналізу рентгенівських знімків, МРТ та УЗД [6];
— безпека, розпізнавання осіб, виявлення підозрілих об'єктів.
— виробництво, контроль якості продукції, автоматизація.

Однією з важливих задач технічного зору є розпізнавання облич, включаючи виявлення штучно згенерованих зображень облич. Це завдання стало особливо актуальним у зв'язку з розвитком технологій Deepfake, які використовують генеративні моделі для створення фальшивих, але реалістичних зображень облич. Для розпізнавання таких облич використовуються спеціалізовані алгоритми машинного навчання та глибинного навчання, такі як MobileNetV2 [7]. Вона є легкою та ефективною моделлю для завдань

класифікації та виявлення об'єктів, включаючи розпізнавання облич. Вона базується на архітектурі глибинних згорткових нейронних мереж, що дозволяє зменшити обчислювальні витрати без втрати точності [8].

Розпізнавання штучно згенерованих облич стикається з проблемою створення надзвичайно реалістичних зображення, які важко відрізнити від справжніх [9]. Генеративні моделі постійно вдосконалюються, тому алгоритми розпізнавання повинні бути достатньо гнучкими, щоб адаптуватися до нових типів підробок. Часто штучно згенеровані зображення використовуються у низькій якості або з навмисно введеними артефактами для ускладнення розпізнавання [10].

Рисунок 1.1 – Приклад реалістичних зображень згенерованого за допомогою GAN[1]

Технічного зору є критично важливою складовою сучасних систем штучного інтелекту, забезпечуючи комп'ютерам можливість розуміти та інтерпретувати візуальну інформацію. Вона відкриває нові можливості для автоматизації та покращення різних процесів у багатьох галузях, від медицини до автомобільної індустрії. Однак розпізнавання штучно згенерованих облич

[1] Джерело зображення: https://evergreens.com.ua/ua/articles/gan.html

залишається викликом, що вимагає постійного вдосконалення алгоритмів і моделей для забезпечення надійного захисту від підробок. Дослідження і розробка нових методів розпізнавання залишаються пріоритетним напрямком у забезпеченні інформаційної безпеки та боротьбі з дезінформацією.

1.2 Виникнення та розвиток штучних зображень

Розвиток технологій штучного інтелекту та машинного навчання призвів до появи нових можливостей у створенні та маніпуляції зображеннями. Одним з найцікавіших і найпотужніших напрямків цього розвитку є генерація штучних зображень, зокрема зображень облич. Це стало можливим завдяки створенню та вдосконаленню генеративних моделей, таких як Generative Adversarial Networks (GANs) та Variational Autoencoders (VAEs). Цей підрозділ розглядає історію виникнення та еволюцію технологій генерації зображень, а також їхній вплив на сучасний світ.

Історія розвитку технологій штучних зображень тісно пов'язана з розвитком обчислювальних потужностей та алгоритмів машинного навчання. Перші спроби створення штучних зображень були здійснені ще в 1960-х роках, але вони були обмежені простими геометричними формами та низькою роздільною здатністю [11]. Проте, справжній прорив у цій сфері стався з появою алгоритмів глибокого навчання у 2010-х роках.

Однією з перших революційних технологій у цій галузі стала розробка алгоритмів автоенкодерів, які дозволили відтворювати вхідні дані у стиснутому вигляді. Autoencoders були використані для зменшення шуму в зображеннях, створення нових зразків на основі наявних даних та інших завдань [12]. Вони стали основою для подальшого розвитку більш складних моделей, таких як Variational Autoencoders (VAEs) та Generative Adversarial Networks (GANs).

1.2.1 Генеративно-змагальні мережі (Generative Adversarial Networks)

Однією з найважливіших технологій, що сприяли розвитку генерації штучних зображень, стали Generative Adversarial Networks (GANs). GANs були запропоновані Ієном Гудфеллоу та його колегами у 2014 році [13]. Ця технологія використовує два нейронні мережі – генератор та дискримінатор – які змагаються одна з одною. Генератор намагається створити реалістичні зображення, тоді як дискримінатор намагається відрізнити реальні зображення від підроблених. Цей процес з часом призводить до того, що генератор навчається створювати дуже реалістичні зображення. GANs складаються з двох основних компонентів, а саме з генератора, який є нейронною мережею, що створює нові зображення, намагаючись зробити їх якмога більш реалістичними, та з дискримінатора, який також є нейронною мережею, яка намагається відрізнити реальні зображення від згенерованих.

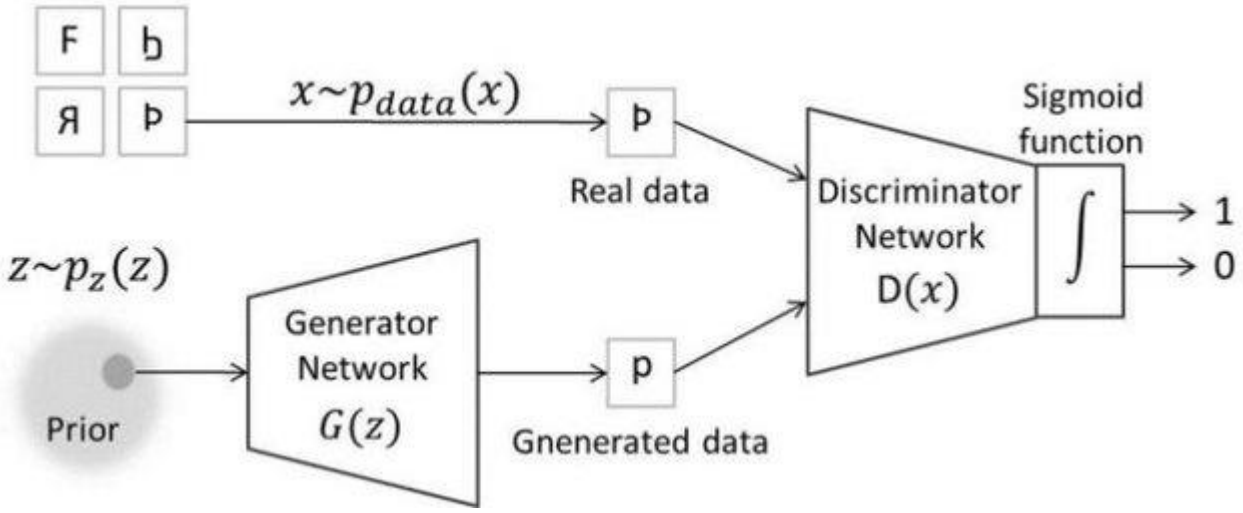

Рисунок 1.2 – Схема роботи Generative Adversarial Networks[2]

[2] Джерело зображення: https://www.researchgate.net/publication/338550526_Infrared_and_Visible_Image_Fusion_wit

Цей підхід виявився надзвичайно ефективним, і GANs швидко стали основним методом генерації реалістичних зображень. Серед найбільш відомих моделей GAN є StyleGAN, який був розроблений дослідниками з NVIDIA[9].

Моделі на основі генеративно-змагальних мереж можуть бути використані для створення навчальних даних, що містять як реальні, так і фальшиві зображення. Це дозволяє покращити тренування дискримінаторів для виявлення підробок. GAN можуть також використовуватися для виявлення більш складних і витончених підробок, які не можуть бути виявлені традиційними методами.

1.2.2 Варіаційні автокодери (Variational Autoencoders)

Variational Autoencoders (VAEs) – ще один важливий клас генеративних моделей, який дозволяє створювати нові зображення, схожі на ті, що входять до навчального набору. Цей клас базується на ідеї латентного простору, в якому кожне зображення представляється у вигляді вектору низької розмірності. Модель Він складається з двох частин: енкодера та декодера. Однією з ключових його особливостей є здатність до інтерполяції між різними зображеннями у латентному просторі, що дозволяє створювати нові, проміжні зображення, а також забезпечувати стабільність у навчанні та добре підходять для обробки великих масивів даних, що робить їх корисними у багатьох практичних застосуваннях. [14].

---

h_a_Generative_Adversarial_Network_and_a_Residual_Network/figures?lo=1

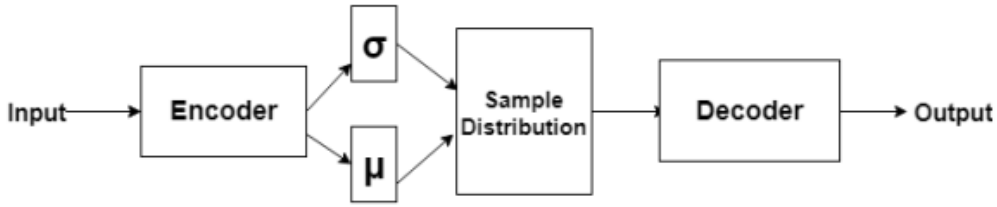

Рисунок 1.3 – Схема роботи Variational Autoencoders[3]

Таблиця 1.1 – Порівняння GANs та VAEs

Технологія	Переваги	Недоліки
GANs	Висока якість згенерованих зображень, здатність до навчання	Важкість у навчанні, потреба у великій кількості даних
VAEs	Стабільність у навчанні, добре підходить для обробки медичних зображень	Низька якість згенерованих зображень у порівнянні з GANs

1.2.3 Інші технології для створення штучних зображень

Крім GANs та VAEs, існує ряд інших технологій, які використовуються для генерації штучних зображень. Наприклад, Flow-based models та PixelCNN дозволяють генерувати зображення з високою роздільною здатністю та деталізацією [15]. Ці моделі використовують різні математичні підходи для моделювання розподілу даних та генерації нових зразків.

Flow-based models, такі як RealNVP та Glow, використовують обернені нейронні мережі для перетворення вхідних даних у латентний простір з можливістю оберненого перетворення. Це дозволяє отримувати дуже детальні та високоякісні зображення. Flow-based models використовують інвертовані нейронні мережі для перетворення даних у латентний простір і назад. Це

[3] Джерело зображення: https://theailearner.com/2018/11/10/variational-autoencoders/

дозволяє моделювати складні розподіли даних і забезпечує високу якість генерації. Однією з популярних моделей є Glow, яка дозволяє генерувати реалістичні зображення з високою роздільною здатністю [16].

PixelCNN використовують згорткові нейронні мережі для моделювання умовної ймовірності кожного пікселя зображення. Це дозволяє генерувати зображення піксель за пікселем, що забезпечує високу точність і контроль над процесом генерації. PixelCNN добре підходять для завдань, де важлива висока деталізація зображень [17].

Сама генерація штучних зображень має широкий спектр застосувань у різних галузях:

— створення спецефектів для кіно, генерація нових персонажів для ігор. Наприклад, у фільмах використовуються CGI (Computer Generated Imagery) для створення реалістичних персонажів та сцен, що було б неможливо зняти традиційними методами [18];
— аналіз та синтез медичних зображень для діагностики та лікування. Генеративні моделі допомагають створювати реалістичні моделі органів для навчання медичних працівників або для планування хірургічних втручань [6];
— розпізнавання облич, виявлення підробок у документах. Технології генерації зображень використовуються для створення тестових даних для навчання систем розпізнавання облич та для виявлення фальшивих документів [19].

Проте, з появою таких технологій з'явилися і нові виклики. Зокрема, використання технологій для створення Deepfake зображень і відео стало серйозною загрозою для приватності та безпеки. Ці технології дозволяють створювати реалістичні фальшиві відео, які можуть використовуватися для дезінформації та маніпуляцій. Наприклад, в одному з відомих випадків, Deepfake відео було використано для створення фальшивих заяв політиків, що могло призвести до серйозних наслідків для суспільства [2];

Крім того, генеративні моделі можуть бути використані для створення фальшивих доказів у судових справах або для шантажу, що підриває правову систему та довіру до цифрових доказів. Це ставить нові вимоги до технологій виявлення підробок та розробки методів захисту від зловживань. Використання таких технологій для шахрайства, наприклад створення фальшивих профілів у соціальних мережах, може призвести до серйозних наслідків для приватного життя людей та їх репутації.

Рисунок 1.4 – Приклад Deepfake зображення:зліва реального та справа штучно згенерованого обличчя[4]

Майбутнє технологій генерації штучних зображень обіцяє ще більше інновацій та викликів. Основні напрямки розвитку включають постійні дослідження спрямовані на підвищення якості та реалістичності згенерованих зображень. В свою чергу, генеративні моделі повинні адаптуватися до нових типів даних та викликів, зокрема для виявлення та боротьби з Deepfake. Це включає розробку нових методів для ідентифікації підробок та захисту від

[4] Джерело зображення https://builtin.com/machine-learning/real-face

зловживань. Наприклад, необхідно створювати алгоритми, які можуть автоматично виявляти фальшиві зображення та відео, аналізуючи аномалії та невідповідності [10]. Обов’язково повинні розроблятися етичні норми та правові регулювання для запобігання зловживанням технологіями генерації зображень.

Розвиток технологій генерації штучних зображень відкрив нові горизонти для наукових досліджень та комерційних застосувань. Проте, з цією технологією з'являються і нові виклики, які потребують вирішення. Це вимагає подальших досліджень і розробок для забезпечення надійності та безпеки використання штучно згенерованих зображень.

1.3 Методи ідентифікації штучних зображень облич

З огляду на зростаючу загрозу використання штучно згенерованих зображень облич, розробка надійних методів для їх виявлення є критично важливою. Це питання стає особливо актуальним у зв'язку з поширенням технологій, таких як Deepfakes, які дозволяють створювати реалістичні, але підроблені відео та зображення. Подібні технології можуть використовуватись для дезінформації, шантажу та інших злочинних цілей, що становить серйозну загрозу для приватного життя, громадської безпеки та довіри до цифрових медіа.

Традиційні методи ідентифікації фальшивих зображень облич включають аналіз піксельних аномалій, текстурних характеристик та геометричних невідповідностей. Ці методи використовують ручні ознаки для виявлення ознак підробки, таких як нерівномірність кольорів, неприродні переходи або неправильне розташування елементів обличчя. Проте, з розвитком технологій генерації зображень, ці методи стають менш ефективними, оскільки сучасні алгоритми можуть створювати зображення з високою точністю і реалістичністю.

Одним з прикладів традиційних методів є аналіз аномалій на рівні пікселів. Цей метод передбачає виявлення невідповідностей у кольорах та текстурах, які

можуть бути ознаками підробки. Інший підхід – аналіз геометрії обличчя, який дозволяє виявляти невідповідності в розташуванні очей, носа, рота та інших елементів обличчя. Проте, ці методи мають обмежену точність і можуть бути обдурені сучасними генеративними моделями.

1.3.1 Згорткові нейронні мережі

Сучасні методи ідентифікації фальшивих зображень облич все частіше використовують алгоритми глибинного навчання. Ці методи дозволяють автоматично витягувати складні ознаки з зображень, що підвищує точність виявлення підробок. Глибинне навчання відзначається здатністю обробляти великі обсяги даних і знаходити складні закономірності, що робить його ефективним для ідентифікації фальшивок.

Одним з найбільш поширених підходів є використання згорткових нейронних мереж. CNN дозволяють автоматично витягувати та аналізувати текстурні та просторові характеристики зображень, що робить їх ефективними для виявлення підробок. Наприклад, моделі ResNet і Inception успішно використовуються для ідентифікації фальшивих зображень облич завдяки їхній здатності витягувати високорівневі ознаки [20].

Моделі на основі CNN можуть бути спеціально натреновані для виявлення фальшивих зображень. Вони здатні аналізувати текстурні невідповідності, які не видно людському оку, і виявляти ознаки підробки з високою точністю. Такі моделі можуть використовуватися для автоматичного моніторингу контенту в соціальних мережах та інших платформах, де поширення фальшивих зображень може мати серйозні наслідки.

1.3.2 Рекурентні нейронні мережі

Іншим підходом є використання рекурентних нейронних мереж (RNN), які можуть враховувати часові залежності у відео. Це особливо корисно для виявлення фальшивих відео, де важливо аналізувати послідовності кадрів для виявлення невідповідностей у рухах та виразах облич. Наприклад, Long Short-Term Memory (LSTM) мережі можуть використовуватися для аналізу відеопослідовностей і виявлення фальшивих відео [21].

RNN можуть аналізувати динаміку рухів та змін обличчя у відео, що дозволяє виявляти невідповідності, які можуть бути ознаками підробки. Ці моделі можуть бути використані для перевірки автентичності відеозаписів, наприклад, у журналістиці або судових справах.

1.3.3 Проблеми та виклики

Крім CNN, RNN та GAN, існують інші підходи до ідентифікації фальшивих зображень облич. Наприклад, методи на основі аналізу аномалій можуть використовувати статистичні моделі для виявлення невідповідностей у зображеннях. Також можуть використовуватися методи на основі фрагментації зображень, які дозволяють аналізувати окремі частини зображення на предмет аномалій.

Але однією з головних проблем у розробці методів ідентифікації лишається постійне вдосконалення алгоритмів генерації зображень. Це означає, що методи виявлення повинні постійно оновлюватися та адаптуватися до нових типів підробок. Крім того, ефективність методів виявлення може знижуватися через обмежену кількість навчальних даних та їхню якість.

Ще однією важливою проблемою є брак універсальних тестових наборів даних, які б дозволяли оцінити ефективність різних методів виявлення підробок. Це ускладнює порівняння результатів різних досліджень та впровадження нових методів у практику. Необхідно розробляти стандартизовані тестові набори даних, які б містили різноманітні типи підробок, щоб забезпечити об'єктивну оцінку ефективності методів.

Крім того, виникають етичні та правові питання, пов'язані з використанням технологій для виявлення фальшивих зображень. Наприклад, використання таких методів у соціальних мережах може призвести до порушення приватності користувачів та зловживання технологіями для цензури.

Подальший розвиток методів ідентифікації штучних зображень облич пов'язаний з удосконаленням існуючих алгоритмів та створенням нових підходів, які можуть враховувати специфічні характеристики фальшивих зображень. Наприклад, використання гібридних моделей, які поєднують різні типи нейронних мереж, може підвищити точність виявлення підробок. Також важливим напрямком є розробка алгоритмів, які можуть автоматично оновлюватися та навчатися на нових даних, що забезпечує їхню актуальність у довгостроковій перспективі [22].

Іншим перспективним напрямком є використання методів багатофакторної аутентифікації, які поєднують аналіз зображень з іншими біометричними характеристиками, такими як голос, рухи та інші поведінкові ознаки. Це дозволить створювати більш надійні системи для виявлення підробок, які будуть важко обдурити.

Важливим напрямком досліджень є також розробка методів для виявлення фальшивих зображень у реальному часі. Це дозволить виявляти підробки під час їхнього поширення в інтернеті та соціальних мережах, що допоможе швидко реагувати на загрози та мінімізувати їхній вплив.

Ідентифікація штучних зображень облич є важливим завданням у сучасному світі, де технології генерації фальшивих зображень постійно вдосконалюються. Використання глибинного навчання, зокрема CNN, RNN та

GAN, відкриває нові можливості для ефективного виявлення підробок. Проте, постійний розвиток технологій потребує безперервного вдосконалення методів ідентифікації, що є ключовим для забезпечення безпеки та довіри до цифрового контенту. Розробка нових методів та їхнє впровадження у практику допоможе забезпечити надійний захист від загроз, пов'язаних з використанням фальшивих зображень облич.

1.4 Етичні та безпекові аспекти штучно згенерованих зображень

Генерація штучних зображень за допомогою технологій штучного інтелекту, зокрема за допомогою Generative Adversarial Networks (GANs) та Variational Autoencoders (VAEs), несе значні етичні та безпекові виклики. Основні ризики та виклики, пов'язані з використанням штучно згенерованих зображень, включають можливість зловживання технологіями Deepfake, порушення приватності, підрив довіри до медіа та інформаційних ресурсів, а також загрози для безпеки.

Одним з найсерйозніших викликів, пов'язаних з генерацією штучних зображень, є можливість зловживання технологіями Deepfake. Deepfake використовують генеративні моделі для створення реалістичних фальшивих відео та зображень, які можуть бути використані для обману, маніпуляцій та дезінформації. Наприклад, Deepfake можуть бути використані для створення фальшивих відеозаписів політиків, що можуть призвести до серйозних політичних та соціальних наслідків. Відомі випадки використання Deepfake для створення компрометуючих матеріалів, що можуть поширювати недостовірну інформацію і викликати широкий суспільний резонанс .

Іншою важливою проблемою є порушення приватності. Генеративні моделі можуть бути використані для створення підроблених зображень людей без їхньої згоди, що може призвести до компрометації особистої інформації. Наприклад, можна створити фальшиві інтимні зображення людини і поширювати їх в інтернеті, що може мати серйозні юридичні наслідки і підривати довіру до цифрових технологій .

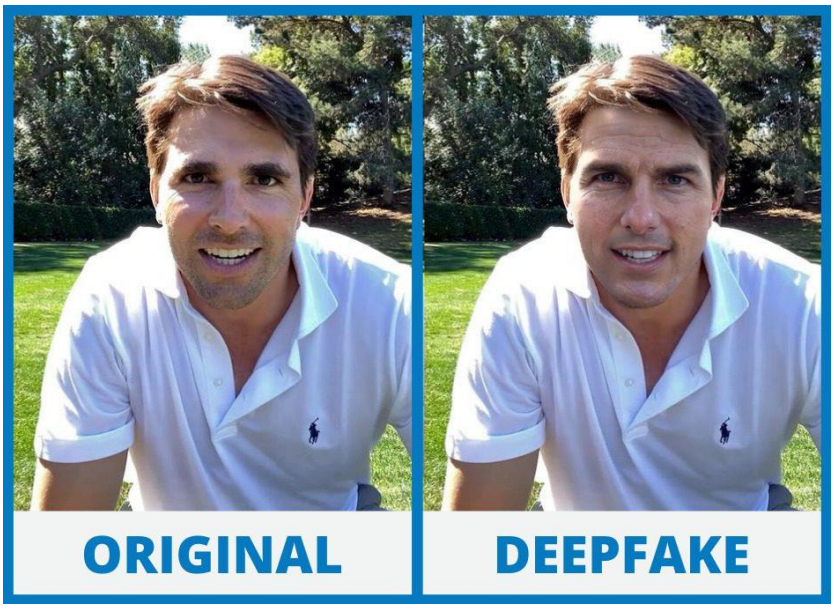

Рисунок 1.5 – Приклад використання Deepfake для створення фальшивих відео[5]

Штучно згенеровані зображення можуть підривати довіру до медіа та інших джерел інформації. Якщо користувачі не можуть бути впевнені в автентичності зображень та відео, це може призвести до зниження довіри до новин та інших інформаційних ресурсів. Це, у свою чергу, може мати негативні наслідки для суспільства в цілому, підриваючи основи соціальної взаємодії та демократії. Наприклад, поширення фальшивих новин та маніпулятивних зображень може призвести до політичної дестабілізації, впливаючи на виборчі процеси та громадську думку .

[5] Джерело зображення: https://www.trymaverick.com/blog-posts/are-deep-fakes-all-evil-when-can-they-be-used-for-good

DeepFake технології становлять серйозну загрозу для безпеки в різних сферах життя. Вони можуть використовуватись для створення підроблених відео та аудіо матеріалів, які вводять в оману громадськість. Це може призвести до поширення дезінформації, маніпуляції громадською думкою та створення паніки в суспільстві. Наприклад, фальшиві відео або фото з політичними лідерами можуть використовуватися для дискредитації їхніх позицій, провокації конфліктів або впливу на виборчий процес. Така дезінформація підриває довіру до засобів масової інформації та офіційних джерел, що є критично важливим для стабільності суспільства.

Одним з найважливіших напрямків для вирішення проблем, пов'язаних з Deepfake, є розробка технологій для їх виявлення. Існують різні методи та алгоритми для виявлення штучно згенерованих зображень та відео, включаючи аналіз аномалій, використання моделей глибокого навчання та інших підходів. Наприклад, алгоритми на основі згорткових нейронних мереж можуть використовуватися для виявлення невідповідностей у текстурі шкіри, анімації обличчя та інших характеристик, які є типовими для Deepfake . Розробка технологій виявлення Deepfake є важливим напрямком досліджень, який може допомогти у боротьбі з фальсифікацією зображень та відео.

Підвищення обізнаності суспільства щодо можливостей та ризиків, пов'язаних з генеративними моделями, є важливим кроком для запобігання зловживанням. Освіта щодо технологій Deepfake, методів їх виявлення та способів захисту приватності допоможе користувачам краще розуміти ризики та відповідально використовувати технології. Інформування громадян про те, як розпізнавати фальшиві зображення та відео, може значно знизити ефективність зловживань цими технологіями .

У 2018 році було створено фальшиве відео, де відомий політик нібито робив заяву, якої він насправді не робив. Це викликало широкий суспільний резонанс та привернуло увагу до проблеми Deepfake. Також, під час виборів у Габоні, підозрювана Deepfake могла сприяти спробі державного перевороту. У Бельгії, політична партія створила Deepfake відео з Дональдом Трампом, яке

викликало політичний скандал, що зрештою виявилося високотехнологічною підробкою [23].

Підозрювана Deepfake відео спричинила дипломатичний конфлікт між Саудівською Аравією та Катаром. У цьому випадку було використано вигадані цитати еміра Катару, що призвело до серйозного загострення відносин між країнами.[24].

Один з найсерйозніших випадків - використання Deepfake для створення фальшивих порнографічних відео. За даними дослідження, між 90% та 95% Deepfake відео є непогоджуваними порнографічними матеріалами. Це призвело до значних психологічних та соціальних проблем для жертв, особливо жінок, чиї обличчя були використані без їхньої згоди [25].

У 2019 році в США, під час слухань у Комітеті з розвідки Палати представників, було обговорено потенційну загрозу Deepfake для майбутніх президентських виборів. Було попереджено, що Deepfake може використовуватися для поширення дезінформації та маніпуляцій виборцями.[26]

Етичні та безпекові аспекти генерації штучних зображень є критично важливими у сучасному світі. Технології, які дозволяють створювати реалістичні фальшиві зображення, несуть серйозні ризики для суспільства, приватності та безпеки. Необхідно розробляти нові методи виявлення, створювати етичні норми та регулювання, а також підвищувати обізнаність суспільства щодо цих технологій. Лише комплексний підхід дозволить мінімізувати ризики та забезпечити безпечне використання генеративних моделей штучного інтелекту.

Зокрема, розробка технологій виявлення Deepfake є важливим напрямком досліджень. Використання моделей глибинного навчання, таких як CNN, RNN, та GAN, допомагає автоматично виявляти ознаки підробки в зображеннях та відео. Такі алгоритми здатні аналізувати текстурні аномалії та динаміку рухів, що є важливим для виявлення фальшивок.

Розробка етичних норм та правових регулювань також є ключовим фактором для запобігання зловживанням технологіями генерації зображень. Законодавчі ініціативи у багатьох країнах вже спрямовані на захист громадян від

використання Deepfake без їхньої згоди, що допомагає зменшити негативні наслідки таких технологій.

Підвищення обізнаності суспільства щодо можливостей та ризиків, пов'язаних з генеративними моделями, є важливим кроком для запобігання зловживанням. Освіта щодо технологій Deepfake, методів їх виявлення та способів захисту приватності допоможе користувачам краще розуміти ризики та відповідально використовувати технології.

Зрештою, лише комплексний підхід, що поєднує технічні, етичні та освітні заходи, дозволить мінімізувати ризики та забезпечити безпечне використання генеративних моделей штучного інтелекту. Це забезпечить захист суспільства від негативних наслідків фальшивих зображень та зміцнить довіру до цифрових технологій.

1.7 Постановка задачі

Розробка ефективної системи ідентифікації штучно згенерованих облич є актуальною задачею у сучасному контексті боротьби з дезінформацією та забезпечення цифрової безпеки. Завдання полягає в тому, щоб створити надійну програмну систему, здатну точно розпізнавати фальсифіковані зображення серед великої кількості реальних.

Основні цілі постановки задачі дипломного проектування.

1. Аналіз існуючих методів та підходів. Вивчення сучасних підходів до ідентифікації штучно згенерованих облич, включаючи традиційні методи аналізу зображень та глибинні нейронні мережі. Це дозволить визначити найбільш ефективні та відповідні для нашої задачі алгоритми та архітектури.
2. Вибір моделей нейронних мереж. На основі аналізу методів

обґрунтувати вибір моделей нейронних мереж. Моделі які будуть точними у задачах класифікації зображень і здатністю обробляти великі обсяги даних.

3. Розробка програмного забезпечення. Створення програмної системи, що інтегрує обрані моделі нейронних мереж. Програмне забезпечення повинно мати інтерфейс для завантаження зображень, проведення їхньої класифікації та відображення результатів.
4. Підготовка та обробка даних. Збір та підготовка збалансованого датасету, що включає реальні та штучно згенеровані обличчя. Виконання нормалізації, аугментації та розділення даних на тренувальні, валідаційні та тестові набори для забезпечення ефективного навчання моделей.
5. Тестування та валідація. Проведення ретельного тестування системи на підготовленому датасеті для оцінки її точності, стабільності та здатності до узагальнення. Використання різних метрик для оцінки продуктивності, таких як точність, втрати, precision та AUC.
6. Оцінка та вдосконалення. Аналіз результатів тестування та визначення напрямків для подальшого вдосконалення системи. Це може включати оптимізацію моделей для покращення продуктивності, розширення набору даних, інтеграцію додаткових функцій та покращення інтерфейсу користувача.

Таким чином, основна мета полягає в створенні ефективної та надійної системи для автоматичної ідентифікації штучно згенерованих облич. Це дозволить забезпечити високу точність і стабільність роботи у різних сферах застосування, таких як цифрова безпека, контроль достовірності інформації та боротьба з дезінформацією.

Висновки до розділу

У першому розділі дослідження було розглянуто основні методи генерації штучних зображень облич, зокрема Generative Adversarial Networks (GAN) та Variational Autoencoders (VAE). Проведено аналіз технологій технічного зору, їх розвиток та застосування у сучасних умовах. Окрему увагу було приділено основам створення штучних зображень, їх використанню та впливу на сучасне суспільство.

Також у розділі було досліджено етичні та безпекові аспекти використання штучно згенерованих зображень, включаючи потенційні ризики та загрози. Визначено необхідність подальших досліджень та розробки методів для ідентифікації таких зображень, що є критично важливим для забезпечення інформаційної безпеки та довіри до цифрового контенту.

РОЗДІЛ 2 РОЗРОБКА МЕТОДІВ ТА АЛГОРИТМІВ ДЛЯ ІДЕНТИФІКАЦІЇ ОБЛИЧ ЗГЕНЕРОВАНИХ У ІНТЕРНЕТІ

У сучасному світі, де технології штучного інтелекту та машинного навчання швидко розвиваються, питання ідентифікації штучно згенерованих зображень стає надзвичайно актуальним. З появою технологій генерації зображень, таких як GAN (Generative Adversarial Networks), створення реалістичних зображень облич стало можливим навіть для неспеціалістів. Це викликає занепокоєння щодо можливого зловживання цими технологіями для дезінформації, фальсифікацій та інших негативних явищ.

У цьому розділі розглядаються сучасні методи та алгоритми, які використовуються для ідентифікації штучно згенерованих облич. Основна увага приділяється алгоритмам MobileNetV2, ResNet50 та VGG16, які показали високу ефективність у вирішенні завдань класифікації зображень. MobileNetV2 є однією з популярних архітектур глибинних нейронних мереж, яка використовується для мобільних та вбудованих систем завдяки своїй легкості та високій продуктивності. ResNet50 та VGG16 є також широко визнаними моделями, що використовуються для різних завдань комп'ютерного зору, включаючи ідентифікацію облич.

Серед ключових проблем, пов'язаних із виявленням штучно згенерованих зображень, є реалістичність цих зображень та постійне вдосконалення генеративних моделей. Наприклад, згідно з дослідженням [26], сучасні моделі GAN здатні створювати зображення, які важко відрізнити від справжніх, що створює серйозні виклики для ідентифікаційних алгоритмів.

Ідентифікація штучно згенерованих облич має велике значення для різних сфер, включаючи медіа, де необхідно перевіряти автентичність візуального контенту, та безпеку, де такі технології можуть бути використані для виявлення фальшивих документів та інших шахрайських дій [27].

Таким чином, у підрозділах цього розділу буде представлено огляд

сучасних методів ідентифікації штучних зображень, теоретичні аспекти алгоритмів MobileNetV2, ResNet50 та VGG16, а також практичні аспекти їх використання для ідентифікації штучно згенерованих облич. Крім того, будуть розглянуті питання налаштування та тренування нейронних мереж на базі цих алгоритмів, тестування та валідації моделей, а також обговорення отриманих результатів. Основна мета цього розділу - продемонструвати ефективність використання MobileNetV2, ResNet50 та VGG16 для ідентифікації штучних зображень облич та надати практичні рекомендації щодо їх впровадження.

Таким чином, даний розділ є важливим етапом у дослідженні проблеми ідентифікації штучних зображень, що дозволить підвищити надійність та точність автоматичних систем виявлення підробок у сучасних умовах інформаційної війни та кіберзлочинності.

2.1 Огляд сучасних методів ідентифікації

Сучасні методи ідентифікації штучно згенерованих облич включають різні підходи та техніки, які базуються на класичних алгоритмах машинного навчання та більш сучасних методах глибинного навчання. Розглянемо детально кожен з них.

Метод опорних векторів (SVM) є одним з найбільш популярних алгоритмів для задач класифікації. SVM будує гіперплощину у просторі високої розмірності, яка максимально розділяє точки різних класів. Випадки, що лежать найближче до гіперплощини, називаються опорними векторами, і саме вони визначають положення гіперплощини. Цей метод використовується для задач ідентифікації зображень завдяки своїй здатності ефективно працювати з великими обсягами даних та високою точністю класифікації. Дерева рішень: Побудова дерева для прийняття рішень, де кожне вузлове рішення приймається на основі певного

атрибуту даних.[28]

Метод опорних векторів застосовується для класифікації зображень облич, які можуть бути підробленими або справжніми. Наприклад, для задачі виявлення deepfake, SVM може використовувати ознаки, витягнуті за допомогою глибинних нейронних мереж, таких як CNN, для побудови гіперплощини, яка розділяє підроблені та справжні обличчя. SVM демонструє високу точність класифікації при обробці великих наборів даних з високою розмірністю ознак.

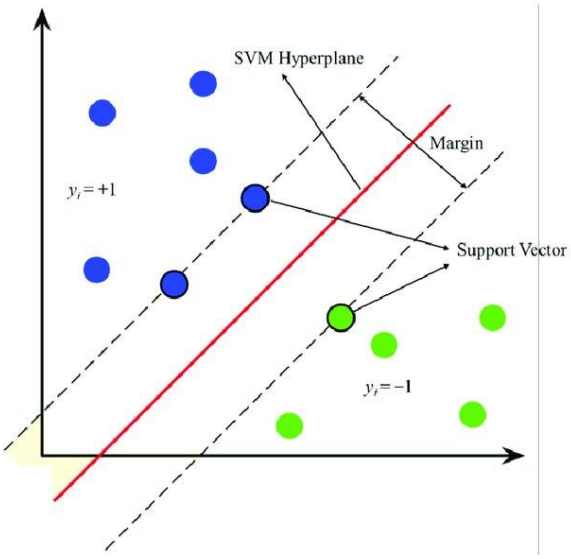

Рисунок 2.1 – Лінійний SVM класифікатора, що розділяє два класи[6]

Дерева рішень є алгоритмом, який використовує ієрархічну структуру для прийняття рішень. Кожний вузол дерева представляє атрибут даних, кожна гілка — можливе значення цього атрибуту, а кожен лист — кінцевий результат або клас. Дерева рішень легко інтерпретувати і вони добре працюють на невеликих наборах даних. Проте, вони можуть бути схильні до перенавчання на великих

[6] Джерело зображення: https://www.researchgate.net/publication/359803757_Efficient_Reliability_Analysis_of_Structures_Using_Symbiotic_Organisms_Search-Based_Active_Learning_Support_Vector_Machine/figures?lo=1

обсягах даних. [29]

Дерева рішень можуть бути використані для побудови моделей ідентифікації, які базуються на різних атрибутах зображень облич. Наприклад, модель може використовувати характеристики текстури, форми та кольору облич для прийняття рішення про автентичність зображення. Дерева рішень легко інтерпретувати, що робить їх корисними для пояснення процесу класифікації.

Наївний байєсів класифікатор є простим ймовірнісним класифікатором, який базується на застосуванні теореми Байєса з "наївним" припущенням про незалежність між ознаками. Незважаючи на своє просте припущення, наївний байєсів класифікатор може бути дуже ефективним для багатьох задач класифікації, особливо для текстових даних та задач виявлення спаму. [30]

Цей класифікатор може бути використаний для виявлення підроблених зображень, аналізуючи ймовірності наявності певних ознак у даних. Наприклад, він може використовувати ймовірнісні моделі для визначення, чи відповідають пікселі зображення типовим розподілам для справжніх облич. Наївний байєсів класифікатор часто використовується у поєднанні з іншими методами для підвищення точності класифікації.

Глибокі нейронні мережі (DNN) є багатошаровими персептронами, які мають здатність навчатися складним патернам у даних. Вони складаються з вхідного шару, декількох прихованих шарів та вихідного шару. Кожен шар складається з нейронів, які використовують нелінійні активаційні функції для обчислення вихідних значень. DNN добре справляються із задачами класифікації зображень завдяки своїй здатності виділяти складні ознаки [31].

Глибокі нейронні мережі використовуються для побудови моделей класифікації, які здатні автоматично виділяти складні ознаки з зображень. Наприклад, DNN можуть бути натреновані на великих наборах даних зображень облич для виявлення патернів, що відрізняють справжні обличчя від підроблених. Завдяки своїй потужності, DNN можуть досягати високої точності навіть у складних задачах ідентифікації.

Згорткові нейронні мережі (CNN) спеціалізуються на обробці зображень та відео. Вони використовують згорткові шари для автоматичного виділення ознак зображень. Кожен згортковий шар складається з фільтрів (ядра згортки), які ковзають по вхідному зображенню та обчислюють згортки для виділення різних ознак, таких як краї, текстури та форми. CNN також включають шари об'єднання для зменшення розмірності даних, що допомагає зменшити кількість параметрів та підвищити обчислювальну ефективність [32].

CNN використовуються для обробки зображень облич завдяки своїй здатності автоматично виділяти ознаки зображень. Наприклад, CNN можуть бути натреновані для виявлення аномалій, таких як неправильні тіні, дивні артефакти або неприродні текстури, які часто присутні на підроблених зображеннях. CNN демонструють високу ефективність у задачах ідентифікації облич завдяки своїй здатності обробляти візуальну інформацію.

Рекурентні нейронні мережі (RNN) призначені для роботи з послідовними даними, такими як відео або часові ряди. Вони мають здатність зберігати контекст попередніх даних у своїх прихованих станах, що дозволяє їм враховувати тимчасові залежності у даних. Основна особливість RNN — наявність зворотних зв'язків, які дозволяють передавати інформацію з одного кроку часу до наступного. Це робить RNN ефективними для задач, де послідовність даних має важливе значення [20].

RNN використовуються для аналізу послідовних даних, таких як відео. Наприклад, для задачі виявлення підроблених відео облич, RNN можуть аналізувати послідовності кадрів, враховуючи тимчасові залежності між ними. Це дозволяє виявляти неприродні рухи або зміни, які можуть свідчити про підробку.

Генеративні змагальні мережі (GAN) складаються з двох нейронних мереж: генератора та дискримінатора. Генератор створює нові зображення, намагаючись зробити їх якомога більш реалістичними, тоді як дискримінатор оцінює, наскільки ці зображення відрізняються від реальних. GAN використовуються для генерації реалістичних зображень, але також можуть бути застосовані для виявлення підробок. Завдяки здатності генерувати високоякісні зображення, вони стають потужним інструментом як для створення, так і для ідентифікації штучно згенерованих облич.

GAN можуть бути використані для тренування моделей, які здатні виявляти підроблені зображення, створені іншими GAN. Наприклад, дискримінатор може бути натренований для виявлення підробок, аналізуючи різні характеристики зображень. GAN також можуть бути використані для створення датасетів підроблених зображень, що допомагає покращити тренування моделей ідентифікації.

Ще одним важливим аспектом для виявлення підроблених зображень є їхня здатність до самонавчання. Завдяки ітеративному процесу навчання генератора та дискримінатора, моделі можуть адаптуватися до нових типів підробок і постійно покращувати свої навички виявлення. Це робить GAN цінним інструментом у боротьбі з новими та більш складними формами фальсифікації зображень, які постійно еволюціонують.

Після детального розгляду сучасних методів ідентифікації штучно згенерованих облич, можна зробити висновок, що найбільш перспективними виявилися методи, засновані на глибинних нейронних мережах, особливо на згорткових нейронних мережах. Саме методи глибинного навчання дозволяють ефективно виділяти складні ознаки зображень, що є критично важливим для задачі ідентифікації штучно згенерованих облич.

Таблиця 2.1 – Таблиця порівняння методів

Метод	Переваги	Недоліки	Застосування для задачі ідентифікації штучних облич
SVM	Висока точність класифікації, добре працює з малими даними	Високі обчислювальні витрати для великих обсягів даних	Обмежене використання через високу обчислювальну вартість на великих наборах даних
Дерева рішень	Легкість інтерпретації, швидке навчання	Схильність до перенавчання, низька продуктивність на великих даних	Може використовуватися як базова модель для попереднього аналізу
Наївний байєсів	Простота реалізації, швидкість навчання	Обмежена точність через припущення про незалежність ознак	Використання в комбінації з іншими методами для підвищення точності
DNN	Висока здатність виділяти складні ознаки	Вимоги до великих обсягів даних та обчислювальних ресурсів	Ефективне використання для задач ідентифікації за умови наявності великих наборів даних
CNN	Висока точність для обробки зображень, автоматичне виділення ознак	Вимоги до великих обсягів даних та обчислювальних ресурсів	Найбільш підходящий для задачі ідентифікації штучно згенерованих облич
RNN	Ефективність для послідовних даних, здатність враховувати тимчасові залежності	Складність навчання, вимоги до великих обчислювальних ресурсів	Обмежене використання для статичних зображень, більш підходить для відео
GAN	Здатність генерувати реалістичні зображення, ефективність в навчанні моделей-детекторів	Складність навчання, ризик генерації неприродних зображень	Може використовуватися для створення синтетичних наборів даних для тренування моделей

2.2 Теоретичні основи методів глибинного навчання

Методи глибинного навчання базуються на складних архітектурах нейронних мереж, які можуть навчатися на великих обсягах даних і виявляти складні патерни. У цьому розділі розглянемо основи глибинного навчання, включаючи згорткові нейронні мережі (CNN), глибокі нейронні мережі (DNN) та рекурентні нейронні мережі (RNN), а також основні принципи їх роботи та особливості, що роблять CNN найбільш підходящими для задачі ідентифікації штучно згенерованих облич.

2.2.1 Архітектура та принципи роботи глибоких нейронних мереж

Глибокі нейронні мережі (DNN) складаються з багатьох шарів нейронів, що дозволяє їм вивчати високорівневі ознаки з вхідних даних. Ключові компоненти DNN включають:

— вхідний шар: Отримує початкові дані для обробки;

— приховані шари: Виконують нелінійні перетворення вхідних даних, використовуючи активаційні функції, такі як ReLU (Rectified Linear Unit) або Sigmoid;

— вихідний шар: Генерує кінцеві результати класифікації або регресії.

Основний принцип роботи DNN полягає у використанні багатошарової архітектури, де кожен шар відповідає за виділення все більш складних ознак з вхідних даних. При цьому використовуються нелінійні активаційні функції, які додають нелінійність у модель, що дозволяє нейронній мережі вивчати більш складні патерни.

Формально, вихід нейрона у прихованому шарі можна записати як у формулі (2.1):

$$y = f(\sum_{i=1}^{n} w_i x_i + b), \tag{2.1}$$

де y – вихід нейрона;

f – активаційна функція (наприклад, ReLU);

w_i – ваги;

x_i – вхідні значення;

b – зміщення (bias).

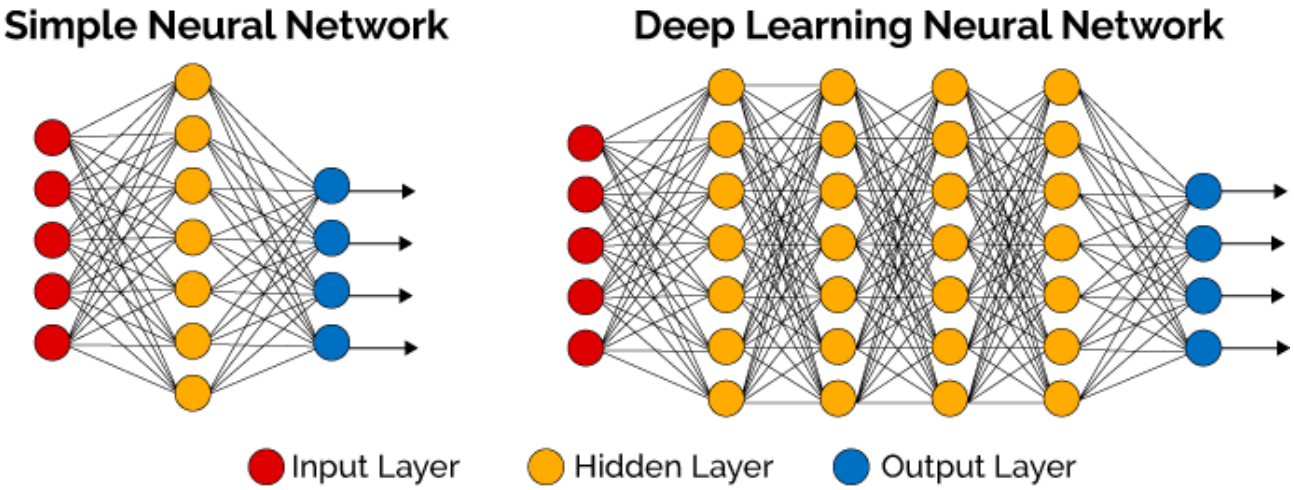

Рисунок 2.2 – Базова схематична архітектура глибокої нейронної мережі[7]

Глибокі нейронні мережі (DNN) можуть використовуватися для різних задач, включаючи класифікацію, регресію та кластеризацію. Вони добре працюють з великими наборами даних, що дозволяє їм навчатися складним патернам і закономірностям [33]. Проте, для задачі ідентифікації зображень облич вони можуть бути менш ефективними, оскільки не мають спеціалізованих механізмів для обробки візуальної інформації. Зокрема, для задач ідентифікації штучно згенерованих облич важливо враховувати особливості зображень, які можуть бути неефективно оброблені стандартними DNN.

Однією з основних проблем цих моделей є їх схильність до перенавчання, особливо коли кількість параметрів моделі значно перевищує кількість доступних навчальних даних. Це може призвести до ситуації, коли модель добре працює на тренувальних даних, але має низьку точність на нових, невідомих даних. Для подолання цієї проблеми використовуються різні техніки регуляризації, такі як Dropout або L2-регуляризація, які допомагають зменшити перенавчання та підвищити узагальнюючу здатність моделі.

DNN добре підходять для задач, де потрібно виявляти складні

---

[7] Джерело зображення: https://www.oreilly.com/library/view/python-natural-language/9781787121423/fa56fade-f7c3-494e-807e-2e1f0970d9be.xhtml

закономірності у великих наборах даних, але для задачі ідентифікації штучно згенерованих облич вони можуть поступаютися спеціалізованим методам.

2.2.2 Особливості та застосування рекурентних нейронних мереж

Рекурентні нейронні мережі призначені для обробки послідовних даних, таких як текст, аудіо або відео. Основна особливість рекурентних нейронних мереж полягає в їх здатності зберігати контекст попередніх даних у своїх прихованих станах, що дозволяє їм враховувати тимчасові залежності у даних. Це досягається через зворотні зв'язки, які передають інформацію з одного кроку часу до наступного. Основні компоненти RNN включають:

— приховані шари: містять нейрони, які отримують введення не лише з поточного часу, але й з попереднього часу, що дозволяє зберігати інформацію про попередні кроки;

— зворотні зв'язки: дозволяють передавати інформацію з одного кроку часу до наступного.

Основна формула роботи RNN може бути описана у формулі (2.2).

$$h_t = f(w_x x_t + w_h h_{t-1} + b_h), \tag{2.2}$$

де h_t – прихований стан у час t;

x_t – вхідне значення у час t;

w_x та w_h – ваги;

b_h – зміщення (bias);

f – активаційна функція, яка може бути сигмоїдною (sigmoid), гіперболічною (tanh) або ReLU [34].

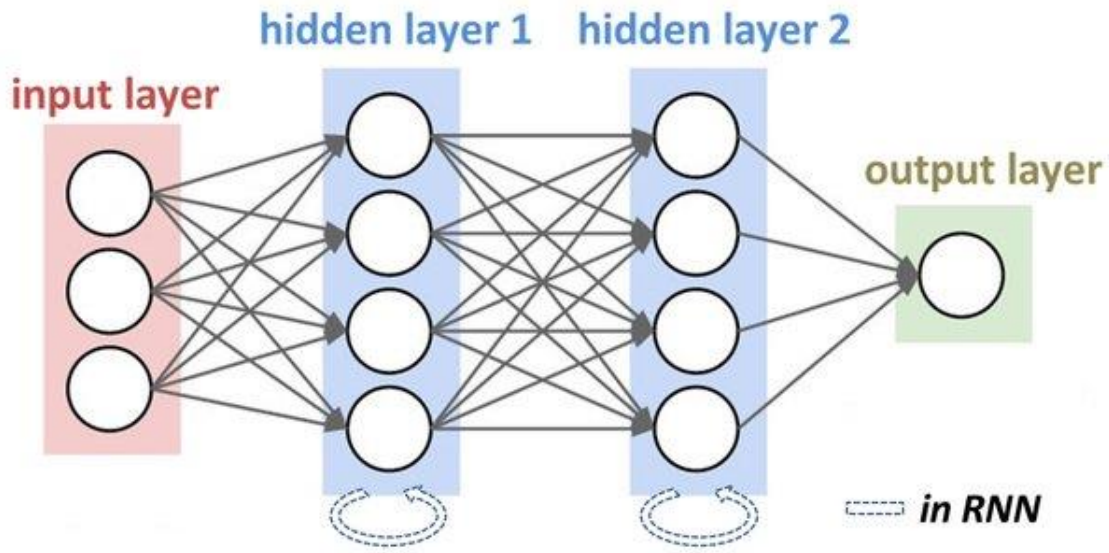

Рисунок 2.3 – Узагальнена архітектура рекурентної нейронної мережі з двома прихованими шарами[8]

Однією з основних проблем RNN є затухання градієнта, коли градієнти стають дуже малими і модель перестає навчатися. Ця проблема виникає через довгі послідовності даних, коли інформація з початку послідовності стає малозначущою для кінця. Для подолання цієї проблеми використовуються вдосконалені архітектури, такі як Long Short-Term Memory (LSTM) та Gated Recurrent Unit (GRU) [35, 36].

Рекурентні нейронні мережі підходять для задач, де важливі тимчасові залежності. Використовуються для обробки природної мови, машинного перекладу, аналізу часових рядів тощо [37]. Ці мережі є потужним інструментом для обробки послідовних даних, але задачі ідентифікації штучно згенерованих облич інакші методи та архітектури глибинного навчання є більш підходящими.

---

[8] Джерело зображення: https://www.researchgate.net/publication/335159004_Classification_and_prediction_of_wave_chaotic_systems_with_machine_learning_techniques/figures?lo=1

2.2.3 Основи та переваги згорткових нейронних мереж

Згорткові нейронні мережі (CNN) є однією з найпотужніших архітектур для обробки зображень і відео. Вони спеціально розроблені для автоматичного виділення ознак з вхідних даних завдяки використанню згорткових і об'єднувальних шарів. Основні компоненти CNN включають:

— згорткові шари (Convolutional Layers), де кожен згортковий шар має кілька фільтрів, які ковзають по вхідному зображенню та обчислюють згортки, виділяючи локальні патерни, такі як краї, текстури та форми. Операцію згортки можна описати як у формулі (2.3);

— шари об’єднання (Pooling Layers), у яких max pooling вибирає максимальне значення з кожного підвіконця матриці, зменшуючи розмірність даних і роблячи модель більш стійкою до зсувів та деформацій. A average pooling обчислює середнє значення з кожного підвіконця матриці;

— повнозв’язні шари (Fully Connected Layers), де після кількох шарів згортки та об'єднання, CNN використовують один або кілька повнозв'язних шарів для остаточного формування вихідного вектору ознак, який використовується для класифікації або іншої задачі.

$$(I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.3}$$

де I – вхідне зображення;

K – ядро згортки;

i та j – координати пікселя у вихідному зображенні.

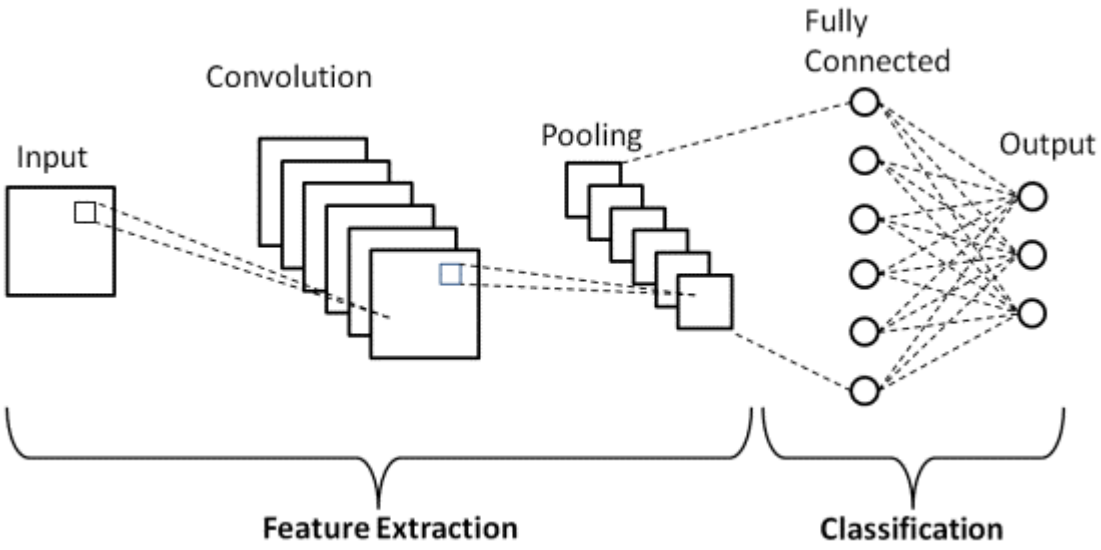

Рисунок 2.4 – Базова архітектура згорткової нейронної мережі[9]

CNN мають кілька ключових переваг, які роблять їх ідеальними для задач ідентифікації облич. Вони автоматично виділяють релевантні ознаки з вхідних зображень, що дозволяє уникнути ручного визначення ознак і спрощує процес побудови моделей. Також вони можуть бути інваріантними до зсувів, обертання та масштабування зображень завдяки використанню шарів об'єднання, що робить їх більш стійкими до різних варіацій вхідних даних. А використання згорткових шарів дозволяє значно зменшити кількість параметрів моделі в порівнянні з традиційними повнозв'язними мережами, що знижує ризик перенавчання та покращує загальну продуктивність моделі.

Якщо порівнювати DNN та CNN, то глибокі нейронні мережі демонструють потенціал у вирішенні складних задач класифікації та регресії. Проте, для задачі ідентифікації штучно згенерованих облич більш підходящими є CNN, які мають спеціалізовані механізми для обробки зображень і виділення релевантних ознак.

При порівнянні CNN та RNN, то потрібно зауважити, що у той час як CNN

[9] Джерело зображення: https://www.upgrad.com/blog/basic-cnn-architecture/

спеціалізуються на обробці просторових даних (зображення, відео), RNN оптимальні для обробки послідовних даних (текст, аудіо). Тому для вирішення задачі ідентифікації штучно згенерованих зображень облич використання згорткових нейронних мереж є цілком виправданим.

У нашій роботі ми будемо використовувати три основні архітектури CNN: MobileNetV2, ResNet50 та VGG16. Кожна з цих архітектур має свої унікальні особливості, що робить їх підходящими для різних застосувань.

MobileNetV2 є однією з найефективніших архітектур згорткових нейронних мереж, розробленою спеціально для мобільних та вбудованих систем. Вона забезпечує оптимальний баланс між продуктивністю та обчислювальними витратами, завдяки чому стає популярною для використання в різних додатках комп'ютерного зору.

Ця модель використовує кілька ключових концепцій, що роблять її ефективною та потужною. Однією з основних особливостей є використання глибинних роздільних згорток. Цей підхід дозволяє розділити традиційну згортку на два окремі етапи: глибинну згортку, яка застосовується окремо до кожного каналу вхідного зображення, та точкову згортку, яка об'єднує виходи глибинної згортки. Це значно зменшує кількість обчислень і параметрів, зберігаючи при цьому високу продуктивність моделі.

Глибинна згортка обробляє кожен канал вхідного зображення окремо, що дозволяє зменшити обчислювальні витрати, а точкова згортка виконує операцію 1x1 згортки, об'єднуючи виходи глибинної згортки. Глибину згортки можна описати формулою 2.4.

$$Y = X * K \tag{2.4}$$

де X – вхідний тензор;

K – ядро згортки.

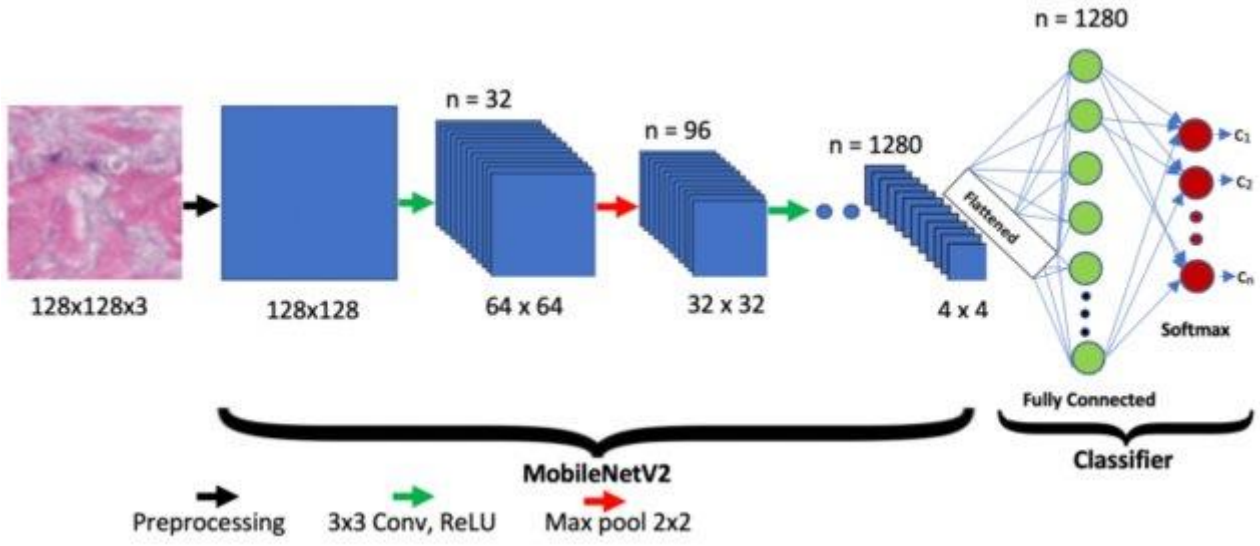

Рисунок 2.5 – Архітектура мережі MobileNetV2[10]

Наступною важливою концепцією, використаною в MobileNetV2, є інвертовані залишкові блоки. У цих блоках вхідні та вихідні дані мають низьку розмірність, тоді як проміжний шар має більшу розмірність. Це дозволяє зберігати обчислювальні ресурси та покращує здатність моделі до навчання. Інвертовані залишкові блоки включають прямі з'єднання, що забезпечують збереження інформації через кілька шарів і допомагають уникнути проблеми затухання градієнта, яка може виникати при тренуванні глибоких мереж.

MobileNetV2 також включає механізми лінійних вузьких місць, які дозволяють уникнути втрат інформації під час згорткових операцій. Це досягається за рахунок використання лінійних активацій після кожного блоку, що допомагає зберегти значущі характеристики даних та зменшує обчислювальні витрати.

Архітектура MobileNetV2 добре підходить для використання в обмежених обчислювальних середовищах, таких як мобільні пристрої та вбудовані системи.

---

[10] Джерело зображення: https://www.researchgate.net/publication/350152088_Deep_Learning_Classification_of_Systemic_Sclerosis_Skin_Using_the_MobileNetV2_Model/figures

Завдяки своїй ефективності вона дозволяє виконувати складні задачі комп'ютерного зору на пристроях з обмеженими ресурсами, забезпечуючи високу продуктивність при низьких обчислювальних витратах.

Застосування MobileNetV2 включає класифікацію зображень, виявлення об'єктів та сегментацію зображень. Вона часто використовується для реального часу аналізу зображень на смартфонах або інших портативних пристроях, що є критичним для багатьох сучасних додатків, таких як доповнена реальність (AR) та автоматичне розпізнавання облич. Наприклад, у додатках доповненої реальності MobileNetV2 дозволяє швидко та ефективно розпізнавати об'єкти та інтегрувати віртуальні елементи у реальний світ, що значно покращує користувацький досвід [4, 7, 38].

ResNet50 є потужною архітектурою глибинного навчання, яка була розроблена для подолання проблеми затухання градієнта, що виникає при тренуванні дуже глибоких нейронних мереж. Ця архітектура складається з 50 шарів і використовує концепцію залишкових блоків, які дозволяють зберігати інформацію з попередніх шарів та покращують здатність моделі до навчання.

Основна ідея ResNet полягає в тому, що кожен залишковий блок включає прямі з'єднання, які забезпечують передачу інформації через кілька шарів, що допомагає уникнути проблеми затухання градієнта. Залишкові блоки дозволяють мережі вчитися залишковим функціям, тобто різницям між вхідними даними та бажаним виходом, замість того, щоб вчитися безпосередньо цим функціям [39].

Архітектура ResNet50 складається з кількох основних компонентів. Все починається з 7x7 згорткового шару з 64 фільтрами та страйдом 2, за яким слідує шар max pooling з розміром страйду 2. Цей шар відповідає за початкову обробку вхідного зображення та зменшення його розмірності [40]. Далі йдуть п’ять блоків, кожен з яких містить кілька залишкових блоків. Кожен залишковий блок складається з трьох шарів: 1x1 згортка, 3x3 згортка, та ще одна 1x1 згортка. Ці шари дозволяють зменшити та потім збільшити кількість каналів, що допомагає зберегти обчислювальні ресурси та збільшити здатність моделі до навчання [41, 42]. Після проходження через всі залишкові блоки, вихід передається через

глобальний середній пулінг, а потім через повнозв'язний шар з 1000 нейронів, який використовує softmax активацію для класифікації зображень [41].

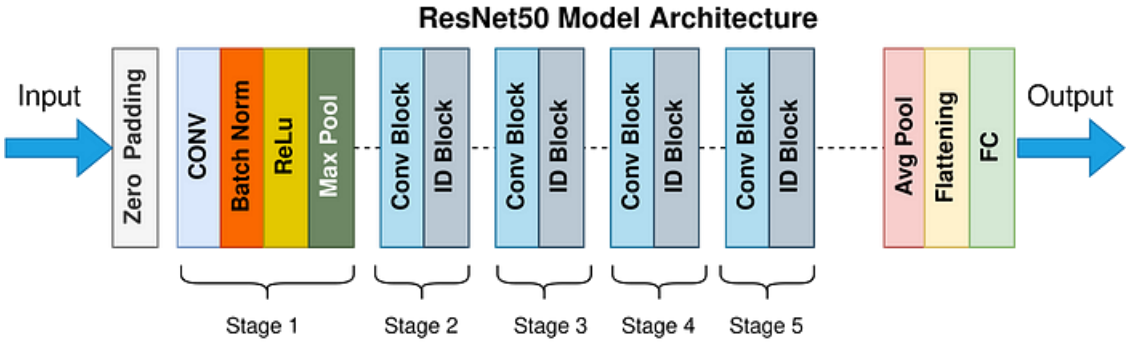

Рисунок 2.5 – Архітектура мережі ResNet50[11]

ResNet50 широко використовується для різних задач комп'ютерного зору, включаючи класифікацію зображень, виявлення об'єктів та сегментацію зображень. Завдяки своїй глибокій архітектурі та використанню залишкових блоків, ResNet50 демонструє високу точність і стабільність навчання навіть на великих наборах даних. Ця архітектура також добре підходить для задач перенесення навчання, де попередньо навчені моделі можуть бути адаптовані до нових задач з меншими обсягами даних [40].

ResNet50 є потужною архітектурою згорткових нейронних мереж, яка використовує залишкові блоки для подолання проблеми затухання градієнта та забезпечує високу продуктивність при навчанні глибоких мереж. Завдяки своїй здатності зберігати інформацію з попередніх шарів та ефективно обробляти складні патерни, ResNet50 є ідеальною для використання в різних задачах комп'ютерного зору, забезпечуючи високу точність та стабільність навчання.

[11] Джерело зображення: https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758

VGG16 є однією з найбільш відомих і широко використовуваних архітектур згорткових нейронних мереж, запропонованою Visual Geometry Group (VGG) з Оксфордського університету. Ця архітектура була представлена в рамках конкурсу ImageNet Large Scale Visual Recognition Challenge у 2014 році і відзначається своєю простою, але ефективною структурою.

Основною особливістю VGG16 є використання послідовних згорткових шарів з невеликим розміром ядра (3x3), що дозволяє моделі виявляти дрібні деталі та складні патерни в зображеннях. VGG16 складається з 16 шарів: 13 згорткових шарів і 3 повнозв'язних шари.

Архітектура VGG16 є такою, спочатку перші два згорткові шари мають 64 фільтри кожен і використовують ядра розміром 3x3 з кроком 1 і заповненням "same" (щоб зберегти розмірність вхідного зображення). Після цього застосовується шар max pooling з розміром вікна 2x2 і кроком 2. Наступні два згорткові шари мають 128 фільтрів кожен з тими ж параметрами ядра та pooling. Далі йдуть три згорткові шари з 256 фільтрами кожен, знову ж таки з тими ж параметрами ядра та pooling. Нарешті, дві групи по три згорткові шари з 512 фільтрами кожен, з аналогічними параметрами, завершуються шаром max pooling [43]. Після згорткових шарів виходи проходять через шар глобального середнього пулінгу, а потім через три повнозв'язні шари. Перші два з них мають по 4096 нейронів і використовують функцію активації ReLU. Останній повнозв'язний шар має 1000 нейронів і використовує softmax активацію для класифікації на 1000 класів ImageNet [44].

Архітектура VGG16 є відносно простою і легкою для реалізації, що робить її популярною серед дослідників та розробників. Використання послідовних згорткових шарів з фіксованим розміром ядра дозволяє досягти високої точності при обробці зображень. Завдяки своїй простій, але потужній структурі, VGG16 є відмінним вибором для багатьох реальних застосувань, включаючи медичну діагностику, автоматичне розпізнавання облич та багато інших.

Ми розглянули основи методів глибинного навчання, включаючи глибокі нейронні мережі (DNN), згорткові нейронні мережі (CNN) та рекурентні

нейронні мережі (RNN). Ці архітектури мають різні структури, принципи роботи та сфери застосування, що робить їх відповідними для різних задач.

Глибокі нейронні мережі (DNN) є базовими архітектурами, які використовують повнозв'язні шари для обробки даних. Вони підходять для задач, де просторові залежності не є критичними, але можуть бути схильними до перенавчання на великих наборах даних через велику кількість параметрів.

Згорткові нейронні мережі (CNN) спеціалізуються на обробці зображень, використовуючи локальні зв'язки та фільтри для виділення ознак. Вони ефективні для задач класифікації зображень, виявлення об'єктів та сегментації, оскільки зберігають просторову інформацію та зменшують кількість параметрів.

Рекурентні нейронні мережі (RNN) призначені для обробки послідовних даних, таких як текст або аудіо. Вони враховують тимчасові залежності, що робить їх ідеальними для задач обробки природної мови та аналізу часових рядів.

Для задачі ідентифікації штучно згенерованих облич найбільш підходящими є CNN, зокрема архітектури MobileNetV2, ResNet50 та VGG16, завдяки їх здатності виділяти просторові ознаки та обробляти великі зображення з меншими обчислювальними витратами.

2.3 Налаштування та тренування моделей

Налаштування та тренування моделей є важливими етапами в процесі розробки ефективної системи для ідентифікації штучно згенерованих облич. У цьому розділі детально розглянемо підготовку даних, налаштування гіперпараметрів, процес тренування та збереження моделей.

Перш ніж розпочати тренування моделей, необхідно підготувати дані, що включає нормалізацію зображень, їх розділення на тренувальний та тестовий набори, а також застосування аугментації даних для збільшення різноманіття

тренувальних зображень.

Нормалізація зображень є важливим кроком, оскільки зменшує розмах значень пікселів і допомагає моделям швидше та стабільніше навчатися. У нашому коді нормалізація виконується шляхом ділення кожного значення пікселя на 255, що приводить значення у діапазон від 0 до 1. Це дозволяє уникнути великих значень градієнтів, які можуть впливати на стабільність навчання.

Розділення даних на тренувальний і тестовий набори є критично важливим для оцінки продуктивності моделі на незалежних даних. У нашому коді використовується співвідношення 80:20 або 70:30, що означає, що 80% (або 70%) даних використовуються для тренування, а решта 20% (або 30%) — для тестування. Це дозволяє оцінювати узагальнюючу здатність моделі та її продуктивність на нових даних.

Аугментація даних — це техніка штучного збільшення розміру тренувального набору шляхом застосування різних перетворень до зображень, таких як обертання, зсув, масштабування, відбивання та зміна яскравості. Це допомагає моделі бути більш стійкою до варіацій вхідних даних та покращує її узагальнюючу здатність. У нашому коді використовуватимемо наступні методи аугментації:

— обертання на 20 градусів;
— горизонтальне відбивання;
— зсув зображення на 10%;
— зміна яскравості на 20%.

Гіперпараметри відіграють важливу роль у процесі навчання моделей. До основних гіперпараметрів належать кількість епох, розмір пакет, навчальна ставка та використання технік регуляризації.

Кількість епох визначає, скільки разів модель пройде через весь тренувальний набір даних. Для MobileNetV2, ResNet50 та VGG16 зазвичай використовується від 20 до 50 епох. У нашому коді встановлено значення 30

епох, що є оптимальним для уникнення перенавчання та забезпечення стабільного навчання. Більша кількість епох може призвести до перенавчання, коли модель надто точно налаштовується на тренувальні дані і втрачає здатність узагальнювати на нових даних.

Розмір пакету визначає кількість зображень, що обробляються моделлю одночасно під час одного кроку навчання. Вибір оптимального розміру пакету є компромісом між швидкістю навчання та стабільністю градієнта. У нашому коді використовується розмір пакету 32, що є типовим значенням для таких задач. Розмір пакету впливає на використання пам'яті GPU та швидкість обчислень: менші пакети зменшують навантаження на пам'ять, але можуть сповільнювати процес навчання, тоді як більші пакети можуть прискорити навчання, але вимагатимуть більше пам'яті.

Навчальна ставка контролює, наскільки швидко модель оновлює свої ваги під час навчання. Висока навчальна ставка може призвести до нестабільного навчання, тоді як надто низька ставка уповільнює процес навчання. У нашому коді використовується метод зменшення навчальної ставки з початкового значення 0.001 до менших значень протягом навчання. Це дозволяє моделі швидко навчатися на початку і поступово стабілізуватися на оптимальних значеннях ваг. Зменшення навчальної ставки протягом навчання допомагає уникнути коливань градієнтів і забезпечує стабільне сходження до оптимального рішення.

Регуляризація допомагає уникнути перенавчання моделі. До основних методів регуляризації належать Dropout та L2-регуляризація. У нашому коді використовується Dropout з ймовірністю 0.5, що дозволяє випадковим чином виключати певний відсоток нейронів під час навчання, зменшуючи таким чином перенавчання. Dropout змушує модель бути більш стійкою до втрати окремих нейронів і знижує її залежність від конкретних шляхів у нейронній мережі. Це покращує здатність моделі до узагальнення і знижує ризик перенавчання на тренувальних даних.

Процес тренування включає кілька кроків: компіляцію моделі, визначення функції втрат, вибір оптимізатора та власне навчання.

Компіляція моделі включає визначення функції втрат, оптимізатора та метрик, які будуть використовуватися для оцінки продуктивності моделі. Функція втрат вимірює розбіжність між передбаченими ймовірностями класів та справжніми мітками класів. Оптимізація функції втрат дозволяє моделі покращувати свої передбачення на тренувальних даних.

Оптимізатор відповідає за оновлення ваг моделі на основі обчислених градієнтів. У нашому коді будемо використовувати оптимізатор Adam, який поєднує в собі переваги методів адаптивного навчання та моменту. Це забезпечує швидке та стабільне навчання. Adam автоматично налаштовує навчальну ставку для кожного параметра, що дозволяє моделі швидше сходитися до оптимального рішення. Крім того, використання моменту допомагає згладжувати коливання градієнтів і прискорює процес навчання.

Навчання моделі буде виконуватися шляхом передачі тренувальних даних через модель протягом заданої кількості епох. Під час кожної епохи модель буде оновлювати свої ваги на основі обчислених градієнтів і функції втрат. Процес навчання також включає валідацію на окремому валідаційному наборі даних для моніторингу продуктивності та уникнення перенавчання. Для тренування моделей MobileNetV2, ResNet50 та VGG16 використовується метод fit, який приймає на вхід тренувальні дані, валідаційні дані, кількість епох, розмір пакету та інші параметри. Під час навчання обчислимо метрики точності та втрат, що дозволяє оцінити продуктивність моделі на тренувальному та валідаційному наборах даних. Регулярний моніторинг цих метрик допомагає вчасно виявляти ознаки перенавчання та приймати відповідні заходи для його зниження, наприклад, зменшення навчальної ставки або застосування додаткових методів регуляризації.

У цьому підрозділі ми розглянули підготовку даних, налаштування гіперпараметрів, процес тренування та збереження моделей.

Підготовка даних включає нормалізацію, розділення на тренувальні та тестові набори, і аугментацію для покращення здатності моделі до узагальнення. Налаштування гіперпараметрів, таких як кількість епох, розмір пакету, навчальна ставка та регуляризація, є критичними для ефективного навчання. Оптимальні значення цих параметрів забезпечують баланс між швидкістю навчання та якістю моделі.

Процес тренування включає компіляцію моделі з використанням оптимізаторів та функцій втрат, а також регулярний моніторинг продуктивності. Це дозволяє оцінити модель і вчасно виявити перенавчання.

Таким чином, правильна підготовка даних, оптимізація гіперпараметрів, ефективне тренування та збереження моделей є ключовими для створення надійної системи ідентифікації штучно згенерованих облич, що забезпечує високу точність та стабільність роботи.

Висновки до розділу 2

Другий розділ присвячений розробці методів та алгоритмів для ідентифікації штучно згенерованих зображень облич. Проведено огляд сучасних методів ідентифікації, включаючи MobileNetV2, ResNet50 та VGG16. Детально розглянуто теоретичні основи цих алгоритмів, їх архітектуру та принципи роботи. Окремо описано процес налаштування та тренування моделей для досягнення високої точності ідентифікації. В результаті, було встановлено, що використання цих моделей забезпечує високу точність виявлення штучно згенерованих зображень облич. Отримані результати підтверджують доцільність застосування цих моделей для вирішення завдань ідентифікації в умовах сучасних викликів, пов'язаних з поширенням Deepfake технологій.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ СИСТЕМИ ІДЕНТИФІКАЦІЇ ШТУЧНО ЗГЕНЕРОВАНИХ ОБЛИЧ

У цьому розділі буде детально розглянуто реалізацію системи ідентифікації штучно згенерованих облич, включаючи вибір та налаштування архітектур нейронних мереж, процес тренування, тестування та оцінка продуктивності моделей.

3.1 Огляд вхідних наборів даних для ідентифікації штучно згенерованих облич

Для реалізації системи ідентифікації штучно згенерованих облич ми використовуємо набір даних: “140k Real and Fake Faces”. Цей датасет включає 70 тисяч реальних облич зібраних з Flickr за допомогою Nvidia, а також 70 тисяч штучних облич, згенерованих за допомогою StyleGAN від Bojan.

Всі 70 тисяч реальних облич отримані з Flickr, що забезпечує різноманітність в умовах освітлення, виразах обличчя, ракурсах та інших параметрах. 70 тисяч штучних облич були згенеровані за допомогою моделі StyleGAN, яка відома своєю здатністю створювати фотореалістичні зображення людей, що не існують.

Всі зображення були зручно об'єднані в один датасет, змінено їх розмір до 256 пікселів, та розділено на тренувальний, валідаційний і тестовий набори для зручності використання у машинному навчанні. Набір містить файли CSV, які забезпечують зручний доступ до метаданих і дозволяють швидко завантажувати необхідні зображення для тренування, валідації та тестування моделей.

#	original_path	id	label	label_str	path
0 – 9999	20000 unique values	1 – 70.0k	0.00 - 0.05 Count: 10,000 (0 – 1)	2 unique values	20000 unique values
0	/kaggle/input/flickrfaceshq-dataset-nvidia-part-3/images1024x1024-20191222T221133Z-012/images1024x10...	18233	1	real	test/real/18233.jpg
1	/kaggle/input/flickrfaceshq-dataset-nvidia-part-3/images1024x1024-20191222T221133Z-012/images1024x10...	54317	1	real	test/real/54317.jpg
2	/kaggle/input/flickrfaceshq-dataset-nvidia-part-7/images1024x1024-20191222T221133Z-034/images1024x10...	40155	1	real	test/real/40155.jpg
3	/kaggle/input/flickrfaceshq-dataset-nvidia-part-1/images1024x1024-20191222T221133Z-004/images1024x10...	12875	1	real	test/real/12875.jpg

Рисунок 3.1 – Приклад додаткової інформації з нашого датасету

Цей датасет є ідеальним для задачі ідентифікації штучно згенерованих облич, оскільки він надає збалансовану кількість реальних та штучних зображень. Висока якість та різноманітність даних дозволяють тренувати моделі з високою точністю та узагальнюючою здатністю. Його використання дозволяє створювати ефективні моделі для ідентифікації штучно згенерованих облич, що є критично важливим у сучасному контексті боротьби з дезінформацією та забезпеченням безпеки у цифровому середовищі.

3.2 Підготовка даних до використання

Зобразимо кількість даних з першого датасету у вигляді стовпчикової діаграми (рис.3.3). Це допоможе візуально оцінити розподіл зображень між

тренувальним, валідаційним і тестовим наборами, а також перевірити збалансованість даних між реальними та штучними зображеннями.

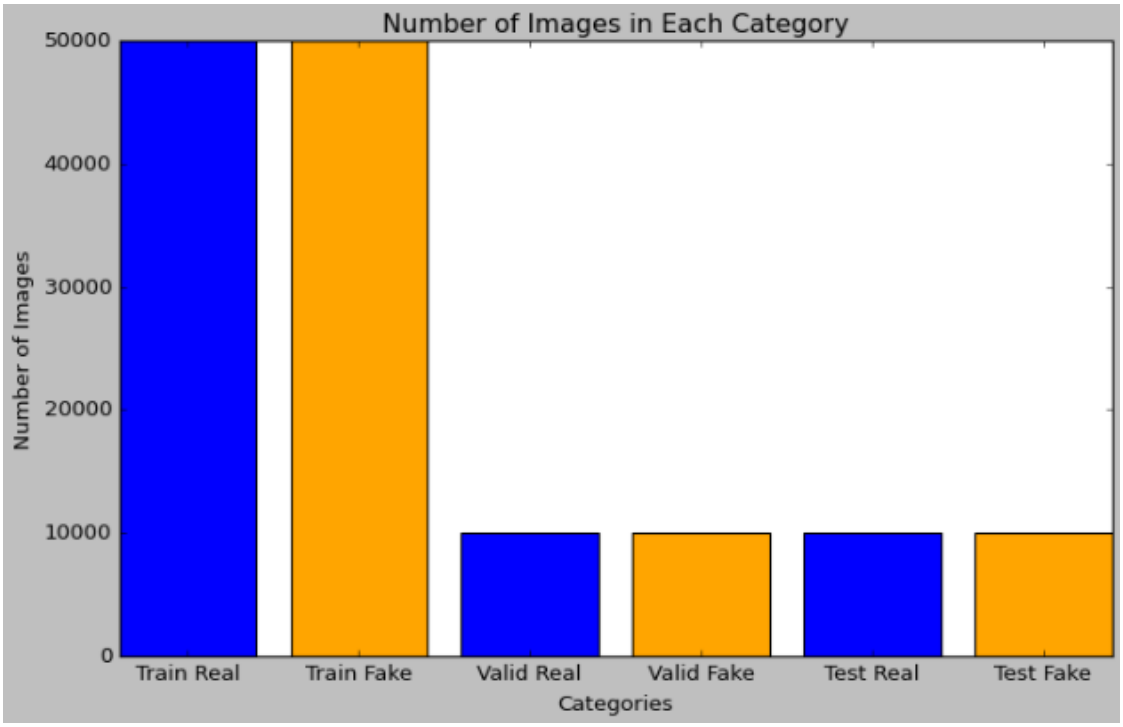

Рисунок 3.2 – Кількість даних у тренувальних, валідаційних та тестових наборах у датасеті

З діаграми видно, що датасет збалансований і включає рівну кількість реальних та штучних зображень для кожного набору (тренувальний, валідаційний, тестовий). Це забезпечує належну основу для тренування, перевірки та оцінки моделей нейронних мереж.

Аналіз даних:

— датасет містить збалансовану кількість реальних та штучних зображень, що є критичним для навчання моделі без зміщення;

— всі зображення були змінені до розміру 256x256 пікселів, що спрощує обробку даних та забезпечує однорідність вхідних даних;

— датасет розділений на тренувальний, валідаційний і тестовий набори, що дозволяє проводити навчання, налаштування

гіперпараметрів та остаточну оцінку моделі на окремих наборах даних.

Таким чином, наш датасет відповідає вимогам для тренування нейронних мереж, забезпечуючи надійний та збалансований набір даних для ідентифікації штучно згенерованих облич.

3.3 Опис програми та архітектури запропонованих нейронних мереж

У цьому розділі ми детально розглянемо програму та архітектури нейронних мереж, які використовувалися для ідентифікації штучно згенерованих облич. Усі моделі приймають на вхід попередньо оброблені дані розміром 256x256 пікселів і працюють з кольоровими зображеннями.

Запропоновано кілька моделей нейронних мереж, кожна з яких була протестована на відповідних наборах даних. Серед розглянутих моделей були раніше згадані моделі MobileNetV2, ResNet50 та VGG16, які демонструють високу ефективність у задачах класифікації зображень.

На рис.3.6 зображено процес роботи програми у вигляді блок-схеми, що відображає послідовність кроків обробки та класифікації зображень.

Для кожної моделі було обрано відповідні гіперпараметри та стратегії тренування, що дозволяє досягти оптимальних результатів. MobileNetV2 використовується завдяки своїй ефективності та меншій обчислювальній складності, що робить її ідеальною для мобільних застосувань. ResNet50 дозволяє зберегти точність за допомогою залишкових блоків, що значно покращує стабільність навчання. VGG16, завдяки своїй простій архітектурі, забезпечує високу точність і є одним з найпопулярніших підходів у задачах класифікації зображень.

Пізніше, запропоновані моделі будуть реалізовані та протестовані з використанням реальних і штучних зображень, що дозволяє оцінити їхню продуктивність у реальних умовах.

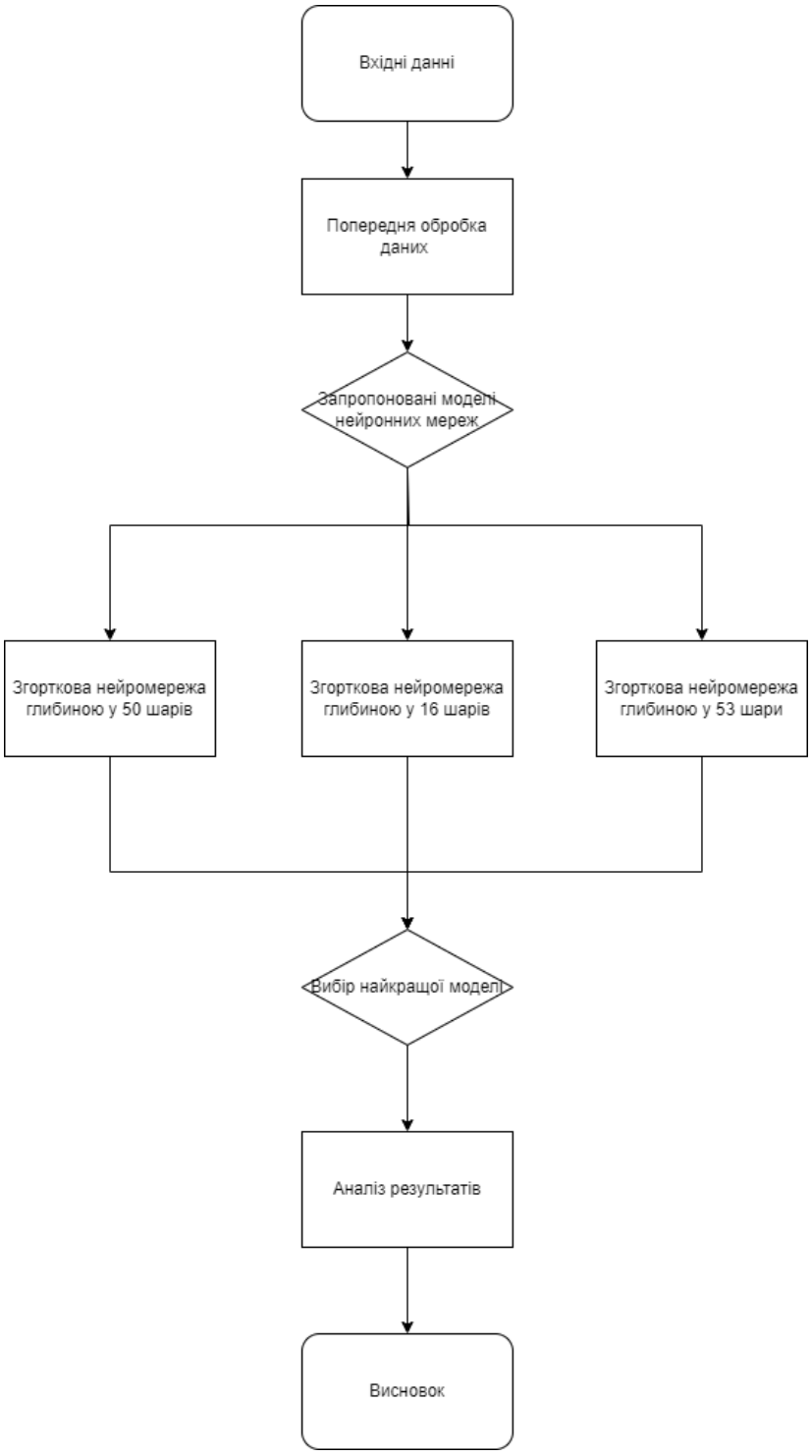

Рисунок 3.3 – Блок-схема роботи програми

Згорткова нейронна мережа, заснована на архітектурі MobileNetV2, складається з кількох основних компонентів і шарів. Архітектура включає такі елементи.

1. Мережа MobileNetV2 - складається з базової мережі, яка включає серію шарів, оптимізованих для обробки зображень. Вона використовує глибинні згортки (depthwise separable convolutions) для зменшення кількості параметрів без втрати якості представлення.
2. Глобальний шар середнього агрегування (Global Average Pooling) - зменшує розмірність вихідних даних з попередніх шарів, перетворюючи їх в одномірний вектор.
3. Щільний шар (Dense) - з 512 нейронів і активацією ReLU, який допомагає навчати більш складні представлення.
4. Шар нормалізації партій (Batch Normalization) - допомагає стабілізувати і прискорити навчання мережі.
5. Шар Dropout - з ймовірністю 0.3, який запобігає перенавчанню шляхом випадкового виключення нейронів під час навчання.
6. Ще один щільний шар (Dense) - з 128 нейронів і активацією ReLU для подальшої обробки даних.
7. Другий шар Dropout - з ймовірністю 0.1, для додаткової регуляризації.
8. Фінальний щільний шар (Dense) - з 2 нейронами і активацією SoftMax, який повертає прогнозовану мітку класу.

Таким чином, архітектура нейронної мережі MobileNetV2 включає основну базову мережу, шари глобального середнього агрегування, щільні шари з функцією активації ReLU, шари нормалізації партій та Dropout, а також фінальний шар з активацією SoftMax для класифікації (рис.3.4).

Model: "sequential"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 512)	655,872
batch_normalization (BatchNormalization)	(None, 512)	2,048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 128)	65,664
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 2)	258

Рисунок 3.4 – Архітектура нейронної мережі на базі MobileNetV2

На рис.3.5 зображено графік втрат для тренувальної та валідаційної вибірок моделі MobileNetV2 в процесі навчання за 20 епох. Горизонтальна вісь представляє кількість епох, тобто кількість разів, коли модель пройшла через весь тренувальний набір даних. Вертикальна вісь представляє значення втрат, що є метрикою, яка показує, наскільки добре модель передбачає правильні результати. Нижчі значення втрат вказують на кращу продуктивність моделі. Синя лінія показує зміну втрат на тренувальному наборі даних протягом епох. Вона зменшується з кожною епохою, що свідчить про те, що модель покращує свої передбачення на тренувальних даних. Помаранчева лінія показує зміну втрат на валідаційному наборі даних протягом епох. Вона також зменшується, що свідчить про покращення узагальнюючої здатності моделі. Графіки втрат для тренувальної і валідаційної вибірок збігаються, що свідчить про гарну узагальнюючу здатність моделі. Модель добре навчається і узагальнює на невідомих даних. Втрата і точність на валідаційному наборі не відрізняються значно від показників на тренувальному наборі, що є позитивним знаком.

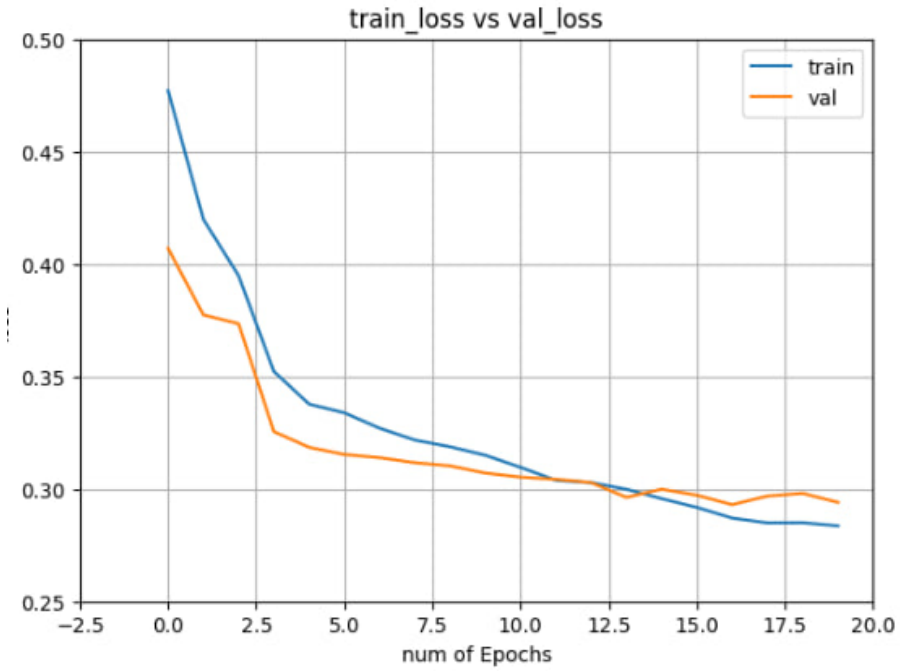

Рисунок 3.5 – Графік втрат для тренувальної та валідаційної вибірок на базі мережі MobileNetV2

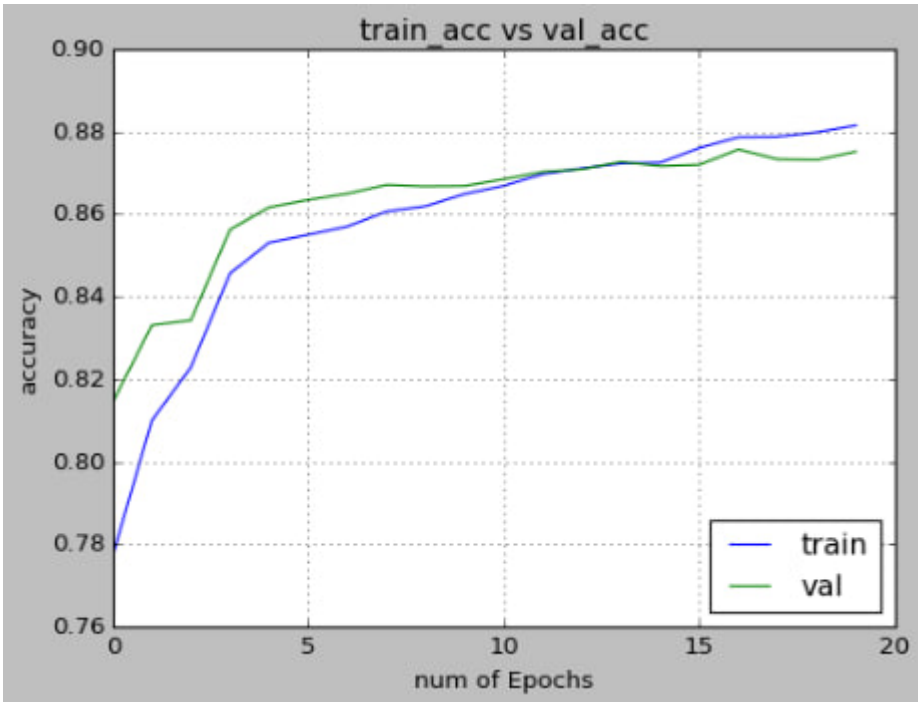

Рисунок 3.6 – Зміни точності для тренувальної та валідаційної вибірок на базі мережі MobileNetV2

На рис.3.6 зображено зміну точності для тренувальної та валідаційної вибірок моделі MobileNetV2 в процесі навчання за 20 епох. Горизонтальна вісь є аналогічною до графіку на рис.3.5, але вертикальна вісь представляє значення точності, що є метрикою, яка показує, наскільки добре модель передбачає правильні результати. Вищі значення точності вказують на кращу продуктивність моделі. Синя лінія показує зміну точності на тренувальному наборі даних протягом епох. Точність стабільно зростає, що свідчить про те, що модель навчається і покращує свої передбачення на тренувальних даних. Зелена лінія показує зміну точності на валідаційному наборі даних протягом епох. Точність на тренувальному наборі даних стабільно зростає, досягаючи приблизно 88% до кінця навчання. Ці результати підтверджують, що модель добре навчається і здатна узагальнювати на невідомих даних, що є важливим критерієм для оцінки якості моделі.

Таблиця 3.1 – Порівняльна таблиця метрик продуктивності моделі на базі мережі MobileNetV2 для навчальних та валідаційних даних

Метрика	Значення на навчальних даних	Значення на валідаційних даних
Accuracy	0.8809	0.8752
Loss	0.2844	0.2940
Precision	0.8800	0.8752
AUC	0.90340	0.8949

Глибока нейронна мережа, заснована на архітектурі VGG16, складається з кількох основних компонентів і шарів. Архітектура включає такі елементи:

1. Мережа VGG16 - складається з базової мережі, яка включає серію згорткових (convolutional) і пулінгових (pooling) шарів, оптимізованих для обробки зображень. Ці шари не використовуються для подальшого навчання і заморожені.
2. Глобальний шар середнього агрегування.
3. Щільний шар з 512 нейронів і активацією ReLU.

4. Шар нормалізації партій.
5. Шар Dropout з ймовірністю 0.3.
6. Ще один щільний шар з 128 нейронів і активацією ReLU.
7. Другий шар Dropout з ймовірністю 0.1, для додаткової регуляризації.
8. Фінальний щільний шар з 2 нейронами і активацією SoftMax, який повертає прогнозовану мітку класу.

Таким чином, архітектура нейронної мережі VGG16 включає основну базову мережу, шари глобального середнього агрегування, щільні шари з функцією активації ReLU, шари нормалізації партій та Dropout, а також фінальний шар з активацією SoftMax для класифікації (рис.3.7).

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14,714,688
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 512)	262,656
batch_normalization (BatchNormalization)	(None, 512)	2,048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 128)	65,664
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 2)	258

Рисунок 3.7 – Архітектура нейронної мережі на базі VGG16

На рис.3.8 зображений графік, який відображає зміну втрат моделі на базі мережі VGG16. Синя лінія залишається на одному рівні і не показує великі зміни, що вказує на стабільне навчання на навчальних даних. Помаранчева лінія має

сильні Помаранчева лінія має сильні коливання і великі піки, що говорить про нестабільність на валідаційних даних. Це може свідчити про проблему з даними, перенавченням або про недостатню регуляризацію даних, що потрібно буде врахувати при оптимізації моделі.

На рис.3.9 зображений графік, який відображає зміну точності моделі на мережі VGG16. Синя лінія показує стабільне зростання точності на навчальних даних, досягаючи близько 85%. Помаранчева лінія також показує зростання точності на валідаційних даних, з деякими коливаннями, але в цілому слідує аналогічній тенденції, що й точність на навчальних даних.

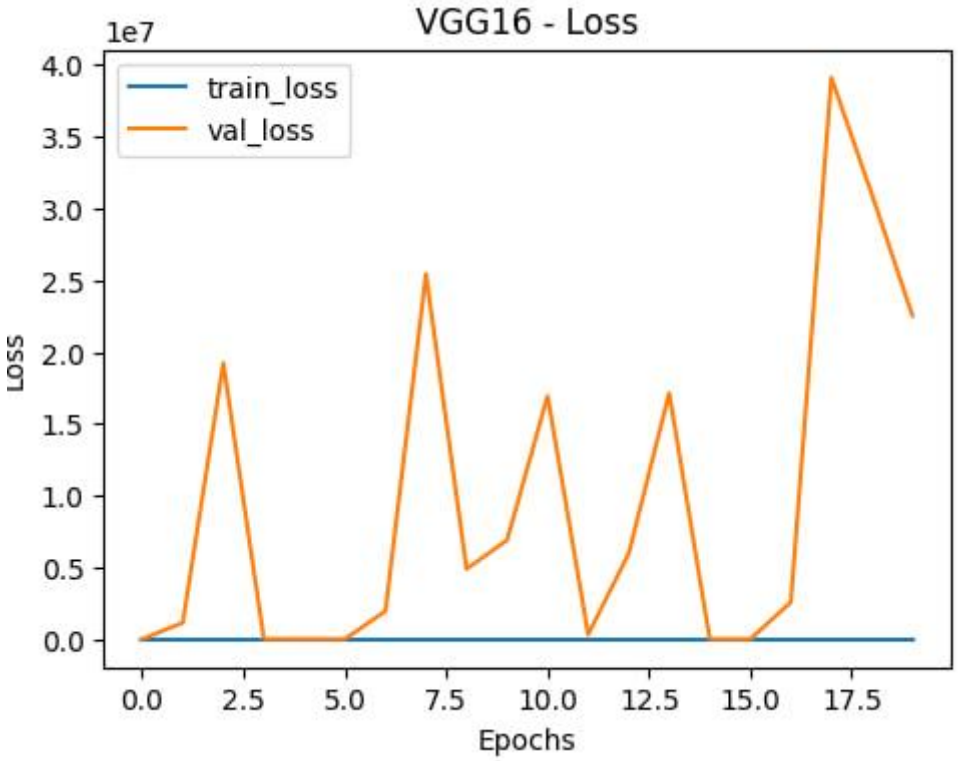

Рисунок 3.8 – Графік втрат для тренувальної та валідаційної вибірок на базі мережі VGG16

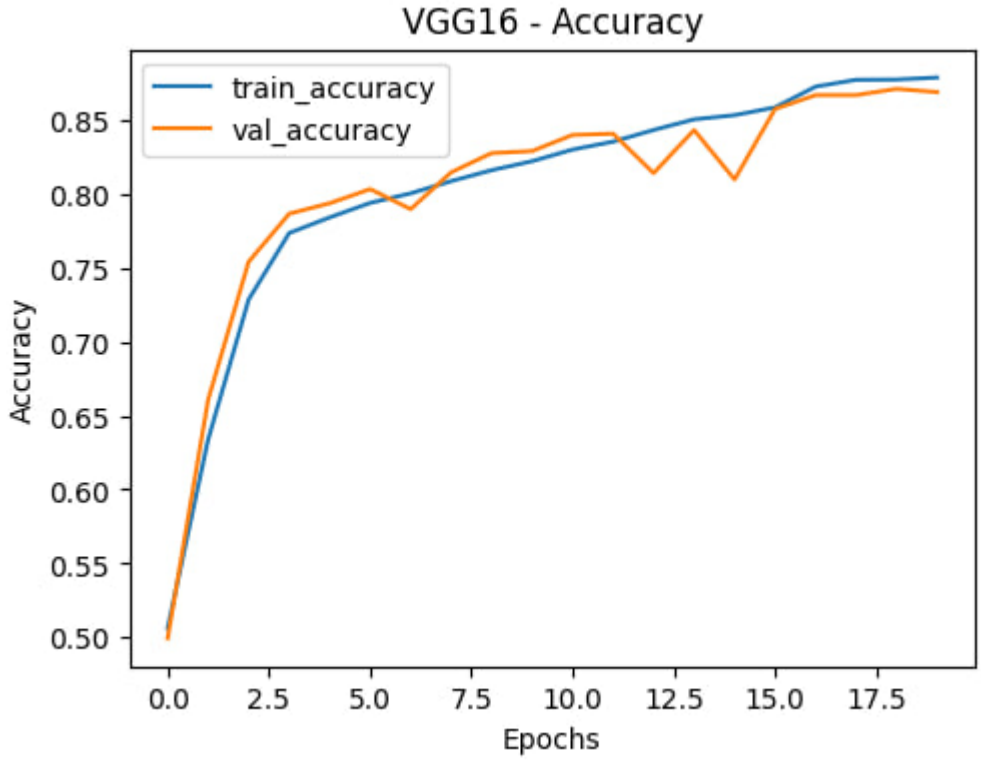

Рисунок 3.9 – Зміни точності для тренувальної та валідаційної вибірок на базі мережі VGG16

Таблиця 3.2 – Порівняльна таблиця метрик продуктивності моделі на базі мережі VGG16 для навчальних та валідаційних даних

Метрика	Значення на навчальних даних	Значення на валідаційних даних
Accuracy	0.8778	0.8687
Loss	0.2913	22521374.0000

Розглядати інші метрики для цієї моделі не є доречним, оскільки вони будуть близькими до випадкового значення, тобто половину одиниці через високі трати.

Глибока нейронна мережа, заснована на архітектурі ResNet50, складається з кількох основних компонентів і шарів. Архітектура включає такі елементи.

1. Мережа ResNet50 - основний компонент, що включає серію згорткових і пулінгових шарів, оптимізованих для обробки зображень. Ці шари попередньо навчені на наборі даних ImageNet і

використовуються без змін.

2. Глобальний шар середнього агрегування.
3. Щільний шар з 512 нейронів і активацією ReLU.
4. Шар нормалізації партій.
5. Фінальний щільний шар з 2 нейронами і активацією SoftMax.

Таким чином, архітектура нейронної мережі ResNet50 включає основну базову мережу, шари глобального середнього агрегування, щільний шар з функцією активації ReLU, шар нормалізації партій, а також фінальний шар з активацією SoftMax для класифікації (рис.3.10).

Model: "sequential_1"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 4, 4, 2048)	23,587,712
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 2048)	0
dense_2 (Dense)	(None, 512)	1,049,088
batch_normalization_1 (BatchNormalization)	(None, 512)	2,048
dense_3 (Dense)	(None, 2)	1,026

Рисунок 3.10 – Архітектура нейронної мережі на базі ResNet50

На рис.3.11 зображений графік, який відображає зміну втрат моделі на базі мережі ResNet50. Синя лінія показує стабільне зниження значення функції втрат на навчальних даних, модель добре навчається і поступово зменшує свою помилку на навчальних даних. Помаранчева лінія має коливання, але загальна тенденція теж до зниження. Коливання можуть свідчити про нестабільність на валідаційних даних або про те, що модель перенавчається на певних епохах.

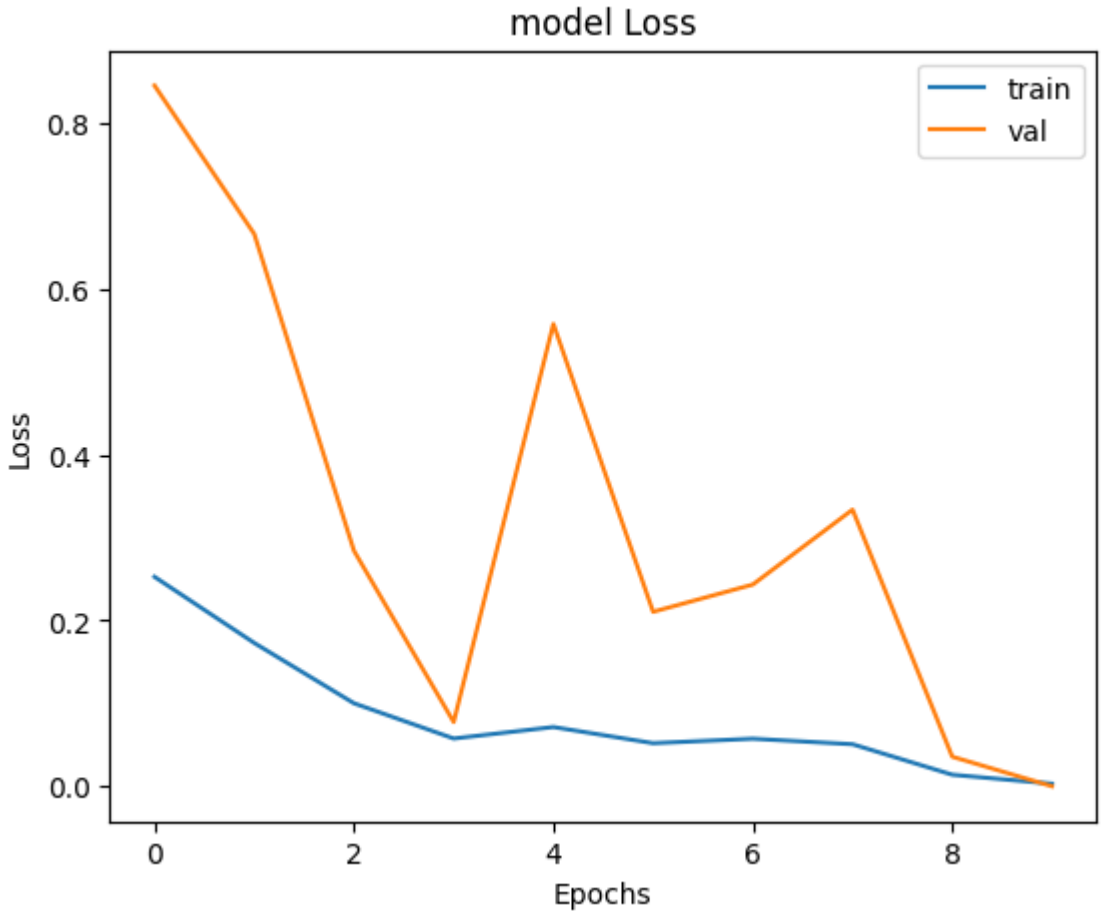

Рисунок 3.11 – Графік втрат для тренувальної та валідаційної вибірок на базі мережі ResNet50

На рис.3.12 зображений графік, який відображає зміну точності моделі на мережі ResNet50 Синя лінія показує стабільне зростання точності на навчальних даних, досягаючи майже 100%. Це вказує на те, що модель добре навчається на навчальних даних. Помаранчева лінія також показує зростання точності на валідаційних даних, але з деякими коливаннями. Загальна тенденція вказує на покращення точності, але коливання можуть бути свідченням того, що модель час від часу переобучається або має проблеми з узагальненням на нових даних, що потрібно врахувати при оптимізації моделі. Загальна тенденція вказує на покращення.

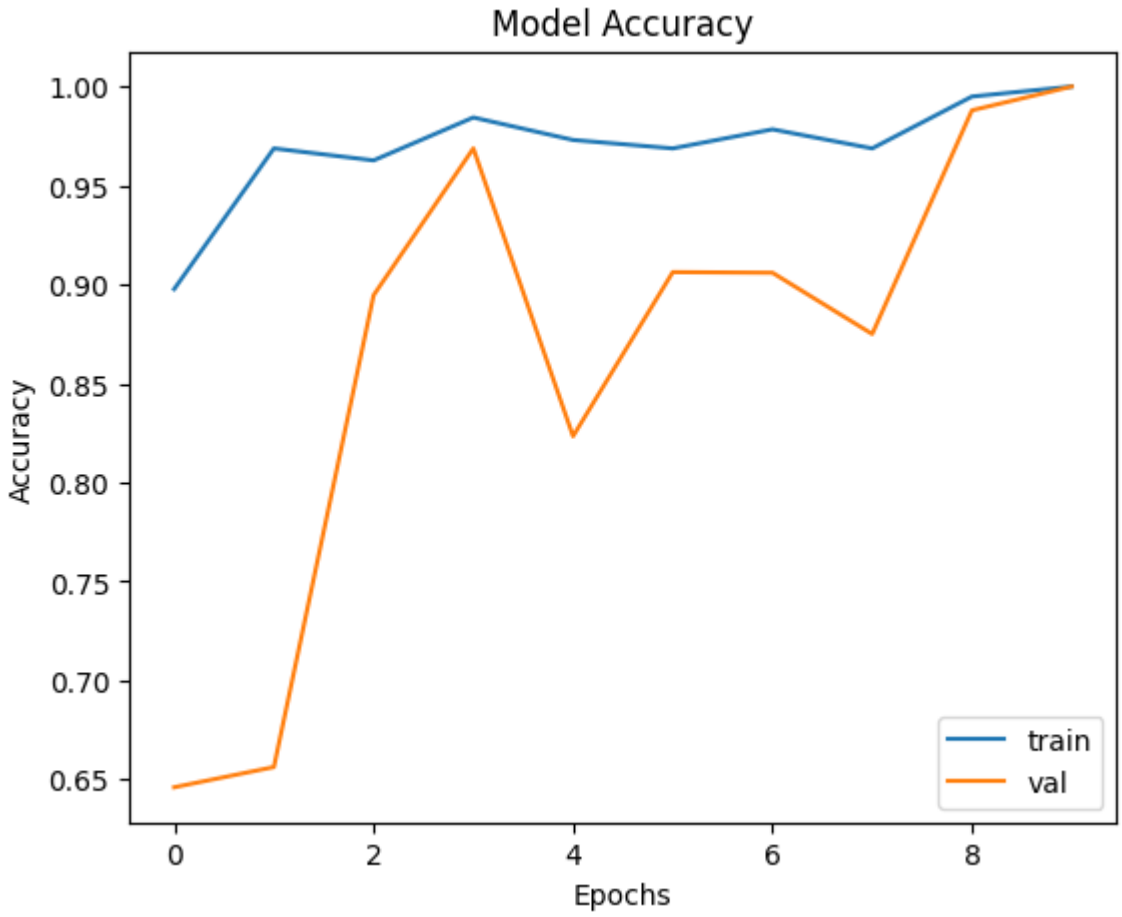

Рисунок 3.12 – Зміни точності для тренувальної та валідаційної вибірок на базі мережі VGG16

Таблиця 3.3 – Порівняльна таблиця метрик продуктивності моделі на базі мережі ResNet50 для навчальних та валідаційних даних

Метрика	Значення на навчальних даних	Значення на валідаційних даних
Accuracy	1.0000	1.0000
Loss	0.0034	2.8323e-04

Розглядати інші метрики для цієї моделі не є доречним, оскільки вони будуть близькими до максимального значення, тобто одинці.

3.4 Демонстрація роботи нейронної мережі для ідентифікації штучно згенерованих облич

У цьому підрозділі буде детально розглянуто процес роботи розробленої нейронної мережі для ідентифікації штучно згенерованих зображень облич. Демонстрація охоплює всі етапи роботи системи: від підготовки вхідних даних до отримання кінцевих результатів класифікації.

Рисунок 3.12 – Приклад реального зображення з датасету для тесту

Модель на основі ResNet50 та MobileNetV2 ідентифікували рис.3.12 як реальне, натомість модель VGG16, зробило висновки, що воно не справжнє.

Рисунок 3.13 – Приклад згенерованого зображення з датасету для тесту

Аналогічна ситуація для рис.3.13, моделі на основі ResNet50 та MobileNetV2 ідентифікували зображення як згенероване, але модель VGG16, зробило висновки, що воно справжнє.

Рисунок 3.14 – Приклад очевидно згенерованого зображення з датасету для тесту

Для легкого рис.3.14, де людське око самостійно може розпізнати не справжнє, усі 3 моделі підтвердили що зображення є штучно згенерованим.

Рисунок 3.15 – Реалістичне згенероване зображення з датасету для тесту

Але є винятки, оскільки для реалістно створенного рис.3.15 усі 3 моделі також підтвердили, що зображення не є справжнім.

3.5 Висновки до розділу

У цьому розділі проведено огляд датасетів, продемонстровано попередню обробку даних і представлено архітектуру запропонованих моделей. Для визначення найкращої моделі, зберемо метрики для всіх моделей в таблицю 3.6.

Таблиця 3.4 – Порівняння моделей НМ на валідаційних данних

Базова НМ	Accuracy	AUC	Precision	Loss
MobileNetV2	0.8752	0.8950	0.8750	0.2940
VGG16	0.8687	≈0.5000	≈0.5000	22521374.0000
ResNet50	1.0000	≈1.0000	≈1.0000	$2.8323e^{-04}$

Бачимо, що найкращою за всіма показниками є згорткова НМ №3, але враховуючи її можливу не точність на валідаційних данних, НМ №1 також слід взяти до уваги.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі буде проведено оцінювання ключових характеристик майбутнього програмного продукту, що спеціалізується на дослідженні демографічного стану. Ця реалізація сприятиме проведенню всіх необхідних досліджень, дозволяючи якісно досліджувати питання не тільки в Україні, але й по всьому світу.

Також у цьому дослідженні представлені різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що впливає на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього використовується метод функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) є технологією, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення можливостей зниження витрат за рахунок більш ефективних варіантів виробництва, а також досягнення кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає визначення послідовності етапів розробки продукту, розрахунок річних витрат та кількості робочих годин, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

Впровадження ФВА дозволяє не лише оптимізувати витрати, але й підвищити загальну якість та конкурентоспроможність програмного продукту на ринку. Це досягається за рахунок детального аналізу всіх етапів розробки та впровадження найбільш ефективних методів виробництва та управління ресурсами.

4.1 Постановка задачі проектування

У роботі використовується метод функціонально-вартісного аналізу для проведення техніко-економічного аналізу розробки системи розпізнавання рентгенівських зображень грудної клітини. Оскільки рішення щодо проектування та реалізації компонентів впливають на всю систему, кожна окрема підсистема повинна відповідати вимогам загальної системи. Тому фактичний аналіз являє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є такими:

— функціонування на персональних комп’ютерах зі стандартним набором компонентів;
— зручність та зрозумілість для користувача;
— висока швидкість обробки даних та доступ до інформації в реальному часі;
— можливість легкого масштабування та обслуговування;
— мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція $F0$ полягає у розробці програмного продукту, який дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Виходячи з цієї функції, можна виділити наступні підфункції:

— $F1$: Вибір мови програмування;
— $F2$: Вибір способу реалізації алгоритмів;

— *F*3: Вибір середовища розробки.

Кожна з цих функцій має декілька варіантів реалізації:

— Для F1:

а) C++;

б) Python.

— Для F2:

а) Розробка власних алгоритмів;

б) Комбінація готових бібліотек та власних алгоритмів.

— Для F3:

а) Visual Studio Code/PyCharm;

б) Jupyter notebook.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис.4.1). Ця карта відображає можливі варіанти реалізації кожної з підфункцій, що забезпечують гнучкість та адаптивність процесу розробки програмного продукту. Морфологічна карта є важливим інструментом для структурованого аналізу та вибору оптимальних рішень у проектуванні.

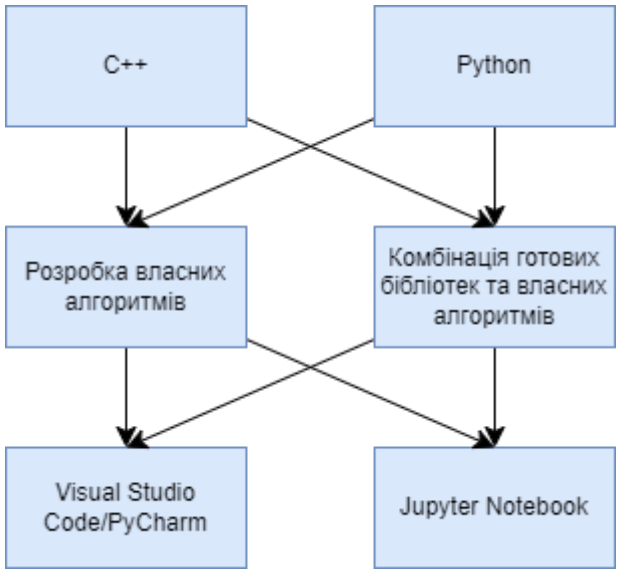

Рисунок 4.1 – Морфологічна карта

Позитивно-негативна матриця показана в таблиці (4.1).

Таблиця 4.1 – Позитивно-негативна таблиця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	А	Швидке виконання програми	Висока складність, потребує більше часу на написання коду
	Б	Легкість у використанні, наявність багатьох готових бібліотек	Низька швидкодія
F_2	А	Точна реалізація необхідного функціоналу	Потребує більше часу на реалізацію, можливість помилок
	Б	Простіше написання коду, легкість у використанні	Обмежена гнучкість
F_3	А	Підтримка великої кількості розширень; Потужні інструменти для розробки	Менше професійних інструментів для відлагодження коду; потребує більше ресурсів системи
	Б	Інтерактивне середовище, зручність для обробки даних та візуалізації	Обмежена підтримка інтеграції з великими проектами, не підходить для розробки великих програм

На основі аналізу позитивно-негативної матриці можна зробити висновок, що при розробці програмного продукту слід відмовитися від деяких варіантів реалізації функцій, оскільки вони не задовольняють вимоги, поставлені перед програмним продуктом.

Функція F_1.

Пріоритет надається загальнодоступності. Щоб полегшити процес написання коду, варіант А слід виключити.

Функція F_2.

Готові бібліотеки для мови програмування Python забезпечують зручність і оптимізацію, тому варіант А слід виключити.

Функція F_3.

Обидва варіанти підходять для розробки. Тому будемо розглядати такі варіанти реалізації програмного продукту:

$F_1\text{б} - F_2\text{б} - F_3\text{б}$

$F_1\text{б} - F_2\text{б} - F_3\text{а}$

Для оцінки якості розглянутих функцій використовується система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня. Для характеристики програмного продукту будемо використовувати такі параметри:

— X1 – швидкодія мови програмування;

— X2 – об’єм пам’яті для обчислень та збереження даних;

— X3 – час навчання даних;

— X4 – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів обираються на основі вимог замовника та умов експлуатації програмного продукту, як показано в таблиці (4.2).

Таблиця 4.2 – Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	120	80	65

Закінчення таблиці 4.2 – Основні параметри програмного продукту

Об’єм пам’яті	X2	Мб	200	150	100
Час попередньої обробки даних	X3	мс	1000	600	200
Потенційний об’єм програмного коду	X4	кількість рядків коду	1000	650	500

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис.4.2 – рис.4.5.

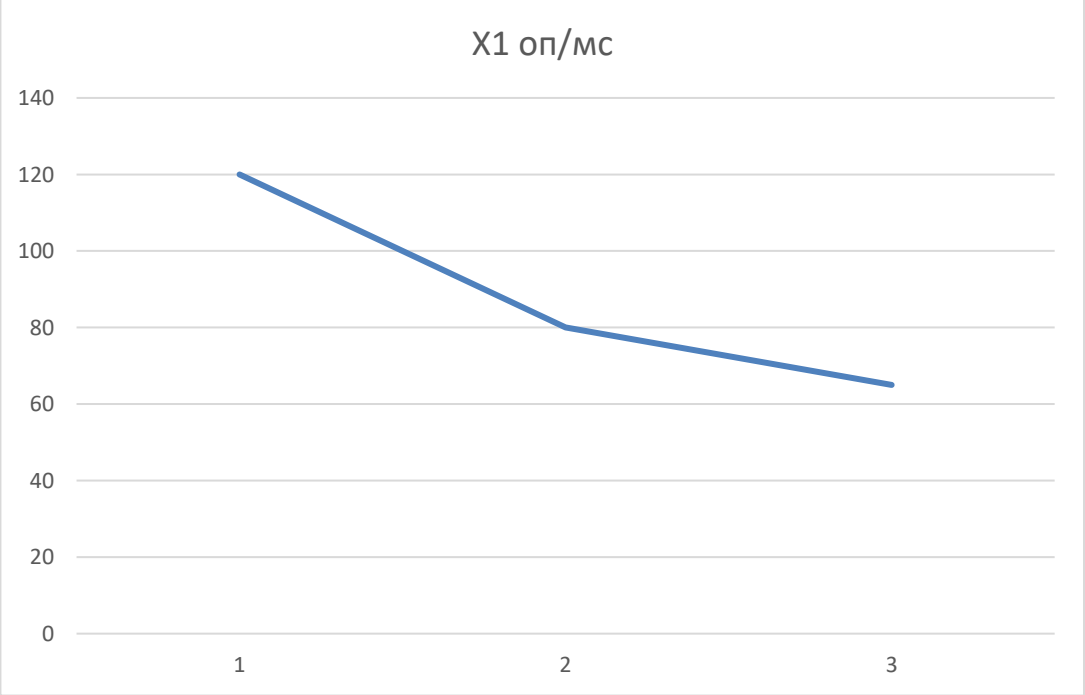

Рисунок 4.2 – X1, швидкодія мови програмування

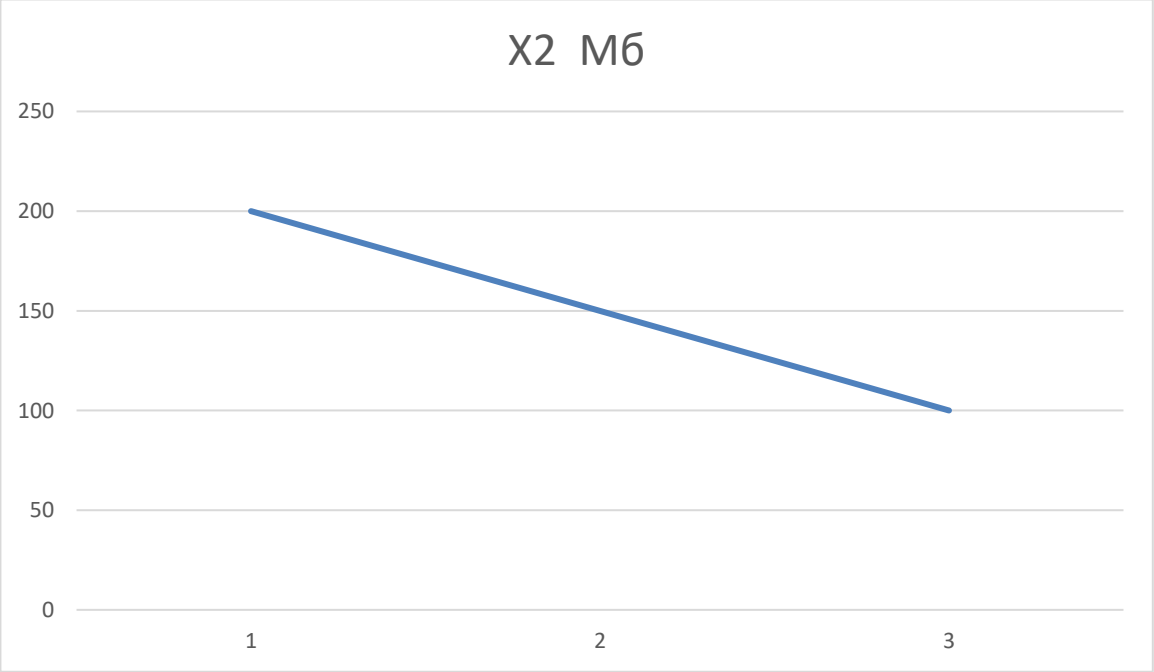

Рисунок 4.3 – X2, об’єм пам’яті

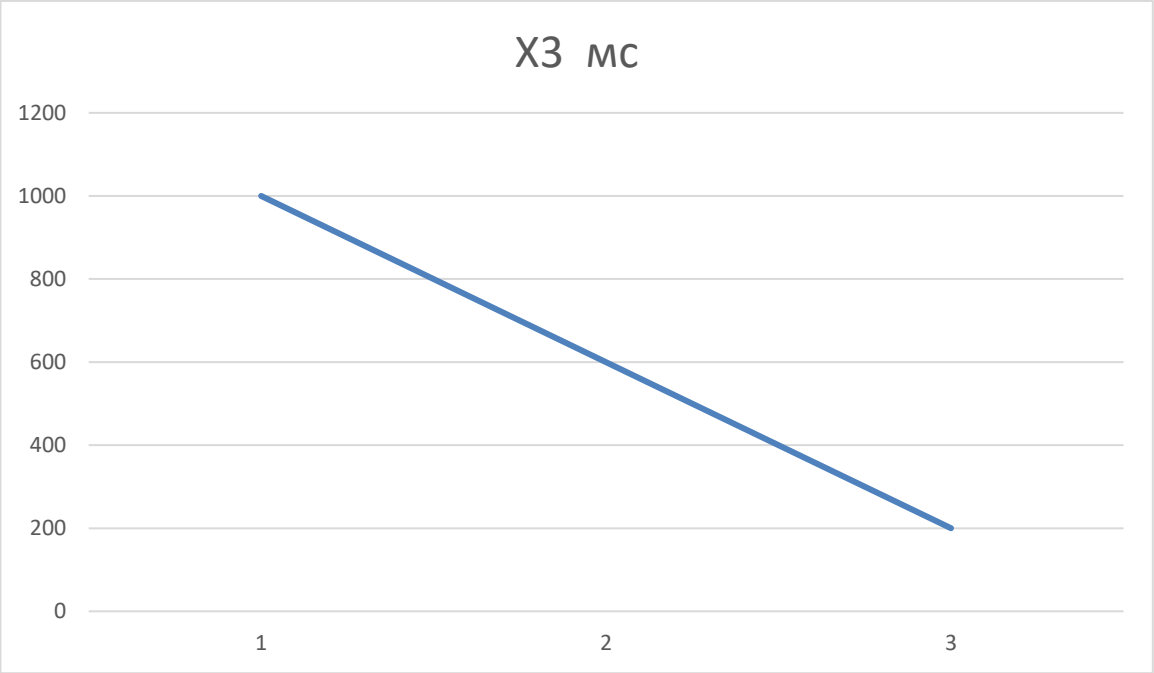

Рисунок 4.4 – X3, час попередньої обробки даних

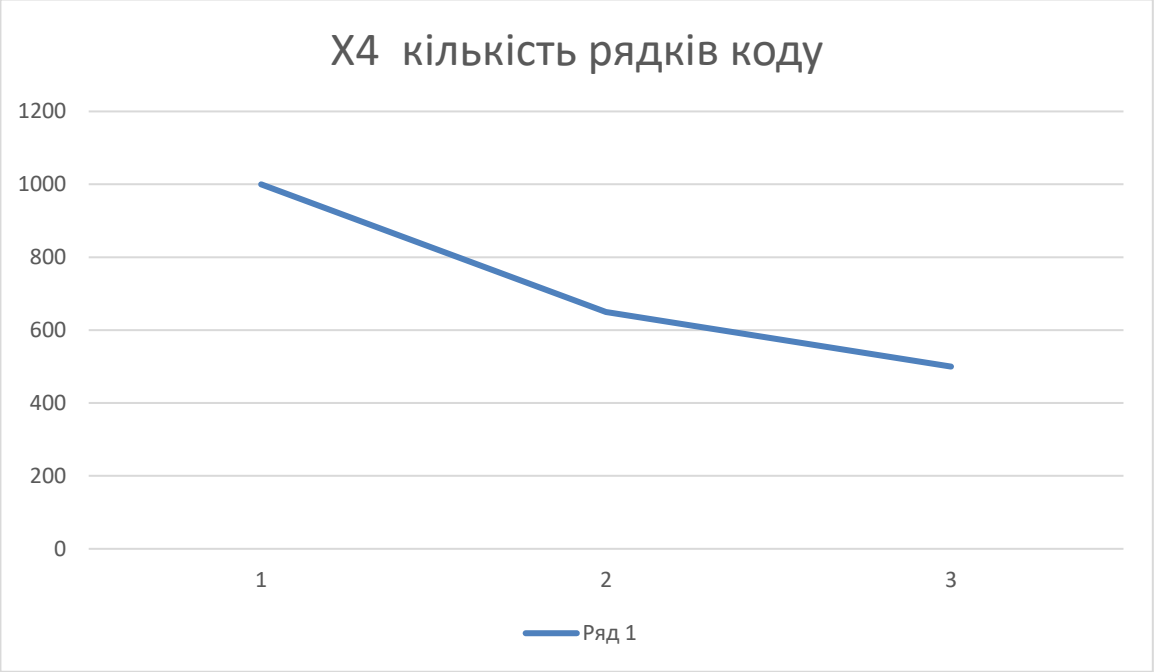

Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення та аналізу кожний експерт оцінює ступінь важливості кожного параметру для досягнення конкретної мети – розробки програмного продукту, який забезпечує найточніші результати при визначенні параметрів моделей адаптивного прогнозування та обчисленні прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 осіб. Процес визначення коефіцієнтів значимості включає:

— визначення рівня значимості параметра шляхом присвоєння різних рангів;

— перевірку придатності експертних оцінок для подальшого

використання;

— визначення оцінки попарного пріоритету параметрів;

— обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені в таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	3	4	3	5	5	5	3	28	4,25	18,0625
X2	Об’єм пам’яті	Mb	5	4	5	5	4	4	5	32	8,25	68,0625
X3	Час попередньої обробки даних	мс	3	5	3	4	2	5	4	25	1.25	76,5625
X4	Потенційний об’єм програмного коду	Кількість рядків коду	2	3	1	3	3	2	1	15	-8,75	76,5625
	Разом		10	10	10	10	10	10	10	95	0	164,25

Для перевірки ступеня достовірності експертних оцінок, визначимо такі параметри:

— сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 95, \quad (4.1)$$

де N – число експертів;

n – число параметрів.

— середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 23{,}75 \tag{4.2}$$

— відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \tag{4.3}$$

Сума відхилень по всім параметрам повинна дорівнювати 0.

— загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 164.25$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 164.25}{7^2(4^3-4)} = 0{,}671 > W_k = 0{,}67 \tag{4.4}$$

Ранжування можна вважати достовірним, оскільки знайдений коефіцієнт узгодженості перевищує нормативний, що дорівнює 0.67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо в таблицю нижче.

Таблиця 4.4 – Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 та X2	<	=	<	=	>	>	<	<	0.5
X1 та X3	=	<	=	>	>	>	<	>	1.5
X1 та X4	>	>	>	>	>	>	>	>	1.5
X2 та X3	>	<	>	>	>	<	>	>	1.5
X2 та X4	>	>	>	>	>	>	>	>	1.5
X3 та X4	>	>	>	>	<	>	>	>	1.5

Числове значення, що визначає ступінь переваги і-го параметра над j-им параметром, можна знайти за такою формулою:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{4.9}$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \tag{4.10}$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця – 4.5 - Розрахунок вагомості параметрів

					Ітерація 1		Ітерація 2		Ітерація 3	
	X1	X2	X3	X4	b_i	$К_{вi}$	b_i^1	$К_{вi}^1$	b_i^2	$К_{вi}^2$
X1	1.0	0.5	1.5	1.5	4.5	0.28	16.25	0.28	59.125	0.27
X2	1.5	1.0	1.5	1.5	5.5	0.34	21.25	0.36	77.875	0.36
X3	0.5	0.5	1.0	1.5	3.5	0.22	12.25	0.2	44.875	0.21
X4	0.5	0.5	0.5	1.0	2.5	0.16	9.25	0.16	34.125	0.16
Всього					16	1	59	1	216	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об’єм пам’яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4.11)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця – 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Пара метри	Абсолют-не значення	Баналь на оцінка	Коефіц ієнт вагомо сті	Коефіцієнт рівня якості
F1	B	X1	14000	4	0.27	1.08
F2	A	X2	22	5	0.36	1.8
F2	B	X3	90	3	0.21	0.63
F3	A	X4	2000	4	0.16	0.64

Визначимо рівень якості кожного параметра завдяки формулі:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (4.12)$$

Маємо:

$$K_{K1} = 1{,}08 + 1{,}8 + 0{,}64 = 3{,}52.$$

$$K_{K2} = 1{,}08 + 0{,}63 + 0{,}64 = 2{,}35.$$

Отже, найкращим є перший варіант, якому відповідає найбільше значення технічного рівня.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання.

1. Розробка проекту програмного продукту.
2. Розробка програмної оболонки.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_О = T_Р \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.13)$$

де $Т_р$ – трудомісткість розробки ПП;

$К_п$ – поправочний коефіцієнт;

$К_{ск}$ – коефіцієнт на складність вхідної інформації;

$К_м$ – коефіцієнт рівня мови програмування;

$К_{ст}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{ст.м}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_р = 80$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_п = 1.9$.

Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1:$К_{ск} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{ст} = 0.8$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 80 \cdot 1.9 \cdot 0.8 \text{ людино-днів}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, ступінь новизни Б), тобто людино-днів

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (121.6 + 22.4 + 4.8 + 22.4) \cdot 8 = 1369.6 \text{ людино-годин}$$
$$Т_{II} = (121.6 + 22.4 + 6.91 + 22.4) \cdot 8 = 1383.48 \text{ людино-годин}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 15000 грн., один аналітик в області даних з окладом 20000. Визначимо середню зарплату за годину за формулою:

$$C_ч = \frac{M}{T_m \cdot t} \text{грн}, \quad (4.14)$$

де $С_ч$ – місячний оклад працівників;

$Т_i$ – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$C_ч = \frac{(2 \cdot 15000 + 20000)}{3 \cdot 21 \cdot 8} = 99.2 \text{грн} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{ЗП} = C_{ч} \cdot T_i \cdot K_{Д}, \quad (4.16)$$

де $C_{ч}$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$K_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$I. C_{ЗП} = 92{,}2 \cdot 1369{,}6 \cdot 1{,}2 = 163037 \text{грн},$$

$$II. C_{ЗП} = 99{,}2 \cdot 1383{,}48 \cdot 1{,}2 = 164689 \text{ грн}.$$

Відрахування на єдиний соціальний внесок становить :

$$I. \quad C_{ВІД} = C_{ЗП} \cdot 0{,}22 = 163037 \cdot 0{,}22 = 35868{,}14 \text{ грн},$$

$$II. \quad C_{ВІД} = C_{ЗП} \cdot 0{,}22 = 164689 \cdot 0{,}22 = 36231{,}58 \text{ грн}.$$

Тепер визначимо витрати на оплату однієї машино-години. ($C_М$)

Оскільки одна ЕОМ обслуговує одного програміста з окладом 20000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_Г = 12 \cdot М \cdot К_З = 12 \cdot 20000 \cdot 0{,}2 = 48000 \text{ грн}.$$

З урахуванням додаткової заробітної плати:

$$C_{ЗП} = C_Г \cdot (1 + К_З) = 48000 \cdot (1 + 0{,}2) = 57600 \text{ грн}.$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0,22 = 57600 \cdot 0,22 = 12672 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 24000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1,1 \cdot 0,25 \cdot 24000 = 6600 \text{ грн,}$$

де $К_{тм}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$ – річна норма амортизації;

$Ц_{пр}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1,1 \cdot 24000 \cdot 0,06 = 1584 \text{ грн,}$$

де $К_р$ – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПП за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) \cdot t_З \cdot К_В = (365 - 104 - 12 - 16) \cdot 8 \cdot 0,8 = 1491,2 \text{ години,}$$

де $Д_к$ – календарна кількість днів у році;

$Д_в$, $Д_с$ – відповідно кількість вихідних та святкових днів;

$Д_р$ – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

$К_В$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$\text{Сел} = \text{Теф} * Nc * Kз * \text{Цен} = \ 1491{,}2 \ * 0{,}2 * 0{,}7 * 5{,}23 = 1091.8$$

де N_c – середньо-споживана потужність приладу;

$K_з$ – коефіцієнтом зайнятості приладу;

$Ц_{ен}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_Н = Ц_{ПР} \cdot 0{,}67 = 24000 \cdot 0{,}67 = 16080 \text{ грн.}$$

Тоді річні експлуатаційні витрати будуть:

$$\text{Секс} = \text{Сзп} + \text{Свід} + \text{Са} + \text{Ср} + \text{Сел} + \text{Сн}, \qquad (4.17)$$

$$\text{Секс} = 57600 + 12672 + 6600 + 1584 + 1091.8 + 16080 = 95627.8$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$\text{Смг} = \frac{\text{Секс}}{\text{Теф}} = \frac{95627.8}{1091.8} = 64.1 \text{ грн/год}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$\text{См} = \text{Смг} * \text{Т} \qquad (4.18)$$

$$\text{I. Сн} = 64.1 * 1369.6 = 87791.36$$

$$\text{II. См} = 64.1 * 1383.48 = 88681.06$$

Накладні витрати складають 67% від заробітної плати:

$$C_{\text{Н}} = C_{\text{ЗП}} \cdot 0{,}67, \quad (4.19)$$

I. $C_{\text{Н}}$ = 163037 · 0,67 = 109234,79 грн,

II. $C_{\text{Н}}$ = 164689· 0,67 = 110341,36 грн.

Отже, вартість розробки ПП за варіантами становить:

$$\text{С}_{\text{ПП}} = \text{С}_{\text{ЗП}} + \text{С}_{\text{ВІД}} + \text{С}_{\text{М}} + \text{С}_{\text{Н}}, \quad (4.20)$$

I. Спп = 163037 + 35868.14 + 87791.36 + 109234.79 = 395931.29 грн

II. Спп=164689+36231.58+88681.06+110341.36=399943 грн

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$\text{К}_{\text{ТЕР}j} = \frac{Kkj}{C\text{ф}j} \quad (4.21)$$

$$K_{\text{ТЕР1}} = \frac{Kkj}{C\text{ф}j} = \frac{3.52}{395931.29} = 8.8 * 10^{-6}$$

$$K_{\text{ТЕР2}} = \frac{Kkj}{C\text{ф}j} = \frac{2.35}{399943} = 5.8 * 10^{-6}$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $\text{К}_{\text{ТЕР1}} = 8.8 * 10^{-6}$

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $\text{К}_{\text{ТЕР}} = 8.8 * 10^{-6}$.

Цей варіант реалізації програмного продукту має такі параметри:

Вибір мови програмування – Python;

Вибір моделі для навчання нейронної мережі – MobileNetV2, VGG16, ResNet50;

Вибір середовища розробки – Jupyter Nootebook.

Висновки до розділу 4

У цій частині було здійснено повний функціонально-вартісний аналіз програмного продукту та проведено оцінку його основних функцій.

У результаті виконання функціонально-вартісного аналізу розроблюваного програмного комплексу були визначені та оцінені основні функції програмного продукту, а також знайдено параметри, які його характеризують. На основі цього аналізу було обрано оптимальний варіант реалізації програмного продукту.

ВИСНОВКИ

У ході цієї роботи було проведено комплексний аналіз та розробку системи для ідентифікації штучно згенерованих облич. На основі досліджень сучасних підходів і методів глибинного навчання було обґрунтовано використання згорткових нейронних мереж (CNN), зокрема архітектур MobileNetV2, ResNet50 та VGG16. Ці моделі забезпечують високу ефективність у задачах класифікації зображень завдяки своїй здатності виділяти складні просторові ознаки.

Для реалізації системи було обрано фреймворки TensorFlow та Keras, які надають потужні інструменти для налаштування, тренування та оцінки моделей нейронних мереж. Ці програмні засоби мають велику підтримку спільноти та добре задокументовані, що значно полегшує процес розробки.

У результаті роботи було створено програмну систему яка інтегрує згадані вище моделі для ідентифікації штучно згенерованих облич. Система обробляє вхідні дані розміром 256x256 пікселів і проводить класифікацію на основі виділених ознак.

Тестування системи на збалансованому датасеті, що складається з 70 тисяч реальних та 70 тисяч штучних зображень, підтвердило її коректність та високу продуктивність. Моделі показали високу точність класифікації та здатність до узагальнення, що робить систему надійним інструментом для задач забезпечення безпеки та боротьби з дезінформацією.

Перспективи вдосконалення системи включають подальшу оптимізацію моделей для покращення продуктивності на мобільних пристроях, розширення набору даних для підвищення узагальнюючої здатності, а також інтеграцію з іншими системами для виявлення дезінформації та фальсифікації зображень. Завдяки цим покращенням, система матиме змогу більш ефективно виконувати свої функції у різних контекстах застосування, забезпечуючи високу точність і стабільність роботи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016.
2. Chesney R., Citron D. Deepfakes and the New Disinformation War: The Coming Age of Post-Truth Geopolitics // Foreign Affairs. 2019. Vol. 98, № 1. P. 147-155.
3. Vaccari C., Chadwick A. Deepfakes and Disinformation: Exploring the Impact on Social Trust and Democracy // International Journal of Communication. 2020. Vol. 14. P. 113-133.
4. Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018. P. 4510-4520.
5. Szeliski R. Computer Vision: Algorithms and Applications. Springer, 2010.
6. Litjens G., Kooi T., Bejnordi B.E., Setio A.A.A., Ciompi F., Ghafoorian M., van der Laak J.A.W.M., van Ginneken B., Sánchez C.I. A survey on deep learning in medical image analysis // Medical Image Analysis. 2017. Vol. 42. P. 60-88.
7. Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861, 2017.
8. Karras T., Laine S., Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2019.
9. Tolosana R., Vera-Rodriguez R., Fierrez J., Morales A., Ortega-Garcia J. DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection // Information Fusion. 2020. Vol. 64. P. 131-148.
10. Sutherland I.E. Sketchpad: A man-machine graphical communication system // Proceedings of the Spring Joint Computer Conference. 1963. P. 329-346.
11. Kingma D.P., Welling M. Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114, 2013.

12. Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y. Generative Adversarial Nets // Advances in Neural Information Processing Systems. 2014. Vol. 27.
13. Doersch C. Tutorial on Variational Autoencoders. arXiv preprint arXiv:1606.05908, 2016.
14. Dinh L., Krueger D., Bengio Y. NICE: Non-linear Independent Components Estimation. arXiv preprint arXiv:1410.8516, 2014.
15. Kingma D.P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions // Advances in Neural Information Processing Systems. 2018. Vol. 31. P. 10215-10224.
16. Oord A.v.d., Kalchbrenner N., Kavukcuoglu K. Pixel Recurrent Neural Networks // Proceedings of the 33rd International Conference on Machine Learning. 2016. P. 1747-1756.
17. James F. The use of CGI in modern cinema // Journal of Media Arts. 2018. Vol. 12, № 2. P. 45-56.
18. Jain A.K., Ross A. Multibiometric systems // Communications of the ACM. 2004. Vol. 47, № 1. P. 34-40.
19. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 770-778.
20. Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Computation. 1997. Vol. 9, № 8. P. 1735-1780.
21. Korshunov P., Marcel S. Deepfakes: a New Threat to Face Recognition? Assessment and Detection. arXiv preprint arXiv:1812.08685, 2019.
22. Benaim D., Copeland C. Deepfakes and the New Disinformation War. Foreign Affairs. URL: https://www.foreignaffairs.com/articles/world/2018-12-11/deepfakes-and-new-disinformation-war (дата звернення: 05.05.2024).
23. Harwell D. Top AI Researchers Race to Detect ‘Deepfake’ Videos: ‘We Are Outgunned’. The Washington Post. URL: https://www.washingtonpost.com/technology/2019/06/12/top-ai-researchers-

race-detect-deepfake-videos-we-are-outgunned/ (дата звернення: 05.05.2024).

24. Deepfake Pornography: Beyond Defamation Law // Yale Cyber Leadership Forum. URL: https://www.cyber.forum.yale.edu/blog/2021/7/20/deepfake-pornography-beyond-defamation-law (дата звернення: 05.05.2024).
25. Deepfake technology raises concerns among lawmakers // CNN. URL: https://edition.cnn.com/2019/01/28/tech/deepfake-lawmakers/index.html (дата звернення: 05.05.2024).
26. Creswell A., White T., Dumoulin V., Arulkumaran K., Sengupta B., Bharath A. A. Generative adversarial networks: An overview // IEEE Signal Processing Magazine. 2018. Vol. 35, № 1. P. 53-65. URL: https://b2a.kz/4ML (дата звернення: 05.05.2024).
27. Korshunov P., Marcel S. Deepfake detection: humans vs. machines. arXiv preprint arXiv:2009.03155, 2020. URL: https://b2a.kz/ULc (дата звернення: 05.05.2024).
28. Cortes C., Vapnik V. Support-vector networks // Machine Learning. 1995. Vol. 20, № 3. P. 273-297. URL: https://link.springer.com/article/10.1007/BF00994018 (дата звернення: 05.05.2024).
29. Quinlan J. R. Induction of decision trees // Machine Learning. 1986. Vol. 1, № 1. P. 81-106. URL: https://link.springer.com/article/10.1007/BF00116251 (дата звернення: 20.05.2024).
30. McCallum A., Nigam K. A comparison of event models for Naive Bayes text classification // AAAI-98 workshop on learning for text categorization. 1998. Vol. 752. P. 41-48. URL: https://www.aaai.org/Papers/Workshops/1998/WS-98-05/WS98-05-007.pdf (дата звернення: 20.05.2024).
31. Hinton G. E., Salakhutdinov R. R. Reducing the dimensionality of data with neural networks // Science. 2006. Vol. 313, № 5786. P. 504-507. URL: https://www.science.org/doi/10.1126/science.1127647 (дата звернення: 20.05.2024).
32. LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. 2015. Vol. 521, №

7553. P. 436-444. URL: https://www.nature.com/articles/nature14539 (дата звернення: 20.05.2024).

33. Hinton G. E., Salakhutdinov R. R. Reducing the dimensionality of data with neural networks // Science. 2006. Vol. 313, № 5786. P. 504-507. URL: https://www.science.org/doi/10.1126/science.1127647 (дата звернення: 20.05.2024).
34. An Introduction to Recurrent Neural Networks and the Math That Powers Them // Machine Learning Mastery. URL: https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/ (дата звернення: 20.05.2024).
35. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network // Physica D: Nonlinear Phenomena.
36. CS 230 - Recurrent Neural Networks Cheatsheet // Stanford University. URL: https://web.stanford.edu/class/cs230/ (дата звернення: 20.05.2024).
37. Recurrent Neural Network Tutorial // DataCamp. URL: https://www.datacamp.com/tutorial/tutorial-for-recurrent-neural-network (дата звернення: 20.05.2024).
38. Tan M., Le Q. V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks // Proceedings of the 36th International Conference on Machine Learning. 2019. Vol. 97. P. 6105-6114. URL: https://proceedings.mlr.press/v97/tan19a/tan19a.pdf (дата звернення: 20.05.2024).
39. Understanding ResNet50 architecture // OpenGenus IQ. URL: https://iq.opengenus.org/resnet50-architecture/ (дата звернення: 20.05.2024).
40. Understanding ResNet-50 in Depth: Architecture, Skip Connections, and Advantages Over Other Networks // Wisdom ML. URL: https://wisdomml.in/understanding-resnet-50-in-depth (дата звернення: 20.05.2024).
41. The Basics of ResNet50 // Weights & Biases. URL: https://wandb.ai/mostafaibrahim17/ml-articles/reports/The-Basics-of-

ResNet50---Vmlldzo2NDkwNDE2 (дата звернення: 20.05.2024).

42. ResNet and ResNetV2 // Keras. URL: https://keras.io/api/applications/resnet/#resnet50-function (дата звернення: 20.05.2024).
43. Beginners Guide to VGG16 Implementation in Keras | Built In. URL: https://builtin.com/machine-learning/vgg16
44. VGGNet-16 Architecture: A Complete Guide | Kaggle. URL: https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide (дата звернення: 20.05.2024).

ДОДАТОК А

```
import numpy as np
import pandas as pd
from tensorflow.keras.applications import MobileNetV2, VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Dense, BatchNormalization, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
import tensorflow as tf
import matplotlib.pyplot as plt
import cv2
import os
from tqdm.notebook import tqdm

print(os.listdir("/140k-real-and-fake-faces"))

dataset_path = "/140k-real-and-fake-faces/real_vs_fake/real-vs-fake"
train_path = os.path.join(dataset_path, 'train')
valid_path = os.path.join(dataset_path, 'valid')
test_path = os.path.join(dataset_path, 'test')

real_train_path = os.path.join(train_path, 'real')
fake_train_path = os.path.join(train_path, 'fake')

real_valid_path = os.path.join(valid_path, 'real')
fake_valid_path = os.path.join(valid_path, 'fake')
```

```
real_test_path = os.path.join(test_path, 'real')
fake_test_path = os.path.join(test_path, 'fake')

print("Train Path:", train_path)
print("Valid Path:", valid_path)
print("Test Path:", test_path)
print("Real Train Path:", real_train_path)
print("Fake Train Path:", fake_train_path)

print("Real Train Files:", os.listdir(real_train_path)[:5])
print("Fake Train Files:", os.listdir(fake_train_path)[:5])

def count_images_in_directory(directory):
    return len([file for file in os.listdir(directory) if file.endswith('.jpg')])

num_real_train = count_images_in_directory(real_train_path)
num_fake_train = count_images_in_directory(fake_train_path)

num_real_valid = count_images_in_directory(real_valid_path)
num_fake_valid = count_images_in_directory(fake_valid_path)

num_real_test = count_images_in_directory(real_test_path)
num_fake_test = count_images_in_directory(fake_test_path)

print(f"Number of real images in training set: {num_real_train}")
print(f"Number of fake images in training set: {num_fake_train}")
print(f"Number of real images in validation set: {num_real_valid}")
print(f"Number of fake images in validation set: {num_fake_valid}")
print(f"Number of real images in test set: {num_real_test}")
print(f"Number of fake images in test set: {num_fake_test}")
```

```
categories = ['Train Real', 'Train Fake', 'Valid Real', 'Valid Fake', 'Test Real', 'Test Fake']
counts = [num_real_train, num_fake_train, num_real_valid, num_fake_valid, num_real_test, num_fake_test]

plt.figure(figsize=(10, 6))
plt.bar(categories, counts, color=['blue', 'orange', 'blue', 'orange', 'blue', 'orange'])
plt.xlabel('Categories')
plt.ylabel('Number of Images')
plt.title('Number of Images in Each Category')
plt.show()

def load_img(path, target_size=(224, 224)):
    image = cv2.imread(path)
    if image is None:
        raise ValueError(f"Image not found at path: {path}")
    image = cv2.resize(image, target_size)
    return image[..., ::-1]

fig = plt.figure(figsize=(10, 10))

real_train_files = os.listdir(real_train_path)
if len(real_train_files) == 0:
    raise ValueError("No real train images found!")

for i in range(min(16, len(real_train_files))):
    plt.subplot(4, 4, i+1)
    plt.imshow(load_img(os.path.join(real_train_path, real_train_files[i])), cmap='gray')
```

```
    plt.suptitle("Real faces", fontsize=20)
    plt.axis('off')

plt.show()

fig = plt.figure(figsize=(10, 10))

fake_train_files = os.listdir(fake_train_path)
if len(fake_train_files) == 0:
    raise ValueError("No fake train images found!")

for i in range(min(16, len(fake_train_files))):
    plt.subplot(4, 4, i+1)
    plt.imshow(load_img(os.path.join(fake_train_path,                fake_train_files[i])),
cmap='gray')
    plt.suptitle("Fake faces", fontsize=20)
    plt.axis('off')

plt.show()

data_with_aug = ImageDataGenerator(horizontal_flip=True,
                                   vertical_flip=False,
                                   rescale=1./255,
                                   validation_split=0.3)

train = data_with_aug.flow_from_directory(train_path,
                                          class_mode="binary",
                                          target_size=(224, 224),
                                          batch_size=16)
val = data_with_aug.flow_from_directory(valid_path,
```

```
                                class_mode="binary",
                                target_size=(224, 224),
                                batch_size=16)

mnet = MobileNetV2(include_top=False, weights="imagenet", input_shape=(224, 224, 3))

tf.keras.backend.clear_session()

model1 = Sequential([tf.keras.layers.InputLayer(input_shape=(224, 224, 3)),
    mnet,
    GlobalAveragePooling2D(),
    Dense(512, activation="relu"),
    BatchNormalization(),
    Dropout(0.3),
    Dense(128, activation="relu"),
    Dropout(0.1),
    Dense(2, activation="softmax")
])

model1.compile(loss="sparse_categorical_crossentropy", optimizer=Adam(), metrics=["accuracy"])

model1.summary()

def scheduler(epoch):
    if epoch <= 2:
        return 0.001
    elif epoch > 2 and epoch <= 15:
        return 0.0001
```

```
    else:
        return 0.00001

lr_callbacks = tf.keras.callbacks.LearningRateScheduler(scheduler)

hist1 = model1.fit(train,
                   epochs=20,
                   callbacks=[lr_callbacks],
                   validation_data=val)

epochs = 20
train_loss = hist1.history['loss']
val_loss = hist1.history['val_loss']
train_acc = hist1.history['accuracy']
val_acc = hist1.history['val_accuracy']
xc = range(epochs)

plt.figure(1, figsize=(7, 5))
plt.plot(xc, train_loss)
plt.plot(xc, val_loss)
plt.xlabel('num of Epochs')
plt.ylabel('loss')
plt.title('train_loss vs val_loss')
plt.grid(True)
plt.legend(['train', 'val'])
plt.style.use(['classic'])

plt.figure(2, figsize=(7, 5))
plt.plot(xc, train_acc)
plt.plot(xc, val_acc)
```

```
plt.xlabel('num of Epochs')
plt.ylabel('accuracy')
plt.title('train_acc vs val_acc')
plt.grid(True)
plt.legend(['train', 'val'], loc=4)
plt.style.use(['classic'])

train = data_with_aug.flow_from_directory(train_path,
                                          class_mode="binary",
                                          target_size=(224, 224),
                                          batch_size=16)

val = data_with_aug.flow_from_directory(valid_path,
                                        class_mode="binary",
                                        target_size=(224, 224),
                                        batch_size=16)

vgg16_model = VGG16(include_top=False, weights="imagenet", input_shape=(224,
224, 3))

model2 = Sequential([
    tf.keras.layers.InputLayer(shape=(224, 224, 3)),
    vgg16_model,
    GlobalAveragePooling2D(),
    Dense(256, activation="relu"),
    Dropout(0.2),
    Dense(64, activation="relu"),
    Dense(2, activation="softmax")
])
```

```
model2.compile(loss="sparse_categorical_crossentropy", optimizer=Adam(), metrics=["accuracy"])

model2.summary()

hist2 = model2.fit(train,
                   epochs=20,
                   callbacks=[lr_callbacks],
                   validation_data=val)

ig = ImageDataGenerator(rescale=1./255.)
train_flow = ig.flow_from_directory(
    base_path + 'train/',
    target_size=(128, 128),
    batch_size=64,
    class_mode='categorical'
)

ig1 = ImageDataGenerator(rescale=1./255.)

valid_flow = ig1.flow_from_directory(
    base_path + 'valid/',
    target_size=(128, 128),
    batch_size=64,
    class_mode='categorical'
)

test_flow = ig.flow_from_directory(
    base_path + 'test/',
    target_size=(128, 128),
```

```
    batch_size=1,
    shuffle = False,
    class_mode='categorical'
)

train_flow.class_indices

from tensorflow.keras.optimizers import Adam

def build_model():
    resnet50 = ResNet50(
        weights='imagenet',
        include_top=False,
        input_shape=(128, 128, 3)
    )
    model = Sequential([
        resnet50,
        layers.GlobalAveragePooling2D(),
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(2, activation='softmax')
    ])
    model.compile(optimizer=Adam(learning_rate=0.001),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model

model3 = build_model()
model3.build(input_shape=(None, 128, 128, 3))
```

```
model3.summary()

hist3 = model.fit(train_flow,
            epochs = 10,
            callbacks=[lr_callbacks],
            validation_data=val)
)

plt.figure(figsize=(14,5))
plt.subplot(1,2,2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['train', 'val'])

plt.subplot(1,2,1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['train', 'val'])
plt.show()

import PIL
from PIL import Image
im1 = Image.open('/140k-real-and-fake-faces/real_vs_fake/real-vs-fake/test/fake/00276TOPP4.jpg')
```

```
im2=im1.resize((128,128))
p1 = np.array(im2)
p1=p1/255
plt.imshow(im1)
p1 = np.expand_dims(p1, axis=0)
p1.shape

val_path = "/140k-real-and-fake-faces/real_vs_fake/real-vs-fake"

plt.figure(figsize=(15, 15))

real_images_path = os.path.join(train_path, 'real')
fake_images_path = os.path.join(train_path, 'fake')

real_image_files = [os.path.join(real_images_path, f) for f in os.listdir(real_images_path) if f.endswith('.jpg')]
fake_image_files = [os.path.join(fake_images_path, f) for f in os.listdir(fake_images_path) if f.endswith('.jpg')]

image_files = real_image_files + fake_image_files
labels = [1] * len(real_image_files) + [0] * len(fake_image_files)

images = []
for file in image_files:
    img = load_img(file, target_size=(224, 224))
    img = img_to_array(img)
    img = np.expand_dims(img, axis=0)
    images.append(img)

images = np.vstack(images)
```

```
predictions_model1 = model1.predict(images)
predictions_model2 = model2.predict(images)

num_images_to_show = 8
for i in range(num_images_to_show):
    plt.subplot(4, 4, i + 1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])

    current_file = real_image_files[i]
    gt = 1
    preds_model1 = np.argmax(predictions_model1[i])
    preds_model2 = np.argmax(predictions_model2[i])
    col_model1 = "g" if preds_model1 == gt else "r"
    col_model2 = "g" if preds_model2 == gt else "r"

    plt.xlabel(f'M1: pred={preds_model1}, M2: pred={preds_model2}, gt={gt}',
color=col_model1 if preds_model1 == preds_model2 else "b")
    plt.imshow(load_img(current_file))

for i in range(num_images_to_show):
    plt.subplot(4, 4, i + num_images_to_show + 1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])

    current_file = fake_image_files[i]
    gt = 0
```

```
    preds_model1 = np.argmax(predictions_model1[i + len(real_image_files)])
    preds_model2 = np.argmax(predictions_model2[i + len(real_image_files)])
    col_model1 = "g" if preds_model1 == gt else "r"
    col_model2 = "g" if preds_model2 == gt else "r"

    plt.xlabel(f'M1: pred={preds_model1}, M2: pred={preds_model2}, gt={gt}', color=col_model1 if preds_model1 == preds_model2 else "b")
    plt.imshow(load_img(current_file))

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