

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
« ____ » _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи відновлення деталей на
розмитих зображеннях за допомогою нейронної мережі”

Виконав здобувач вищої освіти
IV курсу, групи КН-22
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Фідря М.О.
« ____ » _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнова Т.В.
« ____ » _____ 2026 р.

Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань . 12 “Інформаційні технології”
Спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна (освітньо-наукова) програма “Комп’ютерні науки”

ЗАТВЕРДЖУЮ

Завідувач кафедри

д.т.н., проф.

Олексій СМІРНОВ

« 15 » січня 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Фідрі Максиму Олександровичу

(прізвище, ім’я, по батькові)

1. Тема роботи

Програмне забезпечення системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі

2. Керівник роботи

Смірнова Тетяна Віталіївна, канд. техн. наук, доцент

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 116-02 від 06.02.2026 року

3. Строк подання студентом роботи до захисту 23.05.2026 р.

4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи

1 аркуш

Функціональна схема системи

1 аркуш

Діаграма процесів

1 аркуш

Блок-схема алгоритму роботи додатку

2 аркуша

7. Дата видачі завдання « 15 » січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 15 » січня 2026 р.

Підпис керівника

Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 15 » січня 2026 р.

Підпис здобувача

Фідря М.О.
(прізвище та ініціали)

АНОТАЦІЯ

Фідря М.О. Програмне забезпечення системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

Метою розробки є програмне забезпечення системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

Результат роботи – програмна реалізація системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерні науки, відновлення деталей на розмитих зображеннях, нейронна мережа

ABSTRACT

Fidria M.O. Software for the system of restoring details on blurred images using a neural network. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the system of restoring details on blurred images using a neural network.

The purpose of the development is software for the system of restoring details on blurred images using a neural network.

The result of the work is a software implementation of the system of restoring details on blurred images using a neural network.

In the process of working on the software model, an analysis of existing hardware and software tools was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on a PC with Windows 10/11 OS.

The program is developed in the Python environment.

Keywords: computer science, detail recovery in blurred images, neural network

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП.....	3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	5
1.1 Призначення системи.....	5
1.2 Область застосування.....	6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	11
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.....	11
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування.....	22
2.3 Розгорнута постановка завдання	27
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	28
3.1 Опис функціонування системи	28
3.2 Розробка структурної схеми.....	32
3.3 Розробка функціональної схеми	37
3.4 Розробка діаграми процесів.....	53
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	55
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	55
4.2 Захист розробленого програмного забезпечення.....	81
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	87
6 ОСНОВНІ ВИСНОВКИ.....	95
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	97

					ВКРБ-122.26.0032.00.00.ПЗ			
Вим	Арк.	№ докум.	Підп.	Дата	Програмне забезпечення системи відновлення деталей на розмитих зображеннях за допомогою нейронно мережі	Літ.	Аркуш	Аркушів
Розроб.	Фідоря М.О.					Б	1	103
Перев.	Смірнова Т.В.							
Н.контр.	Коваленко А.С.					ЦНТУ КН-22		
Затв.	Смірнов О.А.							

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	—	база даних
ЗНМ	—	згорткові нейронні мережі
ПЗ	—	програмне забезпечення
СКП	—	середньоквадратична помилка
ЦСВС	—	цифрові системи відеоспостереження

К6ПЗ_2026

ВСТУП

Актуальність теми. У сучасному світі активно використовуються системи відеоспостереження. Однак не всі камери дають гарна якість зображення. Тому актуальним є завдання поліпшення якості зображення, з метою розпізнавання й виявлення конкретних об'єктів. У даній роботі розробляється алгоритм, що дозволяє відновлювати деталі на розмитих фотографіях і кадрах відео. Алгоритм, що одержав назву “візуальна модель депроекції”, заснований на так званій “згортковій нейронній мережі” (англ. convolutional neural network, ЗНМ)

Для цього навчається нейромережа, завантажуючи їй пари аналогічних зображень. Перше зображення в парі було розмитим, друге – чітким. Поступово нейромережа навчилася розуміти принципи розмиття в русі й обертати цей ефект назад. У результаті вийшов інструмент інтелектуальної обробки, у який можна завантажити розмите фото й одержати на виході чітку картинку.

Коли нейромережа зіштовхується з неякісним зображенням з розмитими елементами, вона аналізує їх, намагаючись з'ясувати причини розмиття. Потім вона синтезує нові зображення, які поєднують у собі дані як більше чіткої, так і з розмитої частин. Наприклад, у вас є кадри вулиці, на яких будинку виглядають чіткими, а люди, що рухаються, – розмитими. Нова технологія дозволить “перезібрати” знімок, зробивши всі об'єкти на ньому чіткими.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

– Дослідження системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

– Програмна реалізація системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ_2026

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для відновлення деталей на розмитих фотографіях і кадрах відео в системах розпізнавання графічних образів.

Однією з актуальних проблем сучасних інформаційних технологій у цей час залишається розпізнавання образів, зокрема графічно представлених символів.

Для рішення даного завдання використовується ряд алгоритмів, які можна умовно розділити на три категорії:

- алгоритми, засновані на порівнянні шаблонів;
- алгоритми, засновані на методах теорії рішень;
- алгоритми, що використовують нейронні мережі.

Всі перераховані вище групи алгоритмів, мають, у цілому, однакову структуру й містять такі блоки:

- блок навчання системи на наборі образів, подібним тим, які будуть розпізнаватися системою в майбутньому.

По завершенню навчання, його результати зберігаються;

- блок розпізнавання, що використовуючи інформацію, отриману на етапі навчання, відносить незнайомий образ до одному із класів за допомогою певного математичного апарата (класифікатора).

Варто помітити, що шаблонні методи, у силу своєї негнучкості (прихильності до шрифту, масштабу, нахилу символів) не підходять для загального рішення даного завдання, однак у силу своєї простоти, невеликої ресурсоемності, малим обчислювальним витратам вони, з певними модифікаціями, використовуються досить часто при рішенні окремих завдань розпізнавання [1].

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

Метод нейронних мереж, що використовує деякі принципи штучного інтелекту, широко використовується в більшості сучасних систем розпізнавання символів.

Серед його переваг варто згадати універсальність, а також незначні вимоги до проектування алгоритмів у силу особливостей нейромереж.

Недоліками даного методу є значний обсяг обчислювальних ресурсів, необхідних для організації процесу навчання, а також погана передбачуваність результатів розпізнавання.

Найбільшою перевагою таких методів є висока точність розпізнавання.

Алгоритми, засновані на методах теорії рішень, у багато разів спрощують процес навчання, але їхня точність значно уступає точності нейромережових алгоритмів.

Важливою перевагою даних алгоритмів є подання символів у векторній формі (у вигляді набору характеристик), у той час як більшість нейромережових алгоритмів оперують із растровим зображенням.

При використанні деяких класифікаторів теорії рішень, можливо домогтися високого ступеня передбачуваності результатів розпізнавання.

1.2 Область застосування

Розмиті зображення – одна з найпоширеніших проблем, з якими стикаються люди під час роботи з фотографіями. Чи то тремтячий знімок, зроблений смартфоном, розмите в русі фото продукту, відсканований документ з нечитабельним текстом, чи стара сімейна фотографія, яка з часом втратила деталі, розмиття може значно знизити якість зображення та зручність використання.

Саме тут і стають у пригоді інструменти для розмиття зображень за допомогою штучного інтелекту.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

В останні роки штучний інтелект кардинально змінив те, як ми справляємося з усуненням розмиття зображень. Замість того, щоб покладатися на базові фільтри підвищення різкості, сучасні рішення для усунення розмиття зображень на основі штучного інтелекту використовують моделі глибокого навчання для аналізу пікселів, прогнозування відсутніх деталей та інтелектуального відновлення чіткості. В результаті користувачі тепер можуть усунути розмиття файлів зображень онлайн лише кількома клацаннями миші – без необхідності мати професійні навички редагування.

У цій статті ми пропонуємо вичерпний огляд найкращих інструментів для розмиття зображень на основі штучного інтелекту, порівнюємо їхні функції та ціни, а також допомагаємо вам вибрати найкраще рішення відповідно до ваших потреб.

Що таке розмиття зображення за допомогою штучного інтелекту

Визначення зображення з розмиттям за допомогою штучного інтелекту

Штучне усунення розмиття зображення – це процес використання штучного інтелекту для зменшення або видалення розмиття з цифрових зображень. На відміну від традиційних методів редагування фотографій, усунення розмиття зображень на основі штучного інтелекту намагається відновити втрачені деталі, а не просто покращити краї.

Простіше кажучи, інструмент для розмиття зображень на основі штучного інтелекту аналізує розмите зображення, розуміє, як воно має виглядати, а потім відтворює чіткішу версію.

Як працює розмиття зображення за допомогою штучного інтелекту

Більшість інструментів штучного інтелекту для розмиття зображень побудовані на згорткових нейронних мережах, навчених на мільйонах пар зображень. Ці моделі вивчають закономірності з чітких та розмитих зображень, що дозволяє їм передбачати дрібні деталі, такі як краї, текстури та форми.

Розширені інструменти поєднують:

- Алгоритми усунення розмиття зображення.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

- Надмірне масштабування роздільної здатності.
- Зменшення шуму.
- Реконструкція краю.

Цей багатоетапний процес дозволяє штучному інтелекту розмивати файли зображень, зберігаючи при цьому природність та візуальну узгодженість результатів.

Різниця між традиційним підвищенням різкості та розмиттям за допомогою штучного інтелекту

Традиційне збільшення різкості збільшує контрастність по краях, що часто створює ореоли або артефакти. Технологія штучного інтелекту для усунення розмиття зображень, навпаки, відновлює структуру та деталі на глибшому рівні. Це робить усунення розмиття зображень за допомогою штучного інтелекту набагато ефективнішим, особливо для облич, тексту та фотографій продуктів.

Обмеження технології розмиття зі штучним інтелектом

Хоча інструменти штучного інтелекту для усунення розмиття зображень є потужними, вони не є магією. Надзвичайно розмиті зображення, такі як ті, що мають сильне розмиття руху або відсутню інформацію про пікселі, можуть бути не повністю відновлені. Розуміння цих обмежень допомагає встановити реалістичні очікування.

Навіщо потрібні інструменти для розмиття зображень

Виправлення розмиття руху та тремтіння камери

Рухи рук і тремтіння камери є одними з найпоширеніших причин розмиття фотографій, особливо в умовах слабкого освітлення. Інструменти штучного інтелекту для усунення розмиття зображень можуть ефективно виправляти легке або помірне розмиття від руху, аналізуючи візерунки на краях та відновлюючи втрачену різкість. В результаті зображення стають помітно чіткішими без появи різких артефактів.

Відновлення старих зображень та зображень з низькою роздільною здатністю

Старі фотографії або фотографії з низькою роздільною здатністю часто мають розмитість, вицвілі деталі та візуальний шум. Усунення розмиття за

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		8

допомогою штучного інтелекту допомагає відновити чіткість, покращуючи риси обличчя, текстури та контури, зберігаючи при цьому природний вигляд. Це робить старовинні фотографії більш зручними для цифрового архівування, обміну або друку.

Покращення якості зображень для електронної комерції

В електронній комерції якість зображення відіграє вирішальну роль у довірі користувачів та коефіцієнтах конверсії. Надійне рішення для усунення розмиття зображень допомагає продавцям покращити фотографії товарів, підвищуючи різкість та чіткість, роблячи товари більш деталізованими та професійними. Чіткіші візуальні ефекти можуть значно підвищити довіру клієнтів та покращити рішення про покупку.

Покращення читабельності тексту на зображеннях

Розмиті знімки екрана, відскановані документи або сфотографований текст може бути важко читати та інтерпретувати. Інструменти штучного інтелекту для усунення розмиття зображень покращують краї тексту та контрастність, роблячи слова чіткішими та розбірливішими. Це особливо корисно для обміну документами, презентаціями або навчальними зображеннями онлайн.

Як ми оцінювали ці інструменти для розмиття на основі штучного інтелекту

Відновлення якості розмиття та деталей

Ми перевірили, наскільки ефективно кожен інструмент може відновлювати розмиття файлів зображень, пошкоджених різними типами розмиття, включаючи розмиття руху, розмиття фокуса та артефакти стиснення. Особливу увагу було приділено тому, наскільки добре відновлювалися дрібні деталі, краї та текстури без створення неприродних спотворень. Метою було оцінити реальне покращення, а не лише візуальну різкість.

Простота використання та зручність користування

Потужний інструмент для розмиття зображень на основі штучного інтелекту має бути інтуїтивно зрозумілим та доступним для користувачів будь-якого рівня кваліфікації. Ми оцінювали такі фактори, як швидкість завантаження, зрозумілість інтерфейсу та те, наскільки плавно користувачі можуть переходити

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

від завантаження зображення до остаточного завантаження. Оптимізований робочий процес є важливим як для звичайних користувачів, так і для професіоналів.

Підтримувані формати зображень та роздільна здатність

Підтримка поширених форматів зображень, таких як JPG, PNG та HEIC, є важливою для щоденної сумісності. Ми також розглянули, чи кожен інструмент дозволяє виводити зображення з високою роздільною здатністю без надмірних обмежень. Це особливо важливо для користувачів, яким потрібні зображення, придатні для друку або комерційного використання.

Ціни та обмеження безкоштовного використання

Прозоре ціноутворення та обмеження справедливого використання відіграють важливу роль під час вибору інструменту для усунення розмиття зображень. Ми дослідили безкоштовні плани, кредитні системи та моделі підписки, щоб визначити, чи виправдовують пропоновані функції вартість. Інструменти, які чітко повідомляють про обмеження та ціни, як правило, забезпечують кращу загальну цінність.

Найкращі сценарії використання

Кожен інструмент найкраще працює в певних сценаріях, а не в усьому. Ми визначили, чи рішення більше підходить для зображень у соціальних мережах, професійної дизайнерської роботи, очищення документів чи швидких разових виправлень. Розуміння цих переваг допомагає користувачам вибрати правильний інструмент для своїх конкретних потреб.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		10

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Pinterest. Для дизайнерів і не тільки

Pinterest має інструмент візуального пошуку, але заснований на небагато інших алгоритмах. Застосунок, у першу чергу, призначено для пошуку картин, предметів інтер'єра й інших речей які можуть придатися дизайнерам і художникам. Як і Google Lens, Pinterest дозволяє одержати посилання на покупку потрібного вам предмета, якщо це можливо. До того ж, просунуті соціальні функції застосунку допоможуть вам попросити допомоги в співтовариства, якщо раптом алгоритми не видають вам потрібний результат.

Coinoscope. Пошук монет

Так, програму можна назвати досить вузькоспеціалізованим застосунком для нумізматів, адже воно націлено лише на певну категорію предметів, а саме, на монети. Програма може розпізнавати як сучасні, так і древні монети. Однак крім цих можливостей, Coinoscope – це ще й справжній архів з описом монет самих різних епох. Невеликий підручник для починаючих колекціонерів або просто аматорів старовини.

PlantNet. Довідник рослин

PlantNet, як треба з назви, ідентифікувати різні види рослин, включаючи квіти, трави, кактуси, дерева й багато чого іншого. Крім назви рослини, PlantNet дозволяє довідатися подробиці й факти про те, що ви виявили. Якщо ви хочете, вивчити бібліотеку застосунку самостійно. У неї внесено більше 20000 видів найрізноманітнішої флори.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		11

Vivino. Для аматорів гарного вина

Vivino – застосунок, що допоможе вам купити вино кращої якості. Усе, що вам потрібно зробити, це зробити знімок етикетки пляшки. Після цього програма обробить його й надасть вам всю необхідну інформацію. Крім загального рейтингу напою, Vivino покаже відкликання споживачів і навіть середню ціну у вашім регіоні. На даний момент застосунок «знає» близько 9,2 мільйони видів вин.

Food Camera. Лічильник калорій

Food Camera намагається допомогти вам відразу у двох напрямках: стежити за спожитою кількістю калорій, і позбавити вас від необхідності вводити всі вручну. Food Camera поставляється з персональним помічником на базі штучного інтелекту, що працює у зв'язуванні з вашою камерою. Досить просто навести камеру пристрою на продукти живлення, і застосунок видасть вам інформацію про калорійність і енергетичну цінність. Працює також і на напоях. Вся необхідна інформація заноситься в журнал живлення для того, щоб ви могли відслідковувати свої досягнення.

Lookout – застосунок від Google, що розповість сліпим і слабобачучим про навколишнє оточення

Компанія Google випустила Android-застосунок Lookout, що дозволяє сліпим і слабобачучим дізнаватися про об'єкти поруч із ними.

Як повідомляється, програма працює в трьох режимах: вивчення, шопінг і читання. В основному режимі (вивчення) застосунок використовує дані з тилової камери смартфона для розпізнавання об'єктів перед пристроєм у реальному часі. При цьому Lookout озвучує не тільки знайдені об'єкти, але і їхній напрямок у вигляді годинної позиції, наприклад, «комп'ютер на друга година». У режимі шопінгу застосунок працює аналогічно, але настроєно на розпізнавання штрих-кодів і цінників. У режимі зачитування тексту програма розпізнає не об'єкти, а текст на них, і відразу ж зараховує його.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

Також Lookout дозволяє зробити за допомогою смартфона фотографію й одержати її докладний опис. Крім того, користувач може налаштувати, наскільки детально алгоритми будуть описувати об'єкти в кадрі.

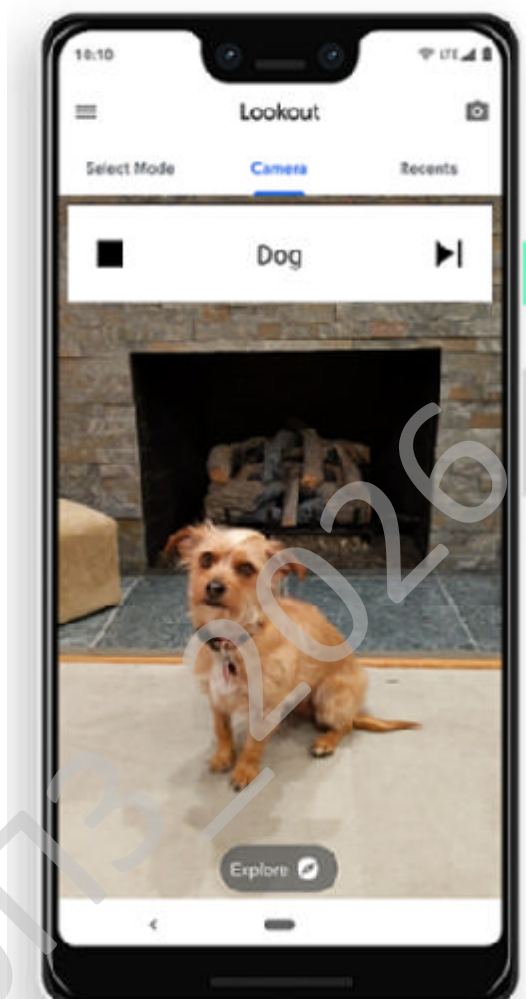


Рисунок 2.1 – Інтерфейс користувача Lookout

Застосунок сумісний із системною службою TalkBack, що дозволяє взаємодіяти з інтерфейсом за допомогою жестів на екрані. Крім того, програму можна використовувати не тільки тримаючи смартфон у руці, але й повісивши його в чохлі на шию. У цьому випадку користувач може призупинити роботу алгоритму розпізнавання, закривши камеру рукою на парі секунд, а також запустити його заново, постукавши два рази по задній поверхні смартфона. Крім

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		13

цього, управляти режимами роботи Lookout можна за допомогою жестів на сканері відбитків пальців.

Seek

Seek – це альтернатива застосунку Shazam, що вміє визначати рослини й тварини по фотографії. Для використання програми, розробленою компанією iNaturalist, досить просто неї скачати, після чого в неї зайти й навести камеру, що активізувалася, вашого мобільного пристрою на об'єкт, що ви хочете ідентифікувати – це може бути будь-яка квітка на клумбах, що на той момент виявиться поруч – після цього програма, задіявши систему розпізнавання зображень, видасть коротку інформацію про рослину, що виявилось в кадрі. Творці застосунку завіряють, що на даному етапі в його базі перебуває більше 30 тисяч різних тварин і рослин, які софт може визначити. У найближчому ж майбутньому автори хочуть у кілька разів збільшити можливості свого дітища, у тому числі за допомогою картинок, які будуть завантажувати самі користувачі.

CamFind: пошук інформації про об'єкт по його фото

CamFind для iOS і Android прагне дати максимальна кількість інформації про сфотографованому вами об'єкті.

Одна з можливостей – пошук товарів в інтернет по їхніх фотографіях і штрихових кодах. Не варто відразу купувати річ, що сподобалася, скористайтесь застосунком, можливо, воно знайде його в мережі по більше привабливій ціні.

Зробіть знімок речі, що сподобалася, і CamFind знайде, де неї купити в інтернеті. Застосунок розпізнає предмети, відомі місця, QR-коди й штрихові коди товарів, і дає безліч посилань на них у мережі. Загалом, прагне дати максимальна кількість інформації про сфотографований об'єкт. Раніше пошуковий движок був реалізований лише в застосунку для iOS, з недавніх пор став доступний і для Android.

Нами було протестовано більше Android-застосунків. Він вже містить чималий список функцій, хоча цілком можливо, що власники iPhone мають на озброєнні більші можливості. Деякі з них нам удалося оцінити, а частина – немає.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

– Сама цікава функція – пошук товарів в інтернеті по зображеннях – працює добре. Ви можете скормити застосунку фотографію з пам'яті телефону або зробити новий знімок. Якщо в об'єктив камери потрапив ряд предметів, ви можете явно вказати утиліті на головний з них. CamFind розпізнає товари по ключових словах і приводить ряд посилань на інтернет-магазини.

Сканування QR-кодів відбувається без запинок. Будь-який лінк відкриється в самому застосунку.

– При зчитуванні баркодів виникли цілком з'ясовні труднощі. Товари закордонного походження визначалися миттєво. Продукція української промисловості ставила CamFind у тупик. З важкими умовами зйомки застосунок упорався. .

– Автори застосунку заявляють про функцію розпізнавання відомих місць. Навіть не намагалися неї протестувати. Шукайте, де попити чай або кава? CamFind вам допоможе. Але не в нашій випадку. Причини розкриті в попередньому абзаці матеріалу. .

– Пошуковий движок уміє розпізнавати афіші фільмів і рекламу ігор, а також перекладати текст. .

– Голосовий пошук не працює з ламаним англійським, прийде тренувати дикцію. .

Виразно, варто простежити за розвитком утиліти. Застосунок не містить реклами, не пропонує супутніх товарів для шуканої речі. Не гальмує й не глючить. І це тільки старт.

На сучасний момент, найбільш затребуваною можливістю CamFind є пошук в інтернеті товару, що цікавить, по його фото або штрих-коду.

Самонавчальна програма від Disney Research для розпізнавання образів

Група дослідників з піттсбургського підрозділу Disney Research розробила систему комп'ютерного зору, що використовує деякі принципи людського зору.

Зокрема, вона містить алгоритми для самонавчання й здатна згодом поліпшувати розпізнавання об'єктів.

Як і більшість інших систем комп'ютерного зору, розробка Disney Research будує концептуальну модель для кожного об'єкта, будь це літак або дозатор мила. При цьому використовується той, якого навчають, алгоритм, що аналізує безліч фотографій даного об'єкта.

Відмінна риса алгоритму Disney Research – у тому, що він згодом використовує цю модель для розпізнавання об'єктів на відео, одночасно видобуваючи нову інформацію про ці об'єкти й доповнюючи модель, закладену споконвічно. Це дозволяє розпізнавати об'єкти в більше широкому діапазоні, навіть якщо вони виглядають інакше, чим на зразках, що зустрічалися раніше.

Концептуальна модель для кожного об'єкта поступово розширюється й уточнюється в міру того, як система зіштовхується з новою інформацією. Теоретично, такий метод може привести до того, що діючи без нагляду, система припише об'єкту невласиві для нього характеристики, через що виникнуть помилки розпізнавання. Але автори програми говорять, що така проблема поки не замічена.

Огляд AIEnhancer

AIEnhancer – це універсальний інструмент для покращення зображень на базі штучного інтелекту, призначений для усунення розмиття файлів зображень та покращення загальної візуальної якості. Він зосереджений на повністю автоматизованій обробці, що дозволяє користувачам швидко відновлювати чіткість та різкість без необхідності будь-яких технічних навичок редагування. Завдяки зручному інтерфейсу та надійним результатам, він добре підходить як для звичайних користувачів, так і для творців контенту.

Основні характеристики:

- Обробка зображень за допомогою штучного інтелекту, що автоматично виявляє та виправляє поширені проблеми розмиття, одним клацанням миші.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

– Зменшення шуму та покращення деталізації для покращення чіткості, зберігаючи при цьому природні текстури.

– Підтримує вивід високої роздільної здатності, що підходить для соціальних мереж, вебсайтів та цифрових публікацій.

Ціна:

– Повністю безкоштовно, без необхідності підписки.

– Без прихованих платежів та водяних знаків.

Переваги:

– Забезпечує природні результати розмиття без надмірного збільшення різкості чи візуальних артефактів.

– Надзвичайно простий у використанні, не потребує налаштування чи навчання.

– Висока швидкість обробки, що дозволяє користувачам покращувати зображення за лічені секунди.

– Стабільна якість виводу, що робить його надійним для багаторазового щоденного використання.

– Вільний доступ робить його дуже зручним для студентів, творців та малого бізнесу.

Недоліки:

– Обмежене ручне керування для користувачів, які бажають дрібних налаштувань.

– Менш ефективний для екстремального розмиття, наприклад, для зображень, що сильно розфокусовані.

Найкраще підходить для користувачів, яким потрібен простий, швидкий та надійний спосіб розмиття файлів зображень для щоденного використання, контенту соціальних мереж та швидкого візуального покращення без витрат та складнощів.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

Огляд Artguru

Artguru – це багатоцільова платформа для покращення зображень та відео на основі штучного інтелекту з потужними можливостями усунення розмиття зображень.

Основні характеристики:

- Штучний інтелект розмиває зображення для фотографій, відео та PDF-файлів.

- Пакетна обробка.

- Додаткові інструменти, такі як видалення фону.

Ціна:

- Доступна безкоштовна пробна версія.

- Плани підписки від \$9,99/місяць.

Переваги:

- Комплексний пакет покращень.

- Підтримує кілька типів файлів.

Недоліки:

- Розширені функції вимагають підписки.

- Продуктивність залежить від розміру файлу.

Найкраще підходить для користувачів, яким потрібно більше, ніж просто інструмент для розмиття зображень, особливо для мультимедійних проектів.

Огляд розмиття зображення

Unblurimage зосереджений саме на швидкому та простому усуненні розмиття зображень.

Основні характеристики:

- Спеціальний механізм розмиття зображень на базі штучного інтелекту.

- Реєстрація не потрібна.

- Підтримує поширені формати зображень.

Ціна:

- Безкоштовне використання (з обмеженнями).

- Платні опції для виходу з вищою роздільною здатністю.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		18

Переваги:

- Надзвичайно простий у використанні.
- Швидка обробка.
- Чистий інтерфейс.

Недоліки:

- Обмежені розширені функції.
- Не ідеально підходить для професійних робочих процесів.

Найкраще для швидкого розмиття файлів зображень онлайн без додаткових інструментів чи налаштування.

Огляд LetsEnhance

LetsEnhance – це професійна платформа для покращення зображень за допомогою штучного інтелекту, яка широко використовується в електронній комерції та дизайні.

Основні характеристики:

- Штучний інтелект для усунення розмиття зображення в поєднанні з високоякісним масштабуванням.
- Кілька режимів покращення.
- Підтримує виведення великих зображень.

Ціна:

- Обмежена кількість безкоштовних кредитів.
- Платні плани починаються приблизно від 9 доларів на місяць, масштабуються залежно від використання.

Переваги:

- Відмінне відновлення деталей.
- Ідеально підходить для комерційних зображень.

Недоліки:

- Безкоштовне використання обмежене.
- Вища вартість для активних користувачів.

Найкраще підходить для професіоналів, яким потрібне високоякісне усунення розмиття та масштабування зображення.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		19

Огляд Pollo.ai

Pollo.ai – це онлайн-інструмент для початківців, розроблений для простого відновлення зображень.

Основні характеристики:

- Обробка зображень зі штучним інтелектом, що розмиває їх, одним клацанням миші.

- Добре підходить для портретів та старих фотографій.

Ціна:

- Доступна безкоштовна пробна версія.
- Платні плани залежать від якості експорту.
- 29 доларів на місяць.

Переваги:

- Дуже простий у використанні.
- Підходить для звичайних користувачів.

Недоліки:

- Обмежені розширені налаштування.
- Проблеми з сильним розмиттям.

Найкраще для користувачів, які вперше знайомляться з усуненням розмиття зображень за допомогою штучного інтелекту та хочуть швидких результатів.

Рекомендовані сценарії та вибір інструментів

Найкращі інструменти для створення зображень у соціальних мережах

Для контенту для соціальних мереж швидкість та простота використання є ключовими факторами. **AIEnhancer** та **Unblurimage** добре працюють для зображень, готових до використання в соціальних мережах, швидко покращуючи різкість та чіткість без надмірної обробки. Їхні прості робочі процеси роблять їх ідеальними для творців, яким потрібні швидкі результати для частих публікацій.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

Найкращі інструменти для фотографій товарів електронної комерції

Високоякісне зображення є важливим для формування довіри та підвищення коефіцієнта конверсії в інтернет-магазинах. **LetsEnhance** забезпечує більш професійні та стабільні результати, особливо для зображень товарів, які потребують високої роздільної здатності та чітких деталей. Він добре підходить для платформ електронної комерції, де якість зображення безпосередньо впливає на ефективність продажів.

Найкращі інструменти для реставрації старих фотографій

Відновлення старих або пошкоджених фотографій часто вимагає ретельного балансу між чіткістю та природним виглядом. **Pollo.ai** та **Artguru** ефективно підходять для цієї мети, допомагаючи оживити вицвілі деталі та зменшити розмиття, зберігаючи при цьому реалістичний вигляд. Ці інструменти добре працюють для особистих архівів, сімейних фотографій та історичних зображень.

Висновок

Технологія штучного інтелекту для усунення розмиття зображень змінила спосіб відновлення та покращення фотографій, дозволяючи користувачам швидко покращувати чіткість та деталізацію, що раніше займало б години під час ручного редагування. Чи то особисті фотографії, контент для соціальних мереж, чи професійні проекти, існують інструменти для будь-якого робочого процесу та бюджету. Серед них **AIEnhancer** вирізняється поєднанням швидкості, простоти та якості. Він повністю безкоштовний, не вимагає технічних знань і забезпечує природний вигляд результатів лише за кілька кліків, що робить його ідеальним для початківців та звичайних користувачів.

Розуміючи ваші потреби та порівнюючи доступні функції, **AIEnhancer** пропонує надійне рішення для постійного досягнення чіткіших, вишуканіших та досконаліших зображень. Його інтуїтивно зрозумілий інтерфейс та швидка обробка забезпечують миттєві професійні результати, встановлюючи баланс між доступністю та високою якістю відтворення, з яким можуть зрівнятися мало які інші інструменти.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		21

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – динамічна інтерпретована об'єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Офіційний сайт мови програмування Python <https://www.python.org/>. Python – багатоцільова мова програмування, яка дозволяє писати код, що добре читається. Відносний лаконізм мови Python дозволяє створити програму, яка буде набагато коротше свого аналога, написаного на іншій мові. Python – багатоплатформова мова програмування. Це означає, що програми на Python можна запускати в різних операційних системах без будь-яких змін.

Ще однією перевагою Python є його стандартна бібліотека, яка встановлюється разом з Python і містить готові інструменти для роботи з операційною системою, веб-сторінками, базами даних, різними форматами даних, для побудови графічного інтерфейсу програм тощо. Програми, написані на мові програмування Python, можуть бути як невеликими скриптами, так і складними системами. Python абсолютно безкоштовний.

Швидкість виконання коду Python

Один з можливих недоліків Python – швидкість виконання коду. Python не є компільованою мовою. Код на Python спочатку компілюється у внутрішній байт-код, який потім виконується інтерпретатором Python. У більшості випадків при використанні Python виходять програми повільніші в порівнянні з такими мовами, як C.

Втім, сучасні комп'ютери мають таку обчислювальну потужність, що для більшості застосунків швидкість розробки важливіша швидкості виконання, а програми на Python зазвичай пишуться набагато швидше.

Окрім того, Python легко розширюється модулями, написаними на C або C++. Такі модулі можуть використовуватися для виконання частин програми, що створюють інтенсивне навантаження на процесор.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

Використання Python

Python використовується для різних цілей: для створення ігор і веб-застосунків, розробки внутрішніх інструментів для різноманітних проектів. Мова також широко застосовується в науковій області для досліджень і розв'язування прикладних завдань.

Застосування мови програмування Python:

1. BitTorrent – протокол для обміну даними.
2. Ubuntu Software Center – вільне програмне забезпечення для пошуку, установки і видалення пакунків в системі Ubuntu Linux.
3. Blender – програма для створення тривимірної комп'ютерної графіки, що включає засоби моделювання, анімації, вимальовування, пост-обробки відео, а також створення відеоігор.
4. GIMP – растровий графічний редактор, із підтримкою векторної графіки.
5. World of Tanks.
6. Вільна енциклопедія Вікіпедія.
7. Пошукова система Google.
8. DropBox – файловий хостинг, що включає персональне хмарне сховище, синхронізацію файлів і програму-клієнт.
9. YouTube – популярне відеосховище.

Версії Python

Мови програмування з часом змінюються – розробники додають в них нові можливості, а також виправляють помилки. Так з'являються різні версії мови. Наприклад, код написаний на Python 2 у більшості випадків не буде працювати у версії Python 3 без внесення додаткових змін.

Процесор є найважливішим компонентом в комп'ютері. Одна з основних функцій процесора – це обробка даних згідно комп'ютерної програми, яка є списком інструкцій, шляхом виконання арифметичних і логічних операцій над фрагментами даних.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

Кожна інструкція в програмі – це команда, яка «повідомляє» процесору, яку операцію він повинен виконати. Процесор комп'ютера може розуміти лише ті інструкції, які написані на машинній мові. Машинна мова – це штучна мова, створена для передачі команд комп'ютеру. За допомогою машинної мови створюються ефективні програми, оскільки розробник отримує доступ до всіх можливостей процесора. Машинна мова – мова низького рівня.

Інструкція машинної мови існує для кожної операції, яку процесор здатний виконати – є інструкція для додавання чисел, є інструкція для віднімання чисел і т.д. Увесь набір інструкцій, який центральний процесор може виконати, відомий як набір інструкцій процесора.

Наприклад, у вас є певна програма, яка зберігається на диску вашого комп'ютера. Для виконання програми, ви здійснюєте подвійний клік на значку програми. Це змушує програму копіюватися з диска в оперативну пам'ять, після чого процесор комп'ютера виконує копію програми, яка знаходиться в оперативній пам'яті.

Коли процесор виконує інструкції програми, він бере участь у процесі, який є відомим як цикл *fetch – decode – execute* (отримати – декодувати – виконати). Цей цикл виконується для кожної інструкції у програмі і складається з трьох кроків:

Отримати

Програма – це послідовність інструкцій на машинній мові. Першим кроком циклу є завантаження (отримання) наступної інструкції з пам'яті в процесор.

Декодувати

Інструкція машинної мови – це двійкове число, яке представляє команду, що повідомляє процесору виконати певну операцію. На цьому кроці процесор декодує інструкцію, яку було «витягнуто» з пам'яті, для визначення того, яка операція повинна виконуватись.

Виконати

Останній крок циклу – виконати операцію.

Хоча процесор комп'ютера розуміє тільки машинну мову, людині непрактично писати програми на машинній мові. Така програма може мати тисячі або навіть мільйони бінарних інструкцій, і написання такої програми буде дуже обтяжливим процесом. З цієї причини була створена мова асемблера як альтернатива машинній мові. Замість використання двійкових чисел для написання інструкцій, мова асемблера використовує короткі слова, відомі як мнемокоди. Незважаючи на те, що мова асемблера не вимагає двійкових інструкцій, як у випадку машинної мови, проте вона вимагає високих знань про процесор. Використовуючи мову асемблера, навіть для найпростішої програми, необхідно написати велику кількість інструкцій.

Мова програмування високого рівня дозволяє створювати складні програми, не знаючи, як працює процесор, і не записуючи великої кількості інструкцій низького рівня. Крім того, більшість мов програмування високого рівня використовують слова, які легко зрозуміти.

Python – одна із популярних сучасних мов програмування високого рівня. Python – інтерпретована мова програмування. Python – це високорівнева інтерпретована мова програмування, на відміну від C++, яка є прикладом компільованої мови програмування. Назва Python відноситься як до мови програмування, так і до інтерпретатора – комп'ютерної програми, яка зчитує початковий код (написаний на Python) і виконує інструкції (команди).

Для перекладу мови високого рівня на машинну мову доступні два типи програм:

1. Компілятор.
2. Інтерпретатор.

Завантаження Python

Версії інтерпретатора Python для різних операційних систем доступні для безкоштовного завантаження за адресою <https://www.python.org/downloads>.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		25

Середовище програмування для Python

Для написання програм використовують текстові редактори або інтегровані середовища розробки, які включають в себе різні інструменти для роботи з кодом: засіб для написання коду (текстовий редактор), інтерактивний інтерпретатор, відлагоджувач тощо.

Текстові редактори та інтегровані середовища програмування для Python:

- IDLE – стандартний редактор Python. Встановлюється разом з Python для користувачів Windows, окремим пакунком для користувачів Linux.
- Notepad++ – безкоштовний текстовий редактор початкового коду, який підтримує велику кількість мов, в тому числі і Python. Лише для користувачів Windows.
- Visual Studio Code – це легкий, але потужний редактор початкового коду, який розповсюджується безкоштовно і доступний у версіях для платформ Linux, Windows і macOS.
- PyScripter – інтегроване середовище розробки для мови програмування Python. Для користувачів Windows. Поширюється безкоштовно.
- Wing IDE 101 – вільне інтегроване середовище для Python, розроблене для навчання програмістів-початківців. Для користувачів Linux, Windows і macOS. Поширюється безкоштовно.
- Geany – вільний текстовий редактор з базовими елементами інтегрованого середовища розробки, доступний для операційних систем Linux, Windows і macOS.
- PyCharm – інтегроване середовище розробки для мови програмування Python. PyCharm є власницьким програмним забезпеченням. Наявна безкоштовна версія Community з усіченим набором можливостей. Для користувачів Linux, Windows і macOS.
- Thonny – IDE для вивчення програмування мовою Python. Для користувачів Linux, Windows і macOS.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформулювати висновки про виконаний обсяг робіт та одержані результати.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		27

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У даній роботі розробляється модель, яка відновлює цінні дані, втрачені з зображень та відео, які були "згорнуті" на менші розміри.

Модель може бути використана для відтворення відео з розмитими зображеннями руху або з нових типів камер, які фіксують рух людини по кутах, але лише як розпливчасті одновимірні лінії. Типове застосування даної моделі є системи відеоспостереження, коли потрібно за размитим кадром виявити об'єкт.

Крім того, вважається що такий підхід колись може бути використаний для перетворення 2D-медичних зображень у більш інформативні, але більш дорогі – 3D-сканування тіла, що може принести користь медичним зображенням у бідніших країнах. Захоплені візуальні дані часто згортають дані численних вимірів часу та простору в один або два виміри, які називаються "проекціями". Наприклад, рентгенограми обрушують тривимірні дані про анатомічні структури на плоске зображення. Або, розгляньте знімок довгих експозицій зірок, що рухаються по небі: Зірки, положення яких з часом змінюються, на нерухомому знімку виглядають як розмиті смуги.

Так само "кутові камери" виявляють переміщення людей по кутах. Це може бути корисно для, скажімо, пожежників, які знаходять людей у палаючих будівлях. Але камери не дуже зручні для користувачів. Наразі вони створюють лише проекції, що нагадують розмиті, чіткі лінії, що відповідають траєкторії та швидкості людини.

Кутові камери

Світло дозволяє нам бачити речі, які нас оточують, але що робити, якщо ми також могли використовувати його, щоб побачити речі, заховані за кутами?

Система візуалізації, яка може працювати з камерами смартфонів,

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		28

використовує інформацію про світловідбивання для виявлення предметів чи людей у прихованій сцені та вимірювання їх швидкості та траєкторії – і все це в режимі реального часу.

Щоб пояснити, уявіть, що ви спускаєтесь передпокою у формі Г і має стіну між собою та деякими предметами за кутом. Ці об'єкти відображають невелику кількість світла на землі у вашій лінії зору, створюючи нечітку тінь, яку називають "півтінь".

Використовуючи відео півтіні, система може зшити разом цілу низку одновимірних зображень, які розкривають інформацію про предмети за кутом.

Хоча ці об'єкти насправді не видно камері, ми можемо подивитися, як їхні рухи впливають на півтінь, щоб визначити, де вони і куди вони йдуть. Таким чином ми показуємо, що стіни та інші перешкоди з краями можна використовувати як "природні камери", що розкривають приховані сцени поза ними.

Можливість бачити перешкоди буде корисною для багатьох завдань, від пожежників, які знаходять людей у палаючих будівлях, до водіїв, які виявляють пішоходів у сліпих місцях.

Більшість підходів для бачення перешкод передбачають спеціальні лазери. Зокрема, дослідники світять камери на конкретні точки, які видно як спостережуваний, так і прихованій сцені, а потім вимірюють, скільки часу потрібно, щоб світло повернулося.

Однак ці так звані "камери часу польоту" є дорогими і можуть легко скинутись від навколишнього світла, особливо на вулиці.

На відміну від цього, техніка не вимагає активного проектування світла в простір, а працює в більш широкому діапазоні внутрішніх і зовнішніх середовищ і з нестандартними споживчими камерами.

Система переглядаючи відео півтіні, створює одновимірні зображення прихованої сцени. Одне зображення не є особливо корисним, оскільки містить неабияку кількість "галасливих" даних. Але спостерігаючи за сценою протягом

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		29

декількох секунд і зшиваючи разом десятки чітких зображень, система може розрізнити окремі об'єкти в русі і визначити їх швидкість і траєкторію.

У системі все ще є деякі обмеження. З очевидних причин вона не працює, якщо на сцені немає світла, і можуть виникнути проблеми, якщо в самій прихованій сцені мало світла. Він також може замикатися, якщо змінюються умови світла, наприклад, якщо сцена знаходиться на вулиці і хмари постійно рухаються по всьому сонцю. У камерах якості смартфонів сигнал також слабшає, коли ви віддаляєтесь від кута.

Винайдена модель "візуальної депроекції", яка використовує нейронну мережу для "вивчення" шаблонів, що відповідають низькомірним проекціям їх оригінальним зображенням високого розміру та відео. З огляду на нові прогнози, модель використовує те, що навчилося відтворювати всі вихідні дані з проекції.

В експериментах модель синтезувала точні відеокадри, що показують людей, що йдуть, витягуючи інформацію з одновимірних, одновимірних ліній, подібних до тих, що виробляються кутовими камерами. Модель також відновила відеокадри з поодиноких розмитих проекцій цифр, що рухаються по екрану, з популярного набору даних Moving MNIST.

Підказки в пікселях

Робота розпочалася як "крута проблема інверсії", щоб відтворити рух, який викликає розмиття руху у фотографії з довгим експозицією. У пікселях проекції є деякі підказки про високомірне джерело.

Наприклад, цифрові камери, що знімають тривалі експозиції, в основному агрегують фотони протягом певного часу на кожному пікселі. Знімаючи рух об'єкта з часом, камера прийме середнє значення пікселів, що фіксують рух. Потім вони застосовують ці середні значення до відповідних висот та ширин нерухомого зображення, що створює розмиті смуги підпису траєкторії об'єкта. Обчисливши деякі зміни інтенсивності пікселів, теоретично можна відтворити рух.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		30

Як зрозуміли дослідники, ця проблема є актуальною у багатьох областях: рентгенівські промені, наприклад, фіксують інформацію про висоту, ширину та глибину анатомічних структур, але вони використовують аналогічну техніку усереднення пікселів, щоб збільшити глибину на 2D-зображення. Кутові камери фіксують відбиті світлові сигнали навколо прихованої сцени, яка несе двовимірну інформацію про відстань людини від стін та предметів. Техніка усереднення пікселів потім згортає ці дані в одновимірне відео – в основному, вимірювання різної довжини в часі в одному рядку.

Побудована загальна модель, заснована на згортковій нейронній мережі (ЗНМ) – модель машинного навчання, яка стала потужною задачею для обробки зображень – яка фіксує підказки про будь-який втрачений вимір у усереднених пікселях.

Синтезування сигналів

Під час тренінгу подавали ЗНМ тисячам пар проекцій та їх високомірних джерел, званих "сигналами". ЗНМ вивчає піксельні структури в проекціях, які відповідають тим, що подаються в сигналах. Живлення ЗНМ – це структура, яка називається "варіативним автокодером", який оцінює, наскільки добре виходи ЗНМ співпадають з її вхідними даних через деяку статистичну ймовірність. З цього, модель засвоює "простір" усіх можливих сигналів, які могли б створити задану проекцію. Це, по суті, створює тип плану для того, як перейти від проекції до всіх можливих відповідних сигналів.

Коли показуються раніше небачені проекції, модель відзначає піксельні схеми і слідує кресленням всіх можливих сигналів, які могли б створити цю проекцію. Потім він синтезує нові зображення, що поєднують усі дані з проекції та всі дані з сигналу. Це відтворює високомірний сигнал.

В ході одного експерименту зібрали набір даних із 35 відеозаписів з 30 людей, що йдуть у визначеній місцевості. Вони обвалили всі кадри на проекції, які використовували для тренування та тестування моделі. Із набору з шести небачених проекцій модель точно відтворила 24 кадри ходи людини, аж до

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		31

положення її ніг і розміру людини, коли вони йшли до камери або від неї. Модель, здається, дізнається, наприклад, що пікселі, які з часом темніші та ширші, ймовірно, відповідають людині, що йде ближче до камери.

3.2 Розробка структурної схеми

Системи відеоспостереження останнім часом здобувають усе більше широку популярність.

Сучасні лінії відеоконтролю будуються на новітніх технічних засобах, які дозволяють виконувати цифрову обробку одержуваного відеозображення.

Цифрове відеоспостереження є ефективним, оскільки його роботу можна зв'язувати з комп'ютерними системами. Цифровий комплект набагато краще в обслуговуванні й коштує дешевше аналогових пристроїв.

Які елементи входять у комплектацію системи?

Цифрові системи відеоспостереження по комплектації можуть відрізнятися друг від друга. Однозначний комплексний состав лінії контролю встановити не можна.

Побудова подібних комплексів залежить від особливостей охоронюваного об'єкта й переваг замовника.

На сьогоднішній день для створення цифрових ліній відеоконтролю використовують спеціальне відеоустаткування. До складу системи можуть входити:

- IP камери.
- Сервери із установленим програмним забезпеченням системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі, що розроблено в даній роботі.
- Реєстратори.
- Монітори.
- Сполучні кабелі (коаксіальні, крученої пари й ін.).

Цифрові системи відеоспостереження (ЦСВС) включають камери, які здатні виконувати запис, зберігання й передачу цифрового відеозображення.



Рисунок 3.1 – Структурна схема системи

У цей час достатнє поширення одержують способи передачі інформації за допомогою локальної або глобальної мережі (IP спостереження). камери бувають провідними й бездротовими. Сервери сполучають у собі різні програмні й апаратні засоби.

З їхньою допомогою відбувається одержання й вивід аудіо- і відеосигналу. реєстратори виконують функцію оцифровки даних.

Можливості цифрового відеоспостереження

ЦСВС придбала величезну популярність і має ряд можливостей. Серед них варто виділити основні:

- Забезпечення високоякісної відтвореної відеоінформації й відеозапису.
- Високошвидкісний доступ до архіву даних.
- Наявність функції збільшення, оцифровки й масштабування записаних відеокадрів.
- Негайний пошук і перегляд необхідного відеозапису по різних параметрах (номеру камери, даті, часу).
- Можливість інтеграції з іншими охоронними підсистемами.
- Ведення трансляції відеоінформації за допомогою локальних або глобальних мереж.
- Відправлення сигналу про тривогу шляхом передачі текстового повідомлення на мобільний телефон або листи по електронній пошті.
- Наявність функції відправлення збереженого відеозапису на знімний носій.

Транслявання відеоінформації є однієї з основних можливостей сучасних цифрових комплексів. Перегляд відеозапису можливий при знаходженні на вилученій відстані від охоронюваного об'єкта.

На відео – як налаштувати IP систему відеоспостереження своїми руками:

У чому відмінність цифрового відеоустаткування від аналогового?

Аналогове встаткування являє собою технічні засоби, які передають отриману інформацію із джерела. При цьому якість відеоінформації не спотворюється. Цифрова техніка дозволяє проводити обробку зображення й виводити її у двійковому виді.

Головною відмінністю між ними є можливість цифрових пристроїв виконувати збереження інформації на комп'ютері.

Цифрова система відеоспостереження забезпечує вивід зображення, розбитого на дрібні квадрати (пікселі).

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		34

У цей час якість розв'язної здатності відеосигналу настільки поліпшено, що людина не здатна відрізнити цифровий кадр від аналогового.

Комплект відеоспостереження такого формату не має обмежень у своїх можливостях по способах і швидкості передачі даних. Цифровий відеоконтроль відрізняє ряд особливостей:

- Можливість передачі одержуваної інформації в цифровій якості.
- Розширення матричних механізмів.
- Наявність можливості дистанційного керування й налаштування системи.
- Відсутність необхідності в установці додаткового (резервного) джерела електроживлення.

Установка новітніх систем не вимагає додаткового кодування, оскільки відеосигнал оцифровується автоматично. На монітор виводиться високоякісне зображення. Розширення матричних механізмів дозволяє одержувати зображення високої якості й розв'язної здатності у форматі Full HD.

Систему відеоконтролю, засновану на цифровому встаткуванні, можна дистанційно налаштувати й управляти її роботою на віддаленій відстані.

Також пристрій можливо вдосконалювати або побудувати на базі вже існуючої лінії відеоконтролю на об'єкті.

Основні аспекти вибору цифрових відеокамер

Для вибору якісної й працездатної відеокамери з метою створення цифрової лінії контролю, необхідно звертати увагу на ряд параметрів:

- Чутливість відеопристрою. Ця технічна характеристика вимірюється в спеціальних одиницях – люксах. Кількість люксів визначає можливість і якість роботи камери при недостатнім висвітленні або його відсутності.
- Розв'язна здатність зображення. Цей параметр визначається телевізійними лініями. Чим вище розв'язна здатність, тим чіткіше і якісніше зображення.

– Матричний розмір. Без спеціальних знань визначити необхідні параметри матриці складно. Тому для звичайного користувача матричний параметр потрібно визначати за принципом: більше – краще.

– Фокусна відстань установлюється залежно від інженерних і конструкторських особливостей охоронюваного периметра.

– Захищеність пристрою від зовнішніх впливів (пилу, вологи й ін.).

Переваги й недоліки одержання оцифрованого відеозображення

До переваг використання цифрового встаткування для створення лінії відеоконтролю відносяться:

– Високоякісне зображення, одержуване від пристрою відеозапису.
– Належний рівень захищеності інформації від незаконного доступу або перегляду.

– Способи побудови системи відеоспостереження.
– Спостереження з використанням персональних комп'ютерів.
– Можливість вилученого доступу й перегляду відеоінформації.
– Розсилання відеоінформації (окремих кадрів) по електронній пошті або за допомогою повідомлень на мобільний телефон.

– Можливість проведення технічного огляду й обслуговування самотужки (за умови володіння мінімумом спеціальних знань).

До недоліків таких ліній відеоконтролю можна віднести:

– Високу вартість комплектувального устаткування.
– Зменшений обсяг архіву даних у зв'язку із записом високоякісної інформації.
– Установлення обмеження на канал зв'язку при передачі даних.
– Трудомісткий процес налаштування конфігурацій кожної відеокамери.

Практичні поради для кращих результатів розмиття

Почніть з найкращого можливого оригінального зображення

Навіть найсучасніші інструменти штучного інтелекту для усунення розмиття зображень дають кращі результати, коли якість оригінального

зображення досить хороша. Зображення з достатнім освітленням, мінімальним шумом і вищою роздільною здатністю надають штучному інтелекту більше візуальної інформації для роботи. Початок з чистішого джерела допомагає досягти чіткіших і природніших результатів.

Уникайте артефактів надмірного загострення

Надмірне покращення може призвести до неприродних текстур, ореолів або різких країв. Часто краще використовувати помірні налаштування та зберігати реалістичний вигляд, ніж надто сильно підвищувати різкість. Незначні покращення зазвичай призводять до візуально приємніших та професійніших результатів.

Коли розмиття за допомогою штучного інтелекту не працює належним чином

Інструменти для усунення розмиття на основі штучного інтелекту мають обмеження, особливо коли зображення надзвичайно розмите або не має важливих візуальних даних. У випадках, коли деталі повністю втрачені, повне відновлення може бути неможливим. Розуміння цих обмежень допомагає встановити реалістичні очікування щодо кінцевого результату.

Поєднання інструментів розмиття з інструментами масштабування

Для найкращих загальних результатів обробку зображень для усунення розмиття можна поєднати з масштабуванням за допомогою штучного інтелекту, коли потрібна вища роздільна здатність. Спочатку усунення розмиття відновлює чіткість, а потім масштабування покращує розмір зображення та деталізацію. Це поєднання особливо добре працює для старих фотографій або зображень з низькою роздільною здатністю.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Функціонально система складається з наступних частин:

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		37

1. База даних журналювання розпізнаних образів. У цю баз даних заносяться усі дані, які відносяться до розпізнаних об'єктів, будь то людина, або автомобіль.

2. База даних образів облич. У цій базі даних зберігаються фотографії усіх працівників установи та відвідувачів, з виділенням характерних точок, для кожного обличчя, за якими можливо ідентифікувати або працівника компанії, або відвідувача.

3. База даних образів цифр та букв на номерах автомобілів. У цій базі даних зберігаються образи усіх цифр та букв, з яких можуть складатися номери автомобілів, а також номери усіх автомобілів, які перетинали кордон приміщення установи, який встаткований відеокамерами спостереження.

4. Відеокамера спостереження, з якої поступає інформація систему розпізнання образів.

5. Система розпізнання образів та відновлення деталей на розмитих зображеннях за допомогою нейронної мережі, яка складається з наступних блоків:

– Блок читання картини з відеокамери. Він призначений для читання картинок з відеокамери й подання даних на блок аналізу та виділення ознак та відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

– Блок аналізу та виділення ознак та відновлення деталей на розмитих зображеннях за допомогою нейронної мережі. Він є основою системи, й за допомогою нижчеописаного алгоритму проводить розпізнання осіб, та машин, які перетнули межу території установи.

– Блок класифікації та опису об'єкта. Він дозволяє, виходячи з даних, отриманих від блоку аналізу та виділення ознак та відновлення деталей на розмитих зображеннях за допомогою нейронної мережі, розподілити куди заносити отримані дані, у поля бази даних, які відповідають за осіб, або у поля бази даних, які відповідають за машини.

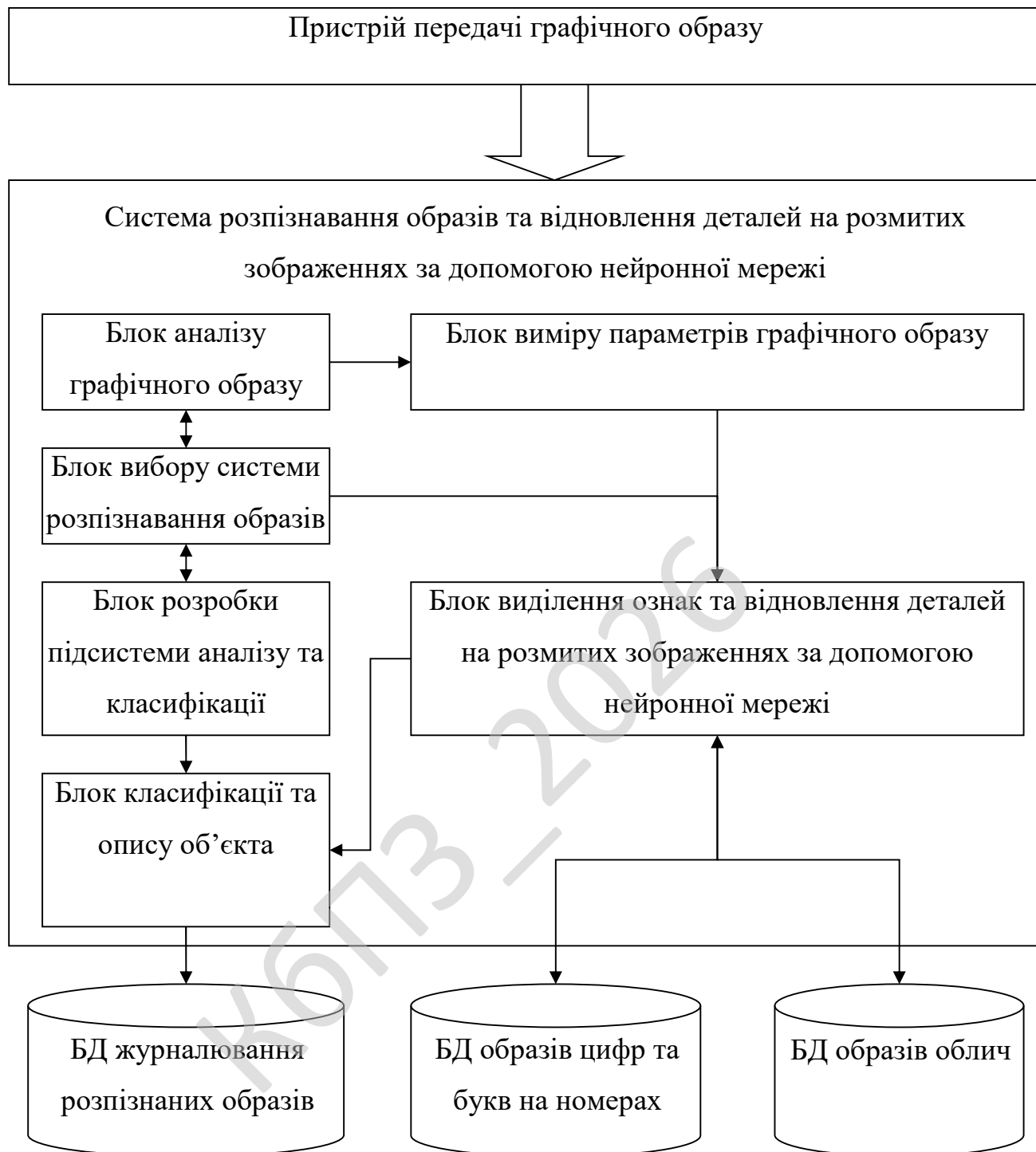


Рисунок 3.2 – Функціональна схема системи

Однак завдання розпізнавання, саме по собі, припускає інтелектуальну обробку отриманої інформації, що представляє певні складності. Безсумнівно, у завданні розпізнавання символів (розпізнавання тексту, автомобільних номерів) досягнуті величезні успіхи. Але, проте, яких-небудь універсальних методів

обробки зображення, порівнянних по продуктивності і якості розпізнавання з людськими здатностями, немає. Наприклад, у завданнях, які відносяться перед експертними системами, потрібно більше глибокий інтелектуальний аналіз і високу швидкодія, цими ж властивостями повинні володіти роботизовані системи обслуговування. Тому обробка зображення в завданні розпізнавання є однією із центральних проблем.

Згорткові нейронні мережі (ЗНМ)

Згорткові нейронні мережі (ЗНМ). Звучить як дивне сполучення біології й математики з домішкою інформатики, але як би воно не звучало, ці мережі – одні із самих впливових інновацій в області комп'ютерного зору. Сьогодні глибинне навчання лежить в основі послуг багатьох компаній: Facebook використовує нейронні мережі для алгоритмів автоматичного проставлення тегів, Google – для пошуку серед фотографій користувача, Amazon – для генерації рекомендацій товарів, Pinterest – для персоналізації домашньої сторінки користувача, а Instagram – для пошукової інфраструктури.

Але класичний, і, можливо, самий популярний варіант використання мереж це обробка зображень. Давайте подивимося, як ЗНМ використовуються для класифікації зображень.

Завдання

Завдання класифікації зображень – це прийом початкового зображення й вивід його класу (кішка, собака й т.д.) або групи ймовірних класів, що найкраще характеризує зображення. Для людей це один з перших навичок, що вони починають освоювати з народження.

Ми опановуємо ним природно, без зусиль, ставши дорослими. Навіть не думаючи, ми можемо швидко й легко розпізнати простір, що нас оточує, разом з об'єктами. Коли ми бачимо зображення або просто дивимося на те, що відбувається навколо, найчастіше ми можемо відразу охарактеризувати місце дії, дати кожному об'єкту ярлик, і все це відбувається несвідомо, непомітно для розуму. Ці навички швидкого розпізнавання шаблонів, узагальнення вже

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

отриманих знань, і адаптації до різного відбитій на фото обстановці не доступні нашим електронним приятелям.

Вводи й виводи

Коли комп'ютер бачить зображення (приймає дані на вхід), він бачить масив пікселів. Залежно від дозволу й розміру зображення, наприклад, розмір масиву може бути 32х32х3 (де 3 – це значення каналів RGB). Щоб було більш зрозуміло, давайте представимо, у нас є кольорове зображення у форматі JPG, і його розмір 480х480. Відповідний масив буде 480х480х3. Кожному із цих чисел привласнюється значення від 0 до 255, що описує інтенсивність пікселя в цій крапці. Ці цифри, залишаючись безглуздими для нас, коли ми визначаємо що на зображенні, є єдиними вступними даними, доступними комп'ютеру. Ідея в тім, що ви даєте комп'ютеру цю матрицю, а він виводить числа, які описують імовірність класу зображення (0.80 для кішки, 0.15 для собаки, 0.05 для птаха й т.д.).

Чого ми хочемо від комп'ютера

Тепер, коли ми визначили завдання, ввід й вивід, давайте думати про те, як підбиратися до рішення. Ми хочемо, щоб комп'ютер міг розрізняти всі дані йому зображення й розпізнавати унікальні особливості, які роблять собаку собакою, а кішку кішкою. У нас і цей процес відбувається підсвідомо. Коли ми дивимося на зображення собаки, ми можемо віднести його до конкретного класу, якщо в зображення є характерні риси, які можна ідентифікувати, такі як лабеті або чотири ноги. Аналогічним образом комп'ютер може виконувати класифікацію зображень через пошук характеристик базового рівня, наприклад границь і скривлень, а потім за допомогою побудови більше абстрактних концепцій через групи згорткових шарів. Це загальний опис того, що роблять ЗНМ. Тепер перейдемо до специфіки.

Біологічні зв'язки

На початку небагато історії. Коли ви вперше почули термін згорткові нейронні мережі, можливо подумали про щось пов'язаному з нейронауками або

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		41

біологією, і почасти були праві. У якімсь змісті. ЗНМ – це дійсно прототип зорової кори мозку. Зорова кора має невеликі ділянки кліток, які чутливі до конкретних областей поля зору. Цю ідею детально розглянули за допомогою приголомшливого експерименту Хьюбел і Визель в 1962 році, у якому показали, що окремі мозкові нервові клітки реагували (або активувалися) тільки при візуальному сприйнятті границь певної орієнтації. Наприклад, деякі нейрони активувалися, коли сприймали вертикальні границі, а деякі – горизонтальні або діагональні. Хьюбел і Визель з'ясували, що всі ці нейрони зосереджені у вигляді стрижневої архітектури й разом формують візуальне сприйняття. Цю ідею спеціалізованих компонентів усередині системи, які вирішують конкретні завдання (як клітки зорової кори, які шукають специфічні характеристики) і використовують машини, і ця ідея – основа ЗНМ.

Структура

Повернемося до специфіки. Що конкретно роблять ЗНМ? Береться зображення, пропускається через серію згорткових, нелінійних шарів, шарів об'єднання й повнозв'язних шарів, і генерується вивід. Як ми вже говорили, виводом може бути клас або ймовірність класів, які найкраще описують зображення. Складний момент – розуміння того, що робить кожний із цих шарів. Так що давайте перейдемо до найважливішому.

Перший шар – математична частина

Перший шар у ЗНМ завжди згортковий. Ви ж пам'єте, який ввід в цього згорткового шару? Як уже говорилося раніше, вступне зображення – це матриця $32 \times 32 \times 3$ з піксельними значеннями. Легше всього зрозуміти, що таке згортковий шар, якщо представити його у вигляді ліхтарика, що світить на верхню ліву частину зображення. Допустимо світло, що випромінює цей ліхтарик, покриває площа 5×5 . А тепер давайте представимо, що ліхтарик рухається по всіх областях вступного зображення. У термінах комп'ютерного навчання цей ліхтарик називається фільтром (іноді нейроном або ядром), а області, на які він світить, називаються рецептивним полем (полем сприйняття). Тобто наш фільтр – це

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		42

матриця (таку матрицю ще називають матрицею ваг або матрицею параметрів). Помітьте, що глибина у фільтра повинна бути такий же, як і глибина вступного зображення (тоді є гарантія математичної вірності), і розміри цього фільтра – $5 \times 5 \times 3$. Тепер давайте за приклад візьмемо позицію, у якій перебуває фільтр. Нехай це буде лівий верхній кут. Оскільки фільтр робить згортку, тобто пересувається по вступному зображенню, він множить значення фільтра на вихідні значення пікселів зображення (заелементне множення). Всі ці множення підсумуються (усього 75 множень). І в підсумку виходить одне число. Помніте, воно просто символізує знаходження фільтра у верхньому лівому куті зображення. Тепер повторимо цей процес у кожній позиції. (Наступний крок – переміщення фільтра вправо на одиницю, потім ще на одиницю вправо й так далі). Кожна унікальна позиція уведеного зображення робить число. Після проходження фільтра по всіх позиціях виходить матриця $28 \times 28 \times 1$, що називають функцією активації або картою ознак. Матриця 28×28 виходить тому, що є 784 різні позиції, які можуть пройти через фільтр 5×5 зображення 32×32 . Ці 784 числа перетворюються в матрицю 28×28 .

Деякі зображення, у тому числі те, що ви бачите нижче, узяті із книги "Нейронні мережі й глибоке навчання" Майкла Нільсена ("Neural Networks і Deep Learning", by Michael Nielsen).

Допустимо, тепер ми використовуємо два $5 \times 5 \times 3$ фільтри замість одного. Тоді вихідним значенням буде $28 \times 28 \times 2$.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		43

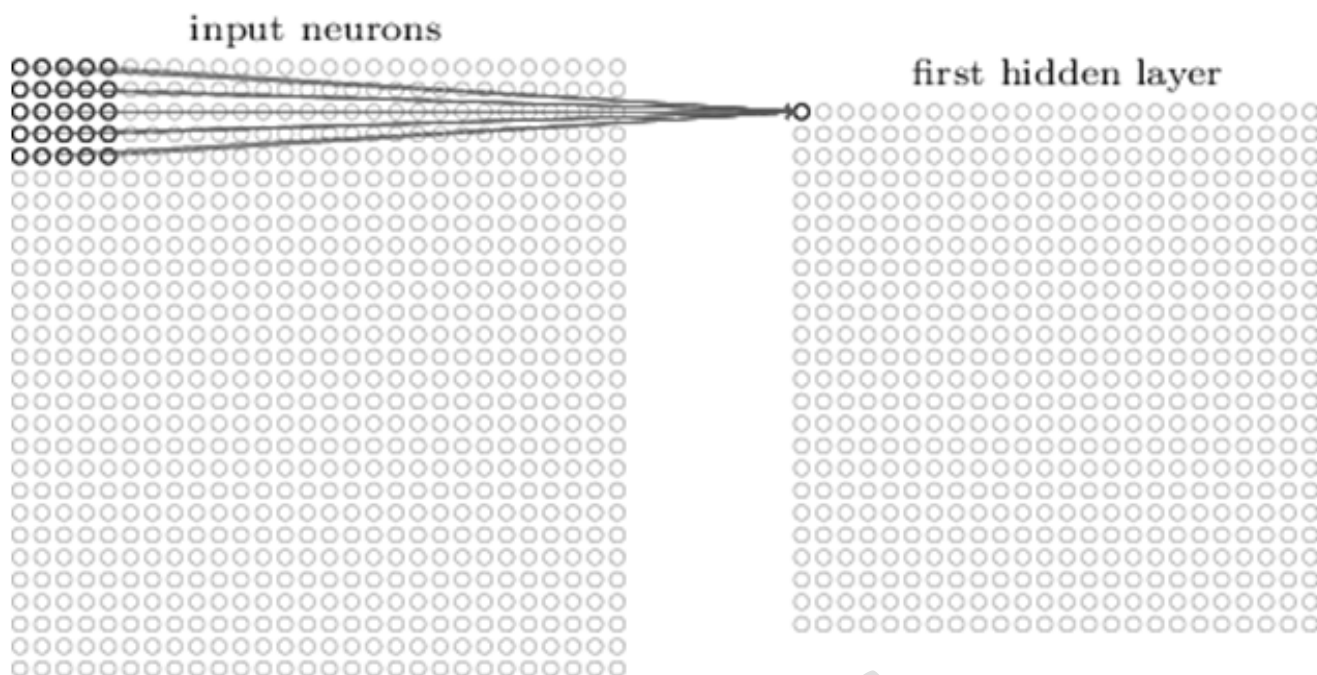


Рисунок 3.3 – Візуалізація фільтра 5x5, що згортається навколо вхідного обсягу й створює карту активації

Перший шар

Давайте поговоримо про те, що ця згортка насправді робить на високому рівні. Кожний фільтр можна розглядати як ідентифікатор властивості. Коли я говорю властивість, я маю на увазі прямі границі, прості кольори й криві. Подумайте про найпростіші характеристики, які мають всі зображення в загальному. Скажемо, наш перший фільтр 7x7x3, і він буде детектором кривих. (Зараз давайте ігнорувати той факт, що у фільтра глибина 3, і розглянемо тільки верхній шар фільтра й зображення, для простоти). У фільтра піксельна структура, у якій чисельні значення вище уздовж області, що визначає форму кривій (пам'ятайте, фільтри, про які ми говоримо, це просто числа).

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0



Рисунок 3.4 – Подання крапок у фільтрі й візуалізація фільтра який визначає криву

Повернемося до математичної візуалізації. Коли в нас у лівому верхньому куті вступного зображення є фільтр, він робить множення значень фільтра на значення пікселів цієї області. Давайте розглянемо приклад зображення, якому ми хочемо привласнити клас, і встановимо фільтр у верхньому лівому куті.

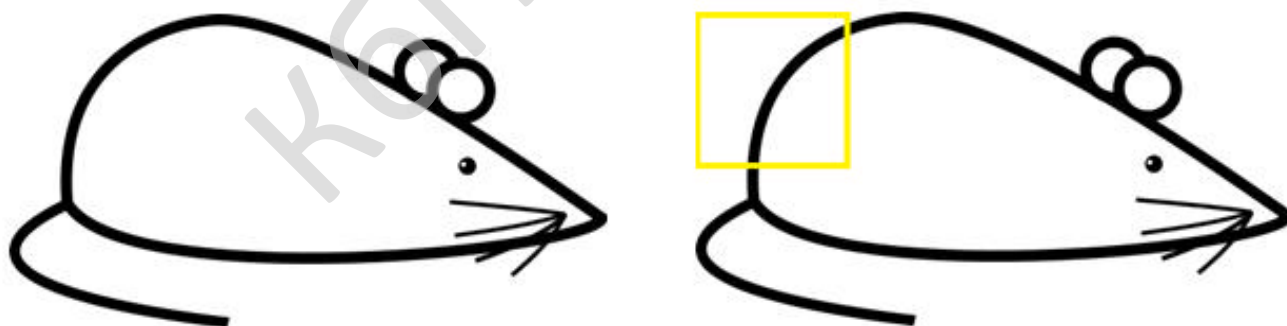


Рисунок 3.5 – Оригінальне зображення й візуалізація фільтра зображення

Помнете, всі що нам потрібно, це помножити значення фільтра на вихідні значення пікселів зображення.



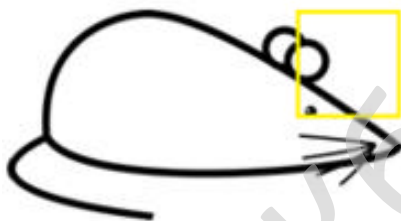
0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0

Рисунок 3.6 – Візуалізація рецептивного поля у вигляді піксельного подання рецептивного поля перемноженого на піксельне подання фільтра

По суті, якщо на вступному зображенні є форма, загалом схожа на криву, що представляє цей фільтр, і всі помножені значення підсумуються, то результатом буде велике значення! Тепер давайте подивимося, що відбудеться, коли ми перемістимо фільтр.



0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0

Рисунок 3.7 – Візуалізація фільтра зображення у вигляді піксельного подання рецептивного поля перемноженого на піксельне подання фільтра

Значення набагато нижче! Це тому, що в новій області зображення немає нічого, що фільтр визначення кривій міг засікти. Помнете, що вивід цього згорткового шару – карта властивостей. У найпростішому випадку, при наявності одного фільтра згортки (і якщо цей фільтр – детектор кривій), карта властивостей покаже області, у яких більше ймовірності наявності кривих. У цьому прикладі в

лівому верхньому куті значення нашої $28 \times 28 \times 1$ карти властивостей буде 6600. Це високе значення показує, що, можливо, щось схоже на криву присутня на зображенні, і така ймовірність активувала фільтр. У правому верхньому куті значення в карти властивостей буде 0, тому що на картинці не було нічого, що могло активувати фільтр (простіше говорячи, у цій області не було кривій). Помнете, що це тільки для одного фільтра. Це фільтр, що виявляє лінії з вигином назовні. Можуть бути інші фільтри для ліній, вигнутих усередину або просто прямих. Чим більше фільтрів, тим більше глибина карти властивостей, і тем більше інформації ми маємо про вступну картинку.

Фільтр, про яке я розповів у цьому розділі, спрощений для спрощення математики згортання. На рисунку нижче видні приклади фактичних візуалізацій фільтрів першого згорткового шару навченої мережі. Але ідея тут та ж. Фільтри на першому шарі згортаються навколо вступного зображення й "активуються" (або роблять більші значення), коли специфічна риса, що вони шукають, є у вступному зображенні.

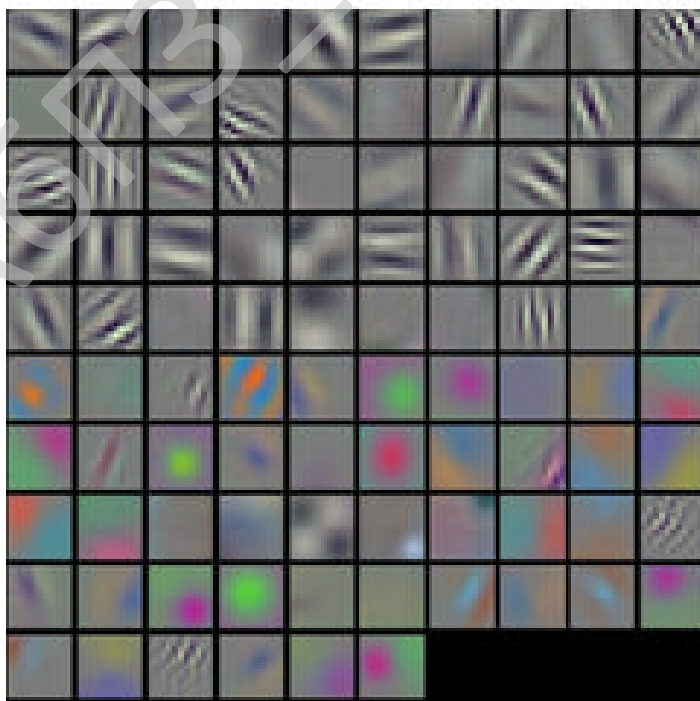


Рисунок 3.8 – Візуалізація фільтрів

Йдемо глибше по мережі

Сьогодні в традиційній згортковій нейронній мережній архітектурі існують і інші шари, які перемешуюються зі згортковими. Я дуже рекомендую тим, хто цікавиться темою, почитати про ці шари, щоб зрозуміти їхню функціональність і ефекти. Класична архітектура ЗНМ буде виглядати так:

Input->Conv->ReLU->Conv->ReLU->Pool->ReLU->Conv->ReLU->Pool->Fully Connected

Останній шар, хоч і перебуває наприкінці, один з важливих – ми перейдемо до нього пізніше. Давайте підсумуємо те, у чому ми вже розібралися. Ми говорили про те, що вміють визначати фільтри першого згорткового шару. Вони виявляють властивості базового рівня, такі як границі й криві. Як можна собі представити, щоб припустити який тип об'єкта зображений на картинці, нам потрібна мережа, здатна розпізнавати властивості більше високого рівня, як наприклад руки, лабеті або вуха. Так що давайте подумаємо, як виглядає вихідний результат мережі після першого згорткового шару. Його розмір 28x28x3 (за умови, що ми використовуємо три фільтри 5x5x3). Коли картинка проходить через один згортковий шар, вихід першого шару стає вступним значенням 2-го шаруючи. Тепер це небагато складніше візуалізувати. Коли ми говорили про перший шар, вводом були тільки дані вихідного зображення. Але коли ми перейшли до 2-му шару, вступним значенням для нього стала одна або кілька карт властивостей – результат обробки попереднім шаром. Кожний набір вступних даних описує позиції, де на вихідному зображенні зустрічаються певні базові ознаки.

Тепер, коли ви застосовуєте набір фільтрів поверх цього (пропускаєте картинку через другий згортковий шар), на виході будуть активовані фільтри, які представляють властивості більше високого рівня. Типами цих властивостей можуть бути півкілця (комбінація прямої границі з вигином) або квадратів (сполучення декількох прямих ребер). Чим більше згорткових шарів проходить зображення й чим далі воно рухається по мережі, тим більше складні характеристики виводяться в картах активації. Наприкінці мережі можуть бути

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		48

фільтри, які активуються при наявності рукописного тексту на зображенні, при наявності рожевих об'єктів і т.д.

Ще один цікавий момент. Коли ви рухаєтеся вглиб мережі, фільтри працюють із все більшим полем сприйняття, а виходить, вони в стані обробляти інформацію з більшої площі первісного зображення (простими словами, вони краще пристосовуються до обробки більшої області піксельного простору).

Повнозв'язні шари

Тепер, коли ми можемо виявити високорівневі властивості, саме круте – це прикріплення повнозв'язного шаруючи наприкінці мережі. Цей шар бере вступні дані й виводить N -просторовий вектор, де N – число класів, з яких програма вибирає потрібний. Наприклад, якщо ви хочете програму по розпізнаванню цифр, в N буде значення 10, тому що цифр 10. Кожне число в цьому N -просторовому векторі являє собою ймовірність конкретного класу. Наприклад, якщо результуючий вектор для програми розпізнавання цифр це $[0,1 \ 0,1 \ 0,75 \ 0 \ 0 \ 0 \ 0 \ 0,05]$, значить існує 10% імовірність, що на зображенні "1", 10% імовірність, що на зображення "2", 75% імовірність – "3", і 5% імовірність – "9" (звичайно, є й інші способи представити вивід).

Спосіб, за допомогою якого працює повнозв'язний шар – це звертання до виходу попереднього шару (який, як ми пам'ятаємо, повинен виводити високорівневі карти властивостей) і визначення властивостей, які більше пов'язані з певним класом. Наприклад, якщо програма пророкує, що на якимсь зображенні собака, у карт властивостей, які відбивають високорівневі характеристики, такі як лабеті або 4 ноги, повинні бути високі значення. Точно так само, якщо програма розпізнає, що на зображенні птах – у неї будуть високі значення в картах властивостей, представлених високорівневими характеристиками начебто крил або дзьоба. Повнозв'язний шар дивиться на те, що функції високого рівня сильно пов'язані з певним класом і мають певні ваги, так що, коли ви обчислюєте добутки ваг з попереднім шаром, то одержуєте правильні ймовірності для різних класів.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		49

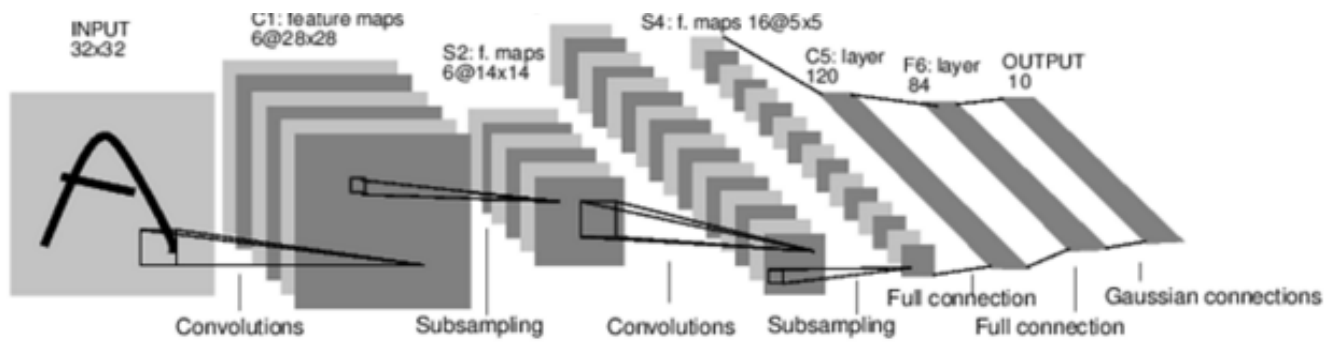


Рисунок 3.9 – Повна згорткова нейронна мережа

Навчання

Це один з аспектів нейронних мереж, про яке я спеціально дотепер не згадував. Імовірно, це найважливіша частина. Можливо, у вас з'явилася безліч питань. Звідки фільтри першого згорткового шару знають, що потрібно шукати границі й криві? Звідки повнозв'язний шар знає, що шукає карта властивостей? Звідки фільтри кожного шару знають, які зберігати значення? Спосіб, яким комп'ютер здатний коректувати значення фільтра (або ваг) – це навчальний процес, що називають методом зворотного поширення помилки.

Перед тим, як перейти до пояснення цього методу, поговоримо про те, що нейронної мережі потрібно для роботи. Коли ми народжуємося, наші голови порожні. Ми не розуміємо як розпізнати кішку, собаку або птаха. Ситуація зі ЗНМ схожа: до моменту побудови мережі, ваги або значення фільтра випадкові. Фільтри не вміють шукати границі й криві. Фільтри верхніх шарів не вміють шукати лабети й дзьоби. Коли ми стаємо старше, батьки й учителі показують нам різні картинки й зображення й привласнюють їм відповідні ярлики. Та ж ідея показу картинки й присвоювання ярлика використовується в навчальному процесі, що проходить ЗНМ. Давайте представимо, що в нас є набір навчальних картинок, у якому тисячі зображень собак, кішок і птахів. У кожного зображення є ярлик з назвою тварини.

Метод зворотного поширення помилки можна розділити на 4 окремих блоки: пряме поширення, функцію втрати, зворотне поширення й відновлення

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

ваги. Під час прямого поширення, береться тренувальне зображення – як пам'ятаєте, це матриця $32 \times 32 \times 3$ – і пропускається через всю мережу. У першому навчальному прикладі, тому що всі ваги або значення фільтра були ініціалізовані випадковим образом, вихідним значенням буде щось начебто $[.1.1.1.1.1.1.1.1.1]$, тобто таке значення, що не дасть переваги якомусь певному числу. Мережа з такими вагами не може знайти властивості базового рівня й не може обґрунтовано визначити клас зображення. Це веде до функції втрати. Помнете, то, що ми використовуємо зараз – це навчальні дані. У таких даних є й зображення і ярлик. Допустимо, перше навчальне зображення – це цифра 3. Ярликом зображення буде $[0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$. Функція втрати може бути виражена по-різному, але часто використовується СКП (середньоквадратична помилка), це $1/2$ помножити на (реальність – пророкування) у квадраті.

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

Приймемо це значення за змінну L . Як ви догадуєтеся, втрата буде дуже високою для перших двох навчальних зображень. Тепер давайте подумаємо про цьому інтуїтивно. Ми хочемо домогтися того, щоб зпрогнозований ярлик (вивід згорткового шару) був таким же, як ярлик навчального зображення (це значить, що мережа зробила вірне припущення). Щоб такого домогтися, нам потрібно звести до мінімуму кількість втрат, що у нас є. Візуалізуючи це як завдання оптимізації з математичного аналізу, нам потрібно з'ясувати, які входи (ваги, у нашій випадку) самим безпосереднім образом сприяли втратам (або помилкам) мережі.

Один зі способів візуалізувати ідею мінімізації втрати – це тривимірний графік, де ваги нейронної мережі (очевидно їх більше, ніж 2, але отут приклад спрощений) це незалежні змінні, а залежна змінна – це втрата. Завдання мінімізації втрат – відрегулювати ваги так, щоб знизити втрату. Візуально нам потрібно наблизитися до самої нижньої крапки чашеподібного об'єкта. Щоб домогтися цього, потрібно знайти похідну втрати (у рамках намальованого

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		51

графіка – розрахувати кутовий коефіцієнт у кожному напрямку) з урахуванням ваг.

Це математичний еквівалент d/d , де W – ваги певного шару. Тепер нам потрібно виконати **зворотне поширення** через мережу, що визначає, які ваги вплинули на втрати, і знайти способи, як їх налаштувати, щоб зменшити втрати. Після того, як ми обчислимо похідну, перейдемо до останнього етапу – відновленню ваг. Візьмемо всі фільтрові ваги й оновимо їх так, щоб вони мінялися в напрямку градієнта.

$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
 w_i – Initial Weight
 η = Learning Rate

Швидкість навчання – це параметр, що вибирається програмістом. Висока швидкість навчання означає, що у відновленнях ваги робилися більші кроки, тому зразку може знадобитися менше часу, щоб набрати оптимальний набір ваг. Але занадто висока швидкість навчання може привести до дуже великого й недостатньо точним скачкам, які перешкоджають досягненню оптимальних показників.

Процес прямого поширення, функцію втрати, зворотне поширення й відновлення ваг, звичайно називають одним періодом дискретизації (або epoch – епохою). Програма буде повторювати цей процес фіксована кількість періодів для кожного тренувального зображення. Після того, як відновлення параметрів завершиться на останньому тренувальному зразку, мережа в теорії повинна бути досить добре навчена й ваги шарів настроєні правильно.

Тестування

Нарешті, щоб побачити, чи працює ЗНМ, ми беремо інший набір зображень і ярликів і пропускаємо зображення через мережу. Порівнюємо виходи з реальністю й дивимося, чи працює наша мережа.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.10. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

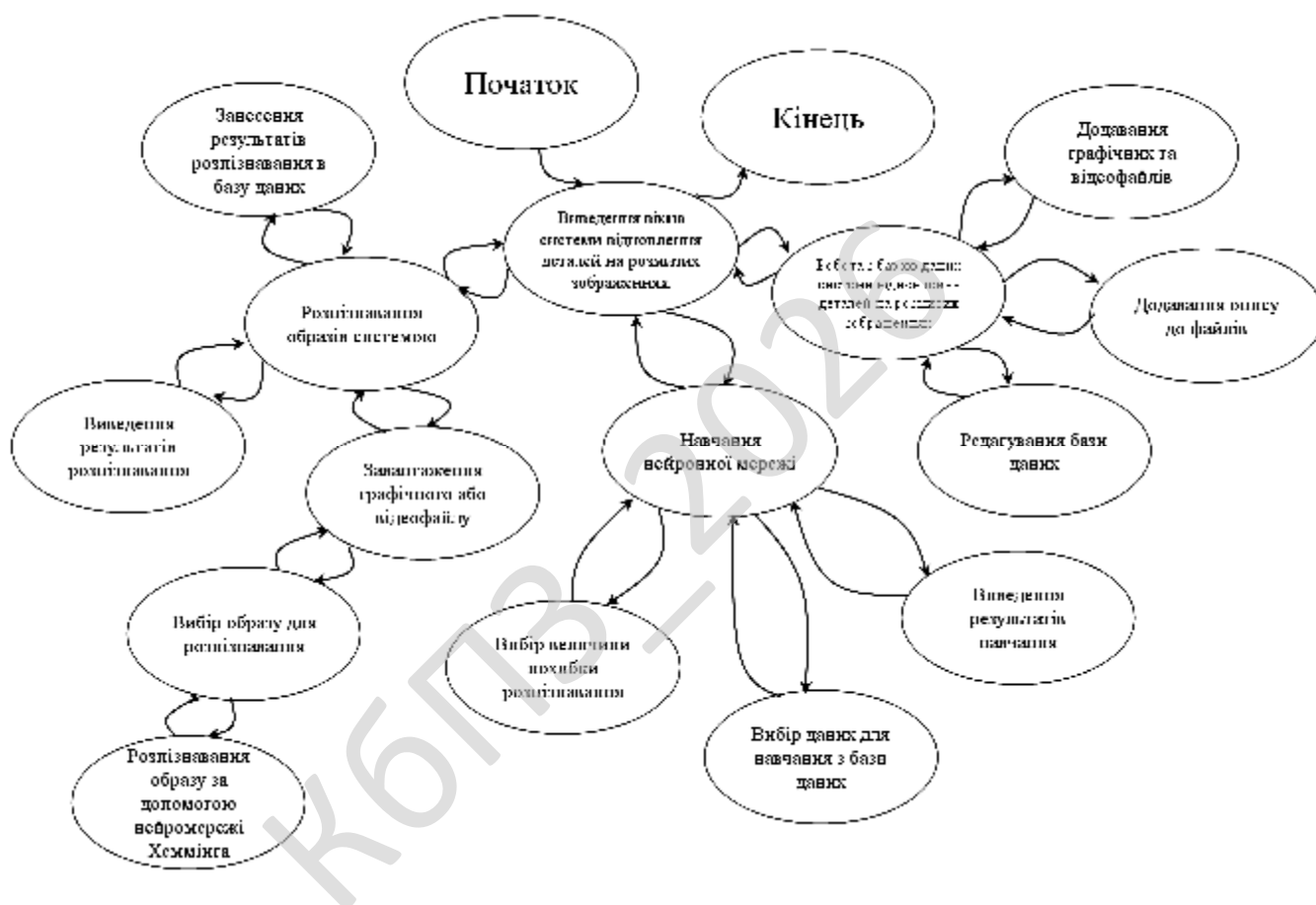


Рисунок 3.10 – Діаграма взаємодії процесів

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходить взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

- Зовнішні по відношенню до системи сутності.
- Потоки даних між елементами трьох попередніх типів.
- Процеси які являють собою трансформацію даних в рамках описуваної системи.
- Сховища даних (репозиторії).

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію бакалаврської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

Було створено блок-схеми роботи системи. Перед їх розглядом необхідно провести роз'яснення який саме тип блок-схем використовується. Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		55

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач. Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції. Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елемента.

Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елемента. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується.

Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення).

Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи). З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми.

Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		57

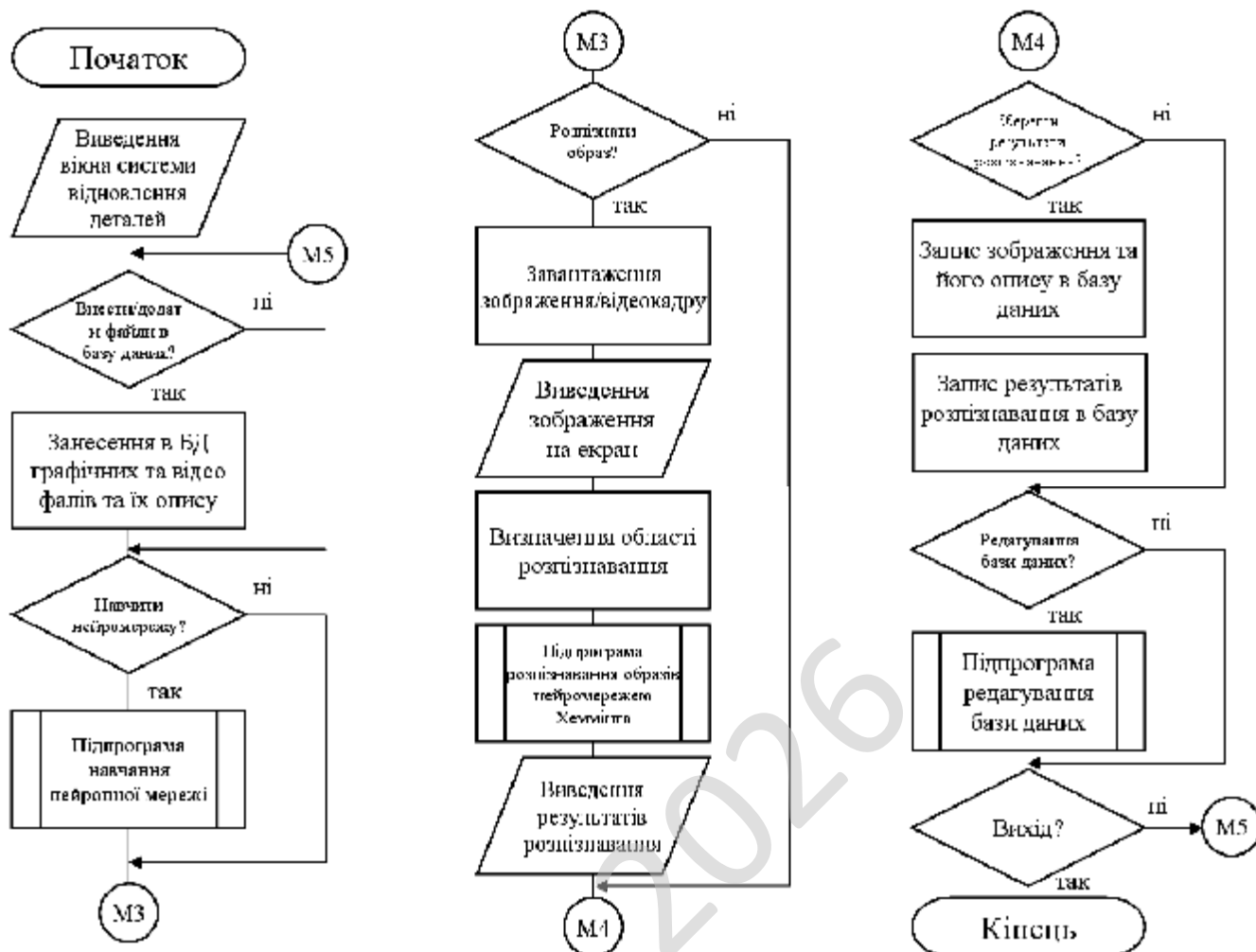


Рисунок 4.1 – Блок-схема основної програми

Було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання

діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

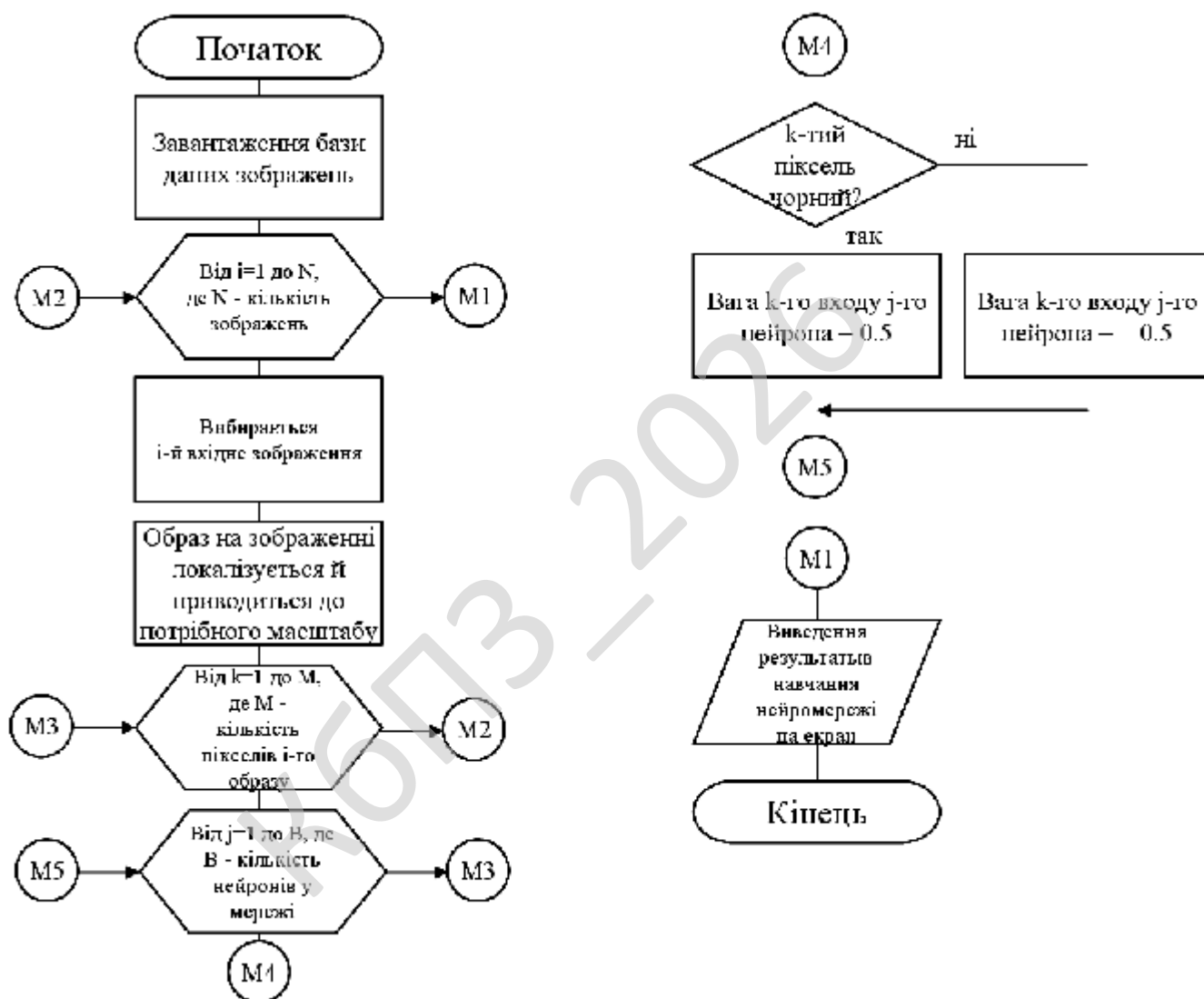


Рисунок 4.2 – Блок-схема роботи підпрограми

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

- асоціації (association relationship);
- включення (include relationship);
- розширення (extend relationship);
- узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме — за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посылатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації

з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортання.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок

діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

SQLite – полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92.

Сирцевий код SQLite поширюється як суспільне надбання (public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		64

З 2018р SQLite, як й JSON та CSV, рекомендований Бібліотекою Конгресу США формат зберігання структурованого набору даних. У 2005 році проект отримав нагороду Google-O'Reilly Open Source Awards.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми.

Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму.

SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок.

Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази.

Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки.

Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті постачання йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки.

Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		65

Завдяки архітектурі рушія можливо використовувати SQLite як на вбудовуваних (embedded) системах, так і на виділених машинах з гігабайтними масивами даних.

Особливості SQLite:

1. Транзакції атомарні, послідовні, ізольовані, і міцні (ACID) навіть після збоїв системи і збоїв живлення.
2. Встановлення без конфігурації – не потребує ані установки, ані адміністрування.
3. Реалізує значну частину стандарту SQL92.
4. База даних зберігається в одному крос-платформовому файлі на диску.
5. Підтримка терабайтних розмірів баз даних і гігабайтного розміру рядків і BLOBів.
6. Малий розмір коду: менше ніж 350KB повністю налаштований, і менш 200KB з опущеними додатковими функціями.
7. Швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій.
8. Простий, легкий у використанні API.
9. Написана в ANSI C, включена прив'язка до TCL; доступні також прив'язки для десятків інших мов.
10. Добре прокоментований сирцевий код зі 100% тестовий покриттям гілок.
11. Доступний як єдиний файл сирцевого коду на ANSI C, який можна легко вставити в інший проект.
12. Автономність: немає зовнішніх залежностей.
13. Крос-платформовість: з коробки підтримується Unix (Linux і Mac OS X), OS/2, Windows (Win32 і WinCE). Легко переноситься на інші системи.
14. Сирці перебувають в суспільному надбанні.
15. Поставляється з автономним клієнтом інтерфейсу командного рядка, який може бути використаний для управління базами даних SQLite.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		66

Інструменти створення та обслуговування БД

Створення та обслуговування БД можуть здійснюватись через текстову консоль SQL-командами або через спеціальні інструменти, у тому числі – з графічним інтерфейсом користувача.

Технології, що підтримують SQLite та мови програмування

Сама бібліотека SQLite написана мовою C. Проте є реалізація бібліотеки на JavaScript `sql.js`, яка дозволяє обробляти файли БД безпосередньо в браузері.

Для інших мов програмування розроблено механізм підключення й роботи з БД через цю бібліотеку: C++, Java, Python, Perl, PHP, Ruby, Haskell, Scheme, Smalltalk, Lua тощо. Засоби для роботи з Tcl включені в комплект постачання SQLite. Повний список наявних засобів можна знайти на сторінці проекту.

Web-інструментарії

У ряді інструментаріїв присутня можливість використання SQLite як бази даних, наприклад: Django; Java; PHP; Ruby on Rails; Trac; Symfony; Parser.

Багато програм підтримують SQLite як формат зберігання даних, зокрема:

1. AmaroK – може використовувати бази даних SQLite як сховище музичної колекції.
2. Gajim – SQLite використовується для зберігання історії контактів.
3. Songbird (як застосунок, заснований на XULRunner 1.9)
4. Banshee.
5. F-Spot.
6. Платформа XUL на рушії Gecko 1.9, XULRunner 1.9 і, потенційно, всі застосунки, засновані на цій платформі, у тому числі й Firefox починаючи з версії 3.0.
7. Google Chrome.
8. Google Gears.
9. Mendeley – менеджер pdf-документів, академічний засіб для дослідження (реалізується desktop & web).

10. Zotero – менеджер інформації, бібліографічний менеджер, додаток Firefox.

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

- Ведення декількох проектів.
- Гнучка система доступу з використанням ролей.
- Система відстеження помилок.
- Діаграми Ганта та календар.
- Ведення новин проекту, документів та управління файлами.
- Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.
- Власна Wiki для кожного проекту.
- Форуми для кожного проекту.
- Облік часових витрат.
- Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.
- Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).
- Створення записів про помилки на основі отриманих листів
- Підтримка LDAP автентифікації.
- Можливість самореєстрації нових користувачів.
- Багатомовний інтерфейс (у тому числі українська мова).
- Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі.

Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		69

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

- номер (ідентифікатор) дефекту;
- хто повідомив про дефект;
- дата і час виявлення дефекту;
- версія продукту, в якій виявлено дефект;
- серйозність (критичність) дефекту та пріоритет рішення;
- опис кроків для відтворення дефекту (неправильної поведінки програми);
- відповідальний за усунення дефекту;
- обговорення можливих рішень та їх наслідків;
- поточний стан виправлення дефекту;
- версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		70

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.
2. Призначений – призначений відповідальний за виправлення дефекту.
3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

- Виправлено (виправлення включені у версію таку-то).
- Дубль (повторює дефект, що вже знаходиться в роботі).
- Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

- «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		71

Гнучка́ розробка програмного забезпечення (Agile software development, agile-методи) – клас методологій розробки програмного забезпечення, що базується на ітеративній розробці, в якій вимоги та розв'язки еволюціонують через співпрацю між самоорганізовуваними багатофункціональними командами.

Гнучка розробка – найкращий засіб для підвищення продуктивності розробників програмного забезпечення.

Більшість гнучких методологій націлені на мінімізацію ризиків, шляхом зведення розробки до серії коротких циклів, що мають назву ітерацій, які зазвичай тривають один-два тижні. Кожна ітерація сама по собі виглядає як програмний проект в мініатюрі, і включає всі завдання, необхідні для видачі мінімального приросту за функціональністю: планування, аналіз вимог, проектування, кодування, тестування і документування. Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі те, що гнучкий програмний проект готовий до випуску наприкінці кожної ітерації. Після закінчення кожної ітерації, команда виконує переоцінку пріоритетів розробки.

Agile акцентує увагу на безпосередньому спілкуванні «віч-на-віч». Більшість agile команд розташовані в одному офісі, його іноді називають bullpen. Як мінімум вона включає і «замовників» (замовники, які визначають продукт, також це можуть бути менеджери продукту, бізнес аналітики або клієнти). Офіс може також включати тестувальників, дизайнерів інтерфейсу, технічних авторів і менеджерів.

Основною метрикою agile методів є робочий продукт. Віддаючи перевагу безпосередньому спілкуванню, agile-методи зменшують обсяг письмової документації в порівнянні з іншими методами. Це привело до критики цих методів як недисциплінованих.

Agile – родина процесів розробки, а не єдиний підхід в розробці програмного забезпечення, і визначається Agile Manifesto. Agile не включає практик, а визначає цінності та принципи, якими керуються успішні команди.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		72

Agile Manifesto розроблений і прийнятий 17 розробниками 11-13 лютого 2001 року на лижному курорті The Lodge at Snowbird в горах Юти. Маніфест підписали представники наступних методологій Extreme programming, Scrum, DSDM, Adaptive software development, Crystal Clear, Feature driven development, Pragmatic Programming. Agile Manifesto містить 4 основні ідеї та 12 принципів. Примітно, що Agile Manifesto не містить практичних порад.

Основні ідеї:

- Особистості та їхні взаємодії важливіші, ніж процеси та інструменти.
- Робоче програмне забезпечення важливіше, ніж повна документація.
- Співпраця із замовником важливіша, ніж контрактні зобов'язання.
- Реакція на зміни важливіша, ніж дотримання плану.

Принципи, які роз'яснює Agile Manifesto:

- задоволення клієнта за рахунок ранньої та безперебійної поставки коштовного програмного забезпечення;
- вітання змін вимог навіть наприкінці розробки (це може підвищити конкурентоспроможність отриманого продукту);
- часта поставка робочого програмного забезпечення (кожен місяць або тиждень або ще частіше);
- тісне, щоденне спілкування замовника з розробниками впродовж всього проекту;
- проектом займаються мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою і довірою;
- рекомендований метод передачі інформації – особиста розмова;
- робоче програмне забезпечення – найкращий вимірник прогресу;
- спонсори, розробники та користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;
- постійну увагу поліпшенню технічної майстерності та зручному дизайну;
- простота – мистецтво не робити зайвої роботи;

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		73

- найкращі технічні вимоги, дизайн та архітектура виходять у самоорганізованої команди;
- постійна адаптація до мінливих обставин.

Маніфест та Принципи гнучкої розробки містять високорівневі ідеї щодо того, як потрібно вибудовувати процес розробки програмного забезпечення, щоб успішно завершувати проекти й створювати команди, в яких приємно та цікаво працювати.

Документи визначають, що потрібно для цього зробити, але не говорять, як це зробити. По-іншому й не могло бути, оскільки Маніфест та Принципи народилися внаслідок консенсусу представників різних (хоча й споріднених) напрямів, які могли знайти спільну основу лише на рівні базових цінностей та принципів.

Критика. Багато керівників проектів, що працюють у традиційних методологіях на кшталт «водоспаду», критикують agile-методи.

Один з повторюваних пунктів критики: при agile-підході часто нехтують створенням «дорожньої карти» розвитку продукту, так само як і управлінням вимогами, в процесі якого і формується така «карта». Гнучкий підхід до управління вимогами не має на увазі далекосяжних планів (по суті, управління вимогами просто не існує в даній методології), а має на увазі можливість замовника раптом і несподівано наприкінці кожної ітерації виставляти нові вимоги, що часто суперечать архітектурі вже створеного і поставленого продукту. Таке іноді призводить до катастрофічних «авралів» з масовим рефакторингом і переробками практично на кожній черговій ітерації.

Крім того вважається, що робота в agile мотивує розробників вирішувати всі прибулі завдання найпростішим і найшвидшим можливим способом, при цьому часто не звертаючи уваги на коректність коду з точки зору вимог базової платформи (підхід «працює, та й добре», при цьому не враховується, що може перестати працювати при найменшій зміні або ж породити важкі до відтворення

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		74

дефекти після реального розгортання у клієнта). Це призводить до зниження якості продукту і накопиченню дефектів.

Методології. Існують методології, які дотримуються цінностей і принципів заявлених в Agile Manifesto, деякі з них:

1. Agile Modeling – набір понять, принципів і прийомів (практик), що дозволяють швидко і просто виконувати моделювання і документування в проектах розробки програмного забезпечення. Не включає в себе детальну інструкцію з проектування, не містить описів, як будувати діаграми на UML.

Основна мета – ефективне моделювання і документування; але не охоплює програмування та тестування, не включає питання управління проектом, розгортання і супроводу системи. Однак включає в себе перевірку моделі кодом.

2. Agile Unified Process (AUP) спрощена версія IBM Rational Unified Process (RUP), розроблена Скоттом Амблером, яка описує просте і зрозуміле наближення (модель) для створення програмного забезпечення для бізнес-застосунків.

3 Agile Data Method – група ітеративних методів розробки програмного забезпечення, в яких вимоги та рішення досягаються в рамках співпраці різних крос-функціональних команд.

4. DSDM заснований на концепції швидкої розробки застосунків (Rapid Application Development, RAD). Являє собою ітеративний і інкрементний підхід, який надає особливого значення тривалій участі в процесі користувача/споживача.

5. Essential Unified Process (EssUP).

6. Екстремальне програмування (Extreme programming, XP).

7. Feature driven development (FDD) – функціонально-орієнтована розробка. Використовуване в FDD поняття функції або властивості (feature) Системи досить близько до поняття прецеденту використання, використовуваному в RUP, істотна відмінність – це додаткове обмеження: «кожна функція повинна допускати реалізацію не більше, ніж за два тижні». Тобто якщо сценарій використання

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		75

досить малий, його можна вважати функцією. Якщо ж великий, то його треба розбити на декілька відносно незалежних функцій.

8. Getting Real – ітераційний підхід без функціональних специфікацій, що використовується для веб-застосунків. У даному методі спершу розробляється інтерфейс програми, а потім її функціональна частина.

9. OpenUP – це ітераційно-інкрементний метод розробки програмного забезпечення. Позиціюється, як легкий і гнучкий варіант RUP. OpenUP ділить життєвий цикл проекту на чотири фази: початкова фаза, фази уточнення, конструювання та передачі. Життєвий цикл проекту забезпечує надання зацікавленим особам та членам колективу точок ознайомлення і прийняття рішень впродовж усього проекту. Це дозволяє ефективно контролювати ситуацію і вчасно приймати рішення про задовільність результатів. План проекту визначає життєвий цикл, а кінцевим результатом є остаточний додаток.

10. Scrum встановлює правила керування процесом розробки та дозволяє використовувати вже існуючі практики кодування, коректуючи вимоги або вносячи тактичні зміни. Використання цієї методології дає можливість виявляти і усувати відхилення від бажаного результату на більш ранніх етапах розробки програмного продукту.

11. Бережлива розробка програмного забезпечення (lean software development). Використовує підходи з концепції бережливого виробництва.

Хоча я реалізовував програму сам, було використано підходи Scrum для саморозвитку та пришвидшенню розробки, розглянемо цей метод. Scrum – підхід управління проектами для гнучкої розробки програмного забезпечення. Скрам чітко робить акцент на якісному контролі процесу розробки.

Підхід вперше описали Гіротакі Такеучі та Ікуджіро Нонака в статті The New New Product Development Game (Гарвардський Діловий Огляд, січ–лют 1986). Вони відзначили, що проекти, над якими працюють невеликі, крос-функціональні команди, зазвичай систематично продукують кращі результати, і пояснили це, як «підхід регбі». У 1991 році ДеГрейс та Шталь у книжці Злі

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		76

проблеми, справедливі рішення послалися на цей підхід, як на Scrum (штовханина; сутичка навколо м'яча (у регбі)), спортивний термін, згаданий в статті Такеучі і Нонака. Кен Швабер на початку 1990-х використовував підхід який привів Scrum в його компанію.

Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом та Швабером на OOPSLA'96 в Остіні. Швабер та Сазерленд протягом наступних років працювали разом щоб обробити та описати весь їхній досвід та найкращі практичні зразки для індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM. Не зважаючи на те, що для Scrum нарікли долю управління проектами з розробки ПЗ, він може також використовуватися в роботі команд обслуговувань програмного забезпечення (software maintenance teams), або як підхід управління розробкою і супроводом програм: Scrum of Scrums.

Scrum – це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи – ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проекту, Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана.

Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		77

Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

Product backlog – це документ, який має список вимог до функціональності, які упорядковані згідно зі ступенем важливості. Product backlog представляє список того, що повинно бути реалізовано. Елементи цього списку називається «історіями» (user story) або елементами backlog-у (backlog items). Product backlog відкритий для редагування усім учасникам Scrum-процесу.

Обов'язкові поля:

1. ID – унікальний ідентифікатор, порядковий номер, який використовується для ідентифікації історій у разі їх перейменування.

2. Назва (Name) – стислий опис історії. Він повинен бути однозначним, щоб і розробники і product owner могли зрозуміти, про що йдеться і відрізнити одну історію від іншої.

3. Важливість (Importance) – ступінь важливості даної історії на погляд product owner 'а. Зазвичай являє собою натуральне число, іноді для цієї цілі використовуються числа Фібоначчі. Чим більше значення, тим більше пріоритет.

4. Попередня оцінка (initial estimate) – початкова оцінка об'єму робіт, необхідного для реалізації історії порівняно з іншими історіями. Вимірюється у story point'ах. Приблизно відповідає числу «ідеальних людино-днів».

5. Як продемонструвати (how to demo) – стисле пояснення того, як завершена задача буде продемонстрована у кінці спринта. Дане поле може являти собою код автоматизованого приймального тесту.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		78

Додаткові поля. Іноді, також, використовуються додаткові поля у product backlog, в основному для того, щоб допомогти product owner'у визначитися з його пріоритетами.

Категорія (track). Наприклад, «панель управління» чи «оптимізація». За допомогою цього поля product owner може легко вибрати усі пункти категорії «оптимізація» і задати їм низький пріоритет.

Компоненти (components) – указує, які компоненти (наприклад, база даних, сервер, клієнт) будуть зачеплені при реалізації історії. Дане поле складається з групи checkbox'ів, які відмічаються, якщо відповідні компоненти потребують змін.

Ініціатор запиту (requestor). Product owner може захотіти зберігати інформацію про усіх замовників, зацікавлених у даній задачі. Це потрібно для того, щоб тримати їх у курсі діла про хід виконання робіт.

ID у системі обліку помилок (bug tracking ID) – якщо ви використовуєте окрему систему обліку помилок, тоді у описі історії корисно зберігати посилання на всі дефекти, які до неї відносяться.

Sprint backlog – містить функціональність, обрану Product Owner із Product Backlog. Всі функції розбиті по задачах, кожна з яких оцінюється командою. Кожен день команда оцінює об'єм роботи, який необхідно провести для завершення задачі.

Burndown chart – показує, скільки вже виконано і скільки ще залишається зробити.

Планування спринта (Sprint Planning Meeting)

Проходить на початку нової ітерації Спринта:

– Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда.

– На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино-годинах.

- Рішення задачі не повинно займати більше 12 годин або одного дня. При необхідності задача розбивається на підзадачі.
- Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт.
- Тривалість наради обмежена зверху 4–8 годинами в залежності від тривалості ітерації, досвіду команди тощо.
- (Перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog.
- (Друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog.

Щоденна нарада (Daily Scrum Meeting)

Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

- починається точно вчасно;
 - всі можуть спостерігати, але говорять тільки обрані;
 - триває не більш ніж 15 хвилин;
 - проводиться в одному і тому ж місці протягом одного спринта.
- Протягом наради кожен член команди відповідає на 3 запитання:
- Що зроблено з моменту попередньої щоденної наради?
 - Що буде зроблено з моменту поточної наради до наступної?
 - Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою).

Демонстрація (Sprint Review Meeting):

- Проходить у кінці ітерації (спринта).
- Команда демонструє внесок функціональності до продукту всім зацікавленим особам.
- Залучається максимальна кількість глядачів.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		80

– Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).

– Обмежена 4-ма годинами в залежності від тривалості ітерації і змін у продукті.

Ретроспектива (Sprint Retrospective):

– Члени команди висловлюють свою думку про минулий спринт.

– Відповідають на два основних запитання: Що було зроблено добре у минулому спринті?; Що потрібно покращити в наступному?

– Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).

– Обмежена 1-3-ма годинами.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм PRESENT – окремий випадок SP-мережі й складається з 31 раунду. Довжина блоку становить 64 біта, а ключі підтримуються в 2 варіантах, 80- і 128-бітні. Такого рівня захисту повинно цілком вистачати для низькозахищених додатків, звичайно використовуваних для розгортання на основі тегів, а крім того, що важливіше, PRESENT багато в чому збігається своїми конструктивними особливостями з потоковими шифрами проекту estream, заточеними на ефективну реалізацію в залозі, що дозволяє нам адекватно порівнювати їх.

Кожний з 31 раундів складається з операції XOR, щоб увести ключ K_i для $1 \leq i \leq 32$, де K_{32} використовується для «відбілювання» ключа, лінійної побітової перестановки й нелінійного шару заміщення (або, попросту говорячи, збільшення стійкості шифрування. Нелінійний шар використовує роздільні 4-бітні S-блоки, які застосовуються паралельно 16 раз на кожному раунді. Шифр, описаний псевдо-кодом представлено на рисунку 4.3.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		81

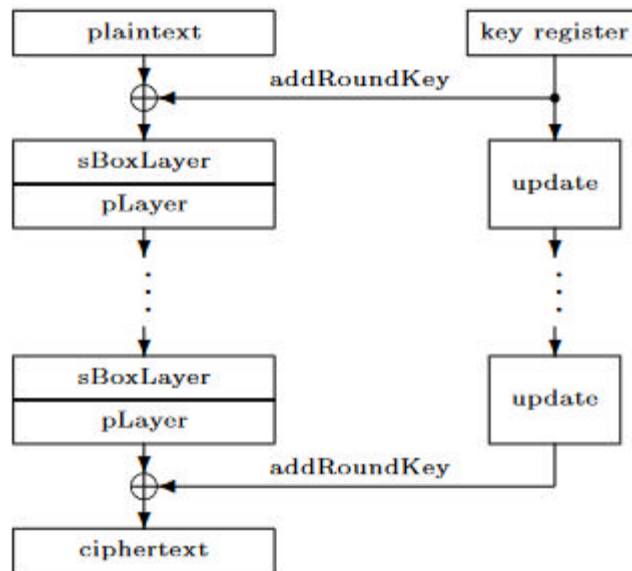


Рисунок 4.3 – Шифр PRESENT

Тепер кожна стадія визначається по черзі. Обґрунтування конструкції наведені нижче, а біти всюди нумеруються з нуля, починаючи із правого в блоці або слові.

Додавання раундового ключа (addRoundKey)

Заданий раундовий ключ $K_i = k_{63}^i \dots k_0^i$, де $1 \leq i \leq 32$, а так само поточний стан $b_{63} \dots b_0$. Додавання раундового ключа до поточного стану відбувається за модулем 2 ($b_j = b_j \oplus k_j^i$, де $0 \leq j \leq 63$).

Шар S-блоків (sBoxlayer)

Використовувані в PRESENT S-блоки відображають 4-бітні блоки в 4-бітні блоки. Дія цього блоку в шістнадцятковій системі числення наведене в наступній таблиці.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Рисунок 4.4 – Дія блоку в шістнадцятковій системі числення

Для шару S-блоків поточний стан $b_{63} \dots b_0$ представляє із себе 16 4-бітних слів $w_{15} \dots w_0$, де $w_i = b_{4i+3} \parallel b_{4i+2} \parallel b_{4i+1} \parallel b_{4i}$ для $0 \leq i \leq 15$. Вихід рамки $S[w_i]$ видає оновлені значення станів очевидним образом.

Шар перестановки (player)

Побітова перестановка, використовувана в PRESENT задається наступною таблицею (біт i стану зміщається на позицію $P(i)$).

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Рисунок 4.5 – Побітова перестановка, використовувана в PRESENT

Перетворення ключа (The key schedule)

PRESENT може використовувати 80- і 128-бітні ключі, однак зосередимося на 80-бітній версії. Надаваний користувачем ключ зберігається в регістрі ключів K , представлений у вигляді $k_{79}k_{78} \dots k_0$. На i -ом раунді 64-бітний раундовий ключ $K_i = k_{63}k_{62} \dots k_0$, що полягає з 64 лівих битов поточного вмісту регістру K . Таким чином, на i -ом раунді маємо:

$$K_i = k_{63}k_{62} \dots k_0 = k_{79}k_{78} \dots k_{16}.$$

Після розпакування раундового ключа K_i регістр ключа $K = k_{79}k_{78} \dots k_0$ оновлюється в такий спосіб:

1. $[k_{79}k_{78} \dots k_1k_0] = [k_{18}k_{17} \dots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \text{round_counter}$

Отже, регістр ключа зрушується на 61 позицію вліво, 4 крайніх лівих біта, що пройшли через S-блок і round_counter значення і складається за модулем 2 з бітами $k_{19}k_{18}k_{17}k_{16}k_{15}$ з K з найменшим значущим бітом з round_counter праворуч.

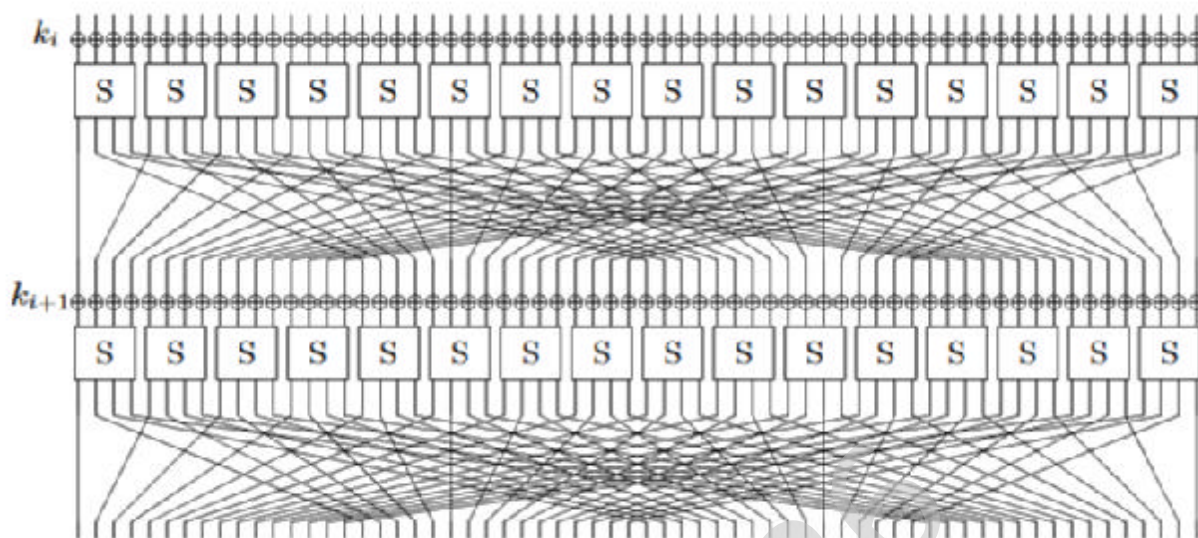


Рисунок 4.6 – Перетворення ключа (The key schedule)

Перетворення ключа для 128-бітного алгоритму можна знайти в додатку до оригінальної статті.

Конструктивні особливості PRESENT

Крім безпеки й ефективної реалізації, основне досягнення PRESENT – його простота. по цьому не дивно, що схожі проекти були прийняті в інших обставинах, і навіть були використані як навчальний посібник для студентів. У даній секції ми обґрунтуємо рішення, прийняті нами при проектуванні PRESENT. Однак, у першу чергу, опишемо очікувані прикладні вимоги.

Цілі й середовище застосування

При проектуванні блокового шифру, застосовного в жорстко обмежених оточеннях, важливо зрозуміти, що ми не створюємо блоковий шифр, який, неодмінно, буде застосовний у багатьох ситуаціях – для цього існує AES. Навпаки, ми націлені на досить специфічне застосування, для якого AES не підходить. Вищесказане визначає нам наступні характеристики:

– Шифр буде реалізований «у залізі»

– Додатки будуть вимагатися лише для регулювання рівня безпеки. Отже, 80-бітний ключ буде здоровим розв'язком. Відзначимо, що такої ж позиції дотримуються розроблювачі потокових шифрів проекту eSTREAM.

– Додатки не припускають шифровки великої кількості даних. Таким чином, реалізація може бути оптимізована для продуктивності або простору без внесення занадто великих змін.

– У деяких застосуваннях можлива ситуація, що ключ буде зафіксований при виробництві. У такому випадку, не треба буде змінювати ключ пристрою (що може вилитися в атаки з маніпуляцією ключем).

– Фізичний обсяг пристрою буде першим пріоритетом, після безпеки, що спричинить обмеження на пікові й середні споживання енергії, і, отже, зрушить швидкодія в область низькопріоритетних параметрів.

– У пристроях, що вимагають найбільш ефективного використання фізичного простору, блоковий шифр найчастіше зможе лише шифрувати дані (encryption-only mode). Таким чином, він зможе бути використаний у запит-відповідь (challenge-response) протоколах авторизації, і, при дотриманні контролю стану, може бути використаний для шифровки й дешифрування переговорів із пристроєм, використовуючи режим лічильника.

Виходячи з таких міркувань, розв'язали створити PRESENT як 64-бітний блоковий шифр із 80-бітним ключем. Шифровка й дешифрування, у цьому випадку, мають приблизно схожі фізичні вимоги. Маючи можливість підтримувати як шифрацію, так і дешифрацію, PRESENT буде компактніше, чим підтримуючий лише шифрацію AES. А у випадку encryption-only виконання, наш шифр виявиться й зовсім понад-легко. Суб-ключі що шифрують будуть обчислюватися на ходу.

У літературі є безліч прикладів атак компромісу між часом, датою й пам'яттю, або атак з використанням парадокса днів народження при шифровці великих обсягів даних. Однак, дані атаки залежать тільки від параметрів шифру й

не використовують внутрішню структуру. Наша мета полягає в тому, щоб ці атаки були кращим, що можуть застосувати проти нас. Атаки стороннього каналу й атаки з безпосереднім зломом чипа загрожують PRESENT тією самою мірою, як і іншим криптографічним примітивам. Однак для ймовірних застосувань, помірні вимоги безпеки роблять вигоду, одержувану зловмисником на практиці, досить обмеженої. В оцінці ризиків, подібні погрози не сприймаються як істотний фактор.

Перестановочний шар

При виборі шару змішування ключа, наша увага до апаратної ефективності вимагає наявності лінійного шару, який може бути реалізований з мінімальною кількістю керуючих елементів (наприклад, транзисторів. це приводить до побітової перестановки. Приділяючи увагу простоті, ми вибрали регулярну бітову перестановку, що допомагає провести прозорий аналіз безпеки.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		86

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система відновлення деталей на розмитих зображеннях за допомогою нейронної мережі. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функціональних кнопок ПЗ; Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші; Верхнього меню; Розділу обрання групи; Розділу виведення результату роботи системи.

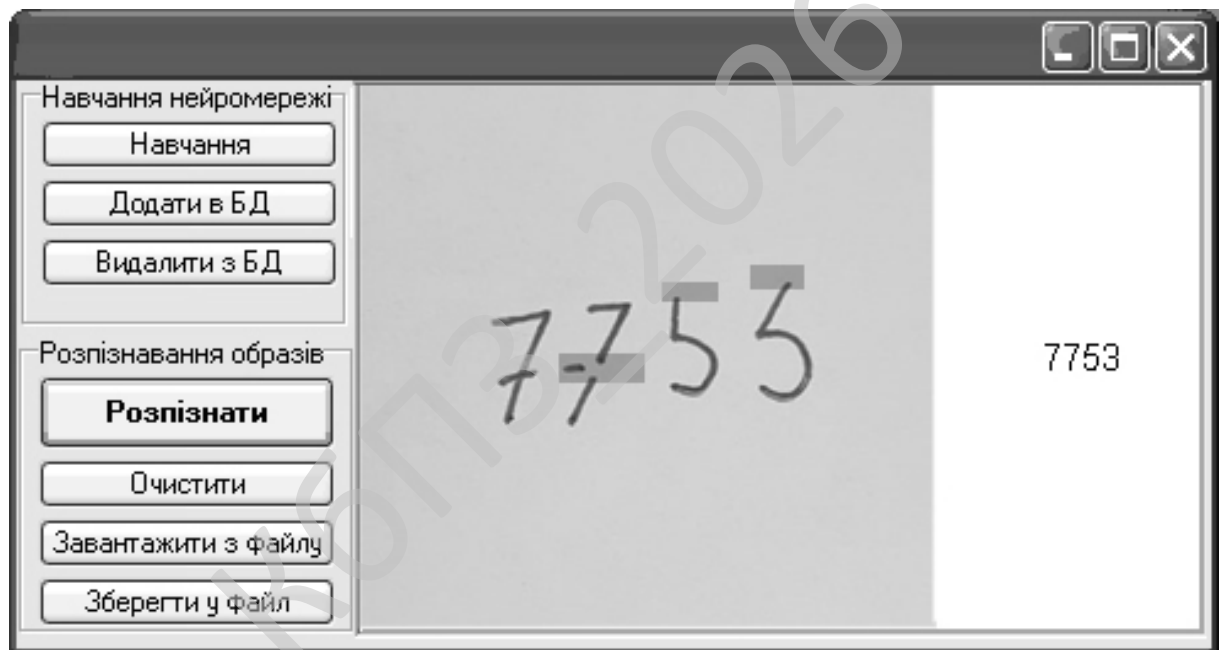


Рисунок 5.1 – Головне вікно ПЗ

Система призначена для відновлення деталей на розмитих фотографіях і кадрах відео в системах розпізнавання графічних образів.

Однією з актуальних проблем сучасних інформаційних технологій у цей час залишається розпізнавання образів, зокрема графічно представлених символів.

Для рішення даного завдання використовується ряд алгоритмів, які можна умовно розділити на три категорії:

- алгоритми, засновані на порівнянні шаблонів;
- алгоритми, засновані на методах теорії рішень;
- алгоритми, що використовують нейронні мережі.

Всі перераховані вище групи алгоритмів, мають, у цілому, однакову структуру й містять такі блоки:

- блок навчання системи на наборі образів, подібним тим, які будуть розпізнаватися системою в майбутньому.

По завершенню навчання, його результати зберігаються;

- блок розпізнавання, що використовуючи інформацію, отриману на етапі навчання, відносить незнайомий образ до одному із класів за допомогою певного математичного апарата (класифікатора).

Варто помітити, що шаблонні методи, у силу своєї негнучкості (прихильності до шрифту, масштабу, нахилу символів) не підходять для загального рішення даного завдання, однак у силу своєї простоти, невеликої ресурсоемності, малим обчислювальним витратам вони, з певними модифікаціями, використовуються досить часто при рішенні окремих завдань розпізнавання. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		88

програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

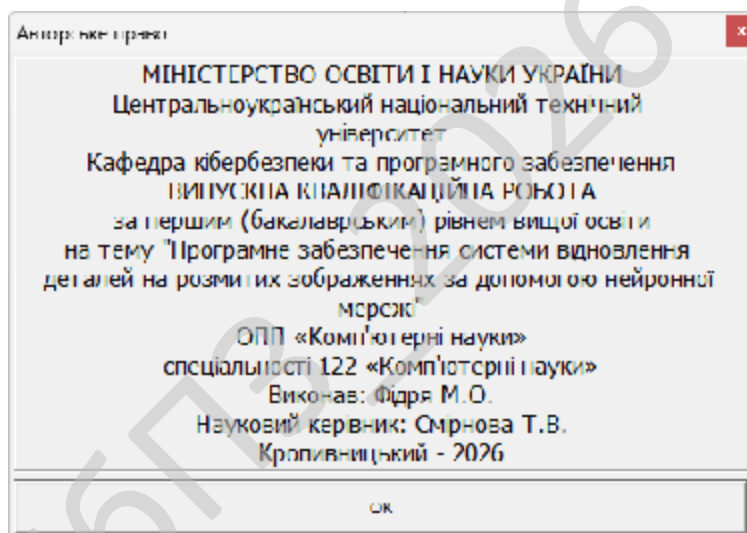


Рисунок 5.2 – Авторське право

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

- Випуск.
- Встановлення та активація.
- Деактивація.
- Адаптація.
- Обновлення.

- Вмонтування.
- Відстежування версій.
- Видалення.
- Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

- Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

- Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

- Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

- відповідності поставленим вимогам;

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		90

- правильна відповідь для усіх можливих вхідних даних;
- виконання функцій за прийнятний час;
- практичність;
- сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

- Гарантується перевірка всіх незалежних маршрутів програми.
- Знаходяться гілки True, False для всіх логічних рішень.
- Виконуються всі цикли (у межах їхніх кордонів та діапазонів).
- Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

- Кількість незалежних маршрутів може бути дуже велика.
- Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.
- У програмі можуть бути пропущені деякі маршрути.
- Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

- Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» — інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

- Як виконуються функції програми.
- Як приймаються вихідні дані.
- Як виробляються результати.
- Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^{10} . Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		92

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс.

– Сформулювати такі очікувані результати, які з високою ймовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилки інтерфейсу;

– Помилки у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилки характеристик (необхідна ємність пам'яті і т.д.);

– Помилки ініціалізації та завершення.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		93

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (zareestruvatisya), zaplativshi avtorovi певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

К6ПЗ_2026

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		94

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.
- Досліджена система відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.
- На основі отриманих результатів досліджень створена програмна реалізація системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання відновлення деталей на розмитих зображеннях за допомогою нейронної мережі.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створенні програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		95

системи відновлення деталей на розмитих зображеннях за допомогою нейронної мережі. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм PRESENT.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

					ВКРБ-122.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		96

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hurbans R. Grokking Artificial Intelligence Algorithms. – Manning, 2020. – 631 p.
2. Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology 1st Edition. – Cambridge University Press, 2008. – 556 p.
3. Kotu V., Deshpande B. Data Science: Concepts and Practice. – Elsevier Science, 2018. – 953 p.
4. Knowledge Base A Complete Guide - 2021 Edition // The Art of Service - Knowledge Base Publishing, 2020. – 306 p.
5. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.
6. Mattmann C. Machine Learning with TensorFlow, Second Edition. – Manning, 2020. – 1124 p.
7. Mueller J.P., Massaron L. Machine Learning For Dummies. – Wiley, 2016. – 714 p.
8. Teofili T. Deep Learning for Search. – Manning, 2019. – 695 p.
9. Rungta K. TensorFlow in 1 Day: Make your own Neural Network. – Publishdrive, 2019. – 587 p.
10. Weidman S. Deep Learning from Scratch: Building with Python from First Principles. – O'Reilly. 2019. – 252 p.
11. Rajasekaran S., Vijayalakshmi Pai G.A. Neural networks, fuzzy logic, and genetic algorithms: synthesis and applications (with cd-rom) Kindle Edition. – PHI, 2013. – 628 p.
12. Вінтенко Б.Ю., Миронець І.В., Смірнов О.А. «Модель комп'ютерно-орієнтованої процедури системи підтримки оперативного персоналу АЕС». *IV Міжнародна науково-практична Інтернет-конференція*

«Інновації та перспективні шляхи розвитку інформаційних технологій (ІПШРІТ-2025)» м.Черкаси 25 листопада 2025 року – Черкаси: ЧДТУ.– 2025. – С.101-103.

13. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing*, 2025. pp. 21-61.

14. Вінтенко, Б.Ю., Миронець, І.В., Смірнов, О.А., Коваленко, О.В., Усік, П.С., Буравченко, К.О., Лисенко, І.А. «Логіко-структурна модель комп'ютерно-орієнтованої процедури системи підтримки оперативного персоналу АЕС». *Кібербезпека: освіта, наука, техніка*. 2025. Том 2 № 30. С. 413-427, 2025.

15. Смірнова, Т.В. «Дослідження методів, моделей та сучасних ІТ-рішень для підтримки технологічних процесів у критичній інфраструктурі держави». *Кібербезпека: освіта, наука, техніка*. 2025. Том 2 № 30. С.195-208, 2025.

16. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка*. 2025. Том 1 № 29. С.704–716, 2025

17. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

18. Kuznetsov, O., Smirnov, O., Akhmetov, B., Alimseitova, Z., Imoize, A.L. «Deep Learning Frontiers in Copy-Move Forgery Detection: Advances, Challenges, and Future Directions». *Advancements in Cybersecurity Next Generation Systems and Applications*, 2025. 202-229.

19. Вінтенко Б., Смірнов О., Миронець І., Смірнова Т., Смірнов С. «Імітаційна модель шляхів вхідних даних комп'ютерної інтелектуальної системи підтримки оператора енергоблоку АЕС». *Комбінаторні конфігурації та їхні застосування: Матеріали XXVII Міжнародного науково-практичного семінару*,

присвяченого 125-річчю Національного університету «Запорізька політехніка» (Запоріжжя-Кропивницький-Київ, 4-6 червня 2025 р.). Запоріжжя: НУ «Запорізька політехніка», 2025. С.82-91.

20. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». *International Review on Modelling and Simulations* 18 (1), 2025. pp. 32-42.

21. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В., Коваленко О.В., Мацуй А.М. «Модель шляхів отримання вхідних даних комп'ютерної інтелектуальної системи підтримки оперативного персоналу АЕС». *Центральноукраїнський науковий вісник. Технічні науки*. 2025. Вип. 11(42), ч. II. С.52-62.

22. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В. «Методи забезпечення відмовостійкості інтелектуальних систем підтримки оператора». *VIII міжнародна науково-практична конференція “Інформаційна безпека та комп'ютерні технології”*, м. Кропивницький. 24-25 квітня 2025 р. – Кропивницький: ЦНТУ. – 2025. – С. 44-46.

23. Смірнов, О.А., Константинова, Л.В., Коноплицька-Слободенюк, О.К., Козірова, Н.В, Якименко, Н.М., Доренський, О.П., Буравченко, К.О. «Дослідження інструментів штучного інтелекту для роботи з базами даних та аналізу даних». *Кібербезпека: освіта, наука, техніка*. 2025. №3(27), С. 429–448.

24. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

25. Smirnov O., Fedorov E., Neskoriadiyeva A., Neskoriadiyeva T. «Intellectual Classification method of Gymnastic Elements Based on Combinations of

Descriptive and Generative Approache». *CEUR Workshop Proceedings* Volume 3664, 2024, Pages 11-23.

26. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

27. Malyukov V., Bebeshko B., Lakhno V., Smirnov O., Malyukova I., Mohylnyi H. «Managing the Purchase-Sale Process of Digital Currencies Under Fuzzy Conditions». *Lecture Notes in Networks and Systems*, 2023, 729 LNNS, pp. 104–112.

28. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

29. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchев, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.

30. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи*, 2023, том 7, № 2, С. 49-56.

31. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». *CEUR Workshop Proceedings*, Volume 3312, 2022, pp. 47-58.

32. Smirnov, O., Neskoriadiеva, T., Fedorov, E., Rudakov, K., Neskoriadiеva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings*, Volume 3187, 2022, pp. 1-12.

33. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022.

34. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebesko, B. «Selection of a Rational Composition of Information Protection Means Using a Genetic Algorithm». In: Rajakumar, G., Du, KL., Vuppapapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

35. Kuznetsov, A., Oleshko, I., Chernov, K., Bagmut, M., Smirnova, T. «Biometric authentication using convolutional neural networks». Lecture Notes in Networks and Systems. Volume 152, 2021, Pages 85-98.

36. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. <https://doi.org/10.1007/s42979-021-00739-w>.

37. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». CEUR Workshop Proceedings Volume 3101, 2021, Pages 192-207.

38. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

39. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

40. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

41. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In:

Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

42. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

43. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

44. Smirnov O. Kuznetsov A., Zaichenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

45. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

46. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

47. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». CEUR Workshop Proceedings, Vol 2588, P. 215-227, 2019.

48. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

49. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

50. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 353-358.

51. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

52. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

53. Smirnov, S., Bulekbaeva, G., Kikvidze, O.G., Lakhno, V., Brzhanov, R., Tabylov, A. «Computer simulation in the MathCAD package of plastic deformation of the deposited layer on the flat surface of the part». Journal of Theoretical and Applied Information Technology Volume 97, Issue 20, 2019, Pages 2467-2484.