

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту**

«На правах рукопису»
УДК [004.8.032.26:004.932.2](043.3)

До захисту допущено:
В.о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ
«__» _____ 2024 р.

**Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою «Системи і методи штучного
інтелекту»
зі спеціальності 122 «Комп'ютерні науки»
на тему: «Пошук відмінностей на зображеннях низької якості за
допомогою нейронних мереж»**

Виконав:
студент II курсу, групи KI-21мп
Володько Володимир Володимирович _____

Науковий керівник:
старший викладач кафедри ШІ, к.т.н.
Шаповал Наталія Віталіївна _____

Рецензент:
доцент кафедри ІКТС ННІТС, к.т.н., доцент
Астраханцев Андрій Анатолійович _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань

Студент: _____

Київ – 2024

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

Рівень вищої освіти – другий (магістерський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В.о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«30» серпня 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Володьку Володимирі Володимировичу

1. **Тема дисертації:** «Пошук відмінностей на зображеннях низької якості за допомогою нейронних мереж», науковий керівник дисертації Шаповал Наталія Віталіївна, к.т.н., затверджені наказом по університету від «08» листопада 2023 р. №5200-с

2. **Термін подання студентом дисертації:** 07.01.2024.

3. **Об'єкт дослідження:** зображення низької якості.

4. **Предмет дослідження:** різні типи нейронних мереж, які можуть застосовуватись для порівняння зображень й визначення наявності структурних змін.

5. **Перелік завдань, які потрібно розробити:**

- дослідити актуальність обраної теми;
- дослідити та проаналізувати методи порівняння зображень;
- вибрати оптимальну модель або систему моделей нейронних мереж для даної задачі;
- навчити моделі та провести експерименти;
- проаналізувати результати роботи моделей;
- порівняти результат з існуючими методами;
- оформити пояснювальну записку.

6. **Орієнтовний перелік графічного (ілюстративного) матеріалу:**
презентація.

7. **Дата видачі завдання:** 30.08.2023.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Затвердження теми МД	01.09.2023 – 06.09.2023	Виконано
2	Ознайомлення зі структурою МД згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	07.09.2023 – 11.09.2023	Виконано
3	Ознайомлення з ДСТУ 3008-2015 та стандартами ЄСПД	12.09.2023 – 13.09.2023	Виконано
4	Проведення дослідження за темою МД під керівництвом наукового керівника	14.09.2023 – 10.10.2023	Виконано
5	Завершення роботи над першим варіантом МД	11.10.2023 – 01.11.2023	Виконано
6	Проведення роботи над експериментальною частиною МД	02.11.2023 – 20.11.2023	Виконано
7	Проведення роботи над програмним продуктом	21.11.2023 – 18.12.2023	Виконано
8	Оформлення МД та аналіз отриманих результатів	19.12.2023 – 01.01.2024	Виконано

Студент

Володимир ВОЛОДЬКО

Науковий керівник

Наталія ШАПОВАЛ

РЕФЕРАТ

Магістерська робота містить: 79 с., 24 табл., 22 рис., додаток і 10 посилань.

Об'єктом дослідження є зображення низької якості.

Предметом дослідження є штучні нейронні мережі.

Метою роботи є розробка моделі або системи моделей нейронних мереж для проведення автоматизованого порівняння зображень низької якості. Завданнями дослідження є аналіз існуючих рішень, вибір нової моделі чи системи моделей нейронних мереж, навчання обраних типів нейронних мереж та аналіз отриманих результатів.

Новизна роботи полягає в застосуванні нової модифікації функції втрат та запропонованій архітектурі мережі-сегментатора на основі «кодувальник-декодувальник».

Актуальність роботи полягає в наявності великих об'ємів стиснутих зображень, що ускладнює вибір та пошук кращого варіанту з найменшою кількістю стиснень та дублюванням інформації на носіях.

Отримані результати свідчать про достатньо високу точність отриманої системи моделей нейронних мереж та її здатність розрізняти невеликі, але структурні на зображеннях ігноруючи при цьому шуми від артефактів стиснення. Подальше дослідження може спрямовуватись в напрямку ускладнення системи додаванням нових моделей та покращенням наявних.

МАШИННЕ НАВЧАННЯ, СІАМСЬКІ НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ЗОБРАЖЕНЬ, ПОРІВНЯННЯ ЗОБРАЖЕНЬ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, КОДУВАЛЬНИК-ДЕКОДУВАЛЬНИК.

ABSTRACT

Master's thesis: 79 p., 24 tables, 22 figures, appendix and 10 references.

The object of research is low quality images.

The subject of research is artificial neural networks.

The aim of the work is to develop a model or system of neural network models for automated comparison of low quality images. The objectives of the study are to analyze existing solutions, select a new model or system of neural network models, train the selected types of neural networks, and analyze the results.

The novelty of the work lies in the application of a new modification of the loss function and the proposed architecture of the segmentation network based on the «encoder-decoder».

The relevance of the work lies in the presence of large volumes of compressed images, which makes it difficult to select and find the best option with the least amount of compression and duplication of information on the media.

The results obtained indicate a fairly high accuracy of the obtained system of neural network models and its ability to distinguish small but structural features in images while ignoring noise from compression artifacts. Further research can be directed towards complicating the system by adding new models and improving existing ones.

MACHINE LEARNING, SIAMESE NEURAL NETWORKS, IMAGE PROCESSING, IMAGE COMPARISON, CONVOLUTIONAL NEURAL NETWORKS, ENCODER-DECODER..

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1 ОГЛЯД МЕТОДІВ ПОРІВНЯННЯ ЗОБРАЖЕНЬ.....	10
1.1 Актуальність проблеми і постановка задачі.....	10
1.2 Огляд існуючих рішень	12
1.2.1 Карти різниці.....	12
1.2.2 Сіамські нейронні мережі.....	13
1.2.3 Сіамська мережа з класифікатором	14
1.2.4 Порівняння описаних рішень	16
1.3 Завдання дослідження.....	17
Висновки до розділу 1	18
РОЗДІЛ 2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ.....	19
2.1 Нейронні мережі.....	19
2.2 Сіамські нейронні мережі.....	20
2.3 Згорткові нейронні мережі	21
2.4 Архітектура «кодувальник-декодувальник»	22
Висновки до розділу 2	23
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПОШУКУ ВІДМІННОСТЕЙ НА ЗОБРАЖЕННЯХ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ.....	24
3.1 Використані бібліотеки.....	24
3.2 Створення датасетів	25
3.3 Вибір моделей на невеликому датасеті.....	26
3.3.1 Перевірка практичної можливості	26
3.3.2 Навчання сіамської мережі.....	27
3.3.4 Навчання сіамської мережі на асиметричному датасеті	29
3.3.5 Навчання класифікатора на основні сіамської мережі	30

3.3.6 Підсумок відбору.....	31
3.4 Навчання сіамської моделі на збільшеному датасеті.....	32
3.4.1 Навчання моделі з параметром $\alpha = 1.5/2$	33
3.4.2 Навчання моделі з параметром $\alpha = 1.35/2$	35
3.4.3 Наступні варіації моделей для навчання.....	37
3.5 Навчання згорткової нейронної мережі.....	37
3.5 Оформлення виводу результатів	42
Висновки до розділу 3	42
РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЄКТУ	47
4.1 Опис ідеї проєкту	47
4.2 Технологічний аудит ідеї проєкту	49
4.3 Аналіз ринкової стратегії проєкту.....	57
4.4 Розроблення маркетингової програми стартап-проєкту	61
Висновки до розділу 4	65
ВИСНОВКИ.....	66
ПЕРЕЛІК ПОСИЛАНЬ.....	68
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ	70

ВСТУП

З поширенням персональних комп'ютерів та інших засобів цифрової обробки, збереження та передачі інформації почала спостерігатися тенденція до накопичення значної кількості інформації, в тому числі й в вигляді зображень. Через збільшення обсягу графічних даних виникла необхідність їх ефективного зберігання та передачі через телекомунікаційні мережі. В умовах з обмеженою кількістю доступної пам'яті та низькою пропускнуою здатністю мереж постало питання щодо ефективного стиснення зображень. Оскільки алгоритми стиснення без втрат інформації не забезпечували необхідного рівня зменшення розмірів зображень, широкого поширення набули алгоритми стиснення з втратами, які забезпечували помітне зменшення простору, необхідного для збереження зображення, жертвуючи при цьому незначним погіршенням якості. Але цей підхід мав один суттєвий недолік – зображення після стиснення не було тотожним оригіналу. Це створило потребу в нових методах порівняння зображень.

Магістерська дисертація має на меті провести дослідження та розробку моделі або системи моделей нейронних мереж для проведення автоматизованого порівняння зображень, які характеризуються низькою якістю. Особлива увага приділяється можливості виділення структурних змін у зображеннях від артефактів, що виникають при їх стисненні та забезпеченні наочності та простоти інтерпретації результату порівняння для людини. Такий алгоритм має вирішити складне завдання ідентифікації та відокремлення важливих відмінностей в зображеннях від артефактів стиснення, що є важливим в контексті багатьох застосувань, включаючи обробку даних відеоспостереження, відстеження шляхів розповсюдження інформації в соцмережах та багато інших галузей.

Актуальність роботи полягає в необхідності ефективних методів автоматизованого порівняння та виявлення змін у зображеннях, що стає актуальним у зв'язку з розширенням використання графічних даних та збільшенням їх обсягу в цифровому світі, накопиченню багаторазово стиснутих з втратами зображень.

РОЗДІЛ 1 ОГЛЯД МЕТОДІВ ПОРІВНЯННЯ ЗОБРАЖЕНЬ

1.1 Актуальність проблеми і постановка задачі

Сучасний розвиток та поширеність інформаційних технологій та генерація все більших об'ємів даних, в тому числі в графічному представленні, створює потреба в розвитку методів, які дали можливість б автоматизовано порівнювати та аналізувати зображення. Здійснення автоматизованого порівняння зображень необхідне в контексті пошукових систем, відеоспостереження, генерації відео, зберігання даних, редагуванні фото та відео.

Одним з аспектів даної проблеми є факт, що значна частина графічних даних піддається стисненню з втратами, що призводить до виникнення артефактів та втрати якості, що унеможлиблює порівняння наявних зображень звичайними методами, перевіряючи тотожність представлених даних. Це означає, що користувач повинен мати можливість обрати два зображення і система має визначити, чи між ними є відмінності, але ігноруючи при цьому результат зміни зображення через його зберігання з втратами.

Поширеною задачею стає не лише зберігання та передача зображень, але і виявлення різниці між оригінальним та стиснутими версіями. Зокрема, часто виникає необхідність видаляти копії одного та того ж зображення, які виникають через передачу через соціальні мережі, месенджери та інші канали, які використовують стиснення зображень.

Актуальність даної магістерської дисертації полягає в розв'язанні цієї складної задачі шляхом застосування нейронних мереж і розвитку нових методів та алгоритмів. Нейронні мережі демонструють багатообіцяючі результати в сфері обробки зображень. Дослідження в цьому напрямі можуть привести до покращення систем пошуку, зберігання зображень, генерації відео та фото, виявлення фейкових.

Отже, дана магістерська дисертація спрямована на розв'язання проблеми пошуку різниці між зображеннями низької якості. Дослідження в цьому напрямі дозволять суспільству більш ефективно обробляти графічні дані і сприятиме розвитку штучного інтелекту.

Об'єктом нашого дослідження є зображення низької якості.

Предметом дослідження є різні типи нейронних мереж, які можуть застосовуватись для порівняння зображень й визначення наявності структурних змін.

Метою дослідження є створити модель або системи моделей нейронних мереж, що з високою точністю буде визначати структурні різницю між вдома зображеннями, ігноруючи при цьому артефакти стиснення.

Загальну постановку задачі подано нижче.

Нехай I – це зображення, яке представляють собою масиви чисел в діапазоні $[0, 255]$ розміром $H \times W \times 3$, де H – висота зображення, W – ширина, 3 – кількість каналів на піксель. Кожен піксель зображення характеризується трьома компонентами кольору згідно моделі RGB: червоний (R), зелений (G) та синій (B).

Також задано два типи перетворення L та T :

$Y^L = L(I)$ – зображення I збережене з втратами так, що $Y^L \neq I$, але відмінності є малопомітними для людського ока.

$Y^T = T(I)$ – зображення I^T з доданим на нього текстом або іншими змінами, що легко помітні. Відповідно $Y^{TL} = L(T(I))$

Нехай дано зображення $Y \in \{Y^L, Y^{TL}\}$. Задача полягає у визначенні, до якого типу Y належить, тобто вирішенні такої задачі бінарної класифікації:

$$\text{Знайти } C(Y) = \begin{cases} 0, & \text{якщо } Y = Y^L, \\ 1, & \text{якщо } Y = Y^{TL}, \end{cases}$$

де $C(Y)$ – мітка класу для Y .

1.2 Огляд існуючих рішень

Для кращого вивчення питання пропонується розглянути існуючі варіанти алгоритмів порівняння зображень, які дозволяють оцінити схожість зображень. Визначити їх переваги, недоліки, якість їх роботи в заданій задачі.

1.2.1 *Карти різниці*

Попіксельне порівняння зображень, шляхом віднімання значень кольорів пікселів одного зображення від відповідних пікселів іншого, є досить простим і базовим методом оцінки різниці між зображеннями. Цей підхід полягає в тому, що для кожного пікселя обчислюється різниця в інтенсивності (або кольорі) між відповідними пікселями на обох зображеннях.

Метод полягає в тому, що від кожного пікселя на одному зображенні віднімаємо відповідний піксель на іншому зображенні. Якщо зображення містить кілька каналів – кожен канал порівнюється окремо. Далі береться абсолютне значення даної різниці. Результатом є нове зображення, де кожен піксель представляє різницю між відповідними пікселями на вихідних зображеннях.

Отримане зображення різниці часто називається "картою різниці" або "мапою різниці". Вона може відображати, де саме відбулися зміни між зображеннями. Зазвичай, на цій карті різниці області з невеликими відмінностями відображаються темніше, тоді як області з значущими змінами стають яскравішими. Приклад порівняння зображень цим методом представлено на рис. 1.1.



Рисунок 1.1 – Оригінальне зображення, результат стиснення, карта різниці, карта різниці зі збільшеною яскравістю

Цей метод є простим способом визначити наявність змін між зображеннями, але він не завжди може точно визначити природу цих змін або розрізнити артефакти від справжніх об'єктів. Також він чутливий до незначних різниць у яскравості та кольорі пікселів і може бути вразливим до шуму на зображеннях.

1.2.2 Сіамські нейронні мережі

Сіамська нейронна мережа – це архітектура глибокого навчання, яка використовує дві або більше ідентичні підмережі з спільними вагами та параметрами [1]. Ці мережі призначені для вирішення задач порівняння, визначення схожості або класифікації між вхідними даними. Головна мета цієї архітектури – навчити модель відображати дві різні вхідні точки даних у простір ознак таким чином, щоб подібні дані були близькими в цьому просторі, а різні – далекими.

Загальну структуру сіамських мереж подано на рис. 1.2.

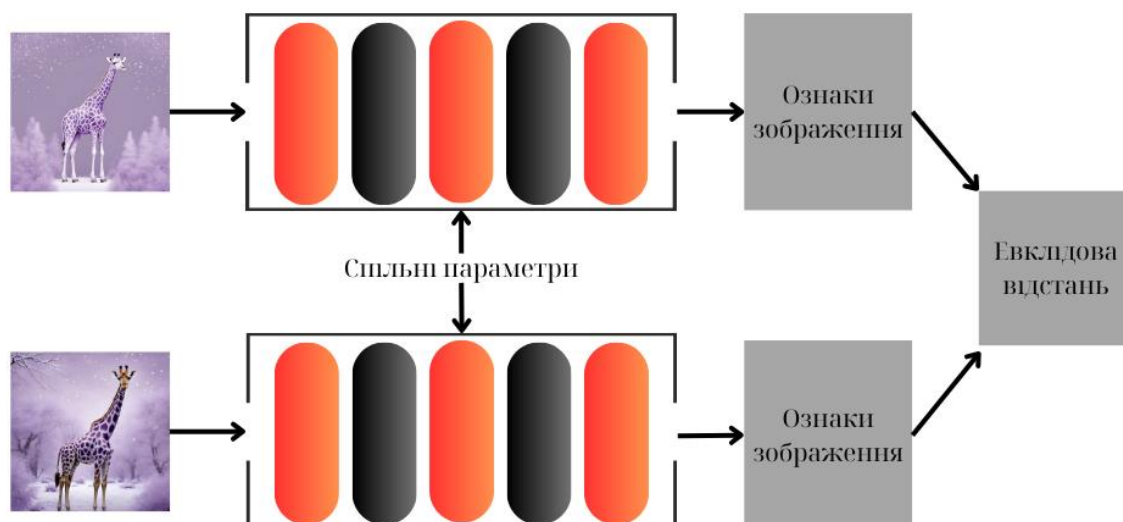


Рисунок 1.2 – Архітектура сіамської нейронної мережі

Дозволяє порівнювати різні зображення, як правило, заданого типу. Поширений варіант для порівняння зображень обличч, оскільки мережа виділяє конкретні ознаки на зображеннях при навчанні, які є важливими при порівнянні, є стійкою до шумів. В роботі [1] продемонстровано використання даного типу мережі з різними варіаціями архітектури підмережі в задачі порівняння обличч. В роботі [2] за допомогою аналогічного типу моделі нейронної мережі з різними архітектурами підмереж порівнюється схожість фотографій різних пам'яток архітектури й досягає точності 92.6%.

1.2.3 Сіамська мережа з класифікатором

Варіант модифікації сіамської нейронної мережі, який було запропоновано в роботі [3]. Модифікація полягає в зміні підходу перетворення відображення з простору ознак в мітки класів – подавати x_1 , x_2 як вхідні дані до іншої нейронної мережі-класифікатора, що дозволяє використовувати в якості гілок натреновані

на розпізнавання об'єктів нейронні мережі. Евклідова відстань в такому разі не застосовується, її заміняє нейромережа-класифікатор. Дана архітектура продемонстровано на рис. 1.3.

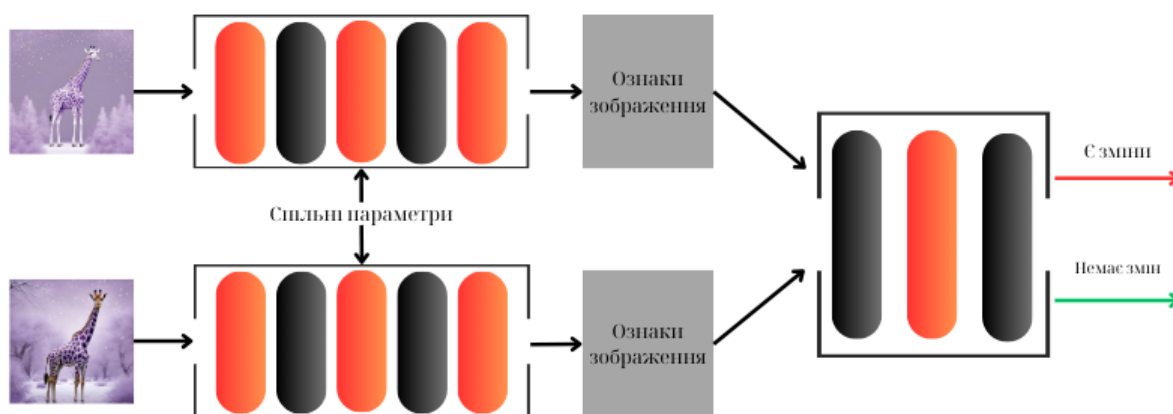


Рисунок 1.3 – Архітектура модифікованої сіамської нейронної мережі з класифікатором

Характеристики архітектури подібні до звичайної сіамської мережі, але дозволяє використовувати універсальні класифікуючі нейронні мережі в якості гілок сіамської мережі без їх спеціального навчання для конкретної області. Теоретично можуть покращити результат сіамської мережі завдяки більш складному, ніж звичайна ейлерова відстань, методу визначення схожості ознак. В роботі [3] продемонстровано використання мережі даного типу для порівняння супутникових знімків на наявність структурних змін. Модель продемонструвала точність 96.5% при тестуванні на датасеті з супутниковими знімками визначаючи з достатньою точністю наявність чи відсутність змін в умовах різного освітлення та балансу білого на зразках фотографій.

1.2.4 Порівняння описаних рішень

Сформуємо таблицю (табл. 1.1) з порівнянням описаних підходів, виставивши кожному рішенню відповідний бал за кожним з критеріїв від 0 до 2, де 0 – найгірший результат з представлених, 2 – найкращий. Оцінки вказують лише на переваги одного критерію над іншим, без оцінки якості даної переваги. Вплив кожного з критеріїв на кінцеву оцінку також не є рівнорозподіленим.

Таблиця 1.1 – Порівняння існуючих рішень

<i>Критерій</i>	<i>Карти різниці</i>	<i>Сіамська мережа</i>	<i>Сіамська мережа з класифікатором</i>
Простота	2	1	0
Наочність	2	0	0
Швидкість роботи	2	1	0
Точність	0	1	2
Нечутливість до шумів	0	2	2

Отже, як помітно з табл. 1.1 карти різниці є простими, швидкими й наочними, але при цьому мають низьку точність при наявності будь-якого типу змін та чутливість до шумів. Нейронні мережі, незважаючи на їх складність використання та гіршу швидкість роботи, яку необхідно компенсувати збільшенням використовуваних при порівнянні ресурсів, мають перевагу в точності та менш чутливі до шумів.

1.3 Завдання дослідження

Завданням даного дослідження є вирішення проблеми автоматизованого порівняння зображень шляхом визначення типу змін в зображенні – артефакти від стиснення чи внесені структурні зміни.

Важливим моментом є забезпечення високої точності визначення змін та мінімальна чутливість до артефактів стиснення.

Результати цього дослідження можуть мати застосування в різних галузях, пов'язаних з обробкою зображень. Конкретні завдання дослідження описані в наступних пунктах:

- аналіз існуючих рішень, їх пристосованість до конкретної задачі;
- розробка нейронної мережі чи системи нейронних мереж, що здатна розрізняти зміни на зображеннях з високою точністю;
- підготовка, навчання та тестування моделі на великому обсязі зображень з максимальною різноманітністю, для забезпечення універсальності отриманої моделі;
- аналіз отриманих результатів, порівняння їх з існуючими варіантами.

Для досягнення даної мети потрібно зробити наступне:

- провести аналіз наявних наборів зображень;
- на основі набору зображень згенерувати датасет;
- обрати алгоритм предобробки вхідних зображень;
- перевірити точність визначення наявності змін наявними методами;
- створити покращену модель для максимізації точності чи якісного покращення результату, наприклад, виділивши змінну область, визначити тип змін, тощо;
- підібрати найкращі гіперпараметри для моделі для отримання найкращих результатів.

Така модель має відповідати наступним вимогам:

- забезпечувати високу точність результатів:
- складність моделі повинна відповідати практичним потребам.

Визначивши вимоги до моделі можемо переходити до вибору технологій та інструментів.

Висновки до розділу 1

У цьому розділі описано актуальність проблеми, об'єкт, предмет та мету дослідження, зроблено огляд та порівняння існуючих рішень наведених в літературі на тему порівняння зображень, визначено завдання, які потрібно вирішити. Також сформульовано вимоги, на основі яких будуть сформульовані критерії перевірки отриманого результату. Розглянуто сіамські нейронні мережі та проаналізовано їх використання для порівняння зображень, що були стиснуті з втратами.

РОЗДІЛ 2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

2.1 Нейронні мережі

Нейронні мережі – це складні математичні моделі, які імітують роботу нейронів у людському мозку. Вони використовуються для обробки інформації, вирішення завдань штучного інтелекту та машинного навчання. Нейронні мережі мають широкий спектр застосувань, включаючи визнання образів, обробку природної мови, рекомендаційні системи, автономну навігацію, генетичні алгоритми та інше.

Основні аспекти нейронних мереж представлено нижче.

Структура – нейронні мережі складаються зі штучних нейронів, які взаємодіють один з одним. Нейрони групуються в різні шари, такі як вхідний, прихований і вихідний шари. Кожен нейрон отримує вхідні сигнали, обчислює вагову суму та застосовує активаційну функцію для генерації вихідного сигналу.

Навчання – нейронні мережі навчаються на прикладах. Під час навчання ваги нейронів адаптуються, щоб мережа навчалася вирішувати конкретне завдання. Популярні методи навчання включають зворотнє поширення помилки (backpropagation), яке коригує ваги, щоб зменшити помилку мережі.

Функції активації – кожен нейрон має функцію активації, яка визначає відповідь нейрона на вхідний сигнал. Популярні функції активації включають сигмоїду, ReLU (Rectified Linear Unit), тангенс та інші.

Існують різні типи нейронних мереж, включаючи звичайні шаровані мережі, згорнуті нейронні мережі (CNN), рекурентні нейронні мережі (RNN), а також багато інших модифікацій.

2.2 Сіамські нейронні мережі

Сіамські нейронні мережі – це спеціальний тип нейронних мереж, призначений для порівняння і визначення схожості між двома вхідними даними. Вони знайшли застосування в задачах розпізнавання образів, пошуку подібних об'єктів, аутентифікації осіб, виявлення аномалій та багатьох інших областях. Основною ідеєю за сіамськими нейронними мережами є створення подвійних гілок мережі з ідентичними вагами, які обробляють два вхідні зразки паралельно. Основні особливості сіамської нейронної мережі подано нижче.

1. Два вхідних набори даних: кожен набір оброблюється однією гілкою сіамської мережі, яка має свою власну копію архітектури. Ці гілки входять відокремлено в нейронну мережу.
2. Ваги та параметри: всі підмережі (гілки) мають однакові ваги та параметри. Тобто вони навчаються на однакових даних та спільно змінюються під час тренування.
3. Відображення в простір ознак: кожна гілка відображає вхідні дані в простір ознак, використовуючи шари нейронів та функції активації.
4. Обчислення відстані або схожості: результати відображення кожної з гілок порівнюються, визначається відстань між ними або їх подібність.
5. Функція втрат: враховує відстань або схожість, щоб змусити модель навчатися правильно класифікувати або порівнювати вхідні дані. Наприклад, для бінарної класифікації, це може бути функція запропонована в роботі [4]:

$$\begin{aligned} \text{loss}^B(x_1, x_2) = & (1 - \text{label}) \cdot (x_1 - x_2, x_1 - x_2) + \\ & + \text{label} \cdot \min(\text{margin}^2 - (x_1 - x_2, x_1 - x_2), 0), \end{aligned} \quad (1)$$

де *label* – це значення з множини $\{0, 1\}$, яке є міткою тотожності і різності класів відповідно, *margin* – наперед заданий гіперпараметр навчання, що задовільняє умову $margin > 0$.

6. Отримати мітку класу при навчанні з вище заданою функцією втрат можна за допомогою наступної формули:

$$C^S(I, Y) = \begin{cases} 0, \text{ якщо } \text{loss}^B(x_1, x_2) < \text{margin}, \\ 1, \text{ якщо } \text{loss}^B(x_1, x_2) \geq \text{margin}, \end{cases} \quad (2)$$

де x_1, x_2 – результат відображення в простір ознак нейронною мережею зображень I та Y відповідно.

2.3 Згорткові нейронні мережі

Згорткові нейронні мережі (Convolutional Neural Networks, CNNs) – потужним інструментом у галузі обробки зображень і інших подібних завдань. Основна їх ідея полягає в ефективному виявленні та використанні локальних особливостей во вхідних даних, таких як пікселі в зображеннях. Архітектура створена на основі структури зорової кори.

Відмінністю від решти типів нейронних мереж є наявність згорткових шарів. Особливості згорткових шарів полягають в:

- використовують фільтри (ядра) для сканування вхідних даних та виділення їхніх візуальних особливостей;
- фільтри переміщаються по вхідному зображенню, виконуючи операцію згортки, яка взаємодіє з регіонами зображення;
- вихід згорткового шару називається картою ознак.

Також в згорткових нейронних мережах застосовуються агрегувальні шари. Вони використовуються для зменшення просторових розмірів карти ознак,

що дозволяє зменшити кількість обчислень та забезпечує інваріантність до малих змін в вхідних даних.

CNN дозволяють автоматично вивчати ієрархічні та локальні функції, які представлені в зображеннях, роблячи їх дуже ефективними для завдань, пов'язаних із зображеннями, такими як класифікація об'єктів, визначення об'єктів, розпізнавання обличчя, сегментація та інші.

2.4 Архітектура «кодувальник-декодувальник»

Архітектура «кодувальник-декодувальник» – це структура нейронної мережі, яка являє собою глибоку згорткову нейронну мережу, що складається з двох основних частин: кодувальника і декодувальника (encoder-decoder) [5]. Ця архітектура часто використовується для завдань обробки сигналів, зображень та тексту, де важливо взаємодіяти з великим обсягом вхідних даних та генерувати вихідні представлення.

Кодувальник призначений для зменшення розміру вхідних даних та виділення їх важливих особливостей, зазвичай в кодувальник використовуються згорткові та агрегувальні шари. Вихід кодувальника представляє собою вектор ознак, що містить інформацію про важливі характеристики вхідних даних.

Декодувальник призначений для відновлення даних з отриманого коду від кодувальника. Зазвичай містить транспоновані згорткові шари та оператори підвищення роздільної здатності для відновлення просторових розмірів. Зазвичай, саме вихідні дані декодувальника і є виходом нейронної мережі.

Між кодувальником та декодувальником можуть бути додаткові компоненти, що допомагають у передачі інформації. Архітектури, такі як U-Net, засновані на ідеї encoder-decoder та містять додаткові шляхи для зберігання та використання деталей під час відновлення вихідних даних.

Архітектура encoder-decoder використовується у багатьох задачах, включаючи сегментацію зображень, генерацію зображень, машинний переклад та інші, оскільки вона ефективно працює з узагальненням та відновленням інформації з великого обсягу вхідних даних.

Висновки до розділу 2

В цьому розділі роботи розглянуто ключові теоретичні основи для подальшої роботи. Зокрема розглянуто загальний підхід до нейронних мереж, типи нейронних мереж, які найчастіше застосовуються при обробці зображень та процес навчання. Окремо описано згорткові нейронні мережі, які є основним типом нейронних мереж, що застосовуються для обробки зображень і мають значну перевагу в продуктивності перед повнозв'язними лінійними мережами та зменшують чутливість мережі до шумів у вхідних даних. Також наведено опис сіамських нейронних мереж, оскільки вони найчастіше застосовуються для порівняння об'єктів і демонструють хороші результати при порівнянні об'єктів і визначення їх схожості, хоч і без можливості виділити чи більш конкретно описати різницю об'єктів. Ще однією описаною архітектурою є «кодувальник-декодувальник» – це тип мережі, який застосовують в задачах сегментації зображень. І хоч він не призначений для виявлення змін чи їх відсутності на зображенні, такий тип мереж можна застосовувати для виділення зміненої області на зображеннях, де попередньо відомо про наявність змін, що допоможе забезпечити наочність результатів порівняння.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПОШУКУ ВІДМІННОСТЕЙ НА ЗОБРАЖЕННЯХ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

У світі, насиченому зображеннями, порівняння та обмін фотографіями стали невід’ємною частиною нашого цифрового життя. В сучасному інтернет-просторі користувачі активно взаємодіють зі своїми зображеннями, обмінюються ними, редагують їх, зберігають та надсилають результати. Завдяки цьому контенту виникає потреба в ефективних інструментах порівняння та аналізу зображень, які будуть ефективно працювати з зображеннями після їх багаторазового стиснення й появи відповідних артефактів. У даному розділі було розроблено систему моделей сіамської та згорткової нейронної мережі для визначення наявності редагування зображення не через артефакти при стисненні та виділення відредагованої області, якщо їх виявлено.

3.1 Використані бібліотеки

Використані бібліотеки:

1. PIL (Python Imaging Library) – використовується для роботи з зображеннями.
2. torch – бібліотека для конструювання нейронних мереж та їх навчання.
3. numpy – використовується для предобробки даних.
4. torchvision – доповнення до бібліотеки torch для конвертації результату виводу нейронної мережі.
5. matplotlib.pyplot – використовується для побудови графіків.
6. tkinter – для реалізації графічного інтерфейсу для виводу результатів.

3.2 Створення датасетів

Початковий етап дослідження полягав у створенні датасетів, які будуть використовуватись для перевірки моделі, та для навчання кожної з нейронних мереж. Оскільки задача мала специфіку наявності змін від стиснення зображень та структурних змін від редагування, готового датасету не існувало, але є можливість згенерувати датасет, маючи набір зображень та програмними засобами додати артефакти стиснення та редагування. Для цього було використано Також це дозволяло змінювати коефіцієнт стиснення i , відповідно, кількості артефактів на зображеннях, змінювати тип редагування, обирати співвідношення кількості пар зображень з наявністю та відсутністю структурних змін. Також це дозволяло створювати маску для навчання згорткової мережі для виділення відредагованої області та нормалізувати розміри для всіх зображень.

Для початкової перевірки практичної можливості результату був створений датасет на основі 327 зображень, що поділені на 279 прикладів в тренувальному наборі та 48 в тестовому. Основою для формування датасетів слугували зображення, які були вилучені із кешу месенджера Telegram на особистому мобільному пристрої. Відбір та використання цих зображень обумовлені необхідністю використання реальних та різноманітних фотографій, які надають адекватний та репрезентативний набір даних для подальших етапів нашого дослідження. Розмір датасету цілеспрямовано був невеликим, щоб пришвидшити перевірку ідей і базове порівняння, не орієнтуючись на фінальний результат.

Для фінального навчання відібраних моделей було застосовано датасети згенеровані на основі частини Open Images Dataset розміром 156541 зображень. Для зменшення вимог до апаратного забезпечення при навчанні він був розбитий на частини. Для навчання сіамської мережі це було 15 частин по 10000 пар в кожній, де 9000 пар – тренувальний набір і 1000 – тестовий. Кожна пара – це оригінальне зображення та зображення з артефактами стиснення й, з заданою

ймовірністю, додатково відредагована перед стисненням. Для згорткової мережі, що виділяє відредаговану область, датасет був розбитий на 31 частину по 5000 наборів з аналогічних пар зображень, маски відмінності та карти різниці, для пришвидшення навчання. В кожній частині було 4500 тренувальних та 500 тестових наборів, всі набори мали наявність відредагованих змін.

Оскільки зображення зменшувались в розмірах варіанти з тестування предобробки за допомогою різних типів фільтрації було відкинуто, оскільки інтерполяція при зменшенні зображень виконує дане завдання.

3.3 Вибір моделей на невеликому датасеті

Перевірка практичної можливості відокремлення зображень, що відрізняються артефактами стиснення та відредагованих, перевірка доцільності використання різних моделей й пріоритетність їх навчання на більшому датасеті.

3.3.1 Перевірка практичної можливості

Для визначення практичної можливості відділити просто стиснуті з втратами зображення від таких, що мають додаткові зміни застосовано модель, запропоновану в роботі [3]. В якості гілок сіамської нейронної мережі для відображення в простір ознак використовується модель VGG16 з преднавченими вагами imagenet та навчанням нейронної мережі, що класифікує результат відображення. Співвідношення кількості пар зображень з наявністю та відсутністю структурних змін в тренувальному та тестовому наборах – 1:1. Функція втрат – функція перехресної ентропії (cross-entropy loss function). Графік зміни функції втрат зображень на рис. 3.1.

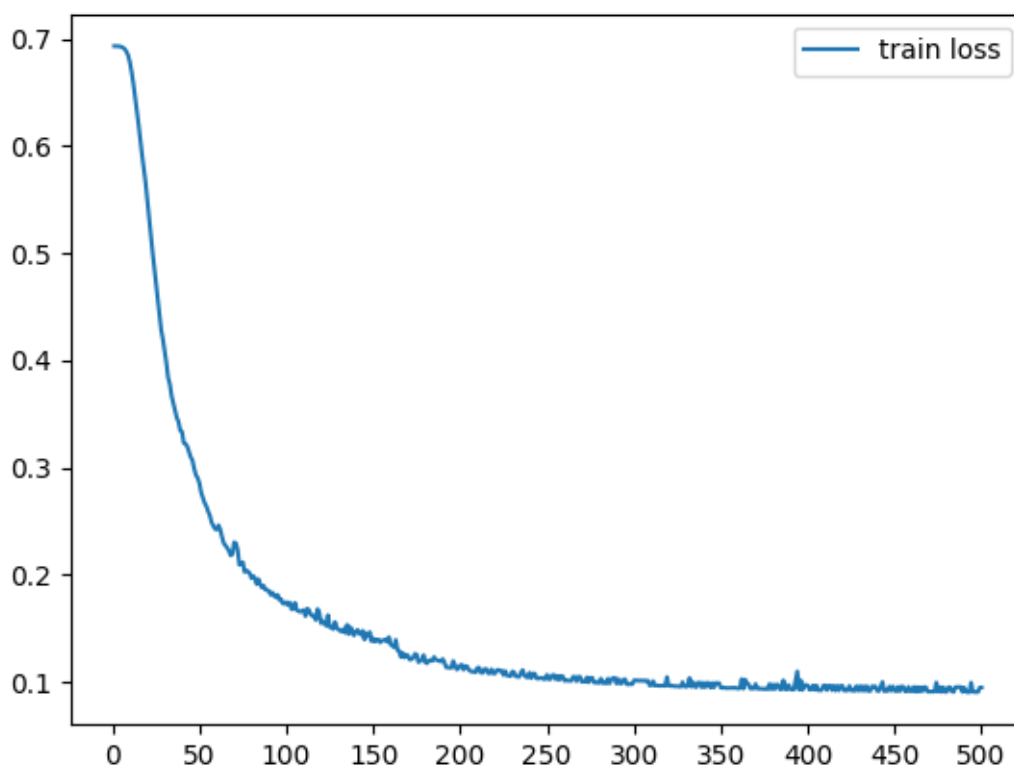


Рисунок 3.1 – Графік функції втрат в першому експерименті

Найкращу точність на тестовій вибірці продемонструвала модель навчена протягом 350 епох – 66.67%, що демонструє практичну можливість відокремити відредаговані зображення від лише стиснутих.

3.3.2 Навчання сіамської мережі

Наступним варіантом моделі було навчання сіамської мережі для відображення зображення в простір ознак, для отримання міток класів використовувалась функція 23 гіперпараметром $margin = 3$. Графік зміни функції втрати зображень на рис 3.2.

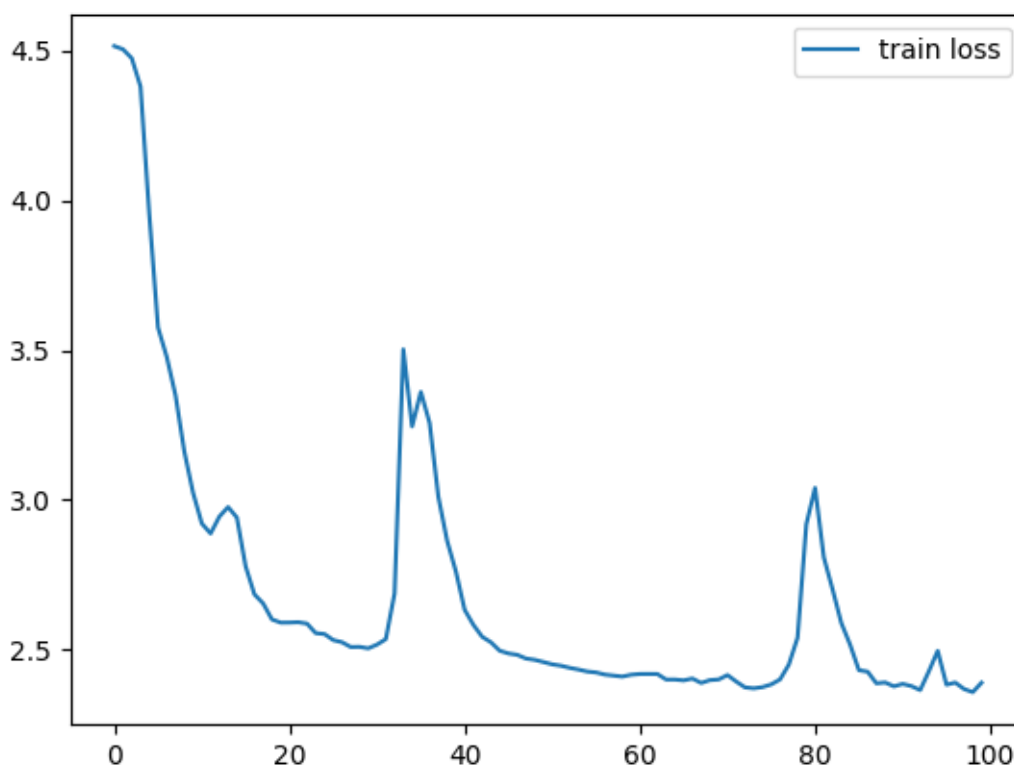


Рисунок 3.2 – Графік функції втрат під час навчання сіамської мережі

Отримана точність при навчанні протягом 100 епох була 52% на тестовій вибірці. Відображення x_1 , x_2 були завжди близькі один до одного, як для пар з Y^{LT} , так і для пар з Y^L , а зменшення функції втрат, як видно з графіку на рис. 3.2, було нестабільним. Як висновок, припускаю що маючи 50% пар зображень з Y^L утворюється локальний оптимум, коли результати відображення в простір ознак є подібними для всіх зображень, незалежно від вхідних значень. Збільшення гіперпараметру *margin* приводило до переповнення значення помилки після кількох епох й унеможлилювали подальше навчання.

3.3.4 Навчання сіамської мережі на асиметричному датасеті

Для вирішення проблеми нестабільного спадання значення функції втрат та покращення результатів в датасеті було змінено співвідношення кількості пар з структурними змінами та без так, щоб в тренувальному наборі було співвідношення 9:1, в тестовому залишено 1:1. Графік зміни функції втрат зображено на рис. 3.3.

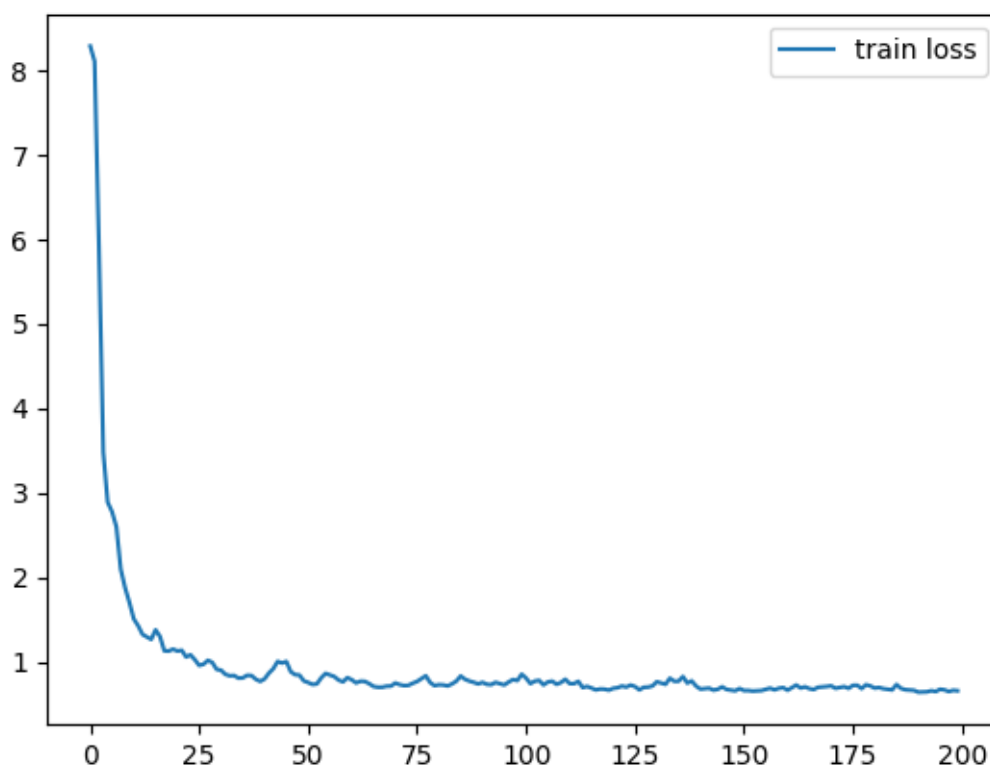


Рисунок 3.3 – Графік функції втрат під час навчання сіамської мережі з асиметричним датасетом

Навчена протягом 200 епох мережа демонструвала точність класифікації 68.75% на тестовій вибірці, що демонструє значно кращий результат в порівнянні з симетричним датасетом та демонструє практичну можливість за допомогою сіамської мережі розрізняти зображення з наявними структурними

змінami та лише стиснуті. Результат при цьому демонструє незначне підвищення точності в порівнянні з застосуванням преднавчених ваг мережі-класифікатора.

3.3.5 Навчання класифікатора на основі сіамської мережі

Використовується модель запропонована в роботі [3]. В якості гілок сіамської нейронної мережі для відображення в простір ознак використовується сіамська мережа навчена на попередньому кроці та навчанням нейронної мережі, що класифікує результат відображення. На рис. 3.4 зображено графік значення функції втрат протягом перших 5000 епох навчання.

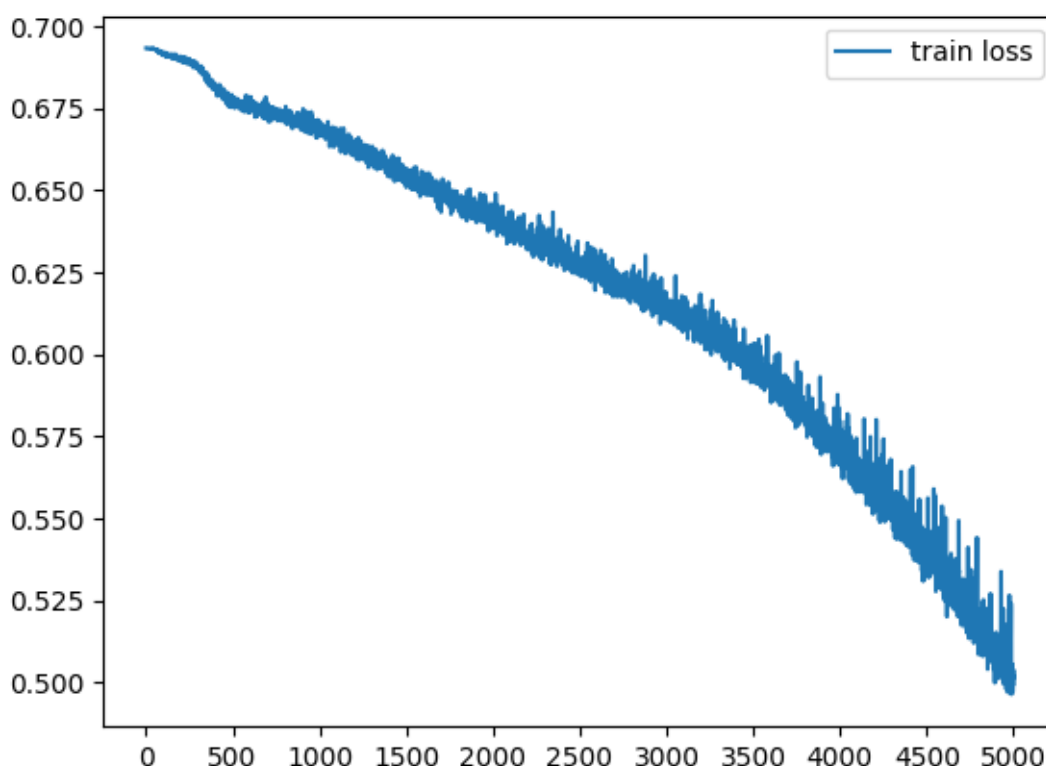


Рисунок 3.4 – Графік функції втрати мережі класифікатора результатів виводу сіамської нейронної мережі

Мережа демонструвала повільне покращення точності й досягла 68.75% на тестовій вибірці і не покращувалась з продовженням навчання. По графіку функції втрат помітно, що він спадає нестабільно й має відносно незначне зменшення значень.

3.3.6 Підсумок відбору

Для зручності порівняння отриманих результатів їх наведено в табл. 3.1.

Таблиця 3.1 – Короткий результати

<i>№ n/n</i>	<i>Класифі катор</i>	<i>Короткий опис</i>	<i>Точність</i>	<i>Короткий висновок з результату</i>
1	Нейрон на мережа	VGG16 з набором ваг imagenet і навчання класифікатора відображення	66.67%	Підтверджено практичну можливість віддлення типів пар зображень
2	Функція C ^s	Навчання сіамської мережі	52%	Виявлено проблему з поширеністю помилки першого роду
3	Функція C ^s	Сіамська мережа з несиметричним датасетом	68.75%	Підтверджено можливість застосування сіамської мережі й можливість покращити результат
4	Нейрон на мережа	Сіамська мережа з пункту 3 та навчання класифікатора відображення	68.75%	Повторено, але не покращено, результат лінійного розділення,

Як результат відбору на меншому датасеті було вирішено тренувати сіамську нейронну мережу на несиметричному датасеті й приділити увагу співвідношенню помилок першого та другого роду в результатах моделей. Навчання класифікатора для результатів відображення зображення в простір ознак сіамською нейронною мережею було визначено як можливий шлях покращення в майбутньому без повторних експериментів в даній роботі.

3.4 Навчання сіамської моделі на збільшеному датасеті

Навчання сіамської мережі на збільшеному датасеті відбувалось на вже асиметричному наборі з співвідношенням пар зображень з структурними змінами до зображень, що відрізняються через стиснення, 9:1. В результаті найкраща точність під час навчання становила 74.3% й з продовженням навчання результат не покращився, на останньому етапі він становив 71.3%. При чому серед невірно класифікованих прикладів 92.23% та 79.79% відповідно складала похибка першого роду. Це свідчить, що простої зміни співвідношення недостатньо. Для вирішення даної проблеми формулу (1) було змінено на формулу (3):

$$\begin{aligned} \text{loss}^{\text{BM}}(x_1, x_2) = & (1 - \text{label}) \cdot (x_1 - x_2, x_1 - x_2)^\alpha + \\ & + \text{label} \cdot \max(\text{margin}^2 - (x_1 - x_2, x_1 - x_2), 0). \end{aligned} \quad (3)$$

3.4.1 Навчання моделі з параметром $\alpha = 1.5/2$

Зменшено вплив похибки першого роду на значення функції втрат та асиметричний датасет. На рис. 3.5 зображено графік функції втрат під час навчання.



Рисунок 3.5 – Графік функції втрат при навчанні з $\alpha = 1.5/2$

Варто відзначити, що найменші значення втрат було на перших етапах і подальше навчання не покращувало ситуацію. Це дозволяє припустити, що основні результати будуть помітні вже на перших етапах навчання і можна скоротити час перевірки гіпотез при варіації параметру α та інших гіперпараметрів моделі. Для впевненості в даному припущенні також на рис. 3.6 наведено графік точності класифікації використовуючи формулу (3).

Для визначення, чи потрібно змінювати значення α і як наведемо графік помилки другого роду на рис. 3.7.

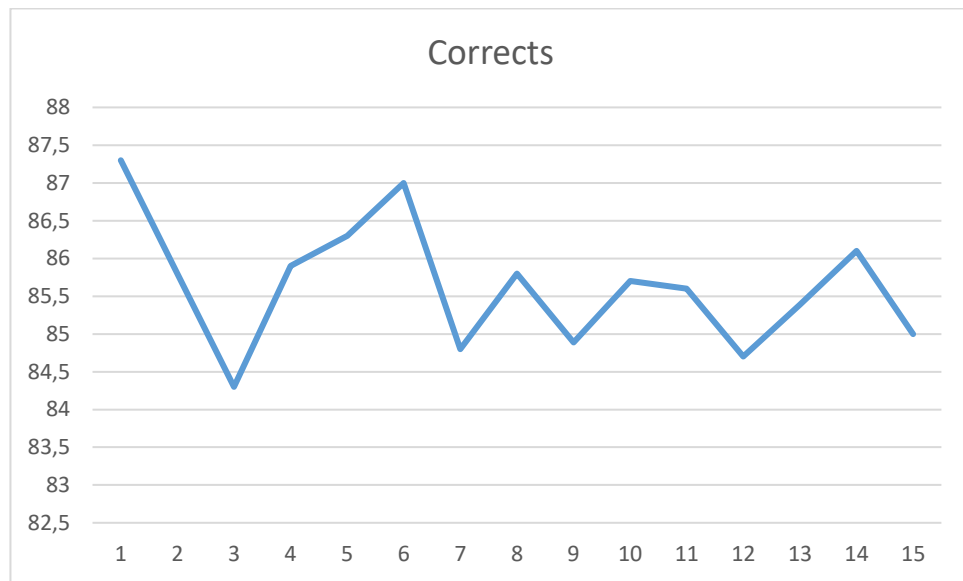


Рисунок 3.6 – Графік точності класифікації сіамської мережі

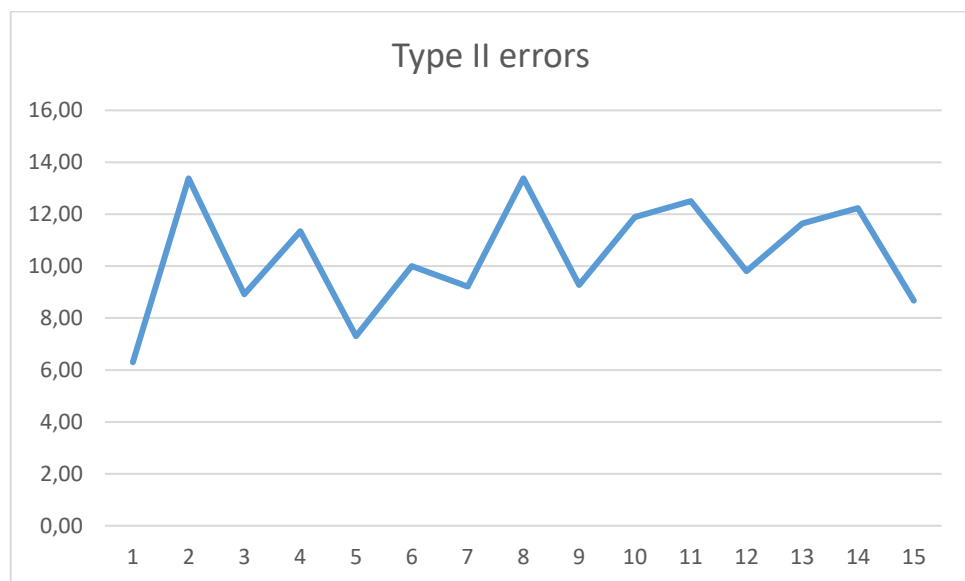


Рисунок 3.7 – Графік помилки другого роду

Оскільки, згідно графіку на рис. 3.7, помилка першого роду все ще зустрічається в кілька разів частіше, в наступному експерименті зменшимо параметр α .

3.4.2 Навчання моделі з параметром $\alpha = 1.35/2$

Зменшивши параметр α проведемо навчання мережі, кількість етапів буде зменшено до 5. Графік функції втрат наведено на рис. 3.8.



Рисунок 3.8 – Графік функції втрат при навчанні з $\alpha = 1.35/2$

Варто відмітити, що значення на графіку є більшими в порівнянні з значеннями попереднього варіанту, але оскільки були внесені зміни в функцію обчислення втрат, їх не варто порівнювати між собою. А для порівняння результатів навчання на рис. 3.9 та рис. 3.10 наведено графіки точності та кількості помилок другого роду відповідно.

З графіка точності помітно покращення результатів до 89,6%, що є найкращим результатом протягом експериментів. Графік помилки другого роду ж демонструє, що співвідношення наблизилось до 1:1.

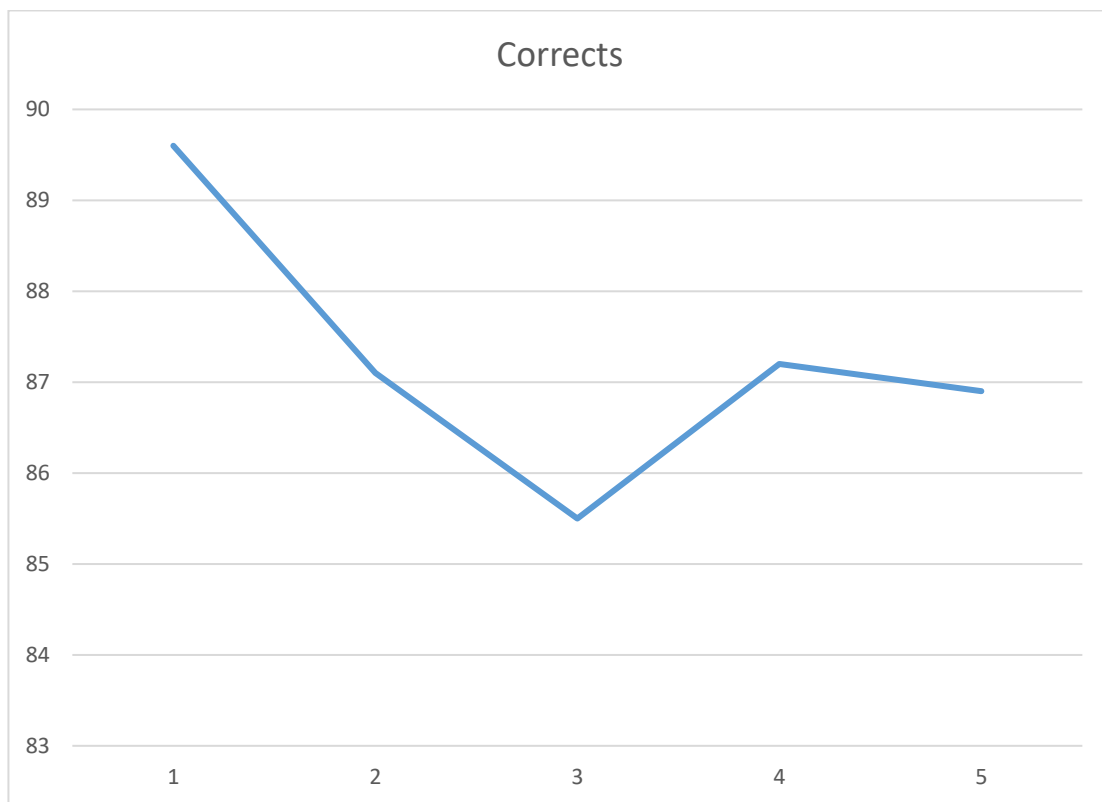


Рисунок 3.9 – Графік точності при навчанні з $\alpha = 1.35/2$

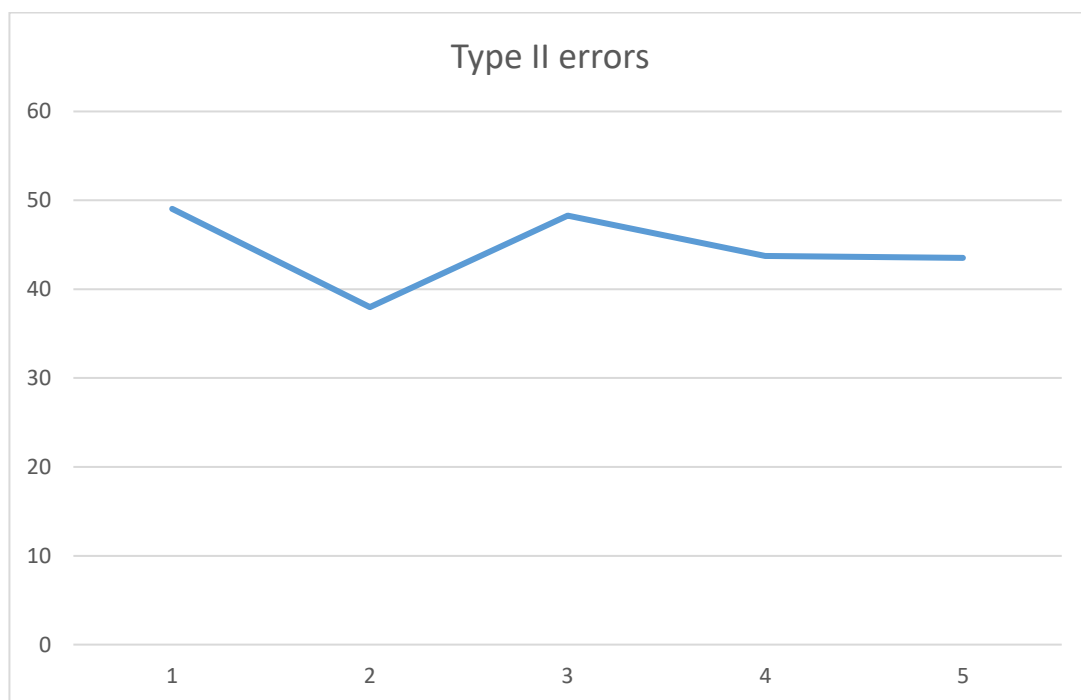


Рисунок 3.10 – Графік помилки другого роду при навчанні з $\alpha = 1.35/2$

3.4.3 Наступні варіації моделей для навчання

Оскільки основні зміни результату помітні вже на перших етапів для пришвидшення вибору моделі було прийнято рішення відразу зупиняти навчання, якщо початковий результат гірший ніж в пункті 3.3.2 та не демонструє тенденцію до навчання.

Зменшення значення α призвело до збільшення помилки другого роду до 62% на першому етапі з поступовим збільшенням, точність зменшилась до 80,2% з тенденцією до незначного зменшення. З варіантів архітектури нейронної мережі найкращий результат продемонструвала зображена на рис. 3.11, яка була застосована при навчанні в пункті 3.3.2.

3.5 Навчання згорткової нейронної мережі

Окрім лише класифікації наявності структурних змін на зображенні для наочності варто виділити сегмент, що відрізняється. Перший варіант сегментатора, котрий планувався як універсальний засіб визначення наявності змін та відразу їх виділення не демонстрував практично значущих результатів, оскільки орієнтувався на форму зміненої області і виділяв схожі об'єкти, навіть якщо вони присутні на обох зображеннях. Спроба навчити сегментатор на основі вихідних даних сіамської нейронної мережі також завдала невдачі через відсутність в результаті даних про розміщення області змін на зображенні.

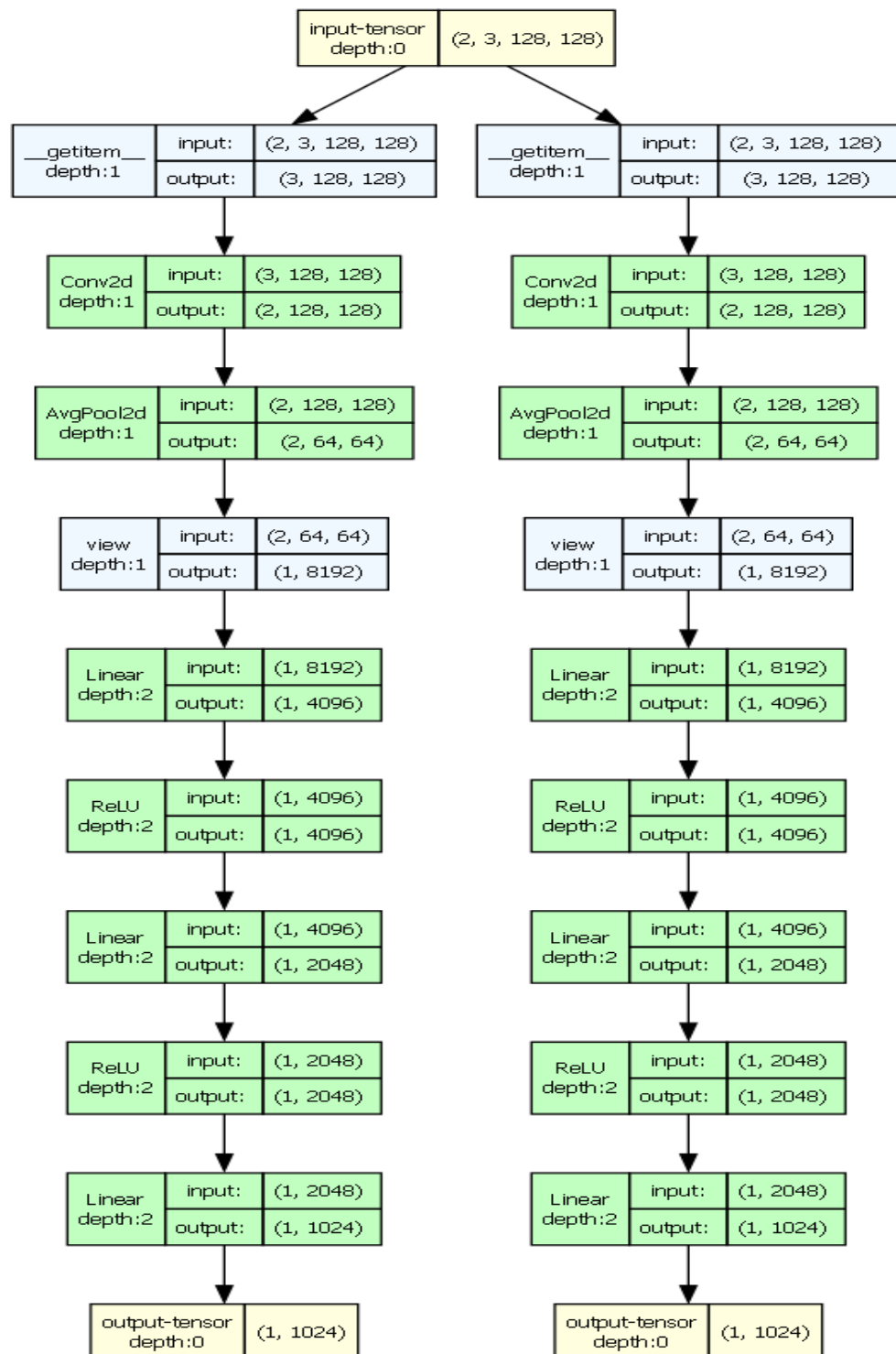


Рисунок 3.11 – Фінальна архітектура сіамської нейронної мережі

Для покращення результату було запропоновано наступне:

- звузити область застосування сегментатора до умов, коли вже відомо про наявність структурних змін;

- подібно до сегментаторів на основі тексту використати результат відображення сіамської нейронної мережі як допоміжні дані;
- використати карту різниці в якості вхідних даних.

Результатом стала архітектура, яку можна розділити на три частини: кодувальник, конкатенатор та декодувальник. Кожна з цих частин детально зображена на рис. 3.12–3.14.

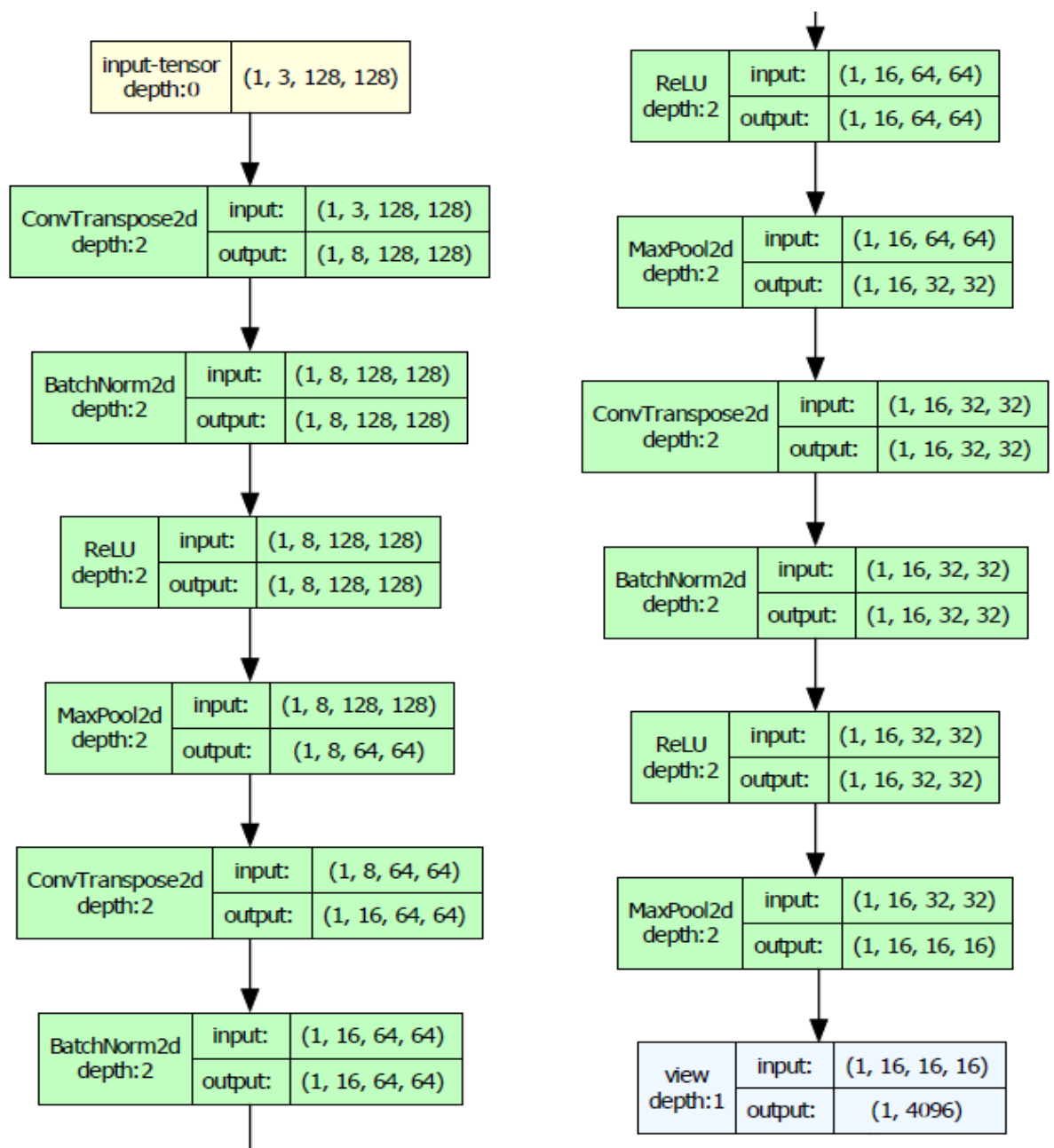


Рисунок 3.12 – Архітектура кодувальника моделі-сегментатора

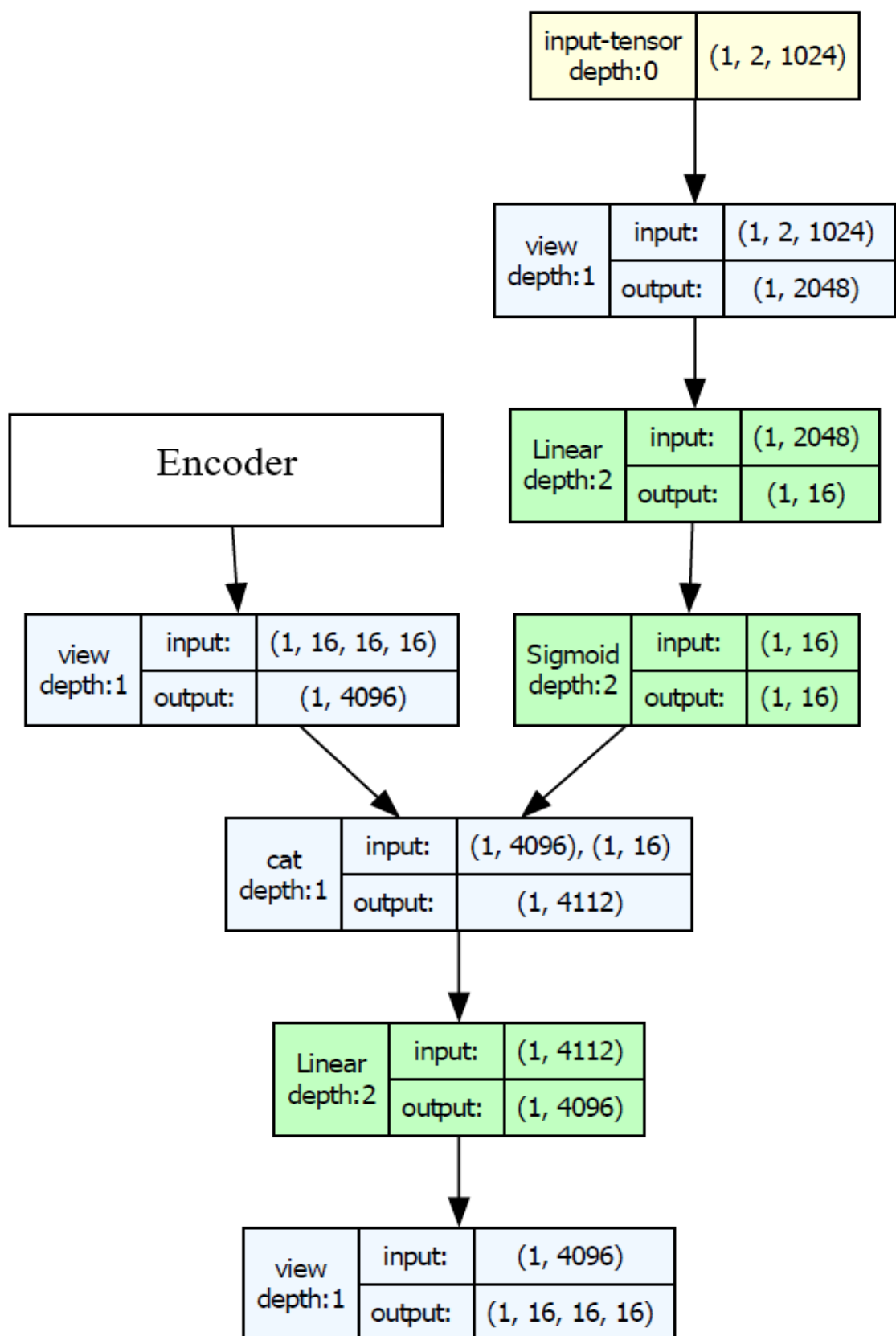


Рисунок 3.13 – Запропонована архітектура конкатенатора моделі-сегментатора

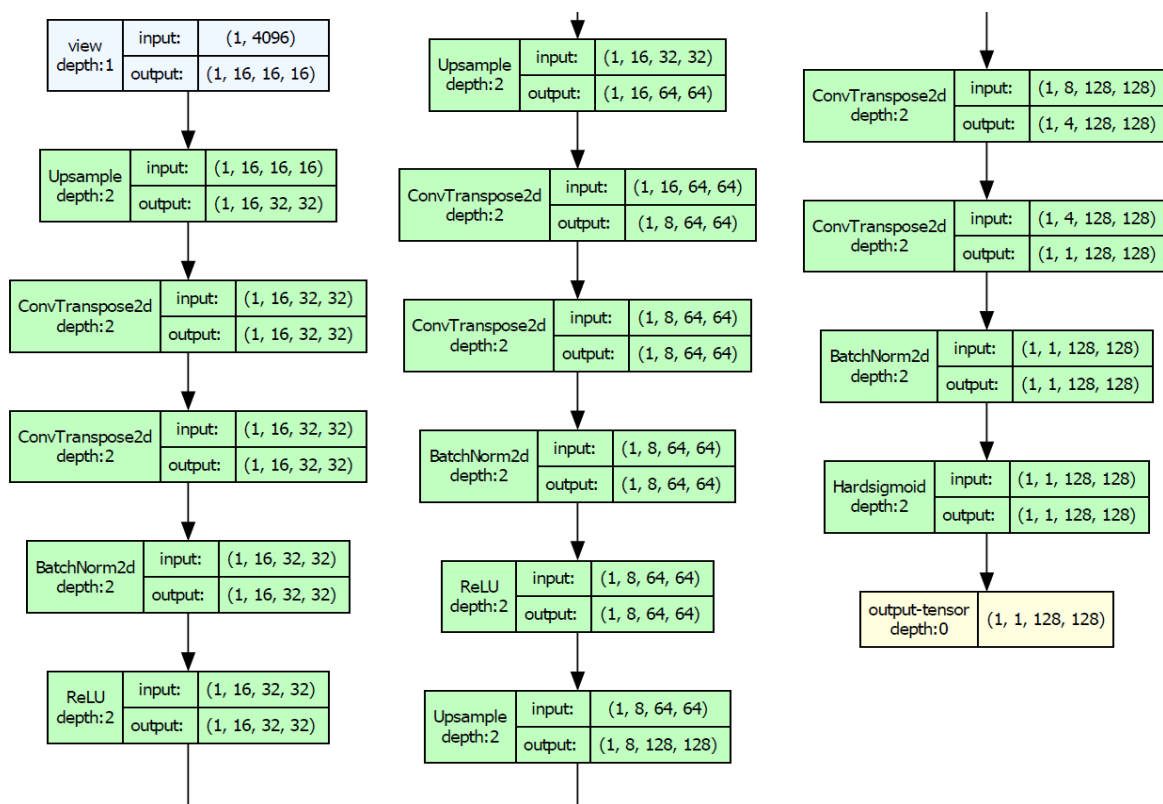


Рисунок 3.14 – Архітектура декодувльника моделі-сегментатора

Графік функції втрати під час навчання зображено на рис. 3.15. В якості функції втрат використовується попіксельна Binary Cross Entropy Loss. Найкращий отриманий результат становив 0,024.

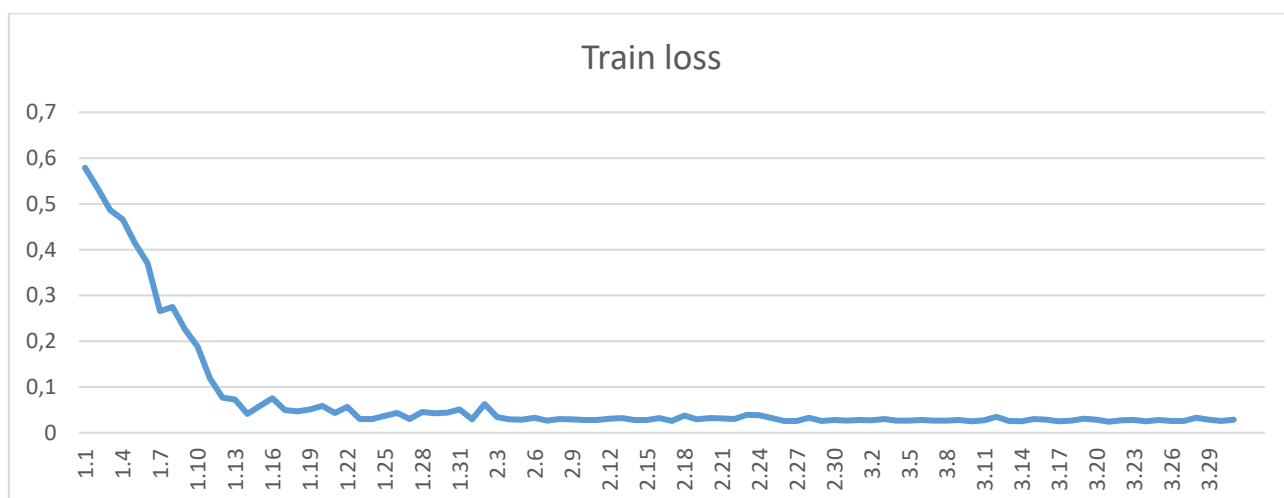


Рисунок 3.15 – Графік функції втрат під час навчання сегментатора

3.5 Оформлення виводу результатів

Для зручного виводу результатів сіамська нейронна мережа та сегментатор були об'єднані в одній програмі з вибором зображень та виводом числового значення відмінності зображень, оцінки наявності змін на основі критерію з формули (2) та карти різниці. Якщо пару зображень було класифіковано як такі, що мають структурні відмінності, то виведено результат сегментації зміненої області. Приклади виводу результатів зображено на рис. 3.16–3.19.

Висновки до розділу 3

В цьому розділі показано процес створення програмного продукту. Проведено початковий відбір моделей на зменшеному датасеті для підтвердження можливості відділяти зображення з артефактами стиснення від пар зображень з структурними змінами та оцінки перспективності навчання різних моделей сіамських мереж. На збільшеному датасеті навчено сіамську нейронну мережу відділяти зображення з та без змін. Запропоновано модифікацію функції втрат для вирішення проблеми дизбалансу помилки першого та другого типів та застосування датасету з різним кількостями прикладів класів в тренувальній вибірці. Навчено мережу-сегментатор виділяти змінні області за допомогою карти різниці та результатів відображення сіамської мережі. Запропоновано варіант модифікації «кодувальник-декодувальник» архітектури для покращення універсальності моделі за допомогою додавання конкатенатора з сіамською мережею між блоками кодувальника та декодувальника. Реалізовано програму, що використовуючи систему з нейронних мереж порівнює зображення з практично доречним рівнем точності. Виділення зміненої області також працює достатньо точно, демонструючи 89.7% точності при визначенні наявності змін, хоч і демонструє некоректний результат в складних умовах.

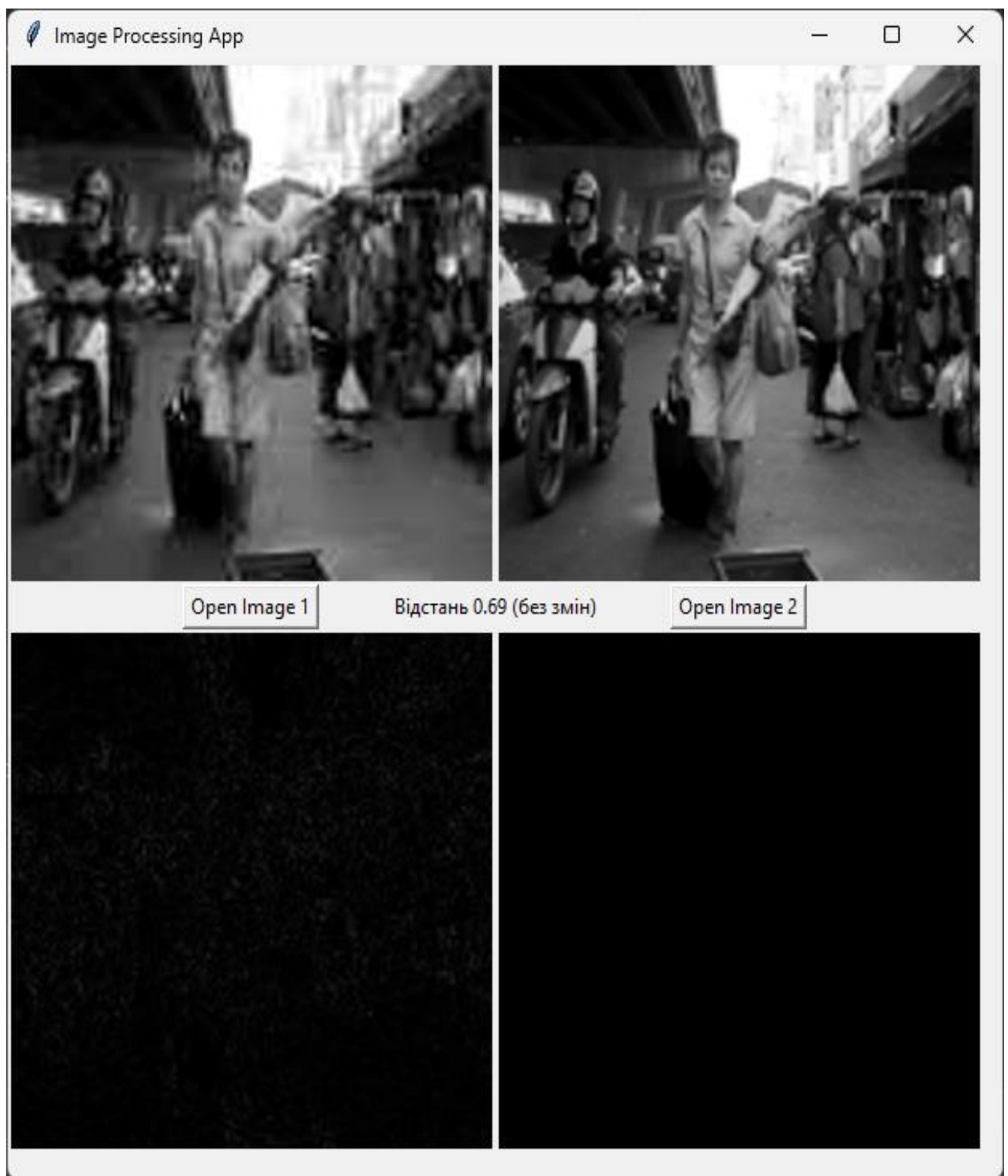


Рисунок 3.16 – Результат програми. Коректний результат для зображення без структурних змін

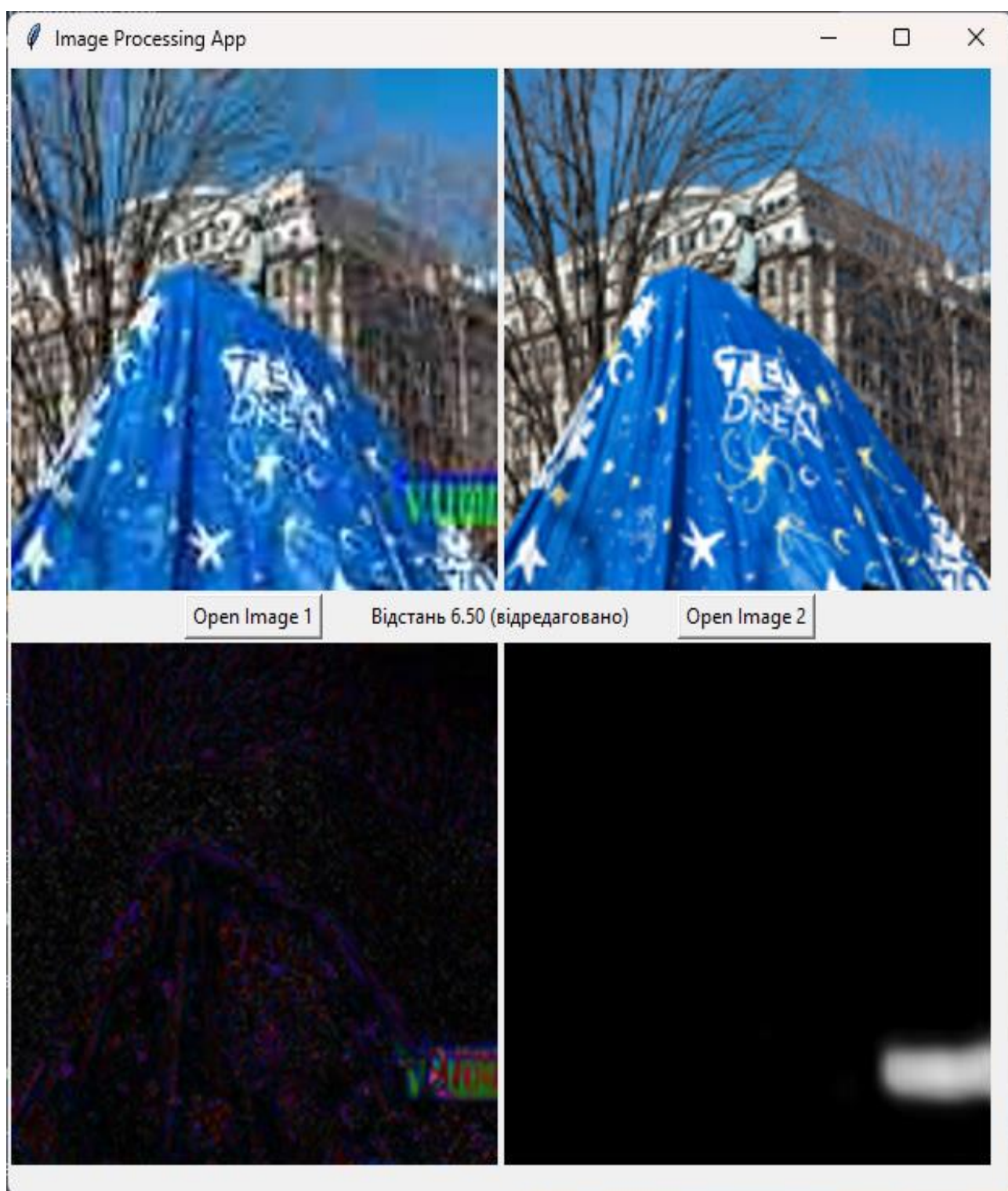


Рисунок 3.17 – Результат програми. Коректний результат для зображення з структурними зміна

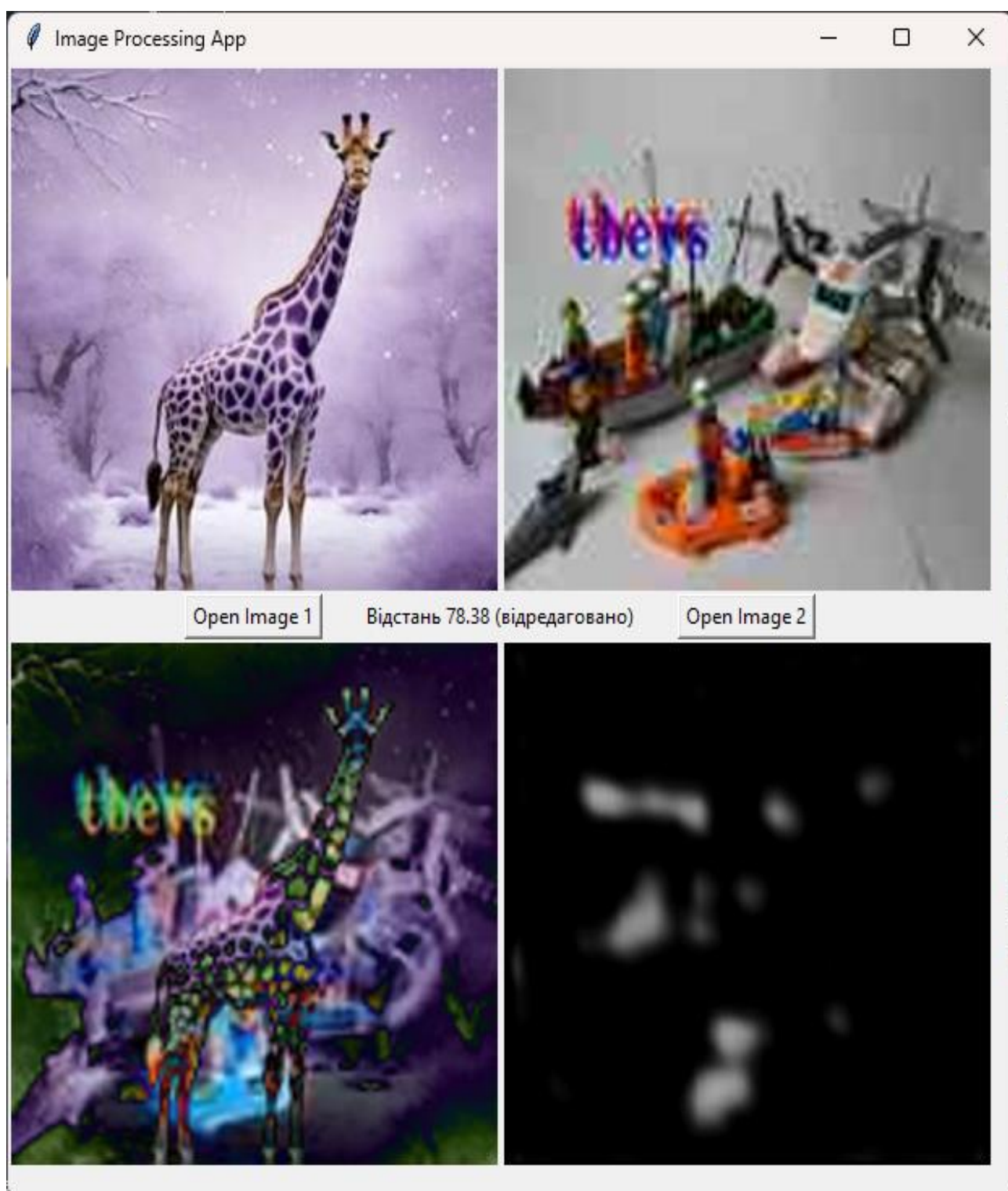


Рисунок 3.18 – Результат програми. Коректно класифіковане, але некоректно виділена область змін через значні відмінності

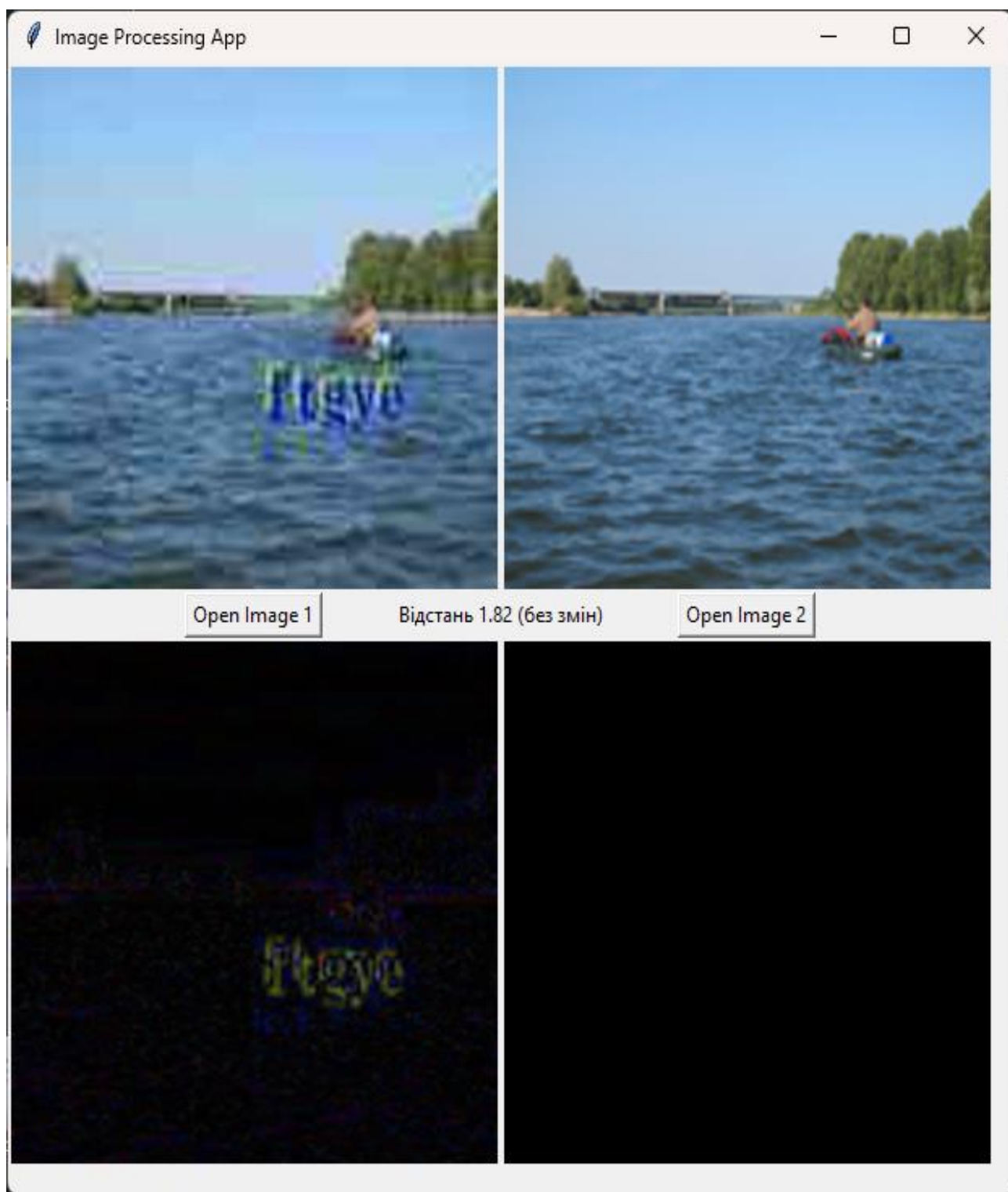


Рисунок 3.19 – Результат програми. Некоректно класифіковане зображення з структурними змінами

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЄКТУ

Стартап – це нова компанія або проєкт, який націлений на розвиток та впровадження інноваційних ідей чи продуктів на ринку. Вони часто починаються з обмежених ресурсів та стають осередком для розвитку новаторських концепцій і технологій. Описаний проєкт націлений на розробку і впровадження інноваційного рішення для точного порівняння та аналізу зображень, надаючи ефективний інструмент для порівняння конкретних зображень або пошуку схожих зображень в сховищі даних.

4.1 Опис ідеї проєкту

В межах підпункту було проаналізовано і подано у вигляді таблиць:

- зміст ідеї;
- можливі напрямки застосування;
- основні вигоди для користувачів;
- відмінність від існуючих рішень.

Перші три пункти подані у вигляді таблиці (табл. 4.1) і дають уявлення про зміст ідеї та потенційних клієнтів.

Аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї;
- визначення попереднього кола конкурентів (проєктів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних

показників для ідеї власного проєкту та проєктів-конкурентів відповідно до визначеного вище переліку;

- проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають:
- гірші значення (W, слабкі);
- аналогічні (N, нейтральні) значення;
- кращі значення (S, сильні) (табл. 4.2).

Таблиця 4.1 – Опис ідеї стартап-проєкту

<i>Зміст ідеї</i>	<i>Напрямки застосування</i>	<i>Вигоди для користувачів</i>
Дана система нейронних мереж дозволяє порівнювати зображення й знаходити на них структурні відмінності ігноруючи при цьому артефакти від стиснення	Порівняння двох заданих зображень	Швидке виділення різних областей зображення
	Автоматичний пошук тотожних зображень в сховищі даних	Можливість швидкого напівавтоматизованого очищення сховища від дублюючих даних

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ n/n	<i>Техніко-економічні характеристики ідеї</i>	<i>Товари/концепції потенційних конкурентів</i>			W	N	S
		<i>Мій проєкт</i>	<i>Функція дублікатів зображень IOS</i>	<i>Засоби порівняння зображень онлайн</i>			
1.	Форма виконання	Мобільний застосунок та web-сайт	Частина системного додатку «Фотографії»	Web-сайт	+		

Закінчення табл. 4.2

№ n/n	Техніко- економічні характеристики ідеї	Товари/концепції потенційних конкурентів			W	N	S
		Мій проект	Функція дублікатів зображень IOS	Засоби порівняння зображень онлайн			
2.	Собівартість	Низька	Низька	Низька		+	
3.	Функціонал	Порівняння зображень ігноруючи артефакти стиснення, надання числового значення їх близькості	Пошук дублюючих зображень	Порівняння двох зображень й створення карти різниці й маски на її основі			+

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит ідеї проєкту

В межах даного підрозділу було проведено аудит технології, за допомогою якої можна реалізувати ідею проєкту (технології створення товару). Визначення технологічної здійсненності ідеї проєкту передбачає аналіз таких складових (табл. 4.3):

- за якою технологією буде виготовлено товар згідно ідеї проєкту?
- чи існують такі технології, чи їх потрібно розробити/додати?

– чи доступні такі технології авторам проекту?

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ n/n	Ідея проєкту	Технології реалізації	Наявність технологій	Доступність технологій
1	Створення системи нейронних мереж для порівняння зображень	Використання мови програмування Python	Наявні	Доступні
		Використання мови програмування Java	Відсутні	Недоступні
		Використання мови програмування C++	Відсутні	Недоступні
Обрана технологія реалізації ідеї проєкту: мова програмування Python.				

За результатами аналізу таблиці зроблено висновок щодо можливості технологічної реалізації проекту. Технологічним шляхом реалізації проекту було обрано такі технології, як Python через доступність та безкоштовність.

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (табл. 4.4).

За результатами аналізу таблиці 4.4 було зроблено висновок, що ринок є привабливим для входження.

Надалі були визначені потенційні групи клієнтів, їх характеристики та сформовано орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проєкту

<i>№ п/п</i>	<i>Показники стану ринку (найменування)</i>	<i>Характеристика</i>
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум. од	20000000000
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі, %	30%

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проєкту

<i>Потреба, що формує ринок</i>	<i>Цільова аудиторія</i>	<i>Відмінності у поведінці різних потенційних цільових груп клієнтів</i>	<i>Вимоги споживачів до товару</i>
Порівняння зображень що стиснуті з втратами	Будь-яка людина	Віддають перевагу онлайн сервісам доступних без встановлення	Простота використання, доступність, наочність
Пошук схожих зображень в сховищі пристрою	Користувачі смартфонів	Цікавить точність схожості дублів, простота використання	Швидкість порівняння, доступність

Після визначення потенційних груп клієнтів було проведено аналіз ринкового середовища: складено таблиці факторів, що сприяють ринковому впровадженню проєкту, та факторів, що йому перешкоджають (табл. 4.6, 4.7).

Таблиця 4.6 – Фактори загроз

<i>№ п/п</i>	<i>Фактор</i>	<i>Зміст загрози</i>	<i>Можлива реакція компанії</i>
1	Конкуренція	Вихід на ринок продуктів з кращими характеристиками	Покращення точності порівняння за допомогою нових типів нейронних мереж, оптимізація швидкості порівняння.
2	Зміна потреб користувачів	Користувачам необхідний сервіс з більшим/новим функціоналом	Проектування гнучкої архітектури, що дасть змогу розширювати функціонал в разі потреби.

Таблиця 4.7 – Фактори можливостей

<i>№ п/п</i>	<i>Фактор</i>	<i>Зміст можливості</i>	<i>Можлива реакція компанії</i>
1	Поява нових прискорювачів для навчання ШІ	Зменшення ціни навчання нейромережі, можливість покращити наявну мережу	Тренування нових типів нейромережі на більшому та різноманітнішому датасеті.
2	Поява нових типів нейромереж	Поява нових типів нейромереж для обробки зображень, що демонструватимуть кращі результати, ніж наявні в даний момент	Додавання мережі нового типу в архітектуру системи нейромереж як додаткову можливість

Надалі було проведено аналіз пропозиції: визначили загальні риси конкуренції на ринку (табл. 4.8).

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

<i>Особливості конкурентного середовища</i>	<i>В чому проявляється дана характеристика</i>	<i>Вплив на діяльність підприємства</i>
1. Тип конкуренції: чиста	Існує величезна кількість конкурентів на ринку.	Якісно провести рекламу.
2. За рівнем конкурентної боротьби: міжнародний	Компанії-конкуренти з різних країн	Створити основу ПП таким чином, щоб можна було легко переробити даний ПП для використання у галузях інших країн.
3. За галузевою ознакою: міжгалузева	Продукт може використовуватись для різних галузей	Постійне вдосконалення продукту, що не має прив'язки до сфери
4. Конкуренція за видами товарів: товарно-видова	Конкуренція між видами ПП, їх особливостями.	Створити ПП, враховуючи досвід конкурентів
5. За характером конкурентних переваг: нецінова	Використання нейронних мереж дає якісно кращі результати ніж карти різниці й методи на їх основі	Удосконалення моделі. Застосування передових технологій для кращих результатів, ніж в конкурентів.
6. За інтенсивністю: марочна	Популярність бренду дозволяє новим клієнтам знаходити даний інструмент	Реклама, участь у конференціях, семінарах, співпраця з іншими компаніями.

Було проведено аналіз конкуренції у галузі за моделлю М. Портера (табл. 4.9).

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

	<i>Прямі конкуренти в галузі</i>	<i>Потенційні конкуренти</i>	<i>Постачальники</i>	<i>Клієнти</i>	<i>Товари замітники</i>
<i>Складові аналізу</i>	<i>Навести перелік прямих конкурентів</i>	<i>Визначити бар'єри входження в ринок</i>	<i>Визначити фактори сили постачальників</i>	<i>Визначити фактори сили споживачів</i>	<i>Фактори загроз з боку заміників</i>
	Diffchecker, Apple	Наявність вже існуючих рішень	—	Контроль якості продукту	Наявність більш широкого функціоналу, зручнішого інтерфейсу та авторитет
Висновки:	Досить інтенсивна конкурентна боротьба з іншими гравцями	Є можливість виходу на ринок, але є і конкуренти	—	Клієнти диктують умови роботи на ринку: зручний інтерфейс	Необхідно випускати ПЗ не гірше, ніж у конкурентів та розширяти функціонал

За результатами аналізу було зроблено висновок про можливість роботи на ринку з огляду на конкурентну ситуацію.

Цей висновок був врахований при формулюванні переліку факторів конкурентоспроможності у наступному пункті. На основі аналізу конкуренції, проведеного в таблиці, а також із урахуванням характеристик ідеї проєкту (табл. 4.2), вимог споживачів до товару (табл. 4.5) та факторів маркетингового середовища (табл. 4.6, 4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності.

Аналіз оформлено у (табл. 4.10).

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

<i>№ n/n</i>	<i>Фактор конкурентоспроможності</i>	<i>Обґрунтування</i>
1	Ціна	Один із факторів для вибору продукту клієнтом.
2	Якість	Один із факторів для вибору продукту клієнтом.
3	Зручність роботи з ПП	Один з факторів популярності ПП

За визначеними факторами конкурентоспроможності (табл. 4.10) проведено аналіз сильних та слабких сторін стартап-проєкту (табл. 4.11).

Фінальним етапом ринкового аналізу можливостей впровадження проєкту є складання SWOT-аналізу (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 4.11).

Перелік ринкових загроз та ринкових можливостей було складено на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проєкту

№ n/n	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			-3	-2	-1	0	1	2	3
1	Ціна	15					*		
2	Якість	20		*					
3	Зручність роботи з програмою	15				*			

Таблиця 4.12 – SWOT-аналіз стартап-проєкту

<i>Сильні сторони:</i> Якість результату, простота використання	<i>Слабкі сторони:</i> Дуже насичений ринок, вузька спеціалізація
<i>Можливості:</i> насичення ринку новим методом порівняння; різноманітна клієнтура, вдосконалення системи	<i>Загрози:</i> Конкуренція

На основі SWOT-аналізу було розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проєкту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проєкти конкурентів, що можуть бути виведені на ринок.

Визначені альтернативи були проаналізовані з точки зору строків та ймовірності отримання ресурсів (табл. 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проєкту

<i>№ n/n</i>	<i>Альтернатива ринкової поведінки</i>	<i>Ймовірність отримання ресурсів</i>	<i>Строки реалізації</i>
1	PR, просування бренду	50%	12 місяців
2	Перехід на безкоштовне розповсюдження	75%	6 місяців
3	Партнерство для об'єднання продукції	65%	3 місяці

Після аналізу було обрано альтернативу №2.

4.3 Аналіз ринкової стратегії проєкту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: було проведено опис цільових груп потенційних споживачів (табл. 4.14).

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

<i>№ n/n</i>	<i>Опис профілю цільової групи потенційних клієнтів</i>	<i>Готовність споживачів сприйняти продукт</i>	<i>Орієнтовний попит в межах цільової групи</i>	<i>Інтенсивність конкуренції в сегменті</i>	<i>Простота входу у сегмент</i>
1	Користувачі смартфонів	Висока	Середній	Сильна	Просто

Закінчення табл. 4.14

<i>№ n/n</i>	<i>Опис профілю цільової групи потенційних клієнтів</i>	<i>Готовність споживачів сприйняти продукт</i>	<i>Орієнтовн ий попит в межах цільової групи</i>	<i>Інтенсивніс ть конкуренції в сегменті</i>	<i>Просто та входу у сегмент</i>
2	Розробники застосунків	Середня: велика конкуренція і наявність власних відділів розробки	Середній	Слабка	Середня
3	Інтернет- аудиторія	Висока	Низький	Сильна	Просто
Які цільові групи обрано: 1,2,3					

За результатами аналізу потенційних груп споживачів було обрано цільові групи, для яких буде запропоновано даний товар, та визначено стратегію охоплення ринку - стратегію диференційованого маркетингу (компанія працює з декількома сегментами).

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (табл. 4.15).

Наступним кроком обрано стратегію конкурентної поведінки (табл. 4.16).

Таблиця 4.15 – Визначення базової стратегії розвитку

<i>№ n/ n</i>	<i>Обрана альтернатива розвитку проєкту</i>	<i>Стратегія охоплення ринку</i>	<i>Ключові конкурентоспроможні і позиції відповідно до обраної альтернативи</i>	<i>Базова стратегія розвитку</i>
1	Постійне оновлення і покращення продукту	Ринкове позиціонування на індивідуальних користувачів	Швидкодія, якість результатів	Концентрований маркетинг

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

<i>№ n/n</i>	<i>Чи є проєкт «першопроходцем» на ринку?</i>	<i>Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?</i>	<i>Чи буде компанія копіювати основні характеристики товару конкурента, і які?</i>	<i>Стратегія конкурентної поведінки</i>
1	Ні	Компанія буде і шукати нових, і забирати існуючих у конкурентів споживачів	Так, дизайн інтерфейсу для простоти використання	Зайняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (табл. 4.5), а також в залежності від обраної базової стратегії розвитку (табл. 4.15) та стратегії конкурентної поведінки (табл. 4.16) розроблено стратегію позиціонування (табл. 4.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 – Визначення стратегії позиціонування

<i>№ n/n</i>	<i>Вимоги до товару цільової аудиторії</i>	<i>Базова стратегія розвитку</i>	<i>Ключові конкурентоспроможні позиції власного стартап проекту</i>	<i>Вибір асоціацій, які мають сформувати комплексну позицію власного проекту</i>
1	Якість результату, простота використання, зручний інтерфейс	Стратегія диференціації	Позиція на основі порівняння фірми з товарами конкурентів; Відмінні особливості споживача	Економія часу; Зручність застосування; та точність результату

Результатом виконання підрозділу стала узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап-компанії на ринку.

4.4 Розроблення маркетингової програми стартап-проєкту

Сформовано маркетингову концепцію товару, який отримає споживач. Для цього підсумовано результати попереднього аналізу конкурентоспроможності товару (табл. 4.18). Концепція товару – письмовий опис фізичних та інших характеристик товару, які сприймаються споживачем, і набору вигод, які він обіцяє певній групі споживачів.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

<i>№ n/ n</i>	<i>Потреба</i>	<i>Вигода, яку пропонує товар</i>	<i>Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)</i>
1	Якість результату	Порівняння враховує наявність артефактів стиснення.	Необхідно покращити порівняння зображень низької якості, щоб артефакти стиснення не впливали на результат
2	Зручність застосування	Нативна підтримка мови програмування Python	Розробка зручного прикладного програмного інтерфейсу
3	Практичність та точність результату	Користувач отримує точні (з малою похибкою розбіжності) результати порівняння.	Користувач на виході роботи ПП отримує результат порівняння, котрий відповідає необхідним показникам варіативності та точності.

Розроблено трирівневу маркетингову модель товару: уточняється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 4.19).

1-й рівень. При формуванні задуму товару вирішується питання щодо того, засобом вирішення якої потреби і / або проблеми буде даний товар, яка його основна вигода. Дане питання безпосередньо пов'язаний з формуванням технічного завдання в процесі розробки конструкторської документації на виріб.

2-й рівень. Цей рівень являє рішення того, як буде реалізований товар в реальному/ включає в себе якість, властивості, дизайн, упаковку, ціну.

3-й рівень Товар з підкріпленням (супроводом) – додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін).

Таблиця 4.19 – Опис трьох рівнів моделі товару

<i>Рівні товару</i>	<i>Сутність та складові</i>		
I. Товар за задумом	Генерація палітри кольорів зображення за допомогою інтелектуальних систем		
II. Товар реальному виконанні	<i>Властивості/характеристики</i>	<i>М/Нм</i>	<i>Вр/Тх/Тл/Е/Ор</i>
	1. Індивідуальний підхід.	1. Нм	1. Технологічна
	2. Низька ціна.	2. Нм	2. Економічна
	3. Простота у використанні.	3. Нм	3. Технологічна
	Якість: внутрішнє тестування		
	Пакування: відсутнє		
	Марка: PixelVista		

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проєкту), яке передбачає

аналіз ціни на товари-аналоги або товари субститутути, а також аналіз рівня доходів цільової групи споживачів (табл. 4.20). Аналіз проведено експертним методом.

Таблиця 4.20 – Визначення меж встановлення ціни

<i>№ n/n</i>	<i>Рівень цін на товари- замінники</i>	<i>Рівень цін на товари- аналоги</i>	<i>Рівень доходів цільової групи споживачів</i>	<i>Верхня та нижня межі встановлення ціни на товар/послугу</i>
1	Умовно- безкоштовні	Умовно- безкоштовні	Середній	0\$-10\$

Наступним кроком є визначення оптимальної системи збуту, в межах якого було прийняте рішення (табл. 4.21)

Таблиця 4.21 – Формування системи збуту

<i>№ n/n</i>	<i>Специфіка закупівельної поведінки цільових клієнтів</i>	<i>Функції збуту, які має виконувати постачальник товару</i>	<i>Глибина каналу збуту</i>	<i>Оптимальна система збуту</i>
1	Канал нульового рівня	Продаж	0 (напрямую)	Власна

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Таблиця 4.22 – Концепція маркетингових комунікацій

<i>№ п/ п</i>	<i>Специфіка поведінки цільових клієнтів</i>	<i>Канали комунікацій, якими користують ся цілові клієнти</i>	<i>Ключові позиції, обрані для позиціонуван ня</i>	<i>Завдання рекламного повідом- лення</i>	<i>Концепція рекламного звернення</i>
1	Інтернет-аудиторія, звикла до користування пошуковими системами	Інтернет	Низька ціна, простота використання, якість результату	Показати переваги рішення над конкурентами, виділити ключові особливості	Створення сайту продукту, розповсюдження інформації про продукт, SEO оптимізація для виводу його в пошуку.

Визначено, що придбання продукту буде проводитись через мережу Інтернет. Розповсюдження інформації про продукт буде проводитись виключно через Інтернет, адже аудиторія даного продукту активно користується всесвітньою мережею.

Результатом підрозділу стала ринкова (маркетингова) програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого впроваджено проєкт, та відповідну обрану альтернативу ринкової поведінки.

Висновки до розділу 4

В даному розділі проведено підготовчий аналіз для впровадження розробленої системи нейронних мереж для порівняння зображень низької якості в якості стартап проєкту. Досліджено подібні конкурентні системи, встановлено сильні та слабкі сторони системи в порівнянні з ними. Також було досліджено можливі шляхи розповсюдження продукту та його ймовірну аудиторію, рівень доходів та ймовірну ціну продукту, що розробляється.

Проведено аналіз потенційних ризиків і можливостей, а також розраховані основні фінансово-економічні показники проєкту. Отримані результати кажуть про те, що реалізація проєкту є доцільною. Було визначено сильні сторони проєкту: зручність у використанні, ціна, якість результатів. Серед слабких варто виділити повільне навчання та . Варто відмітити можливість реклами продукту на онлайн ресурсах із зазначенням сильних сторін проєкту.

ВИСНОВКИ

В цій роботі проведено порівняння існуючих рішень для порівняння зображень, описано їх переваги та недоліки, наведено основи роботи сіамських нейронних мереж і їх модифікацій.

Для досягнення мети роботи запропоновано кроки описані нижче.

1. Навчити сіамську мережу класифікувати пари зображень з структурними відмінностями та лише артефактами стиснення. Для отримання найкращого результату і зменшення впливу дисбалансу похибок першого і другого родів додатково запропоновано:

- 1.1. використати датасет з співвідношенням класів 9:1 замість звичного 1:1;
- 1.2. модифікувати функцію втрат (1) в (3).

Це дало змогу досягти точності 89.7%.

2. Навчити нейронну мережу сегментувати змінні області зображення використовуючи карту різниці. Для покращення результату та зменшення впливу форми змін на результат запропоновано:

- 2.1. звузити область допустимих вхідних даних, відсіявши пари зображення без структурних змін;
- 2.2. використати архітектуру «encoder-concatenator-decoder» для використання даних з сіамської нейронної мережі.

Це дало змогу досягти значення функції втрат попіксельної бінарної ентропії 0,024.

Проведено перевірку практичної можливості класифікації типу змін в парі зображень та оцінку різних моделей нейронних мереж на зменшеному датасеті. Навчено сіамську нейронну мережу для визначення наявності структурних змін в парі зображень та згорткову нейронну мережу-сегментатор з архітектурою «encoder-concatenator-decoder» для виділення зміненої області. Об'єднано навчені моделі в систему з двох нейронних мереж й написано програму з графічним інтерфейсом, що здійснює порівняння двох зображень та демонструє

результати роботи, використовуючи сіамську мережу для відсіювання пар зображень без змін перед застосуванням сегментатора.

Для покращення програмного продукту можна доповнити систему нейронною мережею, що буде класифікувати результат відображення в простір ознак, отриманий з сіамської мережі. Другим напрямом подальшої роботи є створення сегментатора для випадків, коли між зображенням є великі відмінності, на яких поточна модель сегментатору продемонструвала некоректні результати. Третій потенційний шлях покращення – додавання вибору алгоритму стиснення і, можливо, автоматичне визначення алгоритму, використання якого спричинило появу артефактів.

ПЕРЕЛІК ПОСИЛАНЬ

1. S. Chopra, R. Hadsell and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 2005, pp. 539-546 vol. 1, doi: 10.1109/CVPR.2005.202.
2. Melekhov I., Kannala J., Rahtu E. Siamese network features for image matching. *23rd international conference on pattern recognition (ICPR)*, Cancun, 4–8 December 2016. 2016. P. 378–383. DOI: <https://doi.org/10.1109/ICPR.2016.7899663>.
3. F. Rahman, B. Vasu, J. V. Cor, J. Kerekes and A. Savakis, "SIAMESE NETWORK WITH MULTI-LEVEL FEATURES FOR PATCH-BASED CHANGE DETECTION IN SATELLITE IMAGERY," *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Anaheim, CA, USA, 2018, pp. 958-962, doi: 10.1109/GlobalSIP.2018.8646512.
4. Stefanovic S. # 019 siamese network in pytorch with application to face similarity - master data science 07.11.2021. Master Data Science. URL: <https://datahacker.rs/019-siamese-network-in-pytorch-with-application-to-face-similarity> (дата звернення: 11.10.2023).
5. El Adoui, Mohammed, Sidi Ahmed Mahmoudi, Mohamed Amine Larhman, and Mohammed Benjelloun. 2019. "MRI Breast Tumor Segmentation Using Different Encoder and Decoder CNN Architectures" *Computers* 8, no. 3: 52. DOI: <https://doi.org/10.3390/computers8030052>
6. Терейковський, І. А. Штучні нейронні мережі: базові положення : навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою «Системне програмування та спеціалізовані комп'ютерні системи» спеціальності 123 Комп'ютерна інженерія. І. А. Терейковський, Д. А. Бушуєв, Л. О. Терейковська ; КПІ ім. Ігоря Сікорського). Київ : КПІ

- ім. Ігоря Сікорського, 2022. 123 с. URL: <https://ela.kpi.ua/handle/123456789/50135>.
7. State-of-the-art in artificial neural network applications: a survey . I. A. Oludare et al. *Heliyon*. 2018. Vol. 4, no. 11. e00938. DOI: <https://doi.org/10.1016/j.heliyon.2018.e00938>.
 8. Class-balanced siamese neural networks S. Berlemont et al. *Neurocomputing*. 2018. Vol. 273. P. 47–56. DOI: <https://doi.org/10.1016/j.neucom.2017.07.060>.
 9. Filtering noise from images with wavelet transforms / J. B. Weaver et al. *Magnetic resonance in medicine*. 1990. Vol. 21, no. 2. P. 288–295. DOI: <https://doi.org/10.1002/mrm.1910210213>.
 10. Барченко К., Білошкурський С., Гармаш В. Аналіз методів фільтрації зображень. *Вісник Хмельницького національного університету. Серія: «Технічні науки»*. 2012. № 4. С.79–86. URL: http://journals.khnu.km.ua/vestnik/pdf/tech/2012_4/18bar.pdf.

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
# файл train.py
import argparse
import torch
import torchvision.transforms.functional as FP
import random
import numpy as np
import time
import matplotlib.pyplot as plt
import csv
import torch.nn as nn
from net import *
from contrastive import ContrastiveLoss
from torch.autograd import Variable
from torchvision import transforms
import os
from PIL import Image
criterion_bce = nn.BCELoss()
global loss_max, loss_current
loss_max = 10000
loss_current = 10000

class Dataset(object):

    def __init__(self, d, x, label):
        self.size = label.shape[0]
        self.d = torch.from_numpy(d)
        self.x = torch.from_numpy(x)
        self.label = torch.from_numpy(label)

    def __getitem__(self, index):
        return (self.d[index],
                self.x[index],
                self.label[index])

    def __len__(self):
        return self.size

def create_pairs(diffs, data, labels, model_s, use_cude = False):
    x0_data = []
    x1_data = []
    df_data = []

    for i in range(len(diffs)):
        d = np.transpose(diffs[i], (2, 0, 1))
        df_data.append((d / 255.0))

    for i in range(0, len(data), 2):
        d = (data[i], data[i + 1])
        d0 = np.transpose(d[0], (2, 0, 1))
        d1 = np.transpose(d[1], (2, 0, 1))

        x0_data.append(d0 / 255.0)
        x1_data.append(d1 / 255.0)

    df_data = np.array(df_data, dtype=np.float32)
    x0_data = np.array(x0_data, dtype=np.float32)
    x1_data = np.array(x1_data, dtype=np.float32)
    labels = np.array(labels, dtype=np.float32)
```

```

x_res = []
for i in range(len(x0_data)):
    x0_d = torch.from_numpy(x0_data[i])
    x1_d = torch.from_numpy(x1_data[i])
    x0_d, x1_d = Variable(x0_d), Variable(x1_d)

    if use_cude:
        x0_d, x1_d = x0_d.cuda(), x1_d.cuda()
    x0_r, x1_r = model_s(x0_d, x1_d)
    res = torch.concat((x0_r[0], x1_r[0]), -1)
    x_res.append(res.cpu().detach().numpy())
    if (i + 1) % 100 == 0:
        print(f'{i + 1} pictures is converted')

return np.array(df_data), np.array(x_res), np.array(labels)

def create_iterator(diffs, data, label, model_s, use_cude = False):
    d, x, label = create_pairs(diffs, data, label, model_s, use_cude)
    ret = Dataset(d, x, label)
    return ret

def load_image(input_image):
    img_path = os.path.join(dataset_root_folder, input_image)
    img = Image.open(img_path)
    return np.asarray(img)

def im_save(img, text=None):
    from datetime import datetime
    import calendar
    npimg = img.numpy()
    plt.axis("off")
    if text:
        plt.text(75, 8, text, style='italic', fontweight='bold',
                bbox={'facecolor': 'white', 'alpha': 0.8, 'pad': 10})

    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.savefig(f'test-
                {calendar.timegm(datetime.now().timetuple())}.png')
    plt.cla()

dataset_root_folder = 'D:\\Mag\\dataset_2\\train_0'
test_file = f'{dataset_root_folder}\\test.txt'
train_file = f'{dataset_root_folder}\\train.txt'

def load_dataset(path):
    labels = []
    final_features = []
    diffs = []
    with open(path) as csvfile:
        read_csv = csv.reader(csvfile, delimiter=',')
        i = 0

        for row in read_csv:
            try:
                i += 1
                img1 = row[0].strip()
                img2 = row[1].strip()
                mask = row[2].strip()

```

```

        diff = row[3].strip()

        img1_ar = load_image(img1)
        img2_ar = load_image(img2)
        mask_ar = load_image(mask)
        diff_ar = load_image(diff)

        final_features.append(img1_ar)
        final_features.append(img2_ar)
        labels.append(mask_ar)
        diffs.append(diff_ar)
    except:
        print(f'Error picture {row} is not load')
    if i % 100 == 0:
        print(f'{i} pictures is load')

    return np.array(diffs), np.array(final_features), np.array(labels)

def main(model_name, iter):
    parser = argparse.ArgumentParser()
    parser.add_argument('--epoch-start', '-es', type=int, default=1,
                        help='Number of sweeps over the dataset to train')
    parser.add_argument('--epoch', '-e', type=int, default=1,
                        help='Number of sweeps over the dataset to train')
    parser.add_argument('--batchsize', '-b', type=int, default=32,
                        help='Number of images in each mini-batch')
    parser.add_argument('--no-cuda', action='store_true',
                        default=False,
                        help='disables CUDA training')
    parser.add_argument('--model', '-m', default='', # model-second-epoch-30.pth
                        help='Give a model to test or train')
    parser.add_argument('--train-plot', action='store_true',
                        default=True,
                        help='Plot train loss')
    parser.add_argument('--save-images', '-s', action='store_true',
                        default=True,
                        help='Save test images')
    parser.add_argument('--train', '-t', action='store_false',
                        default=True,
                        help='train')

    args = parser.parse_args()
    args.model = model_name
    args.cuda = not args.no_cuda and torch.cuda.is_available()
    print("Args: %s" % args)
    train_train_diffs, train_data, train_labels = None, None, None

    if args.train:
        print(f'Train file: {train_file}')
        train_diffs, train_data, train_labels = load_dataset(train_file)
    print(f'Test file: {test_file}')
    test_diffs, test_data, test_labels = load_dataset(test_file)

    first_model_state = torch.load('D:\\Mag\\v4_89\\model-best.pth')
    first_model = SiameseNetworkFirst_v4()
    first_model.load_state_dict(first_model_state)

    if args.cuda:

```



```

        first_model.cuda()

first_model.eval()

train_iter = None
if args.train:
    train_iter = create_iterator(
        train_diffs,
        train_data,
        train_labels,
        first_model,
        args.cuda)

test_iter = create_iterator(
    test_diffs,
    test_data,
    test_labels,
    first_model,
    args.cuda)

# model
model = CoSegNet()
if args.cuda:
    model.cuda()

# Loss and Optimizer
optimizer = torch.optim.Adam(model.parameters())

kwargs = {'num_workers': 1, 'pin_memory': True} if args.cuda else {}
train_loader = None
if args.train:
    train_loader = torch.utils.data.DataLoader(
        train_iter,
        batch_size=args.batchsize, shuffle=True, **kwargs)

test_loader = torch.utils.data.DataLoader(
    test_iter,
    batch_size=1, shuffle=True, **kwargs)

def train(epoch):
    train_loss = []
    model.train()
    start = time.time()
    start_epoch = time.time()
    total_loss = 0
    for batch_idx, (d, x, labels) in enumerate(train_loader):
        batch_idx += 1

        if args.cuda:
            d, x, labels = d.cuda(), x.cuda(), labels.cuda()
            d, x, labels = Variable(d), Variable(x), Variable(labels)
            pred_mask = model(d, x)

        pred_mask = torch.transpose(pred_mask, 0, 1)
        # Loss 1
        # Pixel-wise Binary Cross Entropy Loss
        loss_final = criterion_bce(pred_mask, labels.unsqueeze(0))

        loss_final.backward()
        optimizer.step()

        loss = loss_final.item()

```



```

if len(args.model) != 0:
    saved_model = torch.load(args.model)
    model.load_state_dict(saved_model)
    print('Load model')
    if args.cuda:
        model.cuda()
if args.train:
    train_loss = []
    for epoch in range(args.epoch_start, args.epoch + 1):
        train_loss.append(train(epoch))
    del train_iter, train_data, train_labels, train_loader

    if args.train_plot:
        lenth = len(train_loss)
        step = 1 if lenth < 10 else lenth // 10
        lenth = lenth if lenth % step == 0 else lenth + step
        print(lenth)
        plt.gca().cla()
        plt.plot(train_loss, label="train loss")
        plt.xticks(range(-1, lenth + 1, step),
                    range(args.epoch_start - 1, args.epoch_start + lenth + 1,
                          step))
        plt.legend()
        plt.draw()
        plt.savefig('train_loss.png')
        plt.gca().clear()
    loss_res = test(model)
    del test_iter, test_data, test_labels, test_loader

    return loss_res

k = -1
if __name__ == '__main__':
    global_loss = []
    is_first = True
    for e in range(5):
        with open('result.txt', 'a') as f:
            print(f'=====|||||EPOCH
                  {e}|||||=====', file=f)
        for q in range(0, 31):
            k += 1
            print(f"corects_current is {loss_current}")
            if loss_current < loss_max:
                loss_max = loss_current
                if os.path.exists('model-best.pth'):
                    os.remove('model-best.pth')
                os.system('copy    model-segment-epoch-1.pth    model-
best.pth')
            if not is_first:
                if os.path.exists('last.pth'):
                    os.remove('last.pth')
                os.rename('model-segment-epoch-1.pth', 'last.pth')
            is_first = False
            dataset_root_folder = f'D:\\Mag\\dataset_2\\train_{q}'
            test_file = f"{dataset_root_folder}\\test.txt"
            train_file = f"{dataset_root_folder}\\train.txt"

            is_first = False
            if q == 0 and e == 0:
                global_loss.append(main('', k))
                continue

```

```

        global_loss.append(main('last.pth', k))

    lenth = len(global_loss)
    step = 1 if lenth < 10 else lenth // 10
    lenth = lenth if lenth % step == 0 else lenth + step
    print(lenth)
    plt.gca().cla()
    plt.plot(global_loss, label="train loss")
    plt.xticks(range(-1, lenth + 1, step), range(1 - 1, 1 +
        lenth + 1, step))
    plt.legend()
    plt.draw()
    plt.savefig('global_train_loss.png')
    plt.gca().clear()

# файл net.py
from collections import OrderedDict
import torch
import torch.nn as nn
import torchvision.models as models
import torch.nn.functional as F

class VGGFirst:
    def __init__(self):
        self.model =
            models.vgg16(weights=models.VGG16_Weights.IMAGENET1K_V1)
        self.__call__ = self.forward

    def pre_forward_once(self, x):
        x = torch.unsqueeze(x, 0)
        x = self.model(x)
        return x

    def eval(self):
        pass

    def __call__(self, x1, x2):
        return self.forward(x1, x2)

    def forward(self, input1, input2):
        output1 = self.pre_forward_once(input1)
        output2 = self.pre_forward_once(input2)
        return output1, output2

class SiameseNetworkFirst_v4(nn.Module):
    def __init__(self):
        super(SiameseNetworkFirst_v4, self).__init__()
        self.features = nn.Sequential(
            # conv1
            nn.Conv2d(3, 2, 3, padding=1),
            nn.AvgPool2d(2, stride=2)
        )

        self.classifier = nn.Sequential(
            nn.Linear(64 * 64 * 2, 64 * 64),
            nn.ReLU(),
            nn.Linear(64 * 64, 64 * 64 // 2),
            nn.ReLU(),
            nn.Linear(64 * 64 // 2, 1024)
        )

        self.to_euler = nn.Sequential(
            nn.Linear(1024, 8)
        )

```

```

def pre_forward_once(self, x):
    for layer in self.features:
        if isinstance(layer, nn.MaxPool2d):
            x, location = layer(x)
        else:
            x = layer(x)

    x = x.view(1, -1)
    x = self.classifier(x)
    return x

def forward_once(self, x):
    x = self.pre_forward_once(x)

    # x = x.view(x.size()[0], -1)
    # x = self.to_euler(x)
    return x

def forward(self, input1, input2):
    output1 = self.forward_once(input1)
    output2 = self.forward_once(input2)
    return output1, output2

def pre_forward(self, input1, input2):
    output1 = self.pre_forward_once(input1)
    output2 = self.pre_forward_once(input2)
    return output1, output2

class CoSegNet(nn.Module):
    def __init__(self, input_channels=2, output_channels=2):
        super(CoSegNet, self).__init__()

        self.input_channels = 3 + 2 # RGB + picture 1 and picture 2
        self.output_channels = output_channels # 2 = Foreground +
            Background

        # 128*128 => 64*64
        self.encoder1 = nn.Sequential(
            nn.ConvTranspose2d(3, 8, kernel_size=3, padding=1),
            nn.BatchNorm2d(8),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2)
        )
        # 64*64 => 32*32
        self.encoder2 = nn.Sequential(
            nn.ConvTranspose2d(8, 16, kernel_size=3, padding=1),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2)
        )
        # 32*32 => 16*16
        self.encoder3 = nn.Sequential(
            nn.ConvTranspose2d(16, 16, kernel_size=3, padding=1),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(2, stride=2)
        )

        self.linear_f = nn.Sequential(
            nn.Linear(1024 * 2, 16),

```

```

        nn.Sigmoid()
    )

    self.linear = nn.Sequential(
        nn.Linear(16 * (16 * 16) + 16, 16 * (16 * 16))
    )

    # Decoder
    # 16*16 => 32*32
    self.decoder1 = nn.Sequential(
        nn.Upsample(scale_factor=2, mode="nearest"),
        nn.ConvTranspose2d(16, 16, kernel_size=3, padding=1),
        nn.ConvTranspose2d(16, 16, kernel_size=3, padding=1),
        nn.BatchNorm2d(16),
        nn.ReLU()
    )
    # 32*32 => 64*64
    self.decoder2 = nn.Sequential(
        nn.Upsample(scale_factor=2, mode="nearest"),
        nn.ConvTranspose2d(16, 8, kernel_size=3, padding=1),
        nn.ConvTranspose2d(8, 8, kernel_size=3, padding=1),
        nn.BatchNorm2d(8),
        nn.ReLU()
    )
    # 64*64 => 128*128
    self.decoder3 = nn.Sequential(
        nn.Upsample(scale_factor=2, mode="nearest"),
        nn.ConvTranspose2d(8, 4, kernel_size=3, padding=1),
        nn.ConvTranspose2d(4, 1, kernel_size=3, padding=1),
        nn.BatchNorm2d(1),
        nn.Hardsigmoid()
    )

def forward(self, diff, features):
    # Decoding
    # 3 layers
    # diff = input

    size = diff.size()
    diff = self.encoder1(diff)
    diff = self.encoder2(diff)
    diff = self.encoder3(diff)
    enc_size = diff.size()
    diff = diff.view(size[0], -1)
    features = features.view(size[0], -1)
    features = self.linear_f(features)
    concat = torch.cat([diff, features], dim=1)

    concat = self.linear(concat)

    concat = concat.view(enc_size)

    # print('decoder3')
    features = self.decoder1(concat)
    features = self.decoder2(features)
    features = self.decoder3(features)

    return features

class SiameseMetricNet(nn.Module):
    def __init__(self):
        super(SiameseMetricNet, self).__init__()

```

```

        self.metricnet = nn.Sequential(
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Linear(128, 64)
        )

    def forward(self, feature):
        # Global Average Pooling (GAP)
        gap_feature_vector = F.avg_pool2d(feature,
            kernel_size=feature.size()[2:])
        gap_feature_vector = gap_feature_vector.squeeze()

        return self.metricnet(gap_feature_vector)

class SiameseDecisionNet(nn.Module):
    def __init__(self):
        super(SiameseDecisionNet, self).__init__()
        self.decisionnet = nn.Sequential(
            nn.Linear(128, 32),
            nn.ReLU(),
            nn.Linear(32, 1),
            nn.Sigmoid()
        )

    def forward(self, vector):
        return self.decisionnet(vector)

# файл Contrastive.py
import torch
import torch.nn
import torch.nn.functional as F

class ContrastiveLoss(torch.nn.Module):
    global_margin = 3.5
    def __init__(self, margin=global_margin):
        super(ContrastiveLoss, self).__init__()
        self.margin = margin

    def check_type_forward(self, in_types):
        assert len(in_types) == 3

        x0_type, x1_type, y_type = in_types
        assert x0_type.size() == x1_type.shape
        assert x1_type.size()[0] == y_type.shape[0]
        assert x1_type.size()[0] > 0
        assert x0_type.dim() == 2
        assert x1_type.dim() == 2
        assert y_type.dim() == 1

    def forward(self, x0, x1, y):
        self.check_type_forward((x0, x1, y))

        euclidean_distance = F.pairwise_distance(x0, x1, keepdim=True)

        loss_contrastive = torch.mean((1 - y) *
            torch.pow(euclidean_distance, 1.35) +
            y *
            torch.pow(torch.clamp(self.margin - euclidean_distance,
                min=0.0), 2))
        return loss_contrastive

```