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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-1
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Федоров Б.С.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Улічев О.С.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

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
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Федорову Богдану Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга*

2. Керівник роботи *Улічев Олександр Сергійович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 32-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Федоров Б.С. Дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Метою розробки є дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Об'єктом дослідження є процес розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Предметом дослідження є методи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Методи дослідження базуються на методах розпізнавання графічних образів, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

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 архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.4 Sydney.

Ключові слова: комп'ютерні науки, розпізнавання графічних образів, нейронна мережа Хеммінга

ABSTRACT

Fedorov B.S. Research and software implementation of the graphic image recognition system using the Hamming neural network. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software was developed, which is intended for the system of recognition of graphic images using the Hamming neural network.

The purpose of the development is the research and software implementation of a graphic pattern recognition system using a Hamming neural network.

The object of research is the process of recognizing graphic images using a Hamming neural network.

The subject of research is the methods of recognizing graphic images using the Hamming neural network.

Research methods are based on graphic image recognition methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the graphic pattern recognition system using the Hamming neural network.

In the process of working on the software model, an analysis of existing hardware and software 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 PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Delphi 10.4 Sydney environment.

Keywords: computer science, graphic image recognition, Hamming neural network

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ПЗ	–	програмне забезпечення

ВСТУП

Актуальність теми. Можливість автоматично знімати, обробляти, аналізувати та використовувати зображення та відеовміст сьогодні стає все більш важливою завдяки покращенню обчислювальної потужності, зростаючій кількості камер і швидкому зростанню використання мобільних пристроїв. Ми зосереджені на тому, щоб зробити комп'ютери інтелектуальнішими, щоб вони могли автоматично аналізувати та вивчати дані зображення, розпізнавати об'єкти та автоматично розуміти вміст зображення. Очікується, що ці технології значно зроблять наше життя багатшим, безпечнішим і зручнішим. Поле досліджень охоплює широкий спектр тем, пов'язаних із зображеннями. Обробка зображень включає методи покращення, відновлення, сегментації, стиснення та водяних знаків. Досконалішим прикладом є адаптивне масштабування зображення, яке інтелектуально забезпечує гарну якість перегляду для різних пристроїв відображення. Комп'ютерне бачення охоплює всі види аналізу візуального вмісту, наприклад, калібрування камери, тривимірне моделювання, автоматичну локалізацію та вивчення навколишнього середовища в стереобаченні, виявлення та розпізнавання осіб, об'єктів і подій у системах спостереження, а також відеоконтент аналіз та багато інших методів. Розпізнавання образів стосується методів автоматичної класифікації та кластеризації даних, які мають багато застосувань у комп'ютерному зорі.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

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

– Огляд існуючих систем розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

– Дослідження системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

– Програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Об'єктом дослідження є процес розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Предметом дослідження є методи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Методи дослідження базуються на методах розпізнавання графічних образів, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

– Розроблено вітчизняний продукт розпізнавання графічних образів за допомогою нейронної мережі Хеммінга, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у роботі збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для розпізнавання графічних образів, що може застосовуватися у наступних завданнях:

1. Категоризація зображень:

– Ключові слова стокових зображень.

– Категорії для систем пошуку та рекомендацій.

2. Робототехніка:

– Картування середовища та навчання.

– Виявлення перешкод.

– Ідентифікація аномалій продукту.

– Профілактичне обслуговування.

3. Споживацькі товари:

Автономне оформлення замовлення · Рівні запасів на полицях · Захоплення товару в кошик · Аналіз проходу · Відповідність товару

4. Автомобільна страховка

– Оцінка збитків.

– Ідентифікація автомобільних запчастин.

– Зображення претензій.

5. Логістика:

– Дані про контейнери з верфів.

– Ідентифікація руху активів партіями.

6. Аерофотозйомка:

– Супутники.

– Дрони.

– Топографія.

– Аерофото/відео.

7. Автономні транспортні засоби:

– Тривимірна хмара точок.

– Лідар малого/дальнього радіусу дії.

– Планування траєкторії.

– Позначення пішоходів.

– Перехрестя.

– Світлофори.

– Велосипеди.

– Об'єкти.

8. Розпізнавання обличчя:

– Підготовка відповідних наборів зображень.

– Визначення унікальних атрибутів.

– Відповідні аксесуари тощо.

9. Точне землеробство:

– Розмежування культур і ґрунтів.

– Точне застосування пестицидів.

– Виявлення аномалій культур.

10. Потужні інструменти розпізнавання зображень для навчання ваших моделей ШІ:

– Точка, яка визначає об’єкт або демаркує частину об’єкта. Ідеально підходить для виявлення ключових точок і картографування, показу руху, пози об’єкта 6Dof, спортивного аналізу та медицини, віртуальної та доповненої реальності, робототехнічних маніпуляцій і візуальної навігації.

– Теги можна використовувати для опису видимих об’єктів і класифікації цілих зображень, окремих областей, елементів і конкретних атрибутів. Ідеально підходить для пошукових систем, систем рекомендацій і платформ пошуку.

– Обмежувальна рамка – це прямокутник, накреслений навколо кінцівок об’єкта. Ідеально підходить для ідентифікації об’єктів, класифікації, локалізації

та оцінки пошкоджень для автострахування, ідентифікації продуктів для роздрібної торгівлі та виявлення аномалій продукту для виробництва.

– Багатокутники дозволяють анотаторам створювати ідеальні піксельні силуети аморфних об’єктів. Ідеально підходить для виявлення об’єктів, семантичної сегментації, сегментації екземплярів, об’ємної сегментації та складених об’єктів.

– Лінії – це ряд точок, які можна використовувати для визначення нахилу, напрямку або краю. Сплайни – це серія безперервних нерегулярних кривих, які дозволяють анотатору створити криву, яка максимально наближена до ідеальної відповідності точок даних. Ідеально підходить для визначення розмітки та інтерполяції.

– Розпізнавання обличчя ґрунтується на розрізненні тонких ключових атрибутів обличчя людини під різними кутами та при різному освітленні. Ідеально підходить для безпеки та підтвердження особи, безпеки водія, організації фото- та відеозйомки, емоційного інтелекту та соціальних мереж.

– Поєднуючи дані датчика 3D LiDAR із 2D-зображеннями та відео, анотатори можуть створювати анотації прямокутної форми для відстеження тривимірних об’єктів у просторі та часі. Ідеально підходить для автономних транспортних засобів, виробництва та сільського господарства.

1.2 Область застосування

Областями застосування системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга є наступні:

– Відеоспостереження.

– Відстеження та розпізнавання об’єктів.

– Калібрування камери.

– Поліпшення та відновлення зображення.

– Сегментація зображення.

– 3D моделювання та реконструкція.

– Автономна навігація та розуміння середовища,.

– Аналіз відео.

– Індексація зображень і відео.

– Стиснення зображень і відео.

– Обчислювальна фотографія.

– Приховування інформації та цифрові водяні знаки.

– Аналіз дистанційного зондування та сейсмічних даних.

– Алгоритми кластеризації.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Проведемо огляд програм розпізнавання графічних образів.

Nero Kwik Media

Nero Kwik Media є безкоштовним менеджером не тільки фотографій, але також відео й музики. З його допомогою можна знаходити, редагувати, програвати, організовувати, синхронізувати різні файли мультимедіа. Утиліта підтримує основні формати графічних файлів, відео й слайд-шоу.

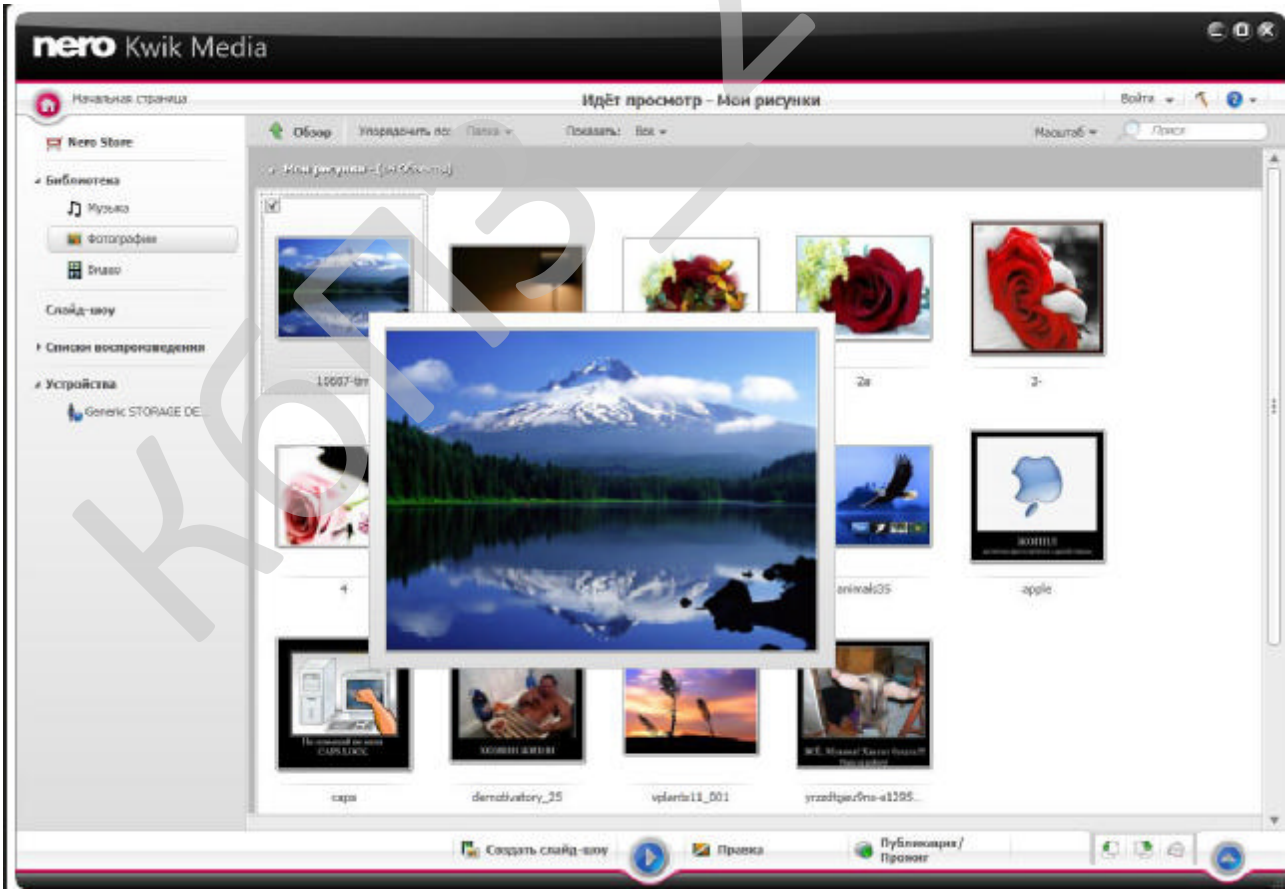


Рисунок 2.1 – Інтерфейс користувача Nero Kwik Media

Nero Kwik Media добре зарекомендувала себе і як програма для редагування фотографій. Вона здатна зняти ефект червоних очей, поліпшити якість знімків, вирізати непотрібні частини фото. Приємна особливість програми – алгоритм розпізнавання осіб. Це відмінний механізм для сортування й створення фотобібліотек із присутніми на знімках людьми.

Нейромережа Google

У червні 2012 року група дослідників з Google запустила нейромережу на кластері 1000 комп'ютерів (16 тис. процесорних ядер; 1 млрд зв'язків між нейронами). Експеримент став одним із самих масштабних в області штучного інтелекту, причому систему створювали для рішення практичних завдань.

Самонавчальна нейромережа – досить універсальний інструмент, який можна використовувати на різних масивах даних. У компанії Google її застосували для поліпшення точності розпізнавання мови: «Ми одержали зменшення на 20-25% кількості помилок при розпізнаванні, – говорить Винсент Ванхоук (Vincent Vanhoucke), керівник відділу розпізнавання мови в Google. – Це значить, що багато з людей одержать безпомилковий результат». Нейромережа оптимізувала алгоритми для англійської мови, але Ванхоук говорить, що аналогічні поліпшення можуть бути досягнуті й для інших мов і діалектів.

Нейромережа використовується також у проекті Google Street View для обробки маленьких фрагментів фотографій, де потрібно визначити – є число на фрагменті номером чи вдома ні. Дивно, але в цьому завданні нейромережа показує кращу точність розпізнавання, ніж люди.

У майбутньому нейромережа буде використана в інших продуктах Google, таких як пошук зображень, окуляри Google Glass і автомобілі Google з безпілотним керуванням. Один зі співробітників проекту по розробці нейромережі Джефф Дин (Jeff Dean) говорить, що в автомобілі система може враховувати контекстну інформацію, у тому числі інформацію з лазерних далекомірів або, наприклад, звук мотора. Джефф Дин говорить, що потужна нейромережа здатна використовувати багато контекстної інформації в процесі

тренування – із цієї причини вони й вирішили створити такий великий кластер з 1000 серверів, у той час як більшість дослідників тестують нейромережі на одному комп'ютері.

Перші результати експерименту з нейромережею Google були опубліковані в червні 2012 року. Тести показали, що нейромережа успішно піддається самонавчанню. Після перегляду 10 мільйонів випадкових кадрів з Youtube у нейромережі сформувалися нейрони, що селективно реагують на присутність осіб на зображеннях. На думку вчених, нейромережа Google у процесі самонавчання працювала приблизно так само, як працюють нейрони в зоровій корі головного мозку ссавців. З тим застереженням, що нейромережа Google, незважаючи на свої масштаби, все таки набагато менше по кількості вузлів, ніж нейромережа зорової кори.

FaceReader™

Компанія-Розроблювач: Noldus Information Technology (Нідерланди).

Програма може вірно інтерпретувати такі вираження особи, як «щасливе», «смутне», «сердите», «здивоване», «перелякане», «незадоволене» і «нейтральне», як видно на рисунку 2.2. Крім того, FaceReader здатний по особах людей визначати їхній вік, стать й етнічну приналежність. FaceReader не має потреби в навчанні й додатковому наналаштуванні.

У програмі реалізовані технології комп'ютерного зору. Зокрема, це метод Active Template, що полягає в накладенні на зображення особи деформуємого шаблона.

Також, реалізований метод Active Appearance Model, за допомогою якого можна створювати штучну модель особи з урахуванням контрольних точок і деталей поверхні, і порівнювати неї зі зразками, закладеними до пам’яті.

Класифікація відбувається методами нейронних мереж із тренувальним набором в 2 000 фотографій.

Рисунок 2.2 – Інтерфейс користувача FaceReader™

Можливості програми:

– середній відсоток розпізнавання емоцій дорівнює 89%. Для деяких емоцій він вище, для деяких нижче;

– нахил особи може бути кожним у площині, його система виявить;

– програма працює із завантажується відео у форматах з кодеками MPEG1, MPEG2, Xvi, Div4, Div5, Div6, DV-AVI і uncompressed AVI, причому визначати емоції можна пофреймно, або повністю при перегляді всього відео. Також, FaceReader може працювати зі статичними зображеннями, а також у реальному часі, якщо в користувача підключена веб-камера;

– програма прекрасно візуалізована: завжди можна подивитися гістограми, діаграми, відсоток емоцій, що виражаються. А на таймлайні видні прояви мікровиражень у певний проміжок часу;

– FaceReader генерує два текстових файли, один – це лог прояву емоцій, а інший – статичний, для сполученої з даною програмою унікальної системи керування візуальними даними The Observer XT®, розробленою цією же компанією.

Недоліки програми:

– FaceReader не натренований для розпізнавання дітей до 5ти років.

– Якщо людина в окулярах, то розпізнавання емоцій неточне, або класифікація не ведеться.

– Люди з різним кольором шкіри по-різному сприймаються системою, програма не до кінця адаптована.

– Повернена особа не детектується.

eMotion Software і GladOrSad

Система eMotion Software відома тим, що її засновники розпізнали емоції на картині «Мона Ліза». Результат показав, що вона була на 83% щасливою, 9% відображали відразу, 6% страх і всього на 2% Мона Ліза гнівалася.

А ще система відома тим, що це, по суті, перше комерційне платне «коробкове» рішення. Поряд з даним рішенням, група розроблювачів запустила сайт GladOrSad.com – відповідно, Visual Recognition взяла першість і у відкритті веб-ресурсу, присвяченого онлайн-розпізнаванню емоцій.

Першим відомим користувачем eMotion Software стала компанія Unilever, що впровадила систему розпізнавання в апарат із продажу морозива – Unilever Share Happy. Люди посміхаються автомату, автомат дає за посмішки безкоштовне морозиво!

Якщо людина проявляє емоції, посміхається, супиться або корчить гримасу, тисячі дрібних м'язів особи перебувають у роботі. Система розпізнавання емоцій, або ERS (Emotion-recognition system), створює 3D-модель особи, з виявленням 12 ключових областей, таких як куточки ока й куточки рота.

Рисунок 2.3 – Інтерфейс користувача eMotion Software

У даних програмах алгоритм, що відслідковує, ідентифікує ті ж самі емоції, їх тут шість: гнів, сум, страх, подив, відраза й щастя, а також сьома – це їхнє змішання.

Програмне забезпечення не особливо вимогливо до обчислювальної машини по технічних характеристиках. Про деталі реалізації алгоритму невідомо, тому що технологія тримається в секреті, брошур з поясненнями я також не знайшов, на жаль.

MMER_FEASy – the Fac Analysis System

Знову ж, у розробці використана методологія накладення на особу певної деформуємої маски, Active Appearance Model methodology, що дозволяє

вираховувати потрібні параметри в реальному часі. Робота з маскою продемонстрована на рисунку нижче.

Рисунок 2.4 – Інтерфейс користувача MMER_FEASy

Система використовує три підключаємі модулі – MMER_Lab, MMER_GPU і MMER_Locate.

MMER_Locate забезпечує знаходження особи на зображенні, MMER_Lab класифікує деякі ознаки даного зображення, а MMER_GPU забезпечує ефективну роботу всієї системи:

Програма розпізнає шість базових емоцій, також надає послуги по знаходженню по особах людей віку, статі й етнічної приналежності. Також

система ідентифікує персону якщо при цьому коли-або раніше еталонна фотографія була завантажена в базу.

У додаткові можливості програми входить підключення до інших програм її як модуля для вилучених асистентів, асистентів водіїв, маркетингових досліджень і домашніх мультимедіа-сервісів.

Недоліками програми можна вважати не повний охват даних, що завантажуються, тому що працювати можна тільки з веб-камерою. Погані результати й по вивантаженню даних, де можна переглянути тільки «аватаризацію» особи, тобто замість маски підставляється ця ж особа, але з іншою мімікою.

FaceSecurity

Даний продукт складається з декількох компонентів, що випускаються, заснованих на базі FaceVACS SDK. Це:

– FaceVACS-DBScan with Examiner.

– FaceVACS-PortraitAcquisition.

– FaceVACS-VideoScan.

FaceVACS-DBScan with Examiner

Розробка призначена для обробки унікальних баз і банків даних якої-небудь категорії людей, приміром, співробітників по роботі.

Даний продукт являє собою втілення біометричної ідентифікації по еталоні зразків з бази.

У новій версії програми система використовує новий алгоритм порівняння B5T8 укупі зі старим A14T8, покликаний поліпшити обумовлену подібність.

Крім того, новий компонент Examiner допускає автоматичні перетворення зображення для порівняння в галереї. Це дозволяє розробленим операторам дивитися списки потенційних партнерів при збереженні повного аудита для кожного кроку в процесі.

Також дана розробка допомагає слідчим ідентифікувати особи в місцях злочину по фотографії й відеоспостереженню шляхом зіставлення зображень осіб

у сховище агенств. FaceVACS-Examiner також надає набір інструментів, які допомагають інспекції ідентифікувати особа вчасно, що дозволяє слідчим діяти відповідно до результатів пошуку в самий мінімальний відрізок часу після здійснення злочину.

Особливості:

– Кластерна конфігурація для багатомільйонної обробки бази даних людей.

– Гнучке й зручне керування списком, що дозволяє сортувати його, переглядати й фільтрувати.

– Пакетне й інтерактивне навчання, ідентифікація.

– Глибоке й гнучке керування зв'язаних даних.

– Логи що перенастроюються.

– Прегенерація баз даних (тобто попередня швидка преднастройка перед видачею результату).

– Доступні безлічі різних пошукових вибірок по базі.

FaceVACS-PortraitAcquisition

Створення й оцінка цифрових портретів для фотодокументів, що засвідчують особистість. Робота компонента ілюстрована рисунку 2.5.

Даний компонент спрощує виробництво портретів високої якості для фото на паспорт, права водія та інші документи, які підходять для розпізнавання осіб.

Графічний користувальницький інтерфейс продукту спеціально підбудований для візуального керування й оперирування процесом обробки таких дріб'язків як фронтальна поза, рівномірне висвітлення, окуляри й очі, що замружилися. Програмне забезпечення спеціально настроєне для оцінки на відповідність зображення обов'язковим вимогам і кращим практичним рекомендаціям стандарту ISO 19794-5 фронтального типу зображення. Програма підтримує інтеграцію за допомогою веб-служб (SOAP) для полегшення видачі зробленого документа.

Рисунок 2.5 – Інтерфейс користувача FaceVACS-PortraitAcquisition

Особливості:

– Повна відповідність стандартам ISO 19794-5.

– Надійний і автоматизований процес збору інформації.

– Перевірка фронтальної пози, наявності окулярів, рівномірного висвітлення, розміру голови, розмірів зображення, відкриття рота, повороту голови, перевірка на тоновані стекла, на червоні очі, фронтальний погляд очей, експозиції, кольору шкіри, гарячих точок, різкості.

– Зручний графічний інтерфейс користувача.

– Параметри, що налаштовуються, і граничні значення.

– Підтримується формат цифрових дзеркальних фотокамер від Nikon (D5000) і Canon (EOS 1000D і 1100D EOS).

– Гнучка обрізка, при необхідності.

– Зображення, що налаштовується, при передперегляді (розмір, тип зображення, формат зображення).

– Гнучка інтеграція з веб-сервисами.

FaceVACS-VideoScan

Нове покоління комп'ютерного відеоспостереження автоматично сканує вхідний відеопотік, виявляє трохи осіб і перевіряє наявність можливих збігів в «контрольному списку». Якщо відповідність знайдена, оператори одержують повідомлення в реальному часі.

Додаток містить у собі не тільки виявлення небажаних людей у громадських місцях, а також ідентифікацію високопоставлених клієнтів.

Особливості:

– Автоматичне стеження за особою в реальному часі на декількох відеопотоках.

– У режимі реального часу особа рівняється з еталонами «контрольного списку».

– Реєстрації при нерухливому зображенні або при живому потоці відео в ручному й автоматичному режимі.

– Застосування C++ API і Web Services API.

– Можливість масштабованості в межах «контрольного списку»: розмір, кількість відеопотоків і число видимих осіб на зображенні.

Можливості програм Cognitec

– Надзвичайно висока обробка порівнянь по зразках (900 000 порівнянь у базі за секунду на середньому по силі процесорі).

– Інтеграція з веб-камерами, http-камерами, цифровими фотоапаратами, відеокамерами, а також підтримка зображень у розповсюджених форматах.

– Об'ємні бази даних, інтеграція з Oracle, IBM DB2, MSSQL Server.

Недоліки:

– Обчислення практично у всіх компонентах застосовні тільки до фронтально розташованих осіб (можливі відхилення на 15 градусів, але не більше);

– Світло відіграє велику роль – приміром, зображення особи в тіні компонента не розпізнають.

Не можна не згадати про нашумілому web-рішенні на базі FaceVACS-SDK і FaceVACS-DBScan, порталі MyHeritage.com, де можна будувати своє сімейне дерево на основі розпізнавання особи, а також зрівняти себе зі знаменитостями, зробити морфінг особи, а також розпізнати й позначити себе на фото.

Крім Web-додатків, Cognitec надає API для цифрових рекламних щитів – білбордів (billboard) з метою показу реклами для цільових аудиторій.

Як уже згадувалося вище, Cognitec бере участь і в машинобудуванні, системи даної компанії застосовуються в автомобілях для аналізу осіб водіїв і попутників, а також безпеки, наприклад, шляхом виявлення позиції голови, виявлення розсіяного погляду, виявлення закритих очей.

Ще одним вигідно, що виділяє моментом, Cognitec серед інших компаній є наявність свого власного SDK для мобільних телефонів.

Продукти Affective Computing Research Group

Компанія Розалинди Пикард, Affectiva, відома в першу чергу поставляємими біосенсорами, що носяться Q-Sensor. Але не тільки цим багата компанія. Є величезний досвід впровадження технологій серед Affective computing, або емоційних обчислювальних систем, розробки йдуть із 1995 року. Проектів дуже багато. Це сама найстарша група розроблювачів, що займається даними технологіями.

Є, наприклад, проект AffQuake на базі продукту ID Software Quake 3. Суть у тому, щоб гра реагувала на емоційні сигнали гравця. Геймера обважують датчиками, і якщо йому стає страшно, модифікований Quake одержує

«фізіологічні сигнали» користувача й змушує точно так само боятися віртуальне втілення гравця – воно в страху відступає.

Або, приміром, розроблена іграшка «Емоційний тигр» (Affective Tigger). Даний робот може ідентифікувати п'ять емоційних станів граючої з ним дитини й виразити у відповідь свою емоцію. Якщо дитина стрибає, весело тискає й цілує іграшку, то система розпізнавання емоцій і сенсорна система «тигра» цей фізичний вплив фіксують, після чого демонструється щастя: Affective Tigger сміється й посміхається.

Є цікаве рішення для Web. Це інноваційний продукт компанії Affectiva, збір даних про емоційний стан людей у всесвітній павутині, Affdex. По більшій частині він використовується для маркетингових досліджень.

Одним зі способів розпізнавання емоційного стану по особі в даних розробках є запис у реальному часі з наступним комп'ютерним аналізом – методами порівняння із закладеними зразками (SURF і на основі SIFT–дескрипторів), а також вейвлет-методами. Робота даних методів застосовується в такій програмі як Pupeteer, що оцінює поводження й емоційний стан учнів.

У ході експерименту по даному методі шість базових емоцій комп'ютер визначає з 96-процентною точністю. Рішення примітно ще тим, що розпізнає вкупі з емоціями руху голови, такі як кивок або хитання, мотання зі сторони убік. Використовуються процеси Байєсовського машинного навчання для класифікації емоцій, а також для обчислення статистики й обчислення змішаних станів, коли не можна точно виразити, яка саме емоція превалює.

Про саме програмне забезпечення можна не багато чого сказати, тому що технології закриті. Розробка ведеться на C++, Objective C для iPhone. Графічно оформляється через звичайні інструменти, такі як timeline (або шкала часу), графіки й діаграми. Серед особливостей можна помітити, що всі рішення добре промальовані й адаптовані під замовника, а з недоліків, мабуть, найбільшим є наявність обчислювальної машини з неслабим процесором (вище Core i5) для комфортної роботи з додатками.

Рисунок 2.5 – Інтерфейс користувача Affectiva

Порівняння компаній і проміжні підсумки

Розглянуто рішення лише деяких гравців даного бізнесу. Інші компанії надають свої продукти, призначені для інших завдань, але розроблювальні ними системи так чи інакше цікаві, тому що можуть бути легко вдосконалені до рівня розпізнавання графічних образів. Це програми й рішення, що виконують такі завдання як:

– верифікація особи (системи безпеки й контролю доступу);

– трекінг і відстеження особи (системи відеоспостереження);

– порівняння людей по образу й подобі своєму (системи пошуку);

– анімація особи і його перетворення (системи морфінгу);

– перетворення особи в 3D-моделі (системи моделювання);

– визначення раси, віку й статі людини (системи гендерної класифікації);

– багато чого іншого.

Причому більшість компаній, що розробляють дані програми, надають свій інструментарій (SDK – Software Development Kit) будь-якому розроблювачеві.

Ключовим критерієм застосування таких продуктів є їхня вартість, а також вартість надаваного SDK. Вона коливається від 5$ до 2 000$. Найбільш дорогі продукти навряд чи будуть застосовуватися в невеликих компаніях, де розробки такого роду не є умовою роботи всієї компанії.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

– У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion,

навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Нещодавні досягнення в галузі штучного інтелекту та машинного навчання сприяли розвитку концепцій комп'ютерного зору та розпізнавання зображень. Від керування автомобілем без водія до визначення обличчя для біометричного доступу, розпізнавання зображень допомагає в обробці та категоризації об'єктів на основі навчених алгоритмів. Продовжуйте читати, щоб зрозуміти, що таке розпізнавання зображень і наскільки воно корисне в різних галузях.

Коли справа доходить до ідентифікації зображень, ми, люди, можемо чітко розпізнавати та розрізняти різні характеристики об'єктів. Це тому, що наш мозок несвідомо тренувався з тим самим набором образів, що призвело до розвитку здатності легко розрізняти речі.

Ми навряд чи свідомі, коли інтерпретуємо реальний світ. Зіткнутися з різними об'єктами візуального світу та з легкістю розрізнити їх не становить для нас труднощів. Наша підсвідомість без зайвих зусиль здійснює всі процеси.

На відміну від людського мозку, комп'ютер розглядає візуальні елементи як масив числових значень і шукає шаблони в цифровому зображенні, будь то фото, відео, графіка чи навіть живе, щоб розпізнавати та розрізняти ключові характеристики зображення. Спосіб, у який система інтерпретує зображення, повністю відрізняється від способу, у який система інтерпретує зображення. Комп'ютерне зір використовує алгоритми обробки зображень, щоб аналізувати та розуміти візуальні ефекти з одного зображення або послідовності зображень. Прикладом комп'ютерного зору є ідентифікація пішоходів і транспортних засобів на дорозі за допомогою точної класифікації та фільтрації мільйонів зображень, завантажених користувачами.

Ринкові можливості та сфера комп'ютерного зору

З роками ринок комп'ютерного бачення значно виріс. На даний момент його вартість оцінюється в 11,94 мільярда доларів США та, ймовірно, досягне 17,38 мільярда доларів США до 2023 року при середньорічному темпі зростання 7,80% між 2018 і 2023 роками.

Це пов'язано зі збільшенням попиту на автономні та напівавтономні транспортні засоби, дрони (військового та побутового призначення), носимі пристрої та смартфони. Більше того, зростання впровадження Індустрії 4.0 та автоматизації у виробничих галузях ще більше стимулювало попит на комп'ютерне бачення.

Враховуючи зростаючий потенціал комп'ютерного зору, багато організацій інвестують у розпізнавання зображень для інтерпретації та аналізу даних, що надходять переважно з візуальних джерел, для цілого ряду застосувань, таких як аналіз медичних зображень, ідентифікація об'єктів в автономних автомобілях, розпізнавання облич для цілей безпеки тощо.

Розпізнавання зображень – це здатність системи або програмного забезпечення ідентифікувати об'єкти, людей, місця та дії на зображеннях. Він використовує технології машинного зору зі штучним інтелектом і навченими алгоритмами для розпізнавання зображень через систему камер.

Завдяки останнім досягненням у машинному навчанні та збільшенню обчислювальної потужності машин розпізнавання зображень захопило світ штурмом.

Автомобільна промисловість, електронна комерція, роздрібна торгівля, обробна промисловість, безпека, відеоспостереження, охорона здоров'я, сільське господарство тощо можуть мати широке застосування розпізнавання зображень.

Як працює розпізнавання зображень?

Цифрове зображення являє собою матрицю числових значень. Ці значення представляють дані, пов'язані з пікселем зображення. Інтенсивність різних пікселів усереднюється до одного значення, що представляє себе у форматі

матриці.

Інформація, яка надходить до систем розпізнавання, – це інтенсивність і розташування різних пікселів на зображенні. За допомогою цієї інформації системи вчаться відображати взаємозв'язок або шаблон у наступних зображеннях, які надаються їй у рамках процесу навчання.

Після завершення процесу навчання продуктивність системи на тестових даних перевіряється.

Щоб підвищити точність системи для розпізнавання зображень, переривчасті вагові коефіцієнти нейронних мереж змінено для підвищення точності систем.

Деякі з алгоритмів, які використовуються для розпізнавання зображень (розпізнавання об'єктів, розпізнавання облич), це SIFT (масштабно-інваріантне перетворення ознак), SURF (прискорені надійні функції), PCA (аналіз основних компонентів) і LDA (лінійний дискримінантний аналіз).

Проблеми розпізнавання зображень

Варіація точки огляду: у реальному світі об'єкти зображення вирівняні в різних напрямках, і коли такі зображення надходять у систему, система передбачає неточні значення. Коротше кажучи, система не може зрозуміти, що зміна вирівнювання зображення (зліва, справа, знизу, вгорі) не змінить його, і тому це створює проблеми з розпізнаванням зображення.

Варіація масштабу: варіації розміру впливають на класифікацію об'єкта. Чим ближче ви розглядаєте об'єкт, тим більшим він виглядає за розміром, і навпаки

Деформація: об'єкти не змінюються, навіть якщо їх деформувати. Система навчається на ідеальному зображенні та формує уявлення про те, що певний об'єкт може мати лише певну форму. Ми знаємо, що в реальному світі форма змінюється і, як наслідок, виникають неточності, коли система зустрічає деформоване зображення об'єкта.

Міжкласові варіації: Певний об'єкт змінюється в межах класу. Вони

можуть бути різної форми, розміру, але при цьому представляти один і той же клас. Наприклад, ґудзики, стільці, пляшки, сумки бувають різного розміру та зовнішнього вигляду.

Оклюзія: певні об'єкти перешкоджають повному перегляду зображення та призводять до того, що в систему надходить неповна інформація. Необхідно розробити алгоритм, чутливий до цих варіацій і складатися з широкого діапазону вибірок даних.

Для навчання моделей нейронної мережі навчальний набір повинен мати різновиди, що стосуються одного класу та кількох класів. Різновиди, доступні в навчальному наборі, гарантують, що модель точно прогнозує під час тестування на тестових даних. Однак, оскільки більшість зразків розташовані у випадковому порядку, перевірка наявності достатньої кількості даних вимагає ручної роботи, яка є виснажливою.

Обмеження звичайних нейронних мереж для розпізнавання зображень:

– Величезна доступність даних ускладнює їх обробку через обмежену доступність апаратного забезпечення.

– Труднощі в інтерпретації моделі, оскільки нечітка природа моделей забороняє її застосування в ряді областей.

– Розробка займає більше часу, і, отже, гнучкість погіршується часом розробки. Хоча доступність таких бібліотек, як Keras, робить розробку простою, їй бракує гнучкості у її використанні. Крім того, Tensorflow забезпечує більше контролю, але він складний за своєю природою та потребує більше часу на розробку.

Роль згорткових нейронних мереж у розпізнаванні зображень

Згорткові нейронні мережі (CNN) відіграють вирішальну роль у вирішенні зазначених вище проблем. Його основні принципи черпали натхнення з нашої зорової кори.

CNN вносить зміни в режим роботи. На входи CNN не подаються повні числові значення зображення. Замість цього повне зображення ділиться на кілька

невеликих наборів, кожен з яких виступає як зображення. Невеликий розмір фільтра ділить повне зображення на невеликі частини. Кожен набір нейронів підключений до невеликої ділянки зображення.

Потім ці зображення обробляються подібно до звичайного процесу нейронної мережі. Комп'ютер збирає візерунки щодо зображення, а результати зберігаються у форматі матриці.

Цей процес повторюється, доки системі не буде надіслано повне зображення в бітах. Результатом є велика матриця, що представляє різні шаблони, які система захопила із вхідного зображення.

Ця матриця знову знижується (зменшується розмір) за допомогою методу, відомого як Max-Pooling. Він витягує максимальні значення з кожної підматриці та створює матрицю набагато меншого розміру.

Ці значення представляють візерунок на зображенні. Ця сформована матриця надходить до нейронних мереж як вхід і вихід визначає ймовірність класів у зображенні.

Під час фази навчання різні рівні функцій визначаються та позначаються як низький, середній і високий. Функції низького рівня включають колір, лінії та контраст. Функції середнього рівня визначають краї та кути, тоді як функції високого рівня ідентифікують клас і конкретні форми чи секції.

Таким чином, CNN зменшує вимоги до обчислювальної потужності та дозволяє обробляти зображення великого розміру. Він чутливий до змін зображення, що може надавати результати з вищою точністю, ніж звичайні нейронні мережі.

Обмеження CNN

Завдяки складній архітектурі можна передбачити об'єкти, обличчя на зображенні з точністю 95%, що перевищує людські можливості, тобто 94%. Однак, навіть незважаючи на його видатні можливості, існують певні обмеження у його використанні. Набори даних із мільярдом параметрів вимагають високого обчислювального навантаження, використання пам'яті та високої потужності

обробки. Використання яких потребує належного обґрунтування.

Використання розпізнавання зображень:

– Безпілотники: безпілотники, оснащені можливостями розпізнавання зображень, можуть забезпечувати автоматичний моніторинг, перевірку та контроль активів, розташованих у віддалених районах, на основі зору.

– Виробництво: перевірка виробничих ліній, регулярна оцінка критичних точок у приміщенні. Контроль якості кінцевої продукції для зменшення браку. Оцінка стану працівників може допомогти виробничим галузям мати повний контроль за різними видами діяльності в системах.

– Автономні транспортні засоби: автономні транспортні засоби з розпізнаванням зображень можуть ідентифікувати дії на дорозі та вживати необхідних заходів. Міні-роботи можуть допомогти галузям логістики знаходити та транспортувати об’єкти з одного місця в інше. Він також зберігає базу даних з історією переміщення продукту, щоб запобігти його втрати або крадіжці.

– Військове спостереження: виявлення незвичайних дій у прикордонних районах і можливості автоматичного прийняття рішень можуть допомогти запобігти проникненню та врятувати життя солдатів.

– Діяльність у лісі: безпілотні літальні апарати можуть стежити за лісом, прогнозувати зміни, які можуть призвести до лісових пожеж, і запобігати браконьєрству. Він також може забезпечити повний моніторинг величезних земель, до яких люди не мають легкого доступу.

3.2 Розробка структурної схеми

Розпізнавання зображень – це завдання ідентифікації цікавих об’єктів на зображенні та розпізнавання, до якої категорії належить зображення. Розпізнавання зображень, розпізнавання фотографій і розпізнавання зображень – це терміни, які використовуються як синоніми.

Коли ми візуально бачимо об’єкт або сцену, ми автоматично

ідентифікуємо об’єкти як різні екземпляри та пов’язуємо їх з окремими визначеннями. Однак візуальне розпізнавання є дуже складним завданням для машин, яке потребує значної потужності обробки.

Розпізнавання зображень за допомогою штучного інтелекту є давньою проблемою досліджень у сфері комп’ютерного зору. Хоча різні методи імітації людського зору розвивалися з часом, спільною метою розпізнавання зображень є класифікація виявлених об’єктів у різні категорії (визначення категорії, до якої належить зображення). Тому його ще називають розпізнаванням об'єктів.

За останні роки машинне навчання, зокрема технологія глибокого навчання, досягло великих успіхів у багатьох задачах комп’ютерного зору та розуміння зображень. Отже, методи глибокого навчання розпізнавання зображень досягають найкращих результатів з точки зору продуктивності (обчислених кадрів за секунду/FPS) і гнучкості. Далі в цій роботі ми розглянемо найефективніші алгоритми глибокого навчання та моделі ШІ для розпізнавання зображень.

Значення та визначення розпізнавання зображень

У сфері комп’ютерного зору такі терміни, як сегментація, класифікація, розпізнавання та виявлення, часто використовуються як взаємозамінні, а різні завдання збігаються. Хоча це здебільшого не проблема, все стає заплутаним, якщо ваш робочий процес вимагає від вас конкретного виконання певного завдання.

Розпізнавання зображень проти комп’ютерного зору

Терміни розпізнавання зображень і комп’ютерний зір часто використовуються як синоніми, але насправді вони різні. Насправді розпізнавання зображень – це застосування комп’ютерного зору, яке часто вимагає виконання кількох завдань комп’ютерного зору, таких як виявлення об’єктів, ідентифікація зображень і класифікація зображень.

Розпізнавання зображень проти локалізації об’єктів

Локалізація об’єктів – це ще одна підмножина комп’ютерного зору, яку

часто плутають із розпізнаванням зображень. Локалізація об’єкта означає визначення розташування одного або кількох об’єктів на зображенні та малювання обмежувальної рамки навколо їхнього периметра. Однак локалізація об'єктів не включає класифікацію виявлених об'єктів.

Розпізнавання зображення проти виявлення зображення

Терміни розпізнавання зображень і виявлення зображень часто використовуються замість один одного. Однак є важливі технічні відмінності.

Виявлення зображень – це завдання взяти зображення як вхідні дані та знайти в ньому різні об’єкти. Прикладом є розпізнавання обличчя, де алгоритми прагнуть знайти шаблони обличчя на зображеннях. Коли ми суворо займаємось виявленням, нам байдуже, чи є виявлені об’єкти значущими. Мета виявлення зображення полягає лише в тому, щоб відрізнити один об’єкт від іншого, щоб визначити, скільки різних об’єктів присутні на зображенні. Таким чином, навколо кожного окремого об'єкта малюються обмежувальні рамки.

З іншого боку, розпізнавання зображень – це завдання ідентифікації об’єктів, що цікавлять зображення, і розпізнавання, до якої категорії чи класу вони належать.

Як працює розпізнавання зображень?

Використання традиційного комп’ютерного зору

Звичайний підхід комп’ютерного бачення до розпізнавання зображень – це послідовність (конвеєр комп’ютерного бачення) фільтрації зображення, сегментації зображення, виділення ознак і класифікації на основі правил.

Однак розробка таких конвеєрів вимагає глибоких знань у обробці зображень і комп’ютерного бачення, багато часу на розробку та тестування з ручним налаштуванням параметрів. Загалом, традиційне комп’ютерне бачення та системи розпізнавання зображень на основі пікселів дуже обмежені, коли йдеться про масштабованість або можливість повторного використання їх у різних сценаріях/розташуваннях.

Використання машинного та глибокого навчання

Розпізнавання зображень за допомогою машинного навчання, з іншого боку, використовує алгоритми для вивчення прихованих знань із набору даних хороших і поганих зразків. Найпопулярнішим методом машинного навчання є глибоке навчання, де в моделі використовуються кілька прихованих шарів нейронної мережі.

Впровадження глибокого навчання в поєднанні з потужним апаратним забезпеченням штучного інтелекту та графічним процесором стало можливим для великих проривів у сфері розпізнавання зображень. Завдяки глибокому навчанню, класифікація зображень і алгоритми розпізнавання облич досягають продуктивності, що перевищує людський рівень, і виявляють об’єкти в реальному часі.

Проте збалансувати продуктивність і ефективність обчислень досить складно. Апаратне та програмне забезпечення з моделями глибокого навчання мають бути ідеально узгоджені, щоб подолати проблеми з вартістю комп’ютерного зору.

Таким чином, можливість завжди використовувати найновіший алгоритм має прямі наслідки щодо витрат: найпотужніший і ефективний алгоритм потребує в рази дешевше обладнання або досягає в рази кращої продуктивності на еквівалентному обладнанні порівняно зі старими алгоритмами.

Протягом багатьох років ми спостерігали значні стрибки в продуктивності алгоритму комп’ютерного зору:

– У 2017 році алгоритм Mask RCNN був найшвидшим детектором об’єктів у реальному часі в еталонному тесті MS COCO з часом висновку 330 мс на кадр.

– Для порівняння, алгоритм YOLOR, який був випущений у 2021 році, досягає часу висновку 12 мс на тому ж тесті, навіть перевершуючи популярні алгоритми глибокого навчання YOLOv4 і YOLOv3.

– А в липні 2022 року алгоритм YOLOv7 навіть значно перевершив YOLOR як за швидкістю, так і за точністю.

Порівняно з традиційним підходом комп'ютерного зору до ранньої обробки зображень 20 років тому, глибоке навчання вимагає лише інженерних знань інструменту машинного навчання, а не досвіду в конкретних областях машинного зору для створення ручних функцій. У той час як ранні методи вимагали величезної кількості навчальних даних, новіші методи глибокого навчання вимагали лише десятків навчальних зразків.

Однак глибоке навчання вимагає ручного маркування даних для анотування хороших і поганих зразків, процес називається анотацією зображення. Процес навчання на основі даних, позначених людьми, називається навчанням під наглядом. Процес створення таких мічених даних для навчання моделей штучного інтелекту вимагає трудомісткої людської роботи, наприклад, щоб анотувати стандартні дорожні ситуації під час автономного водіння.

Процес систем розпізнавання зображень

Є кілька кроків, які є основою роботи систем розпізнавання зображень:

1. Набір даних із навчальними даними. Для моделей розпізнавання зображень потрібні навчальні дані (відео, зображення, фото тощо). Нейронним мережам потрібні навчальні зображення з отриманого набору даних, щоб створити сприйняття того, як виглядають певні класи. Наприклад, модель розпізнавання зображень, яка визначає різні пози (модель оцінки пози), потребує кількох екземплярів різних поз людини, щоб зрозуміти, чим вони відрізняються одна від одної.

2. Навчання нейронних мереж для розпізнавання зображень Зображення зі створеного набору даних подаються в алгоритм нейронної мережі. Це аспект глибокого або машинного навчання створення моделі розпізнавання зображень. Навчання алгоритму розпізнавання зображень дає змогу розпізнаванню зображень за допомогою згорткових нейронних мереж ідентифікувати конкретні класи. Існує кілька добре перевірених фреймворків, які сьогодні широко використовуються для цих цілей.

3. Тестування моделі штучного інтелекту Навчену модель потрібно

протестувати із зображеннями, які не є частиною навчального набору даних. Це використовується для визначення зручності використання, продуктивності та точності моделі. Таким чином, приблизно 80-90% повного набору даних зображень використовується для навчання моделі, а решта даних зарезервовано для тестування моделі. Ефективність моделі вимірюється на основі набору параметрів, які вказують на відсоток впевненості в точності тестового зображення, неправильні ідентифікації тощо.

Розпізнавання зображень за допомогою машинного навчання

До того, як графічні процесори (графічні процесори) стали достатньо потужними, щоб підтримувати масові паралельні обчислювальні завдання нейронних мереж, традиційні алгоритми машинного навчання були золотим стандартом для розпізнавання зображень.

Моделі розпізнавання зображень машинного навчання

Давайте розглянемо три найпопулярніші моделі машинного навчання розпізнавання зображень:

– SVM Support Vector Machines працюють, створюючи гістограми зображень, які містять цільові об’єкти, а також зображень, які їх не містять. Потім алгоритм бере тестове зображення та порівнює навчені значення гістограми зі значеннями різних частин зображення, щоб перевірити відповідність.

– Моделі набору функцій Моделі набору функцій, як-от Scale Invariant Feature Transformation (SIFT) і Maximally stable extremal regions (MSER), працюють, беручи зображення, яке потрібно відсканувати, і зразок фотографії об’єкта, який потрібно знайти, як еталон. Потім модель намагається піксельно зіставити характеристики зразка фотографії з різними частинами цільового зображення, щоб перевірити, чи знайдено збіги.

– Алгоритм Віоли-Джонса. Широко використовуваний алгоритм розпізнавання обличчя ще до CNN (конволюційної нейронної мережі). Він працює шляхом сканування облич і виділення ознак, які потім пропускаються

через підвищуючий класифікатор. Це, у свою чергу, генерує низку посилених класифікаторів, які використовуються для перевірки тестових зображень. Щоб знайти успішний збіг, тестове зображення має генерувати позитивний результат від кожного з цих класифікаторів.

Моделі глибокого навчання розпізнавання зображень

У розпізнаванні зображень використання згорткових нейронних мереж (CNN) також називається глибоким розпізнаванням зображень. CNN не поступаються традиційним методам машинного навчання. CNN не тільки швидші та забезпечують найкращі результати виявлення в розпізнаванні зображень за допомогою машинного навчання, але вони також можуть виявляти кілька екземплярів об'єкта всередині зображення, навіть якщо зображення злегка деформоване, розтягнуте або змінене в іншій формі.

У глибокому розпізнаванні зображень згорткові нейронні мережі навіть перевершують людей у таких завданнях, як класифікація об'єктів за дрібними категоріями, як-от конкретна порода собаки чи вид птахів.

Найпопулярніші моделі глибокого навчання, такі як YOLO, SSD і RCNN, використовують шари згортки для аналізу цифрового зображення чи фотографії. Під час навчання кожен шар згортки діє як фільтр, який навчається розпізнавати певний аспект зображення перед тим, як його передати наступному.

Один шар обробляє кольори, інший шар формує і так далі. Зрештою, сукупний результат усіх цих шарів разом враховується під час визначення відповідності.

Популярні алгоритми розпізнавання зображень

Для розпізнавання зображень або фотографій кілька алгоритмів є кращими. Хоча всі ці алгоритми глибокого навчання, їхній фундаментальний підхід до того, як вони розпізнають різні класи об'єктів, відрізняється. Давайте розглянемо деякі з найпопулярніших моделей розпізнавання зображень на сьогодні:

– Швидший регіональний CNN (швидший RCNN) Faster RCNN (Region-

based Convolutional Neural Network) є найкращим у сімействі алгоритмів розпізнавання зображень R-CNN, включаючи R-CNN і Fast R-CNN. Він використовує регіональну мережу пропозицій (RPN) для виявлення функцій разом із швидким RCNN для розпізнавання зображень, що робить його суттєвим оновленням у порівнянні з попередником (примітка: швидкий RCNN проти швидшого RCNN). Швидший RCNN може обробити зображення за 200 мс, тоді як швидкий RCNN займає 2 секунди або більше.

– Одиночний детектор (SSD) RCNN малюють рамки навколо запропонованого набору точок на зображенні, деякі з яких можуть накладатися. Одноразові детектори (SSD) дискретизують цю концепцію, розділяючи зображення на обмежувальні рамки за замовчуванням у формі сітки з різними пропорціями. Потім він об'єднує карти функцій, отримані в результаті обробки зображення в різних співвідношеннях сторін, щоб природно обробляти об'єкти різних розмірів. Це робить твердотільні накопичувачі дуже гнучкими, точними та простими в навчанні. Реалізація SSD може обробити зображення протягом 125 мс.

– Ти дивишся лише раз (YOLO) YOLO розшифровується як You Only Look Once, і відповідно до назви алгоритм обробляє кадр лише один раз, використовуючи фіксований розмір сітки, а потім визначає, чи містить рамка сітки зображення чи ні. Для цього алгоритм виявлення об'єктів використовує метрику достовірності та кілька обмежувальних рамок у кожній рамці сітки. Однак він не вдається до складності кількох пропорцій чи карт функцій, і, таким чином, хоча це дає результати швидше, вони можуть бути дещо менш точними, ніж SSD. Дуже популярною моделлю YOLO є її третя версія під назвою YOLOv3; остання і найпотужніша версія – YOLOv7. Легкий, оптимізований варіант YOLO під назвою Tiny YOLO може обробляти відео зі швидкістю до 244 кадрів в секунду або 1 зображення за 4 мс.

Як застосовувати моделі розпізнавання зображень

Розпізнавання зображень за допомогою Python

Коли справа доходить до розпізнавання зображень, Python є мовою програмування, яку вибирають більшість спеціалістів із обробки даних та інженерів комп'ютерного зору. Він підтримує величезну кількість бібліотек, спеціально розроблених для робочих процесів ШІ, включаючи виявлення та розпізнавання зображень:

– Крок №1. Щоб налаштувати комп'ютер для виконання завдань розпізнавання зображень Python, вам потрібно завантажити Python і встановити пакети, необхідні для виконання завдань розпізнавання зображень, включаючи Keras.

– Крок №2: Keras – це високорівневий API глибокого навчання для запуску програм ШІ. Він працює на TensorFlow /Python і допомагає кінцевим користувачам розгортати програми машинного навчання та ШІ за допомогою легкого для розуміння коду.

– Крок №3. Якщо ваш комп'ютер не має відеокарти, ви можете використовувати безкоштовні екземпляри GPU онлайн на Google Colab. Для цілей класифікації тварин існує добре позначений набір даних, відомий як « Animals-10 », який ви можете знайти на Kaggle. Набір даних можна завантажити абсолютно безкоштовно.

– Крок №4.

Після того, як ви отримаєте онлайн-набір даних від Kaggle, отримавши маркер API, ви можете розпочати кодування на Python після повторного завантаження необхідних файлів на Диск Google.

Щоб отримати докладніші відомості про впровадження на конкретній платформі, у кількох добре написаних статтях в Інтернеті крок за кроком описано процес налаштування середовища для штучного інтелекту на вашому комп'ютері чи Colab, яким ви можете користуватися.

Крім того, ознайомтеся з корпоративною платформою розпізнавання

зображень Viso Suite, щоб створювати, розгортати та масштабувати реальні програми без написання коду. Це дозволяє уникнути труднощів з інтеграцією, заощаджує кошти на численні інструменти та є високорозширюваним.

Навчання спеціальної моделі

Спеціальна модель для розпізнавання зображень – це модель ML, яка була спеціально розроблена для конкретного завдання розпізнавання зображень. Це може включати використання спеціальних алгоритмів або модифікації існуючих алгоритмів для покращення їх продуктивності на зображеннях (наприклад, перенавчання моделі).

Хоча попередньо навчені моделі забезпечують надійні алгоритми, навчені на мільйонах точок даних, є багато причин, чому ви можете створити спеціальну модель для розпізнавання зображень. Наприклад, у вас може бути набір даних зображень, який сильно відрізняється від стандартних наборів даних, на яких навчаються поточні моделі розпізнавання зображень. У цьому випадку спеціальну модель можна використовувати для кращого вивчення функцій ваших даних і підвищення продуктивності. Крім того, ви можете працювати над новою програмою, де поточні моделі розпізнавання зображень не досягають необхідної точності чи продуктивності.

Створення спеціальної моделі на основі певного набору даних може бути складним завданням і потребує збору високоякісних даних і анотацій зображень. Це вимагає хорошого розуміння як машинного навчання, так і комп’ютерного зору.

API розпізнавання зображень (хмара) проти Edge AI

Інтерфейси API забезпечують простий спосіб розпізнавання зображень за допомогою виклику хмарної служби API, такої як Amazon Rekognition (AWS Cloud). Подібним чином можна легко використовувати API для розпізнавання об’єктів на зображеннях за допомогою Google Vision API (Google Cloud) для таких завдань, як розпізнавання об’єктів чи облич, розпізнавання тексту чи розпізнавання рукописного тексту.

API розпізнавання зображень, такий як API виявлення об’єктів TensorFlow, є потужним інструментом для розробників, який дозволяє швидко створювати та розгортати програмне забезпечення розпізнавання зображень, якщо варіант використання дозволяє розвантажувати дані (надсилати візуальні елементи на хмарний сервер). Використання API для розпізнавання зображень використовується для отримання інформації про саме зображення (класифікація зображення або ідентифікація зображення) або об’єкти, що містяться (виявлення об’єкта).

Чисто хмарні API комп’ютерного зору корисні для створення прототипів і рішень нижчого масштабу, які дозволяють розвантажувати дані (конфіденційність, безпека, законність), не є критично важливими (підключення, пропускна здатність, надійність) і не працюють у реальному часі (затримка, дані обсяг, високі витрати). Щоб подолати ці межі суто хмарних рішень, останні тенденції розпізнавання зображень зосереджуються на розширенні хмари шляхом використання Edge Computing із машинним навчанням на пристрої.

Щоб дізнатися, як працюють API розпізнавання зображень, який з них вибрати, а також про обмеження API для завдань розпізнавання, я рекомендую вам переглянути наш огляд найкращих платних і безкоштовних API комп’ютерного зору.

Хоча API комп’ютерного бачення можна використовувати для обробки окремих зображень, системи Edge AI використовуються для виконання завдань розпізнавання відео в режимі реального часу шляхом переміщення машинного навчання в безпосередній близькості від джерела даних (Edge Intelligence).

Це забезпечує обробку зображень штучним інтелектом у режимі реального часу, оскільки візуальні дані обробляються без вивантаження даних (завантаження даних у хмару), що забезпечує більш високу продуктивність висновків і надійність, необхідні для систем виробничого рівня.

Платформа ШІ розпізнавання зображень

Якщо ви не хочете починати з нуля та використовувати попередньо

налаштовану інфраструктуру, ви можете перевірити нашу платформу комп'ютерного зору Viso Suite. Корпоративний пакет надає популярне програмне забезпечення для розпізнавання зображень із відкритим вихідним кодом із понад 60 найкращих попередньо навчених моделей. Він також забезпечує збір даних, маркування зображень і розгортання на периферійних пристроях – усе готове та з можливостями без використання коду. Ця платформа AI Vision дозволяє створювати та працювати з програмами в режимі реального часу, використовувати нейронні мережі для завдань розпізнавання зображень та інтегрувати все з існуючими системами.

Для чого використовується розпізнавання зображень?

У всіх галузях промисловості технологія розпізнавання зображень ШІ стає все більш необхідною. Його застосування забезпечує економічну цінність у таких галузях, як охорона здоров'я, роздрібна торгівля, безпека, сільське господарство та багатьох інших.

Програма розпізнавання зображень для аналізу обличчя

Аналіз обличчя є видатним додатком для розпізнавання зображень. Сучасні методи ML дозволяють використовувати відео з будь-якої цифрової камери або веб-камери. У таких програмах програмне забезпечення для розпізнавання зображень використовує алгоритми ШІ для одночасного виявлення обличчя, оцінки пози обличчя, вирівнювання обличчя, розпізнавання статі, виявлення посмішки, оцінки віку та розпізнавання обличчя за допомогою глибокої згорткової нейронної мережі. Аналіз обличчя за допомогою комп'ютерного зору дозволяє системам аналізувати відеокадр або фотографію, щоб розпізнавати особу, наміри, емоційний стан і стан здоров'я, вік або етнічну приналежність. Деякі інструменти розпізнавання фотографій для соціальних медіа навіть націлені на кількісну оцінку рівня сприйнятої привабливості за допомогою оцінки. Інші завдання, пов'язані з розпізнаванням обличчя, включають ідентифікацію зображення обличчя, розпізнавання обличчя та перевірку обличчя, що включає методи обробки зору для пошуку та зіставлення

виявленого обличчя із зображеннями облич у базі даних. Методи розпізнавання глибокого навчання дозволяють ідентифікувати людей на фотографіях або відео навіть у віці або в умовах складного освітлення. Одна з найпопулярніших програмних бібліотек із відкритим вихідним кодом для створення програм розпізнавання облич зі штучним інтелектом називається DeepFace, яка здатна аналізувати зображення та відео.

Розпізнавання зображень для аналізу медичних зображень

Технологія візуального розпізнавання широко використовується в медичній промисловості, щоб комп'ютери могли розуміти зображення, які регулярно отримують протягом курсу лікування. Аналіз медичних зображень стає дуже прибутковою частиною штучного інтелекту. Наприклад, є багато робіт щодо ідентифікації меланоми, смертельного раку шкіри. Програмне забезпечення для розпізнавання зображень із глибоким навчанням дозволяє відстежувати пухлини протягом тривалого часу, наприклад, для виявлення аномалій під час сканування раку молочної залози.

Розпізнавання зображень для спостереження за тваринами

Системи розпізнавання зображень сільськогосподарського машинного навчання використовують нові методи, навчені для визначення типу тварини та її дій. Програмне забезпечення для розпізнавання зображень штучного інтелекту використовується для моніторингу тварин у сільському господарстві, де худобу можна дистанційно контролювати для виявлення хвороб, виявлення аномалій, дотримання вказівок щодо добробуту тварин, промислової автоматизації тощо.

Виявлення шаблонів і об'єктів

Технології штучного інтелекту розпізнавання фотографій і розпізнавання відео корисні для ідентифікації людей, візерунків, логотипів, об'єктів, місць, кольорів і форм. Можливість налаштування розпізнавання зображень дозволяє використовувати його в поєднанні з кількома програмами. Наприклад, після того як програма розпізнавання зображень спеціалізується на виявленні людей у відеокадрі, її можна використовувати для підрахунку людей, популярної

програми комп’ютерного зору в роздрібних магазинах.

Автоматизована ідентифікація рослинного зображення

Ідентифікація рослин на основі зображень отримала швидкий розвиток і вже використовується в дослідженнях і випадках використання природокористування. Недавня дослідницька стаття проаналізувала точність ідентифікації ідентифікації зображень для визначення сімейства рослин, форм росту, форм життя та регіональної частоти. Інструмент виконує пошук зображень, використовуючи фотографію рослини з програмним забезпеченням зіставлення зображень, щоб запитувати результати в онлайн-базі даних. Результати вказують на високу точність розпізнавання штучним інтелектом, де 79,6% із 542 видів на приблизно 1500 фотографіях були правильно ідентифіковані, тоді як родина рослин була правильно ідентифікована для 95% видів.

Розпізнавання зображень їжі

Розпізнавання зображень із глибоким навчанням різних типів їжі використовується для комп’ютерної оцінки дієти. Тому було розроблено програмне забезпечення для розпізнавання зображень, щоб підвищити точність поточних вимірювань споживання їжі шляхом аналізу зображень їжі, зроблених мобільними пристроями та поширених у соціальних мережах. Таким чином, програма розпізнавання зображень використовується для онлайн-розпізнавання шаблонів на зображеннях, завантажених студентами.

Розпізнавання пошуку зображень

Пошук зображень розпізнавання, або візуальний пошук, використовує візуальні функції, отримані з глибокої нейронної мережі для розробки ефективних і масштабованих методів для пошуку зображень. Метою випадків використання візуального пошуку є пошук зображень на основі вмісту для онлайн-додатків розпізнавання зображень. Для вирішення цієї складної проблеми дослідники розробили масштабний візуальний словник із навчального набору функцій нейронної мережі.

Типові програми розпізнавання зображень:

– Застосування №1: промислове розпізнавання зображень для виявлення дефектів і прогнозного аналізу на виробництві.

– Застосування №2: Автоматизоване виявлення вторгнень у розподілених системах безпеки та спостереження.

– Застосування №3: Системи розпізнавання зображень для аналізу корозії та виявлення витоків у нафті та газу.

– Застосування №4: програмне забезпечення для розпізнавання фотографій для виявлення шахрайства у сфері страхування.

– Застосування №5: підрахунок людей і аналіз натовпу в реальному часі в розумних містах.

– Застосування №6: Програма розпізнавання зображень для виявлення зброї (ножі, пістолети).

Структурна схема системи наведена на рисунку 3.1. Структурно система складається з наступних частин:

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 номера автомобілів, а також номера усіх автомобілів, які перетинали кордон приміщення установи, який встаткований відеокамерами спостереження.

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

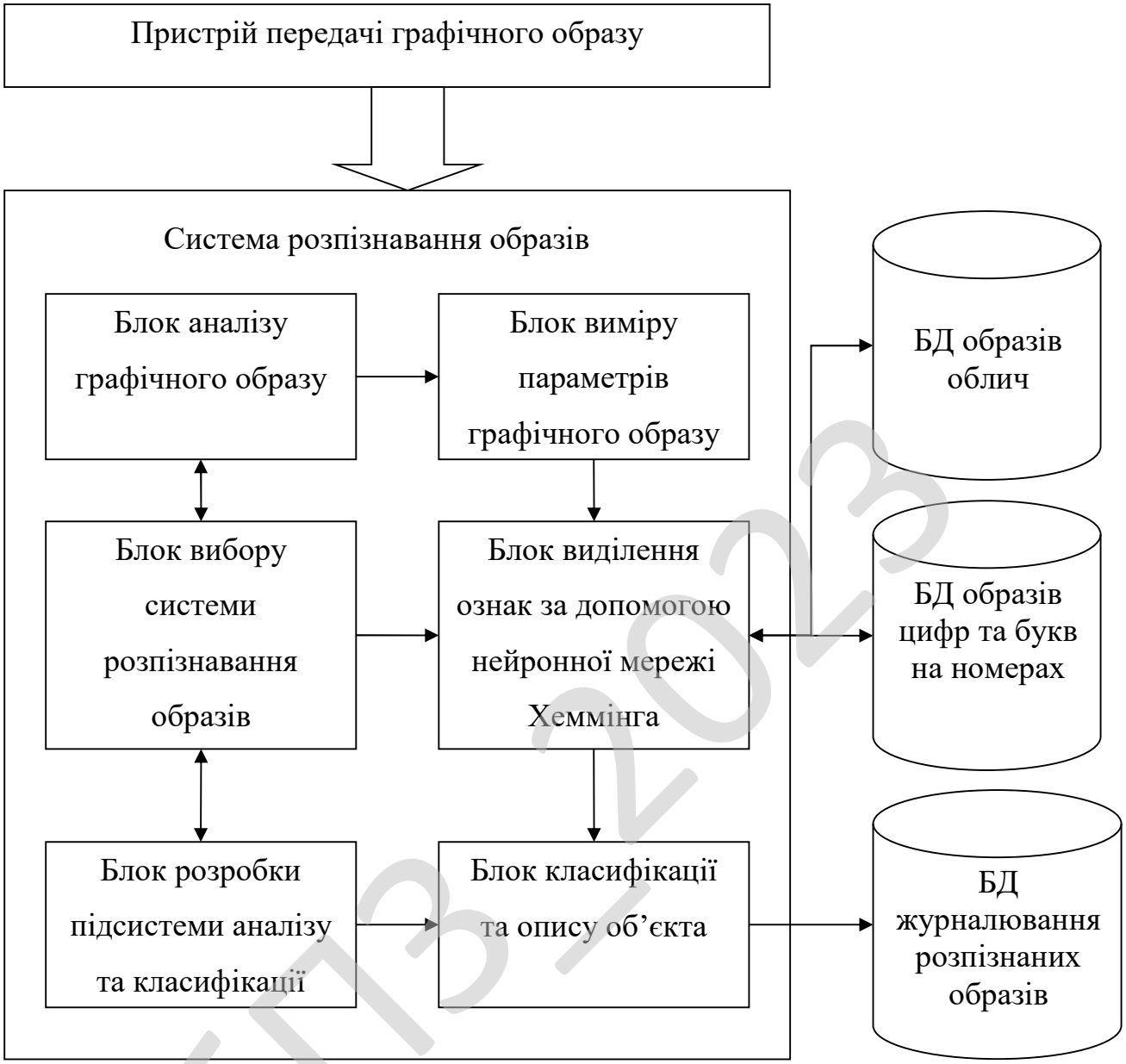


Рисунок 3.1 – Структурна схема системи

5. Система розпізнання образів, яка складається з наступних блоків:

– Блок читання картинки з відеокамери. Він призначений для читання картинок з відеокамери й подання даних на блок аналізу та виділення ознак за допомогою нейронної мережі Хеммінга.

– Блок аналізу та виділення ознак за допомогою нейронної мережі Хеммінга. 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 за допомогою нейронної мережі Хеммінга, розподілити куди заносити отримані дані, у поля бази даних, які відповідають за осіб, або у поля бази даних, які відповідають за машини.

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.

Так як розпізнавання образів відбувається за допомогою алгоритму нейронної мережі Хеммінга, то наведемо цей алгоритм.

Алгоритм нейронної мережі Хеммінга

Опис реалізації алгоритму.

1. Реалізація мережі. Вхідні символи.

Для визначення загальних параметрів НМ – кілька констант:

– Розмір сторони вихідного зображення.

– Розмір сторони зображення для розпізнавання.

– Скільки входів.

– Скільки образів.

– Вага синапсів другого шару.

Розмір вхідного образа (bmp-файл) повинен бути 100x100 точок, а перед обробкою він приводиться до розміру 40x40. Таким чином, кількість входів мережі – 1600 (один вхід – одна точка зображення; параметр N). Кількість

виходів M збігається з кількістю цифр і рівняється 10. Вага негативного зворотного зв'язка другого шару був прийнятий рівним – 0.05.

Кожний нейрон представляється у вигляді простої структури. Шар – це просто масив нейронів. Є два шари.

Тут уже нема рації зберігати значення окремих ваг, так як вони не міняються протягом всієї роботи мережі.

2. Навчання мережі.

Після процедури навчання списки вагових коефіцієнтів синапсів першого шару будуть містити образи еталонних символів.

Алгоритм навчання мережі Хеммінга, адаптований для даного завдання, виглядає так:

2.1. Вибирається і-й вхідний образ.

2.2. Зображення локалізується й приводиться до потрібного масштабу.

2.3. Образ поточечно подається на входи і-го нейрона. Якщо k-а точка образа чорна, то ваги k-го входу привласнюється значення 0.5, у противному випадку – 0.5.

2.4. Перехід на Крок 2.1, поки не вичерпані всі еталонні образи.

3. Локалізація й масштабування зображення.

Для успішної роботи потрібно з'ясувати точний розмір і місце розташування образа. Після локалізації – приводимо образ до розміру 40*40 (масштабування). Ці операції виконуються як на етапі навчання мережі, так і на етапі розпізнавання.

Для еталона, через відсутність перешкод, локалізацію провести дуже просто: досить послідовно переглянути всі точки образа й знайти границі образа:

У даній реалізації використовується стандартний алгоритм зворотного масштабування без інтерполяції.

4. Перекручування зображення.

Щоб не розпізнавати еталонні образи – уведемо перекручування. Додавання N-процентного шуму.

5. Алгоритм розпізнавання.

Загальний алгоритм розпізнавання для мережі Хеммінга складається із чотирьох частин:

5.1. Подача розпізнаваного образа на входи мережі.

5.2. Передача даних з першого шару на другий.

5.3. Обробка даних другим шаром.

5.4. Вибір розпізнаного образа.

5.1. Алгоритм роботи першого етапу виглядає так:

5.1.1. Вибирається черговий нейрон.

5.1.2. Обнуляється його вихід.

5.1.3. Локалізується й приводиться зображення до потрібного масштабу.

5.1.4. Локалізований образ поточечно подається на входи і-го нейрона. Якщо k-я точка образа чорна, то до значення виходу додається значення ваги k-го входу, у противному випадку це значення віднімається.

5.1.5. Значення виходу пропускається через функцію лінійного порога.

5.1.6. Перехід на Крок 5.2, поки не вичерпані всі нейрони першого шару.

5.2. Тепер треба передати дані з виходів першого шару на входи другого й у список результатів попереднього проходу розпізнавання.

5.3. Тепер може починати роботу другий шар. Розглянемо алгоритм його роботи:

5.3.1. Обнуляється лічильник ітерацій.

5.3.2. Запам'ятовуються виходи нейронів у списку результатів попереднього проходу.

5.3.3. Перебирається мережа по нейронах.

5.3.4. і-й нейрон посилає свій вихід на і-й вхід кожного нейрона.

5.3.5. Кожний нейрон, приймаючи значення, накопичує їх, попередньо помноживши на коефіцієнт е (крім випадку, коли нейрон приймає своє ж значення – тоді він повинен помножити його на 1).

5.3.6. Перехід на Крок 5.3.4, поки не будуть оброблені всі нейрони.

5.3.7. Накопичені суми кожний нейрон посилає на свій вихід.

5.3.8. Перехід на Крок 5.3.2, поки виходи нейронів на поточній ітерації не збіжаться з виходами на попередній або поки лічильник числа ітерацій не перевищить деяке значення.

Теоретично, другий шар повинен працювати поки його виходи не стабілізуються, але на практиці кількість ітерацій штучно обмежують.

5.4. Останній Крок – вибір з нейронів другого шару з найбільшим значенням на виході. Його номер і є номер розпізнаного образа.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема системи включає в себе наступні функціональні блоки:

1. Блок розпізнання образів.
2. Блок розпізнавання номерів автомобілів.
3. Блок розпізнавання людей.
4. Блок відеодетекції.
5. Блок динамічного спостереження за порушником.
6. Блок подання сигналу тривоги.
7. Блок документування подій на об’єкті.

Розглянемо ці блоки більш детально.

Блок розпізнання образів

Виділимо етапи при рішенні завдання розпізнавання зображень:

– Сприйняття поля зору.

– Сегментація.

– Нормалізація виділених об'єктів.

– Розпізнавання.

Виходячи із цього, використовуються наступні основні принципи:

– Принцип цілісності – розпізнаваний об'єкт розглядається як єдине ціле, що складається зі структурних частин, зв'язаних між собою просторовими відносинами.

– Принцип двунаправленості – створення моделі ведеться від зображення до моделі й від моделі до зображення.

– Принцип передбачення. Полягає у формуванні гіпотези про зміст зображення.

– Принцип цілеспрямованості, що включає сегментацію зображення й спільну інтерпретацію його частин.

– Нічого не робити без процедури розуміння (сприйняття поля зору).

– Принцип максимального використання моделі проблемного середовища, використання заздалегідь відомих, апріорних параметрів.

Етап інтерпретації зображення не позначений чіткими границями й включається частково в процес сегментації й остаточно завершується на етапі розпізнавання.

Природно задатися питанням: а чи не можна брати зображення й послідовно порівнювати його з еталонами по ряду яких-небудь ознак? Але отут виникає ряд проблем і складностей:

– Тло. Як правило, зображення пред'являються на складному динамічному тлі.

– Орієнтація. Зображення еталона й вхідних зображень відрізняються положенням у поле зору.

– Перешкоди. Вхідні зображення не збігаються з еталонами за рахунок випадкових і локальних перешкод.

– Висвітлення. Відмінності вхідних і еталонних зображень виникає за рахунок зміни освітленості, підсвічування.

– Перетворення. Еталони й зображення можуть відрізняти складні геометричні перетворення.

Блок розпізнавання номерів автомобілів

Система використовується автоматичної реєстрації й розпізнавання автомобільних номерів на контрольно-пропускних пунктах на підприємствах, платних стоянках і гаражних комплексах, на постах ДПС. Захист із системою контролю доступу дозволяє автоматизувати контрольно-пропускний режим. Система дозволяє вести розшук автомобілів у викраденні.

Функціональні можливості:

– автоматична реєстрація автомобільних номерів і розпізнавання;

– збереження номера й відеозапису проїзду транспортного засобу в базі даних із вказівкою дати й часу;

– автоматичне зіставлення автомобільного номера з наявними базами даних (наприклад, автомобілів, що мають право допуску на територію підприємства, або автомобілів, що перебувають у викраденні) і видача відповідного повідомлення операторові;

– автоматизація контрольно-пропускного режиму при використанні із пристроями контролю доступу;

– пошук у базі даних за номером, датою, часом;

– формування звітів за номером, датою, часом.

Характеристики:

– Ширина контрольованої зони проїзної частини – від 1,5 до 3,5 метрів.

– Глибина зони контролю – 10 метрів (для КПП: 2-4 метри).

– Кількість оброблюваних кадрів (з обліком 100 % потокової завантаженості на всіх камерах):

– канал – 25 к/с, до 150 км/год;

– каналу й більше – 16 к/с (сумарне кіл-у кадрів), до 75 км/ч.

– Імовірність розпізнавання – не менш 90%.

– Припустимі кути установки відеокамери.

– По вертикалі – не більше 30°.

– По горизонталі – не більше 20°.

– Крен номерного знака ±15°.

– Освітленість – не менш 50 люкс.

– Розпізнавані номери – 7 типів (тло білий і жовтий).

– Вимога до бази даних – підтримка інтерфейсу ADO.

Блок розпізнавання людей

Система захвата осіб дозволяє виділяти тільки особи людей, вибирати найбільш виразне зображення з декількох варіантів і зберігати їх у базі даних. Створення бази осіб людей на прохідних підприємств, у місцях масового скупчення людей (культурно-розважальні центри, вокзали, стадіони й т.д.) полегшує роботу з архівами при розслідуванні позаштатних ситуацій.

Далі система розпізнавання особи ідентифікує особистість і автоматизує пошук зображень у базах даних.

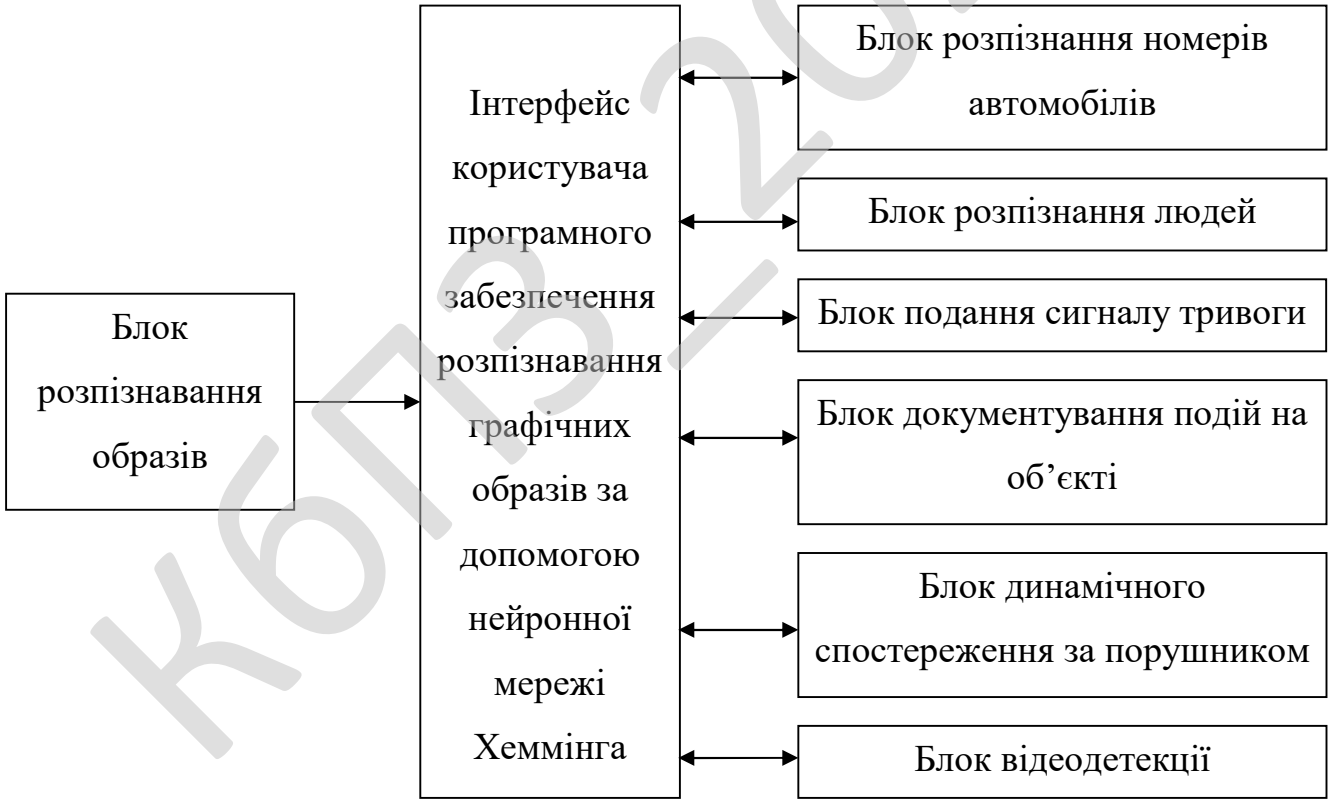


Рисунок 3.2 – Функціональна схема системи

Блок відеодетекції

Виявлення переміщення в зоні спостереження (відеодетекція). Такі пристрої часто вбудовуються в стандартні мультиплексори. При цьому оператор може задавати зону на екрані монітора, рух у якій викликає сигнал тривоги.

Блок динамічного спостереження за порушником

Системи динамічної цілевказівки аналізують зміни координат характерних точок об'єкта, наприклад центра ваги, кольору.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

Блок подання сигналу тривоги

Відеокамера використовується разом з технічним засобом охорони для підтвердження факту спрацьовування останнього.

Блок документування подій на об’єкті

Матеріал відеоархівів може виявитися корисним як доказова база при розслідуванні несанкціонованих дій.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3. Спочатку завантажується процес виведення головного вікна програми.

Він взаємодіє з наступними процесами:

– Процес розпізнання образів.

– Процес навчання нейронної мережі.

– Процес роботи з базою даних.

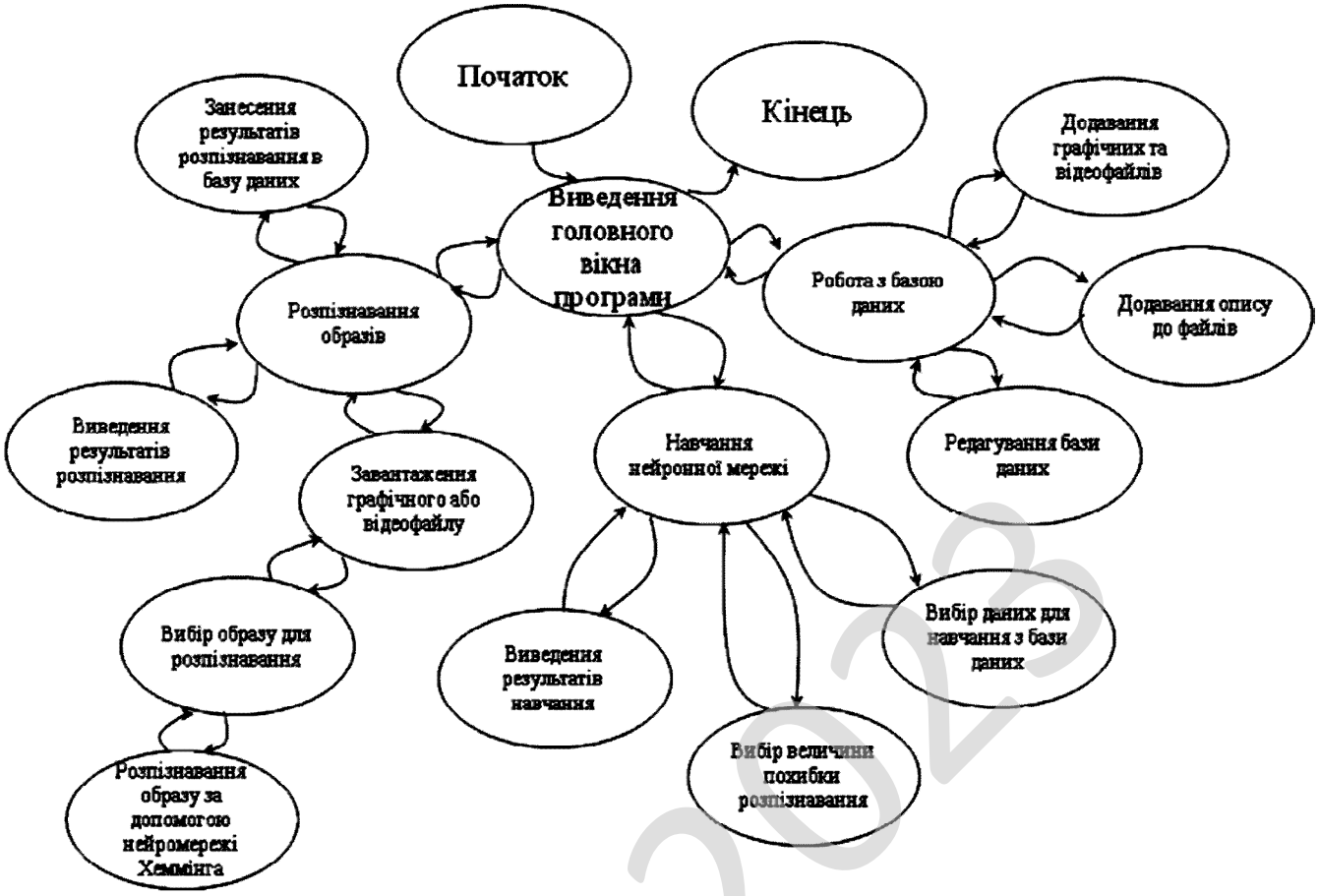


Рисунок 3.3 – Діаграма взаємодії процесів

Процес розпізнавання образів взаємодіє з наступними процесами:

– Процес занесення результатів розпізнання у базу даних.

– Процес виведення результатів розпізнання.

– Процес завантаження графічного або відео файлу.

Процес завантаження графічного або відео файлу взаємодіє з процесом вибору образу для розпізнавання, який, у свою чергу, взаємодіє з процесом розпізнавання образу за допомогою нейромережі Хеммінга.

Процес навчання нейронної мережі взаємодіє з наступними процесами:

– Процес вибору даних для навчання з бази даних.

– Процес вибору величини похибки розпізнавання.

– Процес виведення результатів розпізнавання.

Процес роботи з базою даних взаємодіє з наступними процесами:

– Процес додавання графічних та аудіо файлів.

– Процес додавання опису до файлів.

– Процес редагування бази даних.

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

На рисунку 4.1 зображено блок-схему основної програми. Виконання основної програми складається з наступних кроків:

Крок 1. Виведення основного вікна програми на екран.

Крок 2. Якщо користувач обирає внесення файлів у базу даних, то відбувається завантаження графічних та відеофайлів у базу даних програми.

Крок 3. Якщо користувач обирає дію «Навчання нейромережі», відбуваються кроки 4-9.

Крок 4. Викликається підпрограма навчання нейронної мережі на образах наявних в базі даних.

Крок 5. Якщо користувач обирає дію «Розпізнати образ», то відбувається завантаження зображення або відеокадру.

Крок 6. Виведення завантаженого зображення на екран.

Крок 7. Визначення області розпізнавання на вибраному зображенні.

Крок 8. Виклик підпрограми розпізнавання образів нейромережею Хеммінга.

Крок 9. Виведення результатів розпізнавання образу на екран.

Крок 10. Якщо користувач обирає дію «Збереження результатів розпізнавання», відбуваються кроки 11-12.

Крок 11. Запис зображення та його опису в базу даних.

Крок 12. Запис результатів розпізнавання в базу даних.

Крок 13. Якщо користувач обирає дію «Редагування бази даних», запускається підпрограма редагування бази даних програми.

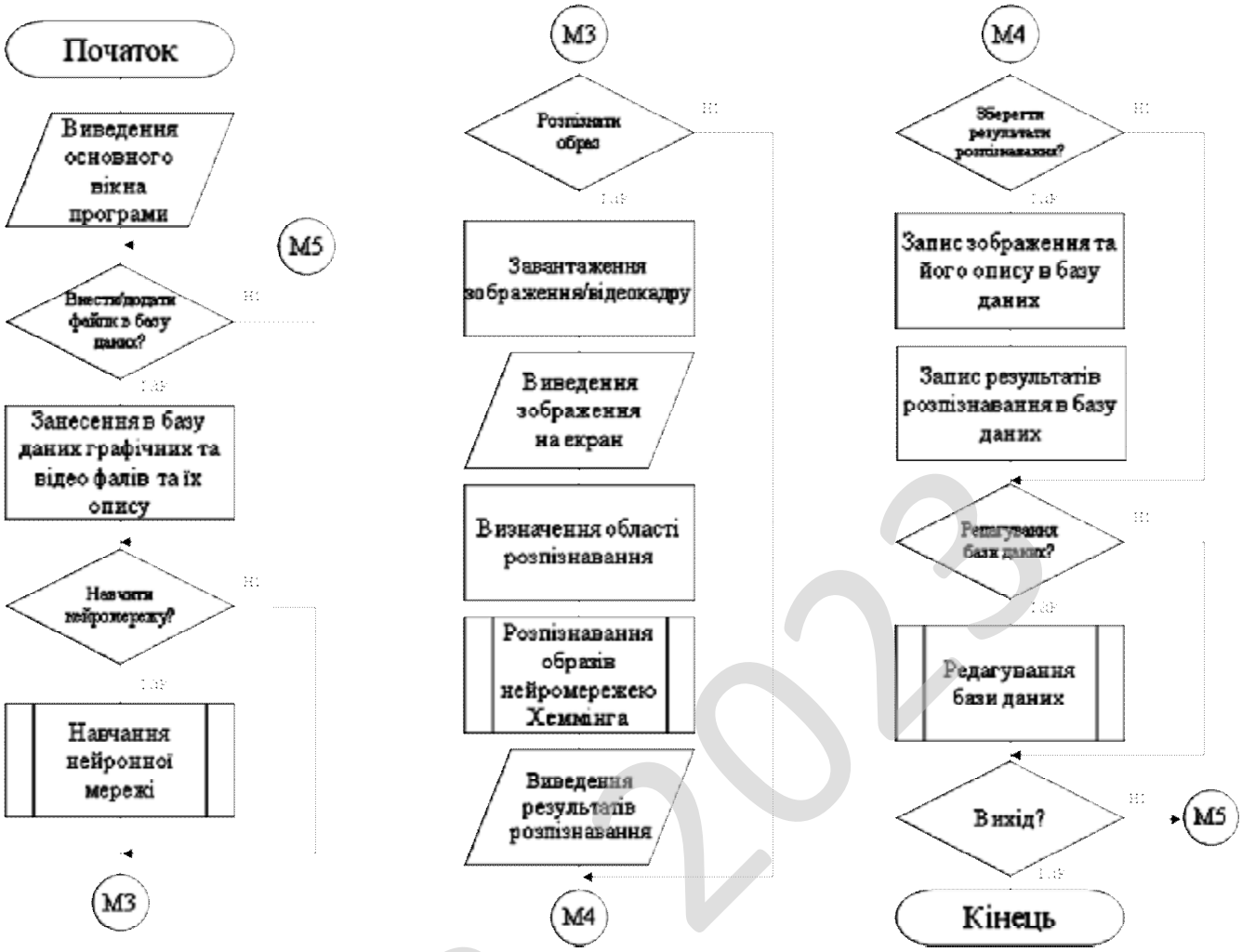


Рисунок 4.1 – Блок-схема основної програми

У розробленому програмному забезпеченні для розпізнавання графічних образів використовується нейромережа Хеммінга. Розглянемо детальніше принципи її реалізації.

Нейронна мережа Хеммінга

Нейронна мережа, що складається із двох шарів, кожний з яких містить число нейронів *M*, рівне числу зберігаємих образів. Нейрони першого шару мають *N* зв'язків, з'єднаних із входами мережі (утворюючими фіктивний нульовий шар). Нейрони другого шару зв'язані між собою негативними зворотними зв'язками. Єдиний позитивний зворотний зв'язок кожний нейрон має із власним виходом.

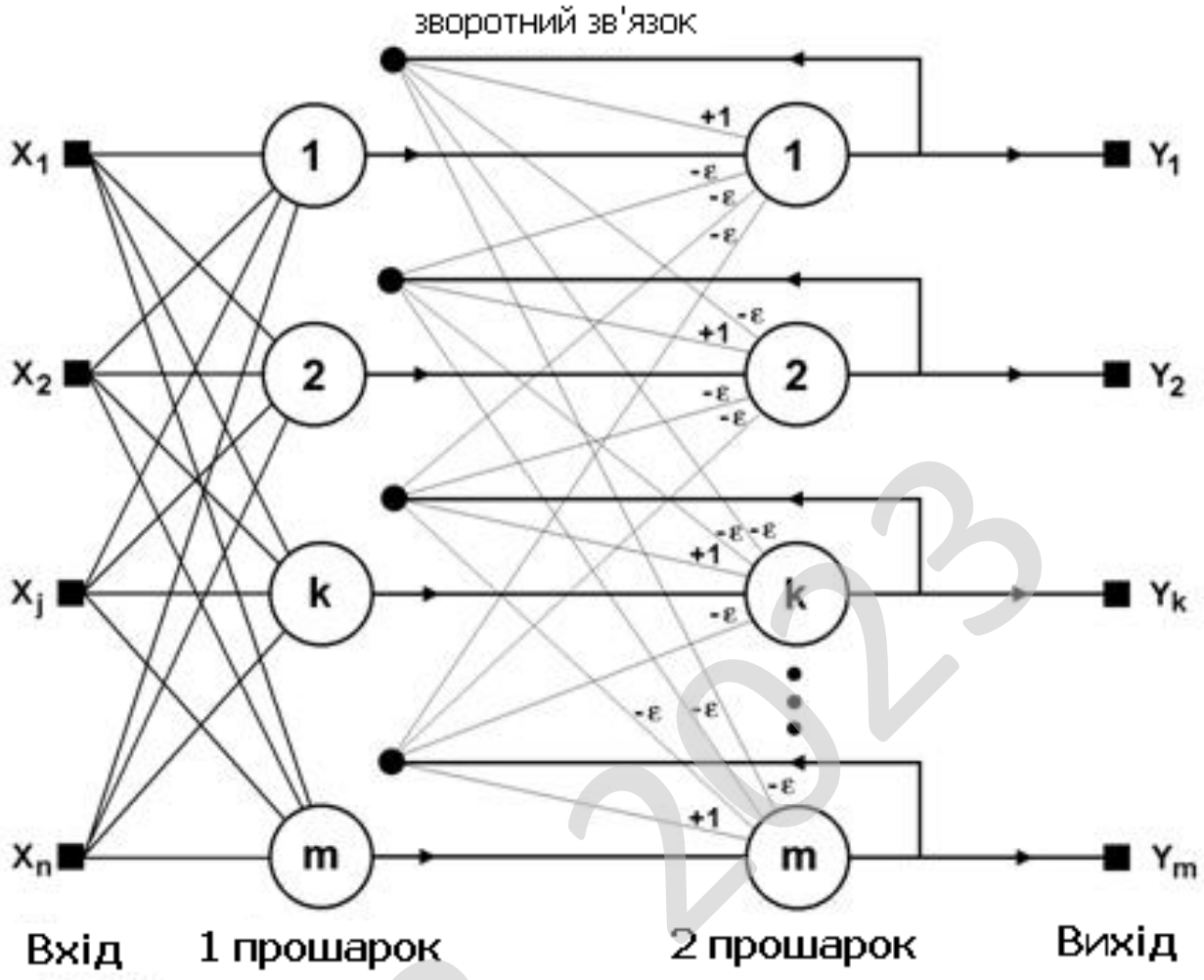


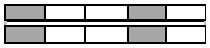
Рисунок 4.2 – Схема нейронної мережі Хеммінга

Нейронні мережі Хеммінга можна використовувати для реалізації асоціативної пам'яті в тих випадках, коли немає необхідності, щоб мережа видавала на виході образ у явному видгляді, а досить тільки його номера (або коду). У порівнянні з мережею Хопфілда мережа Хеммінга має менші витрати на пам'ять і обсяг необхідних обчислень.

Принцип роботи мережі Хеммінга заснований на обчисленні відстані Хеммінга від вхідного образу до всіх образів, збережених мережею. Під відстанню Хеммінга розуміється число різних бітів у двох бінарних векторах. Мережа повинна вибрати образ із мінімальною Хеммінговою відстанню до вхідного. При цьому на виході мережі формується не сам цей образ, а

виконується активізація виходу, асоційованого з ним.

У мережі Хеммінга два шари – перший і другий шари складаються з *m* нейронів і *m* дорівнює числу зразків. Нейрони першого шару мають по *n* вхідних синапсов, де *n* – розмірність вхідних векторів. Нейрони другого шару зв'язані між собою зворотними, негативними зв'язками. Зворотний зв'язок від аксона на власника нейрона дорівнює +1. Суть роботи полягає в знаходженні відстані Хеммінга від тестируемого зразка до всіх зразків. Відстанню Хеммінга називається число різних бітов у двох бінарних векторах.

 – відстань Хеммінга дорівнює 0.

 – відстань Хеммінга дорівнює 2.

Рисунок 4.3 – Ілюстрація поняття відстань Хеммінга

Мережа повинна вибрати зразок з мінімальною відстанню Хеммінга до поданого вхідного сигналу – у результаті активується один вихід, відповідальний за даний еталонний зразок.

При ініціалізації мережі ваговим коефіцієнтам першого шару й порогу активаційної функції привласнюються наступні значення:

$$w_{ik} = \frac{x_i^k}{2},\ i=0...n-1,\ k=0...m-1$$

$$T_k = n / 2,\ k = 0...m-1$$

де x_i^k – і-ий елемент k-ого зразка.

Вагові коефіцієнти гальмуючих синапсів у другому шарі беруть рівними деякій величині $0 < \varepsilon < 1/m$. Синапс нейрона, пов'язаний з його ж аксоном має вагу +1.

Алгоритм роботи мережі Хеммінга наступний:

1. На входи мережі подається невідомий вектор $\mathbf{X} = \{x_i : i=0... n-1\}$, виходячи з якого розраховуються стани нейронів першого шару (верхній індекс у дужках вказує номер шару):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j \text{ , } j=0...m-1.$$

2. Після цього отриманими значеннями ініціалізуються значення аксонів другого шару:

$$y_j^{(2)} = y_j^{(1)}, j = 0...m-1.$$

3. Обчислюються нові стани нейронів другого шару:

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), k \neq j, j = 0...m-1 \text{ ,}$$

і значення їхніх аксонів:

$$y_j^{(2)}(p+1) = f\left[s_j^{(2)}(p+1)\right], j = 0...m-1$$

Активаційна функція f має вигляд порога, причому величина F повинна бути досить великою, щоб будь-які можливі значення аргументу не приводили до насичення.

4. Перевірка, чи змінилися виходи нейронів другого шару за останню ітерацію. Якщо так – перейди до кроку 3. Інакше – завершення роботи.

З оцінки алгоритму видно, що роль першого шару нейронів досить умовна: скориставшись один раз на кроці 1 значеннями його вагових коефіцієнтів, мережа більше не звертається до нього, тому перший шар може бути взагалі виключений з мережі (просто замінений на матрицю вагових коефіцієнтів).

На рисунку 4.4 наведено блок-схема підпрограми розпізнавання графічних образів нейронною мережею Хеммінга, розроблену в даному магістерському проекті. Робота підпрограми складається з наступної послідовності кроків:

Крок 1. Завантаження зображення для розпізнавання нейромережею.

Крок 2. Задання величини похибки розпізнавання ε та максимальної

кількості ітерацій S, які будуть використовуватися в умові виходу з циклу розпізнавання.

Крок 3. Для кожного нейрону першого шару виконуються кроки 4-12.

Крок 4. Вибирається і-й нейрон першого шару.

Крок 5. Обнуляється вихід і-го нейрону.

Крок 6. Образ на зображенні локалізується й приводиться до потрібного масштабу.

Крок 7. Локалізований образ попіксельно подається на входи і-го нейрона.

Крок 8. Для кожного пікселя розпізнаваємого зображення виконується послідовність кроків 9-12.

Крок 9. Виконується перевірка значення кольру k-того пікселя.

Крок 10. Якщо колір k-того пікселя чорний, то до значення виходу і-го нейрона додається значення ваги k-го входу.

Крок 11. Інакше, від значення виходу і-го нейрона віднімається значення ваги k-го входу.

Крок 12. Значення виходу і-го нейрону пропускається через функцію активації.

Крок 13. Обнуляється лічильних ітерацій j=0 циклу розпізнавання.

Крок 14. Запам’ятовуються значення виходів нейронів у списку результатів попереднього проходу.

Крок 15. Для кожного нейрону другого шару виконуються кроки 16-18.

Крок 16. і-й нейрон посилає свій вихід на і-й вхід кожного нейрону мережі.

Крок 17. Якщо нейрон посилає значення сам собі, то значення виходу і-го нейрона обчислюється за формулою:

$$Y = Y + W_k * e,$$

де Y – значення виходу і-го нейрона;

W_k – значення ваги k-го входу;

e – коефіцієнт підсилення.

Крок 18. Інакше, значення виходу і-го нейрона обчислюється за формулою:

$$Y = Y + W_k.$$

Крок 19. Накопичені суми кожний нейрон посилає на свій вихід.

Крок 20. Обчислюється значення похибки прозпізнавання на поточному кроці циклу розпізнавання, за наступною формулою:

$$R := h – h',$$

де h – виходи нейронів на і-тій ітерації;

h' – виходи нейронів на (і-1) ітерації

Крок 21. Збільшується лічильних ітерацій $j = j + 1$.

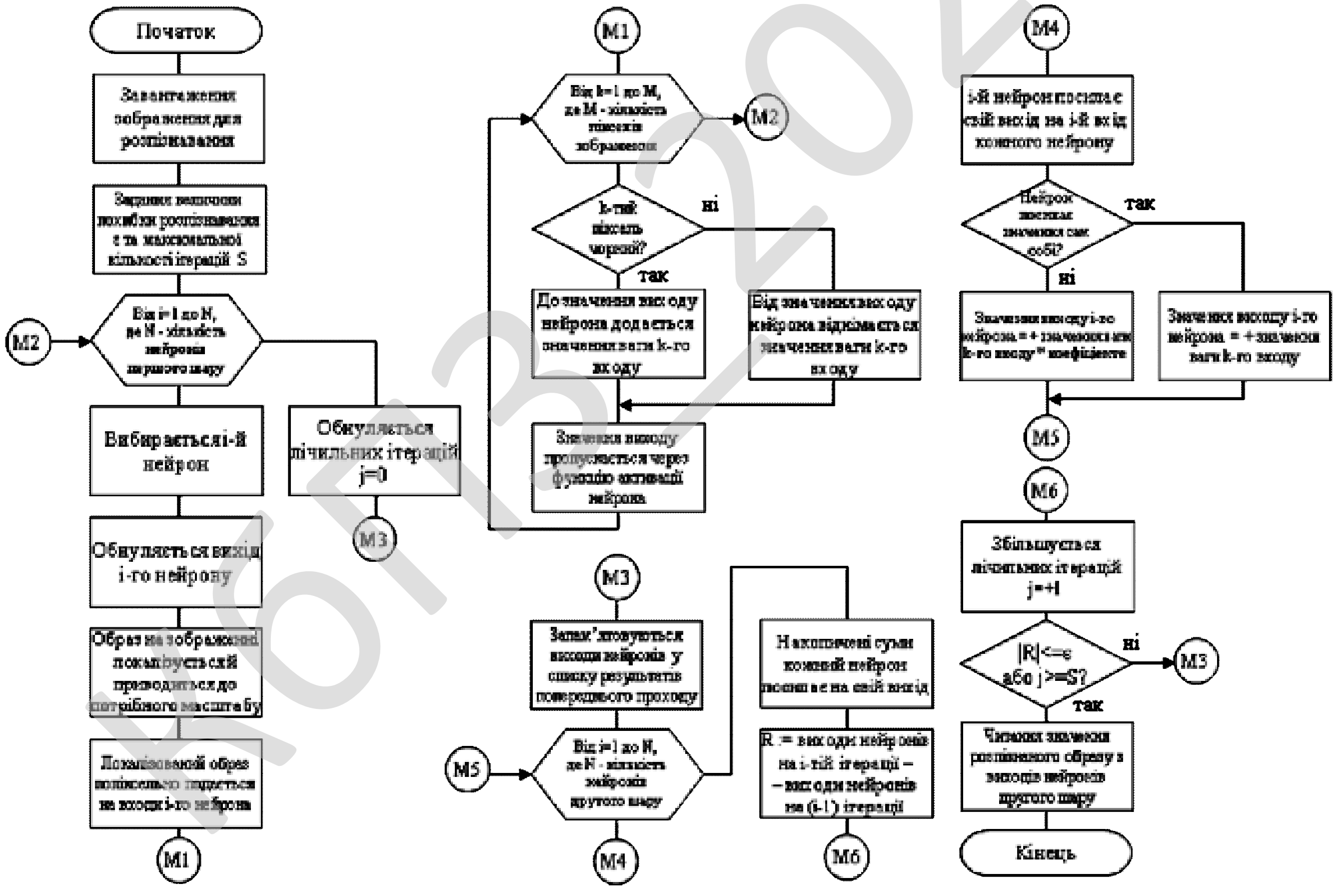


Рисунок 4.4 – Блок-схема підпрограми розпізнавання графічних образів нейронною мережею Хеммінга

Крок 22. Перевіряється абсолютне значення похибки розпізнвання з заданим значенням точності, якщо воно менше чи дорівнює заданому значенню (|R|<=ε), то здійснюється вихід з циклу розпізнвання у зв'язку зі знайденням точної відповіді.

Крок 23. Перевіряється значення лічильника ітерацій циклу розпізнавання, якщо воно більше максимально допустимого значення (j>=S), то здійснюється вихід з циклу розпізнвання у зв'язку з вичерпанням максимального часу роботи мережі, отримана відповідь буде містити деякий процент помилок.

Крок 24. Читання значення розпізнаного образу з виходів нейронів другого шару.

Крок 25. Передача результатів розпізнавання основній програмі.

Розглянемо підпрограму ініціалізації мережі:

```
// Ініціалізація мережі значеннями з таблиці
procedure TForm1.Init;
begin
  // Очистити мережу від зразків
  NeuralNetHem.ResetPatterns;
  // Додати зразки з таблиці до мережі
  Table.First;
  while not Table.Eof do
  begin
    AddPattern(TableLETTERS.AsString);
    Table.Next;
  end;
  Table.First;
  // Ініціалізувати ваги
  NeuralNetHem.InitWeights;
  // Мережа підготовлена до розпізнавання
end;
```

Процедура додавання нового образу до мережі має наступний вигляд:

```
// Додавання нового образу до мережі
procedure TForm1.AddPattern(Value: string);
var
  i: integer;
  xVector: TVectorInt;
begin
```

```
  SetLength(xVector, stgDatabase.RowCount * stgDatabase.ColCount);
  // Перетворення вимвольного рядка у вектор
  for i := 1 to stgDatabase.RowCount * stgDatabase.ColCount do
    try
      if TableLETTERS.AsString[i] = '1' then
        xVector[i - 1] := 1
      else
        xVector[i - 1] := -1;
    except
      xVector[i - 1] := -1;
    end;
  //
  NeuralNetHem.AddPattern(xVector);
end;
```

Процедура розпізнавання образів:

```
// розпізнавання символу
procedure TForm1.btnExecuteClick(Sender: TObject);
var
  i, j: integer;
  xString: string;
begin
  // Подаємо сигнали на вихід мережі
  for i := 0 to stgInput.ColCount - 1 do
    for j := 0 to stgInput.RowCount - 1 do
      if stgInput.Cells[i, j] = '1' then
        NeuralNetHem.Layers[1].Neurons[i * stgInput.RowCount + j].Output :=
1
      else
        NeuralNetHem.Layers[1].Neurons[i * stgInput.RowCount + j].Output :=
-1;
  // Запуск процесу розпізнавання
  NeuralNetHem.Calc;
  // Перетворення виходів мережі до рядка
  xString := '';
  for i := 1 to stgOutput.RowCount * stgOutput.ColCount do
    if NeuralNetHem.Layers[1].Neurons[i - 1].Output = 1 then
      xString := xString + '1'
    else
      xString := xString + ' ';
  // Відобразити результат
  ShowMatrix(stgOutput, xString);
```

```
end;
procedure TNeuralNetHem.Calc;
var
  i: integer;
  xCurrentIter: integer;
  xArray: TVectorFloat;
begin
  SetLength(xArray, InputNeuronCount);
 // Цикл працює поки не стабілізуються виходи
  xCurrentIter := 0;
  repeat
    for i := 0 to InputNeuronCount - 1 do
    begin

// Запам'ятовує попередній Крок ітерації, для
      //  цього використовується нульовий шар
      Layers[SensorLayer].Neurons[i].Output := Layers[1].Neurons[i].Output;
      xArray[i] := Layers[1].Neurons[i].Output;
    end;
    for i := 0 to InputNeuronCount - 1 do
      with Layers[1].Neurons[i] do

 // Розраховується новий стан нейронів і аксонів
        ComputeOut(xArray);
    Inc(xCurrentIter);
  until Stabled or (MaxIterCount = xCurrentIter);
  if Assigned(FOnAfterInit) then
    FOnAfterInit(Self);
  SetLength(xArray, 0);
  xArray := nil;
end;
```

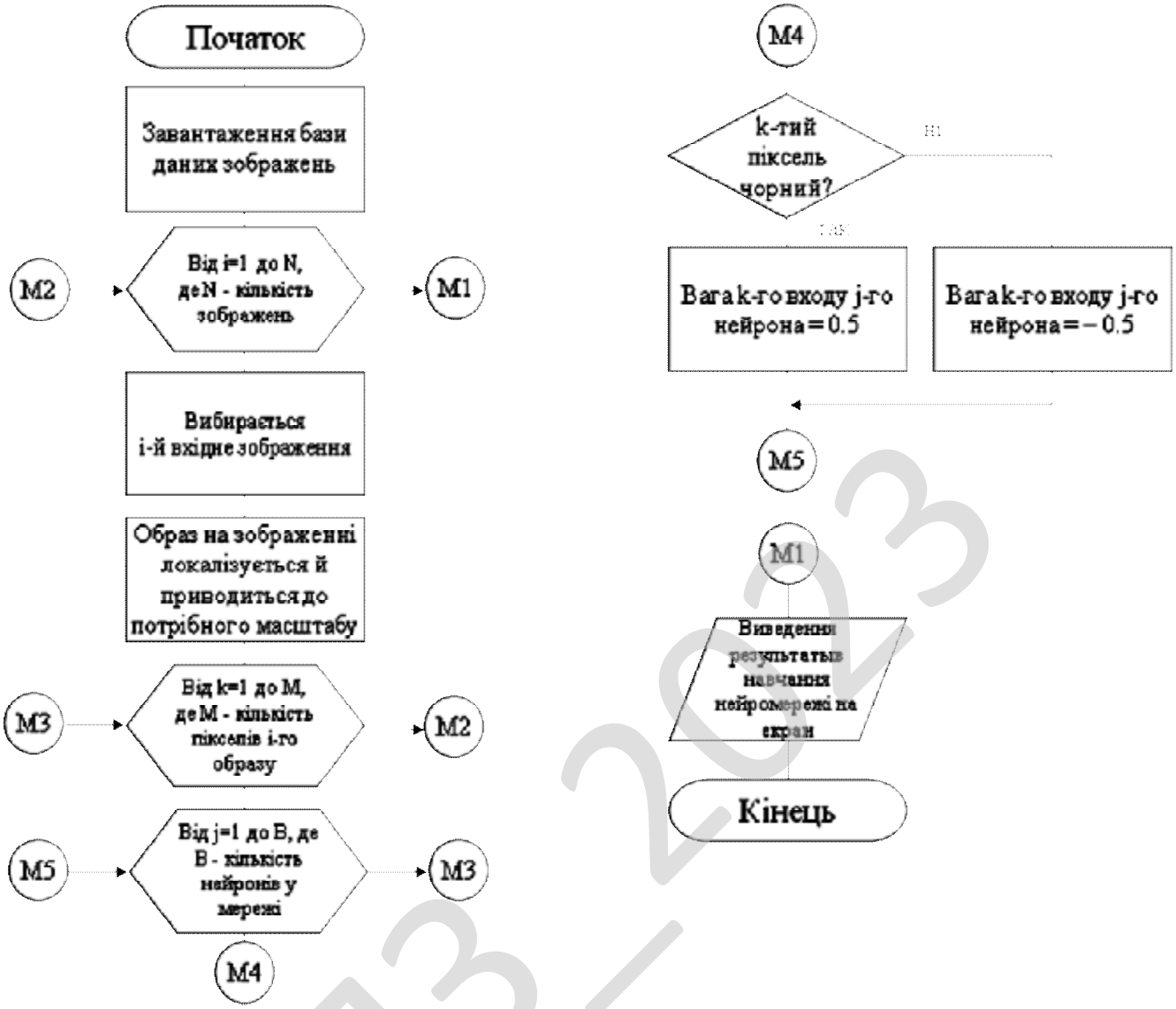


Рисунок 4.5 – Блок-схема підпрограми навчання нейронної мережі

На рисунку 4.5 показана блок-схема підпрограми навчання нейронної мережі. Її виконання складається з наступної послідовності кроків:

Крок 1. Завантаження бази даних зображень та відеофайлів. Вибір з них тих, на множині яких необхідно здійснити процес навчання нейронної мережі.

Крок 2. Для кожного зображення з вибірки повторюється послідовність кроків 3-9.

Крок 3. Вибирається і-й вхідне зображення.

Крок 4. Образ на зображенні локалізується й приводиться до потрібного масштабу.

Крок 5. Для кожного пікселя i-го образу виконується послідовність кроків 6-9.

Крок 6. Для кожного нейрону у нейромережі виконується послідовність кроків 7-9.

Крок 7. Перевіряється значення кольору k-того пікселя образу.

Крок 8. Якщо k-тий піксель чорний, то вага k-го входу j-го нейрона прирівнюється до 0.5.

Крок 9. Інакше – вага k-го входу j-го нейрона прирівнюється до – 0.5.

Крок 10. Виведення результатыв навчання нейромережі на екран.

Розглянемо підпрограму навчання нейронної мережі:

```
procedure TNeuralNetExtended.Train;
var
  i, j, k: integer;
begin
  if FUseForTeach = 100 then
  begin
    PatternCount := FFields[0].DataInCount;
    TestSetPatternCount := 0;
  end
  else
  begin
    PatternCount := Round((FFields[0].DataInCount - 1) * FUseForTeach /
100);
    TestSetPatternCount := FFields[0].DataInCount - PatternCount;
  end;
  if not TeachStopped then
    NormalizeData;
  // формування вхідних значень навчальної множини
  RealOutputIndexCount := OutputFieldCount;
  RealInputIndexCount := InputFieldCount;
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdInput then
    begin
      for j := 0 to PatternCount - 1 do
```

```
        FPatternsInput[j, k] := FFields[i].DataIn[j];
      // запам'ятовує індекс поля
      RealInputIndex[k] := i;
      Inc(k);
    end;
  // формування вихідних значень навчальної множини
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdOutput then
    begin
      for j := 0 to PatternCount - 1 do
        FPatternsOutput[j, k] := FFields[i].DataIn[j];
      // запам'ятовує індекс поля
      RealOutputIndex[k] := i;
      Inc(k);
   end;
// формування вхідних значень тестової множини
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdInput then
    begin
      for j := PatternCount to FFields[i].DataInCount - 1 do
        FTestSetPatterns[j - PatternCount, k] := FFields[i].DataIn[j];
      Inc(k);
    end;
// формування вихідних значень тестової множини
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdOutput then
    begin
      for j := PatternCount to FFields[i].DataInCount - 1 do
        FTestSetPatternsOut[j - PatternCount, k] := FFields[i].DataIn[j];
      Inc(k);
    end;
  // навчання або донавчання мережі
  TeachOffLine;
end;
procedure TNeuralNetBP.TeachOffLine;
var
  j: integer;
  xQuadError: double;
  xNewEpoch: boolean;
```

```
begin
  DoOnBeforeTeach;
  if not ContinueTeach then
  begin
  // ваги ініціалізуються, якщо мережа навчається з "нуля"
    InitWeights;
    FEpochCurrent := 1;
  end;
  Randomize;
  SetLength(FRandomOrder, FPatternCount);
  TeachStopped := False;
  while (FEpochCurrent <= EpochCount) do
  begin
    FTeachError := 0;
    FMaxTeachResidual := 0;
    FRecognizedTeachCount := 0;
    xNewEpoch := True;
    Shuffle;
    for j := 0 to PatternCount - 1 do
    begin
      LoadPatternsInput(FRandomOrder[j]);
      LoadPatternsOutput(FRandomOrder[j]);
      Propagate;
      xQuadError := QuadError;
// перевірка - чи розпізнана приклад з навчальної множині
      if xQuadError < IdentError then
        Inc(FRecognizedTeachCount);
      FTeachError := FTeachError + xQuadError;
  // максимальна помилка на навчальній множині
      if xNewEpoch then
      begin
        FMaxTeachResidual := xQuadError;
        xNewEpoch := False;
      end
      else
        if MaxTeachResidual < xQuadError then
          FMaxTeachResidual := xQuadError;
      CalcLocalError;
      AdjustWeights;
    end;
  // середня помилка на навчальній множині
    FMidTeachResidual := TeachError/PatternCount;
```

```
    // перевірка мережі на узагальнення
      if TestSetPatternCount > 0 then
        CheckTestSet;
      DoOnEpochPassed;
      if StopTeach then
      begin
        TeachStopped := True;
        Exit;
      end;
     Inc(FEpochCurrent);
    end;
    DoOnAfterTeach;
  end;
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою IDEA – симетричний блоковий алгоритм шифрування даних, запатентований швейцарською фірмою Ascom. Відомий тим, що застосовувався в пакеті програм шифрування PGP. У листопаді 2000 року IDEA був представлений як кандидат у проекті NESSIE в рамках програми Європейської комісії IST (англ. Information Societes Technology, інформаційні громадські технології).

Першу версію алгоритму розробили в 1990 році Лай Сюецзя (Xuejia Lai) і Джеймс Мессі (James Massey) зі Швейцарського інституту ETH Zürich (за контрактом з Hasler Foundation, яка пізніше влилася в Ascom-Tech AG) як заміна DES (англ. Data Encryption Standard, стандарт шифрування даних) і назвали її PES (англ. Proposed Encryption Standard, запропонований стандарт шифрування). Потім, після публікації робіт Біхамом і Шаміра по диференціальному криптоаналізу PES, алгоритм був поліпшений з метою посилення криптостійкості і названий IPES (англ. Improved Proposed Encryption Standard, покращений запропонований стандарт шифрування). Через рік його перейменували в IDEA (англ. International Data Encryption Algorythm).

Так як IDEA використовує 128-бітний ключ і 64-бітний розмір блоку, відкритий текст розбивається на блоки по 64 біт. Якщо таке розбиття неможливо, останній блок доповнюється різними способами певною послідовністю біт. Для уникнення витоку інформації про кожному окремому блоці використовуються різні режими шифрування. Кожен вихідний незашифрований 64 – біт ний блок ділиться на чотири підблока по 16 біт кожен, так як всі алгебраїчні операції, що використовуються в процесі шифрування, відбуваються над 16-бітними числами. Для шифрування і розшифрування IDEA використовує один і той же алгоритм.

Позначення операцій:

- $\boxplus$ Додавання за модулем 2^{16}.
- $\odot$ Множення за модулем $2^{16}+1$.
- $\oplus$ Побітова виключна диз'юнкція.

Фундаментальним нововведенням в алгоритмі є використання операцій з різних алгебраїчних груп, а саме:

Додавання за модулем 2^{16}.

Множення за модулем $2^{16}+1$.

Побітова виключна диз'юнкція (XOR).

Ці три операції несумісні в тому сенсі, що ніякі дві з них не задовольняють дистрибутивному закону, тобто:

$$a \odot (b \oplus c) \neq (a \odot b) \oplus (a \odot c).$$

Застосування цих трьох операцій ускладнює криптоаналіз IDEA в порівнянні з DES, який базується виключно на операції виключає АБО, а також дозволяє відмовитися від використання S-блоків і таблиць заміни. IDEA є модифікацією мережі Фейстеля.

Генерація ключів

З 128-бітного ключа для кожного з восьми раундів шифрування генерується по шість 16-бітних подключів, а для вихідного перетворення генерується чотири 16-бітних підключа. Всього буде потрібно 52 = 8 x 6 + 4

різних подключів по 16 біт кожен. Процес генерації п'ятдесяти двох 16-бітних ключів полягає в наступному:

Насамперед, 128-бітний ключ розбивається на вісім 16-бітних блоків. Це будуть перші вісім подключів по 16 біт кожен – $(K_1^{(1)}K_2^{(1)}K_3^{(1)}K_4^{(1)}K_5^{(1)}K_6^{(1)}K_1^{(2)}K_2^{(2)})$

Потім цей 128-бітний ключ циклічно зсувається вліво на 25 позицій, після чого новий 128-бітний блок знову розбивається на вісім 16-бітних блоків. Це вже наступні вісім подключів по 16 біт кожен – $(K_3^{(2)}K_4^{(2)}K_5^{(2)}K_6^{(2)}K_1^{(3)}K_2^{(3)}K_3^{(3)}K_4^{(3)})$

Процедура циклічного зсуву і розбивки на блоки триває до тих пір, поки не будуть згенеровані всі 52 16-бітних підключа.

Шифрування

Структура алгоритму IDEA показана на рисунку 4.6.

Процес шифрування складається з восьми однакових раундів шифрування і одного вихідного перетворення. Вихідний незашифрований текст ділиться на блоки по 64 біта. Кожен такий блок ділиться на чотири підблока по 16 біт кожен. На рисунку ці підблоки позначені D_1, D_2, D_3, D_4. У кожному раунді використовуються свої підключі згідно з таблицею підключів. Над 16-бітними підключами і підблока незашифрованого тексту проводяться наступні операції:

– Множення за модулем $2^{16}+1 = 65537$, причому замість нуля використовується 2^{16}.

– Додавання за модулем 2^{16}.

– Побітове виключне АБО.

В кінці кожного раунду шифрування є чотири 16-бітних підблоки, які потім використовуються як вхідні підблоки для наступного раунду шифрування. Вихідна перетворення являє собою скорочений раунд, а саме, чотири 16-бітних підблоки на виході восьмого раунду і чотири відповідних підключа піддаються операціям:

– Множення за модулем $2^{16}+1$.

– Додавання за модулем 2^{16}.

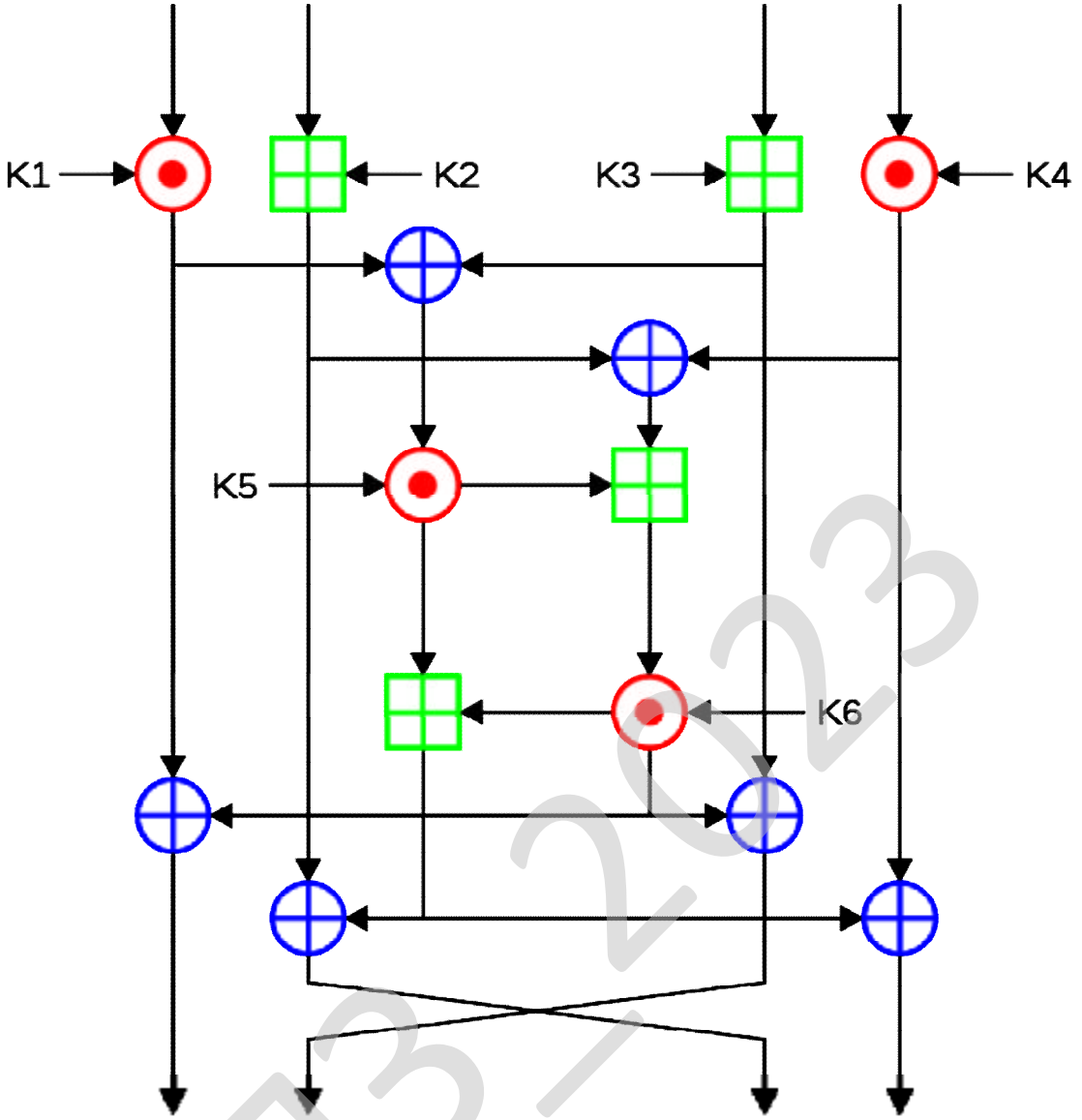


Рисунок 4.6 – Структура алгоритму IDEA

Після виконання вихідного перетворення конкатенація підблоків D_1', D_2', D_3' і D_4' являє собою зашифрований текст. Потім береться наступний 64-бітний блок незашифрованого тексту і алгоритм шифрування повторюється. Так продовжується до тих пір, поки не зашифрують всі 64-бітові блоки вихідного тексту.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення здійснює розпізнавання номерів автомобілів та облич людей за допомогою нейронної мережі Хеммінга. Розпізнавання образів здійснюється з графічних та відеофайлів.

На рисунку 5.1 зображене головне вікно програми.

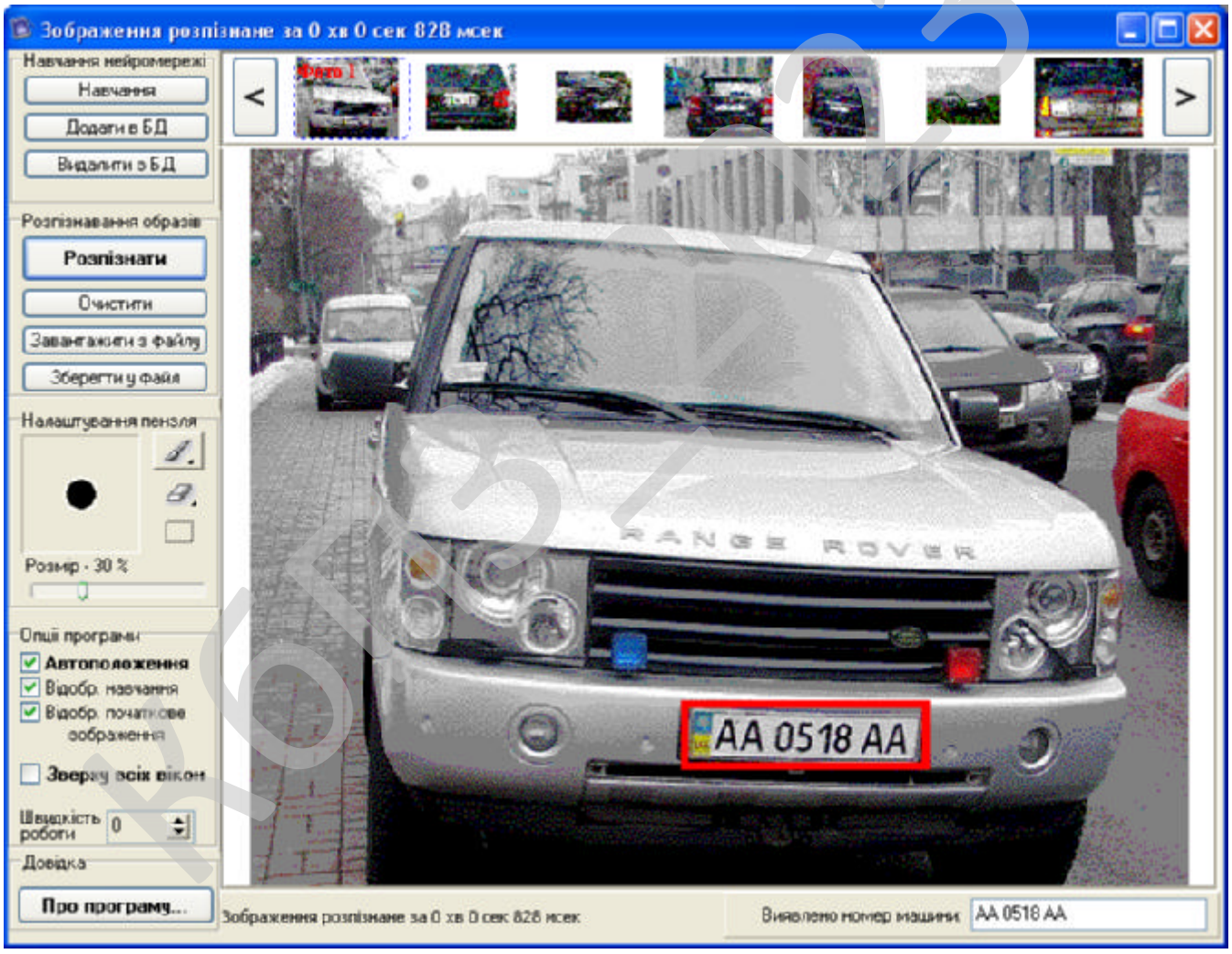


Рисунок 5.1 – Головне вікно програми

Програма володіє наступними функціональними можливостями:

1. Створення бази даних автомобілів.
2. Створення бази даних облич людей.
3. Навчання нейронної мережі на контрольній вибірці даних.
4. Розпізнавання номерів автомобілів.
5. Розпізнавання людей за обличчями.
6. Збереження результатів розпізнавання.
7. Перегляд та редагування бази даних.

На рисунку 5.2 зображене вікно довідки про програму, де можна переглянути короткі відомості про розроблене програмне забезпечення.

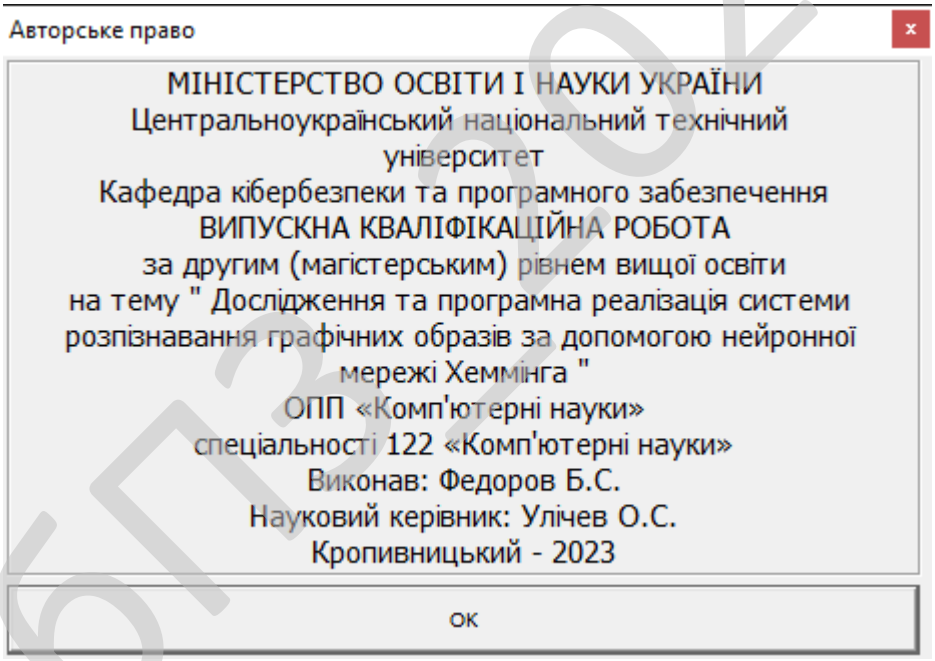


Рисунок 5.2 – Вікно довідки

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Метою розробки є дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Об'єктом дослідження є процес розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Предметом дослідження є методи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Методи дослідження базуються на методах розпізнавання графічних образів, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

– Розроблено вітчизняний продукт розпізнавання графічних образів за допомогою нейронної мережі Хеммінга, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі проведене дослідження та виконана програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	58
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження таблиці 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	60000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B \qquad (7.1)$$

де А – коефіцієнт Боема, А=2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \tag{7.2}$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПV_j, \tag{7.3}$$

де $ПV$j – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \tag{7.4}$$

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 55 = 93 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	93	Ф 7.1-7.4
Впровадження	13	Д13
Всього	134	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів,

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{134 \cdot 1}{48\text{-}5} = 3{,}1 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	8	720	12
Монітор	60	8	480	8
Клавіатура	30	8	240	4
Маніпулятор «мишка»	30	8	240	4
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	1	30	0,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	250	625	10,42
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	44,58

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф^{с}_{др} = \frac{З_ч \cdot n_{міс}}{1,2} \quad (7.6)$$

$$Ф^{с}_{др} = \frac{45 \cdot 2}{1,2} = 75 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=75/(48·8)=0,2 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів–електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп’ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,8	0,2
	Налаштування і конфігурування базової станції безпровідного зв’язку (CMTS)	0,2	
	Розробка та впровадження проектів з організації зв’язку між віддаленими об’єктами, ЛОМ	0,2	
	Забезпечення цілодобової роботи зв’язку клієнтів до мережі Інтернет	0,4	
Всього		1,6	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,2
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		1,6	

Складемо штатний розклад виконавців у таблицю 7.5.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (IT-менеджер)	1	23000	46000
Продакт-менеджер	0,5	16900	16900
Інженер-програміст	3,1	20500	127100
Інженер-електронщик	0,2	15000	6000
Інженер-системотехнік	0,2	17000	6800
Адміністратор мережі	0,2	18000	7200
Системний програміст	0,2	17000	6800
Дизайнер WEB	0,2	18000	7200
Інженер-верстальник	0,2	15000	6000
Бухгалтер-економіст	0,2	19000	7600
Всього за період розробки	$R_{сп}$=6	-	$Ф_{роб}$=237600

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{237600}{6 \cdot 48} = 825 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць.

$S_у$ – питома площа на одне робоче місце, $м^2$,

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$I_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за комерційною пропозицією Інтернет магазину Компбест за 14.10.23 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771
Системний блок		7771
Процесор	Intel Core i7-4790 (4 (8) ядер по 3.6 – 4.0 GHz), 8 MB Smart Cache	-
Системна плата	Intel Q87 Express Chipset, 6x USB 2.0, 4x USB 3.0, 2x PS/2, 1x DVI, 1x DisplayPort, 4x Audio, 1x LAN (RJ-45)	-
Жорсткий диск	256 SSD	-
Оперативна пам’ять	16 GB DDR3	-
Відеокарта	nVidia Quadro K420, 1 GB GDDR3, 128-bit	-
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	-
Корпус	HP EliteDesk 800 G1 Tower	-
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	-
інше	Клавіатура, мишка	-
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійного живлення	Powercom BNT-600AP USB	1400

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	8	11771	9416,8	103584,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	125216,3

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608

Продовження таблиці 7.8

1	2	3	4
Група 5,6			
4. Вимірювальні пристрої	5190	-	-
5. Транспортні засоби	143000	-	
6. Господарський інвентар	28000	-	-
Всього по групі	176190	20	35238
7. Нематеріальні активи	60000	10	6000
Разом	$К_р$= 1769406		$А_р$= 174246

Примітка: вартість автомобіля Ford Focus Ghia 2006 взята по даним з автосалону «Авто-РІА», джерело https://auto.ria.com/uk/auto_ford_focus_33565425.html, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_e}, \quad (7.11)$$

де N_e – Кількість екземплярів програм, шт.

$$З_о = 825 \cdot 134/58 = 1906 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних т суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 1906 \cdot 10 \cdot 0{,}01 = 191 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основно та додаткової зарплати

$$C_{оц} = 0{,}01 \cdot H_c (З_о + З_д), \quad (7.13)$$

де H_c – відрахування на соціальні потреби, %

$$C_{оц} = 0{,}01 \cdot 22(1906+191) = 461 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом *Нг*=15% від основної зарплати

$$Г_{осп} = З_о \cdot H_г \cdot 0{,}01, \quad (7.14)$$

де $H_г$ – загальногосподарські витрати, %

$$Г_{осп} = 1906 \cdot 15 \cdot 0{,}01 = 286 \text{ грн}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,4 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 200 \cdot 0{,}4 = 80 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 20):

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23,7 грн./шт., DVD-R box – 35 грн./шт.

$$З_{М2} = 20 \cdot 23{,}7 = 474 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають ус друкуючі пристрої і становить:

$$З_{МЗ} = \sum Ц_{з}., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{МЗ} = 574+558+570=1702 \text{ грн.}$$

$$З_{М} = (80 + 474 + 1702)/58=39 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційни систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де $Н_п$ – норматив витрат на освоєння нових мов програмування, %

$$О_п = 1906 \cdot 15 \cdot 0{,}01=286 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програ ($N_е$ = 58 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 174246 \cdot 2/(58 \cdot 12)=501 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтям калькуляції

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 1906+191+461+286+39+286+501=3670 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмно продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де P_c – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 50 \cdot 3670 = 1835 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна цін програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмног забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	1906
2. Додаткова зарплата виконавців	$З_д$	191
3. Відрахування на соціальні потреби	$С_{оц}$	461
4. Загальногосподарські витрати	$Г_{осп}$	286
5. Витрати на матеріали	$З_М$	39
6. Освоєння нових операційних систем, мов програмування	$О_п$	286
7. Амортизація основних фондів	$А_м$	501
8. Повна собівартість програмного забезпечення	$С_п$	3670
9. Плановий прибуток	$П_р$	1835
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	5505
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1101
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	6606

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	6606
Всього капітальних витрат	–	6006

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позна-чення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	40260	33550
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	3303
Всього витрат за рік	$І$	40260	36853

Витрати на технічне обслуговування:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \quad (7.23)$$

де $Т_р$ – кількість годин обслуговування системи за рік, *год.*,

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 250 годин на рік до 160 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\ баз} = 250 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 33550\ грн.$$

до

$$З_{р\ нов} = 160 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 21472\ грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$).

$$З_{ел}=П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Витрати на електроенергію при впровадженні нової системи не змінюються.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	6606	–	3303
Всього відрахувань	-	–	6606	–	3303

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (5505 - 3670) \cdot 58 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,2 \cdot 176190 + 0,1 \cdot 60000) \cdot 2/12 = 77389 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{361406}{(5505 - 3670) \cdot 58 \cdot 12 / 2} = 0,62 \text{ років}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (40260\text{-}36853)\text{-}0,5 \cdot 6606 = 104 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{6606}{40260 - 36853} = 1{,}9 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	58
2. Повна собівартість розробленої програми	Грн.	3670
3. Ціна розробленої програми	Грн.	5505
4. Плановий прибуток від реалізації розробленої програми	Грн.	1835
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1769406
7. Загальний прибуток від реалізації програмної продукції	Грн.	106430
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	77389
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,62
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	6606
11.Величина економічного ефекту у користувача програмної продукції	Грн.	104
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	1,9

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона праці – система збереження життя і здоров'я працівників у процесі трудової діяльності, що включає правові, соціально-економічні, організаційні, технічні, санітарно-гігієнічні, лікувально-профілактичні, реабілітаційні та інші заходи.

Згідно закону України “Про охорону праці” [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні;

– Положення про охорону праці;

– Накази з охорони праці;

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи

з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2].

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової діяльності. Їх праця стала більш інтенсивною, напруженою і вимагає значних витрат розумової, емоційної і фізичної енергії. Це призвело до необхідності у знаходженні комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку. Охорона здоров'я робітників, забезпечення безпеки умов праці, ліквідація та профілактика професійних захворювань і виробничого травматизму складає одну з головних турбот людського суспільства.

8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером

Електронно-обчислювальна машина (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно

створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м'язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1. У зазначеному приміщенні працюють 6 людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН

3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	6
Довжина	7
Висота	2,9

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	7
Об'єм, V	$м^3$	не менше 20.0	20,3

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному

приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-55	0,12
Тепла	23-25	50-70	0,1	24-25	50-65	0,1

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер Prinics PicKit M1 Smartphone Photo Printer White, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми україни "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значенн;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 6 м, довжина – 7 м, висота – 2,9 м.

У зазначеному приміщенні працює 4 людей.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою [1]:

$$F=ESKZ/n,$$

де: *F* – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; *E = 300 Лк;*

S – площа освітлюваного приміщення (у нашому випадку *S*=6 х 7 = 42 $м^2$);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5);*

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку *Z = 1,1*);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці [8]; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін (***ρ***$_{стін.}$) і стелі (***ρ***$_{стелі}$), значення коефіцієнтів дорівнюють ***ρ***$_{стін}$ = 50% і ***ρ***$_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де: *S* – площа приміщення, *S* = 42 $м^2$;

h – розрахункова висота підвісу, *h* = 2,9 *м (співпадає з в*исотою стелі, т.я. лампи освітлення закріплюються на стелі)*;*

A – ширина приміщення, *A = 6 м;*

B – довжина приміщення, *B =* 7 *м.*

Підставимо всі значення у формулу та визначемо індекса приміщення: i=1,4.

Знаючи індекс приміщення, за знаходимо *n = 0,29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам) [8].* Підставимо всі значення у формулу, визначемо світловий потік: *F=71689 Лм.*

Для розрахунку дудемо використовувати *світлодіодні стельові панелі Delux LED Panel 41 44Вт.*, світловий потік яких $F_л$ = *3600 Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де: F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення: $N=$

$$71689 / 3600=19,9 \text{ шт.}$$

Приймаємо необхідну кількість *світлодіодних світильників* 20 шт.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з умов поліпшення охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

– Досліджена система розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

– На основі отриманих результатів досліджень створена програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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

Програма реалізована на мові високого рівня Delphi 10.4 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані. 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 IDEA.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 104 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,9 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Федоров Б.С. Дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Henry Lloyd. Interactive Computer Graphics. States Academic Press. 2022. 247 p.

3. Ranjan Parekh. Fundamentals of Image, Audio, and Video Processing Using MATLAB®With Applications to Pattern Recognition. CRC Press. 2021. 406 p.

4. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

5. PeterShirley, SteveMarschner. Fundamentals of Computer Graphics. 2009

6. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

7. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

8. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

9. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. – Д.: НГУ, 2016. – 187 с.

10. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph.. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

11. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

12. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of

Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

13. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

14. 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.

15. 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

16. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

17. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) – 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

18. 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.

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

20. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

21. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

22. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

23. 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.

24. 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.

25. 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.

26. 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.

27. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

28. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

29. Smirnov O. Kuznetsov A., Zaicenko 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

36. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

37. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

38. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

39. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

40. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

41. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

42. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

43. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

44. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". – Випуск 2 (118). т.2. – Х.: ХУПС – 2014. – С. 64-67

45. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник тез VI міжнародної науково-практичної конференції "Проблеми та перспективи розвитку IT-індустрії". м. Харків. 17-18 квітня 2014р. – Харків: ХНЄУ. – 2014. – С. 240.

46. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. Коваленка О.В., Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки України від 26.02.2013 року № 1/11-4368. – Кіровоград: КНТУ 2013. – 257с.

47. Смірнов О.А., Дрєєв О.М., Доренський О.П. «Дослідження впливу ступеня стиснення зображень на оперативність їх доставки у телекомунікаційній системі. Збірник наукових праць "Системи обробки інформації". – Випуск 8(115). – Х.: ХУПС – 2013. – С. 234-239.

48. Смірнов О.А., Доренський О.П., Дрєєв О.М. Аналіз процесів стиснення та відновлення зображень на основі цифрових методів. Наука і техніка Повітряних сил Збройних Сил України. – Випуск 3(12). – Х.: ХУПС. – 2013. – С.122-127.

49. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 17.04.2012 року № 1/11-5249. – Кіровоград: КНТУ 2012. – 250 с.

50. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія» та 8.0925 «Автоматизація й комп'ютерно-інтегровані технології». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 1.12.2011 року № 1/11-11258. – Кіровоград: КНТУ 2012. – 454 с

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 32-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Delphi 10.4 Sydney.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 122 аркуші.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 11.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Улічев О.С.

Дослідження та програмна реалізація системи розпізнавання графічних образів за допомогою нейронної мережі Хеммінга

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 59

Літера: РП

Кропивницький – 2023 року

Файл Picture_processing_Hamming.pas – обробка розпізнаного образу

```
unit Picture_processing_Hamming;
// один з модулів розпізнавання образів, містить:
// -підготовку образа до розпізнання, з обчисленням інваріантів і поворотом
// -читання, запис та інші операції з Базою даних образів
// -навчання нейромережі
// -збереження й запис вагових коефіцієнтів нейромережі
// -автоматичний підбор коефіцієнтів нейромережі
// -захоплення й додавання нового образа в БД


// навчання нейромережі може проиходить досить довгий час
// змінювана цифра - це середньоквадратична помилка:
// на практиці,  якщо вона скакає кілька мінут в однієї й тої ж величини
// значить пійманий локальний мінімум

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls,Picture_Analyz_Hamming,UQPixels, Grids,
ValEdit,math,
  NeuralBaseComp, NeuralBaseTypes, Spin, ComCtrls;

type
  TAddSampleForm = class(TForm)
    SampleImage: TImage;
    SampleOpenDialog: TOpenDialog;
    SampleSaveDialog: TSaveDialog;
    GroupBox1: TGroupBox;
    Label2: TLabel;
    Label3: TLabel;
    matrix_size: TEdit;
    matrix_size: TEdit;
    NumIcons: TEdit;
    Button1: TButton;
    IconTypeSet: TComboBox;
    Label1: TLabel;
    AddSampleButton: TButton;
    BrowseButton: TButton;
    SaveSampleButton: TButton;
    ObjectImage: TImage;
    cellImage: TImage;
    Button2: TButton;
    Label4: TLabel;
    Label5: TLabel;
    shir: TLabel;
    hjkhjk: TLabel;
    Celx: TLabel;
    sdf: TLabel;
    vis: TLabel;
    Label6: TLabel;
    cely: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    ComboBox1: TComboBox;

    NeuralNet_Hamming_Hopf1    : TNeuralNet_Hamming_Hopf;
    NeuralNet_Hamming_BP1: TNeuralNet_Hamming_BP;
    prbEpoch: TProgressBar;
    speEpochCount: TSpinEdit;
    Label9: TLabel;
    sttError: TLabel;
    Button3: TButton;
    neroIMp: TEdit;
```

```
    neroAlfa: TEdit;
    neroRate: TEdit;
    Label10: TLabel;
    Button4: TButton;
    NeuralNet_Hamming_Extended1: TNeuralNet_Hamming_Extended;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    Button9: TButton;
    reader: TEdit;
    ComboBox2: TComboBox;
    recType: TComboBox;
    TrackBar1: TTrackBar;
    param: TLabel;
    Timer1: TTimer;
    lFPS: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    procedure BrowseButtonClick(Sender: TObject);
    procedure AddSampleButtonClick(Sender: TObject);
    procedure SaveSampleButtonClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button2Click(Sender: TObject);
    procedure NeuralNet_Hamming_BP1EpochPassed(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure NeuralNet_Hamming_Extended1EpochPassed(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
      procedure imFeatures(BitmapArr: TByteArr; pixeltype:integer);

var
  AddSampleForm: TAddSampleForm;
  sample_db,n_debug:textfile;
  sample_db_file_name:string ='data\config.db';
  neuro_debug_file_name:string ='data\debug.txt';
  neuro_teach_log_file_name:string ='data\log.txt';
  neuro_weight_file_name:string ='data\neuro.db';
  BitmapArr3:TByteArr;
  recognition_run:boolean=false;
  xVector: TVectorInt;
  xOutputVector: TVectorFloat;
  xInputVector: TVectorFloat;   // Вхідний вектор
  segment_X,segment_Y:integer;
  char_cnt,x_,y_:cardinal;
  debug:boolean =false;
  tm_tick:integer=0;

  implementation

uses upreview,Main_NeuralNet_Hamming_;
{$R *.dfm}
```

```
// щось для роботи з картинками без камери
procedure TAddSampleForm.BrowseButtonClick(Sender: TObject);
var
  Bitmap: Tbitmap;
begin
  if SampleOpenDialog.Execute then
  begin
    Bitmap := Tbitmap.Create;
    Bitmap.LoadFromFile(SampleOpenDialog.FileName);
    SampleImage.Width := Bitmap.Width;
    SampleImage.Height := Bitmap.Height;
  //  RadioTSample.Top := SampleImage.Height;
  //  RadioNotTSample.Top := SampleImage.Height;
    //AddSampleButton.Top := SampleImage.Height ;//+ RadioTSample.Height;
    BrowseButton.Top := SampleImage.Height;// + RadioTSample.Height;
    SampleImage.Picture.Assign(Bitmap);
    Bitmap.Destroy;
  end;
end;



// функція обробки бінарного масиву
// 1. обчислення центра мас об'єкта
// 2. обчислення орієнтації об'єкта
// 3. поворот об'єкта
// 4. вихоплювання об'єкта
// 5. вжимання об'єкта до іконки
// багато частин можна прискорити.
procedure imFeatures(BitmapArr: TByteArr;pixeltype:integer);
var x,y,i,j,xc,yc:integer;
    N,sumx,sumy:cardinal;
    buff:integer;
    max_x,max_y:integer;
    canvas: tcanvas;
    tmp,O,SumUx,SumUy,SumUxy,Ux,Uy,Uxy,C:real;
    r:single;
    s,ss:extended;
    BitmapArr4,BitmapArr2: TByteArr;
    al:real;
    x_new,y_new:integer;
    actual_min_x,actual_min_y,actual_max_x,actual_max_y:integer;
    Bporog,BPixelsInCell,ix,iy,new_size_x,new_size_y,by,bx:integer;
    dx,dy,celx,cely:real;
    line:string;
    longest:integer;
    offset:cardinal;

const MinBPixelsPercent =10;

begin
  max_x:=Length(BitmapArr[0])-1;max_y:=Length(BitmapArr)-1;
  sumx:=0;sumy:=0;N:=0;



      // малюємо зелененьку рамочку
    if debug then
    begin
      if AddSampleForm.visible then
        begin
        Canvas:=AddSampleForm.SampleImage.Canvas;
        Canvas.Pen.Color := clGreen;
        Canvas.Pen.Width := 1;

        Canvas.MoveTo(Round(left),Round(top));
        Canvas.LineTo(Round(right),Round(top));
        Canvas.LineTo(Round(right),Round(buttom));
        Canvas.LineTo(Round(left),Round(buttom));
```

```
        Canvas.LineTo(Round(left),Round(top));
//        Canvas.MoveTo(Round(x - 10),Round(y - 10));

        end;
    end;


  // Алгоритм обчислення центра мас образа
   for y:=top to buttom do
    for x:=left to right do
      begin
      //BitmapArr2[y,x]:=0;
      buff:=BitmapArr[y,x];
      if buff=pixeltype then buff:=1 else buff:=0;
      sumx:=sumx+( x-left)*buff;
      sumy:=sumy+( y-top)*buff;
      N:=N+buff;
      end;


  // одержали координати центра мас образа
  xc:=left+round(sumx/N);
  yc:=top+round(sumy/N);

  // для сегментації
  segment_X:=xc;
  segment_Y:=yc;

    // малюємо хрестик там, де визначився центр мас
    if debug then
    begin
      if AddSampleForm.visible then
        begin
        Canvas:=AddSampleForm.SampleImage.Canvas;
        x:=xc;
        y:=yc;
        Canvas.Pen.Color := clRed;
        Canvas.Pen.Width := 2;

        Canvas.MoveTo(Round(x - 10),Round(y + 10));
        Canvas.LineTo(Round(x + 10),Round(y - 10));
        Canvas.MoveTo(Round(x - 10),Round(y - 10));
        Canvas.LineTo(Round(x + 10), Round(y + 10));
        end;
    end;


    // обчислення декількох допоміжних величин
    SumUx:=0;SumUy:=0;
    for y:=top to buttom do
      for x:=left to right do
        begin
        buff:=BitmapArr[y,x];
        if buff<>pixeltype then buff:=0 else buff:=1;
        SumUx:=SumUx+buff*sqr( x-xc);
        SumUy:=SumUy+buff*sqr( y-yc);
        SumUxy:=SumUxy+buff*( y-yc)*( x-xc);
      end;
    Ux:=1/12+SumUx*1/N; Uy:=1/12+SumUy*1/N; Uxy:=1/12+SumUxy*1/N; C:=sqrt(sqr(
Ux-Uy)+4*sqr(Uxy));

    //Обчислюємо орієнтацію об'єкта
    if Uy>Ux then
      begin
      O:=180/pi*ArcTan(( Uy-Ux+C)/(2*Uxy));
      //tmp:=( Uy-Ux+C)/(2*Ux*Uy);
      end
      else
      begin
```

```
      O:=180/pi*ArcTan((2*Uxy)/( Ux-Uy+C));
      //tmp:=(2*Uxy)/( Ux-Uy+C);
      end;



      // малюємо напрямок орієнтації об'єкта
  if debug then
    begin
        Canvas.Pen.Color := clRed;
        Canvas.Pen.Width := 5;

        Canvas.MoveTo(xc,yc);
        x:= trunc(( right-xc) * cos(O/180*pi)+xc);
        y:= trunc(( buttom-yc) * sin(O/180*pi)+yc);
        Canvas.LineTo(x,y);


    end;

   // Повертаємо об'єкт щодо точки центра мас, на кут орієнтації
   // попутно визначаємо точне розташування повернутого образа

   actual_min_x:=max_x;
   actual_min_y:=max_y;
   actual_max_x:=0;
   actual_max_y:=0;


    // оскільки невідомо - де в об'єкта що стирчить - споконвічно робимо
    // квадратну матрицю - довгої з діагональ первісної
    if ( right-left)>( buttom-top) then
      begin
      longest:=round(1.5*( right-left));

      end else
      begin
      longest:=round(1.5*( buttom-top));
      end;

      SetLength(BitmapArr2,round(longest)+1);
      for y := 0 to High(BitmapArr2) do
        SetLength(BitmapArr2[y],round(longest)+1);


    O:=-O*pi/180;
    for y := top to buttom do
    begin
      for x := left to right do
      begin
        if BitmapArr[y,x]=pixelType then
          begin
          al:=arctan2((y - yc), (x - xc));
          r := sqrt(sqr(x - xc) + sqr(y - yc));
          //x_new:= trunc(xc + r * cos(al + O));
          //y_new:= trunc(yc + r * sin(al + O));
          x_new:= trunc(r * cos(al + O)+longest/2);
          y_new:= trunc(r * sin(al + O)+longest/2);

          if x_new<actual_min_x then actual_min_x:=x_new;
          if y_new<actual_min_y then actual_min_y:=y_new;
          if x_new>actual_max_x then actual_max_x:=x_new;
          if y_new>actual_max_y then actual_max_y:=y_new;
          if (y_new<longest)and(x_new<longest) and(y_new>0) and (x_new>0) then
BitmapArr2[y_new,x_new]:=1;
          //Canvas.Pixels[x_new,y_new]:= clGreen;
          end;
      end;
    end;
```

```
    // прощай квадратна матриця, довгої з діагональ початкової- вихоплюємо
    // заново повернений образ
    new_size_x:=actual_max_ x-actual_min_x;
    new_size_y:=actual_max_ y-actual_min_y;



    SetLength(BitmapArr4,new_size_y+1);
    for y := 0 to High(BitmapArr4) do
      SetLength(BitmapArr4[y],new_size_x+1);

    if Debug then
    begin
      Canvas:=AddSampleForm.ObjectImage.Canvas;
      AddSampleForm.ObjectImage.Height :=new_size_y;
      AddSampleForm.ObjectImage.Width :=new_size_x;

    end;

   // перетворюємо уже повернений масив із квадратного в прямокутний по границі
   // потім це можно буде зробити в наступному кроці, поки однаково він
   // щось малює - те можна й залишити тут

    for y := 0 to new_size_y do
      begin
      for x := 0 to new_size_x do
        begin

          ///// !!!!!!!!!!!!!!!!!!!!!!!!

          if ((y+actual_min_y)<(longest)) and ((x+actual_min_x)<(longest)) and
((y+actual_min_y)>0) and ((x+actual_min_x)>0)  then
          BitmapArr4[y,x]:=BitmapArr2[y+actual_min_y,x+actual_min_x];
          if debug then if BitmapArr4[y,x]>0 then Canvas.Pixels[x,y]:=clRed else
Canvas.Pixels[x,y]:= clWhite;
        end;
      end;

    if debug then
    begin
     //AddSampleForm.ObjectImage.Width :=new_size_x;
     //AddSampleForm.ClientWidth := 640;
    end;


    SetLength(BitmapArr3,IConSize+1);
    for y := 0 to High(BitmapArr3) do
      SetLength(BitmapArr3[y],IconSize+1);


    // знаходимо відповідність між осередком іконки масивом
    if new_size_x> IconSize then
      celx:=new_size_x/IconSize else celx:=1;

    if new_size_y> IconSize then
      cely:=new_size_y/IconSize else cely:=1;


      // якась інформація про образ
    if debug then
      begin
       AddSampleForm.shir.Caption:=inttostr(new_size_x);
       AddSampleForm.celx.Caption:=inttostr(round(celx));
       AddSampleForm.vis.Caption:=inttostr(new_size_y);
       AddSampleForm.cely.Caption:=inttostr(round(cely));
      end;
```

```
    if debug then
      begin
      offset:=round((6*IconSize+1)*(char_ cnt-1)); //
      AddSampleForm.cellImage.width:=(6*IconSize+1)*5;//(6*IconSize+1) +offset;
      AddSampleForm.cellImage.height:=(6*IconSize+1);

      end;


    if debug then
      begin
      canvas:=AddSampleForm.cellImage.canvas;
      Canvas.Brush.Color := ClWhite;//Canvas.FillRect(Canvas.ClipRect);
      for x_:=0 to IconSize+offset do

       begin
       Canvas.MoveTo(Round(6*x_),Round(0));
       Canvas.LineTo(Round(6*x_),Round(6*IconSize));
      end;
      for y_:=0 to IconSize do
       begin
       Canvas.MoveTo(round(0),Round(6*y_));
       Canvas.LineTo(Round(6*IconSize+offset),Round(6*y_));
      end;
      end;

      // Алгоритм зменшення зображення  - всі параметри плаваючі

          if debug then    Canvas.Brush.Color := clBlack;

    Bporog := Round(celx * cely / 100 * MinBPixelsPercent);


    dy:=0;
    dx:=0;
    BPixelsInCell := 0;


    for iy := 0 to IconSize - 1 do
      begin

      for ix := 0 to IconSize - 1 do
        begin
         dy:=0;dx:=0;
         while (dy+iy*cely)<((iy+1)*cely) do
          begin
          while (dx+ix*celx)<((ix+1)*celx) do
            begin
                  if (round(dy+iy*cely)<new_size_y) and
(round(dx+ix*celx)<new_size_x) then
                  if BitmapArr4[round(dy+iy*cely),round(dx+ix*celx)] = 1 then
BPixelsInCell := BPixelsInCell + 1;
                  dx:=dx+1;
            end;
            dy:=dy+1;dx:=0;;
          end;

        if BPixelsInCell > BPorog then
          begin  BitmapArr3[iy,ix] := 0; end
          else
          begin  BitmapArr3[iy,ix] := 1; if debug then
canvas.FloodFill(ix*6+3+offset,iy*6+3,canvas.pixels[ix*6+3+offset,iy*6+3],fsSurf
ace);  end;
         BPixelsInCell:=0;


        end;
      end;
```

```
    // ну от і все - іконка готова - тепер або розпізнаємо її або
    // додаємо в БД.
    if debug then
    begin
    for y := 0 to IconSize - 1 do
    begin
      for x := 0 to IconSize - 1 do
         write(F,BitmapArr3[y,x]);
      writeln(F,' ');
    end;
        writeln(F,'--------------');
    end;

end;

// додавання нового семпла - перетворення його в іконку, додавання в масив
procedure TAddSampleForm.AddSampleButtonClick(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  Icon: TByteArr;
  y: Cardinal;
  IconType: Cardinal;

begin
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);

  SetLength(BitmapArr,QP.Height);
  for y := 0 to High(BitmapArr) do
    SetLength(BitmapArr[y],QP.Width);

    IconType:=IconTypeSet.ItemIndex;
{  if RadioTSample.Checked then
    IconType := Icon
  else
    IconType := IconNot;
 }
  //IconTypes:=IconTypes+1;
  ConvertBitmapToMonoChrome(QP,BitmapArr);
  imFeatures(BitmapArr,1);
  AddSample(BitmapArr,IconType); // реально використовується BitmapArr3

  //Close;
end;

//подія - натискання "Зберегти"
procedure TAddSampleForm.SaveSampleButtonClick(Sender: TObject);
begin
  if SampleSaveDialog.Execute then
  begin
    SampleImage.Picture.Bitmap.SaveToFile(SampleSaveDialog.FileName);
  end;
end;
// читання іконки з файлу
function ReadIcon:TByteArr;
var
  x,y: Integer;
  Arr1:TByteArr;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  buff:string;
  buff2:char;
  cnt:integer;

begin
  SamePixels := 0;
  Width := IconSize;
  Height := IconSize;
```

```
  SetLength(Arr1,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(Arr1[y],IconSize);

  SetLength(IconArr[High(IconArr)].Icon,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(IconArr[High(IconArr)].Icon[y],IconSize);

  cnt:=0;
  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        read(sample_db,buff2);
        Arr1[x,y]:=strtoint(buff2);
            if Arr1[x,y] = 1 then
            begin

            xVector[cnt] := 2;
            xInputVector[cnt] := 0;
            end
            else
            begin
            xVector[cnt] := -1;
            xInputVector[cnt] := 1;
            end;
          cnt:=cnt+1;
        //write(F,Arr1[x,y]);
      end;
      //WRITELN(f);
      readln(sample_db,buff);
    end;

  //NeuralNet_Hamming_Hopf1.AddPattern(xVector);
  result:=Arr1;

end;


// подія відкриття форми, завантаження семплів з файла, ініціалізація й т.п.
procedure TAddSampleForm.FormShow(Sender: TObject);

  var i1,i,j,jj:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff:string;
    buff_file:textfile;
    max_icon_type:integer;

begin
  AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);


  matrix_size.text:=inttostr(IconSize);
  matrix_size.text:=inttostr(IconSize);



  if FileExists(sample_db_file_name) then
    begin
      assignfile(sample_db,sample_db_file_name);
      assignfile(n_debug,neuro_debug_file_name);
      rewrite(n_debug);
      reset(sample_db);
```

```
      //IconTypes:=2;
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
      readln(sample_db,buff); IconTypes:=strtoint(buff);//eIconTypes.Text:=buff;
      readln(sample_db,buff);
SetLength(IconArr,strtoint(buff)+1);NumIcons.Text:=buff;
      // вхідний вектор Нейрона мережа Хопфілда - нейронів стільки ж , скільки й
точок в іконці
      SetLength(xVector, IconSize * IconSize);
      // вихідний вектор багатошарової - дорівнює числу образів
      //IconTypes:=3;
      SetLength(xOutputVector, IconTypes);

      // вхідний вектор багатошарової - дорівнює числу нейронів
      SetLength(xInputVector, IconSize * IconSize);


      //  SetLength(xInputVector, 5);
      //  SetLength(xOutputVector, 1);

      // настроювання нейроної мережі Хопфілда

      AddSampleForm.NeuralNet_Hamming_Hopf1.LayerCount:=1;
      AddSampleForm.NeuralNet_Hamming_Hopf1.InputNeuronCount:=IconSize *
IconSize;

      //NeuralNet_Hamming_Extended1.InputFieldCount:=IconSize * IconSize;
      {

NeuralNet_Hamming_Extended1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);

NeuralNet_Hamming_Extended1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);

NeuralNet_Hamming_Extended1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      }



      // настроювання багатошарової
      //NeuralNet_Hamming_BP1.LayerCount:=2;
      //NeuralNet_Hamming_BP1.LayersBP[0].NeuronCount:=IconSize * IconSize;
      //NeuralNet_Hamming_BP1.LayersBP[1].NeuronCount:=IconSize * IconSize;
      //NeuralNet_Hamming_BP1.LayersBP[1].NeuronCount:=IconTypes;
      //NeuralNet_Hamming_BP1.LayersBP[2].NeuronCount:=IconTypes;

      NeuralNet_Hamming_BP1.ResetPatterns;




      for i:=0 to High(IconArr) do
        begin
        readln(sample_db,buff); IconArr[i].IconType:=strtoint(buff)+1;


        for j:=1 to IconTypes do
          begin
          if j<>IconArr[i].IconType then xOutputVector[ j-1]:=0 else
xOutputVector[ j-1]:=1;
          end;



        IconArr[i].Icon:=ReadIcon;
        // додаємо вектор навчання нейроної мережі Хопфілда
        AddSampleForm.NeuralNet_Hamming_Hopf1.AddPattern(xVector);

        // Додаємо вектор багатошарової мережі
```

```
        //SetLength(xInputVector, 100);
        //NeuralNet_Hamming_Extended1.AddPattern(xInputVector, xOutputVector);
        for i1:=0 to length(xInputVector) do
        write(n_debug,inttostr(round(xInputVector[i1])));
        writeln(n_debug,'');
        // NeuralNet_Hamming_BP1.AddPattern(xInputVector, xOutputVector);
        //SetLength(xInputVector, 256);
        readln(sample_db,buff);
        end;
        writeln(n_debug,'-----');

        {
        readln(sample_db,buff); IconArr[0].IconType:=strtoint(buff);
        xOutputVector[0]:=0;  xOutputVector[1]:=1;
        IconArr[0].Icon:=ReadIcon; // SetLength(xInputVector, 256);
        //NeuralNet_Hamming_Extended1.AddPattern(xInputVector, xOutputVector);
        NeuralNet_Hamming_BP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);

        //SetLength(xInputVector, 256);
        readln(sample_db,buff); IconArr[1].IconType:=strtoint(buff);
        xOutputVector[0]:=1;  xOutputVector[1]:=0;
        IconArr[1].Icon:=ReadIcon; //SetLength(xInputVector, 256);
        //NeuralNet_Hamming_Extended1.AddPattern(xInputVector, xOutputVector);
        NeuralNet_Hamming_BP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);
        }
        // Ініціалізувати ваги Нейроної мережі Хопфілда
        //NeuralNet_Hamming_Hopf1.InitWeights;



      closefile(n_debug);
      closefile(sample_db);
    end;
  //NeuralNet_Hamming_BP1.ResetPatterns;

end;


// запис іконки у файл
procedure WriteIcon(Arr1: TByteArr);
var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
begin
  SamePixels := 0;
  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);

  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        write(sample_db,Arr1[x,y]);
      end;
    writeln(sample_db,'');
    end;
end;

// закриття форми
procedure TAddSampleForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
```

```
    max_icon_type:integer;
begin
//  recognition_run:=not(recognition_run);
  if recognition_run then
    begin
    RobotRecognition.Terminate;
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    recognition_run:=not(recognition_run);
    end;

  //IconTypes:=strtoint(eIconTypes.text);

  {
  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;

  }
end;

// відновлення картинки для збереження семпла
procedure TAddSampleForm.Button2Click(Sender: TObject);
var canvas:tcanvas;
begin
if debug then
begin
      canvas:=AddSampleForm.cellImage.canvas;
      Canvas.Brush.Color := ClWhite;Canvas.FillRect(Canvas.ClipRect);
      Canvas:=AddSampleForm.ObjectImage.Canvas;
      AddSampleForm.ObjectImage.Height :=0;
      AddSampleForm.ObjectImage.Width :=0;

end;
upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
end;

procedure TAddSampleForm.NeuralNet_Hamming_BP1EpochPassed(Sender: TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(NeuralNet_Hamming_BP1.TeachError);
  Application.ProcessMessages;

end;


// навчання нейромережі із зазначеними параметрами
procedure TAddSampleForm.Button3Click(Sender: TObject);
begin


      NeuralNet_Hamming_BP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
      NeuralNet_Hamming_BP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
      NeuralNet_Hamming_BP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      NeuralNet_Hamming_BP1.Init;
 // Учимо багатошарову
 NeuralNet_Hamming_BP1.EpochCount := speEpochCount.Value;
```

```
 prbEpoch.Max := NeuralNet_Hamming_BP1.EpochCount;
 prbEpoch.Position := 0;

 // Запуск процесу навчання (offline)
 //NeuralNet_Hamming_Extended1.TeachOffLine;
 NeuralNet_Hamming_BP1.TeachOffLine;
end;


// перебір параметрів навчання, навчання, записати помилки в лог
// для дослідження  - яка нейромережа краще, і як не потрапити в локальний
// мінімум при навчанні.
procedure TAddSampleForm.Button4Click(Sender: TObject);
  var log:textfile;
  var i,j:integer;
  var cnt:integer;
begin


  NeuralNet_Hamming_BP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
  NeuralNet_Hamming_BP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
  NeuralNet_Hamming_BP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
  NeuralNet_Hamming_BP1.EpochCount := 5000;


  prbEpoch.Max := NeuralNet_Hamming_BP1.EpochCount;
  prbEpoch.Position := 0;
      assignfile(log,neuro_teach_log_file_name);
      append(log);
      writeln(log,'--------------------------');
      closefile(log);
  cnt:=400;
  for i:=1 to 20 do
    begin
    NeuralNet_Hamming_BP1.Alpha:=NeuralNet_Hamming_BP1.Alpha-0.01;
    for j:=1 to 20 do
      begin
      prbEpoch.Position := 0;
      NeuralNet_Hamming_BP1.TeachRate:=NeuralNet_Hamming_BP1.TeachRate-0.005;
      NeuralNet_Hamming_BP1.TeachOffLine;
      assignfile(log,neuro_teach_log_file_name);
      append(log);

      writeln(log,'помилка='+floattostr(NeuralNet_Hamming_BP1.TeachError)+'
альфа='+floattostr(NeuralNet_Hamming_BP1.Alpha)+'
шаг навчання =',floattostr(NeuralNet_Hamming_BP1.TeachRate));
      closefile(log);
      cnt:= cnt-1;
      sttError.Caption := inttostr(cnt);
      //prbEpoch.Position := prbEpoch.Position + 1;
      Application.ProcessMessages;
      end;
    end;

end;

// подія кінця епохи навчання , зрушення процес бара, відображення помилки
procedure TAddSampleForm.NeuralNet_Hamming_Extended1EpochPassed(Sender:
TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(NeuralNet_Hamming_Extended1.TeachError);
  Application.ProcessMessages;

end;


// дозвіл розпізнавання
procedure TAddSampleForm.Button5Click(Sender: TObject);
```

```
begin
  debug:=false;
  recognition_run:=not(recognition_run);


  if recognition_run then
    begin
    RobotRecognition := TRobotRecognition.Create(false);
    RobotRecognition.RecType:=AddSampleForm.recType.ItemIndex;
    AddSampleForm.Button5.Caption:='СТОП'
    end
    else
    begin
    //RobotRecognition.
    RobotRecognition.Terminate;
    //RobotRecognition.Destroy;
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    end;

end;



// збереження образів у файл
procedure TAddSampleForm.Button6Click(Sender: TObject);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
    max_icon_type:integer;
begin
  //IconTypes:=strtoint(eIconTypes.text);


  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;


end;


// збереження ваг нейромережі у файл
procedure TAddSampleForm.Button7Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
begin
  AssignFile(neuro,neuro_weight_file_name);
  rewrite(neuro);
  writeln(neuro,floattostr(NeuralNet_Hamming_BP1.TeachRate));
  writeln(neuro,floattostr(NeuralNet_Hamming_BP1.Momentum));
  writeln(neuro,floattostr(NeuralNet_Hamming_BP1.Alpha));

  for i:=0 to NeuralNet_Hamming_BP1.LayerCount-1 do
    begin
```

```
    for j:=0 to NeuralNet_Hamming_BP1.Layers[i].NeuronCount-1 do
      begin
        for w:=0 to
length(NeuralNet_Hamming_BP1.Layers[i].Neurons[j].FWeights)-1 do
          begin

writeln(neuro,floattostr(NeuralNet_Hamming_BP1.Layers[i].Neurons[j].Weights[w]))
;
          end;
      end;
    end;
  closefile(neuro);
end;



// завантаження ваг нейромережі з файла
procedure TAddSampleForm.Button8Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
  buff:string;
begin
  AssignFile(neuro,neuro_weight_file_name);
  reset(neuro);


  readln(neuro,buff);NeuralNet_Hamming_BP1.TeachRate:=StrToFloat(buff);
  readln(neuro,buff);NeuralNet_Hamming_BP1.Momentum:=StrToFloat(buff);
  readln(neuro,buff);NeuralNet_Hamming_BP1.Alpha:=StrToFloat(buff);
  NeuralNet_Hamming_BP1.Init;
  for i:=0 to NeuralNet_Hamming_BP1.LayerCount-1 do
    begin
    for j:=0 to NeuralNet_Hamming_BP1.Layers[i].NeuronCount-1 do
      begin
        for w:=0 to
length(NeuralNet_Hamming_BP1.Layers[i].Neurons[j].FWeights)-1 do
          begin
            readln(neuro,buff);

NeuralNet_Hamming_BP1.Layers[i].Neurons[j].Weights[w]:=strtofloat(buff);
          end;
      end;
    end;
  closefile(neuro);

end;

procedure TAddSampleForm.Button9Click(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  y: Cardinal;

begin
  debug:=true;
 // upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);


  SetLength(BitmapArr,QP.Height);
  for y := 0 to High(BitmapArr) do
    SetLength(BitmapArr[y],QP.Width);

  ConvertBitmapToMonoChrome(QP,BitmapArr);
```

```
  //ConvertBitmapToMonoChrome(QP2,BitmapArr);
  if length(BitmapArr)>0 then
  segment(BitmapArr);
  debug:=false;
end;

procedure TAddSampleForm.TrackBar1Change(Sender: TObject);
begin
AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);
end;

procedure TAddSampleForm.Timer1Timer(Sender: TObject);
begin
AddSampleForm.lFPS.Caption:=floattostr(round(tm_tick/5*10)/10);
tm_tick:=0;
end;

end.
```

Файл Picture_Analyz_Hamming.pas - модуль розпізнавання графічних образів за допомогою нейронної мережі Хеммінга

```
unit Picture_Analyz_Hamming;
// один з модулів розпізнавання образів, містить:
// -повну функція сегментації й допоміжні процедури
// -підготовку й запуск розпізнавання нейромережею
// -процентне розпізнавання й супутні функції


interface

uses Graphics,SysUtils,Math,UTypeConst,Windows,UQPixels,NeuralBaseComp,Classes,
NeuralBaseTypes;

const
  Icon = 0;
  IconNot = 1;
  min_size=10;

type

  TByteArr = array of array of integer;

  TRobotRecognition = class(TThread) //Потік розпізнання
    private


    public
      //BitmapArr: TByteArr;
      BitmapReady:boolean;
      RecType:integer; // вибір параметрів розпізнання
      procedure Execute(); override; //запуск потоку
  end;


  TBitmap = Graphics.TBitmap;


  //масив байт Nx
  //TByteArr = array of array of byte;


  //семпл
  TIcon = record
    Icon: TByteArr;
    IconName:string;
    IconType: Cardinal;

  end;


type
    //Tlables =array of byte;
    Tlables =array of integer;
    Tlables_cnt=array of word;
    // інформація про сегменти образів
    Tsegments = record
    x:integer; // геометричне положення сегмента
    y:integer;
    top:integer;
    buttom:integer;
    left:integer;
    right:integer;
    segment_type:integer; // тип об'єкта - присвоюється підрограмою розпізнання
    size:integer;// площа об'єкта (для фільтра)
    end;
```

```
    //масив іконок
    TIconArr = array of TIcon;
    TsegmentArr=array of Tsegments;
var
  BitmapArr: TByteArr;
  RobotRecognition:TRobotRecognition;
  leter_types:array [0..6] of char;
  sort_segments:Tsegments;
  segments:TsegmentArr;
  IconArr: TIconArr; //масив семплів
  //параметри перетворення в семпл
  IconSize: Cardinal = 16;
  IConSize: Cardinal = 16;
  MinBPixelsPercent: Cardinal = 80;
  IconTypes:Cardinal = 0;
  F: TextFile;
  left,right,top,buttom:integer;
  neuro_answer:integer;
  lowerest:integer;

procedure segment(BitmapArr: TByteArr);

procedure reader(letters_count:integer);

procedure ConvertBitmapToMonoChrome(var QPSource: TQuickPixels; var BitmapArr:
TByteArr);

//процедура перетворить масив BitmapArr Nx байт (монохромний бітмап) у масив
//IconAtrr[0..IconSize - 1][0..IConSize - 1], розбиваючи на осередки масив
//BitmapArr, MinBPixelsPercent - мінімальне відсоток чорних пікселів в осередку,
//достатніх, щоб зробити чорним елемент меншого масиву IConArr
procedure BitmapToIcon(var BitmapArr: TByteArr; var Icon: TByteArr;
                           IconSize:Cardinal; IconSize: Cardinal;
                           MinBPixelsPercent: Cardinal);

// функція порівнює два масиви байт Nx і видає відсоток співпавших байт
function CompareIcons(Arr1: TByteArr; Arr2: TByteArr): Cardinal;

//додати семпл зазначеного типу
procedure AddSample(var BitmapArr: TByteArr; IconType: Cardinal);

//семпл букви Т рулить
function TSampleRule(var BitmapArr: TByteArr): boolean;

implementation

uses
  Main_NeuralNet_Hamming_,Picture_processing_Hamming,upreview;



// головна функція нитки розпізнавання образів:
// розпізнає весь екран цілком, як один образ
// або б'є по сегментах, і розпізнає кожний окремо

// Вона сама просить зробити їй масив бітмепа на наступній події захоплення
// зображення камерою, як тільки обробить  попередні дані
// тут же вважається й FPS
procedure TRobotRecognition.execute();
begin
  FreeOnTerminate := true;
while not(RobotRecognition.Terminated) do
begin
  if BitmapReady then
    begin

      recognition_run:=false;
      //debug:=false;
      BitmapReady:=false;
```

```
      if rectype =1 then
      begin
        segment(BitmapArr);

      end;
      if rectype =0 then
      begin

        TSampleRule(BitmapArr);

      end;
              tm_tick:=tm_tick+1;
    recognition_run:=true;
    end;
end;
end;

procedure ConvertBitmapToMonoChrome(var QPSource: TQuickPixels; var BitmapArr:
TByteArr);
var
  i,x,y: Integer;
  Pixel: Cardinal;
  Pixel, Pixel, Pixel: byte;
  PixelRes,oldPix,oldPixRes: Cardinal;
  oldPixSum:real;
  oldPixCnt:integer;
  Canvas: TCanvas;

begin

oldPixCnt:=0;
OldPix:=0;
oldPixRes:=1;

  Canvas:=AddSampleForm.SampleImage.Canvas;


  left:=QPSource.Width - 1;
  right:=0;
  top:=QPSource.Height - 1;
  buttom:=0;

  for y := 0 to QPSource.Height - 1 do
  begin

    for x := 0 to QPSource.Width - 1 do
    begin
      Pixel := QPSource.GetPixels24(x,y);
      {
      Pixel := Pixel shr 23;
      Pixel := Pixel shr 15;
      Pixel := Pixel shr 7;
      }
      Pixel := Pixel shr 23;
      Pixel:= Pixel shr 15;
      Pixel := Pixel shr 7;
      //PixelRes := (Pixel and Pixel) and Pixel;
      PixelRes := Pixel;// and Pixel ;

      BitmapArr[y,x] := PixelRes;

     if PixelRes=1 then
      begin
        if x>right then right:=x;
        if y>buttom then  buttom:=y;
        if x<left  then left:=x;
        if y<top  then top:=y;
      end;
```

```
      // що- те на подобі простенького фільтра

      //BitmapArr[y,x] := PixelRes and oldPixRes;

      //BitmapArr[y,x] := round(oldPixSum) and 1;
      if debug then if BitmapArr[y,x]<>OldPix then canvas.Pixels[x,y]:=ClRed;

      //oldPixSum:=abs(oldPixSum+( PixelRes-OldPix)/10);

      OldPix:=PixelRes;


      //oldPixRes:=round((oldPixRes+PixelRes)/10);
      //oldPixRes:=PixelRes;

      //QPDest.SetPixels1(j,i,PixelRes {* clWhite});
    end;

  end;


end;


// обчислення даних нейромережею
// і зняття даних
procedure NeuroCompute(Arr1: TByteArr);
  var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  cnt:integer;
  totle_layers,tmp,i:integer;
  true_exit:boolean;
  buff:string;
  Param,Param2:integer;

begin

  // вхідний параметр зняття даний з нейромережі
  Param:= 100-AddSampleForm.TrackBar1.Position;
  Param2:=AddSampleForm.TrackBar1.Position;

  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);


  // робимо вхідний вектор нейромережі з іконки (у модулів такий формат)
  cnt:=0;
  for x := 0 to Width - 2 do
    begin
      for y := 0 to Height - 2 do
        begin
      // багатошаровий
      if Arr1[y,x]=1 then  xInputVector[cnt] := 0 else xInputVector[cnt] := 1;
      //xInputVector[cnt] := Arr1[y,x];
      //Нейрона мережа Хопфілда
      //addsampleform.NeuralNet_Hamming_Hopf1.Layers[1].Neurons[cnt].Output :=
Arr1[y,x];
            cnt:=cnt+1;
        end;

    end;

    neuro_answer:=0;
    // нейромережа робить прогін у прямому напрямку
    addsampleform.NeuralNet_Hamming_BP1.Compute(xInputVector);

    totle_layers:=AddSampleForm.NeuralNet_Hamming_BP1.LayerCount-1;
```

```
    buff:='';

    // знімаємо даний з нейромережі
    for i := 0 to length(xOutputVector)-1 do
    begin

//xOutputVector[i]:=AddSampleForm.NeuralNet_Hamming_BP1.LayersBP[tmp].Neurons[i]
.Output;

tmp:=round(AddSampleForm.NeuralNet_Hamming_BP1.LayersBP[totle_layers].Neurons[i]
.Output*100);



    // суть алгоритму така - якщо якийсь із вихідних нейронів більше якогось
    // числа (наприклад 0.9, а всі інші вектора менше 0.1 кожний
    // те тоді відповіддю вважається той нейрон, що 0.9
    // ці параметри задаються з форми плазуючої
    // числа 0.9 і 0.1 актуальні по експериментах, але некритично
    // збільшити до 0.8 і 0.2, далі буде розпізнавати всі підряд.

    // подумав ще небагато: можна взагалі шукати максимальна відповідь,
    // як це зроблено у відсотках = тоді буде краще небагато
    // хоча міняти параметр нижче 0.8 однаково небезпечно - значить щось не так
    // на вхід іде.
    if tmp>Param then neuro_answer:=i+1; // собствено 0.9
    buff:=buff+' '+inttostr(tmp)
    end;
    true_exit:=true;
    for i := 0 to length(xOutputVector)-1 do
    begin
    if (neuro_ answer-1)<>i then
      begin

tmp:=round(AddSampleForm.NeuralNet_Hamming_BP1.LayersBP[totle_layers].Neurons[i]
.Output*100);
           if tmp>Param2 then true_exit:=false; // а це 0.1
      end;
    end;
    if not(true_exit) then neuro_answer:=0; // якщо не зустріли нічого іншого
              // більше чим 0.1 те даємо відповідь про успішний розпізнання,
              // інакше говоримо про те, що цей образ провалився.

    //mainform.NeuroCount.Text:=inttostr(neuro_answer);

end;



// це один з інстументів переприсвоєння міток,
// друга частина написана в правилах

function get_parent(var lables:Tlables;var lables_cnt:Tlables_cnt;
new_cnt:integer;test:integer):integer;
begin
  if (lables[new_cnt]<>0)  then result:=get_parent(lables,lables_cnt,
lables[new_cnt],test)
  else result:= new_cnt;
end;


function normalize(var lables:Tlables;var lables_cnt:Tlables_cnt;
index_cnt:integer;new_cnt:integer):integer;
var i,j:integer;
begin
      if lables[new_cnt]<>0 then
//      if lables[index_cnt]
      //lables[index_cnt]<>0
```

```
//      if lables[index_cnt]<>0 then
normalize(lables,lables_cnt,lables[index_cnt],index_cnt);
      //lables[index_cnt]:=new_cnt;
//      lables[new_cnt]:=index_cnt;
      //lables_cnt[index_cnt]
      //lables_cnt[new_cnt]:=lables_cnt[new_cnt]+lables_cnt[index_cnt];
      //lables_cnt[index_cnt]:=0;
      {
      for i:=1 to index_cnt do
        if lables[i]<>0 then
          begin
          for j:=1 to index_cnt do
            begin
            if lables[lables[i]]<>0 then
              begin
              lables[i]:=lables[lables[i]]; // і перепризначуємо влучні
              end;
            end;
          end;
        }
//result:= lables;
end;

// сегментація масиву
// у коментариях: L,m -lables
// цей же алгоритм застосовується в матлабе BWLABEL,
// реалізація цілком може бути іншою BWLABEL

// по суті - прохід по рядах і присвоєння міток за правилами,
// зазначеним нижче, якщо мітка вже привласнена комусь а треба НЕЮ привласнити
// поточну, то пошук її батька, і присвоєння ім'я батька.
// так само захист від присвоєння батька на самого себе.

// У результаті роботи алгоритму - після першого ж проходу масиву
// всі мітки посилаються або один на одного або на батька - він посилається на 0
// нумерація міток починається з 2 (т.до споконвічно масив містить тільки 1 і 0)

procedure segment( BitmapArr: TByteArr);
var i,ii,j :integer;
    width,height,x,y:integer;
    index_cnt:integer;
    lables:Tlables; // мітки, що розставляються спочатку
    lables_cnt:Tlables_cnt; // площа міток
    real_lables:array of integer; // кожна мітка  - унікальний масив
    //debug:boolean;
    seg_debug:textfile;
    seg_debug_file_name:string;
    flag:boolean;
    middel:integer;

begin


  setlength(real_lables,0);
  seg_debug_file_name:='data\seg.txt';
  //debug:=true;
  index_cnt:=1;
  setlength(lables,2);
  setlength(lables_cnt,2);
  height:=length(BitmapArr);
  width:=length(BitmapArr[0]);


  // прохід по масиві
  // первинна установка міток по нижченаписаним правилах
  for i:=1 to height-2 do
    begin
    for j:=1 to width-2 do
      begin
```

```
        if BitmapArr[i,j]>=1 then
            begin

            //   0       0
            // 0 1 ->  0 L

            if  (BitmapArr[i, j-1]=0) and
                (BitmapArr[ i-1,j]=0) then
                  begin
                  setlength(lables,length(lables)+1);
                  setlength(lables_cnt,length(lables_cnt)+1);
                  index_cnt:=index_cnt+1;
                  lables[index_cnt]:=0;
                  lables_cnt[index_cnt]:=1;
                  BitmapArr[i,j]:=index_cnt;
                  end;

            //   0       0
            // L 1 ->  L L

            if  (BitmapArr[i, j-1]>1) and
                (BitmapArr[ i-1,j]=0) then
                  begin
                  BitmapArr[i,j]:=BitmapArr[i, j-1];
                  lables_cnt[BitmapArr[i, j-1]]:=lables_cnt[BitmapArr[i, j-
1]]+1;
                  end;

            //   L       L
            // 0 1 ->  0 L

            if  (BitmapArr[i, j-1]=0) and
                (BitmapArr[ i-1,j]>1) then
                  begin
                  //BitmapArr[i,j]:=get_parent(lables,lables_cnt,BitmapArr[ i-
1,j]);
                  BitmapArr[i,j]:=BitmapArr[ i-1,j];
                  lables_cnt[index_cnt]:=lables_cnt[index_cnt]+1;
                  //lables_cnt[BitmapArr[i,j]]:=lables_cnt[BitmapArr[i,j]]+1;
                  end;

            //   L       L
            // L 1 ->  L L

            if  (BitmapArr[i, j-1]>1) and
                (BitmapArr[i, j-1]=BitmapArr[ i-1,j]) and
                (BitmapArr[ i-1,j]>1) then
                  begin
                  BitmapArr[i,j]:=BitmapArr[ i-1,j];
                  lables_cnt[BitmapArr[i, j-1]]:=lables_cnt[BitmapArr[i, j-
1]]+1;
                  end;


            //   L       L
            // M 1 ->  M L (M:=L)
            // якщо L не вказує на 0, то пошук її самого далекого родича
            // якщо M це далекий родич L те не призначити M лінк на саму себе

            if  ((BitmapArr[i, j-1]>1) and
                (BitmapArr[i, j-1]<>BitmapArr[ i-1,j]) and
               //(lables[BitmapArr[i, j-1]]=0) and
                (BitmapArr[ i-1,j]>1)) then
                  begin
                  if (lables[BitmapArr[ i-1,j]]<>BitmapArr[i, j-1])then
                  begin
                  middel:=get_parent(lables,lables_cnt,BitmapArr[ i-
1,j],BitmapArr[i, j-1]);
```

```
                  if (BitmapArr[i, j-1]<>middel) then
                  begin
                  lables[BitmapArr[i, j-1]]:=middel;
                  end;
                  BitmapArr[i,j]:=middel;//BitmapArr[ i-1,j];
                  end
                  //set_parent(lables,lables_cnt,index_cnt,BitmapArr[ i-1,j]);
                  else
                  begin
                  BitmapArr[i,j]:=get_parent(lables,lables_cnt,BitmapArr[ i-
1,j],BitmapArr[i, j-1]);
                  end;

                  end;

            //   L       L   M<>0
            // M 1 ->  M L (L:=M)
            end; // if BitmapArr[i,j]=1 then

//            assignfile(seg_debug,seg_debug_file_name);
//            append(seg_debug);

//             write(seg_debug,inttostr(BitmapArr[i,j]));
//            closefile(seg_debug);
      end; // for j
//      assignfile(seg_debug,seg_debug_file_name);
//      append(seg_debug);

//      writeln(seg_debug,'');
//    closefile(seg_debug);
    end; //for i

    // отже нормалізуємо масив посилань лейблів один на одного
    // у результаті повинні  вийти набагато менше лейблів, але кожний буде
    // унікальним об'єктом, і піде в підпрограму розпізнання зі своїм номером
    if debug then
      begin
      assignfile(seg_debug,seg_debug_file_name);
      rewrite(seg_debug);
          for i:=1 to height-2 do
            begin
            for j:=1 to width-2 do
              begin
              //if BitmapArr[i,j]>1 then  write(seg_debug,inttostr(1)) else
write(seg_debug,inttostr(0))
              write(seg_debug,inttostr(BitmapArr[i,j]));
              end;
              writeln(seg_debug,'');
            end;
      writeln(seg_debug,'----------');

      for i:=1 to index_cnt do
        writeln(seg_debug,inttostr(i)+' '+inttostr(lables[i])+'
'+inttostr(lables_cnt[i]));
      writeln(seg_debug,'----------');

      end;


      for i:=1 to index_cnt do
        if lables[i]<>0 then
          begin
          for j:=1 to index_cnt do
            begin
            if lables[lables[i]]<>0 then
              begin

              lables[i]:=lables[lables[i]]; // і перепризначуємо влучні
              end;
```

```
            end;
          end;

    // можна сполучити з попереднім
    //
    // !!!!!!!!!!
    // на цьому ж етапі краще шукати границі кожного образа, що б потім
    // не проходити масив заданого стільки разів ,скільки образів знайдене

    // Плюси : кількість повних проходів скоротитися ґрунтовно:
    // Мінуси...м.. напевно їх немає.. так що треба буде зробити обов'язково.
    for i:=0 to  index_cnt do
      begin
        if lables[i]<>0 then
         begin
          lables_cnt[lables[i]]:=lables_cnt[lables[i]]+lables_cnt[i];
//підсумуємо ваги
         end;
      end;

       // Заглушка алгоритму нормалізації - тимчасово!
{
  for i:=1 to height-2 do
    begin
    for j:=1 to width-2 do
      begin
        if BitmapArr[i,j]>1 then
            begin
            if  (BitmapArr[i, j-1]>1) and
                (BitmapArr[i, j-1]<>BitmapArr[ i-1,j]) and
                //(lables[BitmapArr[i, j-1]]=0) and
                (BitmapArr[ i-1,j]>1) then
                  begin
                  BitmapArr[i,j]:=BitmapArr[ i-1,j];

                  //lables[BitmapArr[ i-1,j]]:=BitmapArr[i, j-1];
                  //lables_cnt[BitmapArr[i, j-1]]:=lables_cnt[BitmapArr[i, j-
1]]+1;
                  // спробуємо навпаки
                  lables[BitmapArr[i, j-1]]:=BitmapArr[ i-1,j];
                  lables_cnt[BitmapArr[ i-1,j]]:=lables_cnt[BitmapArr[ i-
1,j]]+1;

                  end;

            end;
      end;

 }


    if debug then
      begin
      for i:=1 to index_cnt do
        writeln(seg_debug,inttostr(i)+' '+inttostr(lables[i])+'
'+inttostr(lables_cnt[i]));
      writeln(seg_debug,'----------');

      end;

    // тепер зробимо прохід по масиві ще раз і перепишемо мітки
    // на нормалізовані, не хочеться витрачати час на ще один прохід

    for i:=1 to height-2 do
    begin
    for j:=1 to width-2 do
      begin
        if BitmapArr[i,j]<>0 then
          begin
```

```
            if lables[BitmapArr[i,j]]<>0 then
             begin

               BitmapArr[i,j]:=lables[BitmapArr[i,j]];

             end;
           end;

      end;
    end;

    if debug then
//        if true then
      begin
//             assignfile(seg_debug,seg_debug_file_name);
//      rewrite(seg_debug);

          for i:=1 to height-2 do
            begin
            for j:=1 to width-2 do
              begin
              write(seg_debug,inttostr(BitmapArr[i,j]));
              end;
              writeln(seg_debug,'');
            end;
//                              closefile(seg_debug);
      end;

    // SetLength(IconArr,Length(IconArr) + 1);

    // дивимося які мітки були унікальними
    if debug then
      begin
      write(seg_debug,'----дивимося які мітки були унікальними-');
      end;

      for i:=2 to index_cnt do
        if lables[i]=0 then
         begin
          if length(real_lables)>0 then
            begin
            // перевірка мітки на унікальність
            flag:=true;
            for j:=0 to (length(real_lables)-1) do
              begin
                if real_lables[j]=i then flag:=false;
              end;
              // якщо так, те унікальна
              if flag then
                begin
                  setlength(real_lables,length(real_lables)+1);
                  real_lables[length(real_lables)-1]:=i;
                end;
            end
            else
            begin
              setlength(real_lables,length(real_lables)+1);
              real_lables[length(real_lables)-1]:=i;
            end
         end;

      if debug then
         begin

         writeln(seg_debug,'--Унікальні мітки--');
           for i:=0 to length(real_lables)-1 do
           begin
             writeln(seg_debug,inttostr(real_lables[i])+'
'+inttostr(lables_cnt[real_lables[i]]));
```

```
       end;


     end;


// ну от, переходимо до суті - властиво до розпізнавання образів:
// на цьому етапі вже є матриця, у якій всі помічено не одиничками,
// а цифрами, до якому сигменту все це ставиться

// тепер треба чи подивитися достатня площа об'єкта
// (або навпаки занадто великувата)
// і послати його розпізнаватися, заодно заготовивши структуру з його даними


char_cnt:=1;
setlength(segments,0);
j:=0;
i:=0;
if length(real_lables)>0 then
begin
for i:=0 to length(real_lables)-1 do
 begin
  if  lables_cnt[real_lables[i]]>min_size then
  begin

        left:=Width - 2;
        right:=0;
        top:=Height - 2;
        buttom:=0;

      for y := 0 to height-2 do
        begin

        for x := 0 to Width - 2 do
          begin

          if BitmapArr[y,x]=real_lables[i] then
            begin
            if x>right then right:=x;
            if y>buttom then  buttom:=y;
            if x<left  then left:=x;
            if y<top  then top:=y;
            end;
          end;
        end;

  //writeln(seg_debug,inttostr(real_lables[i]));
  //top:=0;buttom:= height-2;left:=0;right:= width-2;

  if (right<>0) and (buttom<>0) then
  begin
    setlength(segments,(length(segments)+1));


    // пропускаємо через підпрограму розпізнання
    imFeatures(BitmapArr,real_lables[i]);

    // і так само пропускаємо  через нейромережу
    NeuroCompute(BitmapArr3);
    segments[j].x:=segment_X;
    segments[j].y:=segment_Y;
    segments[j].size:=lables_cnt[real_lables[i]];
    segments[j].segment_type:=neuro_answer;
    j:=j+1;
  end;
  end;
 end;
```

```
     // що -те робимо з типами образів, наприклад читаємо або
     // виставляємо прапорці для інший програми робота.
     if (( j-1)>=0)and(j=length(segments)) then   reader( j-1);

     end;



    // дебаг закінчився
    if debug then
      begin
      closefile(seg_debug);
      end;

end;


procedure reader(letters_count:integer);
var i,j,ii:integer;
begin
          // прочитаний текст - для початку опустошаємо усе з попереднього кроку
          AddSampleForm.reader.Text:='';
         // сортуємо букви
{    if letters_count>1 then
    begin
    for i := letters_count downto 0 do
      begin
   //   if debug then writeln(F,inttostr(segments[i].x)+'
'+inttostr(segments[i].segment_type));
      for ii := 0 to i do
      if segments[ii].x > segments[ii+1].x then
      begin
       sort_segments := segments[ii];
       segments[ii] := segments[ii+1];
       segments[ii+1] := sort_segments;
       end;
      end;
    end;

 }

     for i:=0 to letters_count do
     begin
     if (segments[i].segment_type<length(leter_types))and
(segments[i].segment_type>0) then

AddSampleForm.reader.text:=AddSampleForm.reader.Text+leter_types[segments[i].seg
ment_type];
     end;


     if AddSampleForm.ComboBox2.ItemIndex=1 then
       SayText(AddSampleForm.reader.Text);

end;


//семпл букви Т рулить
// стара функція для розпізнання
// порівнює дві матриці (одна з екрана, інша з масиву еталонів)
// результат дає якщо кількість що збіглися пікселей більше якогось відсотка
// а так само якщо воно найбільше із всіх що збіглися

// по експериментах - нейромережа працює луше від 10% до 30% по цьому
// далі використовувати процентрое порівняння
// практично ні до чого
function TSampleRule(var BitmapArr: TByteArr): boolean;
var
  Icon: TByteArr;
```

```
  y,i,j: Integer;
  MaxCompareTPercent: Cardinal;
  MaxCompareNotTPercent: Cardinal;
  CurComparePercent: Cardinal;
  iconFind:integer;
begin
  Result := false;
  SetLength(Icon,IConSize);
  for y := 0 to IconSize - 1 do
    SetLength(Icon[y],IconSize);


    // якась заглушка.. чи не знаю потрібна чи зараз ні
    if  right<>0 then
  begin
  // перетворимо ВЕСЬ екран в іконку (з першого білого, до останнього білого
  // пікселя.
 imFeatures(BitmapArr,1);

  if length(BitmapArr3)>1 then
 Icon:=BitmapArr3;


  // для процентного порівняння
  MaxCompareTPercent := 0;
  MaxCompareNotTPercent := 0;
  CurComparePercent := 0;


  {
  writeln(F,'--Бітмап--');
  for i := 0 to IconSize - 1 do
  begin
    for j := 0 to IconSize - 1 do
      write(F,Icon[i,j]);
    writeln(F);
  end;
  writeln(F);
  }

  //Порівнюємо сіткою
  NeuroCompute(Icon);


  // порівнюємо по пікселям що співпали
  iconFind:=0;
  for y := 0 to High(IconArr) do
  begin
    CurComparePercent := CompareIcons(Icon,IconArr[y].Icon);
    {
    if (IconArr[y].IconType = Icon) and (CurComparePercent > MaxCompareTPercent)
then
      MaxCompareTPercent := CurComparePercent;
    if (IconArr[y].IconType = IconNot) and (CurComparePercent >
MaxCompareNotTPercent) then
      MaxCompareNotTPercent := CurComparePercent;
    }
     if CurComparePercent > MaxCompareTPercent then
     begin
     MaxCompareTPercent := CurComparePercent; iconFind:=IconArr[y].IconType;
     end;

  end;
  if MaxCompareTPercent < 80 then iconFind:=0;
  //MainForm.Caption := IntToStr(iconFind);
  //MainForm.Caption := IntToStr(MaxCompareTPercent);
  //MainForm.Caption := '-no-';
//  if (MaxCompareTPercent > MaxCompareNotTPercent) and (MaxCompareTPercent >
80) then
```

```
 //   begin {MainForm.Caption := 'OK';} Result := true; end;
 end;
end;

// генеруємо іконку заданого розміру з масиву
procedure AddSample(var BitmapArr: TByteArr; IconType: Cardinal);
var
  Icon: TByteArr;
  x,y: Integer;
  canvas:tcanvas;
begin
  SetLength(IconArr,Length(IconArr) + 1);

  SetLength(Icon,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(Icon[y],IconSize);

  SetLength(IconArr[High(IconArr)].Icon,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(IconArr[High(IconArr)].Icon[y],IconSize);

  //BitmapToIcon(BitmapArr,Icon,IconSize,IconSize,MinBPixelsPercent);

  //1canvas:=AddSampleForm.cellImage.canvas;

  for y := 0 to High(Icon) do
    for x := 0 to High(Icon[Low(Icon)]) do
    begin

      IconArr[High(IconArr)].Icon[y,x] := BitmapArr3[y,x];
      if Icon[y,x]=1 then


    end;

  IconArr[High(IconArr)].IconType := IconType;
end;

// функція порівнює два масиви байт Nx і видає відсоток співпавших байт
function CompareIcons(Arr1: TByteArr; Arr2: TByteArr): Cardinal;
var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  cnt:integer;
begin
  SamePixels := 0;
  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);


//  cnt:=0;

  if  debug  then
  begin
  for x := 0 to Width - 2 do
    begin
      for y := 0 to Height - 2 do
         write(F,Arr1[y,x]);
      write(F,' ');
      for y := 0 to Height - 2 do
        write(F,Arr2[y,x]);
      writeln(F,' ');
    end;
        writeln(F,' ');
  end;
  for y := 0 to Height - 2 do
    for x := 0 to Width - 2 do
    begin
```

```
      if Arr1[y,x]=1 then
      //xInputVector[cnt] := 1
      //Нейрона мережа Хопфілда
      //addsampleform.NeuralNet_Hamming_Hopf1.Layers[1].Neurons[cnt].Output := 1
      else
      //xInputVector[cnt] := 0;
      // Нейрона мережа Хопфілда
      //addsampleform.NeuralNet_Hamming_Hopf1.Layers[1].Neurons[cnt].Output := -
1;

      cnt:=cnt+1;

      if Arr1[y,x] = Arr2[y,x] then
        SamePixels := SamePixels + 1;
    end;
    //addsampleform.NeuralNet_Hamming_Hopf1.Calc;
    Result := Round(SamePixels * 100 / (Width * Height));
end;


// Ця процедура використовувалася раніше
// зараз уже не потрібна! - ніде не викликається

//процедура перетворить масив BitmapArr Nx байт (монохромний бітмап) у масив
//IconAtrr[0..IconSize - 1][0..IConSize - 1], розбиваючи на осередки масив
//BitmapArr, MinBPixelsPercent - мінімальне відсоток чорних пікселів в осередку,
//достатніх, щоб зробити чорним елемент меншого масиву IConArr
procedure BitmapToIcon(var BitmapArr: TByteArr; var Icon: TByteArr;
                           IconSize: Cardinal ; IconSize: Cardinal ;
                           MinBPixelsPercent: Cardinal );


var
  i,j,x,y,bx,by,ix,iy: Integer; //Лічильники
  CSize, CSize: Integer; //Розміри осередку
  BPorog: Cardinal; //поріг чорного
  BPixelsInCell: Cardinal; // кількість чорних пікселей в осередку
  xLeft, xRight, yTop, yBottom : integer;          // абс. коорд. образа
  line:string;
  const color:integer = 0;
begin


  // перетворимо трохи пікселей на ділянці в один осередок, зчитуємо колір
  CSize := Floor(Length(BitmapArr) / IconSize);
  CSize := Floor(Length(BitmapArr[Low(BitmapArr)]) / IconSize);

  Bporog := Round(CSize * CSize / 100 * MinBPixelsPercent);

  bx := 0;
  by := 0;
  BPixelsInCell := 0;

  for iy := 0 to IconSize - 1 do
    for ix := 0 to IconSize - 1 do
    begin
      for y := by to by + CSize - 1 do
        for x := bx to bx + CSize - 1 do
          if BitmapArr[y,x] = 0 then
            BPixelsInCell := BPixelsInCell + 1;
      if BPixelsInCell > BPorog then
        Icon[iy,ix] := 0
      else
        ICon[iy,ix] := 1;
//        line:=line+inttostr(ICon[iy,ix]);
      BPixelsInCell := 0;
      bx := bx + CSize;
      if Round(bx / CSize) >= IConSize then
      begin
```

```
        bx := 0;
        by := by + CSize;
      end;

    end;

end;

initialization
  AssignFile(F,'Sampledebug.txt');
  Rewrite(F);
  leter_types[0]:=' ';
  leter_types[1]:='П';
  leter_types[2]:='И';
  leter_types[3]:='М';
  leter_types[4]:='Л';
  leter_types[5]:='О';
  leter_types[6]:='Д';

finalization
  CloseFile(F);


end.
```

Файл Main_NeuralNet_Hamming_.pas - основна програма

```
unit Main_NeuralNet_Hamming_;

{---------Головний модуль програми керування--------}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, VCap,VCapStrings ,StdCtrls,DirectShow,
ExtCtrls,ComCtrls,ComObj,Active,Speech,
  UPreview, //модуль із компонентами для захоплення відео
  UGraphConfig, //модуль інтерфейс, що реалізує, настроювання камер
  USingleFrame, //модуль перегляд, що реалізує, окремих кадрів
  UTypeConst, //модуль утримуючі загальні типи й константи
  Can_Dll, //модуль імпортує фукції з Can.dll
  UCommunication, //модуль із фукціями для повідомлення по Кан'у
  URazvertka, //модуль циклічного розгорнення
  UCamTimers, //таймери камер
  UMoving, //руху
  URazvertkaConfig, //форма настроювання розгорнення
  URoboRootServer, //робосервер
  URoboServerConfig, //форма настроювання робосервера
  USystemCoordinat, //система координат
  URisovalka, //рисовалка
  UCommunicationForm, //форма висновку інформації про маяк
  Picture_Analyz_Hamming, //аналіз букв
  UQPixels, //швидкі пікселі
  Picture_processing_Hamming,
  URestaran,
  UDebug,
  UEmul,
  UDialog, BCPort,
  about;

const port = 'COM2';

type
  TMainForm = class(TForm)
    Button3: TButton;
    MoveForLine: TButton;
    CamsListBox: TListBox;
    Label7: TLabel;
    RefreshCamsListButton: TButton;
    CamConfigButton: TButton;
    CamsGroupBox: TGroupBox;
    DialogListBox: TListBox;
    Label1: TLabel;
    VideoModeComboBox: TComboBox;
    Label2: TLabel;
    RisovalkaButton: TButton;
    CommunicationButton: TButton;
    AnalyzFrameButton: TButton;
    RazvertkaButton: TButton;
    OnFlyDebugButton: TButton;
    Button1: TButton;
    Button2: TButton;
    TLabel1: TLabel;
    AddSampleButton: TButton;
    RestaranButton: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Label3: TLabel;
    Timer1: TTimer;
    Edit1: TEdit;
```

```
    Edit2: TEdit;
    Label4: TLabel;
    Button7: TButton;
    NeuroCount: TEdit;
    Button8: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure MovingFormButtonClick(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure MoveForLineClick(Sender: TObject);
    procedure RefreshCamsListButtonClick(Sender: TObject);
    procedure CamConfigButtonClick(Sender: TObject);
    procedure CamsListBoxClick(Sender: TObject);
    procedure DialogListBoxDblClick(Sender: TObject);
    procedure VideoModeComboBoxChange(Sender: TObject);
    procedure CommunicationButtonClick(Sender: TObject);
    procedure RisovalkaButtonClick(Sender: TObject);
    procedure AnalyzFrameButtonClick(Sender: TObject);
    procedure RazvertkaButtonClick(Sender: TObject);
    procedure OnFlyDebugButtonClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure AddSampleButtonClick(Sender: TObject);
    procedure RestaranButtonClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    {Для розгорнення}
    //настроїти параметри циклічного розгорнення
    procedure ConfigRazvertka(CamNum: Cardinal);
  end;

function Send:integer; //запис даних у буфер КАН'а для відправлення

type
  mass = array [1..250] of byte;
  reg_array = ^smallint;


function mbConnect(const port: string ; speed: integer;parity: integer;
stopbits: integer;flow: integer): integer;
 cdecl external 'MODBUS.dll' ;
function mbDisconnect(): integer;
 cdecl external 'MODBUS.dll';
function mbExecuteProgramFile(dev: cardinal; filename: string): integer;
 cdecl external 'MODBUS.dll';
function mbReadHoldingRegisters  (dev: cardinal; dest: reg_array; address:
cardinal; count: cardinal): integer;
 cdecl external 'MODBUS.dll';
FUNCTION mbReportDeviceID(dev: CARDINAL;  dest: mass; max_len: cardinal;
actual_len : Pointer): integer;
 cdecl external 'MODBUS.dll';
function mbReset( dev: cardinal): integer;
 cdecl external 'MODBUS.dll';
function mbSetLogDetails(log_errors: boolean;log_messages: boolean;log_data:
boolean): integer;
 cdecl external 'MODBUS.dll';
function mbWriteHoldingRegisters(dev: integer; from: reg_array; address:
cardinal; count: cardinal): integer;
 cdecl external 'MODBUS.dll';
```

```
var
  MainForm: TMainForm; //головна форма

  //масив настроювань камер
  GraphConfigExArr: TGraphConfigExArr;

  //об'єкт головного потоку робосервера
  RoboRootServerManageThread: TRoboRootServerManagThread;

   //прапор "рух по смузі"
  MoveForLineFlag: boolean = false;

  //камера для висновку відео
  ActiveCamIndex: Integer = -1;

  {для TextToSpeech}
  {Центральний інтерфейс,через який виробляються всі дії з мовою}
  fITTSCentral: ITTSCentral;

  {Інтерфейс для зв'язку з аудіопристроєм}
  fIAMM: IAudioMultimediaDevice;

  {Інтерфейс для перебору движків}
  aTTSEnum: ITTSEnum;

  {Покажчик на параметри движка}
  fpModeInfo: PTTSModeInfo;

  NumFound : DWord;
  ModeInfo : TTSModeInfo;

  QPixels: TQuickPixels;
  timercnt: Cardinal = 0;


   i: integer;
   dest1: mass;


{процедура виводить в ListBox
список доступних діалогів у компоненті VideoCapture}
procedure ShowAvialableDialogs(ListBox: TListBox; VideoCapture: TVideoCapture);

{ функція зберігає відео настроювання у файл зі змінних
У разі удачі повертає true, інакше false}
function SaveGraphConfig(var GraphConfigExArr: TGraphConfigExArr): boolean;

{ функція ініціалізує камери
VideoDeviceList - список всіх доступних камер
у випадку удачі функція повертає true, інакше false}
function InitCams(var GraphConfigExArr: TgraphConfigExArr;
                        VideoDeviceList: TStringList): boolean;

//процедура виводить в ComboBox доступні відео режими на компоненті VideoCapture
procedure ShowVideoModes(ComboBox: TComboBox; VideoCapture: TVideoCapture);

//процедура ініціалізує відео настроювання для камери CamName за замовчуванням
procedure DefaultInitCam(CamName: String; var GraphConfigEx: TGraphConfigEx);

//---------------------------------i


//процедура, що вимовляє текст
procedure SayText(Text: String);

function GetVideoDevicesListEm(): TStringList;
```

```
implementation

uses Math;

{$R *.dfm}

function GetVideoDevicesListEm(): TStringList;
begin
  Result := TStringList.Create;
  Result.Add('Samsung SuperShit Cam');
  Result.Add('Hitachi MegaSlow WebCam');
  Result.Add('Електроніка КХ - 1');
end;


{Допоміжні процедури, що не входять у клас форми}

//процедура, що вимовляє текст
procedure SayText(Text: String);
var
  SData: TSData;
begin
  {Цей текст буде прочитаний}
  SData.dwSize := length(Text) + 1;
  SData.pData := pChar(Text);

  fITTSCentral.TextData(CHARSET_TEXT, 0, SData, nil, IID_ITTSBufNotifySink);
end;
//---------------------------------i

{процедура виводить в ListBox
список доступних діалогів у компоненті VideoCapture}
procedure ShowAvialableDialogs(ListBox: TListBox; VideoCapture: TVideoCapture);
var
  d: TCaptureDialog; //всі можливі ідентифікатори діалогів
begin
  ListBox.Clear; //очищення ListBox
  //додавання в ListBox всіх доступних діалогів
  for d := Low(TCaptureDialog) to High(TCaptureDialog) do
  if d in VideoCapture.Dialogs then
    ListBox.Items.AddObject(DialogTitles[d],TObject(d));

end;
//---------------------------------i

{ функція зберігає відео настроювання у файл зі змінних,
у разі удачі повертає true, інакше false}
function SaveGraphConfig(var GraphConfigExArr: TGraphConfigExArr): boolean;
var
  GraphConfigFile: TextFile; //файлова змінна
  GraphConfigStr: String; //рядок з відео настроюваннями
  i: integer; //лічильник
begin
  Result := true;

  //відкриття файлу
  AssignFile(GraphConfigFile,GraphConfigFileName);
  {$ I-I-}
  Rewrite(GraphConfigFile);

  for i := 0 to High(GraphConfigExArr) do
  begin
    //одержання настроювань у строковому форматі
    GraphConfigStr := GraphConfigExArr[i].GraphConfig.SaveGraph;
    //запис у файл
    writeln(GraphConfigFile,GraphConfigStr);
    writeln(GraphConfigFile,GraphConfigExArr[i].ShowPreview);
    writeln(GraphConfigFile,GraphConfigExArr[i].CamFunc);
    writeln(GraphConfigFile,GraphConfigExArr[i].TimerDelay);
```

```
  end;

  //закриття файлу
  CloseFile(GraphConfigFile);
  {$I+}
  if IOResult <> 0 then Result := false;
end;
//---------------------------------i

{ функція ініціалізує камери
VideoDeviceList - список всіх доступних камер
у випадку удачі функція повертає true, інакше false}
function InitCams(var GraphConfigExArr: TGraphConfigExArr;
                        VideoDeviceList: TStringList): boolean;
var
  GraphConfigFile: TextFile; //файлова змінна
  CurLine: String; //рядок файлу
  i: integer; //лічильник
  CamsInitStat: array of boolean; //стан ініціалізованості камер
  TmpConfig: TGraphConfig;//тимчасове зберігання настроювань

begin
  Result := true;

  //розміри масивів
  SetLength(GraphConfigExArr,VideoDeviceList.Count);
  SetLength(CamsInitStat,VideoDeviceList.Count);

  //заповнюємо
  for i := 0 to High(CamsInitStat) do
    CamsInitStat[i] := false;

  //якщо є файл із настроюваннями відео - зчитуємо з нього рядка настроювань
  if FileExists(GraphConfigFileName) then
  begin
    //ініціалізація
    TmpConfig := TGraphConfig.Create;

    //присвоєння файлу
    AssignFile(GraphConfigFile,GraphConfigFileName);

    //проходимо за списком доступних камер і шукаємо їхнього настроювання у
файлі
    for i := 0 to VideoDeviceList.Count - 1 do
    begin
      {$ I-I-}
      Reset(GraphConfigFile);

      //поки не скінчився файл або поки не знайшли настроювання поточної камери
      while (not EOF(GraphConfigFile)) and (not CamsInitStat[i]) do
      begin
        //читаємо рядок з головними настроюваннями
        readln(GraphConfigFile,CurLine);
        TmpConfig.RestoreGraph(CurLine);

        //якщо ці настроювання для поточної камери, те привласнюємо їх їй
        if TmpConfig.VCapSource = VideoDeviceList.Strings[i] then
        begin
          GraphConfigExArr[i].GraphConfig := TGraphConfig.Create;

          GraphConfigExArr[i].GraphConfig.RestoreGraph(CurLine);

          readln(GraphConfigFile,CurLine);
          GraphConfigExArr[i].ShowPreview := StrToInt(CurLine);

          readln(GraphConfigFile,CurLine);
          GraphConfigExArr[i].CamFunc := StrToInt(CurLine);

          readln(GraphConfigFile,CurLine);
```

```
          GraphConfigExArr[i].TimerDelay := StrToInt(CurLine);

          //ця камера проиниціалазирована!
          CamsInitStat[i] := true;
        end
        else
        begin
          //перелистуємо 3-и рядка з іншими настроюваннями
          readln(GraphConfigFile);
          readln(GraphConfigFile);
          readln(GraphConfigFile);
        end;
      end;

      //закриття файлу
      CloseFile(GraphConfigFile);
      {$I+}
      if IOResult <> 0 then
      begin
        Result := false;
        Exit;
      end;
    end;
  end;
  //якщо залишилися не проініціалізовані камери, ініціалізуємо їх за
замовчуванням
  for i := 0 to High(CamsInitStat) do
  begin
    if not CamsInitStat[i] then
      DefaultInitCam(VideoDeviceList.Strings[i],GraphConfigExArr[i]);
  end;
end;
//---------------------------------

//процедура виводить в ComboBox доступні відео режими на компоненті VideoCapture
procedure ShowVideoModes(ComboBox: TComboBox; VideoCapture: TVideoCapture);
var
  i: integer; //лічильник
  CurVideoMode: TVCapMode; //поточний відео режим
begin
  ComboBox.Clear; //очищення
  CurVideoMode := VideoCapture.VCapMode; //запам'ятовуємо тек. відео режим

  //у циклі виводимо доступні й визначаємо поточний відео режими
  for i := 0 to VideoCapture.VCapModeCount - 1 do
  begin
    ComboBox.Items.Add(GetModeString(VideoCapture.VCapModes[i]));
    if IsEqualModes(CurVideoMode,VideoCapture.VCapModes[i]) then
      ComboBox.ItemIndex := i;
  end;
end;
//---------------------------------

//процедура ініціалізує відео настроювання для камери CamName за замовчуванням
procedure DefaultInitCam(CamName: String; var GraphConfigEx: TGraphConfigEx);
begin
 //створення об'єкта основних настроювань камери
 GraphConfigEx.GraphConfig := TGraphConfig.Create;

 with GraphConfigEx.GraphConfig do
  begin
    //ім'я камери
    VCapSource := CamName;

    //аудіо пристрою не використовуємо
    ACapSource := '';

    //аудіо компресори не використовуємо
    AComp := '';
```

```
    //імена файлів для запису відео
    CaptureFileName := 'Capture.avi';
    TempCaptureFileName := 'TempCapture.avi';
    PreallocFileSize := 0;

    //що хочемо одержати
    WantCapture := false;
    WantPreview := false;
    WantBitmaps := false;

    //аудіо не потрібно
    WantAudio := false;
    WantDVAudio := false;
    WantAudioPreview := false;

    //тимчасовий файл
    UseTempFile := true;
    DoPreallocFile := false;
    PreallocFileSize := 0;

    //формат пікселя
    PixelFormat := pfDevice;
    //???
    DVResolution := dvrDontWorry;

    //настроювання відео режиму
    with VCapMode do
    begin
      MediaType := MEDIATYPE_Video;
      MediaSubType := MEDIASUBTYPE_RGB24;
      Width := 640;
      Height := 480;
      BitCount := 24; //???
      FrameRate := 30.000;
      MinFrameRate := 30.000;
      MaxFrameRate := 30.000;
    end;
  end;
  GraphConfigEx.ShowPreview := 0; //потрібно чи показувати форму із зображенням
  GraphConfigEx.CamFunc := CamFuncNone; //камера не функціональна
  GraphConfigEx.TimerDelay := 65; //частота обробки 65 мс
end;
//---------------------------------

{допоміжні процедури, що входять у клас форми
 у всіх процедурах: CamNum - номер камери,
 над якою виробляється дія}

{Для розгорнення}
//настроїти параметри циклічного розгорнення
procedure TMainForm.ConfigRazvertka(CamNum: Cardinal);
begin
  //набудовуємо перше розгорнення
  URazvertkaConfig.LocalRazvertkaConfig := @RazvertkaArr[CamsListBox.ItemIndex];
  RazvertkaConfigForm.ShowModal;
  RazvertkaArr[CamsListBox.ItemIndex].SetRazvertkaConfig(LocalRazvertkaConfig^);
end;
//---------------------------------

{Оброблювачі подій}

//Подія - перед створенням форми
{{Тут відбувається ініціалізація настроювань компонентів і змінних}
procedure TMainForm.FormCreate(Sender: TObject);
var
  VideoDeviceList: TStringList; //список відео пристроїв
  Res1,Res2: HRESULT;
begin
```

```
  {для text to speech}
  {Ініціалізація аудіопристрою}
  Res1 := CoCreateInstance(CLSID_MMAudioDest, Nil,
CLSCTX_ALL,IID_IAudioMultiMediaDevice, fIAMM);
  {Створення об'єкта, що перераховується, для перебору всіх движків у системі за
допомогою інтерфейсу ITTSEnum}
  Res2 := CoCreateInstance(CLSID_TTSEnumerator, Nil,
CLSCTX_ALL,IID_ITTSEnum,aTTSEnum);

  if (Res1 = S_OK) and (Res2 = S_OK) then
  begin
    aTTSEnum.Reset;//Скидаємо на перший
    {Одержуємо другий движок}
    aTTSEnum.Next(1, ModeInfo, @NumFound);
    aTTSEnum.Next(1, ModeInfo, @NumFound);
    aTTSEnum.Select(ModeInfo.gModeID, fITTSCentral, IUnknown(fIAMM));
    {Одержуємо інші}
    {While NumFound > 0 do
     begin
       ComboBox1.Items.Add(String(ModeInfo.szModeName));
       aTTSEnum.Next(1, ModeInfo, @NumFound);
     end;}
  end;
  //ініціалізація модуля спілкування
  Communication := TCommunication.Create;
  Communication.Start;

  //створюємо об'єкт системи координат
  SystemKoordinat := TSystemKoordinat.Create;
  SystemKoordinat.Start;

  //одержуємо список доступних камер
  VideoDeviceList := GetVideoDevicesList();

  //запуск головного потоку робосервера
  if RoboRootServerActive then
    RoboRootServerManageThread := TRoboRootServerManagThread.Create(false);

  //якщо є хоча б одна камера
  if VideoDeviceList.Count > 0 then
    URoboRootServer.VideoExist := true;

  //потрібно проініціалізувати настроювання камер
  InitCams(GraphConfigExArr,VideoDeviceList);

  //заповнюємо список камер
  CamsListBox.Items.Assign(VideoDeviceList);

end;
//---------------------------------

//Подія - перед відображенням форми
//Тут активуються компоненти й настроюються керуючі елементи
//(кнопки, форми й т.д.)
procedure TMainForm.FormShow(Sender: TObject);
var
  i: integer; //лічильник
begin
  //створення масиву компонентів для роботи з камерами
  SetLength(VideoCaptureArr,Length(GraphConfigExArr));
  for i := 0 to High(VideoCaptureArr) do
  begin
    VideoCaptureArr[i] := TVideoCapture.CreateParented(PreviewWindow.Handle);
    VideoCaptureArr[i].RestoreGraph(GraphConfigExArr[i].GraphConfig);
  end;

  //завдання розміру масиву оброблювачів захоплення кадру
  SetLength(BitmapGrabEventArr,Length(GraphConfigExArr));
```

```
  //створення об'єктів оброблювачів кадрів
  for i := 0 to High(GraphConfigExArr) do
  begin
    BitmapGrabEventArr[i] := TBitmapGrabEvent.Create(i);
    VideoCaptureArr[i].OnBitmapGrabbed := BitmapGrabEventArr[i].AnalyzBitmap;
  end;

  //завдання розміру масиву таймерів камер
  SetLength(CamTimerArr,Length(GraphConfigExArr));

  //створення й запуск таймерів для потрібних камер
  for i := 0 to High(GraphConfigExArr) do
  begin
    CamTimerArr[i] := TCamTimer.Create(i);
    if GraphConfigExArr[i].GraphConfig.WantBitmaps then
    begin
      CamTimerArr[i].StartTimer(GraphConfigExArr[i].TimerDelay);
    end;
  end;

  //задаємо розмір масивам розгорнення
  SetLength(RazvertkaArr,Length(GraphConfigExArr));
  SetLength(RazvertkaConfigArr,Length(GraphConfigExArr));

  //задаємо настроювання для розгорнень
  for i := 0 to High(RazvertkaArr) do
  begin
    with RazvertkaConfigArr[i] do
    begin
      a := VideoCaptureArr[i].VCapMode.Width div 2;
      a_corr := 5;
      b := VideoCaptureArr[i].VCapMode.Height div 2;
      y_offset := 0;
      klaster_size := 5;
      porog := 50;
      step := 3;
      fi_step := 0.005;
      RazvertkaArr[i] := TRazvertka.Creat(RazvertkaConfigArr[i]);
    end;
  end;
  //кнопка для руху
  if not Communication.PortEn then
  begin
   // RisovalkaButton.Enabled := false;
  end;
end;
//---------------------------------


//Подія - перед закриттям форми
procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
var
  i: integer;
begin
  //зупинка системи координат
  SystemKoordinat.Stop;
  SystemKoordinat.Destroy;


  //зупинка каналу
  Communication.Destroy;

  //зупинка робосервера
    if RoboRootServerActive then
    RoboRootServerManageThread.Terminate;
```

```
  //зупинка таймерів камер
  for i := 0 to High(CamTimerArr) do
  begin
    if CamTimerArr[i] <> nil then
      CamTimerArr[i].Destroy;
  end;
  SetLength(CamTimerArr,0);

  //знищення оброблювачів події захоплення кадру з камер
  for i := 0 to High(BitmapGrabEventArr) do
  begin
    if BitmapGrabEventArr[i] <> nil then
      BitmapGrabEventArr[i].Destroy;
  end;
  SetLength(BitmapGrabEventArr,0);

  //зупинка й знищення об'єктів для захоплення відео
  for i := 0 to High(VideoCaptureArr) do
  begin
    if VideoCaptureArr[i].Capturing then
      VideoCaptureArr[i].StopCapture;
    if VideoCaptureArr[i].Previewing then
      VideoCaptureArr[i].StopPreview;
    VideoCaptureArr[i].Destroy;
  end;
  SetLength(VideoCaptureArr,0);

  //знищення розгорнень і їхніх настроювань
  for i := 0 to High(RazvertkaArr) do
  begin
    RazvertkaArr[i].Destroy;
  end;
  SetLength(RazvertkaArr,0);
  SetLength(RazvertkaConfigArr,0);
end;
//-----------------------------------

//Подія - натискання на "Рухи"
procedure TMainForm.MovingFormButtonClick(Sender: TObject);
begin
  MovingForm.ShowModal;
end;
//-----------------------------------

//Подія - Натискання "Сервер"
procedure TMainForm.Button3Click(Sender: TObject);
begin
  RoboServerConfigForm.ShowModal;
end;
//-----------------------------------

//Подія - Натискання на "Рух по смузі"
procedure TMainForm.MoveForLineClick(Sender: TObject);
begin
  ForwSpeed := 30;
  MoveForLineFlag := not MoveForlineFlag;
end;
//-----------------------------------

//Подія - натискання на "Обновити" (список камер)
procedure TMainForm.RefreshCamsListButtonClick(Sender: TObject);
var
  VideoDeviceList: TStringList; //список камер
  i: integer; //лічильник
begin
  //одержуємо список камер
  VideoDeviceList := GetVideoDevicesList(true);

  //ініціалізуємо камери заново
```

```
  InitCams(GraphConfigExArr,VideoDeviceList);

  //знищення об'єктів для вже неіснуючих камер
  for i := VideoDeviceList.Count to High(VideoCaptureArr) do
  begin
    CamTimerArr[i].StopTimer;
    CamTimerArr[i].Destroy;
    BitmapGrabEventArr[i].Destroy;
  end;

  //установлюємо нові розміри масивів
  SetLength(VideoCaptureArr,VideoDeviceList.Count);
  SetLength(BitmapGrabEventArr,VideoDeviceList.Count);
  SetLength(CamTimerArr,VideoDeviceList.Count);
  //у циклі створюємо потрібні об'єкти
  for i := 0 to High(VideoCaptureArr) do
  begin
    //об'єкти для захоплення відео
    if VideoCaptureArr[i] = nil then
    begin
      VideoCaptureArr[i] := TVideoCapture.CreateParented(PreviewWindow.Handle);
      VideoCaptureArr[i].RestoreGraph(GraphConfigExArr[i].GraphConfig);
    end;
    //події захоплення кадру
    if BitmapGrabEventArr[i] = nil then
    begin
      BitmapGrabEventArr[i] := TBitmapGrabEvent.Create(i);
      VideoCaptureArr[i].OnBitmapGrabbed := BitmapGrabEventArr[i].AnalyzBitmap;
    end;
    //таймери для захоплення кадрів
    if (CamTimerArr[i] = nil) then
    begin
      CamTimerArr[i] := TCamTimer.Create(i);
      if GraphConfigExArr[i].GraphConfig.WantBitmaps then
        CamTimerArr[i].StartTimer(GraphConfigExArr[i].TimerDelay);
    end;
  end;

  //заповнюємо список на формі
  CamsListBox.Clear;
  for i := 0 to VideoDeviceList.Count - 1 do
    CamsListBox.Items.Add(VideoDeviceList.Strings[i]);
end;
//---------------------------------

//Подія - натискання на "Настроювання" (обраної камери)
procedure TMainForm.CamConfigButtonClick(Sender: TObject);
begin
  if CamsListBox.ItemIndex >= 0 then
  begin
    //передаємо індекс настроювань обраної камери в модуль настроювання камери
    UGraphConfig.GraphConfigExIndex := CamsListBox.ItemIndex;
    //виводимо форму настроювань камери в модальному режимі
    if UGraphConfig.GraphConfigForm.ShowModal = mrOK then
    begin
      //зберігаємо й відновлюємо настроювання камери
      SaveGraphConfig(GraphConfigExArr);

VideoCaptureArr[CamsListBox.ItemIndex].RestoreGraph(GraphConfigExArr[CamsListBox
.ItemIndex].GraphConfig);

      //запускаємо або перезапускаємо або зупиняємо таймери якщо потрібно
      if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
      begin
        if CamTimerArr[CamsListBox.ItemIndex].Active then
          CamTimerArr[CamsListBox.ItemIndex].StopTimer;

CamTimerArr[CamsListBox.ItemIndex].StartTimer(GraphConfigExArr[CamsListBox.ItemI
ndex].TimerDelay);
```

```
      end
      else
      begin
        if CamTimerArr[CamsListBox.ItemIndex].Active then
          CamTimerArr[CamsListBox.ItemIndex].StopTimer;
      end;

      //якщо потрібно показувати зображення на екрані
      if GraphConfigExArr[CamsListBox.ItemIndex].ShowPreview = 1 then
      begin
        ActiveCamIndex := CamsListBox.ItemIndex;
        PreviewWindow.ShowEx;
      end
      else
      begin
        PreviewWindow.Hide;
        ActiveCamIndex := -1;
      end;
      //обновляємо діалоги й відео режими

ShowAvialableDialogs(DialogListBox,VideoCaptureArr[CamsListBox.ItemIndex]);
      ShowVideoModes(VideoModeComboBox, VideoCaptureArr[CamsListBox.ItemIndex]);

      //кнопки
      if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
      begin
        AnalyzFrameButton.Enabled := true;
        OnFlyDebugButton.Enabled := true;
      end
      else
      begin
        AnalyzFrameButton.Enabled := false;
        OnFlyDebugButton.Enabled := false;
      end;
    end;
  end
  else
    ShowMessage('Не обрана камера!');
end;
//----------------------------------

//подія - натискання на Списку камер
procedure TMainForm.CamsListBoxClick(Sender: TObject);
begin
  //виводимо картинку якщо потрібно
  if GraphConfigExArr[CamsListBox.ItemIndex].ShowPreview = 1 then
  begin
    ActiveCamIndex := CamsListBox.ItemIndex;
    PreviewWindow.ShowEx;
  end
  else
  begin
    PreviewWindow.Hide;
    ActiveCamIndex := -1;
  end;
  //виводимо діалоги й відео режими для обраної камери
  ShowAvialableDialogs(DialogListBox,VideoCaptureArr[CamsListBox.ItemIndex]);
  ShowVideoModes(VideoModeComboBox, VideoCaptureArr[CamsListBox.ItemIndex]);

  //ім'я камери
  PreviewWindow.Caption :=
GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapSource;

  //кнопки
  if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
  begin
    AnalyzFrameButton.Enabled := true;
    OnFlyDebugButton.Enabled := true;
  end
```

```
  else
  begin
    AnalyzFrameButton.Enabled := false;
    OnFlyDebugButton.Enabled := false;
  end;
end;
//-----------------------------------

//подія - подвійний натискання на списку діалогів камери
procedure TMainForm.DialogListBoxDblClick(Sender: TObject);
var
  i: integer; //лічильник
begin
  //шукаємо виділений рядок, щоб викликати діалог, ім'я якого в ній написано
  for i := 0 to DialogListBox.Count - 1 do
    if DialogListBox.Selected[i] then
    begin
      MainForm.Enabled := false;

UPreview.VideoCaptureArr[CamsListBox.ItemIndex].ShowDialog(TCaptureDialog(Dialog
ListBox.Items.Objects[i]));
      MainForm.Enabled := true;
    end;
  //зберігаємо новий відео режим
  GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapMode :=
VideoCaptureArr[CamsListBox.ItemIndex].VCapMode;
  SaveGraphConfig(GraphConfigExArr);
  //виводимо сталий режим у списку відео режимів

ShowVideoModes(VideoModeComboBox,UPreview.VideoCaptureArr[CamsListBox.ItemIndex]
);
end;
//-----------------------------------

//Подія - зміна в списку відео режимів
procedure TMainForm.VideoModeComboBoxChange(Sender: TObject);
begin
  //Установлюємо новий відео режим

VideoCaptureArr[CamsListBox.ItemIndex].SetVCapMode(VideoModeComboBox.ItemIndex);
  //змінюємо настроювання
  GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapMode :=
    VideoCaptureArr[CamsListBox.ItemIndex].VCapMode;
  //зберігаємо настроювання
  SaveGraphConfig(GraphConfigExArr);
  //якщо потрібно, те возобнавляем висновок на екран
  if GraphConfigExArr[CamsListBox.ItemIndex].ShowPreview = 1 then
    VideoCaptureArr[CamsListBox.ItemIndex].StartPreview;
end;
//-----------------------------------

//Подія - натискання на "Маяки"
procedure TMainForm.CommunicationButtonClick(Sender: TObject);
begin
  CommunicationForm.Show;
end;
//-----------------------------------

//Подія - натискання на "Рисовалка"
procedure TMainForm.RisovalkaButtonClick(Sender: TObject);
begin
 OKRightDlg.Visible:=true;
// MainForm.Hide;
//  URisovalka.RisovalkaForm.Visible := true;
end;
//-----------------+----------------i

//Подія - Натискання на "Аналіз Кадру"
procedure TMainForm.AnalyzFrameButtonClick(Sender: TObject);
```

```
begin
  if CamsListBox.ItemIndex >= 0 then
  begin
    BitmapGrabEventArr[CamsListBox.ItemIndex].AnalyzFrame := true;
    FrameForm.ShowModal;
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------

//Подія - натискання на "Розгорнення"
procedure TMainForm.RazvertkaButtonClick(Sender: TObject);
begin
  //виводимо форму настроювань розгорнення
  if CamsListBox.ItemIndex >= 0 then
  begin
    URazvertkaConfig.LocalRazvertkaConfig :=
@RazvertkaConfigArr[CamsListBox.ItemIndex];
    RazvertkaConfigForm.ShowModal;

RazvertkaArr[CamsListBox.ItemIndex].SetRazvertkaConfig(URazvertkaConfig.LocalRaz
vertkaConfig^);
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------

//Подія - натискання на "дебаг на льоту"
procedure TMainForm.OnFlyDebugButtonClick(Sender: TObject);
begin
  //якщо обрано камеру
  if CamsListBox.ItemIndex >= 0 then
  begin
    //установлюємо прапорець для дебага на льоту й виводимо форму
    BitmapGrabEventArr[CamsListBox.ItemIndex].OnFlyDebug := true;
    FrameForm.ShowModal;
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------


procedure TMainForm.Button1Click(Sender: TObject);
begin
  Communication.stopsignal := 1;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
  Communication.stopsignal := 0;
end;
//-----------------------------------

//Подія - натискання "Семпли"
procedure TMainForm.AddSampleButtonClick(Sender: TObject);
begin
  if CamsListBox.ItemIndex >= 0 then
  begin
    BitmapGrabEventArr[CamsListBox.ItemIndex].NeedAddSample := true;
  end
  else
    Picture_processing_Hamming.AddSampleForm.ShowModal;
end;

procedure TMainForm.RestaranButtonClick(Sender: TObject);
begin
```

```
  Restaran := TRestaran.Create;
  Restaran.Start(1);
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
  UDebug.DebugForm.Show;
end;

procedure TMainForm.Button6Click(Sender: TObject);
begin
 //mbDisconnect();
 //MainForm.Label3.Caption:='Порт закритий';
end;



function Send:integer;
begin
   result:=mbWriteHoldingRegisters(1,@(Communication.RecvBuff),0,3);
 // mbWriteHoldingRegisters(1,@SendBuff,0,4);
 //result:=SENDMSG(SendMsNb, CANSendBuff)
end;


procedure TMainForm.Button5Click(Sender: TObject);
var
  len_written: integer;
  speeds: array [0..1] of smallint;
begin
  //mbSetLogDetails(true,true,true);
  mbSetLogDetails(false,false,false);
  i:=mbConnect(port,115200,1,0,3);

  if i=0 then
  begin
   MainForm.Label3.Caption:='не вдалося відкрити порт';
  end
  else
  begin
   MainForm.Label3.Caption:='порт відкритий';


   mbReset(1);
   sleep(10);

   mbExecuteProgramFile(1,'image.raw');

   sleep(10);
   mbReportDeviceID(1,dest1,250,(@len_written));

   Communication.PortEn:=true;
  end;



   MainForm.Timer1.Enabled:=true;

   //Speeds[0]:=100;
   //Speeds[1]:=-100;

  // Communication.RecvBuff.w1:=100;
  // Communication.RecvBuff.w2:=-100;
  // Communication.RecvBuff.w3:=0;

   // Send;
  // mbWriteHoldingRegisters(1,@(Communication.RecvBuff),0,3);
end;
```

```
procedure TMainForm.Timer1Timer(Sender: TObject);
begin
//ghgf
end;

procedure TMainForm.Button7Click(Sender: TObject);
var i1,count,tmp,i:integer;
n_debug:textfile;
begin

    assignfile(n_debug,neuro_debug_file_name);
    append(n_debug);

    count:=0;
    tmp:=AddSampleForm.NeuralNet_Hamming_BP1.LayerCount-1;
    NeuroCount.Text:='';

    //    for i1:=0 to length(xInputVector) do
    //    write(n_debug,inttostr(round(xInputVector[i1])));
    //    writeln(n_debug,'');

    addsampleform.NeuralNet_Hamming_BP1.Compute(xInputVector);


    //addsampleform.NeuralNet_Hamming_BP1.
    for i := 0 to length(xOutputVector)-1 do
    begin

xOutputVector[i]:=AddSampleForm.NeuralNet_Hamming_BP1.LayersBP[tmp].Neurons[i].O
utput;
    NeuroCount.Text:=NeuroCount.Text+'-'+floattostr(xOutputVector[i]);
    end;
    //if AddSampleForm.Layers[1].Neurons[i].Output = 1 then
      //Count:=count+1;

  // Нейрона мережа Хопфілда
    {addsampleform.NeuralNet_Hamming_Hopf1.Calc;

  for i := 0 to IconSize* IconSize-1 do
    if AddSampleForm.NeuralNet_Hamming_Hopf1.Layers[1].Neurons[i].Output = 1
then
      Count:=count+1;

  NeuroCount.Text:=inttostr(count);
 }
    closefile(n_debug)
end;

procedure TMainForm.Button8Click(Sender: TObject);
begin
Form5.Show;
end;

end.
```

Файл UPreview.pas – модуль роботи з камерами

```
unit UPreview;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, VCap,USingleFrame,UTypeConst,URazvertka, UCamFunc,URoboRootServer,
  ExtCtrls,UMoving,Picture_Analyz_Hamming,UQPixels,Picture_processing_Hamming;

type
  //захоплений кадр
  TCapturedBitmap = Vcap.TCapturedBitmap;

  TBitmap = Graphics.TBitmap;

  //клас який описує подію захоплення кадру з камери
  TBitmapGrabEvent = class
    private
      EventIndex: Integer; //індекс події (номер камери)
    public
      AnalyzFrame: boolean; //аналіз кадру
      OnFlyDebug: boolean; //дебаг на льоту
      NeedAddSample: boolean; //потрібно додати семпл
      constructor Create(Index: Cardinal); //конструктор
      destructor Destroy; override; //деструктор
      //аналіз захопленого кадру
      procedure AnalyzBitmap(Bitmap: TCapturedBitmap);
  end;

  //тип масиву оброблювачів захоплення кадру з камери
  TBitmapGrabEventArr = array of TBitmapGrabEvent;


  TPreviewWindow = class(TForm)
    Video: TImage;
    procedure VideoCaptureDeviceLost(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure ShowEx;
  end;

var
  PreviewWindow: TPreviewWindow; //форма для висновку зображення з камери
  VideoCaptureArr: TVideoCaptureArr; //масив компонентів для захоплення відео
  BitmapGrabEventArr: TBitmapGrabEventArr;  //масив оброблювачів захоплення
кадру з камери
  QP: TQuickPixels;

  UgliPolosi: TUgliRazrivov; //кути можливих напрямків руху
  Flag: boolean = false;

  implementation

uses
  Main_NeuralNet_Hamming_,UCamTimers,Urestaran;

var
  F: TextFile;
//  BitmapArr: TByteArr;

{$R *.dfm}

//конструктор
```

```
constructor TBitmapGrabEvent.Create(Index: Cardinal);
begin
  inherited Create;
  //індекс оброблювача
  EventIndex := Index;
  //аналіз кадру
  AnalyzFrame := false;
  //дебаг на льоту
  OnFlyDebug := false;
end;
//---------------------------------

//деструктор
destructor TBitmapGrabEvent.Destroy;
begin
  inherited Destroy;
end;
//---------------------------------

//аналіз захопленого бітмапа
procedure TBitmapGrabEvent.AnalyzBitmap(Bitmap: TCapturedBitmap);
var
  RazvertkaTlumitsya: TRazvertka;
  RazvertkaTlumitsyaConfig: TRazvertkaConfig;
  razriv_cnt,razriv_cnt_tlumitsya: integer; // кількість розривів
  UgliRazrivov,UgliRazrivovTlumitsya: TUgliRazrivov; //кути розривів
//  BitmapArr: TByteArr;
  y: Cardinal;
begin
  //якщо потрібно давати відео робосерверу
  if URoboRootServer.VideoInUse and (ActiveCamNum = EventIndex) then
  begin
    if not KadrSending then
    begin
      KadrFormating := true;
      Kadr.Assign(Bitmap);
      //KadrSizeWH := Bitmap.Width;
      KadrFormating := false;
    end;
  end;

  //аналіз одного кадру з виводом інформації про кластери
  if AnalyzFrame and not OnFlyDebug then
  begin
    USingleFrame.FrameForm.DebugFrame(Bitmap,true);
    AnalyzFrame := false;
  end;

  //аналіз потоку кадрів без висновку інформації про кластери
  if OnFlyDebug and not AnalyzFrame then
  begin
    USingleFrame.FrameForm.DebugFrame(Bitmap);
  end;

  //якщо потрібно взяти семпл
  if NeedAddSample then
  begin
    Picture_processing_Hamming.AddSampleForm.SampleImage.Picture.Assign(Bitmap);
    NeedAddSample := false;
    with AddSampleForm do
    begin
      SampleImage.Width := Bitmap.Width;
      SampleImage.Height := Bitmap.Height;
      {
      RadioTSample.Top := SampleImage.Height;
      RadioNotTSample.Top := SampleImage.Height;
      AddSampleButton.Top := SampleImage.Height + RadioTSample.Height;
      BrowseButton.Top := SampleImage.Height + RadioTSample.Height;
      }
```

```
      //height:=GroupBox1.Height+SampleImage.Height;
      //ObjectImage.Left:=SampleImage.Width;
      ObjectImage.Left:=0;
      GroupBox1.top:=SampleImage.Height;  //AddSampleForm.height
      //AddSampleForm.ClientWidth := SampleImage.Width+;
      if not(visible) then ShowModal;
    end;
  end;

  //якщо обрано камеру подія якого відбулося
  if (EventIndex = ActiveCamIndex) then
  begin
    PreviewWindow.ClientWidth := Bitmap.Width;
    PreviewWindow.ClientHeight := Bitmap.Height;
    PreviewWindow.Video.Picture.Bitmap.Assign(Bitmap);

   // Main_NeuralNet_Hamming_.MainForm.TLabel.Caption := 'немає!';
  end;

  //рух по полігону, обробляємо обрану камеру
  if GraphConfigExArr[EventIndex].CamFunc = CamFuncPolygonForw then
  begin
    //розгорнення
    razriv_cnt := RazvertkaArr[EventIndex].Klaster_Analiz(UgliRazrivov,Bitmap);
    //смуга
    Polosa(razriv_cnt,UgliRazrivov,UgliPolosi);
    //може бути T
    //if razriv_cnt = 2 then


    if recognition_run then
    begin
      QP.Attach(Bitmap);

      SetLength(BitmapArr,QP.Height);
      for y := 0 to High(BitmapArr) do
        SetLength(BitmapArr[y],QP.Width);
      ConvertBitmapToMonoChrome(QP,BitmapArr);
      //segment(BitmapArr);

      //TSampleRule(BitmapArr);

        RobotRecognition.BitmapReady:=true;
    end;
  end;

  //тлумиться
  if Restaran <> nil then
  if Restaran.tlumitsya = 1 then
  begin
    with RazvertkaTlumitsyaConfig do
    begin
      a := RazvertkaConfigArr[EventIndex].a;
      a_corr := RazvertkaConfigArr[EventIndex].a_corr;
      b := 10;
      y_offset := RazvertkaConfigArr[EventIndex].y_offset;
      klaster_size := RazvertkaConfigArr[EventIndex].klaster_size;
      porog := RazvertkaConfigArr[EventIndex].porog;
      step := RazvertkaConfigArr[EventIndex].step;
      fi_step := RazvertkaConfigArr[EventIndex].fi_step;
    end;

    RazvertkaTlumitsya := TRazvertka.Creat(RazvertkaTlumitsyaConfig);
    razriv_cnt_tlumitsya :=
RazvertkaTlumitsya.Klaster_Analiz(UgliRazrivovTlumitsya,Bitmap);

    if razriv_cnt_tlumitsya = 2 then
      Restaran.ugol_tlumitsya := (UgliRazrivov[0] + UgliRazrivov[1]) / 2;
```

```
    RazvertkaTlumitsya.Destroy;
  end;

  //простий рух по смузі
  if MoveForLineFlag then
  begin
    razriv_cnt := RazvertkaArr[EventIndex].Klaster_Analiz(UgliRazrivov,Bitmap);
    Polosa(razriv_cnt,UgliRazrivov,UgliPolosi);
    MoveForLine(UgliPolosi);
  end;
end;
//-----------------------------------

//Показати форму
procedure TPreviewWindow.ShowEx;
begin
  if GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Width <= 640 then
    PreviewWindow.ClientWidth :=
GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Width
  else
    PreviewWindow.ClientWidth := 640;
  if GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Height <= 480 then
    PreviewWindow.ClientHeight :=
GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Height
  else
    PreviewWindow.ClientHeight := 480;

  Left := Screen.Width - Width;
  Top := 0;

  Show;
end;
//-----------------------------------

//Подія - зв'язок з камерою загублена
procedure TPreviewWindow.VideoCaptureDeviceLost(Sender: TObject);
begin
  ShowMessage('Зв'язок з відео пристроєм загублена!');
end;

initialization
  //QP2 := TQuickPixels.Create;
  QP := TQuickPixels.Create;
  AssignFile(F,'Preview.txt');
  Rewrite(F);

finalization
  QP.Destroy;
  CloseFile(F);

end.
```

Файл UCamTimers.pas – модуль встановлення таймерів камер

```
unit UCamTimers;

interface

uses
  Windows,URazvertka, UPreview, UTypeConst, SysUtils;

type
   //тип процедури спрацьовування таймера як метод класу
   //TTimeProc = procedure(uID, uMsg: UINT; dwUser, dw1, dw2: DWORD) of object;

  //клас таймера камери
  TCamTimer = class
    private
      uidtimer: UINT; //ідентифікатор таймера
      TimerDelay: Cardinal; //період спрацьовування таймеа (мс)
      TimerIndex: Cardinal; //індекс таймера
    public
      Active: boolean; // чиактивний таймер
      Constructor Create(Index: Cardinal);
      Destructor Destroy(); override;
      procedure StartTimer(Delay: Cardinal);
      procedure StopTimer();
  end;

  //масив таймерів камер
  TCamTimerArr = array of TCamTimer;

var
  CamTimerArr: TCamTimerArr;//масив таймерів камер

//оброблювач таймерів
procedure TimeProc(uID, uMsg: UINT; dwUser, dw1, dw2: DWORD); stdcall;

implementation

Constructor TCamTimer.Create(Index: Cardinal);
begin
  inherited Create;
  TimerIndex := Index;
  Active := false;
end;
Destructor TCamTimer.Destroy();
begin
  StopTimer();
  inherited Destroy();
end;
procedure TCamTimer.StartTimer(Delay: Cardinal);
begin
  TimerDelay := Delay;
  uidtimer := timeSetEvent(TimerDelay,1,@TimeProc,TimerIndex,1);
  Active := true;
end;
procedure TCamTimer.StopTimer();
begin
  timeKillEvent(uidtimer);
  Active := false;
end;
//оброблювач таймерів камер
procedure TimeProc(uID, uMsg: UINT; dwUser, dw1, dw2: DWORD); stdcall;
begin
  VideoCaptureArr[dwuser].CaptureFrame;
end;

end.
```

Файл UGraphConfig.pas – модуль налаштування камер

```
unit UGraphConfig;

{Модуль настроювання камери}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, VCap, Buttons, ExtCtrls, ComCtrls, UTypeConst;


const
  MMInit_WrongDeviceNum = -1; //неправильний номер пристрою
  MMInit_WrongCompNum = -1; //неправильний номер компресора


type
  TGraphConfigForm = class(TForm)
    Label3: TLabel;
    VideoCompBox: TListBox;
    OKBitBtn: TBitBtn;
    GraphConfigGroupBox: TGroupBox;
    WantPreviewCheckBox: TCheckBox;
    PixelFormatComboBox: TComboBox;
    Label4: TLabel;
    CaptureFileNameEdit: TLabeledEdit;
    WantCaptureCheckBox: TCheckBox;
    CamFuncComboBox: TComboBox;
    Label1: TLabel;
    WantBitmapsCheckBox: TCheckBox;
    TimerDelayEdit: TLabeledEdit;
    procedure OKBitBtnClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure WantBitmapsCheckBoxClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  GraphConfigForm: TGraphConfigForm; //форма

  //індекс елемента масиву настроювань
  GraphConfigExIndex: Cardinal;

implementation

uses
  Main_NeuralNet_Hamming_;

{$R *.dfm}

//---------------------------------i

{при натисканні кнопки OK відбувається зчитування настроювань,
заданих користувачем, і присвоєння їх  переданої змінної}
procedure TGraphConfigForm.OKBitBtnClick(Sender: TObject);
var
  i: integer; //лічильник
  SelVCompNum: integer; //номер обраного відео компресора
begin
  //заповнюємо змінну стану камери
  with GraphConfigExArr[GraphConfigExIndex].GraphConfig do
  begin
```

```
    //заповнюємо поле "відео компресор" ім'ям обраного відео компресора
    SelVCompNum := MMInit_WrongCompNum;
    for i := 0 to VideoCompBox.Count - 1 do
      if VideoCompBox.Selected[i] then
        SelVCompNum := i;

    //якщо не один компресор не обраний
    if SelVCompNum = MMInit_WrongCompNum then
      VComp := ''
    else
      VComp := VideoCompBox.Items[SelVCompNum];

    //потрібно записувати відео?
    if WantCaptureCheckBox.Checked then
      WantCapture := true
    else
      WantCapture := false;

    //за замовчуванням не потрібно зображення
    WantPreview := false;
    WantBitmaps := false;

    //потрібно показувати зображення
    if WantPreviewCheckBox.Checked then
    begin
      GraphConfigExArr[GraphConfigExIndex].ShowPreview := 1;
      WantPreview := true;
    end
    else
    begin
      GraphConfigExArr[GraphConfigExIndex].ShowPreview := 0;
    end;

    //потрібні кадри
    if WantBitmapsCheckBox.Checked then
    begin
     if WantPreview = false then
       WantPreview := true;
     WantBitmaps := true;
    end
    else
      WantBitmaps := false;

    //ім'я файлу в який записується відео
    CaptureFileName := CaptureFileNameEdit.Text;

    // функція камери
    GraphConfigExArr[GraphConfigExIndex].CamFunc := CamFuncComboBox.ItemIndex;

    //частота таймера
    GraphConfigExArr[GraphConfigExIndex].TimerDelay :=
StrToInt(TimerDelayEdit.Text);
  end;

  //вибираємо формат подання пікселя
  with PixelFormatComboBox do
  begin
    if Text = 'Визначається пристроєм' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pfDevice
    else if Text = '1 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf1bit
    else if Text = '4 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf4bit
    else if Text = '8 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf8bit
    else if Text = '15 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf15bit
    else if Text = '16 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf16bit
```

```
    else if Text = '24 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf24bit
    else if Text = '32 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf32bit
    else if Text = 'Налаштовуваний' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pfCustom
    else GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat :=
pfDevice;
  end;
end;
//--------------------------------i

{Подія - перед появою форми
виводимо списки доступних
компресорів для стиску відео й відновлюємо галочки у відповідності
с переданими відео настроюваннями}
procedure TGraphConfigForm.FormShow(Sender: TObject);
var
  i: integer; //лічильник
  VideoCompList: TStringList; //аркуш доступних відео компресорів
begin
  Caption := 'Настроювання відео для ' +
GraphConfigExArr[GraphConfigExIndex].GraphConfig.VCapSource;

  //одержуємо список відео кодеків
  VideoCompList := GetVideoCompressorsList(true);

  //виводимо в компоненти отриману інформацію
  VideoCompBox.Items.Assign(VideoCompList);

  //жодна рядок не виділений
  VideoCompBox.ItemIndex := -1;

  //відновлюємо номер відео компресора
  for i := 0 to VideoCompBox.Count - 1 do
  begin
    if VideoCompBox.Items[i] =
GraphConfigExArr[GraphConfigExIndex].GraphConfig.VComp then
      VideoCompBox.ItemIndex := i;
  end;

  //відновлюємо галочки "Потрібно зображення"
  if GraphConfigExArr[GraphConfigExIndex].ShowPreview = 1 then
    WantPreviewCheckBox.Checked := true;
  if GraphConfigExArr[GraphConfigExIndex].ShowPreview = 0 then
    WantPreviewCheckBox.Checked := false;

  //відновлюємо галочку "Потрібно записувати відео"
  if GraphConfigExArr[GraphConfigExIndex].GraphConfig.WantCapture then
    WantCaptureCheckBox.Checked := true
  else
    WantCaptureCheckBox.Checked := false;

  //відновлюємо поле "потрібні кадри"
  if GraphConfigExArr[GraphConfigExIndex].GraphConfig.WantBitmaps then
    WantBitmapsCheckBox.Checked := true
  else
    WantBitmapsCheckBox.Checked := false;

  //відновлюємо формат пікселя
  case GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat of
    pfDevice : PixelFormatComboBox.Text := 'Визначається пристроєм';
    pf1bit : PixelFormatComboBox.Text := '1 біт';
    pf4bit : PixelFormatComboBox.Text := '4 біт';
    pf8bit : PixelFormatComboBox.Text := '8 біт';
    pf15bit : PixelFormatComboBox.Text := '15 біт';
    pf16bit : PixelFormatComboBox.Text := '16 біт';
    pf24bit : PixelFormatComboBox.Text := '24 біт';
    pf32bit : PixelFormatComboBox.Text := '32 біт';
```

```
    pfCustom : PixelFormatComboBox.Text := 'Налаштовуваний';
  end;

  //відновлюємо ім'я файлу для запису відео  CaptureFileNameEdit.Text :=
GraphConfigExArr[GraphConfigExIndex].GraphConfig.CaptureFileName;


  // фукції камери
  CamFuncComboBox.ItemIndex := GraphConfigExArr[GraphConfigExIndex].CamFunc;

  //частота таймера
  TimerDelayEdit.Text :=
IntToStr(GraphConfigExArr[GraphConfigExIndex].TimerDelay);

end;

//подія - натискання на "потрібні кадри"
procedure TGraphConfigForm.WantBitmapsCheckBoxClick(Sender: TObject);
begin
  if WantBitmapsCheckBox.Checked then
  begin
    WantPreviewCheckBox.Enabled := true;
  end
  else
  begin
    WantPreviewCheckBox.Checked := false;
    WantPreviewCheckBox.Enabled := false;
  end;
end;

end.
```

Файл About.pas - довідка

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TAboutForm = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutForm: TAboutForm;

implementation

{$R *.dfm}

procedure TAboutForm.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Дослідження та програмна реалізація системи розпізнавання
графічних образів за допомогою нейронної мережі Хеммінга');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Дреєв О.М.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Федоров Богдан Сергійович');
Memo1.Lines.Add('                  гр. КН-22М-1');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;

procedure TAboutForm.Button1Click(Sender: TObject);
begin
AboutForm.Close;
end;

end.
```