МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

ШАМУРАТОВ ОЛЕКСІЙ ЮРІЙОВИЧ

Кваліфікаційна наукова
праця на правах рукопису
УДК 004.652

Методи та засоби опрацювання зображень для анімації статичних об’єктів

122 – комп’ютерні науки

Дисертація на здобуття наукового ступеня
доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____/О. Ю. Шамуратов/

Науковий керівник

Шаховська Наталія Богданівна

доктор технічних наук, професор

Львів – 2022

АНОТАЦІЯ

Шамуратов О. Ю. Методи та засоби опрацювання зображень для анімації статичних об’єктів. – Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 “Комп’ютерні науки”. – Національний університет «Львівська політехніка», Львів, 2022.

Зміст анотації. Дисертація присвячена побудові методів та засобів для створення анімації об’єктів на зображеннях та розробки мобільного додатку користувача. В сучасних мультимедійних продуктах, аплікаціях відео-індустрії та індустрії ігор дедалі частіше виникає потреба в анімуванні сцен чи певних об’єктів на основі одного зображення. Сьогодні анімування зображень усе частіше використовується в індустрії розваг і виконується за умови обмеження ресурсів – на мобільних пристроях, в режимі реального часу тощо.

У першому розділі «Аналіз алгоритмів пошуку об’єктів на зображеннях, їх класифікації та кластеризації, анімування статичних об’єктів» проаналізовано алгоритми пошуку об’єктів, а саме контурний аналіз та алгоритм штучної нейронної мережі. Також розглянуті методи кластеризації та класифікації даних: класифікація по найближчому середньому значенні; класифікація по відстані до найближчого сусіда; байєсівський підхід. Проаналізовано методи пошуку об’єктів на зображенні за допомогою контурного аналізу та штучних нейронних мереж. Розглянуто алгоритми перетворення зображень та існуючі підходи до створення анімації з статичних зображень.

У другому розділі «Математичні методи пошуку об’єктів на зображеннях, їх класифікації та кластеризації» було розглянуто методи виділення границь об’єктів та проведено аналіз їх ємнісної складності. Також сформовано загальну структуру згорткової нейронної мережі. Розроблено моделі класифікації зображень. Розроблено метод анімування статичних об’єктів та проведено оптимізацію ємнісної складності даного алгоритму.

У третьому розділі «Розроблення алгоритмів пошуку об’єктів на зображеннях, їх класифікації та кластеризації, анімування статичних об’єктів» було розроблено алгоритми пошуку об’єктів, кластеризації та класифікації,

алгоритм анімування статичних об’єктів. Усі алгоритми було зображено за допомогою блок-схем.

У четвертому розділі «Розроблення архітектури та апробація результатів» представлено розробку архітектури системи, обґрунтовано вибір технологій для реалізації системи. Також визначено функціонал системи та засоби, які забезпечать стабільну роботу системи. Представлено готовий додаток для створення анімації статичних об’єктів на зображені.

Основні наукові результати дисертації опубліковано в 8 працях, зокрема: три статті – у наукових фахових періодичних виданнях України; дві - у закордонних фахових періодичних виданнях; 3 публікації – у матеріалах міжнародних та всеукраїнських наукових, науково-технічних конференцій.

Ключові слова: методи кластеризації та класифікації, методи пошуку об’єктів на зображенні, згорткові нейронні мережі, лямбда-архітектура, анімування об’єктів на зображенні, афінні перетворення.

ABSTRACT

Shamuratov O. Methods and means of image processing for animation of static objects. - Qualifying scientific work on the rights of the manuscript. Dissertation for the degree of Doctor of Philosophy in the specialty 122 "Computer Science". - National University "Lviv Polytechnic", Lviv, 2022.

Abstract content. The dissertation is devoted to the construction of methods and tools for animating objects in images and developing a mobile user application. In modern multimedia products, video and gaming applications, there is an increasing need to animate scenes or specific objects based on a single image. Today, image animation is increasingly used in the entertainment industry and is subject to limited resources - on mobile devices, in real time and more.

The first section "Analysis of algorithms for finding objects in images, their classification and clustering, animation of static objects" analyzes the search algorithms for objects, namely contour analysis and algorithm of artificial neural network. Methods of clustering and classification of data are also considered: classification by the nearest average value; classification by distance to the nearest

neighbor; Bayesian approach. Methods of searching for objects in the image with the help of contour analysis and artificial neural networks are analyzed. Image conversion algorithms and existing approaches to creating animation from static images are considered.

The second section "Mathematical methods of finding objects in images, their classification and clustering" considered methods for distinguishing the boundaries of objects and analyzed their capacitive complexity. The general structure of the convolutional neural network is also formed. Image classification models have been developed. A method for animating static objects has been developed and the capacitive complexity of this algorithm has been optimized.

In the third section "Development of algorithms for finding objects in images, their classification and clustering, animation of static objects" were developed algorithms for searching objects, clustering and classification, algorithm for animating static objects. All algorithms were represented by flowcharts.

The fourth section "Architecture development and approbation of results" presents the development of system architecture, substantiates the choice of technologies for system implementation. The functionality of the system and the means that will ensure the stable operation of the system are also defined. A ready-made application for creating animations of static objects in the image is presented.

The main scientific results of the dissertation have been published in 8 works, in particular: three articles - in scientific professional periodicals of Ukraine; two - in foreign professional periodicals; 3 publications - in the materials of international and national scientific, scientific and technical conferences.

Keywords: methods of clustering and classification, methods of searching for objects in the image, convolutional neural networks, lambda architecture, animation of objects in the image, affine transformations.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у фахових виданнях України:

1. Шамуратов О. Ю., Шаховська Н. Б. Анімування об’єктів за допомогою афінних перетворень // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2021. – № 1 (293). – С. 73–77
2. Шамуратов О. Ю., Шаховська Н. Б. Алгоритми контурного аналізу зображень. Науковий вісник НЛТУ України. - 2019. - Т. 29, № 6. - С. 123-127
3. Шамуратов О. Ю. Метод кластеризації об’єктів на зображенні на основі вибору ознак // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2022. – № 3 (309). – С. 260–264

Статті у інших виданнях:

4. Shakhovska N., Shamuratov O. Method for clusters and classifying the objects in images // CEUR Workshop Proceedings. – 2021. – Vol. 3003: – P. 76–82
5. Basystiuk O., Shakhovska N., Bilynska* V., Syvokon O., Shamuratov O., Kuchkovskiy V. The developing of the system for automatic audio to text conversion // CEUR Workshop Proceedings. – 2021. – Vol. 2824. – P. 1–8.
6. Shamuratov O. Object recognition by convolutional neural network for multiple classes using lambda processes // ManažérskaInformatika – 2020. – № 3.

Матеріали конференцій:

7. O. Shamuratov, Y. Ryshkovets, N. Shakhovska and I. Kohut, "The methods for Contour Analysis of Images," *2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT)*, 2019, pp. 41-45, doi: 10.1109/STC-CSIT.2019.8929857.
8. Shakhovska N., Shamuratov* O. The structure of information systems for environmental monitoring // Комп’ютерні науки та інформаційні

технології : матеріали XI Міжнародної науково-технічної конференції CSIT 2016 (Львів, 6–10 вересня 2016 р.). – 2016. – С. 102–107

9. Шамуратов О. Ю. «Лямбда-архітектура згорткової нейронної мережі для пошуку об’єктів». X Міжнародна науково-практична конференція «INNOVATIONS AND PROSPECTS OF WORLD SCIENCE» 25-27 травня 2022 р., Ванкувер, Канада. – 2022. – С. 314–318

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

1. ШНМ – штучна нейронна мережа;
2. ЗНМ – згорткова нейронна мережа;
3. ПНМ – повнозв’язна нейронна мережа;
4. АП – афінні перетворення;
5. СЯЗ – середня яскравість зображення;
6. AWS – Amazon web services;
7. ReLU – rectified linear unit;
8. API – application programming interface;
9. HOG – histogram of oriented gradients.
10. ROI – region-of-interest.
11. ОПІ – область, що представляє інтерес.
12. PCW – Preserve-Curve-Warping.
13. СУБД – Система управління базами даних.
14. СД – сховище даних.
15. ТСД – тимчасове сховище даних.
16. AWPN – angle weighted pseudo normal.

ВСТУП

Актуальність теми. В сучасних мультимедійних продуктах, аплікаціях відео-індустрії та індустрії ігор дедалі частіше виникає потреба в анімуванні сцен чи певних об’єктів на основі одного зображення. Сьогодні анімування зображень усе частіше використовується в індустрії розваг і виконується за умови обмеження ресурсів – на мобільних пристроях, в режимі реального часу тощо. Об’єкти анімування можуть бути різної природи – від обличчя людей, не фізичних об’єктів (аватарів) до будь-чого, визначеного користувачем на зображенні. Під анімацією зображень розумітимемо зміну параметрів обраних об’єктів на зображенні з плином часу. В основі цього лежатиме той факт, що зорова система людини сприймає набір окремих зображень, що подаються з досить високою швидкістю, як безперервний рух.

Створення анімації з однієї картини або фотографії - не нова ідея. Наприклад, зазвичай можна побачити 2D-анімацію, де 2D-фігури людей або тварин на оригінальному зображенні переміщаються, фіксуючи 2D-фон. У простих випадках ці анімації можна створювати за допомогою традиційні техніки попіксельної анімації. Також доволі часто використовуються афінні перетворення. Однак це вимагає ручної праці аніматора для збільшення реалістичності сцени. У випадку використання нейронних мереж (наприклад, технологія DeepFake) необхідно використовувати великі обчислювальні ресурси для тренування, з однієї сторони, і визначених наперед класів об’єктів, які мають анімуватися, з іншої.

Існуючі підходи до створення моделі безпосередньо з фотографій , вимагають введення кількох зображень фотографій, або ручного опрацювання зображення аніматором. Починаючи з 2005 року ведуться дослідження щодо розроблення анімації на основі одного зображення (не набору послідовних зображень і не відео). Розроблені методи для анімації природних явищ, наприклад, руху води і вітру (D. Goldman, Y. Chuang, K. Zheng, R. Szeliski, B. Curless, D. Salesin та ін.), рухів тварин (зокрема, птахів) (X. Xu , L. Wan, X. Liu, T.-T. Wong, L. Wang та C.-S. Leung), реконструкції тривимірних форм тих чи інших об’єктів (автомобілів) і зміни їхнього ракурсу на фото (N. Kholgade, T.

Simon, A. Efros та Y. Sheikh), ряд робіт зі зміни форми тіла і виразу обличчя людей. Уже в 2016 розроблені моделі анімації людей за допомогою Skinned Multi-Person Linear Model і глибоких нейромереж. Проте, усі перераховані роботи вимагають визначення класу об’єкта для анімування, що унеможливлює їх використання для широкого вжитку.

З іншої сторони, з поширенням мобільних телефонів та їх використання не тільки як засобу зв’язку, усе частіше виникає необхідність використання анімації у різних застосунках безпосередньо на телефоні. Усе це спричинює лімітацію у обчислювальних ресурсах навіть для таких задач, як ідентифікація об’єкта, його сегментація та розпізнавання, не кажучи уже про анімацію.

Тому задача дисертаційної роботи розроблення методів та засобів аімування об’єктів в умовах обмежених обчислюваних ресурсів є актуальною.

Зв’язок роботи з науковими програмами, планами і темами. Дисертація виконувалася відповідно до пріоритетних напрямків науково-дослідних робіт Національного університету “Львівська політехніка”, відповідно до координаційних планів Міністерства освіти і науки України. Зокрема, в рамках наукових досліджень, виконуваних відповідно до держбюджетної роботи кафедри систем штучного інтелекту «Інформаційна технологія формування психофізичного портрету в умовах стресових ситуацій» (№ держ. реєстру 0119U002257).

Метою дисертаційної роботи є розроблення методів та засобів опрацювання об’єктів на зображеннях для їх анімування.

Для досягнення поставленої мети в роботі потрібно розв’язати такі **задачі**:

1. провести аналіз сучасного стану сегментації, ідентифікації та анімування зображень;
2. розробити метод контурного аналізу зображень та пошуку об’єкта на зображенні;
3. розробити модель класифікації та кластеризації об’єктів;
4. розробити метод анімування статичних об’єктів на зображеннях;
5. розробити алгоритми сегментації та виділення ознак об’єктів на зображенні;

6. розробити архітектуру системи та програмний засіб для апробації запропонованих методів та моделей.

Об'єкт дослідження – процеси аналізу об'єктів на зображеннях.

Предмет дослідження – методи, моделі та інформаційні технології анімування зображень.

Методи дослідження. У процесі розробки методу контурного аналізу використано дискретні диференціальні оператори. Для розроблення методів класифікації та кластеризації зображень використано методи штучного інтелекту, математичну статистику, нейронні мережі. Для розроблення методу виділення ознак використано теорію геометричних перетворень. Для побудови програмних продуктів застосовано об'єктно-орієнтоване проектування, методи паралельних та розподілених обчислень, лямбда-архітектуру.

Наукова новизна одержаних результатів полягає в тому що:

- *отримав подальший розвиток* метод пошуку об'єкта на зображенні з використання оператора Собеля на основі застосування маски коефіцієнтів для середніх значень, що стало основою для розроблення моделі опрацювання зображення;
- *вперше* розроблено модель класифікації та кластеризації об'єктів на основі аналізу геометричних ознак та гістограми градієнтів інтенсивності та з використанням оригінальної архітектури згорткової нейронної мережі, що дало змогу опрацьовувати об'єкти зі стійкістю до повороту або масштабування об'єкту, зміною ракурсу без необхідності їх повторного аналізу;
- *вперше* розроблено метод анімування статичних об'єктів на зображенні з використанням афінних перетворень та вектора анімації із значеннями, залежними від часу, що дало змогу зберігати пропорції паралельних об'єктів;
- *вдосконалено* метод пошуку об'єкта на зображенні з використанням оператора Собеля та Прюітта шляхом врахування середньої яскравості зображення та застосування алгоритмів адаптивної порогової обробки та

Брезенхема, що дає змогу опрацьовувати масштабовані зображення та зображення, отримані з іншого ракурсу.

Практичне значення одержаних результатів. Практична цінність роботи полягає у доведенні отриманих наукових результатів до конкретних технологій, методик, алгоритмів та програмних продуктів. На основі методів було розроблено архітектуру системи з орієнтуванням на мобільні пристрої. Для застосування алгоритмів у режимі реального часу запропоновано використовувати лямбда-архітектуру, що забезпечує масштабованість пам’яті та зменшує навантаження на сервер у 2 рази. Розроблено мобільний додаток, у якому реалізовано усі наукові результати.

Запропоновані методи і моделі впроваджені у навчальний процес Національного університету «Львівська політехніка» при викладанні дисципліни «Системний аналіз» (підтверджено актом впровадження).

Також результати дисертаційної роботи впроваджено в ТЗОВ SoftServe.

Особистий внесок здобувача. Основні положення та результати дисертаційної роботи одержані автором самостійно. Особисто здобувачеві належать наступні наукові результати: розроблено алгоритм анімування об’єктів за допомогою афінних перетворень [1 5], розроблено метод метод контурного аналізу зображень [2, 6], розроблено метод класифікації та кластеризації об’єктів на зображенні [3], розроблена архітектуру розподіленого опрацювання інформації [4, 7, 8].

Апробація результатів дисертації. Результати дисертаційної роботи доповідались на конференціях: Комп’ютерні науки та інформаційні технології 2016, 2019, X Міжнародна науково-практична конференція «INNOVATIONS AND PROSPECTS OF WORLD SCIENCE. Також результати доповідались на семінарах кафедри систем штучного інтелекту «Національного університету «Львівська політехніка».

Публікації. За результатами виконаних досліджень опубліковано 9 наукових праць, із них 3 статті – у фахових виданнях України, 3 публікації – у міжнародних виданнях, 3 – у збірниках наукових праць конференцій.

Структура та обсяг дисертації. Дисертація складається із вступу, чотирьох розділів, висновків, списку використаних джерел із 91 найменування та додатків. Повний обсяг дисертації складає 152 сторінки, основний зміст викладено на 113 сторінках, де наведено 50 рисунків та 2 таблиці.

РОЗДІЛ 1. АНАЛІЗ АЛГОРИТМІВ ПОШУКУ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ, ЇХ КЛАСИФІКАЦІЇ ТА КЛАСТЕРИЗАЦІЇ, АНІМУВАННЯ СТАТИЧНИХ ОБ'ЄКТІВ.

У розділі проаналізовано алгоритми пошуку об'єктів, а саме контурний аналіз та алгоритм штучної нейронної мережі. Також розглянуті методи кластеризації та класифікації даних:

- класифікація по найближчому середньому значенні;
- класифікація по відстані до найближчого сусіда;
- байєсівський підхід.

Проаналізовано методи пошуку об'єктів на зображенні за допомогою контурного аналізу та штучних нейронних мереж. Розглянуто алгоритми перетворення зображень та існуючі підходи до створення анімації з статичних зображень.

Результати розділу опубліковано в працях автора [1, 2].

1.1. Аналіз існуючих алгоритмів пошуку об'єктів.

1.1.1. Контурний аналіз.

Контур – це сукупність точок, що відокремлюють об'єкт з фону на зображені [1]. У комп'ютерних системах використовуються різні способи кодування контуру. Найбільш відомими є: код Фрімена, двовимірне кодування, полігональне кодування, але ці способи не використовуються в методах контурного аналізу. У методах контурного аналізу контур об'єкта кодується послідовністю комплексних чисел, спочатку фіксується точка, що називається початковою, далі контур аналізується у визначеному напрямі і кожен вектор зміщення записується комплексним числом:

$$\overrightarrow{AB} = a + ib,$$

де a – зміщення точки по осі X, b – зміщення по осі Y. Зміщення береться відносно попередньої точки [2].

Усі методи контурного аналізу базуються на одній із властивостей яскравості – розривності. Для пошуку цієї розривності використовують так звану

ковзну маску [3 – 4], що являє собою квадратну матрицю коефіцієнтів відповідну до групи пікселів вхідного зображення. Дану матрицю використовують для виконання просторової фільтрації зображення [5 – 6].

Контурний аналіз дозволяє визначати, зберігати, порівнювати та проводити пошук об’єктів, що представляються у вигляді своїх зовнішніх обрисів, це і є контури об’єкту. Передбачається, що контур містить усю необхідну інформацію про форму об’єкта, при цьому внутрішні точки об’єкта не беруться до уваги. Такий підхід з одного боку обмежує область застосування алгоритмів контурного аналізу, але з іншого боку аналіз тільки контурів дозволяє проводити обчислення меншої кількості пікселів, що зменшує алгоритмічну складність. Контурний аналіз дозволяє ефективно вирішувати основні проблеми розпізнавання образів: перенесення, поворот та зміна масштабу об’єкта на зображені. Це зумовлене тим, що методи контурного аналізу інваріанті до даних перетворень.

Просторова фільтрація виконується звичайним переміщенням маски по зображенню та обрахуванням в кожній точці градієнту рівня яскравості. Однотонні простори зображення будуть мати низький градієнт рівня яскравості, в вихідному зображенні такі простори зображення темнішають, а у випадку високого рівня градієнту на вихідному зображенні дані простори будуть мати яскравіші лінії. Перед обрахунком градієнту потрібно вирахувати відгук $\boldsymbol{R}$ фільтрації в точці $\boldsymbol{(x, y)}$ [7]:

$$R = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots + w(0,0)f(x,y) + \cdots w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1),$$

тобто це сума добутку коефіцієнтів маски на значення пікселів до яких застосовується маска. Для визначення градієнта яскравості використовуються дискретні аналоги похідних першого і другого порядку [8]. Перша похідна $\boldsymbol{f(x)}$ визначається, як різниця значень сусідніх пікселів, а друга похідна, як різниця значень сусідніх значень першої похідної [9].

$$\frac{\partial f}{\partial x} = f(x+1) - f(x),$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x).$$

Піксель зображення звичайно має дві координати, а отже у випадку двох змінних ***(x, y)*** потрібно просто обрахувати часткові похідні по двом просторовим осям. Обчислення першої похідної зображення виконується на дискретних наближеннях двовимірного градієнта. Градієнт зображення це – вектор [10].

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

У методах контурного аналізу важливим є модуль вектора градієнту функції:

$$|\nabla f| = \sqrt{G_x^2 + G_y^2}$$

Напрям вектору градієнта теж є основною характеристикою при знаходженні контурів:

$$a(x, y) = arctg\left(\frac{G_y}{G_x}\right)$$

Напрям контуру в точці ***(x, y)*** буде перпендикулярний напряму вектору градієнту [11].

Природа об’єктів на зображеннях встановлює правило, що контури завжди замкнуті, це дозволяє однозначно визначити обхід контуру. Останній вектор контуру завжди закінчується на початковій точці з якої починався аналіз контуру об’єкта. Методи контурного аналізу використовують комплексне кодування за для забезпечення використання математичних властивостей векторів комплексних чисел, а саме комплексне кодування можливо описати двовимірними координатами, але різниця тільки в тому, що скалярний добуток для векторів та комплексних чисел відрізняються, саме це дає перевагу методам контурного аналізу. Методи контурного аналізу базуються на таких властивостях контуру об’єктів:

- сума векторів замкнутого контуру дорівнюю нулю, це доводиться тим, що вектори при обході контуру приводять в початкову точку.

- Контур не залежить від паралельного переносу вхідного зображення. Оскільки контур кодується відносно початкової точки, то такий спосіб є стійким до зсуву вхідного контуру об’єкта.
- Поворот об’єкта на зображені на певний кут є рівнозначним повороту всіх векторів, що належать контуру об’єкта, на той самий кут.
- Зміна початкової точки обходу контуру викликає операцію циклічного зсуву векторів контуру.
- Зміна масштабу об’єкту є добуток кожного вектора, що належать контуру, на коефіцієнт масштабування.

Методи контурного аналізу мають дві групи факторів, що погано впливають на результати розпізнавання.

Перша група пов’язана з проблемою переривання при виділенні контуру. Контур – це суворо визначена дискретна структура, але деякі об’єкти можуть слабо виділятися на фоні зображення. Також об’єкт може мати не чітку границю через те, що буде мати однакову яскравість та кольори з фоном зображення, також границя об’єкта може мати шуми, тощо. Всі ці фактори призводять до того, що контур не виділяється зовсім, або він визначається не вірно та не відповідає реальним границям об’єкта. Отже важливим є забезпечення стійкості до шумів на зображенні при розробці алгоритму виділення контурів об’єктів, бо контурний аналіз має сенс тільки у випадку коли контур об’єкта визначено однозначно в усіх точках границі об’єкта.

Друга група факторів, що впливає на точність визначення контурів, пов’язана з самими принципами методів контурного аналізу. Дані методи припускають, що контур описує весь об’єкт повністю і не допускає жодних перетинів з іншими об’єктами, також проблемою є не повна видимість об’єкта на зображенні.

Отже базова задача пошуку об’єктів на зображенні при використанні методів контурного аналізу складається з таких кроків:

1. Попередня обробка зображення, а саме згладжування, знешумленя, бінарізація.
2. Виділення границь об’єктів.

3. Фільтрація контурів об’єктів по периметру, площі, коефіцієнту форми, фрактальності тощо.
4. Приведення контурів до визначеної довжині згладжування.
5. Аналіз всіх знайдених контурів та пошук шаблону максимально схожого на вхідний контур.

1.1.2. Нейронні мережі.

Одним з найбільш ефективних і поширених способів подання і вирішення завдань розпізнавання образів є штучні нейронні мережі (ШНМ). Узагальнена математична модель нейрона (Рис. 1.1):

- вхідні сигнали x_i-дані, що надходять з навколишнього середовища або від інших активних нейронів. Сигнали поділяються на дискретні (подані множиною [0, 1] або [-1, 1]) або неперервні з бідь-якими дійсними значеннями.
- вагові коефіцієнти w_i – визначають силу зв’язку між нейронами;
- рівень активації (потенціал) нейрона $P = \sum w_i x_i$;
- функція активації $Y=f(P)$ – використовується для визначення значення вихідного сигналу, що передається іншим нейронам [12].

Вхідні сигнали x_i множаться на вагові коефіцієнти w_i (синаптичні ваги), і отримана зважена сума $P = \sum w_i x_i$ піддається зміні функцією $f(P)$ (функцією активації). Вхідний сигнал Y також може піддаватися зважування (масштабування). В якості функції активації використовують різні функції, але частіше сигмоідну функцію $Y = \frac{1}{(1+e^{-\lambda P})}$ (Рис. 1.2), а також гіперболічний тангенс, логарифмічну функцію, лінійну та інші. Основна вимога до таких функцій – монотонність [13].

Один із факторів, через що сигмоід використовується у нейронних мережах – це просте вираження його похідної через саму функцію $S`(x)=S(x)\cdot(1-S(x))$, що дозволяє істотно зменшити обчислювальну важкість метода зворотного розподілу помилки. Нейронна мережа реалізує просту регресійну модель для N незалежних змінних. Якщо об’єднати багато нейронів у нейронні структури, то і

функція, що реалізується, може бути наскільки завгодно складною. Нейронна мережа – це сукупність обчислювальних елементів (нейронів), кожний з яких має декілька входів-синапсів та один вихід-аксон [14].

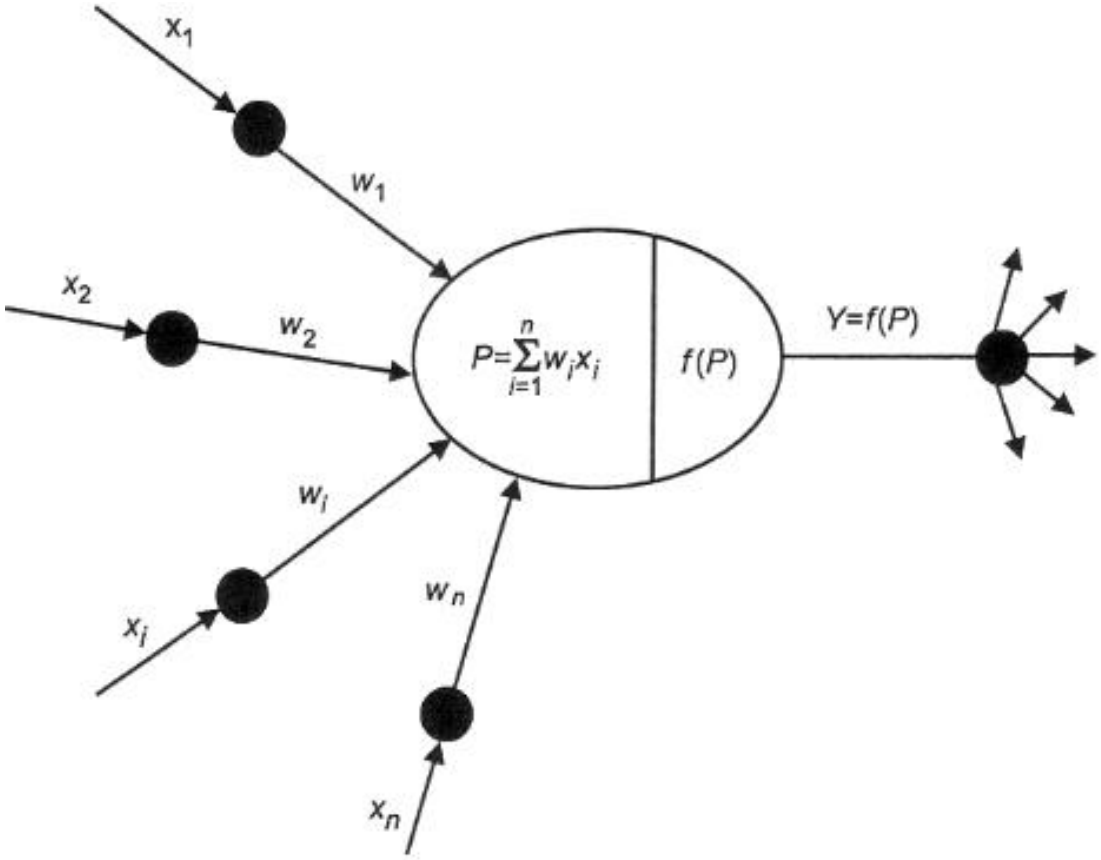

Рис. 1.1. Математична модель нейрона.

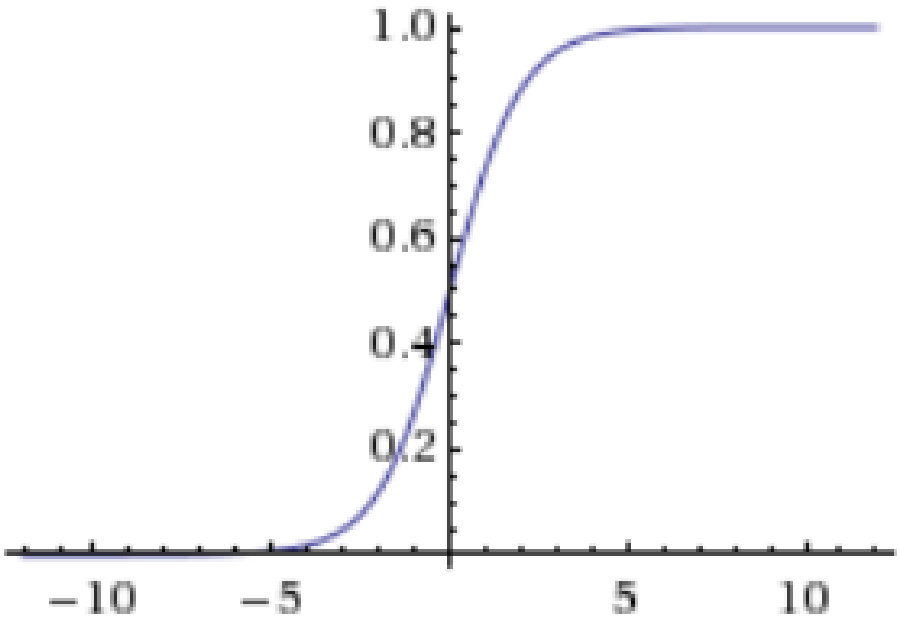

Рис. 1.2. Графік функції логістичного сигмоіду.

Для того, щоб штучна нейронна мережа була спроможна виконувати поставлену задачу пошуку та розпізнавання об’єкта, її необхідно навчити (Рис. 1.3). Процес навчання штучної нейронної мережі – це процес при якому параметри ШНМ налаштовуються шляхом моделювання середовища, що подається на вхід. Тип навчання визначається способами налаштування параметрів. Виділяють таки способи навчання: з наставником та без наставника [81].

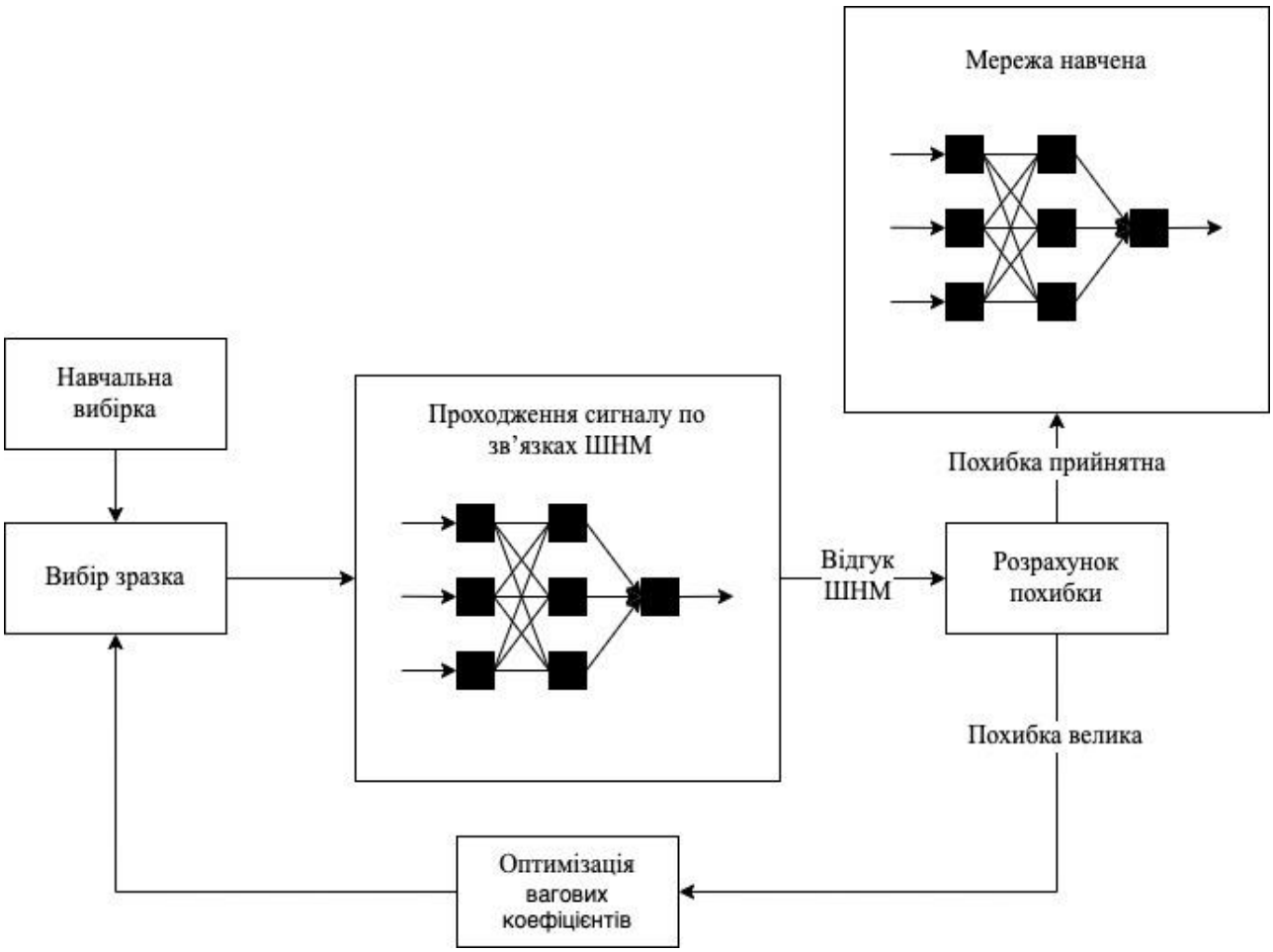

Рис. 1.3. Процес навчання ШНМ.

Процес навчання з наставником являє собою подачі завчасно визначених прикладів зразків вибірки. Кожен зразок подається на вхід до ШНМ, потім проходить обробку в структурі мережі, розраховується вихідний відгук, який порівнюється з відповідним значенням цільового вектора, що являє собою потрібний вихід мережі. Далі за визначеним правилом вираховується помилка, що тягне за собою зміну вагових коефіцієнтів в середині ШНМ, правила зміни

визначаються обраним алгоритмом. Вектори навчальної вибірки подаються на вхід послідовно, обчислюються помилки, ваги оптимізуються для кожного вектору, поки помилка для всієї навчальної вибірки не досягне прийнятого низького рівня [82].

При навчанні без вчителя навчальна множина складається лише з вхідних зразків. Навчальний алгоритм підлаштовує ваги мережі так, щоб виходили узгоджені вихідні вектори, тобто, щоб пред'явлення досить близьких вхідних векторів давало однакові виходи. Процес навчання, отже, виділяє статистичні властивості навчального безлічі і групує подібні вектори до класів. Пред'явлення на вхід вектора з даного класу дасть певний вихідний вектор, але до навчання неможливо передбачити, який вихід проводитиметься даним класом вхідні вектори. Отже, виходи подібної мережі мають трансформуватись у деяку зрозумілу форму, обумовлену процесом навчання. Це не є серйозною проблемою. Зазвичай не складно ідентифікувати зв'язок між входом та виходом, встановлений мережею. Для навчання нейронних мереж без вчителя застосовуються сигнальні методи навчання Хебба та Ойа [83].

Нейронні мережі дуже добре себе показують у задачах розпізнавання образів, через те що він поєднує в собі математичні та логічні обчислення. Нейронна мережа дозволяє обробляти велику кількість факторів, незалежно від їх походження, це дуже хороший універсальний алгоритм. Нейронні мережі дозволяють на основі навчальної вибірки будувати залежності параметрів у вигляді полінома, що дуже спрощує реалізацію розпізнавання образів.

1.2. Проблема класифікації та кластеризації об'єктів.

1.2.1. Поняття та формальна постановка задачі розпізнавання та класифікації.

Клас – множина об'єктів, що мають загальні властивості. Для об'єктів одного класу важливим буде наявність «схожості». Для завдання розпізнавання обов'язково має бути визначено один клас, але може бути визначено довільну кількість класів. Кількість класів описується змінною *S*. Кожен клас повинен мати свою мітку, що його ідентифікує [15].

Класифікація – процес призначення міток класу об’єктів, відповідно до деякого опису властивостей цих об’єктів. Класифікатор – засіб (алгоритм, пристій тощо), який отримує набір ознак об’єкта в якості вхідних даних, а в якості результату видає мітку класу. [16]

Верифікація – процес зіставлення примірника об’єкта з однією моделлю об’єкта або описом класу [17].

Під образом слід розуміти обрану область в просторі ознак, якій належать безліч об’єктів або явищ матеріального світу. Ознакою є кількісний опис тієї чи іншої властивості досліджуваного предмета або явища.

Під простором ознак розуміємо *N-мірний* простір, який будується для заданої задачі розпізнавання. Тут *N* визначається як фіксоване число ознак, що вимірюються, для будь-яких об’єктів. Вектор з простору ознак *x*, відповідний об’єкту завдання розпізнавання це *N-мірний* вектор з компонентами $(x_1, x_2, ..., x_N)$, які є значеннями ознак для даного об’єкта.

Іншими словами, процес розпізнавання образів можна описати, як зарахування об’єкту із заначеними вихідними даними до певного класу за допомогою виділення важливих ознак або властивостей, які характеризують об’єкт, із загальної маси несуттєвих деталей [18].

Проаналізувавши загальні відомості про класифікацію, можна описати формальну задачу класифікації. Дано множина об’єктів, яких необхідно класифікувати з використанням певного алгоритму. Множина об’єктів представлено підмножинами, які називаються класами. Окрім того, маємо додаткову інформацію: опис класів, інформацію про усі об’єкти, представлені в множині, значення атрибутів об’єктів, належність об’єктна до певного класу задана. Ставиться задача за наявною інформацією про класи і опис об’єкта встановити залежність, до якого класу належить цей об’єкт.

Частіше за все в задачах розпізнавання образів аналізуються монохромні зображення, що надає змогу аналізувати зображення як функцію на площині. Якщо розглянути точкову безліч на площині *T*, де функція $f(x, y)$ визначає в кожній точці зображення його характеристику – прозорість, оптичну щільність, яскравість, то така функція є формальним записом зображення [19].

Множина всіх можливих функцій *f(x, y)* на площині *T* є модель безлічі всіх зображень *X*. Додавання твердження подібності між образами дає змогу поставити задачу розпізнавання. Певний вид такої постановки є суттєво залежним від наступних етапів процесу розпізнавання відповідно до тих чи інших підходів.

Виділяють три групи методів класифікації у розпізнаванні образів:

- Порівняння із зразком. До цієї групи належать класифікація по відстані до найближчого сусіда та класифікація по найближчому середньому. Структурні методи розпізнавання також можна віднести до групи порівняння зі зразком.
- Статистичні методи. Як зрозуміло з назви, статистичні методи мають на основі використання деякої статистичної інформації при вирішенні задачі розпізнавання. Такі методи визначають принадність об’єкту до певного класу на основі імовірності. В ряді випадків це зводиться до визначення апостеріорної ймовірності приналежності об’єкта до певного класу, за умови, що ознаки цього об’єкта взяли відповідні значення. Прикладом є метод, побудований на основі наївного Байєса.
- Нейронні мережі (навчання без вчителя, навчання з вчителем, навчання з підкріпленням). Поданий окремо від інших методів розпізнавання, оскільки може працювати на наборі неповних даних [20].

1.2.2. Класифікація по найближчому середньому значенні.

У класичному підході розпізнавання образів, в якому невідомий об’єкт для класифікації визначений у вигляді вектору елементарних ознак. Система розпізнавання на основі визначення ознак може бути реалізована різними методами. Вектори ознак можуть бути збережені в системі заздалегідь в результаті навчання або отримуватися в режимі реального часу на основі будь-яких моделей.

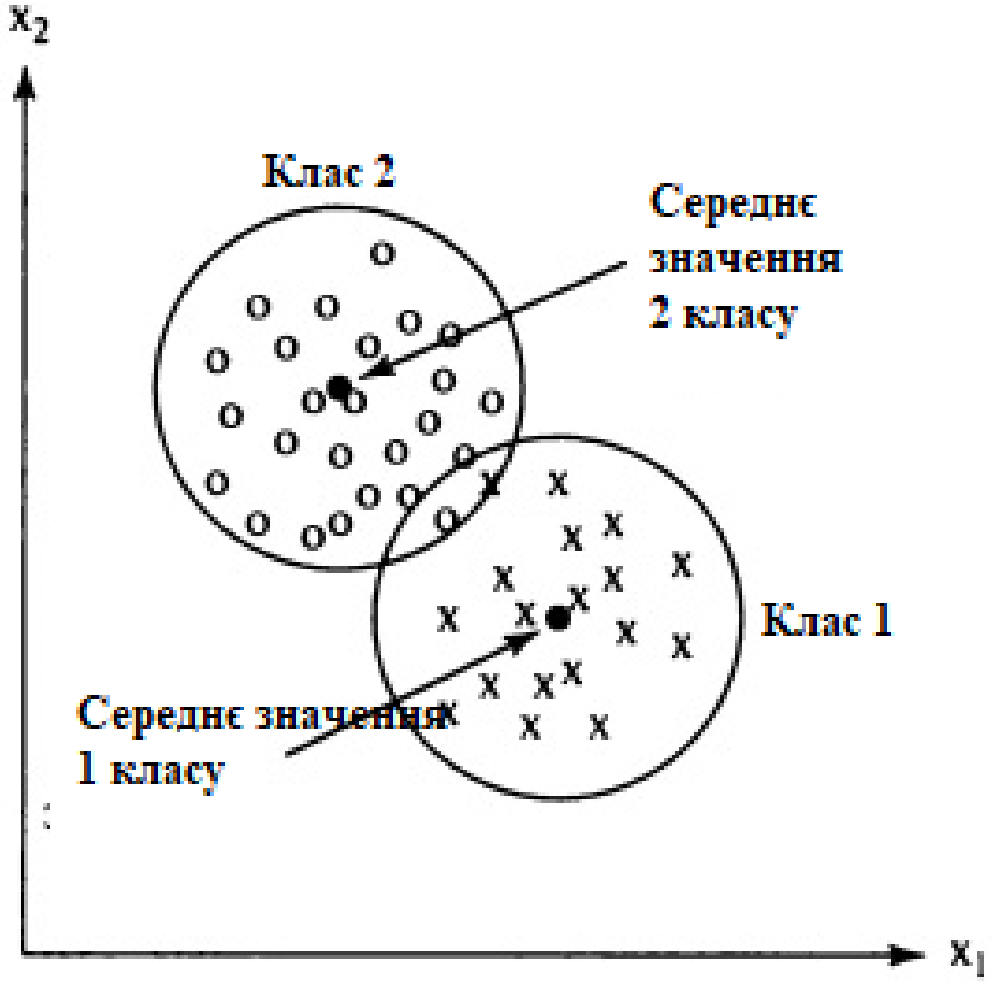

Рис. 1.4. Графічне відображення компактного розташування точок класів.

Простий алгоритм класифікації полягає у побудові груп, які складаються з еталонних даних класу на основі значень вектора математичного очікування класу (центроїда або середнього значення) [21].

$$\underset{x_i}{\rightarrow} = \frac{1}{n_i}\sum_{j=1,n_i} x_{i,j},$$

де *x(i, j) – j-й* еталонний ознака класу *i*, n_j – кількість еталонних векторів класу *i*.

Тоді невідомий об’єкт буде відноситися алгоритмом до класу *i* у випадку, якщо його ознаки знаходяться якнайближче до вектору математичного очікування класу *i*, порівняно з векторами математичних очікувань інших класів [22]. Використання цього методу можливо тільки для задач де точки класів не віддаленні одна від одної і розташовані далеко від точок інших класів (Рис. 1.4).

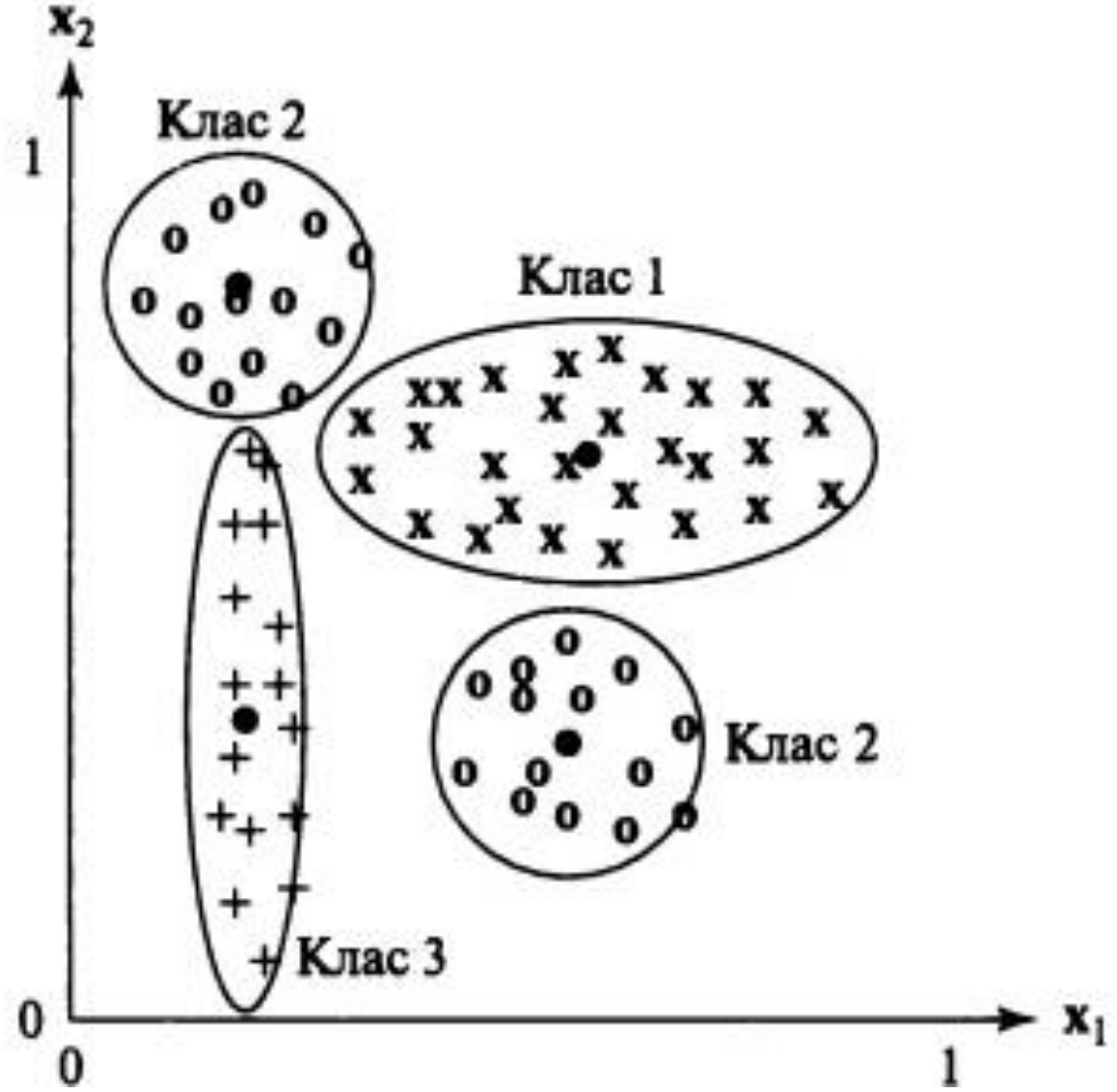

Рис. 1.5. Графічне відображення складної структури розташування точок класів.

Розглянемо випадок коли клас 2 розділений на дві ділянки, що не пересікаються та погано описуються одним середнім значенням. При цьому клас 3 має витягнуту структуру розподілу, зразки 3-го класу з великими значеннями координат x_2 ближче до середнього значення 1-го класу, ніж 3-го (Рис. 1.5). Зміна розрахунку відстані в деяких випадках може вирішити описану проблему.

Потрібно буде врахувати характеристики «розкиду» уздовж кожного координатного напрямку i для значень класу σ_i. Середньоквадратичне відхилення дорівнює квадратному кореню з дисперсії, а евклідова відстань між вектором x і вектором математичного очікування x_c визначається як [23]

$$\|x - x_c\| = \sqrt{\sum_{i=1,d} \left(\frac{x[i]-x_c[i]}{\sigma_i}\right)^2}.$$

В більшості випадків задачі не можливо описати таким простим класом, але дана формула зменшує кількість помилок класифікації.

1.2.3. Класифікація по відстані до найближчого сусіда.

Інший підхід до класифікації полягає у віднесенні невідомого вектора ознак *x* до того класу, до якого цей вектор найбільш близький по окремому зразку. Це правило називається правилом найближчого сусіда. Коли класи перетинаються або мають складну структуру, класифікація по найближчому сусідку є більш ефективною.

Моделі розподілу векторів ознак при використанні даного підходу будуть не потрібні. Цей метод використовує тільки інформацію про відомі еталонні зразки. Методика розв'язання задачі класифікації при використанні даного підходу заснована на обчисленні відстані *x* до кожного класу в наборі даних і пошук мінімальної відстані. Переваги зазначеного підходу можна сформувати таким чином:

- Можливість додавання нових зразків класів в базу даних;
- Використання деревовидних і мережевих структур даних дають змогу зменшити кількість відстаней для обчислення [24].

Для забезпечення більш точнішого вирішення задачі класифікації потрібно обчислювати відстань не для одного найближчого сусіда, що знайдений в базі даних, а для *k* сусідів, в такому випадку забезпечується найкраща вибірка розподілу векторів в *d-вимірному* просторі. При достатній кількості ознак в області простору, що аналізуються, використання значень *k* буде найбільш ефективним.

1.2.4. Байєсівський підхід.

Статистична природа спостережень є осново Байєсівського підходу. Існування міри, що отримана випадково на просторі образів, яка вже відома або може бути оцінена, береться за основу, як припущення. Метою є створення такого класифікатора, який забезпечить вірне визначення найбільш імовірного

класу для тестового образу. Отже розрахунок «найбільш імовірного» класу і буде завданням класифікації в даному підході. Розглянемо метод більш детально.

Задано M класів $\alpha_1, \alpha_2, \ldots, \alpha_M$, а також $P(\alpha_i|x), i=1,2, \ldots, M$ – ймовірність того, що невідомий образ, що визначений вектором ознак x, належить класу α_i. $P(\alpha_i|x)$ називається апостеріорною ймовірністю, оскільки задає розподіл індексу класу після експерименту.

Розглянемо випадок двох класів α_1 та α_2. Типово обрати вирішальне правило таким чином: об'єкт відносимо до того класу, для якого апостеріорна ймовірність вище. Таке правило класифікації по максимуму апостеріорної ймовірності називається Байєсовським: якщо $P(\alpha_1|x) \parallel P(\alpha_2|x)$, то x класифікується як α_1, в іншому випадку α_2. Таким чином, для Байєсовського вирішального правила необхідно отримати апостеріорні ймовірності $P(\alpha_i|x)$, $i=1,2$. Це можливо зробити за формулою Байєса [25].

Нехай $A_1, A_2, \ldots, A_n$ – повна група несумісних подій $\bigcup_{i=1}^{n} A_i$. Тоді апостеріорна ймовірність має вигляд:

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{\sum_{i=1}^{n} P(A_i)P(B|A_i)}$$

де $P(A_i)$ – апріорна ймовірність події A_i, $P(B|A_i)$ – умовна ймовірність події B при умові, що сталася подія A_i.

Отже, задача порівняння на основі апостеріорної ймовірності зводиться до обчислення $P(\alpha_1)$, $P(\alpha_2)$, $P(x|\alpha_1)$, $P(x|\alpha_2)$. При достатній кількості даних для визначення приналежності об'єкту кожному із класів. Тоді ймовірності будуть апріорними, а функції розподілу вектора ознак класу – функціями правдоподібності x по відношенню до α_i [26].

Отже, Байєсівський підхід до аналізу статистичних даних ґрунтується на припущенні про існування для кожного параметру деякого розподілу ймовірностей. Постулювання, як метод виділення для невідомого параметру апріорного розподілу є недоліком цього методу.

Інші статичні методи розпізнавання та класифікації базуються на Байєсівському підході.

1.3. Аналіз методів анімування статичних об’єктів на зображеннях.

Під анімацією зображень розумітимемо зміну параметрів обраних об’єктів на зображенні з плином часу. В основі цього лежатиме той факт, що зорова система людини сприймає набір окремих зображень, що подаються з досить високою швидкістю, як безперервний рух. Під змінними параметрами об’єкта розумітимемо переміщення об’єкта із збереженням його форми.

Афінне перетворення є комбінацією лінійних перетворень, супроводжуваних переносом зображень. Афінні перетворення формують зручну підсистему білінійних перетворень, тому що добуток двох афінних перетворень також є афінним. Це дозволяє представити узагальнену орієнтацію системи точок стосовно довільної координатної системи при збереженні одиничного значення однорідної координати H [27].

Для створення ефекту анімації на статичних зображеннях добре використовувати афінні перетворення. Це досягається завдяки тому, що афінні перетворення мають властивість зберігати пропорції паралельних об’єктів, а саме довжину відрізків на паралельних прямих і площ на паралельних площинах. Така властивість дозволяє будувати паралельні прообрази полігону об’єкта на площині по кінцевому набору точок, що виражають полігон об’єкта.

Сьогодні анімування зображень усе частіше використовується в індустрії розваг і виконується за умови обмеження ресурсів – на мобільних пристроях, в режимі реального часу тощо. Об’єкти анімування можуть бути різної природи – від облич людей [28], не фізичних об’єктів (аватарів) [29] до будь-чого, визначеного користувачем на зображенні.

Анімування забезпечується різними техніками. Так, у [30] використовується метод квазі-випадкової прогулянки для створення зображень та ефектів анімації у них. Проте, ця методика не може бути застосованою до об’єктів невизначеної на момент анімування природи, що не дає змоги застосувати цей метод у режимі реального часу.

Автори у [31] реалізували систему додавання анімації водоспаду шляхом вилучення анімації потоку з послідовності відео. Запропонований спосіб інтегрує оптичний потік, інтегральну лінію згортки, передачу кольору, графічне

вирізання та сплайн-схеми з різною роздільною здатністю, щоб імітувати реальний водоспад на одному зображенні. Він використовує процес сегментації, щоб відокремити необхідний передній план і зайвий фон. Потім аналіз потоку проводиться на цільовому зображенні та вихідному відео. Нарешті, для формування анімації застосовується подібність потоку та процес синтезу. Проте, такий підхід можна застосовувати не до одного зображення, а за наявності серії зображень.

Також існує підхід до побудови анімації за допомогою ейлерових полів руху. Так автори у [86] продемонстрували автоматичний підхід, що дозволяє створювати реалістичне анімаційне циклічне відео перетворенням нерухомого зображення. Метод був спрямований на використання зі сценами, що мають безперервний плавний рух, як-от текуча вода та клуби диму. Припущення базується на спостереженні, що такий тип природнього руху може бути відтворено зі статичного ейлеревого опису руху, тобто єдиного постійного в часі поля потоку, яке визначає миттєвий рух частинки в даному 2D-місці. В основі лежить використання мережі трансляцій вхідного зображення, що базується на кодуванні попередніх рухів природніх сцен, які були зібрані з відео, і їх синтезування полів руху для вихідного зображення. Далі за допомого техніки глибокого викривлення об’єкти, що піддаються анімації, перетворюються через ейлерів рух, а отримані карти викривлених об’єктів декодуються як зображення. Проблему створення безперевних відеотукстур з повторенням, авторами було запропоновано техніку циклічного перетворення відео, за допомогою змішування результатів відтворення кадрів відео як в зворотному порядку так і в звичайному.

Один із нових підходів, а саме анімації нерухомих природних зображень за допомогою викривлення, був представлений авторами у [87]. В цьому дослідженні автори пропонують метод, що деформує пікселі напряму, а не за допомогою нейронних функцій. Для забезпечення створення анімаціям був запропонований підхід Preserve-Curve-Warping (PCW), що забезпечує збереження кривизни вхідних штрихів у результатах. Також авторами був запропонований підхід зменшення артефактів на границях анімованих та

статичних областей за допомогою виконання кроків явного матування для отримання анімованих областей та їх незалежної анімації. Також був описаний метод, що дозволяє виключити статичні пікселі, що виникали в анімованій області, що дало змогу усунути «примарні артефакти» анімації.

Основними обмеженнями для створення методів анімації зі статичних зображень. З одного статичного зображення має бути можливість створити відео на якому анімації будуть піддаватися тільки певні об’єкти. Тобто в результаті має бути отриманий гібрид фотографії та відео. Такі відео містять в собі фон, який залишається нерухомим та об’єкти, які в цей час мають плавний рух, що створює ефект анімації цих об’єктів.

Для опрацювання зображень широко використовують конволюційні нейронні мережі [32]. Однак конволюційні нейронні мережі обмежені відсутністю здатності просторово перетворювати входи.

Важливо виділити те, що маніпуляції будуть проводитися на двовимірних зображеннях. Функція двох дійсних змінних $I(x, y)$, де – це інтенсивність (яскравість) у точці з координатами (x, y), буде описувати зображення в розумінні підходу анімування. Не все зображення буде піддаватися обробці, інколи це буде частина зображення або об’єкт на зображенні, яку в англомовній літературі прийнято називати region-of-interest, ROI (область, що представляє інтерес, ОПІ).

Для обробки на комп’ютері зображення повинно бути дискретизоване і квановане. Цифрове зображення $A(m, n)$ подано у дискретно двовимірному просторі, де m – номер рядка, а n – номер стовпчика. Елемент, розташований на перетині m-ого рядка та n-го стовпця називається *пікселем* (pixel – picture element). Інтенсивність пікселя подана як дійсним, так і цілим числом. Відносна інтенсивність, подана в дійсних числах, зазвичай подається в діапазоні від 0 до 1, а в цілих числах від 0 до 255 [33].

Зазвичай фізичний сигнал, що виникає в точці (x, y), є функцією залежить від багатьох параметрів: (z – глибина, λ – довжина хвилі, t – час). Однак ми будемо розглядати статичні, і частіше монохроматичні зображення.

Існуючі підходи успішно створюють анімацію на статичних зображеннях, але мають такі недоліки, як довгий час обчислень та велику ємнісну складність.

Існує ще і проблема, що не всі об’єкти взагалі піддаються ефекту анімації, яка би виглядала прийнятною з точки зору її природності, тому важливим є подавати користувачу системи саме ті об’єкти, які найчастіше анімувалися іншими користувачами. Адже даний факт буде свідчити про те, що ефект створення анімації саме таких об’єктів був прийнятним для більшості користувачів системи. Також деякі методи не забезпечують інтерактивний контроль руху анімації та узгодженість різних напрямків руху анімації для окремих об’єктів, тому в даному дисертаційному дослідженні будуть розглядатися методи, які мають вирішити данні проблеми.

1.3.1. Основні характеристики зображення.

Розмір: цей параметр інваріантний, але зазвичай його потрібно обирати виходячи з особливостей подальшої обробки.

Глибина кольору, а саме кількість біт, що потрібні для зберігання кольору, визначається спрощенням електронних схем і кратні ступеню двійки. Якщо для зберігання інформації про кольори зображення потрібно виділити всього 1 біт, то таке зображення є бінарним. Зображення для яких зазвичай потрібно 8 біт є напівтонові (gray scale, gray level). Для зберігання кольорових зображень потрібно виділяти 24 біти по 8 біт на кожен з трьох колірних каналів.

Роздільна здатність зображення подається зазвичай в dpi (dot per inch - кількість точок на дюйм). Змінити роздільну здатність можна змінювати в процесі обробки, вплив на зображення при цьому буде мінімальний, але буде змінена відображення на пристрої візуалізації [34].

1.3.2 Методи обробки зображення та типові прикладні задачі.

В основі алгоритмів обробки зображень покладені в основному інтегральні перетворення: згортка, перетворення Фур’є і ін. Також використовуються статистичні методи.

Методи обробки зображень класифікують зазвичай за кількістю пікселів беруть участь в одному кроці перетворення:

- точкові методи в процесі виконання перетворять значення в точці *a(m, n)* в значення *b(m, n)* незалежно від сусідніх точок;
- локальні методи для обчислення значення *b(m, n)* використовують значення сусідніх точок в околиці *a(m, n)*;
- глобальні методи визначають значення *b(m, n)* на основі всіх значень вихідного зображення *A(m, n)*.
- Типові прикладні задачі обробки зображень:
- геометричні перетворення на зображеннях, такі як обертання і масштабування;
- корекція кольору: зміна яскравості, зміна контрасту, квантування кольору, перетворення в інше колірне простір;
- знаходження кореляції між зображенням і зразком;
- комбінування зображень різними способами;
- інтерполяція і згладжування;
- сегментація;
- класифікація;
- кластеризація;
- розпізнавання об’єктів;
- редагування об’єктів;
- ретушування об’єктів;
- розширення (аугментація) динамічного діапазону шляхом комбінування зображень з різною експозицією;
- компенсація втрати різкості, наприклад, шляхом нерізкого маскування [35].

В даній роботі будуть розглянуті задачі розпізнавання об’єктів на зображеннях, їх класифікація та кластеризація, а також комбінування зображень та частин зображень для отримання ефекту анімації статичного об’єкту.

1.4. Постановка задачі.

Задача системи надати користувачу можливість створювати ефект анімації об’єктів на зображені. Для цього, враховуючи обмеження існуючих методів, необхідно розробити нові методи ідентифікації об’єктів на зображенні та анімації об’єктів.

За умовами сучасних тенденцій використання мобільних пристроїв правильним підходом буде розробка мобільного додатку, як основи взаємодії користувача з функціями системи. Отже такий вибір накладає обмеження обчислювальних ресурсів. Тому важливим буде оптимізація процесів системи, які будуть виконуватися саме на пристрої користувача та перенесення ресурсномістких процесів до зовнішніх серверів або з можливістю використання хмарних технологій обчислення та виконання програмного коду.

Для мобільного додатку потрібно спроектувати дружній для користувачів інтерфейс, який би не викликав проблем в освоєні функціоналу системи.

Висновки до першого розділу.

У розділі було проаналізовано існуючі типи підходів до кластеризації та класифікації об’єктів: порівняння зі зразком, статистичні методи, штучні нейронні мережі. Розглянути особливості даних методі, їх переваги та недоліки.

Було розглянуто підходи до вирішення задачі пошуку об’єктів на зображенні за допомогою контурного аналізу та штучних нейронних мереж. Виділено основні принципи даних підходів та описано методи перетворення зображення для ефективнішого виконання даних алгоритмів.

Проаналізовано існуючі підходи до створення анімації статичних об’єктів на зображенні: за допомогою вилучення анімації потоку з послідовності відео, за допомогою ейлерових полів руху, за допомогою викривлення. Було проаналізовано методи виконання анімації за допомогою описаних підходів. Визначено переваги даних методів та їх основні недоліки та запропоновано інший підхід створення анімації за допомогою афінних перетворень.

З урахуванням проаналізованого матеріалу було посталено вимоги до створення та функціонування системи для забезпечення пошуку та анімування об’єктів на зображені.

У розділі здійснено аналіз літературних джерел [1-35, 82, 83, 86, 87].

РОЗІЛ 2. МАТЕМАТИЧНІ МЕТОДИ ПОШУКУ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ, ЇХ КЛАСИФІКАЦІЇ ТА КЛАСТЕРИЗАЦІЇ, АНІМУВАННЯ СТАТИЧНИХ ОБ'ЄКТІВ.

У розділі було розглянуто методи виділення границь об'єктів та проведено аналіз їх ємнісної складності. Також сформовано загальну структуру згорткової нейронної мережі. Розроблено моделі класифікації зображень. Розроблено метод анімування статичних об'єктів та проведено оптимізацію ємнісної складності даного алгоритму.

Результати розділу опубліковано в працях автора [1 – 3, 5].

2.1. Аналіз методів пошуку об'єктів.

2.1.1 Оператор Прюітта

Оператор Прюітта – це диференціальний оператор для виявлення границь об'єктів. Його особливістю є реалізація пошуку країв виконується на основі пошуку різниці значень сірого кольору пікселів у певній області. Просторова фільтрація оператором Прюітта виконується за допомогою маски 3x3, що надає оператору кращий процес виділення границь об'єктів, як в горизонтальному так і в вертикальному напрямах та більшу стійкість до шумів на зображенні.

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

Для даних функцій різниця між сумами по верхній та нижній границі області 3x3 буде відповідати наближеним значенням похідної по осі $\boldsymbol{x}$, а по осі $\boldsymbol{y}$ – різниці між сумами першого та останнього стовпця. Реалізуються ці функції за допомогою маски фільтрації по осям $\boldsymbol{x}$ та $\boldsymbol{y}$ [36].

$$G_y = \begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$$

$$G_x = \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$$

Результати виконання оператора Прюітта на пробному зображенні роздільної здатності 6016x4000px відображено на рис. 2.1, рис. 2.2 та 2.3.

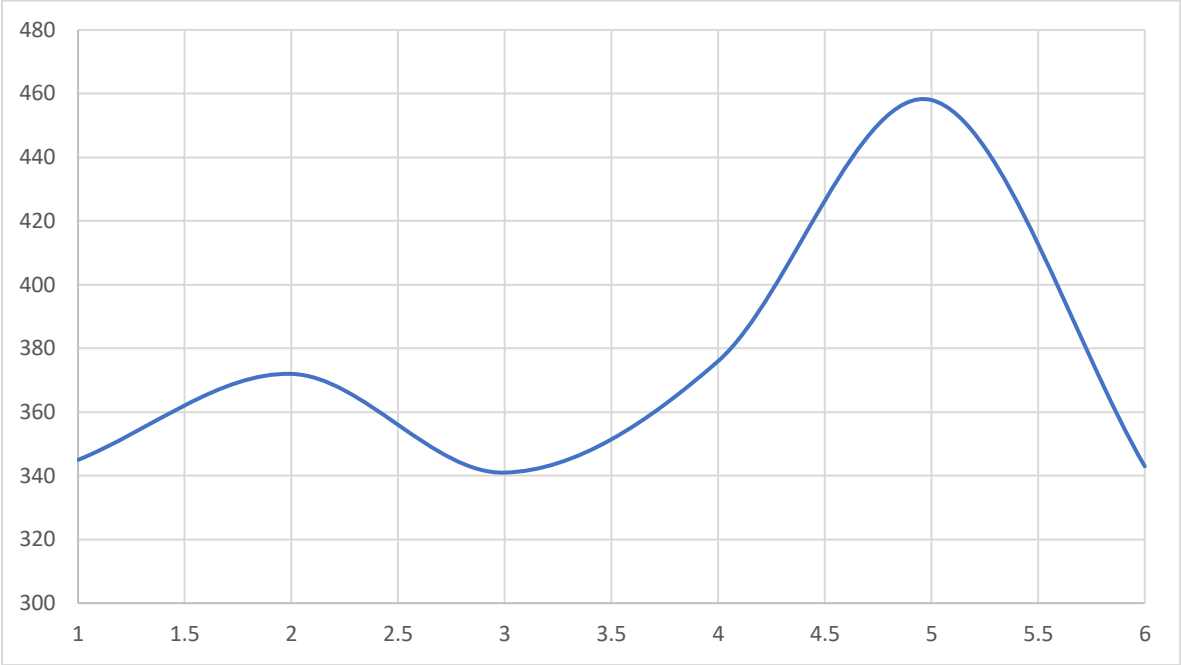

Рис. 2.1. Графік завантаженості оперативної пам’яті (МБ) під час виконання оператора Прюітта.

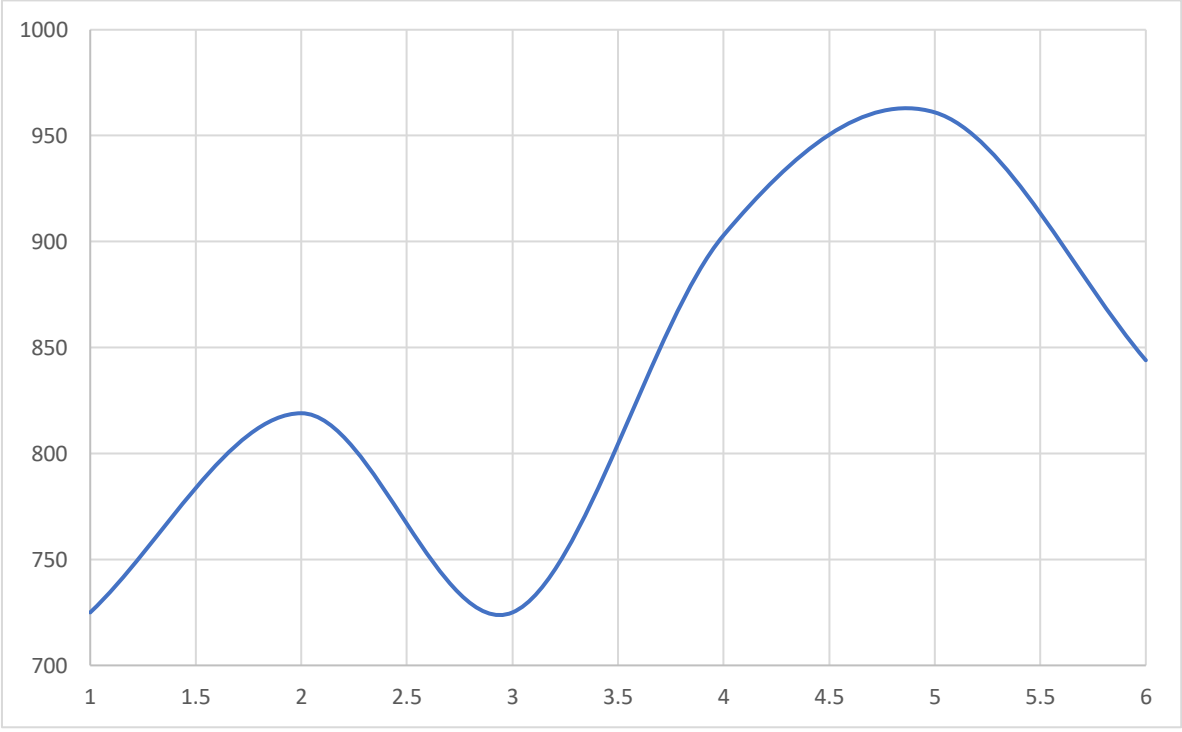

Рис. 2.2. Графік завантаженості процесора (Гц) під час виконання оператора Прюітта.

Результати роботи оператора Прюітта (Рис. 2.3).

Рис. 2.3. Оригінальне зображення (зліва). Результат виконання оператора Прюітта (справа).

Отже виходячи з результатів тестування оператора Прюітта:

- Максимальна кількість оперативною пам’яті, яка була задіяна для виконання методу дорівнює 458 МБ.
- Максимальне навантаження на процесор склало 961 Гц.
- Деякі контуру були пропущенні алгоритмом, але це не є суттєвим, якщо зображення буде мати вищу яскравість, то результати виділення контурів будуть кращі.

2.1.2. Оператор Собеля

Оператор Собеля один із самих відомих методів і може бути застосований для більшості задач. Даний оператор являє собою неточне, але наближене до методу побудови градієнту зображення, а точніше оператор Собеля використовує інтенсивності тільки в визначеній області 3x3 кожного пікселя для розрахунку наближеної відповідності градієнта зображення.

Оператор Собеля – це дискретний диференціальний оператор виділення границь, який об’єднує в собі гауссівське згладжування та диференціальний підхід. Розрахунок наближеного значення яскравості зображення використовуються в даному операторі. У відповідності з яскравістю області зображення, що аналізується, визначенні точки, які перевищують визначене порогове значення, рахуються, як границі об’єкту. Тобто оператор визначає край

по різниці в шкалі значення сірого кольору верхнього та нижнього пікселя, а також сусідніх лівих та правих пікселів, що досягають найвищого значення на границі об’єкта. Оскільки оператор Собеля об’єднує в собі згладжування та диференціювання, це надає дуже високу стійкість до шуму на зображеннях. Отже оператор Собеля краще використовувати для зображень з великою кількістю шумів та помірною шкалою сірого.

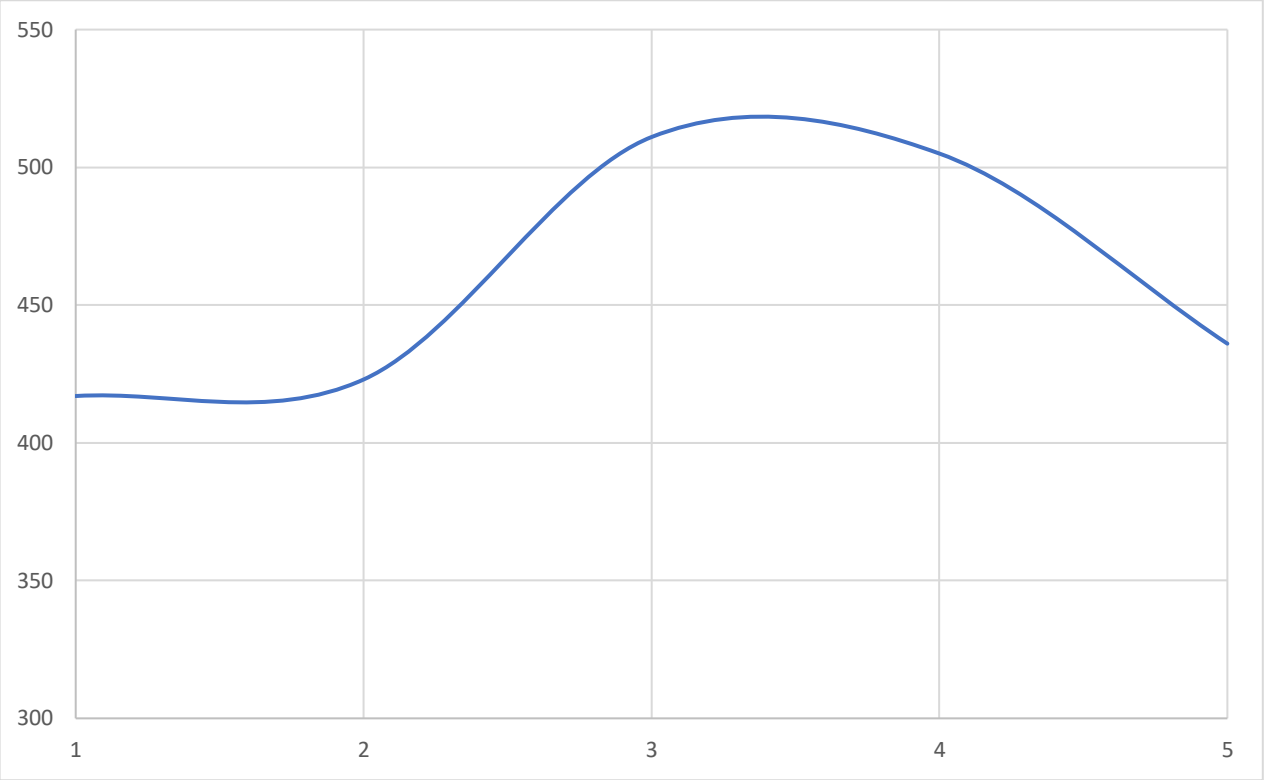

Рис. 2.4. Графік завантаженості оперативної пам’яті (МБ) під час виконання оператора Собеля.

Також відмінністю є використання тільки цілих чисел для значень вагових коефіцієнтів яскравості для оцінки градієнта.

G_x та G_y – це дві матриці, де значення кожної точки дорівнюють похідним по осям x та y відповідно. Ці значення точок розраховуються, як добуток та сума двох матриць, отриманні значення заносяться в піксель нового зображення з такими ж координатами, як і у оригінального зображення.

$$G_y = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

$$G_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

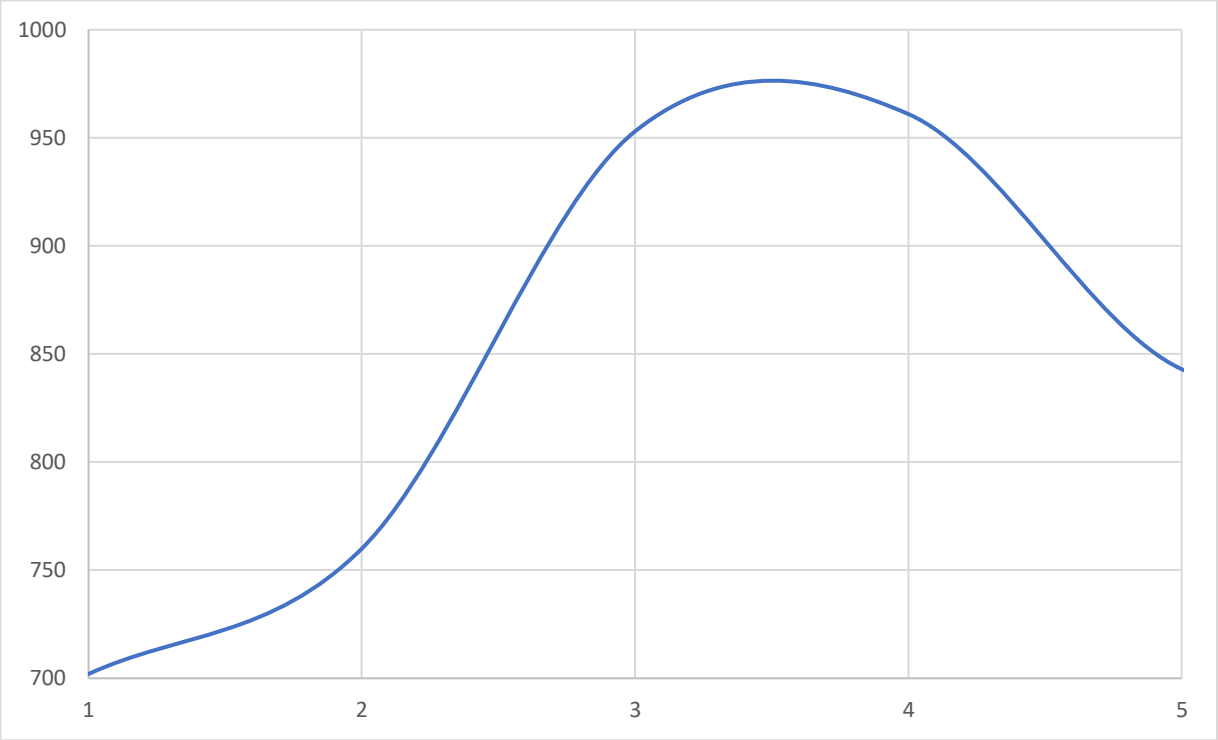

Рис. 2.5. Графік завантаженості процесора (Гц) під час виконання оператора Собеля.

Ще важливою особливістю даного оператор є наявність вагового коефіцієнта для значень середніх [37]:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Для перевірки роботи оператора Собеля використаємо те саме тестове зображення, результати відображено на рис. 2.4, рис. 2.5 та рис. 2.6.

Результати роботи оператора Собеля (Рис. 2.6).

Рис. 2.6. Оригінальне зображення (зліва). Результат виконання оператора Собеля (справа).

Проведений аналіз оператора Собеля дав такі результати:

- Максимальна кількість оперативною пам’яті, яка знадобилася для виконання алгоритму склала 505 МБ, що приблизно на 10% вище ніж результати тесту попереднього оператора.
- Максимальна частота процесора під час роботи алгоритму – 961 Гц, що є таким самим результатом, як і при тестуванні оператора Прюіта.
- Оператор Собеля більш чутливий до перепадів яскравості на областях зображення.

2.1.3. Оператор Кірша.

Особливістю оператора Кірша є використання коефіцієнтів для всіх значень крім середнього. Даний оператор визначає пікселі, що належать границі об’єкту по значенню максимального відгуку у кількох напрямах, тобто маски аналізують значення пікселів по 8 напрямкам і визначають вихідне значення відгуку. Особливістю є адаптивні порогові значення оператора, що дозволяє отримати максимальне можливе точне значення границі об’єкту.

Оператор Кірша використовує маску 3x3 [38]:

$$G_y = \begin{matrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{matrix}$$

$$G_x = \begin{matrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & 3 & -3 \end{matrix}$$

Результати аналізу методу наведені на рис. 2.7, рис. 2.8 та рис. 2.9.

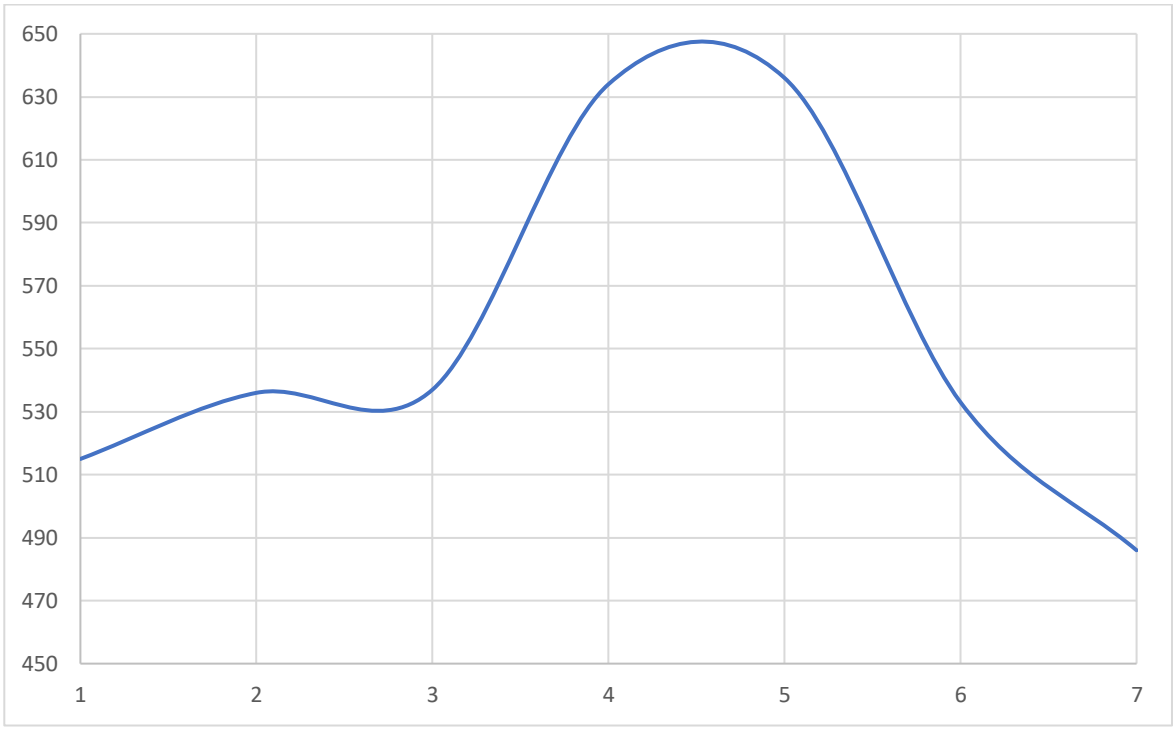

Рис. 2.7. Графік завантаженості оперативної пам’яті (МБ) під час виконання оператора Кірша.

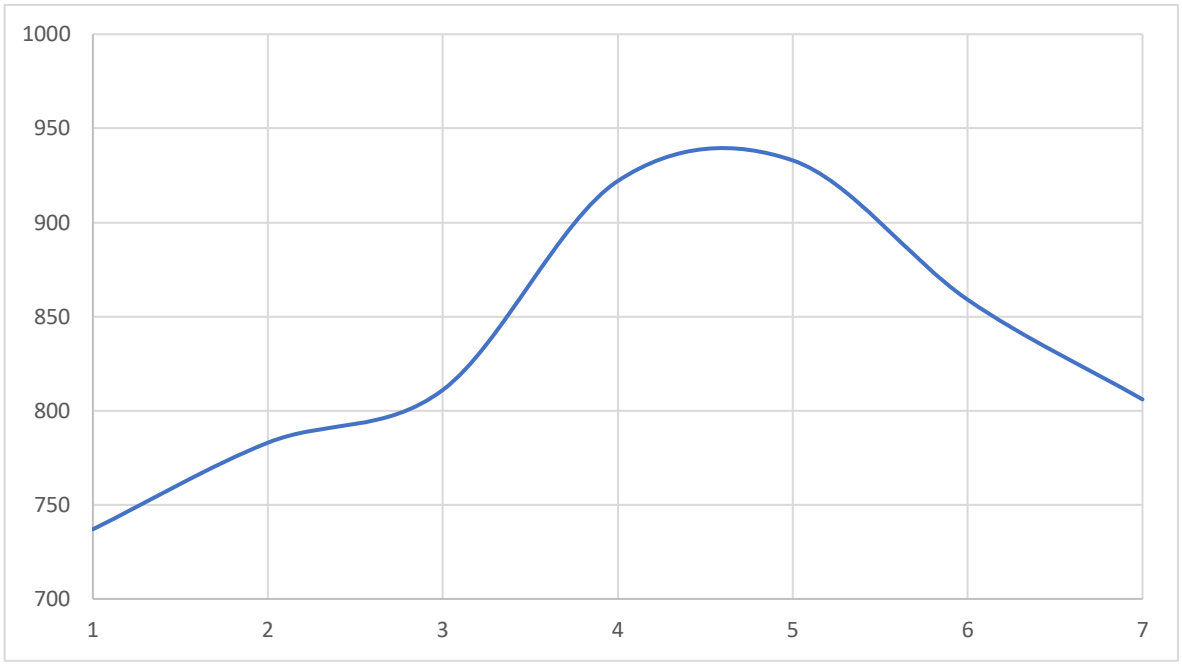

Рис. 2.8. Графік завантаженості процесора (Гц) під час виконання оператора Кірша.

Результати роботи оператор Кірша (Рис.2.9).

Рис. 2.9. Оригінальне зображення (зліва). Результат виконання оператора Кірша (справа).

Отриманні результати тестування оператор Кірша такі:

- Максимальне використання оперативною пам’яті склало 636 МБ, що дає змогу зробити висновок, що оператор має набагато більшу ємнісну складність ніж оператор Прюітта та Собеля.
- Максимальна частота процесора під час роботи алгоритму – 933 Гц, що також є достатньо високим показником затрат обчислювальних ресурсів.
- Вагові коефіцієнти надають оператору Кірша дуже високу чутливість до перепадів яскравості, але при цьому збільшують обчислювальну ємність алгоритму, тому даний алгоритм не підходить до виконання вимог, що були поставленні перед системою.

2.1.4. Оператор Лапласа (5x5).

Розглянемо оператор Лапласа з маскою просторової фільтрації 5x5. Даний метод є розширенням векторного оператора Лапласа і є диференціальним оператор другого порядку.

Базовий алгоритм описується такими кроками:

1. Визначення значення сірого кольору для центрального пікселя області, що аналізується і значення сірого для всіх інших пікселів, що належать даній області.

2. Якщо значення сірого для центрального пікселя вище, то значення центрального пікселя потрібно збільшити. В інакшому випадку значення сірого для центрального пікселю треба зменшити для досягнення збільшення різкості області на зображенні.
3. Розраховуються градієнти по чотирьом та восьми направленням векторів центрального пікселя, потім додаємо отриманні градієнти для визначення відношення між рівнем сірого для центрального пікселя та інших пікселів, що належать області аналізу.
4. На завершальному етапі потрібно провести згладжування значення сірого кольору пікселів за допомогою операції згортки градієнтів.

Якщо пікселі в області мають однакові значення рівня сірого, то результатом згортки буде 0. Позитивне значення згортки області досягається, коли значення рівня сірого центрального пікселя вище, ніж таке саме значення інших пікселів, що належать області. У випадку, якщо значення рівня сірого ц центрального пікселя нижче середнього рівня сірого інших пікселів, що належать області, тоді значення згортки буде негативним числом.

Результат операції згортки обробляється з відповідним коефіцієнтом ослаблення та додається до центрального пікселя для досягнення більшої різкості зображення.

Особливістю цього оператора є використання однакової маски, як для осі $\boldsymbol{x}$ так і для осі $\boldsymbol{y}$.

$$G_{(x,y)} = \begin{matrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 24 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{matrix}.$$

Дана маска описується, як сума різниць центрального елемента з кожним із 24 елементів матриці, це забезпечує рівномірність розподілу перепадів яскравості в усіх напрямках [39-40]. Результати аналізу методу наведено на рис. 2.10, рис. 2.11 та рис. 2.12.

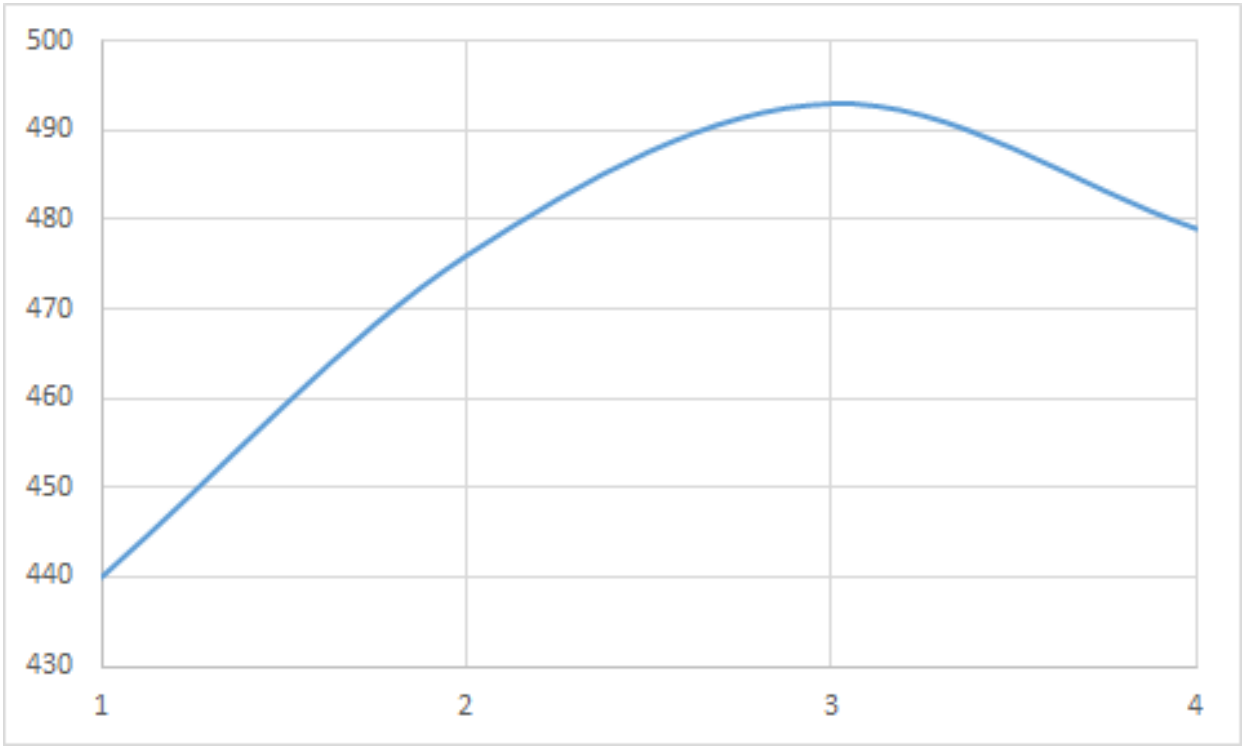

Рис. 2.10. Графік завантаженості оперативної пам’яті (МБ) під час виконання оператора Лапласа.

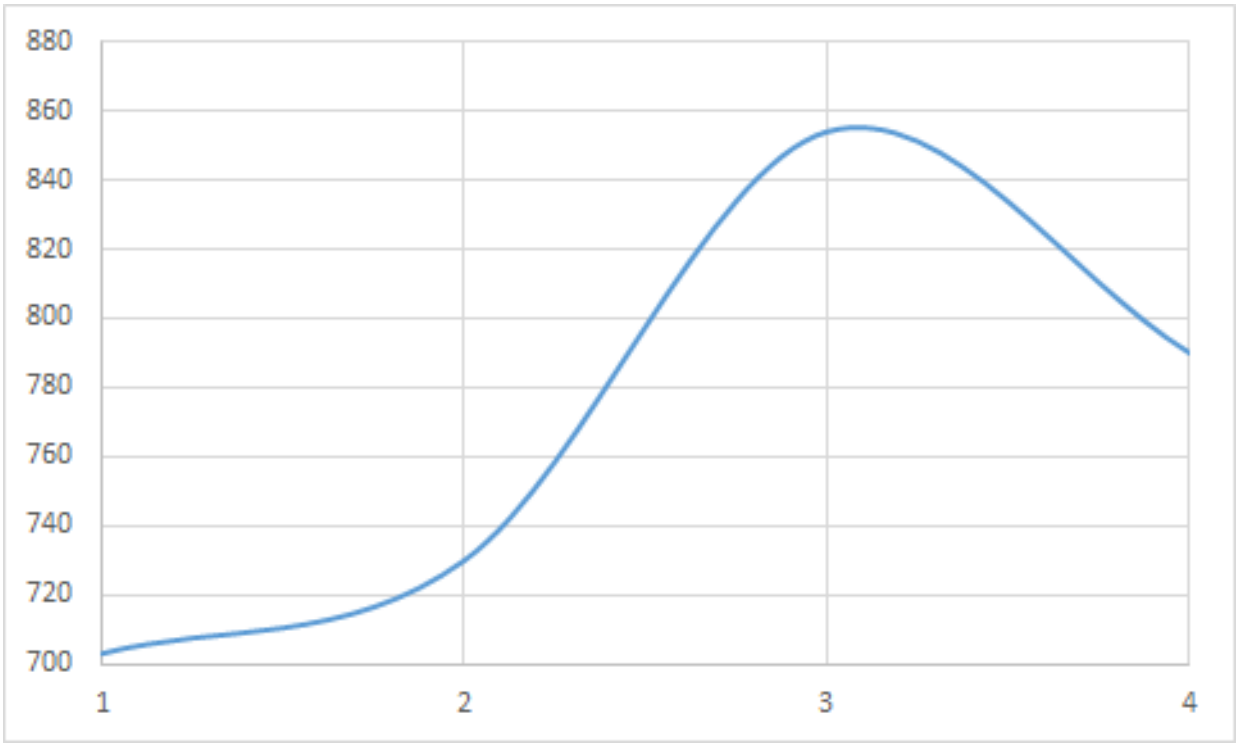

Рис. 2.11. Графік завантаженості процесора (Гц) під час виконання оператора Лапласа.

Рис. 2.12. Оригінальне зображення (зліва). Результат виконання оператора (справа).

Отриманні результати тестування оператора Лапласа такі:

- Максимальне використання оперативною пам’яті склало 493 МБ, що звичайно менше ніж при використанні оператора Собеля.
- Максимальна частота процесора під час роботи алгоритму – 857 Гц, що є найнижчим показником серед всіх операторів, які аналізувалися.
- Даний оператор дуже чутливий до шуму, це призводить до втрати частини інформації про напрям границь, що призводить до переривчастого виявлення границь, втрати контурів є набагато вищими ніж у оператора Прюітта. При цьому ємнісна складність оператора Лапласа менша приблизно така сама, як і в оператор Прюіта, що робить використання алгоритму виділення контурів за методом Прюіта більше доцільним.

Проаналізувавши отримані результати в ході експерементів з нелінійними методами контурного аналіз є можливість зробити такі висновки:

- Найменшу обчислювальну складність мав оператор Прюітта, але частину контурів даний метод пропустив. Тож цей алгоритм добре використовувати для зображень з високою яскравістю на всій області зображення, бо при даних умовах імовірність визначення контуру, що насправді не існує на вхідному зображенні менша, це зумовлене менш чутливою маскою.

- Найбільш ресурсо-затратним методом виявився оператор Кірша. Його перевага це чутлива маска, що дозволяє використовувати його на зображеннях, що мають низьку загальну яскравість.
- Оптимальним методом по ресурсо-затратам та знаходженню контурів є оператор Собеля, це зумовлено використанням маски с коефіцієнтами тільки для середніх значень [41–42].
- Оператор Лапласа використовував найменшу кількість обчислювальних ресурсів, але результати виділення контурів об’єктів були гірші за інші методи.

2.1.5. Згорткова нейронна мережа.

Особливістю згорткових нейронних мереж є те, що в них немає зв’язку між нейронами сусідніх шарів. Даний тип топології побудований на тому, що нейрони наступного шару аналізують частину зображення з попереднього шару та виділяють групи ознак для аналізу в наступних шарах штучної нейронної мережі. Це дозволяє згортковій нейронній мережі поступово виділяти окремі особливі ознаки об’єкту від малозначущих до найбільш важливих.

Наступні причини були вирішальними при виборі топології згорткової нейронної мережі для пошуку об’єктів на зображенні:

- зображення мають великий розмір;
- велика кількість параметрів і класів об’єктів;
- стійкість до зміни масштабу зображення, кутів зйомки камери та інших геометричних спотворень вхідного сигналу [43].

Розглянемо обрану архітектуру згорткової нейронної мережі:

- Вхідний шар. Буде містити вхідне значення пікселів зображення, розмірність шару [70x70x3].
- Згортковий шар. Розмір фільтра 7x7x3. Шар складатиметься з 6 площин з розмірами [64x64x1].

- Підвибірковий шар. Розмір маски 2x2. Він складається з 6 площин розміром [32x32x1].
- Вихідний шар. Розмір [1x1xN], де N – кількість кластерів у базі даних.

В якості функції активації обрано гіперболічний тангенс:

$$f(a) = A \cdot \tanh(S_a),$$

де f(a) - значення шуканого елемента, a - зважена сума сигналів попереднього шару, A - амплітуда цієї функції, S - її положення відносно опорної точки [44].

Переваги, що надає дана функція для вирішення проблеми:

- стандартна логістична функція поступається в швидкості конвергенції перед симетричними функціями активації, такими як гіперболічний тангенс
- функція має неперервну першу похідну;
- зменшення обчислювальної складності за рахунок обчислення простої похідної.

Таблиця 2.1. Результат розпізнавання об’єктів ЗНМ.

Назва класу	Кількість об’єктів у навчальній вибірці	Середній відсоток успішного розпізнавання
Автомобілі	16185	83%
Квіти	3670	68%
Яблука	2528	85%
Листя	6732	71%
Коти	12361	85%

Для проведення аналізу ефективності згорткової нейронної мережі було обрано 5 різних класів об’єктів. Зразки для яких було взято з відкритого доступу проекту TensorFlow [45]. Кожен клас для навчання містив від 2528 до 16185 зображень зразків об’єктів розміром 70x70 пікселів. Для перевірки точності розпізнавання було обрано вибірки по 1000 зразків для кожного класу. Результати наведені в таблиці 2.1.

Аналіз отриманих результатів дає змогу зробити наступні висновки:

- Найнижчий результат було отримано для класу «Квіті», це зумовлено тим, що сама навчальна вибірка була невеликою, а зразки квітів належали різним видам.
- На точність розпізнавання класу «Автомобілі» також вплинуло те, що в вибірці були представлені різні марки та різні типи кузовів, однак через велику кількість зразків навчальної вибірки кінцевий результат точності був вищий.
- Клас «Яблука» мав найвищу точність розпізнавання, це можливо пояснити тим, що зразки об’єктів класу мало відрізнялися по структурі та формі.

2.1.6. Повнозв’язна нейронна мережа.

Повнозв’язна нейронна мережа прямого розподілення – це мережа, в якій кожен нейрон пов’язаний з усіма іншими нейронами, що знаходяться в сусідніх шарах. В такому типу нейронних мереж всі зв’язки спрямовані від вхідних нейронів до вихідних [84].

Для початку потрібно вирішити, як подавати дані на вхід. Найпростіше і майже безальтернативне рішення для ПНМ – це виразити двомірну матрицю зображення у вигляді одномірного вектору. Тобто для зображення розміром 70x70 у нас буде 4900 вхідних нейронів, що вже є дуже високим показником. Тепер враховуючи правила побудови архітектури повнозв’язної нейронної мережі, яке вимагає мати в прихованому шарі кількість нейронів на порядок більше ніж у вхідному. Також потрібно врахувати, що перетворення об’єктів на зображенні в ідентифікацію класу достатньо складне та нелінійне, то потрібно використовувати більше ніж один прихований шар. Якщо взяти до уваги ці вимоги до побудови ПНМ, то будемо мати таки характеристики нейронної мережі:

- Вхідний шар 4900 нейронів.
- Перший прихований шар 50000 нейронів.
- Другий прихований шар 25000 нейронів.
- Вихідний шар N нейронів, де N – кількість кластерів у базі даних.

Для такої конфігурації кількість зв’язків, що навчаються, буде 245 млн. між входами і першим прихованим шаром 12,5 млрд. між першим і другим прихованими шарами. Враховуючи, що нейрона мережа буде мати N-кількість виходів, отже тоді кількість зв’язків буде N·3.06e+17. При навчанні для кожного зв’язку потрібно буде обчислювати градієнт помилки.

Розрахунок ваг нейронів буде проводитися за допомогою методу градієнтного спуску, бо важливим буде пошук локальних мінімумів та максимумів. Важливим моментом є те, що потрібно порахувати похідну помилки по вагам. Таким чином, ми можемо обчислити, куди нам потрібно змістити ваги, щоб у нас гарантовано зменшилася помилка при наступному кроці навчання. Якщо зробити крок дуже широким, то імовірність пропустити локальний мінімум є дуже високим, що дуже суттєво знизить ефективність нейронної мережі [85].

Таблиця 2.2. Результат розпізнавання об’єктів ПНМ.

Назва класу	Кількість об’єктів у навчальній вибірці	Середній відсоток успішного розпізнавання
Автомобілі	16185	72%
Квіти	3670	57%
Яблука	2528	71%
Листя	6732	59%
Коти	12361	73%

Проведемо перевірку роботи повнозв’язної нейронної мережі на тому самому наборі даних, результати представленні в таблиці 2.2. Виходячи з отриманих результатів є можливість зробити висновок, що повнозв’язна нейрона мережа працює гірше ніж згорткова. Це зумовлене тим, що коли ПНМ перетворює зображення в лінійний ланцюжок байтів, безповоротно втрачаються деякі важливі ознаки, і при цьому з кожним шаром ця втрата лише посилюється. Втрата топології зображення, тобто взаємозв’язок між окремими частинами зображення. Постановка завдання розпізнавання в даній роботі має висуває до нейронної мережі характеристику стійкості до зрушень, поворотів та змін

масштабу зображення, тобто нейронна мережа повинна витягувати з даних інваріанти, які залежать від ракурсу об’єкту на зображенні та його масштабу. Також враховуючи кількість зв’язків в ПНМ, які потрібно розраховувати, дана топологія має дуже високу ємнісну складність, що суперечить поставленим вимогам до системи. Аналізуючі отриманні результати найкращим вибором буде використання згорткової нейронної мережі для розпізнавання об’єктів на зображенні. Однак потрібно вирішити проблему кількості нейронів у вихідному шарі, тому що кількість класів об’єктів в системі буде додаватися з часом використання системи користувачами.

2.2. Розроблення моделі класифікації та кластеризації об’єктів.

Щоб знайти об’єкт у зображенні, база даних об’єктів повинна існувати. Наповнення бази планується за рахунок користувачів системи, які будуть обирати об’єкти для подальшої анімації. Для таких об’єктів система повинна вибрати ознаки, класифікувати об’єкт і зберегти його в базі даних. База даних класифікованих об’єктів у майбутньому використовуватиметься для пошуку схожих об’єктів на зображенні та дозволить користувачеві автоматично вибирати об’єкти для подальшої анімації.

Тому визначення об’єкта на зображенні має бути стійким до повороту об’єкта, масштабу зображення та різноманітності ракурсів зображення. Вибираються такі ознаки об’єкта:

- Визначення площі та периметра;
- Визначення радіусів вписаного та описаного кіл;
- Означення сторін описаного прямокутника;
- Визначення кількості та взаємного розташування кутів;
- Побудова гістограми градієнтів об’єктів.

Стандартна формула для *розрахунку площі об’єкта* розраховується шляхом підрахунку кількості елементів, пов’язаних з об’єктом. У разі надходження зображень до системи, площа може бути розрахована за кількістю пікселів у області зображення.

$$S = \sum_x \sum_y \begin{cases} 1, A(x,y) \in O \\ 0, A(x,y) \notin O' \end{cases}$$

де O — множина масиву пікселів A(x, y), що належать об’єкту [46].

Щоб знайти *периметр* об’єкта *P*, достатньо порахувати кількість пікселів, що належать контуру об’єкта. Для забезпечення інваріантності до масштабу об’єкта додається нормований знак U.

$$U = \frac{S}{P^2}.$$

Визначення радіусів вписаного та описаного кіл. Спочатку потрібно визначити геометричний центр об’єкта.

$$C_{(x,y)} = \left(\frac{\sum_x \sum_y xA(x,y)}{\sum_x \sum_y A(x,y)}, \frac{\sum_x \sum_y yA(x,y)}{\sum_x \sum_y A(x,y)} \right),$$

де *x* і *y* — номери рядків і стовпців усіх пікселів *A(x,y)*, що належать об’єкту.

На цьому етапі можна визначити радіуси вписаного та описаного кіл. Це робиться шляхом вибору мінімальної та максимальної довжини вектора, який починається в центрі зображення і закінчується в точках всередині периметра.

$$R_{max} = \max_{0 \le i \le N,} \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2},$$

$$R_{min} = \min_{0 \le i \le N,} \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2},$$

де x_c і y_c – координати центру, *N* – кількість пікселів по периметру, $(x_i, y_i) \in P$ [47]. Нормований параметр *R`* додається до формули.

$$R` = \frac{R_{max}}{R_{min}}.$$

Визначення сторін описаного прямокутника. Визначте максимальне та мінімальне значення абсцис та ординат зображення об’єкта x_{max} і x_{min}, y_{max} і y_{min}, а потім визначте висоту *H* і довжину *L* прямокутника:

$$L = \max_{0 \le i \le N,} x_i - \min_{0 \le i \le N,} x_i;$$

$$H = \max_{0 \le i \le N,} y_i - \min_{0 \le i \le N,} y_i,$$

де N – кількість пікселів по периметру, а $(x_i,y_i) \in P$ [9]. Нормований параметр P визначається за формулою.

$$P = \frac{L}{H}.$$

Визначення кількості та взаємного розташування кутів. Для цього оцініть відстань l між початковою і кінцевою точками фрагмента контуру.

$$l_i = x_i|x_{i-2} - x_{i+2}| + |y_{i-2} - y_{\,i+2}|.$$

Потім перевіряється умова $l_i \leq C$, де C — умовне порогове значення. Визначення даного показника потрібно буде реалізувати на основі минулих властивостей об'єкта. Якщо умова виконується, то точка належить до множини кутових точок [48].

Побудова гістограми градієнта об'єкта. Основною ідеєю є припущення, що зовнішній вигляд і форму об'єктів в області цифрового зображення можна описати розподілом градієнтів інтенсивності. Розраховується напрямок градієнта кожного пікселя. Для двовимірної функції f(x,y) вектор градієнта має такий вигляд:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right].$$

Часткові похідні функцій інтенсивності в координатах x і y є оцінками контрасту в напрямку відповідних осей координат [49]. В якості точної оцінки контрасту в напрямку відповідної осі координат можна усереднювати три різні оцінки контрасту навколо пікселя формул:

$$\nabla_x = \frac{\partial f}{\partial x} \approx \frac{1}{3}\left(\frac{f[x+1,y]-f[x-1,y]}{2} + \frac{f[x+1,y-1]-f[x-1,y-1]}{2} + \frac{f[x+1,y+1]-f[x-1,y+1]}{2}\right),$$

$$\nabla_y = \frac{\partial f}{\partial y} \approx \frac{1}{3}\left(\frac{f[x,y+1]-f[x,y-1]}{2} + \frac{f[x-1,y+1]-f[x-1,y-1]}{2} + \frac{f[x+1,y+1]-f[x+1,y-1]}{2}\right).$$

Для оцінки даних необхідно дотримуватись мінімізації зображення маски оператора контрасту об'єкта. Для зниження обчислювальної складності найкраще використовувати диференціально-вимірну маску [50].

Після підрахунку градієнта інтенсивності слід розділити зображення на комірку та побудувати гістограму об’єкта градієнтів для кожного кишені осередку *G* модуль гістограми відповідає градієнту інтенсивності в точці.

$$G = \sqrt{\nabla_x{}^2 + \nabla_y{}^2}.$$

Потім проводять згладжування гістограм у клітинках, щоб забезпечити інваріантність відносної зміни яскравості та контрасту.

Виділення ознак об’єкта. Щоб знайти подібність між об’єктами, будемо обчислювати семантичну відстань між ознаками.

$$D_{ij} = \sqrt{\sum_{k=1}^{N}(X_{ik} - X_{jk})^2},$$

де D_{ij} – евклідова відстань між точками i і j, які характеризують одну ознаку різних об’єктів. N – найменша кількість значень, що враховуються для ознаки між двома об’єктами, k –- порядковий номер значення ознаки.

Процедура визначення кількості кластерів складається з наступних кроків:

1. Запускається алгоритм k-середніх для K кластерів і відповідна внутрішня дисперсія
   $ð_k = \sum_{i=1}^{N}\sum_{x\in c_i}\|x - m_i\|^2$. x – вектор, що характеризує об’єкт, що входить до кластера сі, загальна відстань між об’єктами та центром кластера. Параметр m_i є центром поточного кластера. Для різних значень K будується множина $ð_k$.
2. Вибір ступеня перетворення $Y = \frac{p}{2}$.
3. Обчисліть стрибки за формулою $J_k = {ð_k}^{-Y} - {ð_{k-1}}^{-Y}$.
4. Для остаточної кількості кластерів виберіть значення, що дорівнює максимальному [51].

2.3. Розроблення методу анімування статичних об’єктів на зображеннях.

Для забезпечення процесу анімації потрібно виконувати паралельний перенос об’єкта на додаткову площину, тому потрібно ввести третю координату, що визначатиме номер паралельної поверхні *(P)* куди буде розташовуватися новий перетворений об’єкт, отже тоді зображення потрібно розглядати у тривимірному просторі. Таким чином це дає можливість робити копію виділеного об’єкту за допомогою використання паралельного переносу на визначену поверхню. Враховуючи ці данні отримуємо координати центру скопійованого об’єкту:

$$(x^{'}, y^{'}, z^{'}) \rightarrow (x + \Delta x, y + \Delta y, z + \Delta z),$$

де Δx, Δy та Δz є нормальними векторами початкової площі зображення.

Кількість об’єктів, що піддаватимуться ефекту анімації будуть визначати кількість паралельних площин. [52].

Повторення афінних перетворень, а саме поворот та перенос надають змогу досягти створення ефекту анімації. Опишемо даний алгоритм більше детально. Поворот точки відносно початку координат:

$$(x^{'}, y^{'}) \rightarrow (x \cdot cos(\varphi) - y \cdot sin(\varphi), x \cdot sin(\varphi) + y \cdot cos(\varphi)).$$

Перенос точки:

$$(x^{'}, y^{'}) \rightarrow (x + m_x, y + m_y).$$

Важливим компонентом для створення ефекту анімації є її напрям [53]. Отже для визначення напрямом анімації потрібен вектор *A(a, b)*. Відносно даного вектору потрібно буде виконувати поворот та перенос точок об’єктів, що піддаються ефекту анімації.

Тоді початковим кроком, для створення ефекту анімації виділеного об’єкту, потрібно вирішити задачу повороту об’єкту навколо точки $A(x_A, y_A)$. Якщо накласти вектор на формули перенесення та повороту точки [54] вийде дана матриця перетворень:

$$(x^{'}, y^{'}, 1) \rightarrow (x, y, 1) \cdot$$

$$\begin{bmatrix} cos(\varphi) & sin(\varphi) & 0 \\ -sin(\varphi) & cos(\varphi) & 0 \\ -x_A \cdot cos(\varphi) + y_A \cdot sin(\varphi) + x_A & -x_A \cdot sin(\varphi) + y_A \cdot cos(\varphi) + y_A & 1 \end{bmatrix},$$

тоді кінцевий вираз для отримання нових координат буде:

$$\begin{cases} x^{'} = x_A + x \cdot cos(\varphi) - x_A \cdot cos(\varphi) - y \cdot sin(\varphi) + y_A \cdot sin(\varphi) \\ y^{'} = y_A + y \cdot cos(\varphi) - y_A \cdot cos(\varphi) - x \cdot sin(\varphi) + x_A \cdot sin(\varphi) \end{cases}.$$

Для отримання ефекту анімації об’єкту потрібно, щоб точки, що йому належать, змінювали позицію за визначений проміжок часу та із визначеною швидкістю. Враховуючи ці особливості координати точки будуть описуватися функцією залежності від часу *t*: $(x^{'}(t), y^{'}(t))$ та $(x(t), y(t))$ відповідно, а отже, тоді формула перенесення точки на вектор отримає вигляд:

$$\begin{cases} x^{'}(t) = x_A + x(t) \cdot cos(\varphi) - x_A \cdot cos(\varphi) - y(t) \cdot sin(\varphi) + y_A \cdot sin(\varphi) \\ y^{'}(t) = y_A + y(t) \cdot cos(\varphi) - y_A \cdot cos(\varphi) - x(t) \cdot sin(\varphi) + x_A \cdot sin(\varphi) \end{cases}.$$

Дана формула тепер схожа на рівняння закону руху точки в координатній площині, тому вводимо такий параметр, як швидкість руху υ, для отримання рівномірної анімації, прискорення буде сталим. На цьому кроці потрібно визначити швидкість руху точки, це можна зробити знаючи відстань, яку має подолати точка за визначений час *T*. Тому відстань буде задаватися відрізком *a* на векторі *A(a, b)*. Тоді маємо рівняння швидкості для точки:

$$\upsilon = \frac{\sqrt{(x_{a_2} - x_{a_1})^2 + (y_{a_2} - y_{a_1})^2}}{T}.$$

Як вже визначалось, що точка буде рухатися рівномірно, тоді розрахувати координати точки в заданий час *t* можна за формулами:

$$x^{'}(t) = x_{a_2} - x_{a_1} + x + \frac{\sqrt{(x_{a_2} - x_{a_1})^2 + (y_{a_2} - y_{a_1})^2}}{T} \cdot t \cdot \cos(\varphi) - x_{a_2} -$$

$$- x_{a_1} \cdot \cos(\varphi) - y + \frac{\sqrt{(x_{a_2} - x_{a_1})^2 + (y_{a_2} - y_{a_1})^2}}{T} \cdot t \cdot sin(\varphi) + y_{a_2} - y_{a_1} \cdot sin(\varphi)^{'},$$

$$y^{'}(t)=y_{a_2}-y_{a_1}+y+\frac{\sqrt{(x_{a_2}-x_{a_1})^2+(y_{a_2}-y_{a_1})^2}}{T}\cdot t\cdot cos(\varphi)-y_{a_2}-$$
$$-y_{a_1}\cdot cos(\varphi)-x+\frac{\sqrt{(x_{a_2}-x_{a_1})^2+(y_{a_2}-y_{a_1})^2}}{T}\cdot t\cdot sin(\varphi)+x_{a_2}-x_{a_1}\cdot sin(\varphi)\ .$$

В формулах присутня достатня кількість тригонометричних функцій, тому потрібно зменшити час операцій машинних розрахунків цих функцій. Для цього використаємо розрахунок збіжності ряду Тейлора по заданій похибці ε [55]:

$$sin(\varphi) = \sum_{i=0}^{\infty}(-1)^i \cdot \frac{\varphi^{2i+1}}{(2i+1)!}.$$

Знаючи *sin(φ)* можемо знайти *cos(φ)*:

$$cos(\varphi) = \pm\sqrt{1 - sin(\varphi)^2}.$$

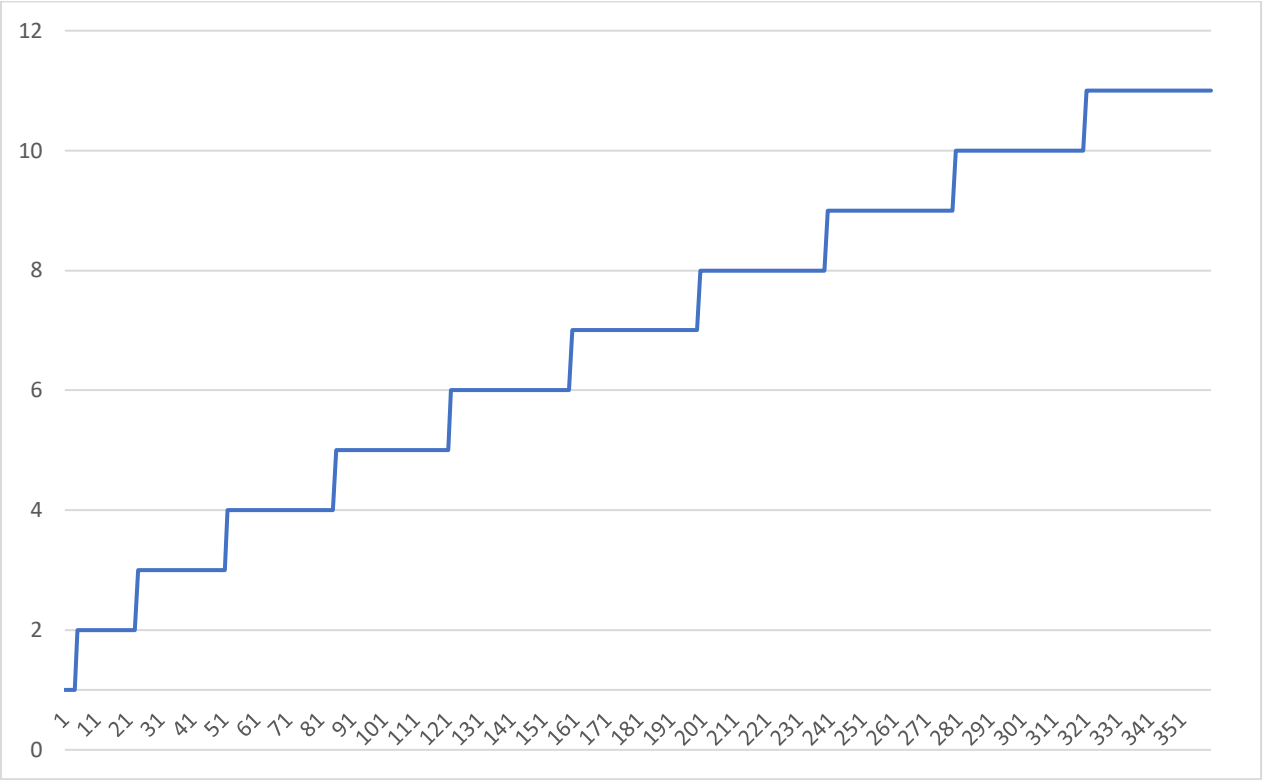

Рис. 2.13. Кількість ітерацій ряду Тейлора.

Достатня точність для розрахунку буде $\varepsilon = 0{,}0001$ [56], тепер можемо розрахувати кількість ітерацій для знаходження значення $sin(\varphi)$ (Рис. 2.13).

Отриманні результати показують, що максимальна кількість ітерацій буде 11, проаналізуємо відрізок [1°, 90°], на даному відрізку максимальна кількість ітерацій 5. Отже можемо застосувати тригонометричні тотожності [57]:

$$sin(\varphi) = sin(180 - \varphi),$$

$$-sin(\varphi) = sin(360 - \varphi),$$

це дозволило отримати максимальну кількість ітерацій 5 при розрахунку усіх кутів на відрізку [1°, 360°] (Рис. 2.14).

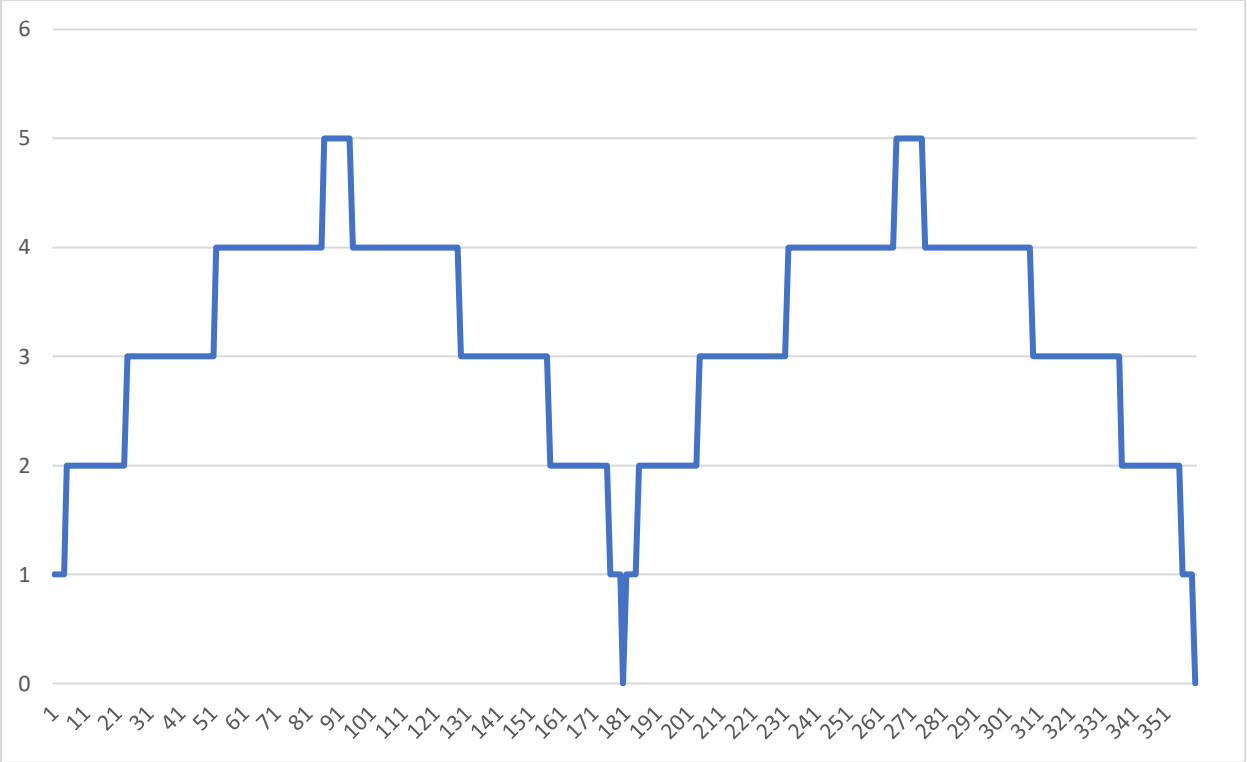

Рис. 2.14. Кількість ітерацій ряду Тейлора при застосуванні тригонометричних тотожностей.

Проаналізуємо обчислювальну складність алгоритму для створення ефекту анімації для обраного об'єкту розміром *n* пікселів. Створення копії об'єкта на новій паралельній площині P, накладає необхідність копіювання кожного пікселя з координатами *(x, y)*. Для виконання цієї процедури потрібно *N* ітерацій. Визначення координат точки в момент часу *t* вимагає *N* + *M* ітерацій, де *M* – кількість ітерацій потрібних для розрахунку значення *sin(φ)*. Врахуємо, що для відтворення анімації плавно потрібно створювати відео, що буде містити 30

кадрів в секунді, тому щоб розрахувати позиції точки на протязі часу *T*, отримаємо *K*, де *K = 30·T*. Отже формула для розрахунку складності *S* для створення ефекту анімації одного об'єкту буде:

$$S = 30TNt_n + Mt_m,$$

де t_n - час на виконання одної ітерації обчислення позиції точки;

m кількість ітерацій потрібних для розрахунку значення *sin(φ)*;

t_m - час на виконання розрахунку одного члена ряду Тейлора.

Тоді для декількох об'єктів:

$$S = 30\sum_{i=2}^{O} TN_i t_n + M_i t_m,$$

де *O* - кількість об'єктів.

Властивості афінних перетворень, що вплинули на вибір даного методу для створення ефекту анімації об'єктів, такі: можливість зберігати пропорції паралельних об'єктів, а саме довжину відрізків на паралельних прямих і площ на паралельних площинах. Також через той факт, що добуток двох афінних перетворень, теж є афінним, тому комбінування методів дозволяє не змінювати ознаки об'єкту при створенні ефекту анімації.

Висновки до другого розділу.

Було проаналізовано ємнісну складність таких алгоритмів контурного аналізу, як оператора Пріютта, оператора Собеля, оператора Кірша та оператора Собеля. Отриманні результати аналізу були таким, що оператор Кірша виявився найбільш затратним на використання ресурсів при обчисленнях. Оператор Лапласа показав оптимальну ємнісну складність, але досить низьку ефективність виділення границь, було втрачено достатньо суттєва кількість контурів об'єктів на зображенні. Тому було обрано два методи для використання в системі, а саме оператори Собеля та Прюітта. Данні оператори показали середню обчислювальну складність серед всіх чотирьох алгоритмів, при цьому достатньо хороші результати при виділенні контурів, основний момент, який потрібно буде врахувати надалі при реалізації системи, що оператор Прюітта добре працює з зображеннями, що мають вищу яскравість.

Було проаналізовано дві топології штучних нейронних мереж: згорткової нейронної мережі та повнозв’язної нейронної мережі. Результати тестування проводилися на однакових тренувальних вибірках зображень роздільною здатністю 70x70 пікселів. Тренувальний набір складався з 5 класів, що містили в собі від 2528 до 16185 зображень Для перевірки точності розпізнавання було обрано вибірки по 1000 зразків для кожного класу. Отриманні результати дали змогу зробити висновок, що використання ПНМ є не доцільним з урахуванням ємнісної складності даної топології, тому в системі для реалізації функції пошуку об’єктів на зображенні буде використана топологія ЗНМ.

Розроблено метод кластеризації та класифікації об’єктів на зображені за допомогою виділення таких ознак: площа та периметр об’єкту; радіусу вписаного та описаного кіл; означення сторін описаного прямокутника; кількість та взаємне розташування кутів; гістограма градієнту.

Розроблено метод анімування статичних об’єктів на основі використання афінних перетворень. Суть метода полягає в повторенні таких афіних перетворень, як поворот та перенос. Дані перетворення об’єкту виконуються відносно заданого вектору анімації за визначений час відображення анімації. Також здійснено оптимізацію обчислювальної складності розробленого методу анімації за рахунок зменшення кількості ітерацій, що потрібні для розрахунку позиції точки в заданий час.

РОЗДІЛ 3. РОЗРОБЛЕННЯ АЛГОРИТМІВ ПОШУКУ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ, ЇХ КЛАСИФІКАЦІЇ ТА КЛАСТЕРИЗАЦІЇ, АНІМУВАННЯ СТАТИЧНИХ ОБ'ЄКТІВ.

У розділі було розроблено алгоритми пошуку об'єктів, кластеризації та класифікації, алгоритм анімування статичних об'єктів. Усі алгоритми було зображено за допомогою блок-схем.

3.1. Розроблення алгоритму пошуку об'єктів.

Кожен об'єкт на зображенні характеризується певними ознаками, ці ознаки можливо отримати після виділення границь всіх об'єктів на зображенні. Для цього буде використовуватися оператор Собеля або оператор Прюітта, вибір методу буде залежати від середньої яскравості зображення. Далі потрібно зробити попередню обробку зображення за допомогою бінарізації. На завершенні маємо отримати зображення, яке буде характеризуватися такими параметрами, як згладжена яскравість, градієнти яскравості та контури або частини контурів об'єктів. Кожен об'єкт на зображенні повинен мати ознаки, які будуть його характеризувати прямо чи опосередковано [58]. Отже для початку потрібно вирахувати середню яскравість зображення *AD*. Для цього потрібно порахувати середнє значення *RGB* кожного пікселя зображення, далі вирахувати середнє для всіх пікселів.

$$AD = \frac{\sum_{i=1}^{n} \frac{R_i + G_i + B_i}{3}}{n},$$

де *n* – кількість пікселів в зображенні.

Значення для кожного показника *RGB* пікселя для сірого дорівнює 128, тому приймаємо, що зображення з *AD >= 128* є яскравішими за зображення з *AD < 128*.

Для бінарізації зображення буде використовуватися алгоритм адаптивної порогової обробки. Такий вибір зумовлений тим, що на вхід будуть подаватися різноманітні зображення, як по якості так і по загальній експозиції. Даний алгоритм базується на порівнянні рівнів яскравості пікселя, що перетворюється,

зі значеннями локальних середніх пікселів, що знаходяться в сусідній області. Пікселі обробляються по черзі, яскравість пікселя порівнюється з середнім значенням яскравості в сусідній області [59]. Основна проблема бінарізації зображень – це вибір вірного порогового значення. Критерії оптимальності даного значення пов’язанні з забезпеченням мінімального викривлення форми об’єктів на зображенні. Якщо поріг нижче оптимального, форма зображення об’єкта спотворюється з допомогою сильного впливу фонових шумів. При цьому контур зображення розширюється внаслідок включення до складу ділянок фону. Коли поріг вище оптимального, форма зображення спотворюється через те, що значна частина точок цього зображення віднесена до фону [60].

Отже, важливим є вибір адекватного порогу, тому для того, щоб покращити вибір даного значення потрібно виділяти на зображенні ті пікселі, які лежать близько до перепадів між об’ктом та фоном та на кордоні різкого перепаду яскравості. Тому, якщо враховувати зазначені специфікації, тоді імовірність, що такий піксель належить об’єкту буде вищий імовірності його знаходження на фоні.

Першим кроком алгоритму буде розрахунок середньої яскравості зображення, щоб обрати метод виділення границь. Якщо середня яскравість зображення буде вищою за 128, то буде обрано оператор Прюітта, в іншому випадку – оператор Собеля.

Другим кроком алгоритму буде виділення границь за допомогою оператор Собеля або оператор Прюітта. Найбільш точні граничні значення можна отримати при аналізу значення яскравості пікселів, що лежать на виділених границях об’єкту.

Пропонується використовувати статистичне середнє значення яскравості пікселів виділених границь

Третім кроком алгоритму буде сегментація зображення при використанні отриманих порогових значень. Важливим є виділення не всіх пікселів, що перевищують порогове значення, а саме групи пікселів, що належать конкретним об’єктам. Для сегментації буде використовуватися алгоритм заповнення замкнутих областей. За допомогою даного методу виділяємо всі пікселі, що

обмежуються границями об’єкту та для яких значення яскравості перевищує або дорівнює пороговому.

Рис. 3.1. Узагальнений алгоритм пошуку об’єктів.

Окремо визначимо процес сегментації зображення, адже він відіграє важливу роль для подальшого використання отриманих об’єктів для роботи алгоритму класифікації та кластеризації. Як згадувалося вище для сегментації

буде використовуватися алгоритм заповнення областей. Цей алгоритм виходить з таких критеріїв, як те, що є заданий визначений піксель, який знаходиться в заданому контурі та існує критерій належності пікселя визначеній області. В даному випадку координати пікселя мають належати площині, що визначається границею об’єкта. Отже даний алгоритм буде поміщати пікселі, що аналізуються в стек, під стеком треба розуміти масив в який будуть поміщатися елементи та послідовно вилучатися за принципом “first in – last out”. Отже загальний алгоритм буде виглядати наступним чином:

- Помістити визначений піксель в стек.
- Поки стек не порожній:
    1. Вилучити піксель зі стеку
    2. Ініціалізувати піксель
    3. Перевірити чи кожний сусідній піксель відповідає описаним критеріям та чи було його ініціалізовано, якщо ні то такий піксель помістити в стек.

Усі виділенні границі об’єктів будемо рахувати багатогранниками, тоді для оптимізації алгоритму потрібно буде визначити менший прямокутник, який належить площині багатогранника, що описується контуром виділеної границі об’єкту. Тоді маємо багатогранник, який обмежує область, заданий списком вершин і ребр.

Отже, для кожного рядка визначаються точки перетину з ребрами багатогранника, які потім упорядковуються координатою x, формуючи інтервали між парами перетинів. Визначення того, який інтервал є внутрішнім для багатогранника, а який ні є досить простим логічним завданням. При цьому якщо рядок проходить через вершину багатогранника, це перетин має бути враховано двічі у випадку, коли вершина є точкою локального мінімуму чи максимуму.

Відповідно до зазначено вище, заповнювати область об’єкту будемо за допомогою модифікованого алгоритму Брезенхема. Для цього потрібно сформувати список відрізків, що формують багатогранник [61]. Побудову починаємо з ініціалізації масивів X_{min} та X_{max}. При цьому масив X_{max}

заповнюється нулями, а масив X_{min} – числом N, яке дорівнює кількості пікселів області, що аналізується, по горизонталі. Потім випадковим чином обираємо значення Y_{min} та Y_{max}.

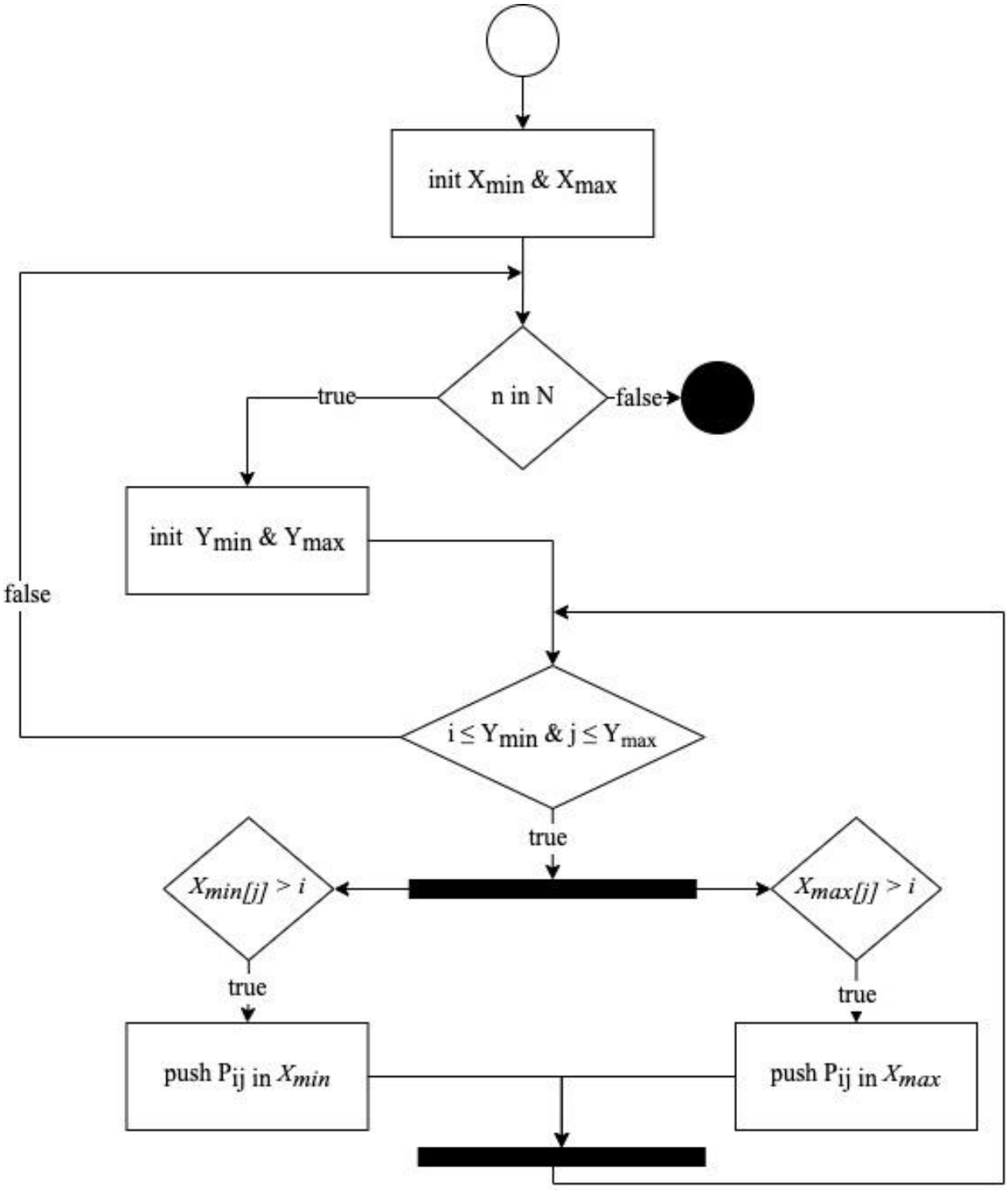

Рис. 3.2. Алгоритм виконання сегментації.

Наступним кроком потрібно заповнити масиви X_{min} та X_{max} границями відрізків багатогранника, для цього потрібно при кожному переході до пікселя P_{ij}, де $i \leq Y_{min}$ та $j \leq Y_{max}$, тоді при формуванні відрізку процес ініціалізації пікселя буде таким:

- Якщо $X_{min[j]} > i$, то додаємо координату i в масив X_{min}.
- Якщо $X_{max[j]} > i$, то записуємо координату i в масив X_{max}.

На цьому кроці алгоритму, щоб отримати потрібно послідовно застосувати алгоритм Брезенхема до всіх сторін багатокутника, отримуємо заповнені масиви границь.

На останньому кроку потрібно ініціалізувати всі пікселі, що належать до відрізків $\{(X_{min[j]},j),(X_{max[j]},j)\}$. Для всіх кожного визначеного пікселя, що належить об'єкту потрібно розрахувати напрямок градієнта, цей крок потрібен для алгоритму класифікації та кластеризації об'єктів.

3.2. Розроблення алгоритму класифікації та кластеризації об'єктів.

Як зазначалося в попередніх підрозділах, для того, щоб провести класифікацію та кластеризацію об'єкта нам потрібно виділити його ознаки: площу та периметр, радіусів вписаного та описаного кіл, сторін описаного прямокутника, кількості та взаємного розташування кутів, градієнтну гістограму об'єкта [62].

Отже першим кроком алгоритму буде розрахунок ознак всіх знайдених об'ктів на зображенні. Тоді нехай N – кількість об'єктів на зображенні, r об'єкт, що аналізується, S – площа об'єкта, P – периметр, R_{max} – радіус описаного кола, R_{min} – радіус вписаного кола, H та L – відповідно висота та довжина описаного прямокутника.

Наступним кроком йде визначення кількості та взаємного розташування кутів, який виражено масивом *A*. На цьому етапі важливим є правильний розрахунок параметра C, яке є умовним пороговим значенням визначення розташування кутів [63]. Враховуючи, що на цьому етапі потрібно рахувати відстань між точками контуру, то порогове значення C будемо визначати, як середнє значення суми шести останніх відстаней:

$$C = \frac{l_{i-6}+l_{i-5}+l_{i-4}+l_{i-3}+l_{i-2}+l_{i-1}}{6},$$

де i це порядковий номер точки контуру K, що аналізується, l – відстань між точками контуру. Якщо $l_i \leq C$, то точка i належить до множини кутових точок T.

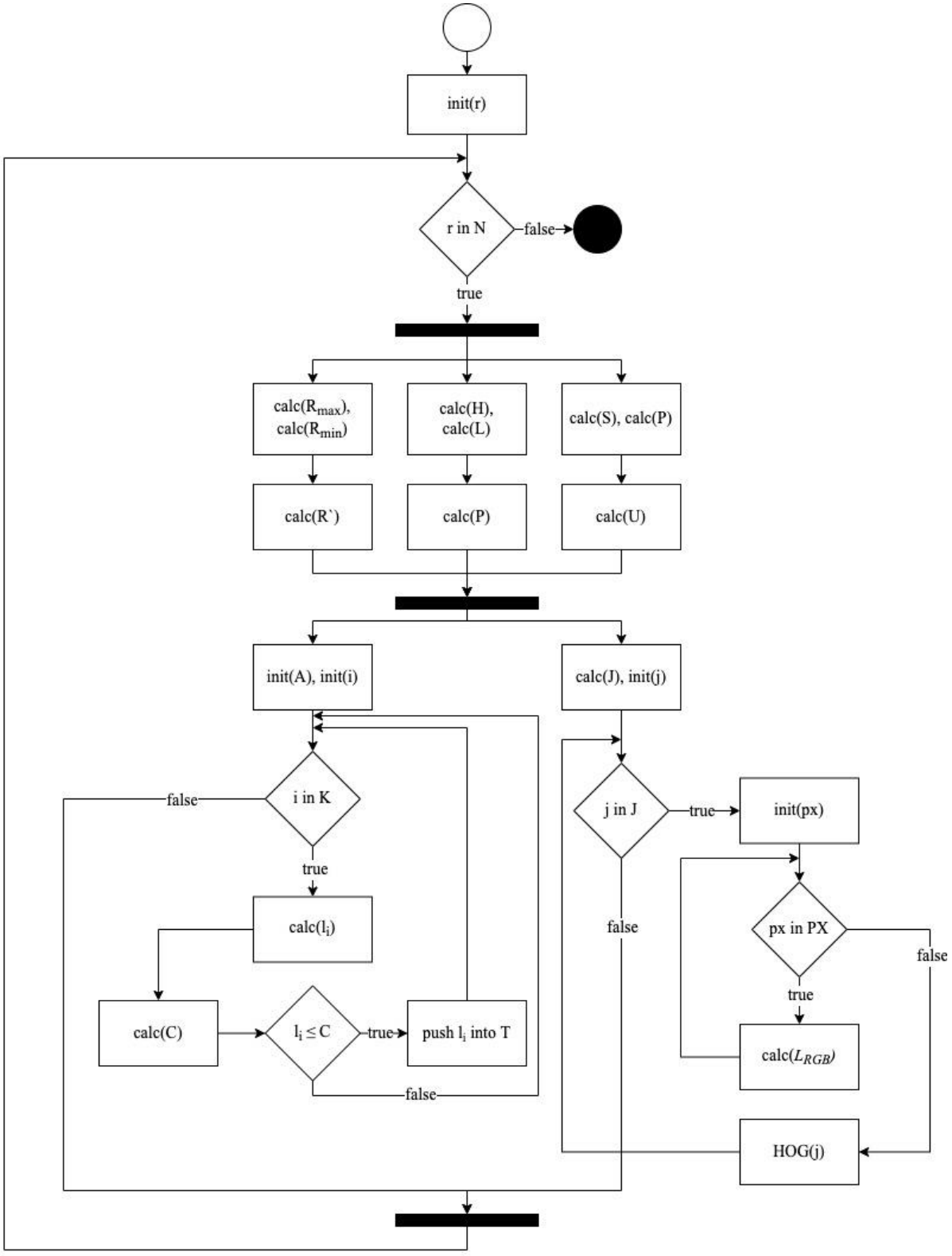

Рис. 3.3. Алгоритм виділення ознак об’єктів на зображенні.

Рис. 3.4. Загальний алгоритм класифікації та кластеризації об'єктів на зображенні.

На даному кроці алгоритму потрібно буде вирахувати гістограму градієнта HOG об’єкта, що аналізується. Алгоритм побудови гістограми градієнта проводиться наступним чином. Для початку потрібно розбити область об’єкту *O* на множину комірок *J* розміром 8×8 пікселів, це зумовлене тим, що такий блок буде мати 128 значень інформації про градієнт, тобто містити значення інтенсивності та напрямку градієнта [64]. Дана множина може бути виражена у вигляді гістограми, що буде мати 9 інтервалів, тому що будемо використовувати беззнаковий градієнт, для зменшення обчислювальної складності. Це дасть методу більшу стійкість до шуму на зображенні.

Отже далі потрібно буде розрахувати гістограму для інтервалів IN = [0°, 20°, 40°, 60°, 80°, 100°, 120°, 140°, 160°]. Градієнти у всіх пікселях *PX* в осередку 8×8 додаються до дев’ятьох інтервалів відповідно до напряму для створення остаточної гістограми градієнта.

Вище описаним методом градієнт стає менш чутливим до шумів на зображенні, але значення гістограми градієнту все ще залежні від середньої яскравості області, що аналізується, а отже нам потрібно зробити ці значення менш чутливими до яскравості. Для цього потрібно провести нормалізацію параметрів *RGB* [65]. Щоб провести нормалізацію приймемо значення *RGB* вектором, тоді є можливість розрахувати довжину L_{RGB}.

$$L_{RGB} = \sqrt{R^2 + G^2 + B^2}.$$

Розрахунок нормалізованих параметрів R`G`B` виконується за формулами.

$$\mathrm{R`} = \frac{R}{L_{RGB}},$$

$$\mathrm{G`} = \frac{G}{L_{RGB}},$$

$$\mathrm{B`} = \frac{B}{L_{RGB}}.$$

Після нормалізації вектору кольорів побудова гістограми градієнту буде більш стійка до змін яскравості об’єкта.

Приклад роботи алгоритму по визначенню об’єкта, що найчастіше піддався створенню анімації на зображені представлено на Рис. 3.5. Зліва оригінальне зображення, а справа знайдений об’єкт.

Рис. 3.5. Приклад роботи алгоритму.

Після того, як було отримано всі характеристики об’єкту алгоритм може перейти до визначення кількості кластерів для знайдених об’єктів, для цього буде використовуватися алгоритм k-середніх для K кластерів. Після побудови кластерів на основі об’єктів з поточного зображення, потрібно порівняти отримані кластери з раніше збереженими в базі даних. Якщо після перевірки поточних кластерів з існуючими в системі буде отримано новий, то його потрібно зберегти в базу даних, а для існуючих оновити нормовані показники.

Наступним кроком нам потрібно передати об’єкти з кластерів, що було оновлено до нейронної мережі, як тренувальну вибірку. Об’єкти з кластерів, що було створено на попередньому кроці алгоритму так само будуть формувати навчальну вибірку для нейронної мережі.

3.3. Розроблення алгоритму анімування статичних об’єктів на зображеннях.

Анімування об’єкта буде проводитися за допомогою афінних перетворень таких як поворот та перенесення (рис. 3.6).

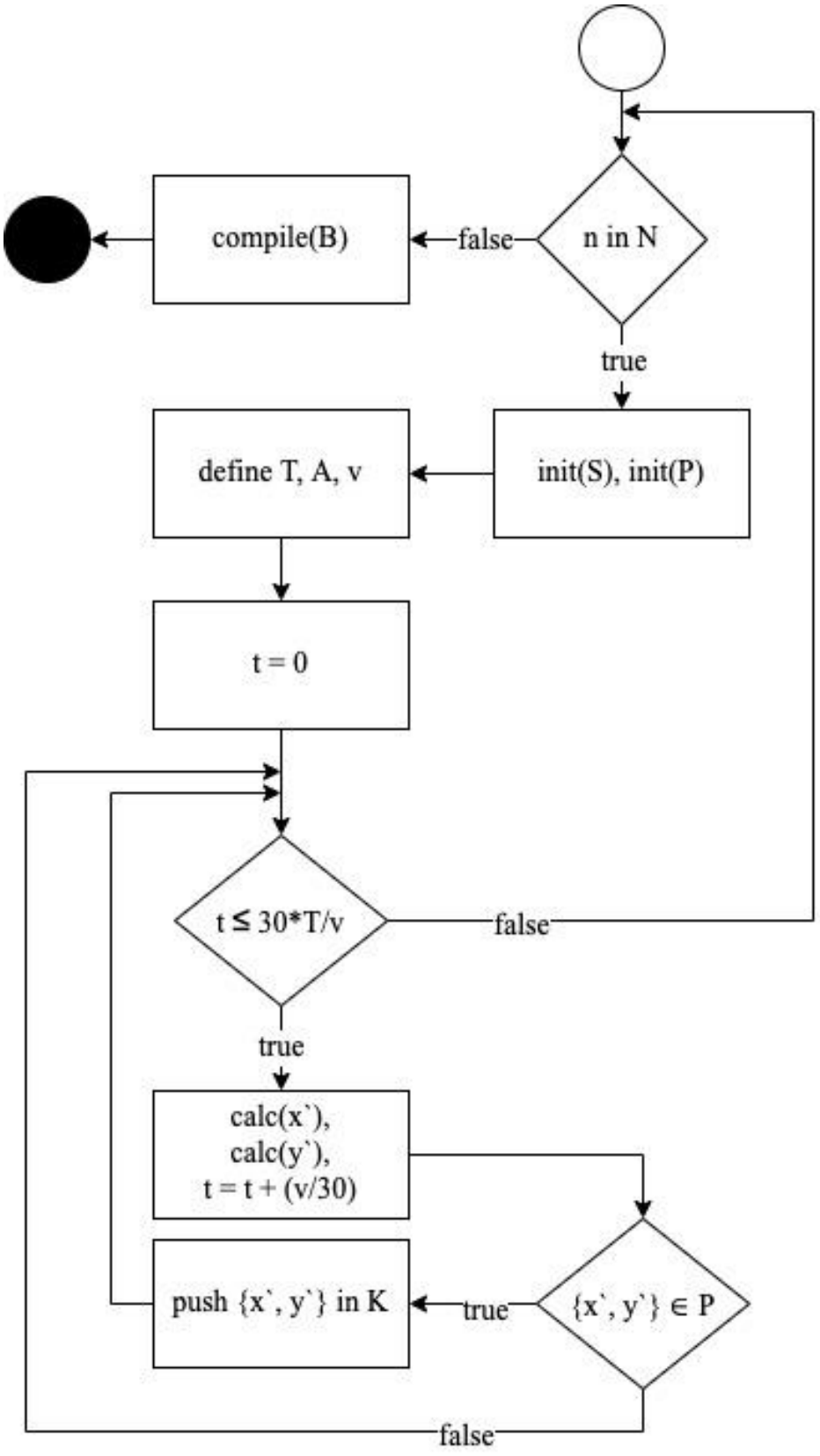

Рис. 3.6. Алгоритм створення ефекту анімації на зображені.

Під об’єктами анімації слід розуміти обрані області *N* на зображені, що будуть обмежені замкнутою кривою, що формує площину *P*.

Отже, перший крок при побудові анімації буде створення копій виділених об’єктів, під кожний об’єкт буде створюватися новий шар *S*, який буде мати розміри оригінального зображення. На кожному шарі буде розміщати потрібний об’єкт в координатах, які відповідають розміщенню об’єкта на вхідному зображенні (Рис. 3.7).

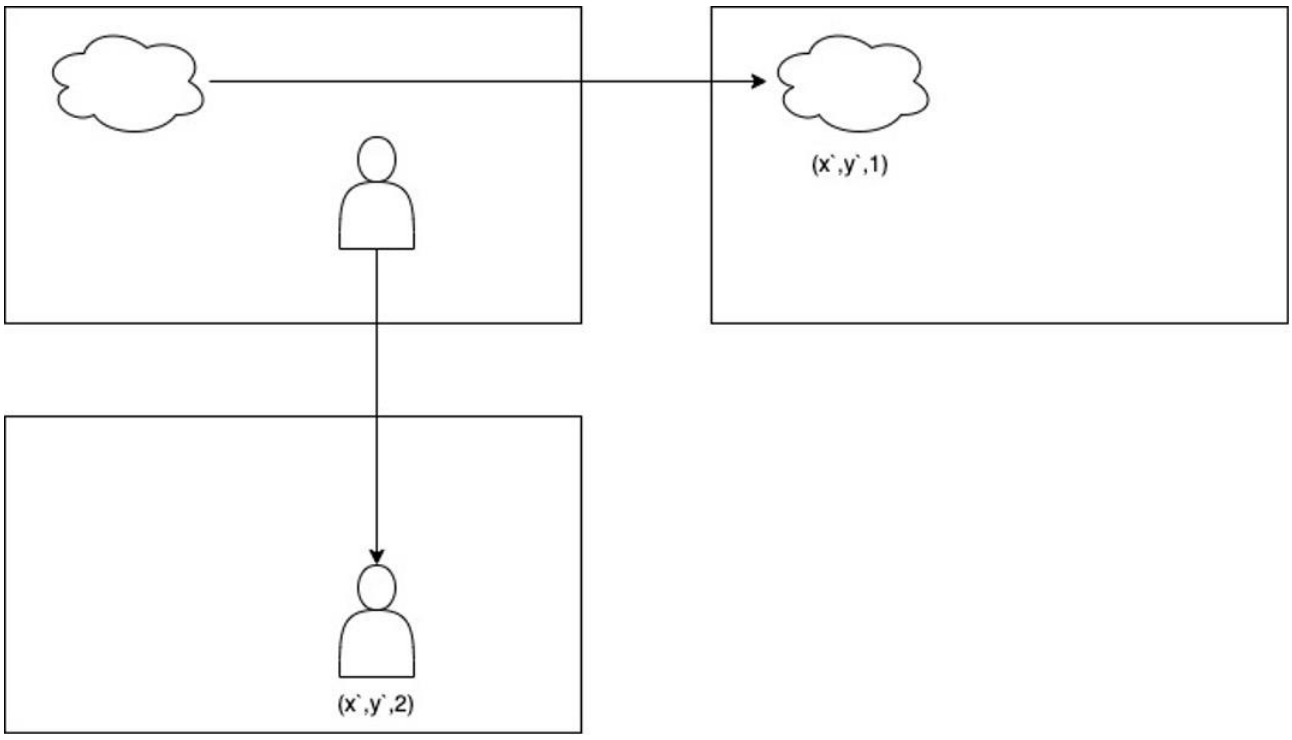

Рис. 3.7. Схема перенесення об’єктів на нові шари.

Другим кроком йде розрахунок переносу та повороту об’єктів на кожному шарі окремо. Розрахунок переносу та повороту для кожного пікселя буде розраховуватися відносно його напрямку, який задається вектором *A(a, b),* та з урахуванням швидкості анімації і часу виконання самої анімації за формулами, що були описані в пункті два.

Кількість позицій кожного пікселя під час анімації залежить від загального часу T та швидкості анімації, при цьому треба врахувати, що для плавного відображення анімації потрібно робити відео з частотою 30 кадрів в секунду. Тому інтервал *I* часу анімації буде мати значення *(0, 30T)*, а крок *i* інтервалу буде

дорівнювати $v/30$ секунди, де v – індекс прискорення, тоді кількість нових позицій пікселя під час анімації буде дорівнювати $30 \cdot T/v$.

Тепер є можливість виконати розрахунки $x`(t)$ та $y`(t)$ для кожного кадру відео підставивши до формул із пункту 2.3 значення з інтервалу (Рис. 3.8). Однак значення $x`$ та $y`$ можуть виходити за границі виділеної області для анімації, тому важливим є перевірка чи значення $x`$ та $y`$ належать площині об'єкту, що піддається анімації. Позиції пікселів, що не проходять перевірку даної умови, не відображаються на поточному кадрі *K* відео.

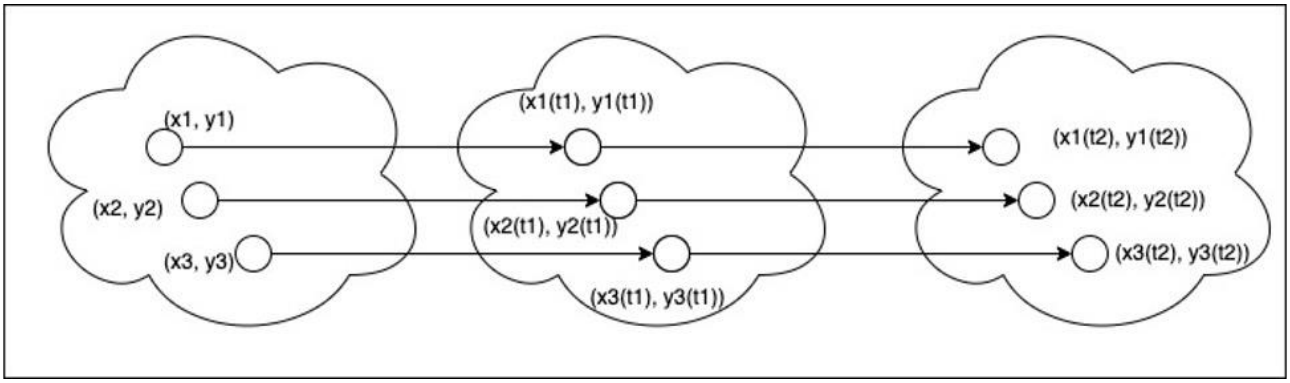

Рис. 3.8. Визначення положення пікселя на зображенні в заданий час.

На останньому кроці алгоритму потрібно сформувати відеоряд *B* з отриманих кадрів [66].

Однак існує проблема під час створення ефекту анімації. Дана проблема полягає в тому, що пікселі в деякий момент часу можуть виходити за межі виділеної області об'єкту, що піддається анімації. Враховуючи, що область це набір точок, то це дає можливість визначити область, як багатокутник, тоді для вирішення даної проблеми, потрібно вирішити задачу належності точки до багатокутнику.

Найбільш розповсюджений підхід – це підрахунок кількості перетинів, який базується на підході кількості перетинів з ребрами багатокутника променем, що має початок в точці, приналежність якої потрібно визначити. Даний алгоритм вирішення задачі належності точки до багатокутника складається з таких кроків:

1. З точки, приналежність якої потрібно визначити, будується промінь в довільному напрямку. В комп’ютерній графіці прийнято обирати позитивний напрям осі абсцис.
2. Рахується кількість перетинів променя з ребрами багатокутника. Для цього запускається цикл, що проходить по всім точкам, що належать ребрам багатокутника.
3. Аналізується кількість перетинів. Якщо число непарне, то точка належить множині точок багатокутника, якщо число парне – не належить [89].

Однак у алгоритму є виродженні випадки (Рис. 3.9) при яких визначення приналежності точки до багатокутника буде трактовано не вірно.

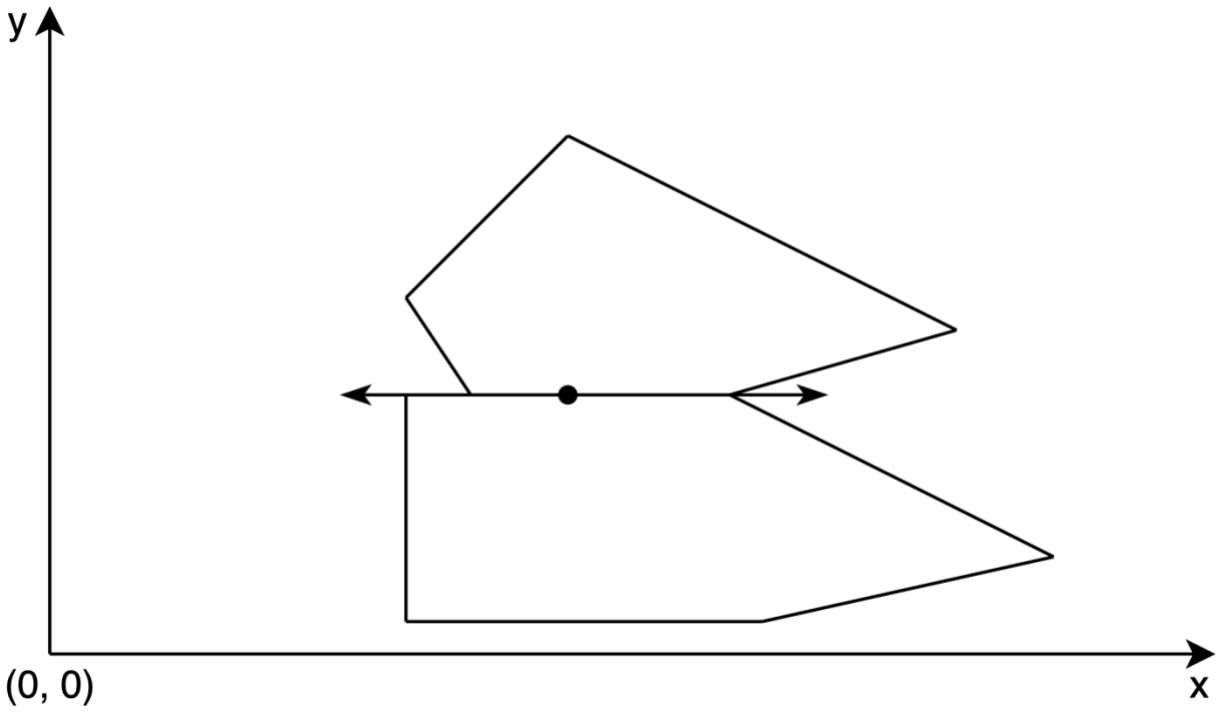

Рис. 3.9. Вироджені випадки перетину променем кута або ребра багатокутника.

Розглянемо даний випадок детальніше, по умовам алгоритму спочатку промінь був побудований в позитивному напрямку осі *X*, та перетнув кут багатокутника. Так в такому випадку є можливість випустити промінь в іншому напрямку та проаналізувати отримані результати, але це збільшує ресурси на обчислення в два рази, що суперечить вимогам, що були поставлені перед

системою. Але навіть з таким підходом все одно існує імовірність, що другий промінь також натрапить на вироджений випадок, наприклад напрям променю буде співпадати з напрямом ребра, що перетинаються. Отже в таких випадках алгоритм може вернути результат, що точка не належить множині точок багатокутника хоча насправді це не відповідає дійсності. При цьому складність даного алгоритму *O(N)* [90]. Враховуючі ці недоліки алгоритму його використання в системі є не доцільним.

Розглянемо метод, що має назву Angle weighted pseudo normal (AWPN). Метод пов’язаний з поняттям полів відстаней зі знаком. Алгоритм виконання пошуку за даним методом складається з таких кроків:

1. Для точки, що розглядається, потрібно знайти найближчу точку, що лежить на ребрі або вершині багатокутника. Для цього з точки, належність якої треба визначити, будуються відрізки до ребер та обчислюється їх відстань.
2. Будується нормаль найближчої точки.
3. Аналізується розташування нормалі, якщо вектор нормалі буде розташовано перпендикулярно до ребра, тоді найближча точка належить ребру багатокутника. Якщо найближча точка є вершиною багатокутника, то нормаль буде являти собою усереднений вектор ребер, що прилягають до вершини.
4. Проводиться обчислення кута між нормаллю найближчої точки та вектору відрізка, що був побудований з точки, що аналізується, до найближчої точки на ребрі. Тоді, якщо кут більше 90°, то точка не належить множині точок багатокутника, в інакшому випадку – точка належить множині точок багатокутника [91].

Обчислювальна складність алгоритму *O(log(N))*, враховуючи, що аналізуватися буде багато точок, а дана обчислювальна складність при великій кількості даних збільшується дуже повільно, даний алгоритм може бути використаний в системі та буде відповідати поставленим вимогам.

Однак для оптимізації алгоритму побудова нормалей ребер та вершин багатокутника, тобто області, анімування буде проводитись до старту алгоритму, адже ці параметри не будуть змінюватися з часом.

Також є можливість обмежити область на якій алгоритм визначення приналежності точки багатокутнику буде запускатися. Ідея полягає в тому, щоб починати аналіз приналежності точки множині точок області анімування в площині розмірністю 20x20 пікселів, що формується точками, які є сусідніми до точок, що обмежують область анімування. Параметри областей також слід будувати до початку виконання алгоритму створення анімації об’єктів. При такому підході кількість точок для яких потрібно буде розраховувати їх приналежність до області анімування буде значно меншою, що пришвидшить виконання алгоритму.

Також існує залежність розрахункової складності алгоритму анімування від кількості пікселів, шо анімуються та часу самої анімації.

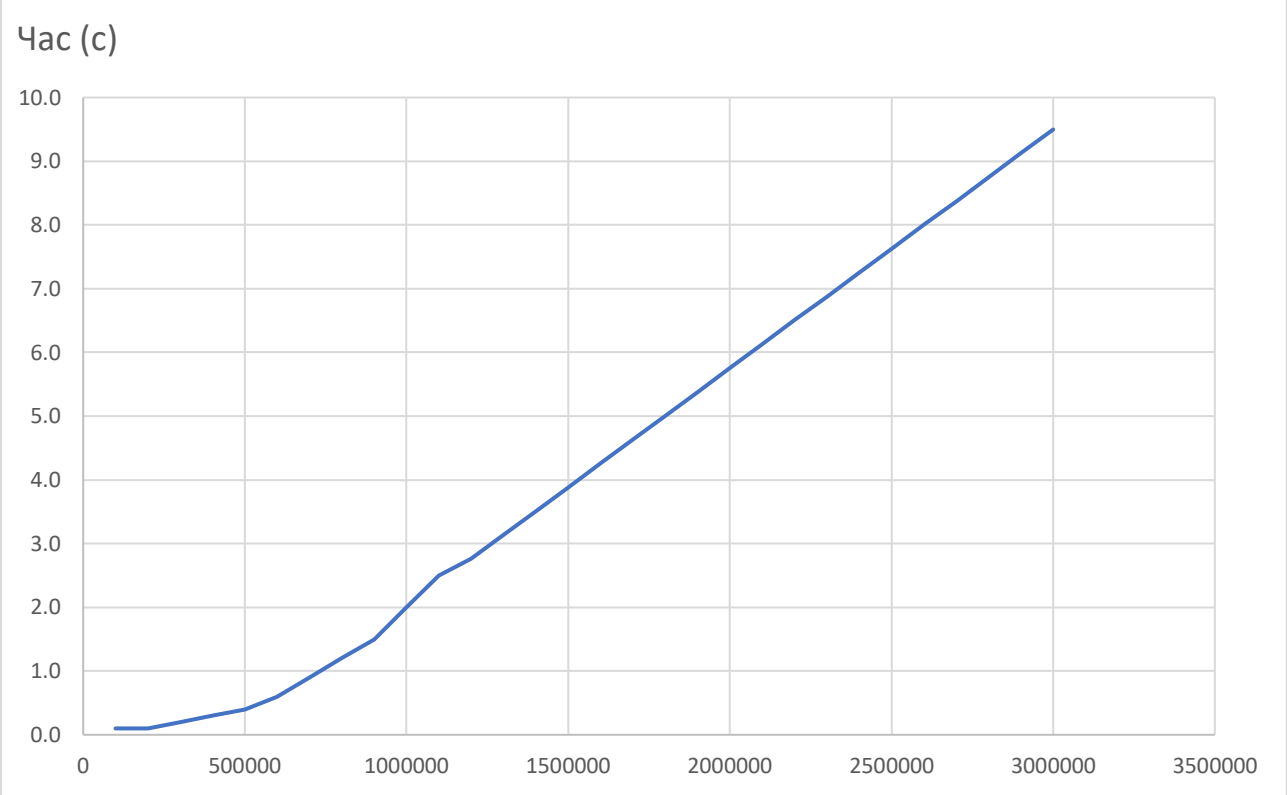

Рис. 3.10. Розрахункова складність алгоритму анімування від кількості пікселів.

Отже було проаналізовано виконання алгоритму для різних розмірностей зображення та часу виконання алгоритму анімації, а час відтворення анімації було обрано 10 секунд. На Рис. 3.10 представлено залежність розрахункової складності алгоритму анімування від кількості пікселів, що піддаються ефекту анімації. Отже, як видно з графіку при досягненні кількості пікселів до 1 млн, то розрахункова складність складає більше 2 секунд і далі росте в лінійній прогресії. Важливим є стиснення зображення до 1млн пікселів, але при цьому нам потрібно зберегти пропорції оригінального зображення. Тому було вирішено, що вхідні зображення потрібно стиснути по найбільшому значенню ширини або висоти зображення. Отже, якщо ширина зображення більше його висоти, тоді стиснення проводиться відносно ширини в іншому випадку відносно висоти. Стиснення проводиться у форматі збереження пропорцій зображення, тобто потрібно порахувати відсоток першого кроку стиснення по найбільшому значенню, а потім на отриманий відсоток стиснути менше значення. Для збереження пропорцій зображення при стиснені потрібно керуватися рівнянням:

$$\frac{w_o}{h_o} = \frac{w_t}{h_t},$$

де w_o, h_o – ширина та висота оригінального зображення, а w_t, h_t – ширина та висота трансформованого зображення відповідно.

При цьому формула розрахунку кількості пікселів зображення:

$$\mathrm{P} = w \cdot h,$$

де w та h - ширина та висота оригінального зображення відповідно.

Коефіцієнт стиснення зображення будемо розраховувати по формулі:

$$k_t = \frac{1000000}{P},$$

де P – кількість пікселів оригінального зображення. Тоді щоб зменшити зображення до 1 млн пікселів, потрібно зменшити ширину та висоту оригінального зображення на отриманий коефіцієнт стиснення, тобто за формулами:

$$w_t = w_o \cdot k_t,$$
$$h_t = h_o \cdot k_t.$$

Де w_o, h_o – ширина та висота оригінального зображення, w_t, h_t – ширина та висота трансформованого зображення, k_t – коефіцієнт стиснення.

Отже тоді кінцева формула прийме вигляд:

$$w_t = \frac{1000000 \cdot w_o}{P},$$

$$h_t = \frac{1000000 \cdot h_o}{P},$$

При цьому рівняння збереження пропорцій буде виконуватись:

$$\frac{\frac{1000000 \cdot w_o}{P},}{\frac{1000000 \cdot h_o}{P}} = \frac{w_o}{h_o}.$$

Такий підхід дасть можливість на виході мати оптимальне за кількістю пікселів зображення, що складається з 1 млн пікселів, але при цьому з урахуванням збереження пропорцій ширини та висоти оригінального зображення, об’єкти та фон зображення не будуть спотворенні.

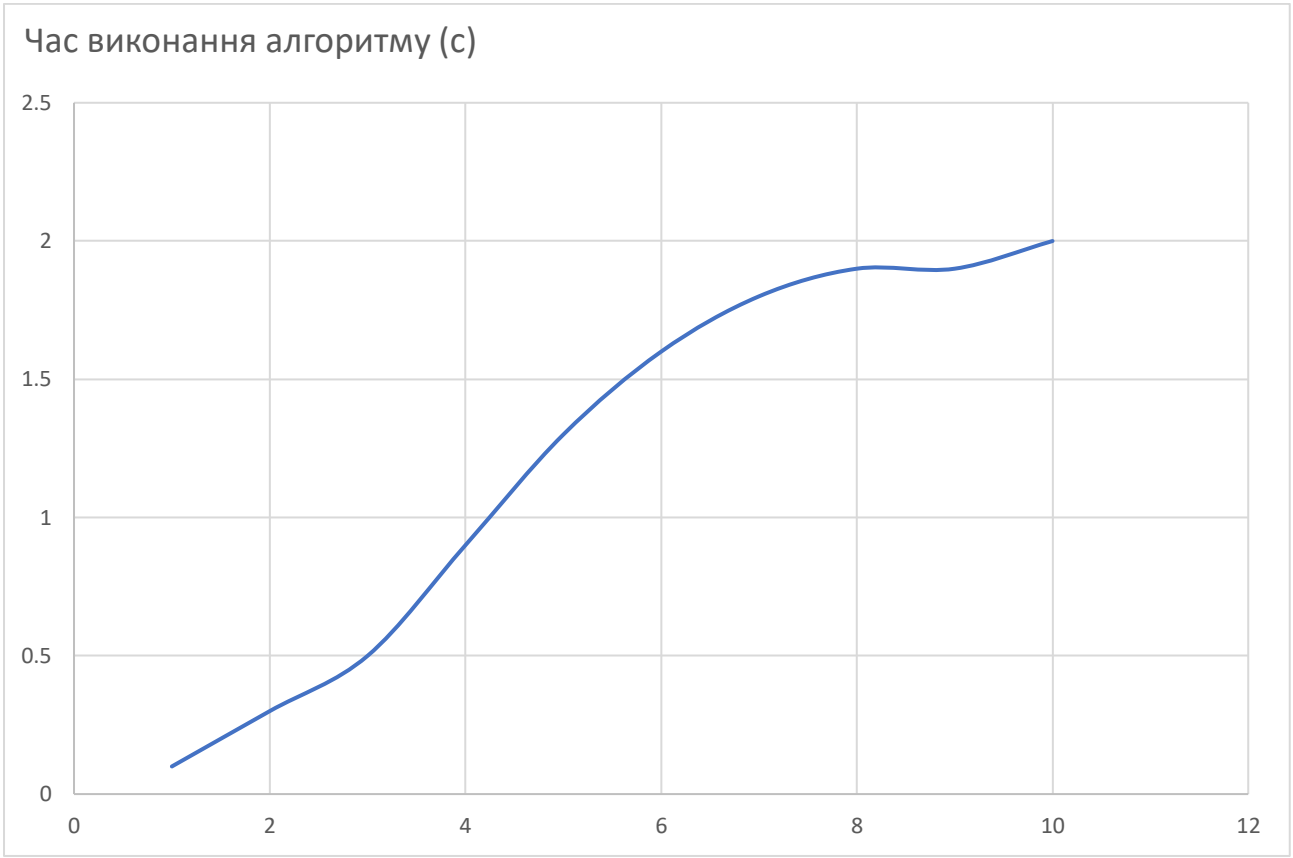

Рис. 3.11. Залежність часу виконання алгоритму від кількості пікселів.

Проаналізуємо також залежність часу виконання алгоритму анімації від часу відтворення анімації. Для аналізу було взято зображення, яке мало 1 млн пікселів. Як видно із графіку (Рис. 3.11) то при часі відтворення анімації більше 3 секунд час виконання алгоритму росте по експоненті, отже найкращим варіантом в умовах обмежених обчислювальних ресурсів буде обрати сталий час анімації, тобто 3 секунди, а для створення ефекту неперервності анімації відео буде повторюватися.

Висновки до третього розділу

В даному розділі було розроблено алгоритми пошук, класифікації та кластеризації, анімування об’єктів.

Для виконання алгоритму пошуку об’єктів, алгоритму класифікації та кластеризації пропонується аналізувати загальну середню яскравість зображення після чого на основі отриманого значення використовується оператор Собеля або оператор Прюітта для виділення границь об’єктів. Для алгоритму пошуку об’єктів на зображенні описано методи попередньої обробки зображення таких, як бінарізація та сегментація зображення. Сегментація виконується за допомогою методу заповнення замкнутих областей з використанням алгоритму Брезенхема.

Сформовано кроки алгоритму кластеризації та класифікації об’єктів на зображені, а саме описанні критерії побудови ознак об’єктів: порядок розрахунку нормованих показників периметру та площі об’єкт, вписаного та описаного кіл, умови розрахунку кількості та взаємного розташування кутів, а також визначено інтервали на яких буде проводитися побудова гістограми градієнту об’єктів. Описано кроки алгоритму для побудови кластерів за допомогою методу k-середніх, формування класів та збереження їх для подальшого використання, як тренувальної вибірки для штучної нейронної мережі, наведено приклади роботи даних методів.

Для анімування статичних об’єктів визначено основні кроки алгоритму: створення копій об’єктів анімації та їх перенос на паралельні площини; розрахунок повороту та переносу об’єктів, що піддаються ефекту анімації;

формування відеоряду з отриманих позицій пікселів об’єкту в кожну ітерацію анімації з урахуванням параметрів анімації.

Проведено оптимізацію виконання алгоритмів, що наведено на графіках, які відображають ємнісну складність розроблених методів та обґрунтовано вибір параметрів для виконання алгоритмів.

Усі розроблені алгоритми зображено за допомогою блок-схем.

РОЗДІЛ 4. РОЗРОБЛЕННЯ АРХІТЕКТУРИ ТА АПРОБАЦІЯ РЕЗУЛЬТАТІВ.

У розділі представлено розробку архітектури системи, обґрунтовано вибір технологій для реалізації системи. Також визначено функціонал системи та засоби, які забезпечать стабільну роботу системи. Представлено готовий додаток для створення анімації статичних об’єктів на зображені.

Результати розділу опубліковано в працях автора [4, 6, 9].

4.1. Архітектури системи.

4.1.1. Визначення підсистем та їх процесів.

Дана система орієнтована для використання на мобільних пристроях і це накладає обмеження на використання обчислювальних ресурсів. Однак деякі операції такі, як кластеризація, класифікація та робота штучної нейронної мережі використовують багато ресурсів отже їх не має можливості виконувати на мобільному пристрої. Для цього було прийняте рішення розділити систему на три основні підсистеми: мобільний додаток, API модуль та Cloud модуль. Розподілення завдань між модулями викладено в діаграмі діяльності системи (Рис. 4.1).

Отже мобільний додаток це модуль взаємодії з користувачами, де користувач зможе завантажити зображення, обирати об’єкти для та параметри анімування. Також в додатку будуть виконуватися процеси для побудови анімації і створення відео. API модуль буде служити для процесів обміну даними між мобільним додатком та Cloud модулем, також буде виконувати операції по отриманню і збереженню даних. Cloud модуль буде використовувати хмарні ресурси для виконання ресурсномістких процесів: виконання кластеризації, класифікації, пошук об’єктів на зображені та тренування штучної нейронної мережі.

Розглянемо більше детально процеси, які будуть виконуватися в системі. Перший процес мобільного додатку це завантаження зображення.

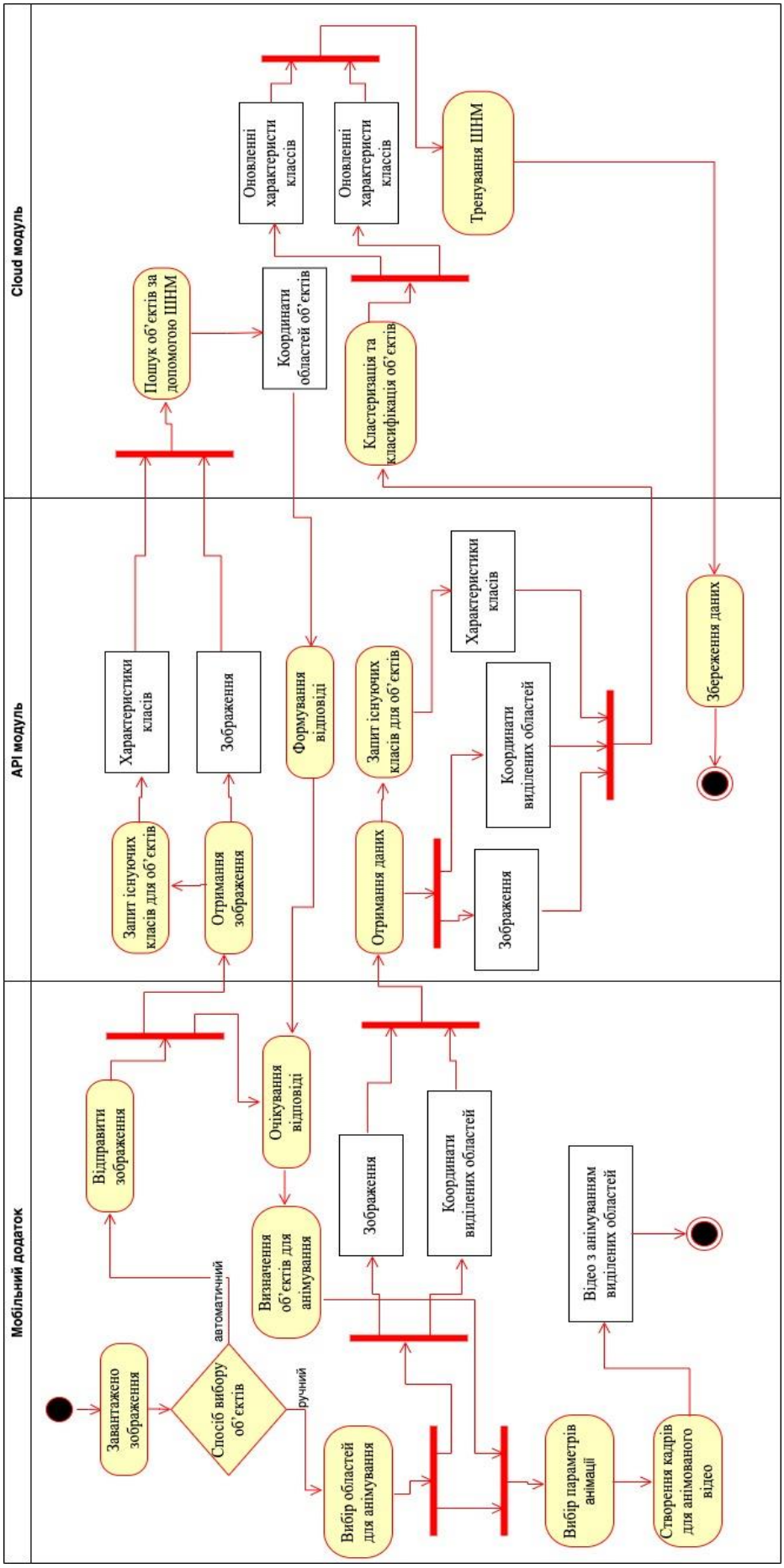

Рис. 4.1. Діаграма діяльності системи.

Зображення може бути отримано, як і з галереї пристрою, так і з камери. Далі існує вибір можливості вибору областей для анімації: автоматичний або ручний. У випадку ручного вибору об’єктів, користувачу потрібно буде виділити області самому. Після виділення областей, система відправить ці данні на API модуль. При виборі автоматичного режиму вибору об’єктів мобільний додаток відправить зображення на API модуль і буде очікувати відповіді від сервера. Після отримання відповіді від сервера з виділеними областями або вибору об’єктів вручну, в мобільному додатку буде можливість вибору параметрів анімації: напрямку, швидкості та тривалості. Далі в додатку виконується процес побудови кадрів відео та створення самого відео, яке відображається користувачу.

Перейдемо до процесів, які виконуються модулем API. При отриманні даних від мобільного додатку, модуль робить запит в сховище даних на отримання характеристик вже існуючих в системі класів об’єктів. Далі отриманні дані зі сховища та мобільного додатку передаються Cloud модулю.

Cloud модуль в залежності від типу запиту виконує процеси пошуку об’єктів на зображені, або процеси класифікації та тренування штучної нейронної мережі. Після виконання пошуку об’єктів Cloud модуль передає данні виділених областей назад в модуль API, який відправляє ці дані в мобільний додаток. Після виконання процесу класифікації Cloud модуль виконує процес тренування нейронної мережі на оновлених та щойно створених даних класів об’єктів. Після виконання цього процесу дані відправляють до модулю API, щоб виконати їх збереження в сховищі даних.

На діаграмі потоків даних (Рис. 4.2) представлено, як буде проводитися обмін пакетами даних між функціями системи. Отже від користувачем систему будуть надаватися такі данні: зображення, координати об’єктів для створення анімації, параметри анімації. Функція збереження зображення буде виконуватися на стороні мобільного додатку та буде зберігати зображення в галерею пристрою та відправляти це зображення на АПІ модуль для подальшої його обробки.

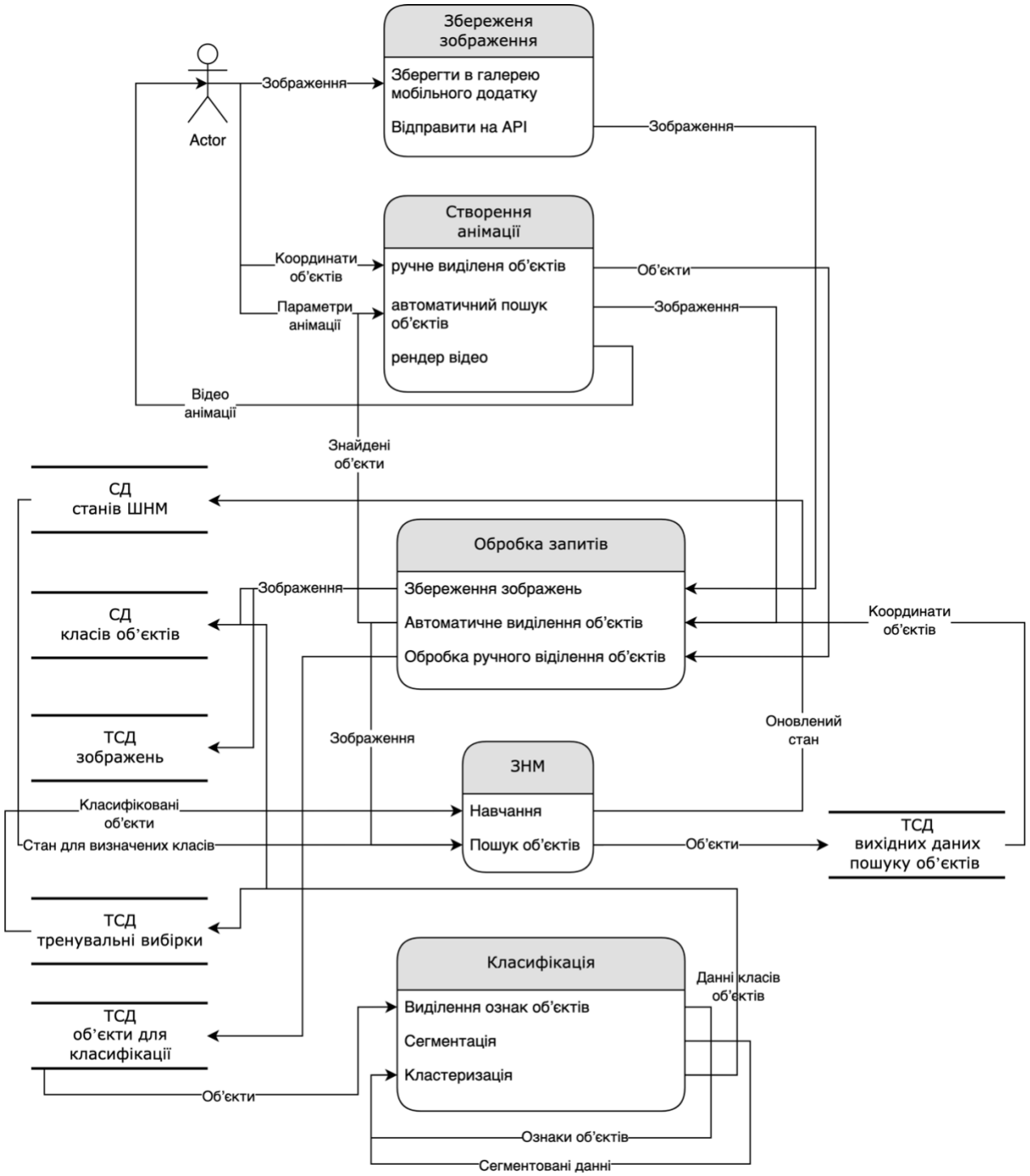

Рис. 4.2. Діаграма потоків даних між функціями системи.

Функція створення анімації приймає дані про координати виділених користувачем об’єктів та введені ним параметри анімації, у випадку активації автоматичного пошуку ОПІ, координати від користувача функцією не приймаються. Функція відправляє виділені користувачем об’єкти на АПІ модуль, який зберігає їх у тимчасове сховище даних, яке використовується функцією класифікації. Функція класифікації виконує процес сегментації,

виділення ознак та побудову кластерів, після чого дані відправляються в ТСД для формування тренувальних вибірок та в сховище даних класифікованих об’єктів. Після отримання сигналу про формування тренувальної вибірки, запускається функція навчання згорткової нейронної мережі, коли тренування було завершено, функція відправляє данні стану нейроної мережі до сховища даних. Якщо було активовано функцію автоматичного пошуку об’єктів, то АПІ модуль активує функцію згорткової нейронної мережі для пошуку об’єктів. Збережений стан мережі для пошуку об’єктів надходить зі сховища даних стану штучної нейронної мережі. Після виконання пошуку, знайдені координати об’єктів надсилаються до тимчасового сховища, АПІ модуль отримує сигнал, що з’явилися дані та відправляє їх до функції створення анімації. Функція створення анімації після отримання координат об’єктів та параметрів анімації запускає процес рендеру відео та надсилає відеофайл користувачу.

Було визначено основні стани системи та пакети даних, які будуть передаватися між функціями системи, це дає змогу перейти до вибору компонентів, що будуть використовуватися для забезпечення стабільної роботи всіх функцій системи та належному контролю за даними, які будуть зберігатися тимчасово для певних процесів, так і даних, які будуть потрібні на протязі всього існування системи.

4.1.2. Опис компонентів системи.

Розглянемо компоненти, які будуть забезпечувати стабільну роботу системи (Рис. 4.3).

Для роботи модулю API буде використовуватися сервери на базі операційної системи Linux. Даний вибір зумовлений тим, що ця операційна система підлягає кращому налаштуванню під потреби конкретної системи, що забезпечить вищу продуктивність та можливість обробляти більше HTTP запитів.

Також системи Linux мають кращу систему захисту, що робить її більш стійкою для можливого злому та запуску шкідливого програмного забезпечення.

Важливою перевагою є безкоштовне використання даної операційної системи, що зменшує витрати на підтримання інфраструктури системи.

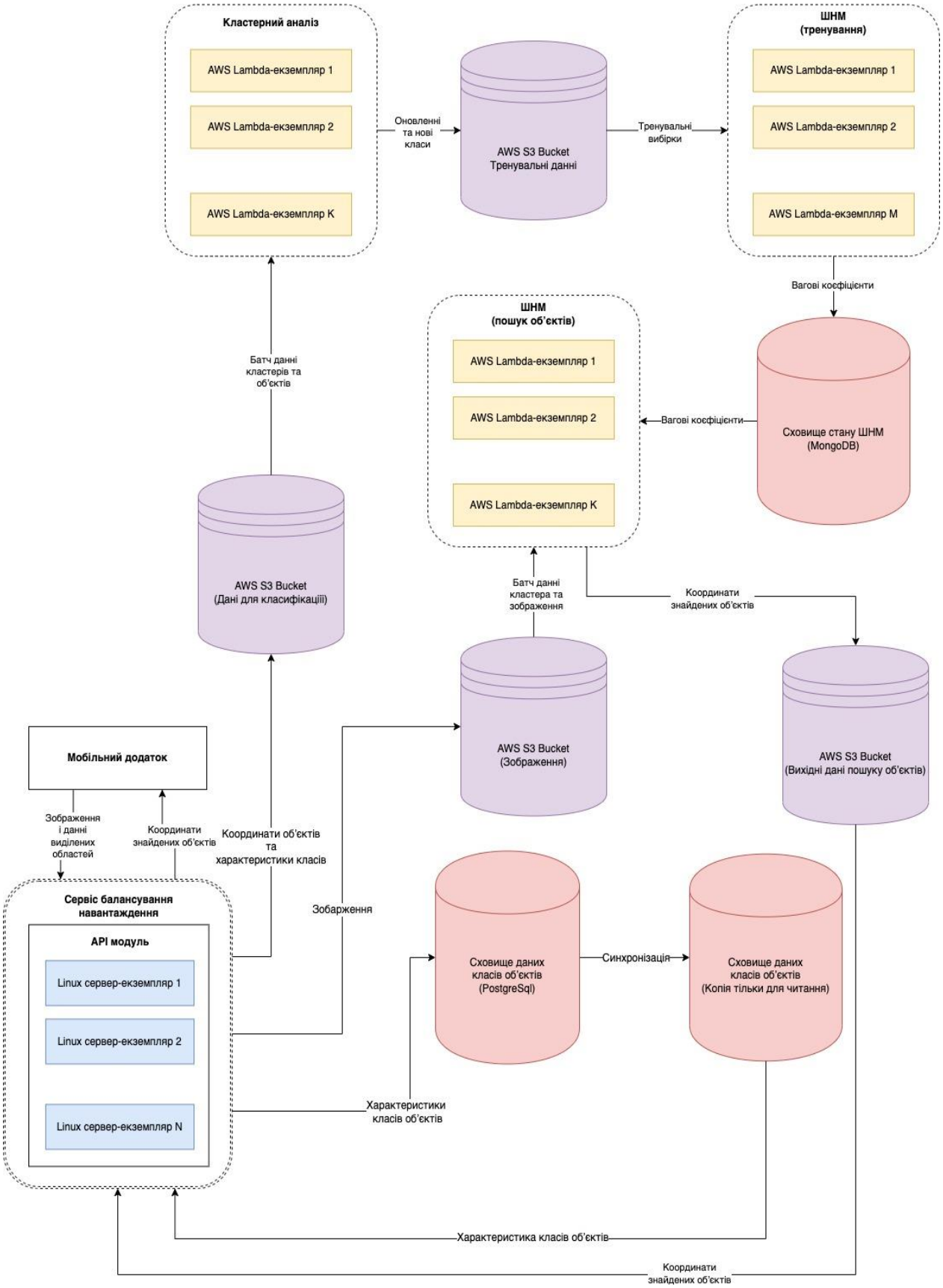

Рис. 4.3. Діаграма компонентів.

Рис. 4.4. Схема роботи сервісу балансування навантаження.

Також для запобігання перевантаження екземплярів сервера модуля АПІ буде використано сервіс балансування навантаження. Балансувальники навантаження є ключовим елементом для створення чудових сучасних інтернет-додатків, оскільки вони забезпечать вашому додатку такі переваги:

- Резервування. Екземпляр серверу може вийти з ладу, але доки залишається хоча б один серверний екземпляр, балансувальник навантаження може спрямовувати трафік клієнта на екзмпляр, що залишилися в робочому стані.

- Масштабованість. Чим більше екземплярів серверів, тим більшу кількість запитів є можливість обробити.

Правило розподілення трафіку буде базуватися на визначенні менш завантаженого екземпляру сервера та перерозподіл запитів від мобільного додатку на цей екземпляр (Рис. 4.4). Це забезпечить відмовостійкість модулю, через створення нових екземплярів сервера при піковому навантажені [67].

Cloud модуль буде використовувати сервіси Amazon для виконання ресурсномістких операцій, а саме AWS Lambda. Даний сервіс буде використовуватися для опрацювання процесів кластеризація та класифікації об’єктів, пошуку об’єктів на зображеннях і тренування штучної нейронної мережі. AWS Lambda це сервіс без серверних обчислень, який запускає програмний код у відповідь на визначену подію, у варіанті використання Cloud модулем, це буде додавання даних в AWS S3 Bucket. Переваги AWS Lambda:

- Повністю автоматизоване адміністрування. Завдяки ретельному управлінню інфраструктурою AWS Lambda виконується у високопродуктивному, відмовостійкому середовищі.
- Вбудована відмовостійкість. AWS Lambda підтримує необхідні обсяги обчислювальних ресурсів у кількох зонах доступності, захищаючи код від несправностей окремих одиниць обладнання або збоїв у роботі центрів обробки даних. AWS Lambda та функції, що працюють у рамках цього сервісу, забезпечують передбачувану та надійну операційну продуктивність.
- Автоматичне масштабування. AWS Lambda викликає код тільки тоді, коли це необхідно, і автоматично масштабує ресурси відповідно до обсягу запитів, що надходять, без будь-якого ручного налаштування [68].

Отже використання сервісу AWS Lambda надає можливість виконувати процеси тільки в той час, коли це потрібно, без необхідності підтримувати сервери. Також важливим є те, що кожна нова подія викликає окремий екземпляр програмного коду тим самим забезпечує безпечну паралельність обчислень та прибирає можливість падіння всіх процесів через помилку в одному із процесів.

Також динамічне виділення ресурсів забезпечить стабільне виконання програмного коду без потреби слідкувати за завантаженістю серверів [69].

Правилами масштабування AWS Lambda для підсистем будуть такі:

- Підсистема кластерного аналізу. При збережені в бакет модулем API даних про об’єкти, що були виділені користувачем вручну, запустити програмний код, який буде виконувати сегментацію, кластеризацію та класифікацію об’єктів. Кількість екземплярів напряму залежить від кількості необроблених наборів даних.
- Підсистема пошуку об’єктів на зображені. При збережені в бакет модулем API зображення. Кількість екземплярів дорівнює кількості не опрацьованих зображень.
- Підсистема тренування ШНМ. При збережені підсистемою кластерного аналізу характеристик об’єктів опрацьованого класу. Кількість екземплярів дорівнює кількості отриманих класів для тренування.

4.1.3. Опис сховищ даних.

Для зберігання тимчасових даних, що забезпечують обмін між підсистемами Cloud модуля буде використовуватися AWS S3 Bucket. Перевагами даного сервісу є автоматична масштабованість пам’яті для забезпечення зберігання даних, можливість налаштування автоматичного видалення даних через певний інтервал даних [70]. Використання цього сервісу забезпечить стабільну передачу великої кількості даних, надійне зберігання та автоматичне видалення після того, як ці данні стануть не потрібні. Також це зменшить навантаження на сервера баз даних, які вже будуть зберігати “постійні” данні. AWS S3 Bucket буде використовуватися Cloud модулем для зберігання даних, що використовуються процесами класифікації, опрацювання даних нейронною мережею та обміну даними з API модулем.

Для зберігання характеристик класів об’єктів обрано СУБД PostgreSQL. Причинами для вибору цієї СУБД стала можливість зберігати в даній СУБД геометричні данні та багатовимірні масиви, а також більші ліміти на розміри таблиць, строк та полів.

Рис. 4.5. Структура бази даних для зберігання класів об’єктів та їх характеристик.

Для зберігання характеристик кластерів буде використовуватися багатовимірні масиви, а для зберігання характеристик об’єктів, що відносяться до кластеру будемо використовувати тип геометричних даних [71].

Розглянемо структуру бази даних для зберігання класів об’єктів та їх характеристик (Рис. 4.5).

Таблиця «classes» буде використовуватися для зберігання отриманих класів об’єктів, що були отриманні після виконання алгоритму кластеризації та класифікації. Має таки поля:

- id – тип даних integer. Первинний ключ таблиці, поле генерується за допомогою Postgres sequences.
- unique_hash_name – тип даних varchar(32), обмеження – унікальний індекс, не може мати значення null. Поле бути зберігати випадково згенерований md5 хеш. Новий хеш буде додаватися в базу даних, якщо при виконанні алгоритму класифікації буде знайдено об’єкт, який не буде віднесений до жодного із існуючих класів в системі.
- created_at – тип даних timestamp, обмеження – не може мати значення null. Поле зберігає дату створення класу в системі.
- las_element_added_at – тип даних timestamp, обмеження – не може мати значення null. Поле зберігаю дату, коли був доданий останній об’єкт в зв’язну таблицю objects.

Таблиця «objects» буде містити в собі данні всіх об’єктів, які будуть додаватися в систему. Об’єкти будуть додаватися, як і після виконання алгоритму пошуку об’єктів на зображеннях так і у випадку, якщо користувач виділяв об’єкти в ручному режимі. Має такі поля:

- id – тип даних integer. Первинний ключ таблиці, поле генерується за допомогою Postgres sequences.
- unique_hash_name – тип даних varchar(32), обмеження – унікальний індекс, не може мати значення null. Поле бути зберігати згенерований хеш об’єкта. Хеш буде генеруватися на основі даних з поля object_representation за допомогою md5 алгоритму, використання цього поля обмежить попадання

в базу повністю однакових об’єктів у випадку, якщо користувач повторно виділяв на одному зображенні ті самі об’єкти. Дане обмеження дасть змогу зменшити об’єм таблиці за рахунок тільки унікальних даних. Використання 32 символьного хешу для кращої продуктивності таблиці при зберіганні унікальних ключів.

- geometric_representation – тип даних geometry, обмеження – не може мати значення null. Поле використовується для зберігання гемотричного відображення області об’єкта, що було отримано в процесі сегментації.
- object_representation – тип даних blob, обмеження – не може мати значення null. Поле бути зберігати оригінальний контент виділеного об’єкта на зображенні.
- segmentation_representation – тип даних blob, обмеження – не може мати значення null. Поле буде містити контент об’єкту, який був отриманий після виконання процесу сегментації.
- created_at – тип даних timestamp, обмеження – не може мати значення null. Поле зберігає дату, коли об’єкт був доданий в систему.
- class_id – тип даних integer, обмеження – не може мати значення null. Зовнішній ключ, що зв’язаний з таблицею «classes».

Таблиця «images» буде використовуватися для зберігання всіх зображень, що були завантаженні користувачами в систему.

- id – тип даних integer. Первинний ключ таблиці, поле генерується за допомогою Postgres sequences.
- unique_hash_name – тип даних varchar(32), обмеження – унікальний індекс, не може мати значення null. Поле бути зберігати згенерований md5 хеш зображення на основі поля image_representation. Як і у випадку таблиці «objects» обмеження цього поля дозволить не зберігати повторно завантажене одне і те саме зображення.
- image_representation – тип даних blob, обмеження – не може мати значення null. Зберігає контент самого зображення, як масив двійкових даних.

- created_at – тип даних timestamp, обмеження – не може мати значення null. Поле зберігає дату, коли зображення було завантажене в систему.

Таблиця «object_features» буде містити дані, що були отримані після виконання алгоритму виділення ознак об’єкту. Містить такі поля:

- id – тип даних integer. Первинний ключ таблиці, поле генерується за допомогою Postgres sequences.
- hog – тип даних json, обмеження – не може мати значення null. Зберігає асоціативний масив даних про гістограму градієнту об’єкта.
- corners – тип даних json, обмеження – не може мати значення null. Зберігає масив знайдених кутів об’єкта, що отримані при виконанні алгоритму виділення
- normalized_u – тип даних double, обмеження – не може мати значення null. Поле створено для зберігання значень нормованого показника *U*, що отримується по формулі з після визначення площі та периметру об’єкта.
- normalized_r – тип даних double, обмеження – не може мати значення null. Поле буде зберігати нормований параметр *R`*, який описує відношення описаного та вписаного кіл.
- normalized_p – тип даних double, обмеження – не може мати значення null. Поле зберігатиме нормовану ознаку *P* об’єкта, що отримується, як відношення висоти і довжини описаного прямокутника.
- object_id – тип даних integer, обмеження – унікальний ключ, не може мати значення null. Зовнішній ключ, що зв’язаний з таблицею «objects».
- created_at – тип даних timestamp, обмеження – не може мати значення null. Поле зберігає дату, коли дані були додані в таблицю.

Зв’язки між таблицями:

- Таблиця «objects» має зв’язок з таблицею «classes» багато до одного, при цьому існує обмеження на те, що сутність «object» повинна мати визначену сутність «class», тому зв’язок жорсткий.

- Таблиця «object_features» зв’язана з таблицею «objects» типом зв’язку один до одного. Для сутності «object_features» існує обмеження, що вона обов’язково має мати визначену сутність «object», тому зв’язок жорсткий.
- Таблиця «images» та таблиця «objects» мають зв’язок типу багато до багатьох. Такий тип зв’язку між цими сутностями зумовлений тим, що існує імовірність, що система визначить ідентичні об’єкти на зображеннях, але система не визначить ці зображення, як однакові.

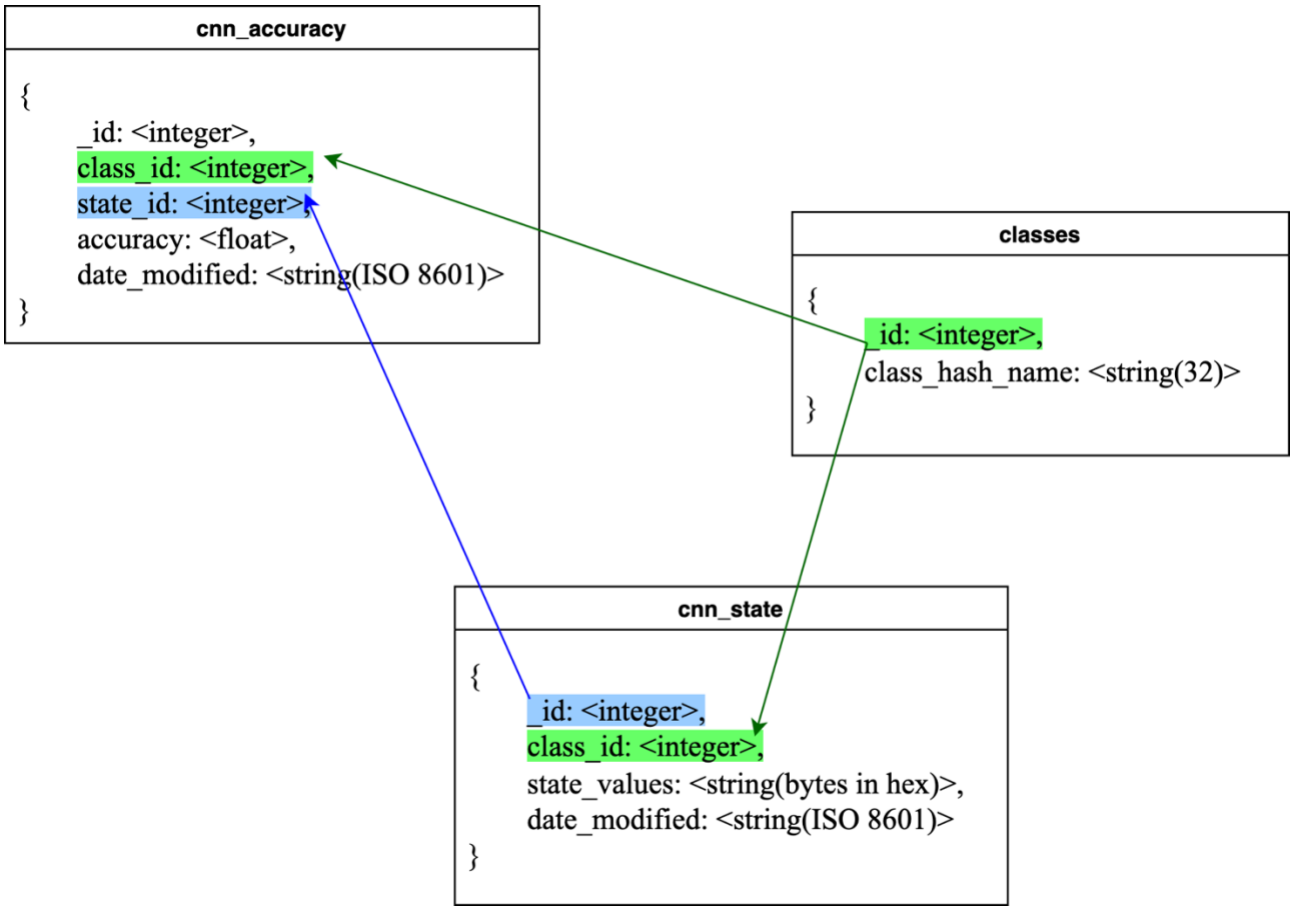

Рис. 4.6. Структура бази даних для зберігання стану згорткової штучної нейронної мережі.

Для зберігання стану тренування штучної нейронної мережі обрано базу даних типу NoSql – MongoDB. Цей вибір зумовлений тим, що ці дані є слабоструктурованими та в них важко виділити відношення. Також важливим є те, що робота тренування штучної нейронної мережі пов’язане з великою кількістю даних, а MongoDB забезпечує набагато швидшу обробку запитів на

запис та отримання даних [72], і також використання окремої БД для зберігання стану штучної нейронної мережі забезпечить зменшення навантаження на СУБД PostgreSQL з якою взаємодіє модуль API, що зменшить імовірність блокування таблиць зі сторони підсистеми нейронної мережі, при якому модуль API би не зміг швидко опрацювати запит і повернути відповідь до мобільного додатку.

Розглянемо структуру сховища даних для зберігання стану згорткової штучної нейронної мережі (Рис. 4.6). MongoDB оперує даними у форматі JSON, тому структура даних описана в цьому форматі. Сам тип таких баз даних не є реляційним, але у випадках коли потрібно посилатися на документ в іншій колекції, то для цього в документах колекції, що пов’язана зберігають ідентифікатор батьківського документа.

Сховище даних стану згорткової нейронної мережі має три колекції: «classes», «cnn_accuracy» та «cnn_state».

Документи колекції «classes» будуть мати таку структуру:

- _id – тип integer, буде відповідати значенню ідентифікатора класу зі сховища даних PostgreSQL.
- class_hash_name – тип string, 32 символьний хеш класу.

Структура документів колекції «cnn_ state»:

- _id – тип integer, унікальний ідентифікатор документу. Генерується за допомогою unix timestamp microseconds.
- class_id – тип integer, ідентифікатор документу класу з яким пов’язаний документ cnn_state.
- state_values – тип даних string. Поле для збереження стану згорткової нейронної мережі, данні будуть зберігатися, як байти закодовані в шістнадцятковому коді.
- date_modified – тип даних string. Поле зберігає дату, як строку в форматі ISO 8601, створення документу.

Структура документів колекції «cnn_accuracy»:

- _id – тип integer, унікальний ідентифікатор документу. Генерується за допомогою unix timestamp microseconds.

- class_id – тип integer, ідентифікатор екзумляру документу cnn_accuracy.
- state_id – тип integer, ідентифікатор екзумляру документу cnn_state.
- accuracy – тип даних float, значення точності розпізнавання об’єктів згортковою нейронною мережею. Вираховується для збереженого стану в екземплярі документу cnn_state, що пов’язаний з даним екземпляром.
- date_modified – тип даних string. Дата створення документу в форматі ISO 8601.

Враховуючі, що документи в колекціях «cnn_accuracy» та «cnn_state» будуть з’являтися кожен раз, коли в клас буде додано нові об’єкти, то для зменшення об’єма даних, що зберігаються в базі, потрібно видаляти документи, що вже є не актуальними. Тому будуть використовуватися такі критерії не актуальності документів. Для колекції «cnn_accuracy» зберігати в базі даних 5 документів, що мають найвищий показник точності розпізнавання, тобто найвище значення поля accuracy. При видалені не актуальних документів також рахувати пов’язані документи з колекції «cnn_state», як не актуальні та видаляти їх також. Для колекції «classes» не буде критеріїв не актуальності документів, тому що всі класи об’єктів, що були збережені в систему в той чи інший період часу будуть використанні. Таким чином буде забезпечуватися мінімально можливий об’єм сховища даних, але при цьому без суттєвих проблем для ефективності роботи системи. При цьому завжди буде можливість повернутися до більш стабільної версії стану згорткової нейронної мережі у випадку, якщо при додаванні нового об’єкту в класи і використання цієї вибірки, як тренувальної, буде отримано так званий ефект перенавчання нейронної мережі.

4.2. Розробка інтерфейсу користувача

Для розробки інтерфейсу користувача було обрано підхід прототипування. Прототипування – це процес створення робочого інтерфейсу на базі намальованих макетів або вайрфреймів, який дозволяє одразу зобразити, як інтерфейс додатка так і варіанти його використання [80].

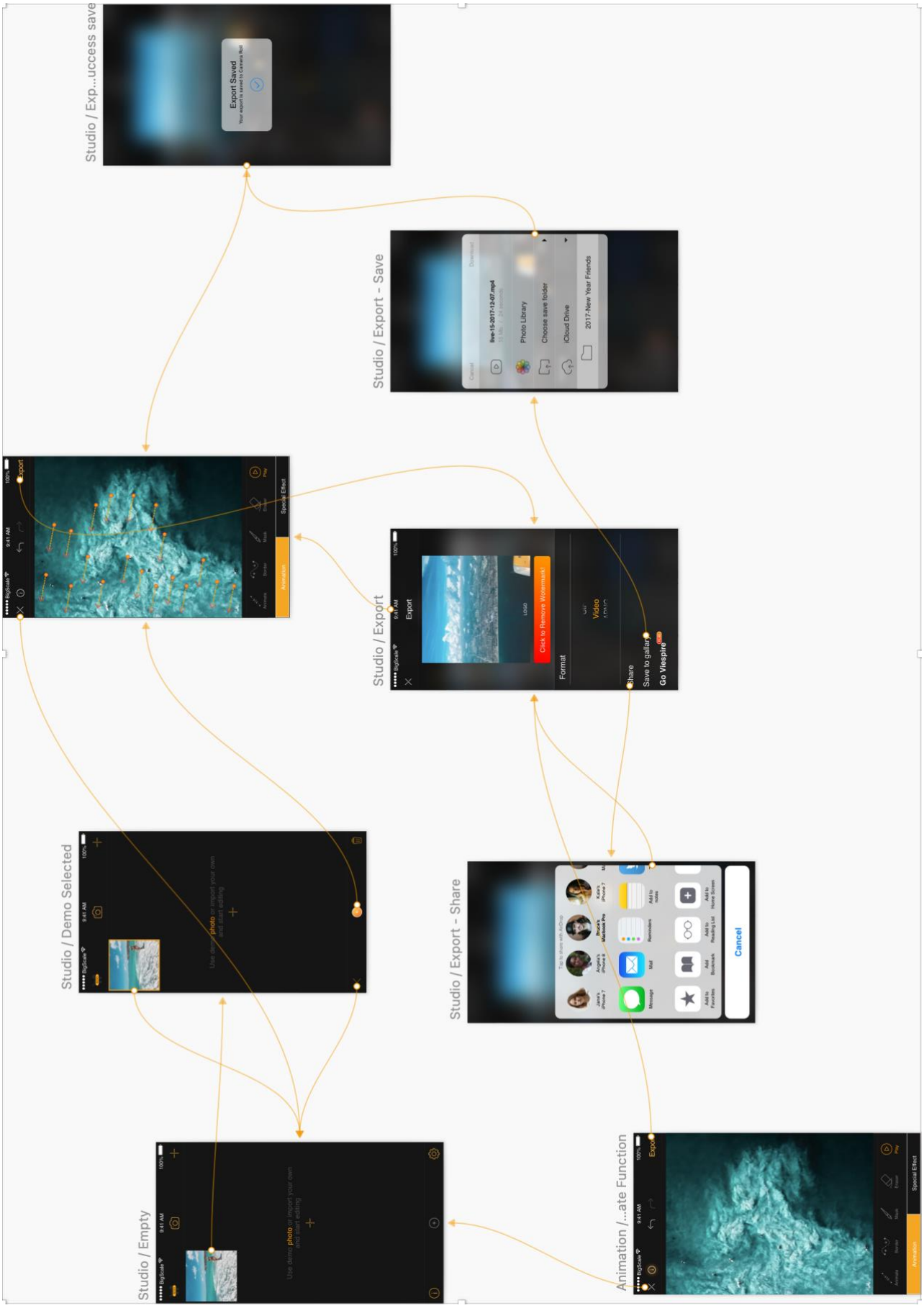

Рис. 4.7. Прототип інтерфейсу мобільного додатку.

Вайрфрейм це образ дизайну низької точності. Він має чітко показувати:

- основну групу контенту;
- структуру подання інформації.
- опис та базова візуалізацію взаємодії між інтерфейсом та користувачем.

Прототип – це середина на шляху до високоякісного інтерфейсу фінального продукту.

Вони повинні дозволити розробникам полегшити такі завдання:

- опис, як користувач сприймає контент та взаємодіє з інтерфейсом;
- тестування основи взаємодії користувача з інтерфейсом за аналогією з фінальним продуктом.

Отже задачею прототипування є симуляція фінальної взаємодії між користувачем та інтерфейсом. Для прототипа дозволяються не суттєві відмінності від інтерфейсу фінальний продукту, але має відображати основний функціонал додатку та позицій меню для користувача.

Взаємодії мають бути точно змодельовані та мати значну схожість із фінальним функціоналом інтерфейсу. Незалежність між інтерфейсом і функціональністю бекенда зазвичай упускається для зниження витрат і прискорення циклу розробки.

Для розробки інтерфейсу мобільного додатку є можливість пропустити стадію побудови вайрфреймів через те, що інтерфейс не є складним та перенавантаженим різними кнопками та меню, кількість функціонала доступного користувачу також достатня для створення одразу прототипу інтерфейсу.

За допомогою прототипу є можливість перевірити зручність використання майбутнього функціоналу та інтерфейсу для користувачів. На прототипі є можливість побудувати зв’язки між різними екранами додатку та кнопки, які відповідають за зміни стану додатку.

Розглянемо прототип додатку Рис. 4.7. більш детально. Прототип інтерфейсу складається з восьми фреймів, що забезпечують виконання доступу користувачу до функціоналу систему. Кількість фреймів інтерфейсу має бути оптимальним, щоб забезпечувати доступ до всіх функцій системи, але при цьому

не ускладнювати користувачу розуміння роботи додатку та доступ до функцій системи, тому прототип розробленого інтерфейсу відповідає даним вимогам.

Початковим екраном для додатку буде галерея зображень, що були завантажені в систему, при цьому користувач не має доступу до зображень інших користувачів. На даному екрані в користувача буде можливість видалити зображення, або вибрати зображення для створення анімації, при видаленні зображення з системи оригінальне зображення лишається в галереї пристрою користувача. Наступним екраном є редактор анімації, екран доступний користувачу після вибору зображення в галереї та кліку по кнопці запуску редактора. На екрані редактора у користувача є меню для налаштування параметрів анімації, редагування стану проекту та можливість повернутися на попередній екран. Також на даному екрані доступна кнопка експорту відео з анімацією, при натисненні якої, користувачу стає доступним екран вибору типу експорту: поділитися або зберегти. При виборі типу поділитися буде перехід на екран з вибором додатків в які є можливість поширити відео, якщо ж обрано збереження відео, то воно буде завантажене в галерею пристрою користувача. Після вибору способу експорту відео користувачу буде відображено екран з прогресом рендеру відеоряду анімації.

Прототипування дозволило створити простий та інтуїтивно зрозумілий інтерфейс користувача для різних типів екранів, що є дуже важливим при проектуванні додатків для мобільних пристроїв.

4.3. Розробка функціоналу.

4.3.1. Обґрунтування вибору засобів реалізації та опис модулів.

Отже для розробки мобільного додатку було обрано платформу iOS та мову програмування Swift. Ця мова програмування створено відносно не давно, але має переваги над мовою Objective-C. Swift запозичив багато з Objective-C, але він визначається не посиланнями, а типами змінних, які обробляє компілятор. За аналогічним принципом працюють багато скриптових мов. У той же час, він надає розробникам багато функцій, які раніше були доступні в Objective-C, такі як визначені найменування, узагальнення та перевантаження операторів.

Swift має вбудований великий пласт поширених програмних конструкцій, розроблених із застосуванням сучасних програмних патернів:

- Змінні повинні бути ініціалізовані до використання.
- Індекси масивів повинні перевірятися на помилку виходу за межі масиву.
- Цілі числа перевіряються на можливість переповнення.
- Опціонали гарантують, що значення nil будуть явно опрацьовані.
- Керування сміттям та керування пам’яттю.
- Обробка помилок дозволяє здійснювати контрольоване поновлення від непередбачених помилок.

Код, побудований з допомогою Swift, скомпільовано та оптимізовано таким чином, щоб максимально використати обчислювальні ресурси. Частина функцій мови виконується швидше, ніж інші мови програмування. Наприклад, сортування комплексних об’єктів виконується в 3,9 разів швидше, ніж у Python, і майже в 1,5 рази швидше, ніж у Objective-C [73].

Для розробки алгоритмів класифікації, кластеризації та реалізації штучної нейронної мережі обрано мову програмування Python. Python є популярною та потужною інтерпретованою мовою. На відміну від R, Python є повною мовою і платформою, які можна використовувати як для досліджень, чисельних розрахунків, так і для розробки систем. У Python є також багато модулів та бібліотек, які призначенні для виконання завдань машинного навчання [76].

Для розробки модулю API буде використовуватися мова програмування PHP8.1 з використанням фреймвьорка Symfony. Мова програмування PHP була спреціально розроблена для використання веб серверах, а версія PHP забезпечує найкращу стабільність та швидкість обробки запитів і виконання коду [74]. Фрейвьорк Symfony адаптовано для побудови REST API сервісів та налічує багато вбудованих пакетів, також для роботи з різними СУБД, що прискорить розробку даного модулю та його стабільну роботу і зручність підтримки [75]. Вибір програмної архітектури REST для побудови модуля API була зумовлена такими факторами, як можливість використовувати високопродуктивного та надійного зв’язку в необхідному масштабі. Легкість впровадження та масштабування, прозорість та крос-платформна сумісність будь-якої системи

API. Основні принципи архітектурного стиля REST: єдиний інтерфейс, відсутність збереження стану, многорівнева система, місткість кешу, стандартизація кодів запиту та відповіді.

Принцип єдиного інтерфейсу є конструктивною основою веб-сервісу на архітектурі REST. Цей принцип визначає стандартизацію структури запиту та відповіді. Стандартизований формат відповіді називають поданням. Подання може відрізнятися від того в якому форматі дані зберігаються на сервері, але сервер не може вернути данні в форматі, який не визначений в документації.

Принцип єдиного інтерфейсу накладає чотири архітектурних обмеження:

- Запити мають ідентифікувати ресурси. Це відбувається з допомогою єдиного ідентифікатора ресурсів.
- Відповідь має мати достатньо інформації у поданні, щоб клієнт міг ідентифікувати ресурс для його зміни чи видалення. Сервер має виконувати цю умову, зазвичай це досягається за рахунок відправки ідентифікаторів ресурсу або додаткових метаданих, які описують ресурс.
- Клієнт має отримати інформацію про доступні процеси обробки ресурсів подання. Це реалізується за допомогою відправки у відповіді метаданих, які описують інструкції, як клієнт може використати отриманий ресурс.
- Клієнт має отримати інформацію про всі зв’язані ресурсі, що необхідні для виконання описаного процесу. Реалізація цього обмеження може бути у варіантах: сервер надсилає ідентифікатори пов’язаних ресурсів; сервер надсилає гіперпосилання на пов’язані ресурси.

Відсутність збереження стану цей принцип REST архітектури відноситься до методу зв’язку між клієнтом і сервером. Суть цього принципу полягає в тому, що сервер виконує кожен запит клієнта незалежно від інших запитів даного клієнта та від інших клієнтів. Клієнт може запитувати ресурси паралельно, а сервер має забезпечити ізоляцію даних процесів. Це конструктивне обмеження REST API передбачає, що сервер може щоразу повністю зрозуміти та виконати запит.

Принцип багаторівневості системи має забезпечувати можливість клієнту підключатися до інших авторизованих посередників між клієнтом та сервером, а

сервер зобов’язаний надати відповідь клієнту. При цьому сервер може надсилати запити іншим серверам на отримання ресурсів, які будуть оброблені на стороні сервера, а потім будуть переданні клієнту у відповіді. Даний принцип надає можливість проектувати веб-сервіс, як простий інтерфейс для клієнта та сховати складну структуру отримання та обробки даних на стороні сервера. Тобто сам сервіс може складатися з деякої множини підсистем, які будуть сховані від користувача для зручності роботи з ресурсами.

Місткість кешу. Даний принцип забезпечує можливість кешування клієнтом отриманих ресурсів. Сервер при цьому має надавати метадані, які будуть надавати клієнту інформацію, як довго зберігати кешований ресурс або перед тим як повторно віддати ресурс, що не був змінений передати метаданні, що на клієнті наразі зберігається актуальна версія ресурсу. Цей принцип дає змогу зменшити кількість інформації, яку потрібно передавати.

Стандартизація кодів запиту та відповіді. Цей принцип полягає в стандартизації дій над ресурсами в залежності від HTTP запиту клієнта. Тобто запит POST – це створення ресурсу, GET – це отримання ресурсу, DELETE – це видалення ресурсу. При цьому сервер також має відповідати стандартизованими HTTP кодами в залежності від результату виконання запиту, наприклад код 201 – це ресурс був створений, 404 – ресурс не був знайдений на сервері [88].

Розроблений модуль API, буде мати такі методи:

- «/image» – тип запиту POST. В тілі запиту обов’язково має будти переданий компонент «Image». У випадку, коли запит був успішно оброблений, метод вертає 201 код та у тілі відповіді компонент «DatabaseRowInformation», що містить в собі ідентифікатори зображення в базі даних. Метод буде зберігати завантажене в мобільний додаток зображення в базу даних системи.
- «/selected-objects» – тип запиту POST. Метод приймає в тілі запиту компонент «ImageSelectedObjects», цей параметр є обов’язковим. Якщо метод успішно обробив запит, то відповідь буде надіслана 201 кодом, а в тілі відповіді буде переданий масив компонентів «DatabaseRowInformation», який буде містити ідентифікатори об’єктів в

базі даних системи. Даний метод реалізовано для збереження в системі об’єктів, які користувач виділяв для створення анімації, при використанні ручного режиму вибору ОПІ. Далі ці об’єкти пройдуть процес класифікації та кластеризації і будуть використанні в тренувальних вибірках для ЗНМ.

- «/objects-searching» – метод підтримує тип запитів POST та GET. Якщо мобільним додатком буде надіслано POST запит на даних метод, то буде розпочато процес пошуку об’єктів на зображені, що було збережено в системі раніше. В даному випадку метод очікує, що тіло запит обов’язково буде містити компонент «DatabaseRowInformation» в якому будуть ідентифікатори зображення в базі даних. У відповідь надсилається код стану 202, а е тілі відповіді компонент «ProcessInformation», який містить ідентифікатор процесу в системі. При отриманні GET запиту модуль буде перевіряти чи процес пошуку об’єктів був завершений, обов’язковим є параметр *process_id*, який передається в рядку запиту. Якщо процес пошуку об’єктів ще триває, то у відповідь буде надісланий код стану 102 та пусте тіло відповіді, при цьому затримка перед повторним запитом має складати не менше 2 секунд. Якщо процес пошуку об’єктів був виконаний успішно, то метод верне відповідь зі станом 200 та у тілі відповіді масив компонентів «FoundObject», що будуть містити координати пікселів знайдених об’єктів.

У випадку помилки обробки запиту всі методи вертають код стану 500 та компонент «Error». Виключенням є метод «/objects-searching» у випадку GET запиту може додатково вернути код стану 404, якщо не було знайдено процесу з переданим методу ідентифікатором *process_id*.

Компоненти даних, які використовуються для передачі інформації між модулем API та мобільним додатком: «DatabaseRowInformation», «Image», «PixelCoordinate», «FoundObject», «SelectedObject», «ImageSelectedObjects», «ProcessInformation» та «Error».

Компонент «DatabaseRowInformation» використовується для передачі інформації, що можуть ідентифікувати зображення або об’єкт в базі даних. Має таку структуру:

1. id – ідентифікатор рядка в базі даних в якому збережено зображення чи об’єкт, має тип даних integer.
2. unique_hash_name – унікальний 32 символьний хеш, що був присвоєний зображенню чи об’єкту системою, тип даних string.

Компонент «Image» використовується для передачі зображення на сервер. Структура компоненту:

1. file_name – string. Назва файлу картинки в пристрої користувача.
2. file_content – тип даних sting, формат hexadecimal bytes. В цьому параметрі передається відображення зображення у вигляді байтів закодованих у шістнадцятковий код.

Для передачі координат пікселів, що належать контуру об’єктів на зображенні використовується компонент «PixelCoordinate». Структура компоненту складається з параметрів *x* та *y*, що відповідають положенню пікселя по осі абсцис та ординат відповідно. Тип даних параметрів integer.

В компоненті даних «FoundObject» передається інформація про координати об’єкту, що був знайдений на зображенні після виконання процесу пошуку об’єктів. Компонент має єдиний параметр *pixel_coordinates,* що складається з масиву компонентів «PixelCoordinate».

Компонент «SelectedObject» використовується для передачі інформації про об’єкт, що був виділений користувачем вручну для створення анімації. Структура компоненту даних:

1. content – тип даних sting, формат hexadecimal bytes. В цьому параметрі, у вигляді байтів закодованих у шістнадцятковий код, передається відображення об’єкту.

2. pixel_coordinates – тип даних array. Параметр складається з масиву компонентів даних «PixelCoordinate», що місять координати пікселів, які належать контуру виділеного об’єкту.

Компонент даних «ImageSelectedObjects» буде використовуватися для передачі даних про масив виділених користувачем об’єктів на зображенні. Має такі параметри:

1. image_info – тип даних object. Складається з одного екземпляра компоненту «DatabaseRowInformation», в якому зберігається інформація про ідентифікатори зображення в базі даних системи.
2. objects – тип даних array. Містить в собі масив компонентів типу «SelectedObject», тобто інформацію про всі виділенні користувачем об’єкти на зображенні з ідентифікаторами з поля image_info.

Для передачі ідентифікатор процесу пошуку об’єктів, використовується компонент даних «ProcessInformation». Даний компонент містить тільки один параметр *id*, який має тип даних string, а формат даних uuid.

Для передачі деталізованих даних про помилку, що виникла при обробці запиту методами API реалізовано компонент «Error». Структура параметрів компоненту:

1. type – тип даних string. Містить інформацію про тип помилки, що виникла в системі.
2. title – тип даних string. Коротка назва помилки в системі.
3. status – тип даних integer. Числовий код, який відображає статус помилки в системі. 300 – помилка низького рівня критичності, 400 – помилка помірного рівня критичності, 500 – критична помилка в системі.
4. detail – тип даних string. Розширений опис помилки.

Параметри всіх компонентів даних модуля API є обов’язковими для передачі, як у тілі запиту, так і у тілі відповіді методів. Винятком є параметр detail у компонентів «Error», який може не містити в собі інформацію у випадку, якщо

не був сформований детальний опис помилки системи. Розроблені методи модулю API в повному обсязі відповідають обраному стандарту REST API та їх кількість достатня для виконання функціоналу, який був описаний у вимогах до системи.

4.3.2. Опис топології згорткової нейронної мережі.

Розглянемо більш детально архітектуру згорткової нейронної мережі (Рис. 4.8), яка буде використовуватися для пошуку об’єктів.

На вхід згорткової нейронної мережі будуть подаватися растрові зображення розмірністю 70x70 пікселів. При виборі зображень більшої розмірності буде суттєво рости обчислювальна складність, тому данні параметри було обрано, як оптимальні для зменшення обчислювальної складності та збереження прийнятної точності розпізнавання.

Растрові зображення можуть бути розбиті на три канали RGB, тому вхідний шар має 3 карти розмірністю 70x70. Вхідні данні нормалізуються в діапазоні [0, 1] за формулою:

$$N = \frac{c}{255},$$

де c – значення каналу конкретного пікселя.

Згортковий шар містить в собі 6 карт розмірністю 64x64, синаптичні ядра розмірністю 7x7 включено до кожної карти. При виборі більшої кількості карт точність розпізнавання покращувалась на 5%, але розрахункова складність збільшувалася на 40% відносно обраної топології (Рис. 4.9 та 4.10). Якщо обрати меншу розмірність ядер, то ЗНМ не виділяла низьку важливих ознак, що зменшувало точність розпізнавання. Отже тому данні параметри приймаються, як оптимальні для забезпечення роботи алгоритму.

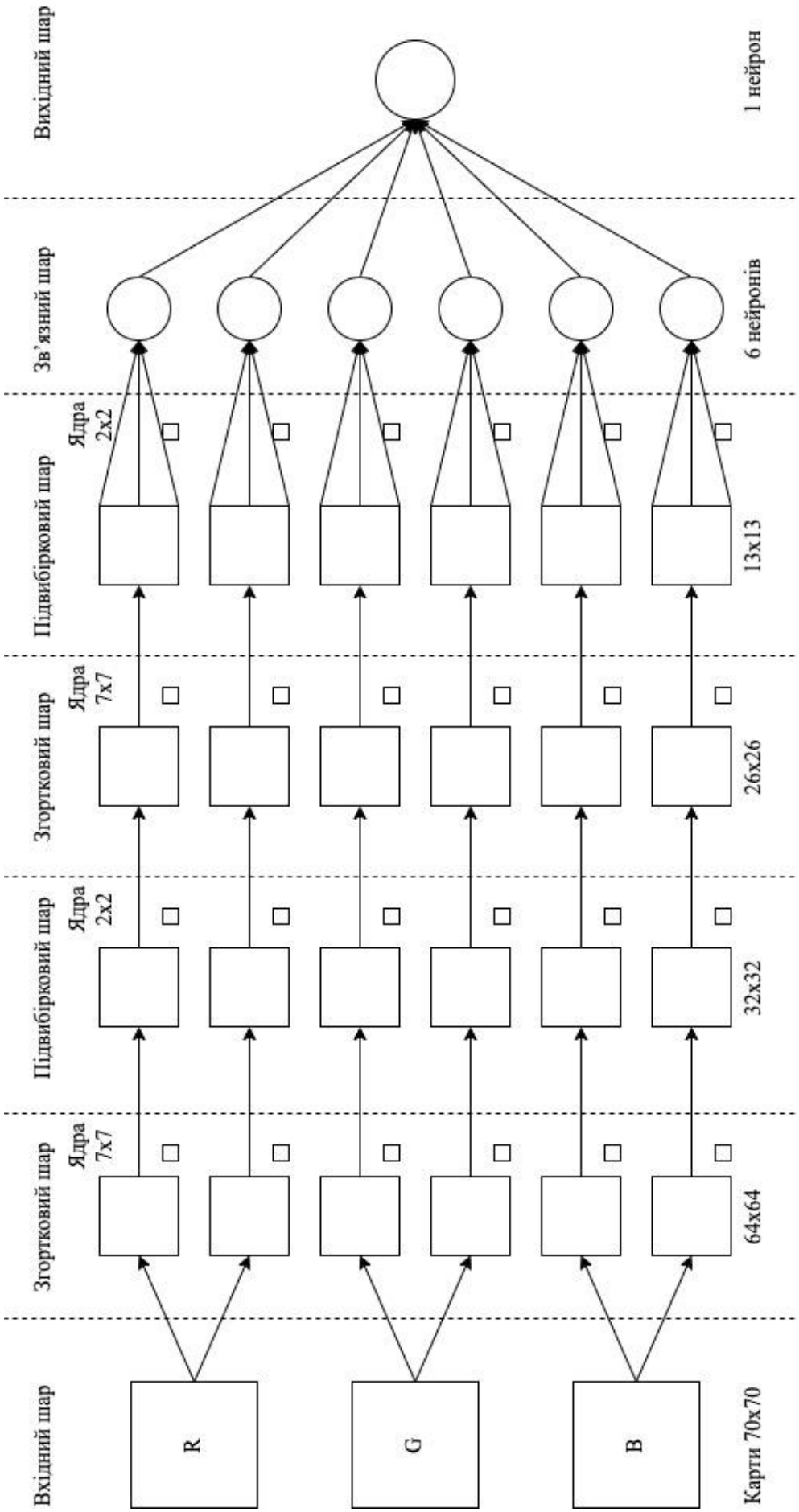

Рис. 4.8 Архітектура згорткової нейронної мережі.

Розміри карт розраховувалися за формулою:

$$w = w_i - w_c + 1\,,$$

$$h = h_i - h_c + 1\,,$$

де w_i, h_i – ширина та висота карти попереднього шару; w_c, h_c – ширина та висота ядра поточного шару [77].

На початку значення кожної карти згорткового шару дорівнюють 0. Значення ваг ядер задаються випадковим чином з діапазону [-0.5, 0.5]. Згортання буде проводитися за формулою:

$$(Im \cdot g)[n,n] = \sum_{k,l} Im[n-k, n-l] * g[k,l],$$

де Im – матриця вхідного зображення, g – ядро згортки, n – розмірність, k, l – конкретні значення в матриці на кроці циклу [78].

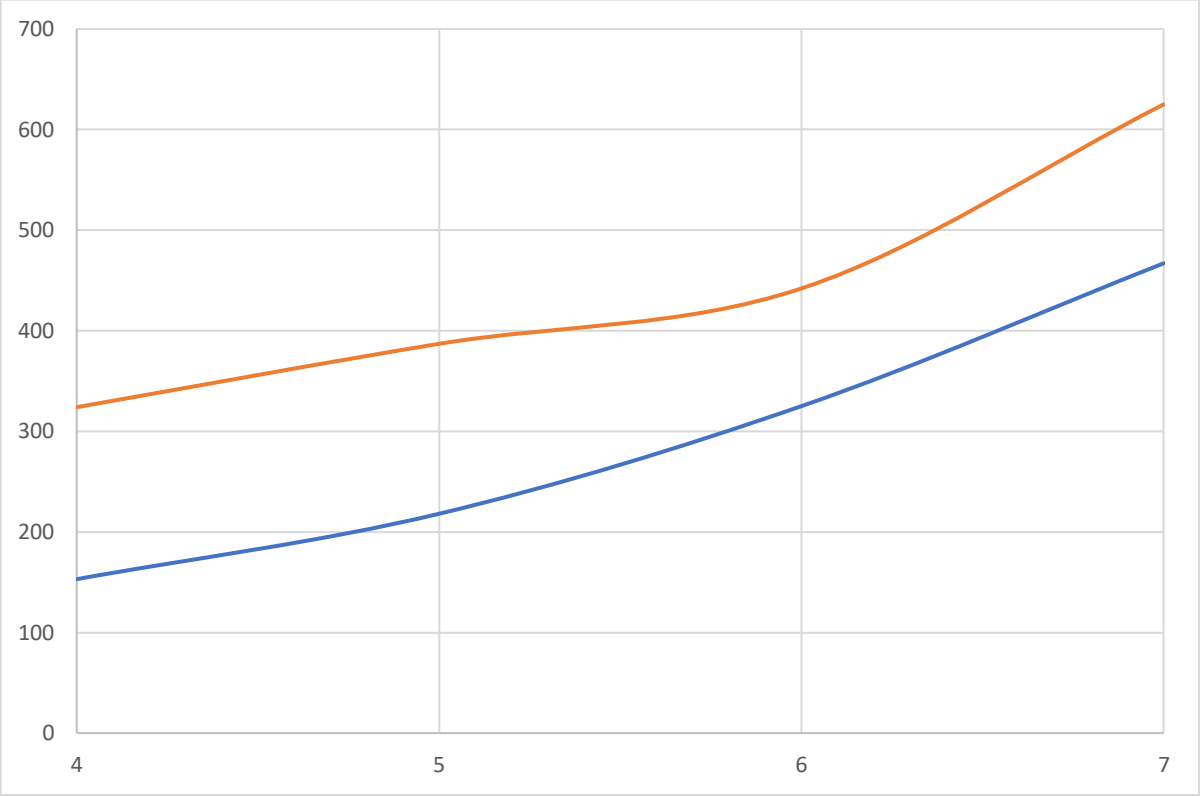

Рис. 4.9. Залежність використання обчислювальних ресурсів (оперативна пам’ять, процесор) в залежності від кількості карт ЗНМ.

Підхід більшого розміру матриці було обрано, як метод обробки країв зображення, це забезпечує ліпший аналіз ознак об’єктів на границях зображення.

Отже кінцевий варіант формули згортки, якщо враховувати гіперболічний тангенс функцією активації, прийме вигляд:

$$x_i^l = tanh\left(\sum_j x_j^{l-1} \cdot k_i^l + b_i^l\right),$$

- де x_i^l – карта ознак i вихідного шару l;
- b^l – коефіцієнт зсуву шару l для карти ознак i;
- k_i^l – ядро згортки шару l для карти ознак i.

Рис. 4.10. Залежність середньої точності розпізнавання від кількості карт ЗНМ.

За допомогою функції RelU буде проводитися формування ознак підвибірки. Кількість класів об'єктів, що аналізуються ШНМ, визначають кількість нейронів вихідного шару [79]. Однак, якщо класи додаються динамічно, як у випадку з даної системою, їх кількість буде збільшуватися з часом використання системи. Такий підхід накладає проблеми масштабування ЗНМ, отже тому було прийнято рішення використовувати лямбда-архітектурний підхід для алгоритму нейронної мережі та зберігати в базі даних отриманні вагові коефіцієнти для кожного класу.

Розроблений підхід прибирає обмеження відповідності класів та нейронів вихідного шару, це дає змогу мати один нейрон вихідного шару, який буде визначати приналежність об'єкту певному класу. Для самого алгоритму

згорткової нейронної мереєі будуть запускатися окремі процеси. Кількість процесів буде відповідати кількості класів, що аналізуються, а вагові коефіцієнти будуть отримуватися з БД. Такий підхід дозволить масштабувати нейрону мережу та зменшить розрахункову складність при навчанні мережі.

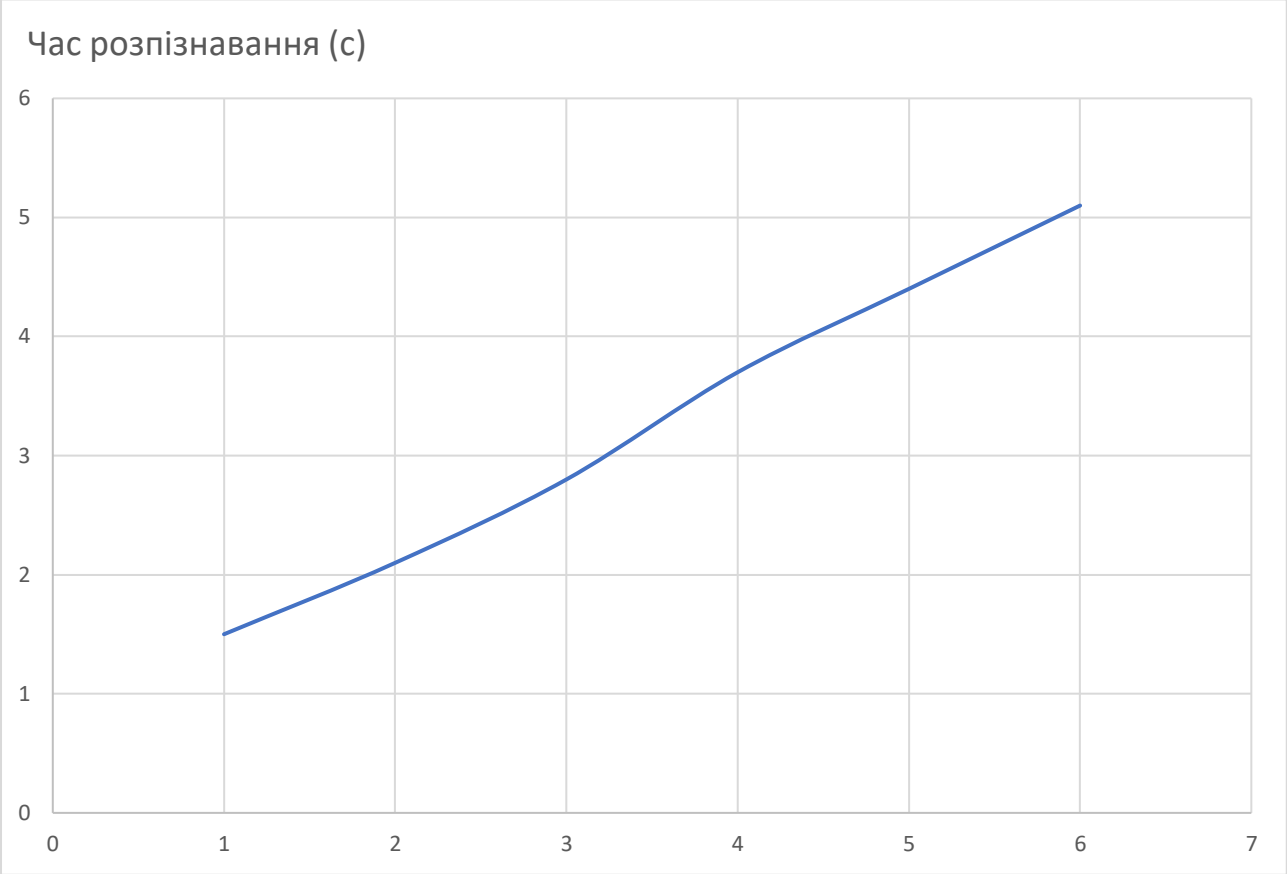

Рис. 4.11. Час виконання розпізнавання без використання лямбда-архітектури.

Проаналізуємо час виконання алгоритму розпізнавання з використанням лямбда-архітектури (Рис. 4.12) зі стандартним підходом до побудови вихідних шарів (Рис. 4.11). Аналіз будемо проводити відносно кількості класів об’єктів, які потрібно розпізнати на зображенні. Виходячи із отриманих даних експерименту є можливість зробити висновок, що при використанні лямбда-архітектурного підходу час виконання алгоритму не залежить від кількості класів, а тільки від складності структури визначеного класу, тобто час виконання буде дорівнювати часу, який було витрачено на розпізнавання найбільш складного класу об’єктів. При цьому при використанні стандартного підходу до побудови згорткових нейронних мереж, де кількість нейронів вихідного шару

дорівнює кількості класів, що аналізуються, час дуже залежить від кількості класів, що одночасно подаються на нейронну мережу.

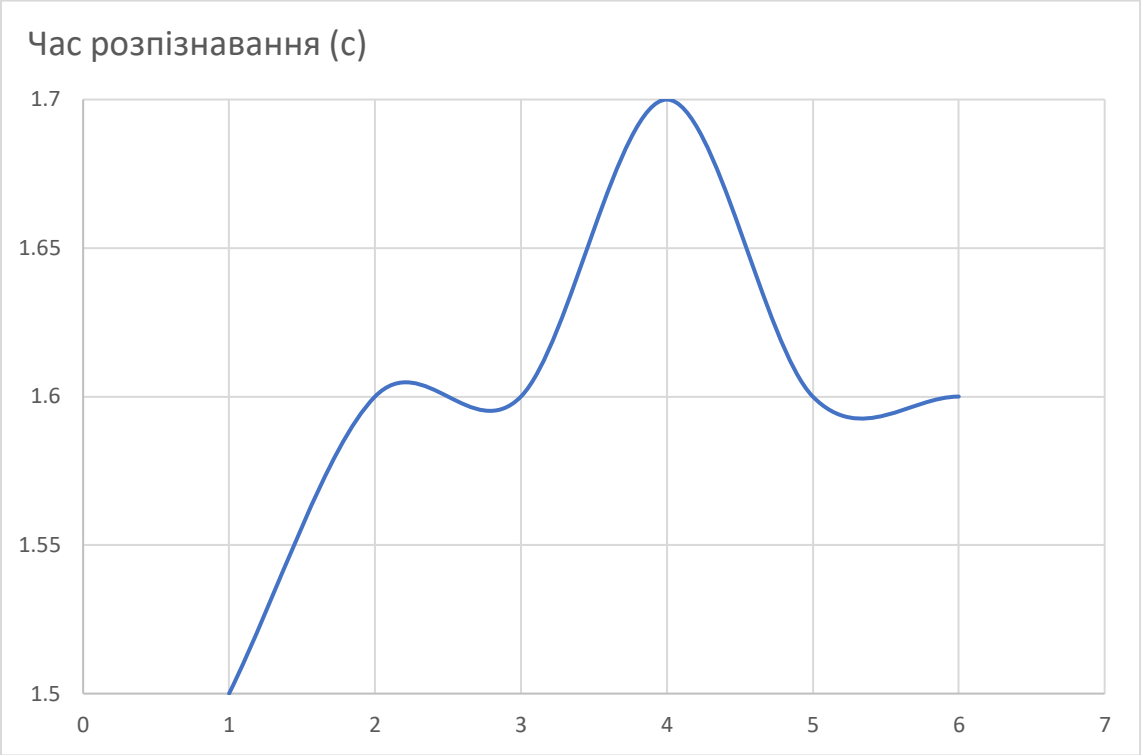

Рис. 4.12. Час виконання розпізнавання з використанням лямбда-архітектури.

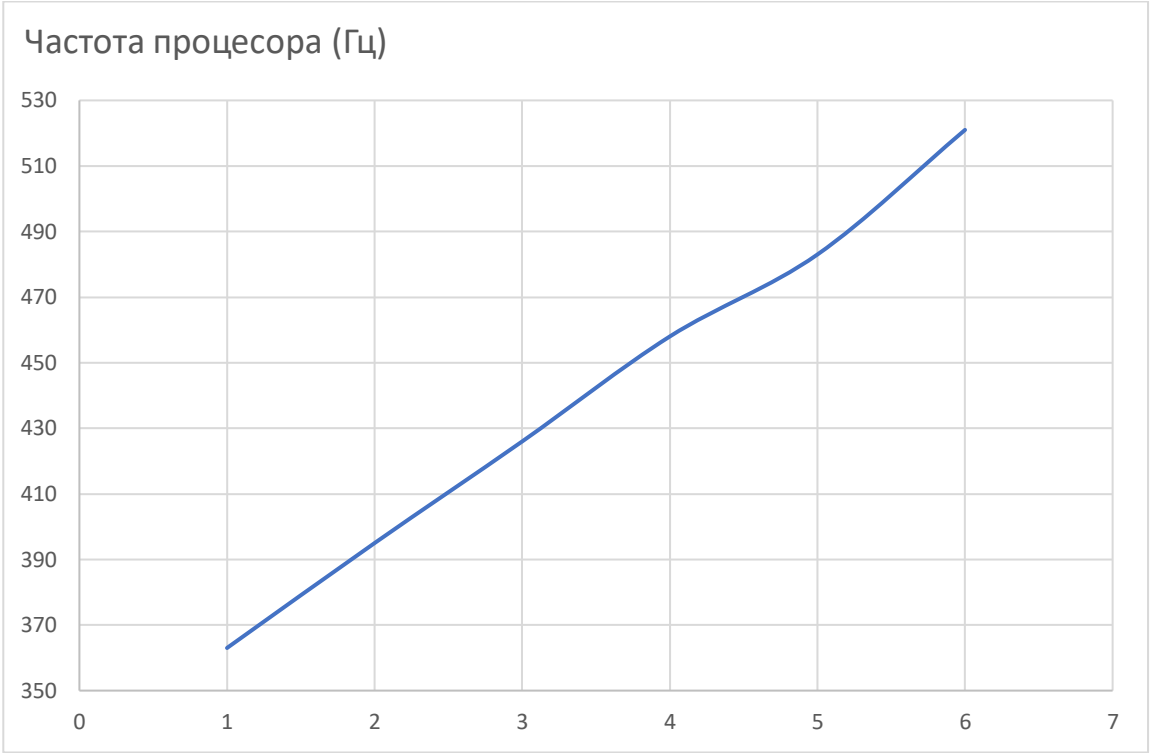

Рис. 4.13. Завантаженість процесора при використанні стандартного підходу до побудови нейронної мережі.

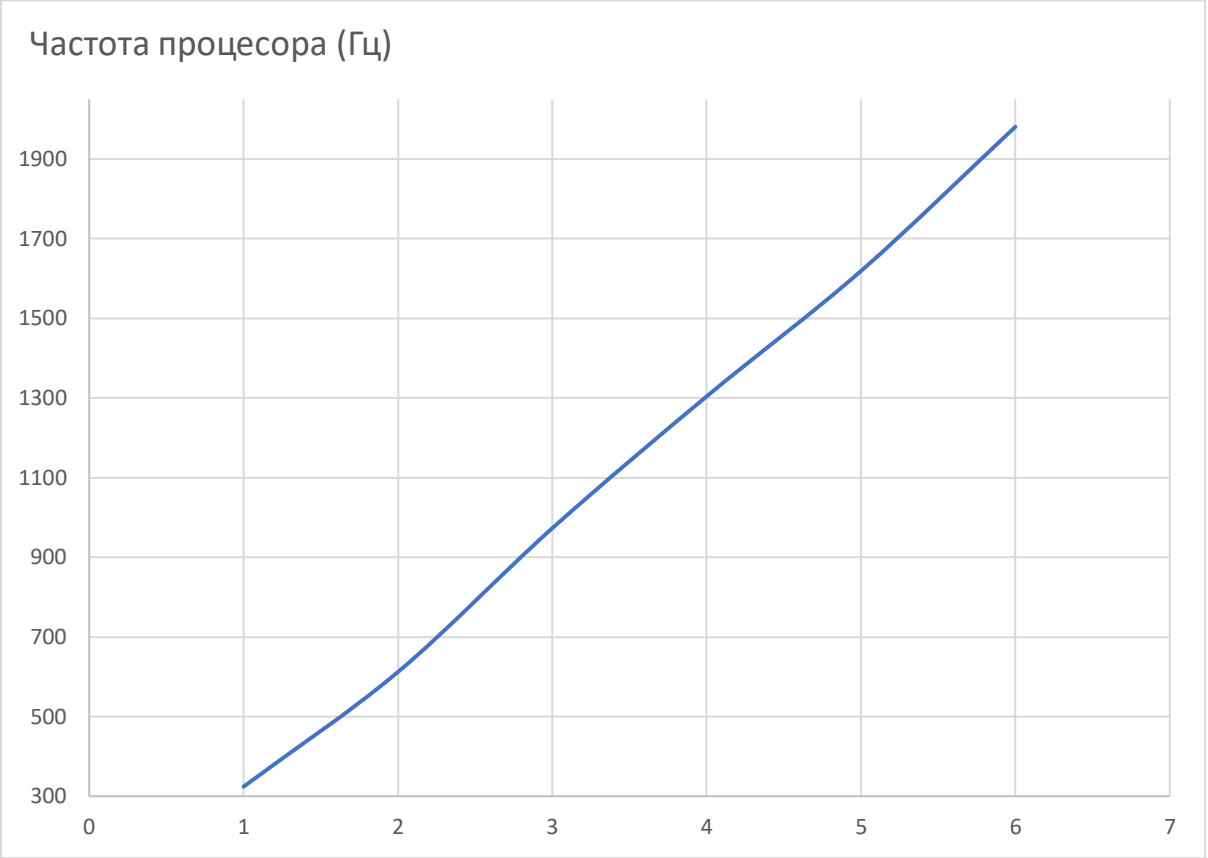

Рис. 4.14. Завантаженість процесора при використанні лямбда-архітектури.

Однак при підході з використанням лямбда-архітектури найбільш проблемним є завантаженість обчислювальних ресурсів (Рис. 4.14).

Дана архітектура програє стандартному підходу (Рис. 4.13) до побудови топології згорткових нейронних мереж. Але, як зазначалося вище лямбда-архітектурний підхід дозволить запускати кількість процесів, які дорівнюють кількості класів, що потрібно проаналізувати, тоді як при використанні нейронної мережі в стандартному підході, нам потрібно буде розраховувати ваги вихідних нейроннів для всіх класів наявних в системі. Тому із збільшенням кількості класів в системі оптимальним буде розпізнавати тільки ті класи об’єктів, які подаються на вхід.

4.4. Апробація результатів.

Розглянемо готовий мобільний додаток для користувача системи. Для того, щоб почати створення анімації об’єктів, користувачу потрібно завантажити в

додаток зображення, це можливо зробити як і з галереї так і зробити фото з камери пристрою.

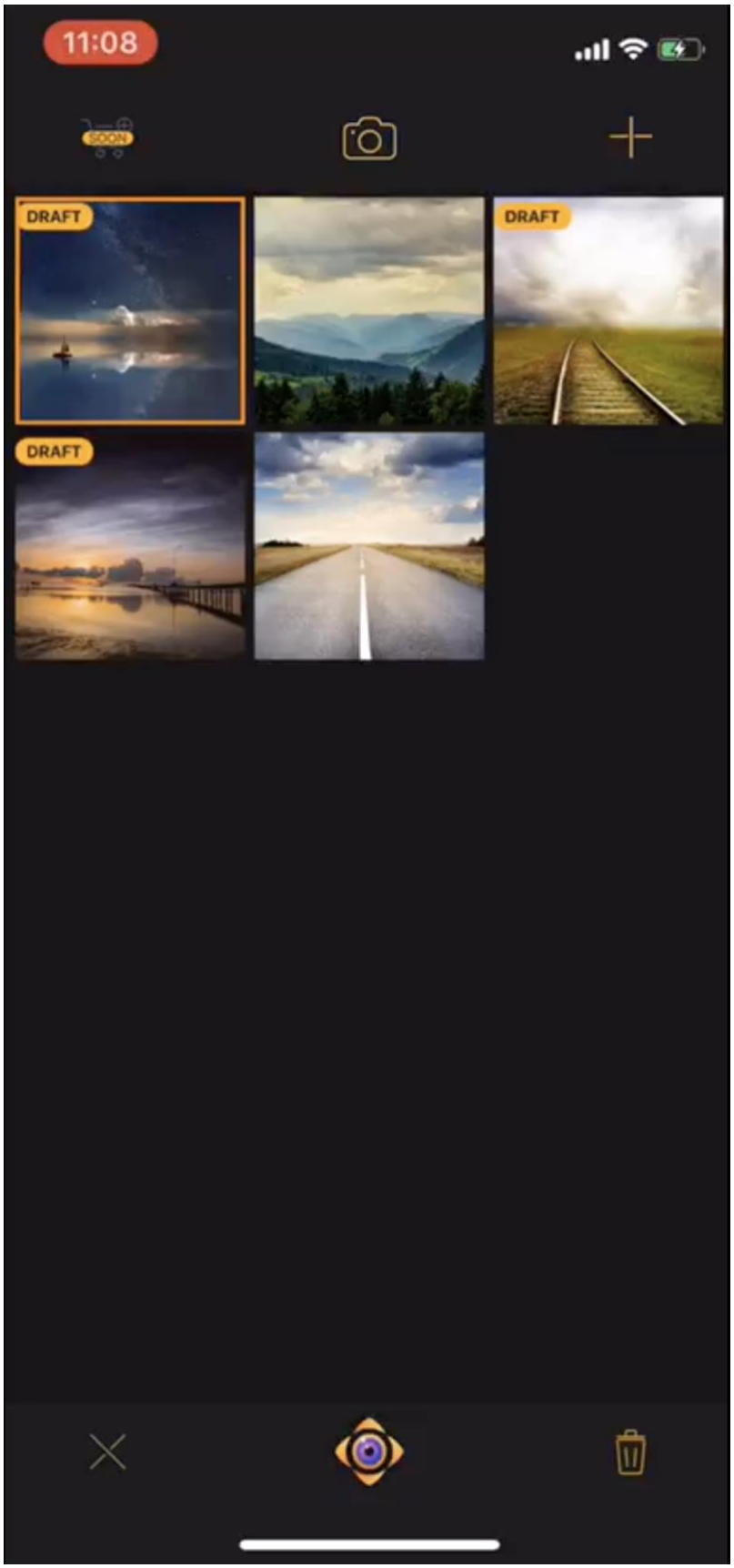

Рис. 4.15. Екран галереї завантажених зображень в додатку.

Рис. 4.16. Вікно редактору анімації.

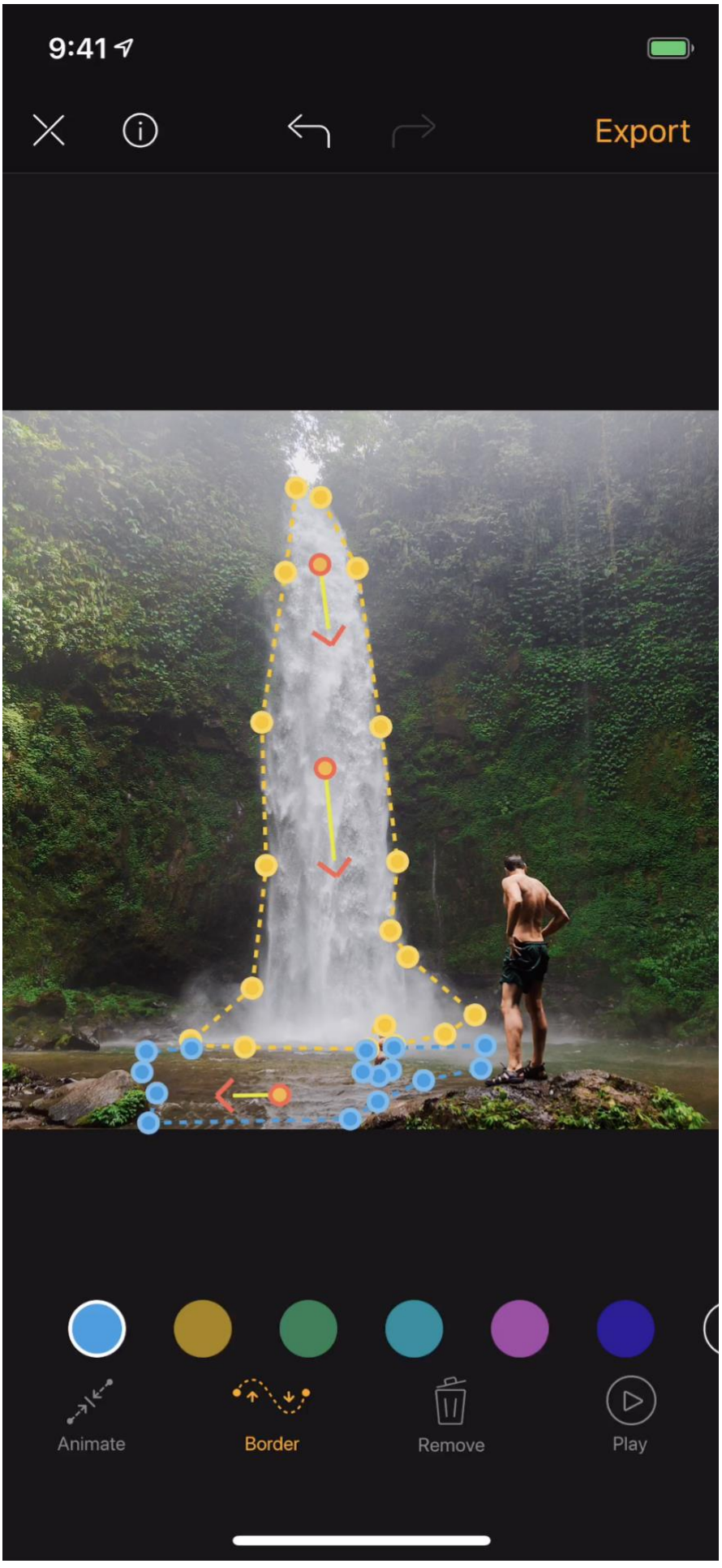

Рис. 4.17. Приклад налаштувань редактора для створення анімації.

На Рис. 4.15 показано початковий екран мобільного додатку на якому є кнопки для завантаження відео з галереї, з камери пристрою. Можливість видалити зображення з додатку та кнопка для переходу в редактор створення анімації. Для всіх завантажених зображень зберігається стан з редактора створення анімації, тому користувач в любий момент може повернутися до редагування раніше створених проектів в редакторі.

На Рис. 4.16 зображено вікно редактора анімації. На екрані редактора існують дві панелі редагування: верхня та нижня. На нижній знаходяться кнопки: Animate, Border, Remove, Play, а на верхній – кнопки: Close, Manual, Undo, Redo та Export.

Кнопка Animate призначена для визначення векторів анімації на зображені. Вектори анімації зображаються на екрані, як направленні стрілочки. За допомогою них користувач може обирати кут та напрямок анімації області.
Кнопка Border служить для виділення областей анімування. Для зручності користувача, щоб легше було розрізняти лінії обмеження, зроблено вибір кольору ліній (Рис. 4.18).

Вікно редактору анімації з вибором швидкості анімації подано на рис 4.19. Для того, щоб побудували лінію обмеження області, користувачу потрібно виставляти кутові точки на вигинах об’єкту, а далі додаток автоматично з’єднає точки переривчастою лінією. Приклад, як виглядає побудова двох областей для анімування зображено на Рис. 4.17.

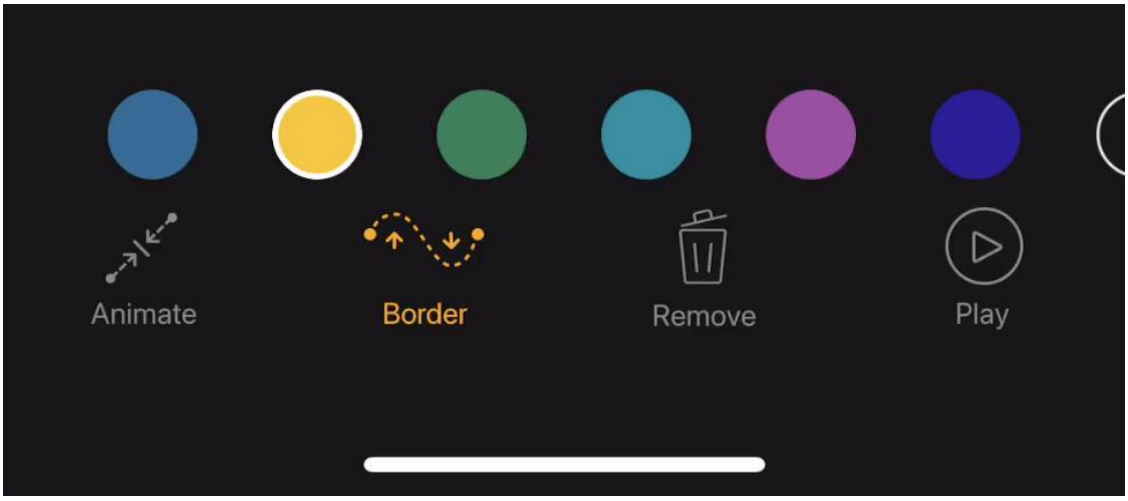

Рис. 4.18. Вікно редактору анімації з вибором кольору лінії обмеження.

Кнопка Remove має простий функціонал призначення для видалення стрілок, анімації та точок обмежувальних ліній.

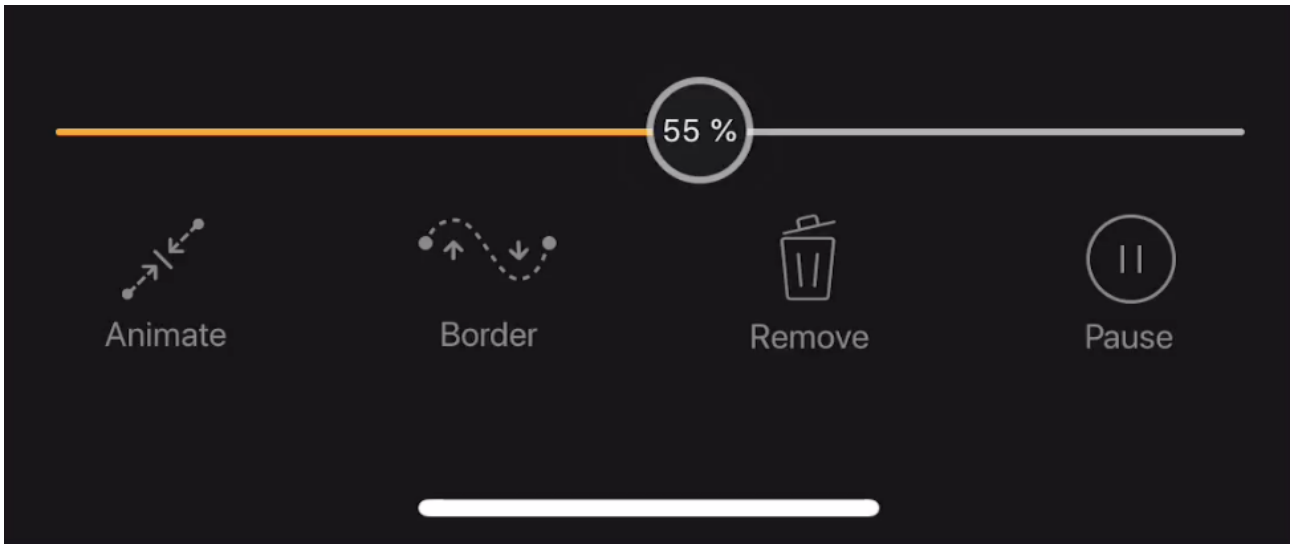

Рис. 4.19. Вікно редактору анімації з вибором швидкості анімації.

Кнопка Play запускає анімацію за заданими користувачем в редакторі параметрам. При натисканні кнопки, з’являється повзунок вибору швидкості анімації (Рис. 4.19). При зміні швидкості анімації зміни одразу з’являються в вікні редактора, щоб користувачу зручніше було обрати параметр, який би забезпечував найкращий результат. Сама анімація обмежена 4 секундами, але налаштована на повторювання, що створює ефект її неперервності.

Перейдемо до розгляду верхнього меню. Кнопка Exit зображена в меню хрестиком, при її натисненні користувача поверне до екрану з галереєю зображень, при цьому всі налаштовані параметри поточного проекту будуть збережені.

Кнопки Undo та Redo використовується для відміни або повернення стані параметрів анімування. Кнопка Export дозволяє користувачу зберегти відео в популярних відео-форматах, або розмістити його одразу в соціальних мережах. При натисканні кнопки Export з’являється меню з вибором типу збереження відео (Рис. 4.20). Після вибору типу збереження, почнеться процес рендеру відео.

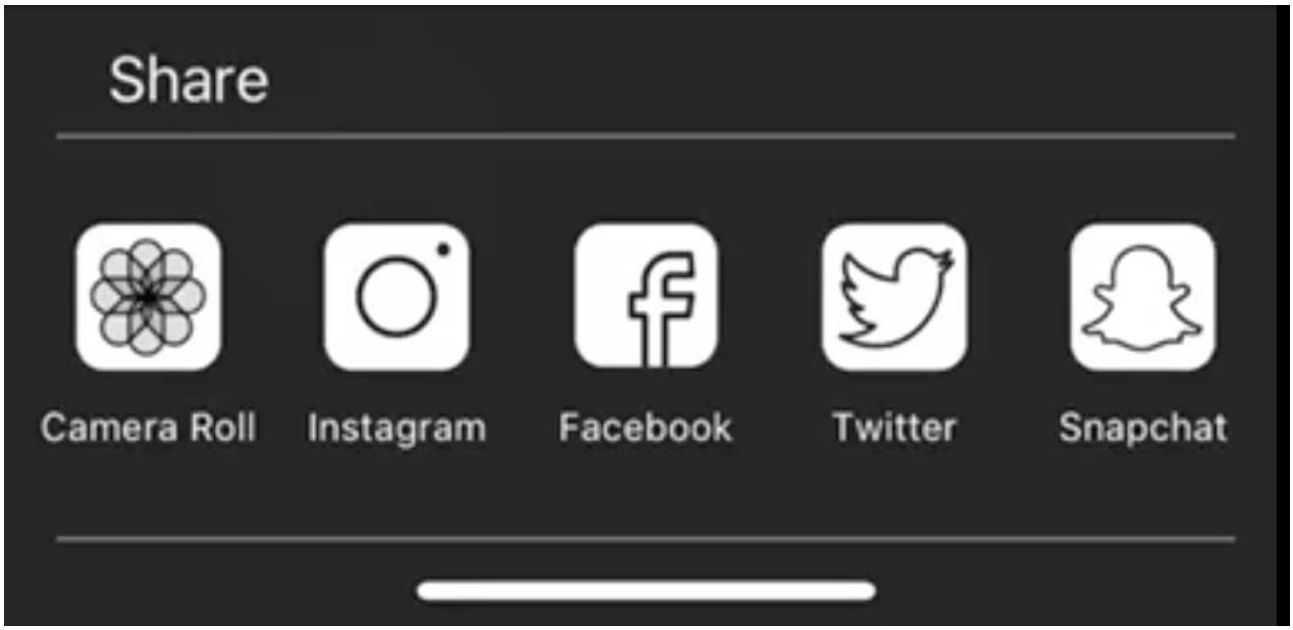

Рис. 4.20. Меню вибору типу збереження.

Висновки до четвертого розділу

В даному розділі було розроблено діаграму станів системи, на якій відображено основні функції системи та як вони взаємодіють між собою. На діаграмі потоків даних надано інформацію про те, якими пакетами даних функції системи мають обмінюватися між собою в процесі роботи. Було представлено на діаграмі обрані компоненти системи, які забезпечують стабільну роботу функцій та процесів, обґрунтовано за допомогою експериментів та відображено на графіках вибір сервісів для реалізації процесів системи та обраного архітектурного підходу. Архітектура системи було обрано з урахуванням обчислювальних можливостей мобільних пристроїв, тому було вирішено, обчислення, які мають велику ємнісну складність, виконувати на хмарних сервісах із використанням лямбда-архітектурного підходу.

Проведено прототипування інтерфейсу користувача, що дало змогу обрати інтуїтивно зрозумілий інтерфейс користувача та відобразити можливості використання системи користувачем.

Розроблено архітектуру згорткової нейронної мережі, яка би забезпечувала стабільну роботу пошуку об’єктів на зображеннях та дотримувалась концепції лямбда-архітектурного підходу.

В розділі представлено отриманні результати роботи, а саме проілюстровано, за допомогою скріншотів, роботу системи для створення анімації статичних об’єктів на зображеннях.

Проаналізувавши отриманні результати можна зробити висновок, що було досягнуто основні вимоги до створення системи.

ВИСНОВКИ

У дисертаційній роботі розв’язано актуальне наукове завдання розроблення методів та засобів опрацювання об’єктів на зображенні для їх анімування.

1. Здійснено аналіз предметної області, інформаційних процесів опрацювання об’єктів на аналіз, а саме: виділення контурів, ідентифікації, класифікації та анімування. Це дало змогу виділити невирішеня задачі існуючими методами та здійснити постановку задачі.

2. Розроблено метод контурного аналізу та пошуку об’єкта на зображенні з використання оператора Собеля на основі застосування маски коефіцієнтів для середніх значень. Це стало основою для розроблення моделі опрацювання зображення. Також метод контурного аналізу може бути реалізований на обмежених обчислювальних ресурсів 500 МБ оперативної пам’яті та 1 Ггц частоти процесора, що цілком відповідає поставленим вимогам для використання на мобільних пристроях.

3. Розроблено модель класифікації та кластеризації об’єктів із попереднім застосуванням аналізу геометричних ознак та гістограми градієнтів інтенсивності та з використанням оригінальної архітектури згорткової нейронної мережі. Як функцію активації використано гіперболічний тангенс. В структурі нейронної мережі є 2 зготкових шари, 2 підвибіркових шари та один повнозв’язний шар. Це дало змогу опрацьовувати об’єкти зі стійкістю до повороту або масштабування об’єкту, зміною ракурсу без необхідності їх повторного аналізу. Також це дало змогу зберігати проаналізовані об’єкти в базі даних бех необхідності їх повторного опрацювання.

4. Розроблено метод анімування статичних об’єктів на зображенні із застосуванням афінних перетворень, а також вектора анімації із значеннями, залежними від часу. Це дозволило зберігати пропорції паралельних об’єктів.

5. Розроблено метод пошуку об’єкта на зображенні з використанням оператора Собеля та Прюітта шляхом врахування середньої яскравості зображення та застосування алгоритмів адаптивної порогової обробки та

Брезенхема. Це уможливило опрацювання масштабованих зображень та зображення, отримані з іншого ракурсу.

6. На основі методів було розроблено архітектуру системи з орієнтуванням на мобільні пристрої. Для застосування алгоритмів у режимі реального часу запропоновано використовувати лямбда-архітектуру, що забезпечує масштабованість пам’яті та зменшує навантаження на сервер у 2 рази. Розроблено мобільний додаток, у якому реалізовано усі наукові результати.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Здор С.Е., Шираков В.Б. Оптический поиск и распознавание. – М.: Наука, 1973. – 237 с.
2. Fukumi M., Omatu S., Takeda F. Rotation – invariant neural pattern recognition system with application to coin recognition, // IEEE Trans. – 1992. – V. 3, No. 2. – P. 272 – 278.
3. Leibe, Bastian, and Bernt Schiele. "Analyzing appearance and contour based methods for object categorization." 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.. Vol. 2. IEEE, 2003
4. Chen, Hao, et al. "DCAN: Deep contour-aware networks for object instance segmentation from histology images." Medical image analysis 36 (2017): 135-146.
5. Yang, J., Price, B., Cohen, S., Lee, H., & Yang, M. H. (2016). Object contour detection with a fully convolutional encoder-decoder network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 193-202).
6. Peleshko, Dmytro, et al. "Design and implementation of visitors queue density analysis and registration method for retail videosurveillance purposes." 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP). IEEE, 2016.
7. Richard O. Duda, Peter E. Hart. Pattern classification and scene analysis / Richard O. Duda, Peter E. Hart - New York : Wiley, 1973 – 482 pages.
8. Canny J.F. (1983) Finding edges and lines in images, Master`s thesis, MIT. AI Lab. TR – 720 pages.
9. Довбиш А. С. Основи теорії розпізнавання образів: навч. посіб. : у 2 ч. / А. С. Довбиш, І. В. Шелехов. Суми: Сумьский державний університет, 2015. Ч.1. 109 с.
10. Gupta, Dipalee, and Siddhartha Choubey. "Discrete wavelet transform for image processing." International Journal of Emerging Technology and Advanced Engineering 4.3 (2015): 598-602.

11. Marr, David, and Ellen Hildreth. "Theory of edge detection." Proceedings of the Royal Society of London. Series B. Biological Sciences 207.1167 (1980): 187-217.
12. Bartfai G.A. Comparison of Two ART-base Neural Networks for Hierarchical Clustering // ANNES'95, The Second New Zealand International Two-Stream Conference On Artifical Neural Networks and Expert Systems, IEEE Computer Society Press, 1995.–P.83-86.
13. Carpenter G.A., Grossberg S., Markuzon N., Reynolds J.H., Rosen D.B. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps // IEEE Trans. Neural Networks, 1992.– vol.3.–P.698-713.
14. Moody J, Darken C.J., Fast Learning In Netwoks of Locally Tuned Processing Units // Neural Computation, 1995.–vol.1.– P.281-294
15. Leibe, Bastian, and Bernt Schiele. "Analyzing appearance and contour based methods for object categorization." 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.. Vol. 2. IEEE, 2003
16. Marques de Sá J. P. Pattern recognition: concepts, methods, and applications / Marques de Sá J. P. — Springer, 2001. — 318 p.
17. T. Hastie et al., The Elements of Statistical Learning, Second Edition, Springer Science+Business Media, 2009, 745 p.
18. B.S. Everitt, S.Landau, M. Leese. Cluster Analysis. – London: Arnold Publishers, 2001. – 343p.
19. B. Mirkin. Mathematical Classification and Clustering, volume 11 of Nonconvex Optimization and Its Application. Kluwer Academic Publishers, August 1996.
20. L. Kaufman, P.J. Rousseeuw. Finding Groups in Data: An Introdution to Cluster Analysis. – New York: John Wiley and Sons, November 1990. – 365p
21. A. K. Jain, R. C. Dubes. Algorithms for Clustering Data. – New Jersey: Prentice Hall, 1988. – 334p.

22. Fern, X.Z., Brodley, C.E. Clustering ensembles for high dimensional data clustering // In Proc. International Conference on Machine Learning, 2003. P.186-193.
23. Олдендерфер М. С., Блэшфилд Р. К. Кластерный анализ / Факторный, дискриминантный и кластерный анализ: пер. с англ.; Под. ред. И. С. Енюкова. – М.: Финансы и статистика, 1989. – 215 с.
24. Scholkopf B., Smola A., Muller K. Kernel Principal Component Analysis, Advances in Kernel Methods-Support Vector Learning, 1999.
25. Sanford A.D., Moosa I.A. A Bayesian network structure for operational risk modelling in structured finance operations //Department of Accounting and Finance Faculty of Business and Economics, Monash University, Australia, 2005. – 15 p.
26. Yang Y. A comparative study on feature selection in text categorization / Y.Yang, J.O.Pedersen – Waltham: Morgan Kaufmann Publishers, 1997. – 412-420 p.
27. A. Y. Doroshenko, I. Z. Achour (2018). Algorithm for automatic loop parallelization for graphics processing units. *Problems in programming*, (1), 36-45.
28. C. Cao, H. Wu, Y. Weng, T. Shao, K. Zhou (2016). Real-time facial animation with image-based dynamic avatars. *ACM Transactions on Graphics*, *35*(4).
29. MISHRA, Taniya; REICHENBACH, George Alexander; EL KALIOUBY, Rana. Avatar image animation using translation vectors. U.S. Patent No 10,628,985, 2020.
30. A. Neumann, F. Neumann, T. Friedrich (2017). Quasi-random Agents for Image Transition and Animation. *arXiv preprint arXiv:1710.07421*.
31. Lin, C. Y., Huang, Y. W., & Shih, T. K. (2019). Creating waterfall animation on a single image. Multimedia Tools and Applications, 78(6), 6637-6653.
32. S. Ye, R. Ohtera. Japanese Animation Style Transfer Using Deep Neural Networks. *In 2017 International Conference on Information, Communication and Engineering (ICICE)*. 2017. №17. P. 492-495.

33. Потапов А.А. Новейшие методы обработки изображений / А.А.Потапов, А.А.Пахомов, С.А.Никитин. - М.: Физматлит, 2008. - 496 с
34. Яне Б. Цифровая обработка зображений / Б. Яне. – Москва: Техносфера, 2007. - 584с.
35. Гашников М.В., Глумов Н.И., Ильясова Н.Ю., Мясников В.В., Попов СБ., Сергеев В.В., Сойфер В.А.и др. Методы компьютерной обработки изображений / Под ред. В.А. Сойфера. – 2-е изд., испр. – М.: ФИЗМАТЛИТ, 2003. – 784 с.
36. Prewitt, Judith MS. "Object enhancement and extraction." Picture processing and Psychopictorics 10.1 (1970): 15-19
37. I. Sobel and G. Feldman, “A 3x3 Isotropic Gradient Operator for Image Processing,” Pattern Classification and Scene Analysis, 1973, pp. 271-272.
38. Kirsch R. Computer determination of the constituent structure of biological images, Computers and Biomedical Research, 4, 1971 — P. 315–328.
39. Moon, P. and Spencer, D. E. "The Meaning of the Vector Laplacian.", J. Franklin Inst 1953 — pp. 551-558.
40. Kapustiy, B., B. P. Rusyn, and V. A. Tajanov. "“A new Approach to Determination of Correct Recognition Probability of the Set Objects." Upravlyayushchie Sistemy i Mashiny 2 (2005): 8-12.
41. Furgala Y.,Mochulsky Y.,Rusyn B. “Evoluation of objects recognition efficiency on mapes by verious methods/Proceedings of the 2018 IEEE 2nd International Conference on Data Stream Mining and Processing DSMP 2018,pp.595-598.
42. Kosar, Oleh, and Nataliya Shakhovska. "An Overview of Denoising Methods for Different Types of Noises Present on Graphic Images." Conference on Computer Science and Information Technologies. Springer, Cham (2018) 38-47
43. J.Loy, “Neural Network Projects with Python: The ultimate guide to using Python to explore the true power of neural networks through six projects”, Packt Publishing, p. 308, 2019.

44. Brannon W. C., “Object Detection in Low-spatial-resolution Aerial Imagery Using Convolutional Neural Networks”, p. 79, 2019.
45. Dataset: веб-сайт. URL: https://knowyourdata-tfds.withgoogle.com.
46. P. Arabie, L. J. Hubert, G. De Soete, “Clustering and Classification”, p. 500, 1996.
47. D. Parks, “Object Detection and Analysis: A Coherency Filtering Approach”, p. 172, 2008.
48. Yongqiang Z., Chen Y., Seong G. K., Quan P., Yongmei C., “Multi-band Polarization Imaging and Applications” 1st ed., p. 204, 2016.
49. Manikandan S., “Vision Based Assistive System for Label and Object Detection”, p. 64, 2015.
50. Salma H., “Object Detection Using Histogram Of Gradients”, p. 52, 2018.
51. Wu J., “Advances in K-means Clustering: A Data Mining Thinking”, Springer Science & Business Media, p. 180, 2021.
52. Каган В.Ф. Основы теории поверхностей в тензорном изложении. Москва : Рипол-классик,. 2013. 518 с.
53. OpenGL Transformation: веб-сайт. URL: http://www.songho.ca/opengl/gl_transform.html.
54. Д. Роджерс, Дж. Адамс. Математические основы машинной графики. Москва : Мир. 2001. 604 с.
55. Abramowitz, M.; Stegun, C. A. (1972), "§23.1: Bernoulli and Euler Polynomials and the Euler-Maclaurin Formula", Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables (9th printing ed.), New York: Dover, pp. 804–806.
56. Дронов В. Macromedia Flash Professional 8. Графика и анимация. Москва : БХВ-Петербург. 2013. 750 с.
57. Ильин В. А., Садовничий В. А., Сендов Б. Х. Математический анализ, ч. 1, изд. 3, ред. А. Н. Тихонов. М.: Проспект, 2004
58. Глухова Н.В. Інформаційна технологія для аналізу кольорових зображень газорозрядного випромінювання / Н.В. Глухова, Л.А. Пісоцька. Перспективні технології та прилади. № 12. 2018. С. 48–52.

59. Баловсяк С.В. Анализ рентгеновских муаровых изображений деформированных кристаллов по радиальным распределениям энергетического спектра Фурье / С.В. Баловсяк, С.Н. Новиков, И.М. Фодчук, И.В. Яремчук. Металофізика, новітні технології. 2019. Том 41. № 3. С. 389–402.
60. Povstyanyi V.A. Peculiarities of gas-discharge luminescence of biological fluid from the achilles tendon in the late postmortem period / V.A. Povstyanyi, L.A. Pesotskaya, N.V. Glukhova, N.M. Yevdokimenko, L.R. Nikogosyan, Ye.L. Koshelnik. Journal of Education, Health and Sport. Licensee Open Journal Systems of Kazimierz Wielki University in Bydgoszcz, Poland. 2017. No. 7(2), Pp. 498–508.
61. Роджерс Д. Алгоритмические основы машинной графики. — М.: Мир, 1989. — С. 54-63. — ISBN 5-03-000476-9.
62. Obed Appiah. Fast Generation of Image's Histogram Using Approximation Technique for Image Processing Algorithms / Obed Appiah, J. Ben Hayfron-Acquah. Image, Graphics and Signal Processing. 2018. No. 3. Pp. 25–35.
63. Navon E. et al. Color image segmentation based on adaptive local thresholds // Image and Vision Computing. – 2012. – № 23. – P. 69–85.
64. Niblack W. An introduction to digital image processing. – Prentice Hall, Englewood Cliffs.–1986. – 231 p
65. LeCun Y. et al. Learning algorithms for classification: A comparison on handwritten digit recognition.//Neural networks: the statistical mechanics perspective. – 1995. –T. 261. –P. 276.
66. P. Viola and M.J. Jones, «Rapid Object Detection using a Boosted Cascade of Simple Features», proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001), 2001
67. Linux Advanced Tutorials: веб-сайт. URL: https://www.linux.org/forums/linux-advanced-tutorials.125/.
68. AWS Lambda: веб-сайт. URL: https://aws.amazon.com/lambda/?nc1=h_ls.
69. AWS Lambda Features: веб-сайт. URL: https://aws.amazon.com/lambda/features/?pg=ln&sec=hs.

70. Amazon S3: веб-сайт. URL: https://aws.amazon.com/s3/?nc1=h_ls
71. Documentation → PostgreSQL 14: веб-сайт. URL: https://www.postgresql.org/docs/current/intro-whatis.html
72. MongoDB – Start with Guides: веб-сайт. URL: https://www.mongodb.com/docs/guides/atlas/cluster/
73. The Swift Programming Language. 2022. «Apple Inc. Cupertino, CA 95014 408-996-1010».
74. PHP Documentation: веб-сайт. URL: https://www.php.net/docs.php
75. Symfony Documentation: веб-сайт. URL: https://symfony.com/doc/current/index.html
76. The Python Tutorial: веб-сайт. URL: https://docs.python.org/3/tutorial/index.html
77. Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E. Imagenet classification with deep convolutional neural networks, NIPS. 2012.
78. LeCun Y., Y. Bengio. Convolutional Networks for Images, Speech, and Time-Series, in Arbib, M. A. (Eds), The Handbook of Brain Theory and Neural Networks, 1995.
79. Szegedy C. Deep neural networks for object detection / C. Szegedy, A. Toshev, D. Erhan. // NIPS. – 2013.
80. User Interface (UI) Prototypes: An Agile Introduction: веб-сайт. URL: http://agilemodeling.com/artifacts/uiPrototype.htm
81. Benchaib Y. Specialized Learning for Neural Classification of Cardiac Arrhythmias / Y. Benchaib, M.A. Chikh // Journal of Theoretical and Applied Information Technology. – 2009. – Vol. 6, No 1. – Pp. 92-100.
82. Kadhim Al-Shayea Q. Neural Networks in Bank Insolvency Prediction / Q. Kadhim Al-Shayea, G.A. El-Refae, S.F. El-Itter // International Journal of Computer Science and Network Security. – 2010. – Vol. 10, No 5. – Pp. 240-245.
83. Narendra K.S. Identification and control of dynamic systems using neural networks / K.S. Narendra, K. Pathasarthy // IEEE Trans. Neural Networks. – 1990. – Vol. 1, No 1. – Pp. 1-27.

84. Seetha M. Artificial Neural Network and Other Methods of Image Classification / M. Seetha, I.V. Muralikrishna, B.L. Deekshatulu // Journal of Theoretical and Applied Information Technology. – 2008. – Vol. 4, No 11. – Pp.1039-1053.
85. Lippman R.P. An Introduction to Computing with Neural Nets / R.P. Lippman // IEEE ASSP Magazine. – 1987. – Vol. 3. No 4. – Pp. 4-22.
86. Holynski A., Curless B., M Seitz S., Szeliski R. 2020. Animating Pictures with Eulerian Motion Fields. arXiv preprint arXiv: 2011.15128 – 2020.
87. T. Le, Chih-Kuo Yeh, Ying-Chi Lin, Tong-Yee Lee. Animating still natural images using warping. Computer Science ACM Transactions on Multimedia Computing, Communications, and Applications. – 2022.
88. RESTful API: веб-сайт. URL: https://aws.amazon.com/what-is/restful-api/?nc1=h_ls
89. Weiler K. An Incremental Angle Point in Polygon Test, in Heckbert, Paul S. (ed.), Graphics Gems IV, San Diego, CA, USA: Academic Press Professional, Inc. – 1994.
90. Shimrat, M., Algorithm 112: Position of point relative to polygon / M. Shimrat // Communications of the ACM – 1962. – Vol. 5. Issue 8.
91. Hormann K., Agathos A. The Point in Polygon Problem for Arbitrary Polygons. / K. Hormann, A. Agathos // Computational Geometry. – 2001. Vol. 20. No 3. – pp. 131-144.

ДОДАТОК А. ПРОГРАМНИЙ КОД РЕАЛІЗАЦІЇ ПОВНОЗВ'ЯЗНОЇ НЕЙРОННОЇ МЕРЕЖІ

Клас реалізації шарів ПНМ:

```
import numpy as np

class Dense:
    def __init__(self, in_size, out_size, reg_lambda=0.0):
        self.W = np.random.normal(
            scale=1, size=(out_size, in_size)) * np.sqrt(2 / (in_size + out_size))
        self.b = np.zeros(out_size)
        self.reg_lambda = reg_lambda
        self.final_dW = 0
        self.final_db = 0

    def forward(self, x):
        self.x = x
        if np.array(x).shape[0] != self.W.shape[1]:
            print('X is not the same dimention as in_size')
        return (np.dot(self.W, self.x) + self.b)

    def get_reg_loss(self):
        return 0.5 * self.reg_lambda * (np.linalg.norm(self.W, ord='fro')**2)

    def backward(self,
                 dz,
                 learning_rate=0.001,
                 mini_batch=False,
                 update=True,
                 len_mini_batch=None):
        self.dW = np.outer(dz, self.x)
        self.db = dz
        self.dx = np.dot(dz, self.W)

        if (self.reg_lambda != 0):
            self.dW += self.reg_lambda * self.W

        if mini_batch == True:
            self.final_dW += self.dW
            self.final_db += self.db

        if update == True:
            if mini_batch == True:
                self.W = self.W - learning_rate * self.final_dW / len_mini_batch
                self.b = self.b - learning_rate * self.final_db / len_mini_batch
                self.final_dW = 0
                self.final_db = 0
            else:
                self.W = self.W - learning_rate * self.dW
                self.b = self.b - learning_rate * self.db

        return self.dx


class Dropout:
    def __init__(self, p=0.5):
        self.p = p

    def forward(self, x, train=True):
        if not train:
            self.mask = np.ones(*x.shape)
```

```
            return x
        self.mask = (np.random.rand(*x.shape) > self.p) / (1.0 - self.p)
        return x * self.mask

    def backward(self, dz, lr=0.001):
        return dz * self.mask
```

Клас реалізації функцій активації:

```
import numpy as np

class Softmax:
    def forward(self, x):
        x = x - np.max(x)
        self.p = np.exp(x) / np.sum(np.exp(x))
        return self.p

    def backward(self, dz):
        jacobian = np.diag(dz)
        for i in range(len(jacobian)):
            for j in range(len(jacobian)):
                if i == j:
                    jacobian[i][j] = self.p[i] * (1 - self.p[j])
                else:
                    jacobian[i][j] = -self.p[i] * self.p[j]
        return np.matmul(dz, jacobian)


class ReLu:
    def forward(self, x):
        self.x = x
        return np.maximum(0, x)

    def backward(self, dz):
        dz[self.x < 0] = 0
        return dz
```

Клас реалізації розрахунку ентропії:

```
import numpy as np

class CrossEntropy:
    def forward(self, y_true, y_hat):
        self.y_hat = y_hat
        self.y_true = y_true
        self.loss = -np.sum(self.y_true * np.log(y_hat))
        return self.loss

    def backward(self):
        dz = -self.y_true / self.y_hat
        return dz
```

Клас реалізації ПНМ:

```
from layers import Dense, Dropout
from loss import CrossEntropy
from activations import ReLu, Softmax
import numpy as np

class MnistNetMiniBatch:
```

```
    def __init__(self):
        self.d1_layer = Dense(784, 100)
        self.a1_layer = ReLu()
        self.drop1_layer = Dropout(0.5)

        self.d2_layer = Dense(100, 50)
        self.a2_layer = ReLu()
        self.drop2_layer = Dropout(0.25)

        self.d3_layer = Dense(50, 10)
        self.a3_layer = Softmax()

    def forward(self, x, train=True):
        net = self.d1_layer.forward(x)
        net = self.a1_layer.forward(net)
        net = self.drop1_layer.forward(net, train)

        net = self.d2_layer.forward(net)
        net = self.a2_layer.forward(net)
        net = self.drop2_layer.forward(net, train)

        net = self.d3_layer.forward(net)
        net = self.a3_layer.forward(net)

        return (net)

    def backward(self,
                 dz,
                 learning_rate=0.01,
                 mini_batch=True,
                 update=False,
                 len_mini_batch=None):

        dz = self.a3_layer.backward(dz)
        dz = self.d3_layer.backward(
            dz,
            learning_rate=learning_rate,
            mini_batch=mini_batch,
            update=update,
            len_mini_batch=len_mini_batch)

        dz = self.drop2_layer.backward(dz)
        dz = self.a2_layer.backward(dz)
        dz = self.d2_layer.backward(
            dz,
            learning_rate=learning_rate,
            mini_batch=mini_batch,
            update=update,
            len_mini_batch=len_mini_batch)

        dz = self.drop1_layer.backward(dz)
        dz = self.a1_layer.backward(dz)
        dz = self.d1_layer.backward(
            dz,
            learning_rate=learning_rate,
            mini_batch=mini_batch,
            update=update,
            len_mini_batch=len_mini_batch)

        return dz


def compute_acc(X_test, Y_test, net):
```

```
    '''Not one-hot encoded format'''
    acc = 0.0
    for i in range(X_test.shape[0]):
        y_h = net.forward(X_test[i])
        y = np.argmax(y_h)
        if (y == Y_test[i]):
            acc += 1.0
    return acc / Y_test.shape[0]


if __name__ == 'main':
    loss = CrossEntropy()
    net = MnistNetMiniBatch()
    learning_rate = 0.001
    L_train = []
    L_test = []
    Acc_train = []
    Acc_test = []
    len_mini_batch = 10
    for it in range(100):
        L_acc = 0.
        sh = list(range(train_x.shape[0]))
        np.random.shuffle(sh)
        for i in range(train_x.shape[0]):
            x = train_x[sh[i]]
            y = train_y_oh[sh[i]]
            y_h = net.forward(x)
            L = loss.forward(y, y_h)
            L_acc += L
            dz = loss.backward()
            if i % len_mini_batch == 0:
                dz = net.backward(
                    dz,
                    learning_rate,
                    update=True,
                    len_mini_batch=len_mini_batch)
            else:
                dz = net.backward(dz, learning_rate)
        L_acc /= train_y_oh.shape[0]
        L_train.append(L_acc)
        acc = compute_acc(train_x, train_y, net)
        Acc_train.append(acc)
        L_e_acc = 0.
        for i in range(test_x.shape[0]):
            x = test_x[i]
            y = test_y_oh[i]
            y_h = net.forward(x)
            L = loss.forward(y, y_h)
            L_e_acc += L
        L_e_acc /= test_y_oh.shape[0]
        L_test.append(L_e_acc)
        acc = compute_acc(test_x, test_y, net)
        Acc_test.append(acc)

        learning_rate = learning_rate * 0.99

        print(
            "{} epoch. Train : {} . Test : {}. acc : {} . val_acc: {}".format(
                it + 1, L_acc, L_e_acc, Acc_train[-1], Acc_test[-1]))
```

ДОДАТОК Б. ПРОГРАМНИЙ КОД РЕАЛІЗАЦІЇ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

```
import numpy as np

def get_axis_indexes(kernel_axis_length, center_index):
    axis_indexes = []
    for i in range(-center_index, kernel_axis_length - center_index):
        axis_indexes.append(i)
    return axis_indexes

def get_axes_indexes(kernel_size, kernel_center):
    indexes_a = get_axis_indexes(
        kernel_axis_length=kernel_size[0],
        center_index=kernel_center[0]
    )
    indexes_b = get_axis_indexes(
        kernel_axis_length=kernel_size[1],
        center_index=kernel_center[1]
    )
    return indexes_a, indexes_b

def load_weight_from_npy(weight_name, load_path):
    weight = np.load(load_path, allow_pickle=True).item().get(weight_name)
    return weight

class Conv2d:
    def __init__(
        self, in_channels, out_channels, kernel_size, stride=1,
        kernel_center=(0, 0), padding=0, convolution=False
    ):
        self.stride = stride
        self.in_channels = in_channels
        self.out_channels = out_channels
        if isinstance(kernel_size, tuple):
            self.kernel_size = kernel_size
        else:
            self.kernel_size = (kernel_size, kernel_size)
        self.kernel_center = kernel_center
        self.convolution = convolution
        self.padding = padding

        self.init_weights()

    def load_weights(self, weight_name, bias_name, load_path):
        self.conv_w = load_weight_from_npy(weight_name, load_path)
        self.conv_b = load_weight_from_npy(bias_name, load_path)

    def init_weights(self):
        self.conv_w = self.init_weight(self.kernel_size,
                                       self.in_channels*self.out_channels)
        self.conv_b = self.init_weight((1, 1), self.out_channels)

    def init_weight(self, weight_shape, weight_count):
        weight = []
        for i in range(weight_count):
            weight.append(2*np.random.random(weight_shape)-1)
        return weight

    def convolution_feed_x_l(self, y_l_minus_1, w_l, print_demo=False):
        stride = self.stride
        indexes_a, indexes_b = get_axes_indexes(w_l.shape, self.kernel_center)
```

```
        y_l_minus_1 = np.pad(y_l_minus_1, self.padding)
        h_out = int(
            (y_l_minus_1.shape[0] - (self.kernel_size[0]-1) - 1) / stride + 1
        )
        w_out = int(
            (y_l_minus_1.shape[1] - (self.kernel_size[1]-1) - 1) / stride + 1
        )
        x_l = np.zeros((h_out, w_out), dtype=np.float32)
        if self.convolution:
            g = 1  # convolution
        else:
            g = -1  # cross-correlation
        for i in range(h_out):
            for j in range(w_out):
                demo = np.zeros([y_l_minus_1.shape[0], y_l_minus_1.shape[1]],
dtype=np.float32)
                result = 0
                element_exists = False
                for a in indexes_a:
                    for b in indexes_b:
                        # check that indexes of a, b did not crossed the
                        # boundaries of the y_l_minus_1
                        if (
                            i*stride - g*a >= 0
                            and j*stride - g*b >= 0
                            and i*stride - g*a < y_l_minus_1.shape[0]
                            and j*stride - g*b < y_l_minus_1.shape[1]
                        ):
                            # convert indexes of a, b to a range of positive
                            # numbers to extract data from w_l
                            result += \
                                y_l_minus_1[i*stride - g*a][j*stride - g*b] * \
                                w_l[indexes_a.index(a)][indexes_b.index(b)]
                            demo[i*stride - g*a][j*stride - g*b] = \
                                w_l[indexes_a.index(a)][indexes_b.index(b)]
                            element_exists = True
                if element_exists:
                    x_l[i][j] = result
                    # print demo matrix for tracking the convolution progress
                    if print_demo:
                        print('i=' + str(i) + '; j=' + str(j) + '\n', demo)
        return x_l

    def convolution_back_dEdw_l(self, y_l_minus_1, dEdx_l, print_demo=False):
        stride = self.stride
        y_l_minus_1 = np.pad(y_l_minus_1, self.padding)
        w_l_shape = self.conv_w[0].shape
        indexes_a, indexes_b = get_axes_indexes(w_l_shape, self.kernel_center)
        dEdw_l = np.zeros((w_l_shape[0], w_l_shape[1]), dtype=np.float32)
        if self.convolution:
            g = 1  # convolution
        else:
            g = -1  # cross-correlation
        for a in indexes_a:
            for b in indexes_b:
                demo = np.zeros([y_l_minus_1.shape[0], y_l_minus_1.shape[1]],
dtype=np.float32)
                result = 0
                for i in range(dEdx_l.shape[0]):
                    for j in range(dEdx_l.shape[1]):
                        # check that indexes of a, b did not crossed the
                        # boundaries of the y_l_minus_1
                        if (
```

```
                            i*stride - g*a >= 0
                            and j*stride - g*b >= 0
                            and i*stride - g*a < y_l_minus_1.shape[0]
                            and j*stride - g*b < y_l_minus_1.shape[1]
                        ):
                            result += \
                                y_l_minus_1[i*stride - g*a][j*stride - g*b] * \
                                dEdx_l[i][j]
                            demo[i*stride - g*a][j*stride - g*b] = \
                                dEdx_l[i][j]
                # convert indexes of a, b to a range of positive
                # numbers to extract data from w_l
                dEdw_l[indexes_a.index(a)][indexes_b.index(b)] = result
                # print demo matrix for tracking the convolution progress
                if print_demo:
                    print('a=' + str(a) + '; b=' + str(b) + '\n', demo)
        return dEdw_l

    def convolution_back_dEdy_l_minus_1(
        self, dEdx_l, w_l, y_l_minus_1_shape, print_demo=False
    ):
        indexes_a, indexes_b = get_axes_indexes(w_l.shape, self.kernel_center)
        dEdy_l_minus_1 = np.zeros((y_l_minus_1_shape[0] + 2*self.padding,
                                   y_l_minus_1_shape[1] + 2*self.padding),
dtype=np.float32)
        if self.convolution:
            g = 1  # convolution
        else:
            g = -1  # cross-correlation
        for i in range(dEdy_l_minus_1.shape[0]):
            for j in range(dEdy_l_minus_1.shape[1]):
                result = 0
                demo = np.zeros([dEdx_l.shape[0], dEdx_l.shape[1]], dtype=np.float32)
                for i_x_l in range(dEdx_l.shape[0]):
                    for j_x_l in range(dEdx_l.shape[1]):
                        a = g*i_x_l*self.stride - g*i
                        b = g*j_x_l*self.stride - g*j
                        if (
                            a in indexes_a
                            and b in indexes_b
                        ):
                            a = indexes_a.index(a)
                            b = indexes_b.index(b)
                            result += dEdx_l[i_x_l][j_x_l] * w_l[a][b]
                            demo[i_x_l][j_x_l] = w_l[a][b]
                dEdy_l_minus_1[i][j] = result
                    print('i=' + str(i) + '; j=' + str(j) + '\n', demo)
        # cut initial y_l_minus_1 shape with no padding from dEdy_l_minus_1
        dEdy_l_minus_1 = \
            dEdy_l_minus_1[self.padding:(y_l_minus_1_shape[0]+self.padding),
                           self.padding:(y_l_minus_1_shape[1]+self.padding)]
        return dEdy_l_minus_1

    def __call__(self, y_l_minus_1):
        x_l = []
        for i in range(self.in_channels):
            for j in range(i*self.out_channels, (i + 1)*self.out_channels):
                # for each y_l_minus_1, the convolution is called
                # out_channels-times to create "intermediate" x_l
                x_l.append(
                    self.convolution_feed_x_l(y_l_minus_1[i], self.conv_w[j]))
        x_l_final = []
        for i in range(self.out_channels):
```

```
            x_l_final.append(0)
            for j in range(self.in_channels):
                # the "final" x_l is the sum of the "intermediate" x_l
                # received from each y_l_minus_1
                x_l_final[-1] += x_l[j*self.out_channels + i]
            # add bias to the x_l
            x_l_final[-1] += self.conv_b[len(x_l_final)-1]
        self.y_l_minus_1 = y_l_minus_1  # need for backprop
        return x_l_final

    def backprop(self, dEdx_l, learning_rate):
        list_of_dEdy_l_minus_1 = []
        for i in range(self.out_channels):
            # due to the fact that one bias refers to the whole feature map,
            # dE/db_l is the sum of all the elements of dE/dx_l
            dEdb_l = dEdx_l[i].sum()
            self.conv_b[i] = self.conv_b[i] - learning_rate * dEdb_l
        for i in range(self.in_channels):
            dEdy_l_minus_1 = 0
            k = 0
            for j in range(i*self.out_channels, (i + 1)*self.out_channels):
                dEdw_l = self.convolution_back_dEdw_l(
                    y_l_minus_1=self.y_l_minus_1[i],
                    dEdx_l=dEdx_l[k],
                )
                # the backprop for dE/dy_l_minus_1 accumulates data from all
                # the corresponding feature maps
                dEdy_l_minus_1 += self.convolution_back_dEdy_l_minus_1(
                    dEdx_l=dEdx_l[k],
                    w_l=self.conv_w[j],
                    y_l_minus_1_shape=self.y_l_minus_1[i].shape,
                )
                self.conv_w[j] = self.conv_w[j] - learning_rate * dEdw_l
                k += 1
            list_of_dEdy_l_minus_1.append(dEdy_l_minus_1)
        return list_of_dEdy_l_minus_1

class Sigmoid:
    def __call__(self, x_l):
        x_l = np.array(x_l, dtype=np.float32)
        y_l = 1 / (1+np.exp(-x_l))
        self.y_l = y_l  # need for backprop
        return y_l

    def backprop(self, dEdy_l):
        dy_ldx_l = self.y_l * (1 - self.y_l)
        dEdx_l = dEdy_l * dy_ldx_l
        return dEdx_l

class Softmax:
    def __call__(self, x_l):
        x_l = np.array(x_l, dtype=np.float32)
        y_l = np.exp(x_l) / np.exp(x_l).sum()
        self.y_l = y_l  # need for backprop
        return y_l

    def backprop(self, dEdy_l):
        dy_ldx_l = np.zeros((self.y_l.shape[1], self.y_l.shape[1]), dtype=np.float32)
        for i in range(dy_ldx_l.shape[1]):
            for j in range(dy_ldx_l.shape[1]):
                if i == j:
                    dy_ldx_l[i][i] = self.y_l[0][i]*(1 - self.y_l[0][i])
                else:
```

```
                    dy_ldx_l[i][j] = - self.y_l[0][i]*self.y_l[0][j]
        dEdx_l = np.dot(dEdy_l, dy_ldx_l)
        return dEdx_l

class ReLU:
    def __call__(self, x_l):
        x_l = np.array(x_l, dtype=np.float32)
        # zero out the elements that do not pass the condition
        y_l = np.where(x_l > 0, x_l, 0)
        self.y_l = y_l  # need for backprop
        return y_l

    def backprop(self, dEdy_l):
        dy_ldx_l = np.where(self.y_l <= 0, self.y_l, 1)
        dEdx_l = dEdy_l * dy_ldx_l
        return dEdx_l

class Maxpool2d:
    def __init__(
        self, kernel_size, stride=1, kernel_center=(0, 0), padding=0,
        convolution=False
    ):
        if isinstance(kernel_size, tuple):
            self.kernel_size = kernel_size
        else:
            self.kernel_size = (kernel_size, kernel_size)
        self.kernel_center = kernel_center
        self.stride = stride
        self.convolution = convolution
        if convolution and padding > 0:
            raise AssertionError("Padding in maxpooling can not be in "
                                 "convolution, due to the inverted kernel")
        assert padding <= int(min(self.kernel_size) / 2), \
            "Pad should be smaller than or equal to half of kernel size."
        self.padding = padding

    def maxpool(self, y_l):
        # padd by -inf to be sure these values won't be selected by maxpooling
        y_l = np.pad(y_l, self.padding, constant_values=-np.inf)
        indexes_a, indexes_b = get_axes_indexes(self.kernel_size,
                                                self.kernel_center)
        stride = self.stride
        h_out = int(
            (y_l.shape[0] - (self.kernel_size[0]-1) - 1) / stride + 1
        )
        w_out = int(
            (y_l.shape[1] - (self.kernel_size[1]-1) - 1) / stride + 1
        )
        y_l_mp = np.zeros((h_out, w_out), dtype=np.float32)
        y_l_mp_to_y_l = np.zeros((h_out, w_out), dtype='<U32')
        if self.convolution:
            g = 1  # convolution
        else:
            g = -1  # cross-correlation
        for i in range(h_out):
            for j in range(w_out):
                result = -np.inf
                element_exists = False
                for a in indexes_a:
                    for b in indexes_b:
                        if (
                            i*stride - g*a >= 0
                            and j*stride - g*b >= 0
```

```
                            and i*stride - g*a < y_l.shape[0]
                            and j*stride - g*b < y_l.shape[1]
                        ):
                            if y_l[i*stride - g*a][j*stride - g*b] > result:
                                result = y_l[i*stride - g*a][j*stride - g*b]
                                # subtract self.padding from coords to make
                                # backprop correct for padded matrices
                                i_back = i*stride - g*a - self.padding
                                j_back = j*stride - g*b - self.padding
                                element_exists = True
                if element_exists:
                    y_l_mp[i][j] = result
                    y_l_mp_to_y_l[i][j] = str(i_back) + ',' + str(j_back)
        return y_l_mp, y_l_mp_to_y_l

    def __call__(self, y_l):
        """Feedforward of a maxpooling layer."""
        list_of_y_l_mp = []
        self.list_of_y_l_mp_to_y_l = []
        for i in range(len(y_l)):
            y_l_mp, y_l_mp_to_y_l = self.maxpool(y_l[i])
            list_of_y_l_mp.append(y_l_mp)
            # y_l_mp_to_y_l stores the text (coords of the selected elements
            # during feedforward) and needed for backprop: dE/dy_l_mp -> dE/dy_l
            self.list_of_y_l_mp_to_y_l.append(y_l_mp_to_y_l)
        # take an input shape to restore it on the backprop stage
        self.y_l_shape = y_l[0].shape
        return list_of_y_l_mp

    def backprop(self, dEdy_l_mp):
        """Backpropagation of a maxpooling layer."""
        list_of_dEdy_l = []
        for i in range(len(dEdy_l_mp)):
            # dEdy_l will expand as new elements are added
            dEdy_l = np.zeros(self.y_l_shape, dtype=np.float32)
            for k in range(dEdy_l_mp[i].shape[0]):
                for e in range(dEdy_l_mp[i].shape[1]):
                    # each element of the dEdy_l_mp must be placed in the
                    # dEdy_l; to do this, we extract the necessary destination
                    # coordinates from the list_of_y_l_mp_to_y_l
                    coordinates = self.list_of_y_l_mp_to_y_l[i][k][e]
                    coordinate_row = int(coordinates[:coordinates.find(',')])
                    coordinate_col = int(coordinates[coordinates.find(',')+1:])
                    # addition - in case one element was selected as the
                    # maximum several times
                    dEdy_l[coordinate_row][coordinate_col] += dEdy_l_mp[i][k][e]
            list_of_dEdy_l.append(dEdy_l)
        return list_of_dEdy_l


class Linear:
    """Applies a linear transformation to the incoming data.
    Fully connected layer.
    Args:
        in_features (int): Size of each input sample.
        out_features (int): Size of each output sample.
    """

    def __init__(self, in_features, out_features):
        self.in_features = in_features
        self.out_features = out_features
        self.init_weights()
```

```
    def load_weights(self, weight_name, bias_name, load_path):
        self.fc_w = load_weight_from_npy(weight_name, load_path)
        self.fc_b = load_weight_from_npy(bias_name, load_path)

    def init_weights(self):
        self.fc_w = self.init_weight((self.in_features, self.out_features))
        self.fc_b = self.init_weight((1, self.out_features))

    def init_weight(self, weight_shape):
        weight = 2 * np.random.random(weight_shape) - 1
        return weight

    def __call__(self, y_l_minus_1):
        """Feedforward of a linear layer."""
        x_l = np.dot(y_l_minus_1, self.fc_w) + self.fc_b
        self.y_l_minus_1 = y_l_minus_1  # need for backprop
        return x_l

    def backprop(self, dEdx_l, learning_rate):
        """Backpropagation of a linear layer.
        Args:
            dEdx_l: de/dx_l matrix.
            learning_rate (float): The learning rate for training using SGD.
        """
        dEdw_l = np.dot(self.y_l_minus_1.T, dEdx_l)
        dEdb_l = dEdx_l
        dEdy_l_minus_1 = np.dot(dEdx_l, self.fc_w.T)
        self.fc_w = self.fc_w - learning_rate * dEdw_l
        self.fc_b = self.fc_b - learning_rate * dEdb_l
        return dEdy_l_minus_1

class CrossEntropyLoss:
    def __call__(self, target, predict):
        return -target * np.log(predict)

    def backprop(self, target, predict):
        return -(target/predict)

class MSELoss:
    def __call__(self, target, predict):
        return (target - predict)**2

    def backprop(self, target, predict):
        return predict - target

class Flatten:
    def matrices2vector(self, matrices):
        vector = np.array([[]], dtype=np.float32)
        self.matrix_shape = matrices[0].shape
        for i in range(len(matrices)):
            reshaped_matrix = np.reshape(
                matrices[i], (1, self.matrix_shape[0]*self.matrix_shape[1]))
            vector = np.hstack((vector, reshaped_matrix))
        return vector

    def vector2matrices(self, vector):
        matrices = []
        matrix_size = self.matrix_shape[0]*self.matrix_shape[1]
        for i in range(0, vector.size, matrix_size):
            matrix = np.reshape(vector[0][i:i+matrix_size], self.matrix_shape)
            matrices.append(matrix)
        return matrices
```

ДОДАТОК В. ПРОГРАМНИЙ КОД РЕАЛІЗАЦІЇ КОНТУРНОГО АНАЛІЗУ

Оператор Собеля:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

imagePath = '10.jpg'
img = cv2.imread(imagePath)

lenna_img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

GrayImage = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

Gaussian = cv2.GaussianBlur(GrayImage, (3, 3), 0, 0, cv2.BORDER_DEFAULT)

Median = cv2.medianBlur(Gaussian, 5)

 x = cv2.Sobel (Median, cv2.CV_8U, 1, 0, ksize = 3)
 y = cv2.Sobel (Median, cv2.CV_8U, 0, 1, ksize = 3)
 absX = cv2.convertScaleAbs (x)
absY = cv2.convertScaleAbs(y)
Sobel = cv2.addWeighted(absX, 0.5, absY, 0.5,0)
cv2.imshow('dilation2', Sobel)
cv2.waitKey(0)

ret, Binary = cv2.threshold(Sobel, 170, 255, cv2.THRESH_BINARY)
cv2.imshow('dilation2',Binary)
cv2.waitKey(0)

titles = ['Source Image','Gray Image', 'Gaussian Image', 'Median Image',
          'Sobel Image', 'Binary Image']
images = [lenna_img, GrayImage, Gaussian, Median, Sobel, Binary]
for i in xrange(6):
   plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
   plt.title(titles[i])
   plt.xticks([]),plt.yticks([])
plt.show()
```

Оператор Прюітта:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

imagePath = '10.jpg'
img = cv2.imread(imagePath)img_RGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

lenna_img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
GrayImage = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

Gaussian = cv2.GaussianBlur(GrayImage, (3, 3), 0, 0, cv2.BORDER_DEFAULT)
Median = cv2.medianBlur(Gaussian, 5)

kernelx = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]], dtype=int)
kernely = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]], dtype=int)
x = cv2.filter2D(grayImage, cv2.CV_16S, kernelx)
y = cv2.filter2D(grayImage, cv2.CV_16S, kernely)
 # Turn uint8
```

```
absX = cv2.convertScaleAbs(x)
absY = cv2.convertScaleAbs(y)
Prewitt = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)

 titles = ['Source Image','Gray Image', 'Gaussian Image', 'Median Image',
          'Sobel Image', 'Binary Image']
images = [lenna_img, GrayImage, Gaussian, Median, Prewitt, Binary]
for i in xrange(6):
   plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
   plt.title(titles[i])
   plt.xticks([]),plt.yticks([])
plt.show()
```

ДОДАТОК Г. СХОВИЩА ДАНИХ

```
create table classes
(
	id serial,
	unique_hash_name varchar(32) not null,
	created_at timestamp not null,
	last_element_added_at timestamp not null,
	constraint classes_pk
		primary key (id)
);

create unique index classes_unique_hash_name_uindex
	on classes (unique_hash_name);

create table objects
(
	id serial,
	unique_hash_name varchar(32) not null,
	geometric_representation geometry not null,
	object_representation blob not null,
	segmentation_representation blob not null,
	class_id integer not null,
	created_at timestamp not null,
	constraint objects_pk
		primary key (id),
	constraint objects_classes_id_fk
		foreign key (class_id) references classes
);

create unique index objects_unique_hash_name_uindex
	on objects (unique_hash_name);

create table objects_features
(
	id serial,
	hog json not null,
	corners json not null,
	normalized_u double precision not null,
	normalized_r double precision not null,
	normalized_p double precision not null,
	object_id integer not null,
	created_at timestamp not null,
	constraint objects_features_pk
		primary key (id),
	constraint objects_features_objects_id_fk
		foreign key (object_id) references objects
);

create unique index objects_features_object_id_uindex
	on objects_features (object_id);

create table images
(
	id serial,
	unique_hash_name varchar(32) not null,
	image_representation blob not null,
	created_at timestamp not null,
	constraint images_pk
		primary key (id)
);

create unique index images_unique_hash_name_uindex
```

```
	on images (unique_hash_name);

create table images_has_objects
(
	object_id integer not null,
	image_id integer not null,
	constraint images_has_objects_objects_id_fk
		foreign key (object_id) references objects,
	constraint images_has_objects_images_id_fk
		foreign key (image_id) references images
);
```

ДОДАТОК Д. ОПИС МОДУЛЯ API.

```
openapi: 3.0.3
info:
  title: animation_module_api
  description: animation_module_api
  version: 1.0.0
servers:
  - url: 'https://localhost:8080/api'
    description: "develop"
components:
  securitySchemes:
    ApiKeyAuth:
      type: apiKey
      name: X-Auth-Token
      in: header
  schemas:
    Image:
      type: object
      properties:
        file_name:
          type: string
        file_content:
          type: string
          format: hexadecimal
      required:
        - file_name
        - file_content
    DatabaseRowInformation:
      type: object
      properties:
        id:
          type: integer
        unique_hash_name:
          type: string
      required:
        - id
        - unique_hash_name
    PixelCoordinate:
      type: object
      properties:
        x:
          type: integer
        y:
          type: integer
      required:
        - x
        - y
    FoundObject:
      type: object
      properties:
        pixel_coordinates:
          type: array
          items:
            $ref: "#/components/schemas/PixelCoordinate"
      required:
        - pixel_coordinates
    SelectedObject:
      type: object
      properties:
        pixel_coordinates:
          type: array
          items:
```

```
            $ref: "#/components/schemas/PixelCoordinate"
        content:
          type: string
          format: hexadecimal
      required:
        - pixel_coordinates
        - content
    ImageSelectedObjects:
      type: object
      properties:
        image_info:
          type: object
          items:
            $ref: "#/components/schemas/DatabaseRowInformation"
        objects:
          type: array
          items:
            $ref: "#/components/schemas/SelectedObject"
      required:
        - image_info
        - objects
    ProcessInformation:
      type: object
      properties:
        id:
          type: string
          format: uuid
      required:
        - id
    Error:
      type: object
      properties:
        type:
          type: string
        title:
          type: string
        status:
          type: integer
        detail:
          type: string
      required:
        - type
        - title
        - status

paths:
  /image:
    post:
      summary: Save image to database.
      security:
        - ApiKeyAuth: [ ]
      requestBody:
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Image"
        required: true
      responses:
        201:
          description: Return image information from database
          content:
            application/json:
              schema:
```

```
              $ref: "#/components/schemas/DatabaseRowInformation"
      500:
        description: Internal server error.
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Error"
/selected-objects:
  post:
    summary: Save objects to database.
    security:
      - ApiKeyAuth: [ ]
    requestBody:
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/ImageSelectedObjects"
      required: true
    responses:
      201:
        description: Return objects information from database
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: "#/components/schemas/DatabaseRowInformation"
      500:
        description: Internal server error.
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Error"
/objects-searching:
  post:
    summary: Starts process of searching for objects.
    security:
      - ApiKeyAuth: [ ]
    requestBody:
      content:
        application/json:
          schema:
            $ref: "#/components/schemas/DatabaseRowInformation"
      required: true
    responses:
      202:
        description: Return information of process id.
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: "#/components/schemas/ProcessInformation"
      500:
        description: Internal server error.
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/Error"
  get:
    summary: Checks whether the object search process has been completed.
    security:
      - ApiKeyAuth: [ ]
```

```
parameters:
  - name: process_id
    in: path
    description: The process ID to check.
    required: true
    schema:
      type: string
responses:
  102:
    description: Process in progress.
  200:
    description: Returns the pixel coordinates of the found objects in the image.
    content:
      application/json:
        schema:
          type: array
          items:
            $ref: "#/components/schemas/FoundObject"
  404:
    description: Process with id not found.
    content:
      application/json:
        schema:
          $ref: "#/components/schemas/Error"
  500:
    description: Internal server error.
    content:
      application/json:
        schema:
          $ref: "#/components/schemas/Error"
```

ДОДАТОК Е. АКТИ ВПРОВАДЖЕННЯ

"ЗАТВЕРДЖУЮ"
Проректор з наукової роботи
Національного університету
«Львівська політехніка»
_____ І.В.Демидов
_____ 2022 р.

АКТ
використання наукових результатів
дисертаційної роботи Шамуратова Олексія Юрійовича,
представленої на здобуття наукового ступеня доктора філософії

Комісія в складі: голови комісії - начальника науково-дослідної частини д.т.н., с.н.с Небесного Р.В. та членів комісії - завідувача кафедри СШІ Шаховської Н.Б., професора кафедри СШІ Яковини В.С., доцента кафедри СШІ Хавалка В.М., доцента кафедри СШІ Кривенчука Ю.П. цим актом підтверджують, що результати дисертаційної роботи Шамуратов О.Ю., зокрема

- метод контурного аналізу зображень та пошуку об'єкта на зображенні;
- модель класифікації та кластеризації об'єктів;
- метод анімування статичних об'єктів на зображеннях

використано у науково-дослідних роботах фінансованих Міністерством освіти і науки України, що виконувались на кафедрі систем штучного інтелекту і включено до звіту:
«Інформаційна технологія формування психофізичного портрету в умовах стресових ситуацій» (№ держ. реєстру 0119U002257).
Отримані автором результати використано:

- при розробленні систем анімування зображень;
- при розробленні засобів пошуку об'єктів на зображенні;
- при розробленні засобів класифікації об'єктів на зображенні.

Голова комісії:

начальник
науково-дослідної частини
д.т.н., с.н.с Р.В.Небесний

Члени комісії:

завідувача кафедри СШІ Н.Б.Шаховська
доцент кафедри СШІ М.І.Мельникова

доцент кафедри СШІ В.М.Хавалко

доцент каефдри СШІ Ю.П.Кривенчук

"ЗАТВЕРДЖУЮ"
Проректор з науково-педагогічної роботи
Національного університету
"Львівська політехніка"
_____ **О.Р.Давидчак**
"___" _____ 2022 р.

А К Т

про впровадження в навчальний процес результатів
дисертаційної роботи

Шамуратова Олексія Юрійовича

Цей акт складено про те, що результати дисертаційної роботи Шамуратова Олексія Юрійовича впроваджено у навчальний процес кафедри "Системи штучного інтелекту " Національного університету "Львівська політехніка".

Впровадження результатів дисертаційної роботи полягає в їхньому використанні при викладанні навчальних дисциплін як окремих розділів лекційних курсів, так і в циклах лабораторних робіт.

Зокрема для викладання дисципліни «Системний аналіз» для студентів освітньо-кваліфікаційного рівня «бакалавр», що навчаються за напрямом 122 "Комп'ютерні науки" використано такі результати:

- загальні принципи організації складних інформаційних систем;
- формалізоване представлення системи.

У лекційному курсі «Комп'ютерна графіка» для студентів кваліфікаційного рівня «бакалавр», що навчаються за напрямом 122 "Комп'ютерні науки", використано такі результати:

- контурний аналіз зображень;
- метод анімування зображень;
- метод класифікації зображень.

Директор ІКНІ,
д.т.н., професор М.О. Медиковський

Завідувач кафедри СШІ,
д.т.н., професор Н.Б.Шаховська

Доцент кафедри СШІ,
к.е.н., доцент Н.І. Бойко

ЗАТВЕРДЖУЮ

Заступник директора ТОВ «Українські інформаційні технології»

«20» червня 2022 р.

АКТ

про впровадження результатів дисертаційної роботи
аспіранта кафедри «Системи штучного інтелекту»
Національного університету «Львівська політехніка»
Шамуратова Олексія Юрійовича

Цей акт підтверджує, що результати дисертаційної роботи Шамуратова О.Ю. були використані для розроблення системи з анімування зображень у м. Львів у 2021-2022рр.

Впровадження дисертаційних досліджень О.Ю. Шамуратова полягає у наступному:

- Розроблено метод контурного аналізу та пошуку об'єкта на зображенні з використання оператора Собеля на основі застосування маски коефіцієнтів для середніх значень. Це стало основою для розроблення моделі опрацювання зображення. Також метод контурного аналізу може бути реалізований на обмежених обчислювальних ресурсів 500 МБ оперативної пам'яті та 1 Ггц частоти процесора, що цілком відповідає поставленим вимогам для використання на мобільних пристроях.
- Розроблено модель класифікації та кластеризації об'єктів із попереднім застосуванням аналізу геометричних ознак та гістограми градієнтів інтенсивності та з використанням оригінальної архітектури згорткової нейронної мережі. Як функцію активації використано гіперболічний тангенс. В структурі нейронної мережі є 2 зготкових шари, 2 підвибіркових шари та один повнозв'язний шар. Це дало змогу опрацьовувати об'єкти зі стійкістю до повороту або масштабування об'єкту, зміною ракурсу без необхідності їх повторного аналізу. Також це дало змогу зберігати проаналізовані об'єкти в базі даних бех необхідності їх повторного опрацювання.

Даний акт не є підставою для взаємних фінансових розрахунків.

Андрій Агеєв — Менеджер Центру майстерності по розробці програмного забезпечення

Тріска Роман — Менеджер Центру майстерності по розробці програмного забезпечення