Вінницький національний технічний університет
Факультет інформаційних електронних систем
Кафедра біомедичної інженерії та оптико-електронних систем

Бакалаврська кваліфікаційна робота на тему:

ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНА СИСТЕМА ДЛЯ ОБРОБЛЕННЯ ПРОФІЛЮ ЛАЗЕРНОГО ПРОМЕНЯ З КЛАСИФІКАЦІЄЮ ПЛЯМОПОДІБНИХ ЗОБРАЖЕНЬ

Виконала: студентка 4 курсу, групи КОІС-21б спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка» за освітньою програмою «Комп'ютеризовані оприко-інформаційні системи»
Ярова О.А.
Керівник: д.т.н., професор каф. БМІОЕС
Заболотна Н.І.
«10» червня 2025 р.
Рецензент: к.т.н., доцент каф. АІІТ
Барабан М.В.
«16» червня 2025 р.

Допущено до захисту
Зав. кафедри БМІОЕС
«17» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інформаційних електронних систем
Кафедра біомедичної інженерії та оптико-електронних систем
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 15 «Автоматизація та приладобудування»
Спеціальність – 152 «Метрологія та інформаційно-вимірювальна техніка»
Освітньо-професійна програма – «Комп'ютеризовані оптико-інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри БМІОЕС
_____ Коваль Л.Г.
"21" березня 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
Яровій Олені Андріївні

1. Тема роботи: Інформаційно-вимірювальна система для оброблення профілю лазерного променя з класифікацією плямоподібних зображень.
Керівник роботи: Заболотна Наталія Іванівна, д.т.н., проф. кафедри БМІОЕС.
затверджені наказом вищого навчального закладу від «20» березня 2025 року №97

2. Термін подання студентом роботи 15.06.2025.

3. Вихідні дані до роботи:
3.1 Функціональне призначення системи – класифікація плямоподібних зображень в процесах оброблення профілю лазерного променя для застосування в інформаційно-вимірювальній системі.
3.2 Вимоги до зображень – мінімальна розмірність 128x128 пікселів.
3.3 Мова програмування – об'єктно-орієнтована.
4. Зміст текстової частини (перелік питань які потрібно розробити):
Вступ. 1 Аналіз предметної області та обґрунтування доцільності розробки інформаційно-вимірювальної системи для оброблення профілю лазерного променя. 2 Аналіз методів та комп'ютерне моделювання оброблення плямоподібних зображень. 3 Програмно-апаратна реалізація інформаційно-вимірювальної системи для оброблення профілю лазерного променя. Висновки. Список використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): 1. Схема вимірювання профілю лазерного променя для напівпровідникової лазерної системи 2. Узагальнена схема процесу нейромережевого оброблення зображень лазерного променя. 3. Структурна організація обраної для моделювання НМ 4. Апаратний макет інформаційно-вимірювальної системи.5 Екранна форма головного програмного модуля системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Заболотна Н.І., д.т.н., проф. каф. БМІОЕС	20.03.2025	10.06.2025

7. Дата видачі завдання «20» березня 2025 р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Приміт
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	03.03.2025	20.03.2025	викон.
2	Аналіз літературних джерел. Попередня розробка основних розділів.	20.03.2025	29.03.2025	викон.
3	Аналіз предметної області оброблення профілю лазерного променя.	30.03.2025	05.05.2025	викон.
4	Аналіз методів та комп'ютерне моделювання оброблення плямоподібних зображень.	06.05.2025	17.05.2025	викон.
5	Програмно-апаратна реалізація інформаційно-вимірювальної системи для оброблення профілю лазерного променя	18.05.2025	31.05.2025	викон.
6	Оформлення пояснювальної записки та ілюстративної частини	01.06.2025	06.06.2025	викон.
7	Нормоконтроль	06.06.2025	09.06.2025	викон.
8	Попередній захист БКР, доопрацювання, рецензування БКР	10.06.2025	16.06.2025	викон.
9	Захист БКР ЕК	19.06.2025	20.06.2025	викон.

Студентка _____ Ярова О.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Заболотна Н.І

АНОТАЦІЯ

Ярова О.А. Інформаційно-вимірювальна система для оброблення профілю лазерного променя з класифікацією плямоподібних зображень. Бакалаврська кваліфікаційна робота зі спеціальності 152 – Метрологія та інформаційно-вимірювальна техніка, освітня прогрма – Комп'ютеризовані оптико-інформаційні системи. Вінниця: ВНТУ, 2025.

Бакалаврська кваліфікаційна робота складається з 86 сторінок формату А4, на яких є 30 рисунків, 1 таблиці, список використаних джерел містить 43 найменування.

Метою роботи є розширення функціональних можливостей інформаційно-вимірювальної системи для оброблення профілю лазерного променя шляхом застосування нейромережевої класифікації плямоподібних зображень.

В роботі проаналізовано предметну область профілювання лазерного променя; наведено обґрунтування доцільності розробки інформаційно-вимірювальної системи для оброблення профілю лазерного променя з нейромережевою класифікацією плямоподібних зображень; проаналізовано методи оброблення та розпізнавання зображень; обгрунтовано вибір нейромержевого підходу та промодельовано нейромережеве оброблення профілю лазерного променя з класифікацією плямоподібних зображень; а також здійснено програмно-апаратну реалізацію інформаційно-вимірювальної системи для оброблення профілю лазерного променя.

Ключові слова: інформаційно-вимірювальна система, лазер, профіль лазерного променя, класифікація зображень, нейронні мережі, оброблення зображень.

ABSTRACT

Yarova O.A. Information-measuring system for processing the laser beam profile with spot image classification. Bachelor's qualification thesis in the specialty 152 – Metrology and information and measuring equipment, educational program – Computerized optical information systems. Vinnytsia: VNTU. 2025.

The bachelor's thesis consists of 86 pages of A4 format, which includes 30 figures, 1 table, and a list of references containing 43 items.

The purpose of the research is to expand the functionality of the information-measuring system for processing the laser beam profile by applying neural network classification of spot images.

The thesis analyzes the subject area of laser beam profiling; provides a justification for the feasibility of developing an information-measuring system for processing the laser beam profile with neural network classification of spot images; analyzes methods of image processing and recognition; justifies the choice of the neural network approach and performs computer simulation of neural network processing of the laser beam profile with spot image classification; and also provides software and hardware implementation of the information-measuring system.

Keywords: information-measuring system, laser, laser beam profile, image classification, neural networks, image processing.

ЗМІСТ

ВСТУП 4
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНОЇ СИСТЕМИ ДЛЯ ОБРОБЛЕННЯ ПРОФІЛЮ ЛАЗЕРНОГО ПРОМЕНЯ 7
1.1 Аналіз предметної області оброблення профілю лазерного променя 7
1.2 Прикладне значення та застосування інформаційно-вимірювальних систем для оброблення профілю лазерного променя 11
1.3 Аналіз систем-аналогів оброблення профілю лазерного променя 12
1.4 Постановка задачі оброблення профілю лазерного променя 15
1.5 Висновки до розділу 1 17
2 АНАЛІЗ МЕТОДІВ ТА КОМП’ЮТЕРНЕ МОДЕЛЮВАННЯ ОБРОБЛЕННЯ ПЛЯМОПОДІБНИХ ЗОБРАЖЕНЬ 18
2.1 Класифікаційний аналіз методів оброблення та розпізнавання зображень 18
2.2 Нейромережевий підхід для розпізнавання та класифікації зображень 21
2.3 Комп’ютерне моделювання нейромережевого оброблення профілю лазерного променя з класифікацією плямоподібних зображень 29
2.4 Висновки до розділу 2 34
3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНОЇ СИСТЕМИ ДЛЯ ОБРОБЛЕННЯ ПРОФІЛЮ ЛАЗЕРНОГО ПРОМЕНЯ 35
3.1 Реалізація апаратної складової інформаційно-вимірювальної системи для оброблення профілю лазерного променя 35
3.2 Узагальнена структурно-функціональна модель інформаційно-вимірювальної системи для оброблення профілю лазерного променя 42

3.3 Програмна реалізація інформаційно-вимірювальної системи для оброблення профілю лазерного променя з класифікацією плямоподібних зображень 51
3.3.1 Обґрунтування вибору мови програмування 51
3.3.2 Розробка алгоритму головного програмного модуля 52
3.3.3 Тестовий приклад роботи та аналіз результатів 55
3.4 Висновки до розділу 3 58
ВИСНОВКИ 60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 62
Додаток А (обов’язковий) Протокол перевірки кваліфікаційної роботи 67
Додаток Б (обов’язковий) Ілюстративна частина 68
Додаток В (обов’язковий) Лістинг програми 74

ВСТУП

Прискорений розвиток цифрових технологій та оптоелектроніки вплинув на масове поширення застосування лазерів у різноманітних сферах життєдіяльності людини. Застосування лазерних технологій потребує постійного контролю за технічним станом лазерної системи. Для цього необхідно мати точні динамічні дані про характеристики лазерного випромінювання, які можна отримати з його профілю в поточний момент часу, а також по можливості передбачити зміну характеристик променя на певний відрізок часу, достатній для вчасної безпечної зупинки роботи лазерної системи чи автоматичного калібрування лазера для продовження його роботи [1-3].

Актуальною задача профілювання лазерного променя є ще й тому, що реальні умови використання лазера не є ідеальними чи лабораторними. Через це поведінка лазерного променя в деяких ситуаціях стає непередбачуваною, що може призвести до некоректної роботи лазерної системи, та її функціональної невідповідності поставленій задачі [2, 4, 5].

Поширення лазерного випромінювання в атмосфері супроводжується дуже великим набором явищ лінійної і нелінійної взаємодії. За якісними ознаками зазначені явища можна розділити на наступні основні групи: рефракція променів лазерного пучка; поглинання енергії лазерного пучка атмосферними газами; розсіювання енергії лазерного пучка частками аерозолів на флуктуаціях щільності повітря та ін.; флуктуації параметрів лазерних пучків, обумовлені атмосферною турбулентністю. Кожна з перерахованих груп явищ взаємодії лазерного випромінювання з атмосферою може виявлятися в областях як лінійної, так і нелінійної оптики. У той же час кожна з цих груп має чіткі специфічні особливості, що повинні враховуватися при відповідних теоретичних і експериментальних дослідженнях [4, 6-8].

Таким чином виникає задача профілювання лазерного променя, ідентифікації та розпізнавання окремих характеристик профілю лазерного пучка. Ще раз відзначимо, що оскільки поведінка лазера стає нелінійним процесом, що досить важко аналізувати, то традиційні підходи вже не є достатньо ефективними на сучасному етапі розвитку технологій. Через це варто звернути увагу на технології штучного інтелекту, а саме на вирішення задачі розпізнавання та ідентифікації зображань профілю лазерного променя на основі штучних нейронних мереж [1, 9-12].

Об’єкт дослідження – процес класифікації плямоподібних зображень при обробленні профілю лазерного променя в інформаційно-вимірювальній системі.

Предмет дослідження – моделі нейромережевої класифікації плямоподібних зображень та програмно-апаратні засоби оброблення профілю лазерного променя.

Метою роботи є розширення функціональних можливостей інформаційно-вимірювальної системи для оброблення профілю лазерного променя шляхом застосування нейромережевої класифікації плямоподібних зображень.

Задачі дослідження:

1. Здійснити аналіз предметної області оброблення профілю лазерного променя.

2. Здійснити аналіз методів та комп’ютерне моделювання оброблення плямоподібних зображень.

3. Розробити узагальнену структурно-функціональну модель інформаційно-вимірювальної системи для оброблення профілю лазерного променя.

4. Розробити програмно-апаратну реалізацію інформаційно-вимірювальної системи для оброблення профілю лазерного променя та нейромережевої класифікації плямоподібних зображень.

Апробація та публікації.

Основні результати дослідження пройшли апробацію на Міжнародній науково-практичній конференції "Молодь в науці: дослідження, проблеми, перспективи (МН-2025)".

За матеріалами бакалаврської кваліфікаційної роботи опубліковано тези доповіді [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНОЇ СИСТЕМИ ДЛЯ ОБРОБЛЕННЯ ПРОФІЛЮ ЛАЗЕРНОГО ПРОМЕНЯ

1.1 Аналіз предметної області оброблення профілю лазерного променя

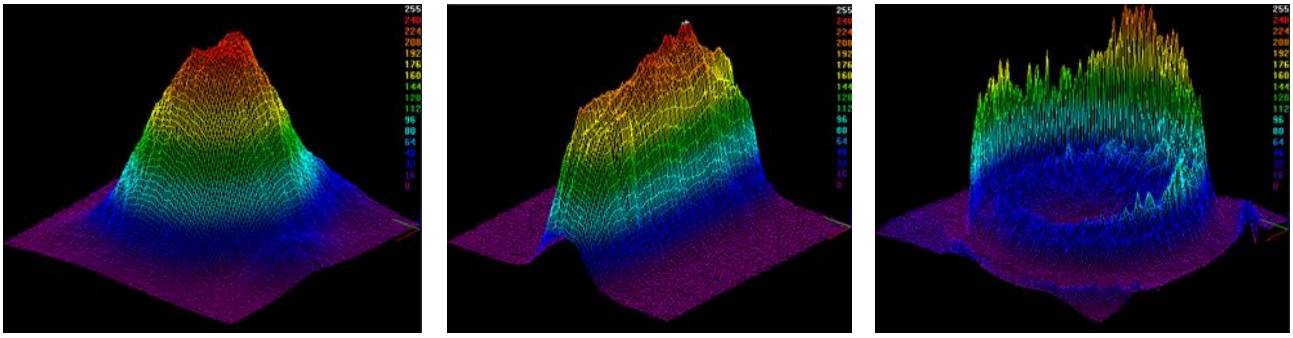
Відомо, що поширення лазерного випромінювання в атмосфері Землі супроводжується набором явищ як лінійної, так і нелінійної взаємодії. При цьому жодне з цих явищ не виявляється окремо. Адже атмосфера є газоподібною оболонкою, що базується головним чином на температурних коливаннях і оточує Землю та простирається на кілька сотень кілометрів над поверхнею. Більш ніж 98 % атмосфери за своїм обсягом складається з елементів азоту і кисню. Головні складові атмосфери – водяна пара, вуглекислий газ, окис азоту, чадний газ і озон. Атмосфера Землі – абсорбуюче середовище. Поглинання відбувається, коли фотон радіації поглиається газоподібною молекулою атмосфери, що перетворює фотон у кінетичну енергію молекули. Отже, поглинання – механізм, яким нагрівається атмосфера. Розсіювання електромагнітних хвиль у видимих і інфрачервоних довжинах хвилі відбувається, коли випромінювання поширюється через певні повітряні молекули і частки [13, 14].

На даний час в галузях лазерної обробки матеріалів, лазерній локації, оптичному зв'язку, поліграфії й інших областях техніки відчувається гостра необхідність більш широкого впровадження оптико-електронних інтелектуальних систем з автоматичним коректуванням похибок формованого світлового випромінювання. Причинами цих похибок можуть бути: дестабілізуючий вплив механічних чи кліматичних факторів, нестабільність характеристик джерела випромінювання, перешкоди в оптичному тракті і т.п. Забезпечення прийнятної якості корекції потребує безупинного динамічного контролю характеристик світлового випромінювання, наприклад його профілю, просторового розподілу його інтенсивності, у тому числі оцінки відхилення

зазначеного розподілу від вихідного чи еталонного розподілу [1, 2, 6, 8, 15].

Значення вимірювання профілю променя полягає в тому, що густина енергії, концентрація і колімація світла є його взаємопов'язаними складовими характеристиками. Розповсюдження лазерного променя у просторі значно змінює його профіль [2, 3, 6, 8].

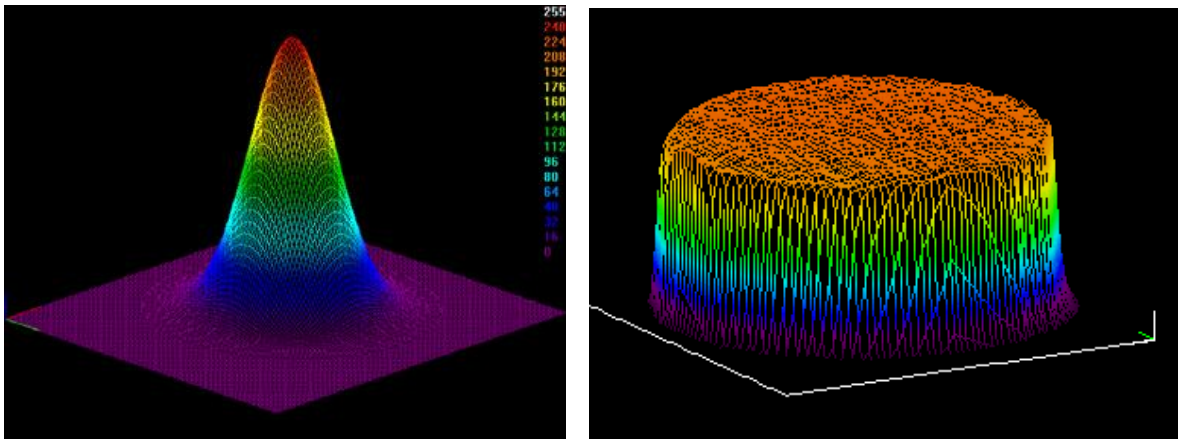
Наприклад, на рис. 1.1 наведено звичайні профілі лазерного променя, які ілюструють існуючу їх різновидність. Так як наведені різновидності присутні в профілях лазерного променя, важливого значення набуває проблема вимірювання профілю лазерного променя для різноманітних практично-прикладних застосувань, особливо якщо густина енергії пов'язана із продуктивністю лазера [3, 16].



а) б) в)

Рисунок 1.1 – Різновидності лазерних профілів:
а) HeNe; б) ексімерний лазер; в) азотний кільцевий лазер

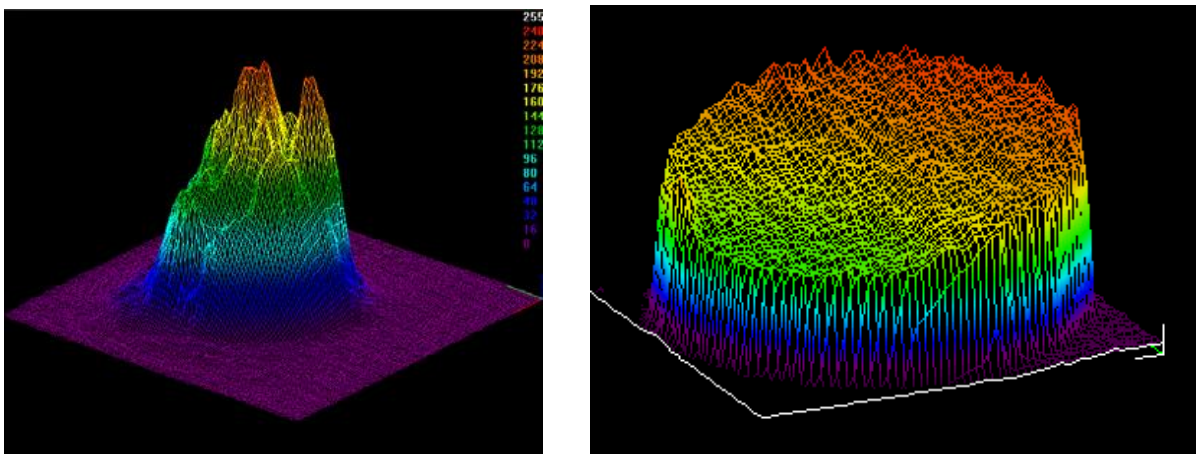
Прикладом двох різних типів профілів ідеальних лазерних променів для різних практичних цілей є промінь Гауса та промінь з плоскою вершиною. Промінь Гауса передбачає найвищу сконцентрованість сфокусованого світла, а промені з плоскою вершиною передбачають незмінний розподіл енергії у визначеній ділянці. Приклади профілів цих двох променів зображено на рис. 1.2 [3, 16-19].



а) б)

Рисунок 1.2 – Типи профілів ідеальних лазерних променів:
а) ідеальний промінь Гауса для найбільшої концентрації енергії;
б) ідеальний промінь з плоскою вершиною для стабільного лазерного освітлення

Проте, на практиці лазери рідко відображають найбільш незмінний інфрачервоний профіль. Деколи лазерні промені Гауса нерівномірно високоструктуровані, а лазерні промені з плоскою вершиною – нерівномірні по всій вершині або нахилені відносно енергії з однієї сторони до іншої. Рис. 1.3 ілюструє деякі реальні приклади деформованих профілів променя. Наприклад, високоструктурований промінь не фокусував би енергію, як ідеальний промінь Гауса (рис. 1.3 а). Нахилений промінь з плоскою вершиною не давав би рівномірного освітлення як ідеальний, і міг би спричиняти значні деформації в процесі його застосування (рис. 1.3 б) [3, 16].



а) б)

Рисунок 1.3 – Приклади деформованих профілів
лазерного променя Гауса (а) та променя з плоскою вершиною (б)

Значення деформованих профілів лазерного променя змінюються в залежності від практичного застосування. В наукових застосуваннях нелінійні процеси прямо пропорційні до квадратів або кубів значень густини випромінювання. Такі „не Гаусові” профілі можуть мати пікову енергію меншу за 50% від тієї, що промінь Гауса мав би від загальної потужності або енергії при тих самих умовах. Тому нелінійний процес може призводити до 12 % або 25 % втрат від очікуваних значень пікової енергії. Це призводить в свою чергу до 300% або 700% помилки на експерименті, який має бути точним в межах ±5% [2,3,6,8].

Так як лазерні технології є одними з найперспективніших та застосовуються в численних галузях людської діяльності, відповідні пристрої, що використовують лазери, потребують високого ступеня автоматичного контролю лазерним променем для коректного виконання поставлених завдань. Тому, актуальним є дослідження процесів формування профілю променя, який може бути зручно представлений у вигляді плямового зображення із спектральним розподілом кольорів за інтенсивністю випромінювання, адже за результатами його аналізу можна опосередковано зробити висновок про технічний стан лазера. Для цього необхідно відповідним чином обробити, класифікувати та розпізнати плямові зображення.

Велика увага приділяється даному науково-практичному напряму і в країнах ЄС та США, підтвердженням чого є аналогічні наукові дослідження та програмно-апаратні розробки таких зарубіжних компаній, як Ophir Optronics Solutions Ltd, Coherent Auburn Group, Axiom Optics, тощо. В Україні цими питаннями займаються в більшості науково-навчальні установи, зокрема, такі як Інститут фізики напівпровідників ім. В.Є. Лашкарьова НАНУ тощо [16, 20-26].

1.2 Прикладне значення та застосування інформаційно-вимірювальних систем для оброблення профілю лазерного променя

Використовуючи класифікацію та розпізнавання плямоподібних зображень профілю лазерного променя можна буде відслідковувати та аналізувати поведінку лазерного променя при його практичному застосуванні, тобто коли лазер буде розповсюджуватись не в ідеальних або ж лабораторних умовах.

Пристрої, які використовують лазери, такі як принтери, волоконна оптика, пристрої зв`язку та ін. потребують високого ступеня контролю лазерним променем, щоб коректно виконувати задані користувачем завдання. Рівномірність, направленість і стабільність, а також картина поля моди лазерного променя звичайного лазерного діоду, що застосовується в пристроях, може різко спотворюватись за рахунок розрегулювання та нецентрованості колімуючої оптики установки, що спричиняє до неочікуваного зниження продуктивності пристрою [2, 6, 8, 9, 15].

Актуальність даних досліджень також підтверджується надзвичайно активним застосуванням лазерів в медицині [27-29]. Одним з них є фоторефракційна кератотомія (ФРК), в якій промінь з плоскою вершиною застосовується для того, щоб виконувати корекцію зору. Якщо гомогенізатор, який формує плоску вершину профілю лазерного променя, знаходиться за межею регулювання та існує 50% нахил на плоскій вершині, то корекція гостроти зору

ока може становити 4 діоптрії на одній стороні райдужної оболонки, з лише 2 діоптріями на протилежній стороні. Цей факт необхідно враховувати для ФРК операцій, які можуть спричиняти до того, що пацієнт залишиться з невідповідною корекцією зору після операції [30].

В промислових застосуваннях лазерів найбільш потужні лазери Nd:YAG та CO_2 лазери застосовуються багаторазово. Промислові процеси різання та зварювання прямо пов'язані із вимірюванням характеристик профілю лазерного променя. Наприклад, Nd:YAG лазер з подвійним піком може спричинити одну вирізку в напрямку X та різні вирізки в напрямку Y. Також, промінь зі „збідненим профілем" може спричиняти до свердління отворів невідповідного різного діаметру [2, 6, 8, 9].

1.3 Аналіз систем-аналогів оброблення профілю лазерного променя

Найбільш подібним аналогом до даної інформаційно-вимірювальної системи для оброблення профілю лазерного променя є система Laser Beam Application (рис. 1.4 та рис. 1.5) [3, 16].

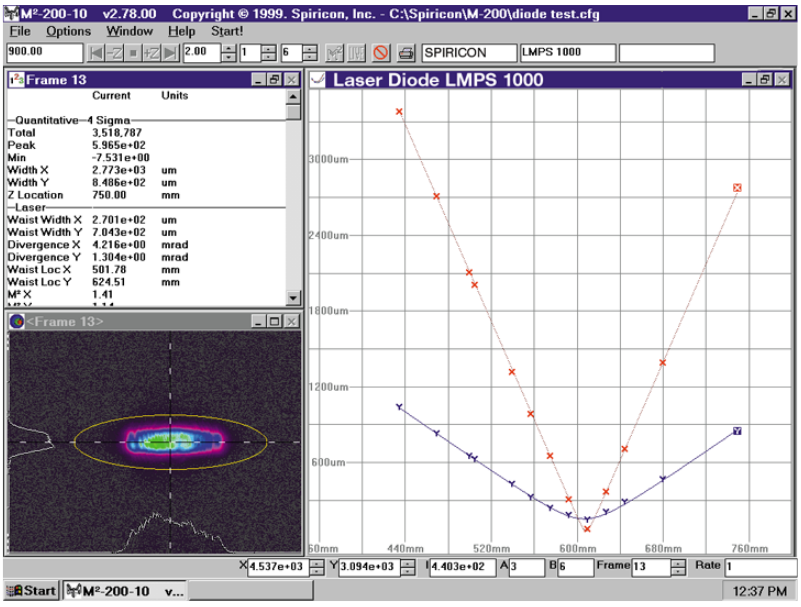


Рисунок 1.4 – Загальний вигляд програми-аналогу

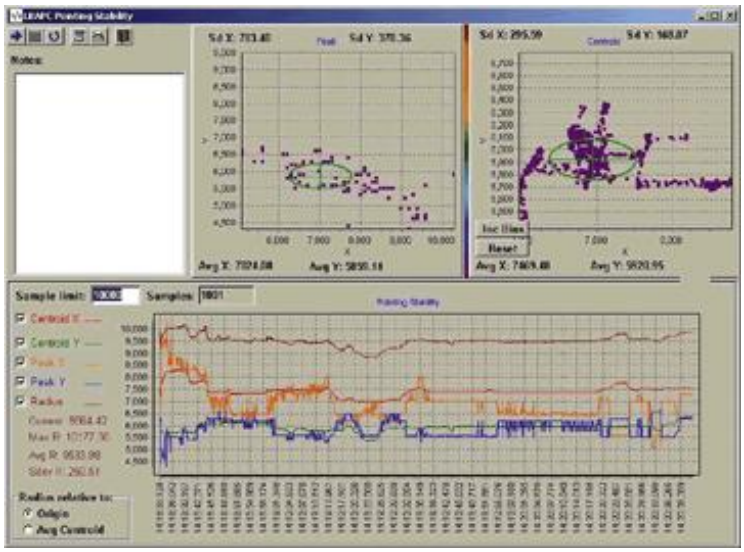


Рисунок 1.5 – Додаткові опції для профілювання лазерного променя програми-аналогу

Розробкою програмного забезпечення для розпізнавання зображень в межах задачі профілювання лазерних променів, яка на тепер є актуальною та економічно доцільною, займаються такі компанії як [20-26]:

- MS MacroSystems, Netherlands;
- ScienceGL, Cambridge, USA;
- Ophir-Spiricon LLC, North Logan, USA;
- Axiom Optics, Cambridge, USA;
- LightMachinery, Ottawa, Canada, тощо.

Проблемою аналізу якості лазерного променя також займаються вже досить давно, тому існує велика кількість підходів до оцінки лазерного променя. Найбільш розповсюдженими є такі технології [16, 20-26]:

– Ophir-Spiricon – система засобів вимірювання і аналізу профілю лазерного випромінювання. Надає можливість вимірювання параметрів лазерного випромінювання в діапазонах від дальньої ультрафіолетової до далекої інфрачервоної (ТГц) області та від нановат до кіловат.

– BeamView – система діагностики лазерного променя, заснована на використанні ПЗС (CCD) камери для отримання зображення. Крім цифрової або аналогової камери включає PCI-карту і високошвидкісний фреймгреббер (може підтримувати роботу з чотирма камерами одночасно).

– LaserCAM-HR – система вимірювання профілю лазерного променя з CCD-камерою. Володіє підвищеним розміром матриці (1280x1024 пікселів) і високим порогом насичення (16 мкВт/160 нДж). Дана камера не вимагає комп'ютерної плати і підключається безпосередньо через роз'єм USB.

– BeamMaster – вимірює профіль і ширину променя, еліптичність і позицію центроїда, потужність випромінювання. Є кілька режимів візуалізації, можливість запису у файл, інтерфейс RS-232 для зовнішнього управління. BeamMaster використовує для отримання інформації про профіль променя поєднання скануючої шторки, що перетинає промінь, з фотодіодом. За один скан, шторка перетинає промінь сім разів в різних напрямках, забезпечуючи високу точність і достовірність реконструкції профілю променя. Подібна методика допомагає отримати дуже високе розширення (до 0.1 мкм) і вимірювати промені до 3 мкм в діаметрі. Діапазон спектральної чутливості визначається діапазоном фотодіода, і може становити 190-1100 нм (кремнієвий фотодіод) або 800-1800 нм (арсенід галію).

Розглянуті системи-аналоги мають досить потужні технічні та функціональні можливості для профілювання лазерних променів, проте в ході здійсненого аналізу відзначено, що у них в недостатній мірі розглянуто питання впровадження методів і засобів штучного інтелекту у процес профілювання лазерних променів, що є актуальною задачею на даному етапі розвитку науки і техніки.

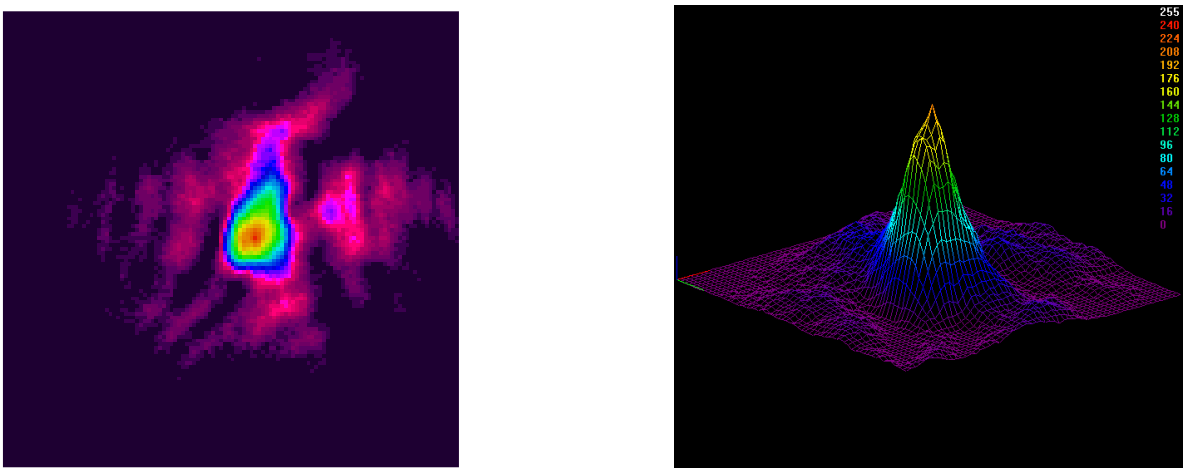
1.4 Постановка задачі оброблення профілю лазерного променя

Існує велика кількість груп методів вимірювання профілю лазерного променя, такі як неелектричні методи, електронні методи, методи на основі механічних скануючих пристроїв, на основі спеціалізованих камер (в тому числі CCD). Найбільшого застосування набули методи вимірювання профілю лазерного променя на основі CCD-камер [2, 3, 6, 8, 9, 15, 16].

Обладнання систем профілювання лазерного променя на основі камери містить сучасний комп'ютер, карту механізму захоплення кадру, щоб перетворити сигнал у цифрову форму, а також програмне забезпечення, щоб управляти картою механізму захоплення кадру, показуючи профілі лазерного променя та здійснювати відповідні кількісні обчислення необхідних характеристик, які є складовими профілю. Такі камери як CCD використовуються для хвиль видимого діапазону або піроелектричних хвиль, для інших довжин хвиль використовуються відповідно інші види камер. Таким чином, камера, по суті, є цифровим перетворювачем оптичних сигналів для подальшої їх комп’ютерної обробки.

Сучасне програмне забезпечення дає дуже чіткі 2D та 3D зображення (рис. 1.6), а також виконує надскладні цифрові операції аналізу відповідних складових характеристик профілю лазерного променя [3, 16-19].

В даній роботі зосереджено увагу на дослідженні відповідностей між функцією деформації та геометричними характеристиками 2D зображень профілю лазерного променя, зокрема на методах аналізу розкиду геометричних характеристик сигналу лазерної траси та їх відновлення.



а) б)

Рисунок 1.6 – Високоструктурований лазерний промінь, виміряний за допомогою CCD камери в 2D (а) і 3D (б) форматі

В даній роботі розглядаються вихідні сигнали системи на прикладі послідовностей серій лазерних зображень. Варто також відзначити, що подані методи можуть бути поширені на багатовимірні зображення. Під класифікацією будемо розуміти віднесення кожного окремо взятого зображення відеоряду лазерного променя до одного із попередньо введених еталонних класів, в залежності від значень геометричних характеристик сигналу лазерної траси [1].

Нейронні мережі дають додаткові можливості в моделюванні нелінійних явищ і розпізнаванні хаотичної поведінки. Завдяки своїй великій гнучкості (на одній топології можна реалізувати багато різних відображень), нейронні мережі можуть охоплювати найрізноманітніші структури у фазовому просторі [31, 32].

В даній роботі, на основі проведеного детального аналізу особливостей предметної області, а також попереднього огляду існуючих методів розпізнавання та класифікації було обрано метод нейронних мереж, так як він найкраще підходить для вирішення поставленої задачі.

Об’єкт оброблення, а саме лазерний пучок, що є основою профіля лазерного променя, буде описуватись множиною координат X та Y енергетичного центру зображення відеоряду лазерного променя, а також

усередненим значенням по кожному сектору плямоподібного зображення. Вхідними даними нашого програмного продукту будуть усереднені значення по кожному сектору плямоподібного зображення. Вихідними даними буде віднесення плямоподібного зображення лазерної відеоряду до певного класу. Залишається лише промоделювати обрану технологію розпізнавання та класифікації, використовуючи сучасну обчислювальну техніку і відповідне програмне забезпечення. В результаті чого буде створено власне програмне рішення для отримання необхідного вирішення поставленої задачі.

Нейроподібне розпізнавання базується на застосуванні та подальшій перевірці ефективності нейронних мереж у ролі моделі для контрольованого навчання на прикладі обробки плямоподібних зображень.

1.5 Висновки до розділу 1

Здійснено аналіз предметної області оброблення профілю лазерного променя. Досліджено, що поширення лазерного випромінювання в атмосфері супроводжується набором явищ як лінійної, так і нелінійної взаємодії. При цьому жодне з цих явищ не виявляється окремо. Відзначено, що на даний час у сферах лазерної обробки матеріалів, лазерній локації, оптичному зв'язку, поліграфії й інших областях техніки відчувається гостра необхідність більш широкого впровадження оптико-електронних інтелектуальних систем з автоматичним коректуванням похибок формованого світлового випромінювання. Описано прикладне значення та наведено обґрунтування доцільності розробки інформаційно-вимірювальної системи для оброблення профілю лазерного променя. Наведено аналіз систем-аналогів оброблення профілю лазерного променя, в ході якого відзначено необхідність і актуальність впровадження методів і засобів штучного інтелекту у системи профілювання лазерних променів. Наведено постановку задачі оброблення профілю лазерного променя із застосуванням нейромережевої класифікації плямоподібних зображень.

2 АНАЛІЗ МЕТОДІВ ТА КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ ОБРОБЛЕННЯ ПЛЯМОПОДІБНИХ ЗОБРАЖЕНЬ

2.1 Класифікаційний аналіз методів оброблення та розпізнавання зображень

Для розпізнавання зображень використовуються численні методи, що базуються на різних системних вимогах до обсягу обчислень, швидкості прийняття рішення, допустимого рівня зашумленості тощо. Крім того, задачу розпізнавання зображень доцільно розділити на два етапи [33]:

1. Попередня обробка зображення для подальшого розпізнавання.
2. Процедура розпізнавання.

Попередня обробка зображень може застосовувати методи виділення контурів зображення, скелетизації, згладжування, нарощування областей, адаптивне підвищення контрастності, сегментація тощо. Численні дослідження дають можливість зробити висновок, що попередня обробка зображення є визначною для подальшого успішного його розпізнавання. Для багатокольорових зображень, зображень з градаціями кольорів найбільш значимою і результативною є сегментація, що полягає в розбитті зображення на складові частини за певними принципами. Досить поширеним є використання методу порогової сегментації та керованого вододілу тощо. Так як алгоритми сегментації зображень спершу розроблялись для півтонових зображень, то їх існує велика кількість. Але сегментація кольорових зображень є складнішою задачею. Якщо піксель напівтонового зображення може мати 256 відтінків яскравості, то у випадку кольорового зображення окремий піксель може мати один з 256 відтінків яскравості кожної з трьох компонент кольору, що визначає 16777216 можливих кольорів. Тому виникають труднощі при адаптації методів сегментації напівтонових зображень для обробки кольорових [33, 34].

Для розпізнавання зображень на практиці широко застосовуються кореляційні, ознакові методи, методи головних компонент, первинна та об’єктна векторизації тощо. Зазначені методи різняться за обчислювальною складністю, зручністю реалізації на ПК, завадостійкістю. Але прив’язка до детермінованих алгоритмів, низький ступінь інваріантності до факторів мінливості зображень обмежує їх застосування. Основні методи розпізнавання зображень показано на рис.2.1 [35, 36].

Рисунок 2.1 – Загальна класифікація методів розпізнавання зображень

Одним з підходів до розпізнавання багатокольорових зображень, що забезпечує гнучкість, властивість до узагальнення, і відповідно стійкість до мінливості зображень, є нейромережевий підхід [31, 32, 37, 38].

Нейронні мережі можна також застосовувати для одновимірного і багатовимірного аналізу, певним чином сформував множину незалежних входів і залежних від них виходів.

Нейронні мережі є досить потужним та універсальним засобом для вирішення задач розпізнавання та класифікації. Дана технологія може застосовуватись в будь-якій предметній області, незалежно від її природи. Тобто нейронні мережі не містять особливості предметної області, одне і теж рішення може (при належній обробці даних) може бути застосоване при вирішені різних задач. Нейронні мережі є дуже гнучкою та універсальною технологією. Відомо багато як структур, так і методів навчання нейронних мереж [31, 32].

Загалом, розпізнання зображень можна визначити як віднесення вхідних даних до певного класу з допомогою виділення істотних ознак або властивостей, що характеризують ці дані з загальної маси неістотних деталей.

При всьому різноманітті різних алгоритмів і методів розпізнавання зображень, типовий метод розпізнавання складається з трьох компонент [33-36]:

1. Перетворення вихідного зображення в початкове уявлення (може включати в себе як передобробку, так і математичні перетворення, наприклад обчислення головних компонент);

2. Виділення ключових характеристик (наприклад береться перший n головних компонент або коефіцієнтів дискретного косинусного перетворення);

3. Механізм класифікації (моделювання): кластерна модель, метрика, нейронна мережа тощо.

Крім цього, застосування методу розпізнавання спирається на апріорну інформацію про предметну область і коригується експериментальною інформацією, що з'являється по ходу застосування методу.

Рисунок 2.2. – Узагальнена структурно-функціональна схема процесу розпізнавання зображень

2.2 Нейромережевий підхід для розпізнавання та класифікації зображень

Для того, щоб добитися високої ефективності, нейронні мережі використовують багато взаємозв'язків між елементарними комірками обрахунків – нейронами.

Процедура, яка використовується для процесу навчання, є алгоритмом навчання. Ця процедура виставляє у визначеному порядку синаптичні ваги нейронної мережі для забезпечення необхідної структури взаємозв'язків нейронів [31, 32].

Зміна синаптичних ваг являє собою традиційний метод налаштування нейронних мереж. Цей підхід дуже близький до теорії лінійних адаптивних

фільтрів, яка давно заявила про себе і застосовується в багатьох областях діяльності людини. Однак нейронні мережі можуть змінювати власну топологію. Це обумовлене тим фактом, що нейрони в мозку людини постійно відмирають, а нові синаптичні зв'язки постійно створюються.

Використання нейронних мереж забезпечує такі корисні властивості систем [31, 32, 38]:

1. Нелінійність (nonlinearity). Штучні нейрони можуть бути лінійними і нелінійними. Нейронні мережі, побудовані із з'єднань нелінійних нейронів, самі є нелінійними. Нелінійність є особливо важливою властивістю, а саме коли фізичний механізм, який відповідає за формування вхідного сигналу, також є нелінійним (наприклад, мова людини, зміна профілю лазерного променя, тощо).

2. Відображення вхідної інформації в вихідну. Однією з популярних парадигм навчання є навчання з вчителем. Воно являє собою зміну синаптичних ваг на основі набору позначених навчальних прикладів. Кожний приклад складається з вхідного сигналу і відповідного йому бажаного відгуку. З цієї множини випадковим чином вибирається приклад, а нейронна мережа модифікує синаптичні ваги для мінімізації розходження бажаного вихідного сигналу і формуємого мережею згідно обраному статистичному критерію. При цьому модифікуються вільні параметри мережі. Раніше використані приклади можуть потім бути застосовані знову, але вже в іншому порядку. Це навчання проводиться до тих пір, поки зміни синаптичних ваг не стануть несуттєвими. Таким чином, нейронна мережа навчається на прикладах, складаючи таблицю відповідностей вхід/вихід для конкретної задачі. Такий підхід змушує згадати непараметричне статистичне навчання. Цей напрям статистики має справу з оцінками, не пов'язаними з якою-небудь конкретною моделлю, чи з біологічної точки зору, з навчанням з нуля. Тут термін «непараметричний» використовується для акцентування того, що спочатку не існує ніякої початкової моделі вхідних даних.

3. Адаптивність. Нейронні мережі мають властивість адаптувати свої синаптичні ваги до змін навколишнього середовища. Нейронні мережі, навчені діяти в визначеному середовищі, можуть бути легко перенавчені для роботи в умовах незначних коливань параметрів середовища.

Більш того, для роботи в нестаціонарному середовищі (де статистика змінюється з плином часу) можуть бути створені нейронні мережі, які змінюють синаптичні ваги в реальному часі. Архітектура нейронних мереж може бути об’єднана з їх властивістю до адаптації, що приводить до створення моделей адаптивної класифікації образів, адаптивної обробки сигналів і адаптивного керування. Відомо, що чим вище адаптивні властивості системи, тим більш стійкою буде її робота в нестаціонарному середовищі.

4. Очевидність відповіді. В контексті задачі класифікації зображень можна розробити нейронну мережу, яка збирає інформацію не тільки для визначення конкретного класу, але і для збільшення достовірності рішення, що приймається. Потім ця інформація може бути використана для виключення сумнівних рішень, що підвищить ефективність нейронної мережі.

5. Контекстна інформація. Знання представляються в самій структурі нейронної мережі за допомогою її стану активації. Кожний нейрон мережі потенційно може бути підданий впливу усіх інших її нейронів. Як наслідок, існування нейронної мережі безпосередньо пов’язане з контекстною інформацією.

6. Відмовостійкість. Штучні нейронні мережі потенційно відмовостійкі. Це означає, що при поганих зовнішніх умовах їх ефективність падає несуттєво. Наприклад, якщо пошкоджений якийсь нейрон чи його зв'язок, доступ до запам’ятованої інформації буде ускладнений. Але, враховуючи розподілений характер зберігання інформації в нейронних мережах, можна говорити, що тільки суттєві пошкодження структури нейронної мережі сильно вплинуть на її ефективність. Тому зниження якості роботи нейронної мережі проходить

повільно. Щоб гарантувати відмовостійкість роботи нейронної мережі, в алгоритм навчання потрібно закладати відповідні поправки.

7. Масштабованість. Паралельна структура нейронних мереж потенційно прискорює рішення деяких задач і забезпечує масштабованість нейронних мереж в рамках технології VLSI (Very Large Scale Integrated). Однією з переваг технології VLSI є можливість представляти достатньо складну поведінку за допомогою ієрархічної структури.

8. Єдність аналізу і проєктування. Нейронні мережі є універсальним механізмом обробки інформації. Це значить, що одне і те ж проєктне рішення нейронної мережі може використовуватись в багатьох предметних областях.

Модель нейронних мереж визначається структурою (чи топологією) мережі і алгоритмом навчання. Існують сотні видів структур нейронних мереж і відповідних алгоритмів навчання, і кожний із них призначений для вирішення визначеного типу задач [31, 32].

Класифікація нейроних мереж

Нейронні мережі розрізняють за різноманітними критеріями, такими як структура мережі (зв'язків між нейронами), особливості моделі нейрона, особливості навчання мережі, тощо [31, 32, 38].

За структурою нейронні мережі можна розділити на неповнозв’язні (або шарові) і повнозв’язні, з випадковими й регулярними зв'язками, із симетричними й несиметричними зв'язками (рис.2.5) [32].

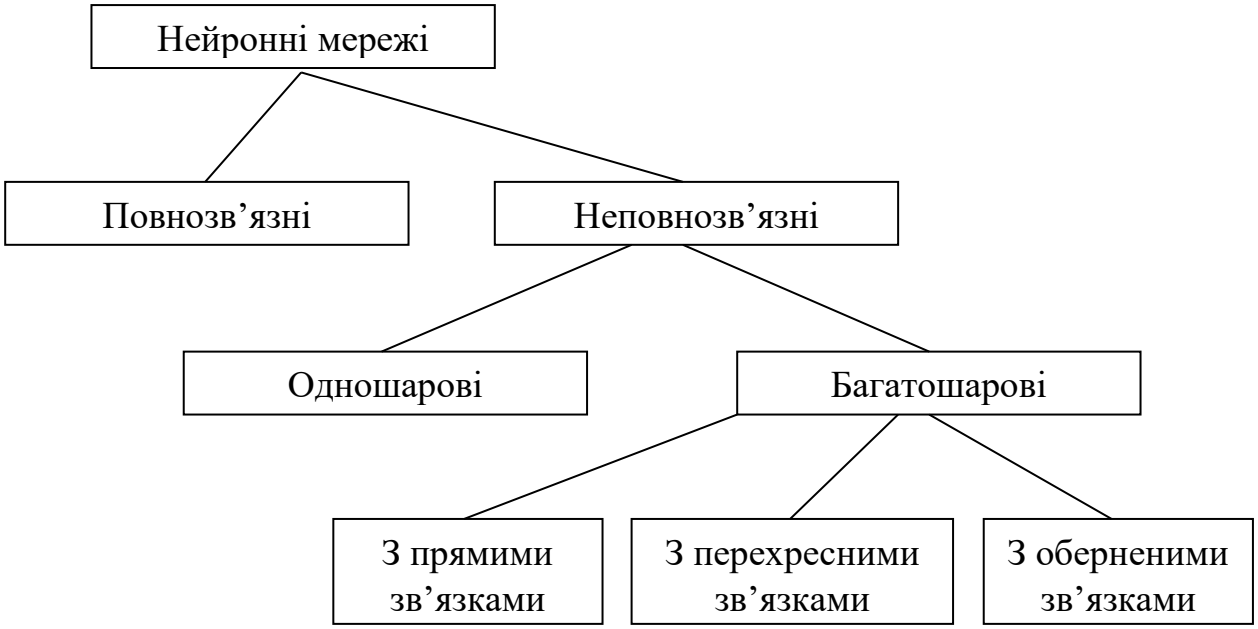


Рисунок 2.3 – Узагальнена класифікація нейронних мереж за структурою

Неповнозв’язні нейронні мережі (описувані неповнозв’язним орієнтованим графом названі перцептронами), підрозділяються на одношарові (найпростіші перцептрони) і багатошарові, із прямими, перехресними й зворотними зв'язками. У нейронних мережах із прямими зв'язками нейрони j-ого шару по входах можуть з'єднуватися тільки з нейронами і-их шарів, де $j > i$, тобто з нейронами нищерозміщених шарів. У нейронних мережах з перехресними зв'язками допускаються зв'язки всередині одного шару, тобто вище наведена нерівність заміняється на $j >= i$. У нейронних мережах зі зворотними зв'язками використовуються зв'язки j-ого шару по входах з i-им при $j < i$. Крім того, по виду зв'язків розрізняють перцептрони з регулярними й випадковими зв'язками [32, 39].

В результаті аналізу структур нейронних мереж, зупинимося на структурі багатошарового персептрона. Цей вибір обумовлений тим, що дана структура більш універсальна, її легко можна доповнювати або спрощувати, на ній вільно реалізовуються багато алгоритмів навчання нейронних мереж [39].

В якості алгоритму навчання оберемо алгоритм зворотного розповсюдження помилки. Такий вибір обумовлений точністю навчання даним методом, його підтримкою багатьма засобами моделювання, легкістю навчання та адаптуванням під визначену задачу [32].

Архітектура FeedForward BackPropagation була розроблена на початку 1970-х років декількома незалежними авторами: Вербор (Werbor); Паркер (Parker); Румельгарт (Rumelhart), Хінтон (Hinton) і Вільямс (Williams). Загалом, Backpropagation досить популярна, ефективна і легка модель навчання для складних, багатошарових мереж. Вона використовується в різних типах програмних додатків і породила великий клас нейронних мереж з різними структурами і методами навчання [32, 33].

Типова мережа Backpropagation має вхідний шар, вихідний шар і принаймні один прихований шар. Теоретично, обмежень щодо кількості прихованих шарів не існує, але на практиці зазвичай застосовують один або два (рис. 2.4) [32, 39].

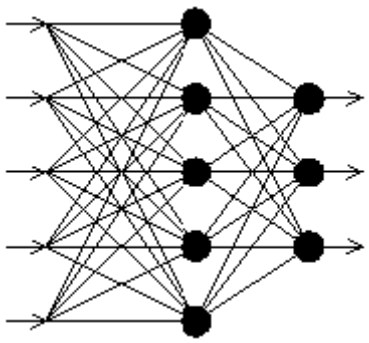


Рисунок 2.4 – Структура нейронної мережі на основі багатошарового перцептрона

Нейрони організовані в пошарову структуру з прямою передачею сигналу. Кожен нейрон мережі продукує зважену суму своїх входів, пропускає цю величину через передаточну функцію і видає вихідне значення. Мережа може моделювати функцію практично будь-якої складності, причому кількість шарів і

кількість нейронів у кожному шарі визначають складність функції. Визначення кількості проміжних шарів і кількості нейронів у них є важливим при моделюванні мережі. Застосовуючи архітектуру до визначених задач використовують загальні правила, зокрема [31, 32, 39]:

1. Кількість входів і виходів мережі визначаються кількістю вхідних і вихідних параметрів досліджуваного об'єкта, явища, процесу, і т.п.. На відміну від зовнішніх шарів, кількість нейронів схованого шару обирається емпіричним шляхом. У більшості випадків достатня кількість нейронів складає $n \leq n_{вх} + n_{вих}$, де $n_{вх}$, $n_{вих}$ – кількість нейронів у вхідному і, відповідно, у вихідному шарах.

2. Якщо складність у відношенні між отриманими і бажаними даними на виході збільшується, кількість нейронів схованого шару повинна також збільшитися.

3. Якщо моделюємий процес може розділятися на багато етапів, потрібний додатковий прихований шар (шари). Якщо процес не розділяється на етапи, тоді додаткові шари можуть допустити перезапам'ятовування і, відповідно, неправильне загальне рішення.

Після того, як визначено кількість шарів і кількість нейронів у кожному із них, потрібно знайти значення для синаптичних ваг і порогів мережі, здатних мінімізувати помилку спродукованого результату. Саме для цього існують алгоритми навчання, де відбувається підлаштування моделі мережі до наявних навчальних даних. Помилка для конкретної моделі мережі визначається шляхом проходження через мережу всіх навчальних прикладів і порівняння спродукованих вихідних значень з бажаними значеннями. Множина помилок створює функцію помилок, значення якої можна розглядати, як помилку мережі. В якості функції помилок частіше використовують суму квадратів помилок.

Алгоритм навчання мережі [31, 32]:

1. Ініціалізація мережі: вагові коефіцієнти і зміни в мережі приймають малі випадкові значення.

2. Визначення елемента навчальної вибірки (вхід – вихід). Входи (x_1, x_2... x_n), повинні розрізнятися для всіх прикладів навчальної вибірки.

3. Обчислення вихідного сигналу:

$$S_{i_m} = \sum_{i_{m-1}}^{N_{m-1}} W_{i_m j_{m-1}} Y_{i_{m-1}} - b_{i_m} \tag{2.1}$$

$Y_{Im}= f(S_{Im})$;

Im=1, 2, ..., N,

m=1, 2, ..., L.

де *S* - вихід суматора, *w* - вага зв'язку, *y* - вихід нейрона, *b* - зсув, *i* - номер нейрона, *N* - кількість нейронів у шарі, *m* - номер шару, *L* - кількість шарів, *f* - передаточна функція.

4. Налаштування синаптичних ваг:

$$w_{ij}(t+1)=w_{ij}(t)+rg_jx_{i'} \tag{2.2}$$

де w_{ij} - вага від нейрона i або від елемента вхідного сигналу i до нейрона j у момент часу t, $x_{i'}$ - вихід нейрона i, r - швидкість навчання, g_j - значення похибки для нейрона j.

Якщо нейрон з номером j належить останньому шару, тоді

$$g_j=y_j(1-y_j)(d_j-y_j) \tag{2.3}$$

де d_j - бажаний вихід нейрона j, y_j - потоковий вихід нейрона j.

Якщо нейрон з номером j належить одному з шарів з першого по передостанній, тоді

$$g_j = X'_j(1 - X'_j)\sum_k g_k W_{jk} \tag{2.4}$$

де k проходять усі нейрони з номером на одиницю більше, ніж у того, якому належить нейрон j.

Тип вхідних сигналів – цілі або дійсні. Тип вихідних сигналів – дійсні з інтервалу, заданого передаточною функцією нейрона. Тип передатної функції –

сигмоїдальна. У нейронних мережах застосовуються декілька варіантів сигмоїдальних передаточних функцій [31, 32].

Функція Фермі (експонентна сигмоїда) [31, 32]:

$$f(S) = \frac{1}{1 + e^{-2aS}} \tag{2.5}$$

де S – вихід суматора нейрона.

Раціональна сигмоїда [31, 32]:

$$f(S) = \frac{S}{|S| + a} \tag{2.6}$$

Гіперболічний тангенс [31, 32]:

$$f(S) = th\frac{S}{a} = \frac{e^{-\frac{S}{a}} - e^{-\frac{S}{a}}}{e^{\frac{S}{a}} + e^{-\frac{S}{a}}} \tag{2.7}$$

Згадані функції відносяться до однопараметричних. Значення функції залежить від аргументу та одного параметра. Також використовуються багатопараметричні передаточні функції, наприклад [31, 32]:

$$f(S) = p_1 \frac{S}{|S| + p_2} + p_3 \tag{2.8}$$

Сигмоїдальні функції є монотонно зростаючими і мають відмінні від нуля похідні по всій області визначення. Ці характеристики забезпечують правильне функціонування і навчання мережі.

2.3 Комп'ютерне моделювання нейромережевого оброблення профілю лазерного променя з класифікацією плямоподібних зображень

Серед існуючих парадигм нейронних мереж (НМ) в рамках вирішення задач класифікації програмний пакет моделювання STATISTICA Neural Networks (SNN) дозволяє виконати моделювання таких НМ: лінійні (Linear), багатошарові персептрони (MLP), мережі на радіально-базисних функціях (RBF), ймовірнісні мережі (PNN) [40, 41].

Елементами вхідного вектора даної нейромережевої моделі є усереднені значення інтенсивності кольору по 30 зонам отримані в результаті попередньої обробки плямоподібних зображень. Змінні вхідного вектора представлені дробовими додатними числами з точністю 10^{-7} і можуть приймати значення від 0 до 1. Фрагмент даних файлу з вхідними даними (j1.sta) наведено на рис. 2.5. В ньому VAR1-VAR30 – вхідні змінні, GOOD та BAD – еталонні виходи НМ

З одного лазерного відеоряду (2044 зображення) за результатами виявлення характерних форм плям в залежності від рівня спотворення для навчання нейронної мережі було відібрано 140 зображень (70 «хороших» і 70 «поганих»). Передусім, необхідно визначити критерій оптимальної складності мережі – емпіричний метод оцінки похибки узагальнення. Оскільки похибка узагальнення визначена для даних, котрі не належать до навчальної множини, очевидним вирішенням проблеми є поділ даних на 3 множини. Вбудований в SNN «Майстер розв’язків задач» за замовчуванням розподіляє вхідні набори на навчальну, контрольну та тестову множину у співвідношенні 2:1:1. В дослідженні, що проводилось, було виконано такий розподіл даних:

1) навчальна вибірка (80 наборів змінних), що забезпечує налаштування ваг в процесі навчання;

2) контрольна вибірка (30 наборів змінних), що забезпечує контроль процесу навчання та допомагає запобігти перенавчанню мережі;

3) тестова вибірка (30 наборів змінних), що призначення для оцінки властивостей класифікації вже навченої мережі.

Data Set Editor (j1)

Variables 80 2 Cases 80 30 30

	VAR24	VAR25	VAR26	VAR27	VAR28	VAR29	VAR30	GOOD	BAD
01	0.6261801	0.6327745	0.6205433	0.6343469	0.6291394	0.6143033	0.6043503	1	-1
02	0.6255991	0.6305769	0.6276668	0.6184522	0.6238199	0.6189827	0.6042149	1	-1
03	0.6243067	0.6213735	0.6221929	0.6173172	0.6187953	0.6191804	0.6027055	1	-1
04	0.6163222	0.6293095	0.6194144	0.6108118	0.621392	0.6185113	0.6018373	1	-1
05	0.6205264	0.6192547	0.6151587	0.6131391	0.6142378	0.612155	0.6020831	1	-1
06	0.6163399	0.6152316	0.612952	0.6072558	0.614488	0.6128824	0.6012374	1	-1
07	0.6123275	0.610988	0.6121784	0.6056077	0.6100399	0.6136592	0.6008488	1	-1
08	0.6173384	0.6112532	0.611297	0.6079947	0.6113834	0.6135455	0.6002996	1	-1
09	0.6111994	0.611616	0.606788	0.6079348	0.6056527	0.6135861	0.5987792	1	-1
10	0.612489	0.6129728	0.611923	0.606652	0.6114997	0.6094601	0.5988715	1	-1
11	0.6097156	0.6053341	0.6066649	0.6093091	0.6077901	0.6051668	0.5985909	1	-1
12	0.6139615	0.6093777	0.607987	0.6066117	0.6131445	0.6066875	0.5989118	1	-1
13	0.6109659	0.6081273	0.6062961	0.6064223	0.6111837	0.6098323	0.5976054	1	-1
14	0.6072602	0.6063935	0.6055043	0.6068719	0.6074015	0.6088096	0.5988631	1	-1
15	0.6147147	0.6090512	0.6114833	0.6053693	0.6074368	0.609107	0.6009606	1	-1
16	0.6100399	0.6043762	0.6079151	0.6070853	0.6154503	0.6115753	0.60041	1	-1
17	0.6105301	0.6119352	0.6081489	0.6056645	0.607008	0.602046	0.599249	1	-1
18	0.6081518	0.6055129	0.6004677	0.5999053	0.6011256	0.6039216	0.5975992	1	-1
19	0.6116376	0.6092451	0.6058824	0.6021597	0.6029956	0.6068201	0.5999097	1	-1
20	0.6068264	0.6062328	0.6040115	0.6023302	0.599655	0.6017998	0.5985507	1	-1

Рисунок 2.5 – Фрагмент масиву вхідних даних

“Майстром розв’язків задач” SNN було побудовано 637 мереж, з яких за критеріями балансу між розміром мережі і значенням помилки та заміною вже знайдених мереж на нові знайдені мережі з кращими показниками було відібрано 10 нейронних мереж з найкращими показниками значень помилки (для мереж, що мають кількість виходів більше 1 показник Performance не є достовірним та інформативним), які показані на рис. 2.6.

Network Set Editor (j1)

Current network 8 Detail shown Basic Options...

	Type	Error	Inputs	Hidden
01	Linear	0.7237867	30	-
02	RBF	0.6133457	30	32
03	RBF	0.5772269	30	34
04	RBF	0.6994971	30	10
05	RBF	0.6896635	30	17
06	MLP	0.5614887	30	38
07	MLP	0.5560491	30	58
08 *	MLP	0.5985287	30	16
09	MLP	0.6549163	30	25
10	MLP	0.6418512	30	16

Рисунок 2.6 – Перелік найкращих варіантів структурної організації нейронної мережі

Очевидним є факт, що найбільшу помилку має лінійна мережа, зважаючи на характер поставленої задачі. НМ на радіально-базисних функціях мають непогані показники, і мінімальна помилка серед RBF мереж досягається мережею під номером “03” і має значення 0,5772269. В RBF-мережах процес навчання відбувається значно швидше, ніж в багатошарових персептронах, але при цьому для досягнення високої ефективності розпізнавання вони значно програють в розмірах та, відповідно, в кількості обчислень. Тому вибір кращої мережі був між MLP з номером “07” зі значенням помилки 0, 5560491 і 58 нейронами у прихованому шарі та “08” зі значенням помилки 0,5985287 і 16 нейронами у прихованому шарі. Зважаючи на складність організацію сьомої НМ та існуючі рекомендації щодо кількості нейронів у прихованому шарі було обрано нейронну мережу “08”, яка зображена на рис. 2.7.

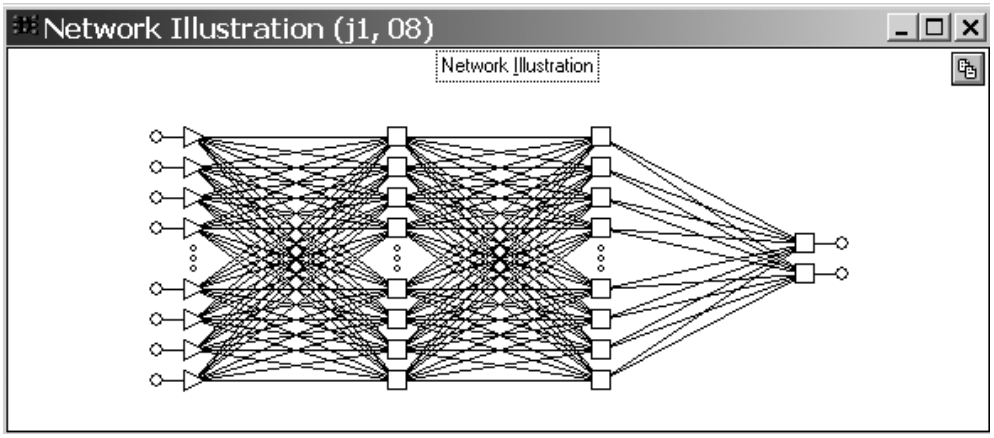


Рисунок 2.7 – Структурна організація обраної для моделювання НМ

Функція відображення простору вхідних даних Data Clustering у заданій системі координат (вісі абсцис та ординат визначається довільними двома наборами елементів вхідного вектора) ілюструє лінійну нероздільність класів, до яких необхідно віднести подані на входи зображення (рис. 2.8).

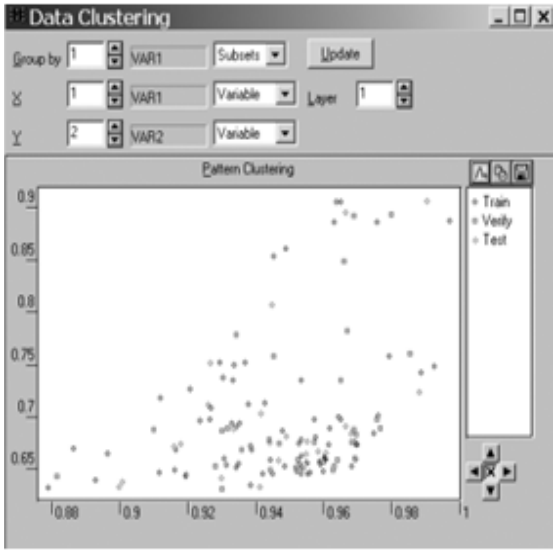


Рисунок 2.8 – Екранна форма вікна DataClustering,

Відповідно, нейронній мережі складно навчитись розпізнавати деякі зображення і значення помилки їх розпізнавання значно вище за середнє. Вікно CaseErrors (рис. 2.9) дозволяє легко виявити такі елементи вибірки на гістограмі. Дана гістограма також наочно надає можливість оцінити загальну якість розпізнавання зображень вибірки за рівнем стовпців-помилок.

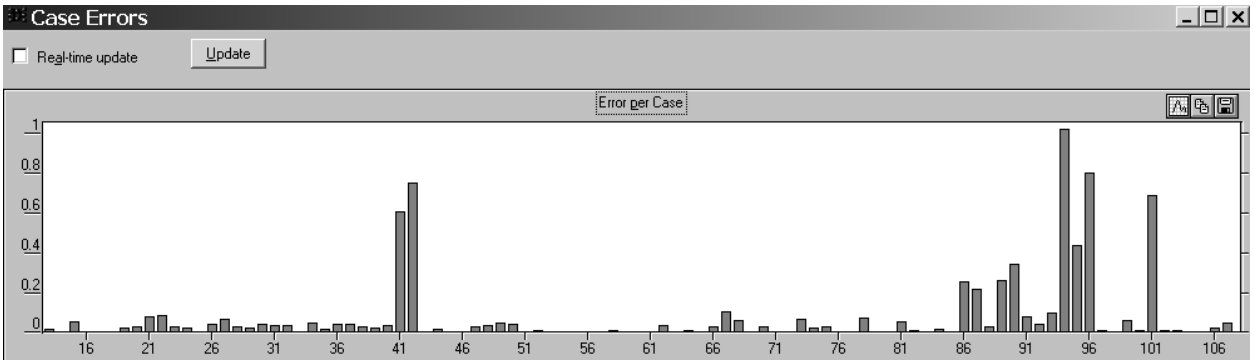


Рисунок 2.9 – Гістограма помилок розпізнавання

Для плямових зображень неоднозначними для нейромережевого розпізнавання є зображення з порядковими номерами 41, 42, 94, 06 та 101, які проілюстровані на рис.2.10. Всі ці зображення належать класу «поганих».

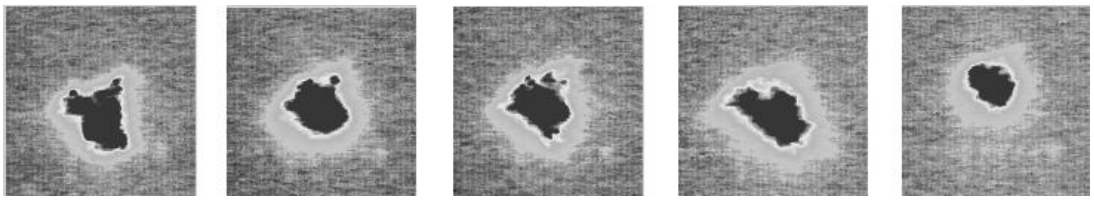


Рисунок 2.10 – Вхідні плямові зображення з найгіршою точністю розпізнавання

2.4 Висновки до розділу 2

Здійснено класифікаційний аналіз методів оброблення та розпізнавання зображень. Наведено характерні особливості процесу розпізнавання зображень. Відзначено перспективність застосування нейромержевого підходу для задач розпізнавання та класифікації зображень. Здійснено класифікаційний аналіз нейронних мереж та методів їх навчання. Здійснено комп’ютерне моделювання нейромережевого оброблення профілю лазерного променя з класифікацією плямоподібних зображень у програмному пакеті моделювання STATISTICA Neural Networks, що дозволило експериментально дослідити перспективи застосування різних типів нейронних мереж та методів їх навчання для вирішення задачі нейромережевої класифікації плямоподібних зображень.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНОЇ СИСТЕМИ ДЛЯ ОБРОБЛЕННЯ ПРОФІЛЮ ЛАЗЕРНОГО ПРОМЕНЯ

3.1 Реалізація апаратної складової інформаційно-вимірювальної системи для оброблення профілю лазерного променя

При застосуванні лазера на практиці, потрібно весь час проводити корекцію лазеру (в тому числі й вручну), адже його параметри дуже чутливі до впливу зовнішнього середовища, а це дуже сильно ускладнює застосування лазера. Ідентифікація та розпізнавання плямоподібних зображень в системах профілювання лазерного променя дозволяє автоматизувати процес використання лазера на практиці.

Розглянемо в загальному вигляді апаратний склад найпростішої системи вимірювання профілю лазерного променя для напівпровідникової лазерної системи (рис. 3.1). За допомогою наведеної системи профілювання лазерного променя можна виміряти такі основні характеристики [2, 3, 6, 8]:

– профіль пучка в довільному перетині;

– еліптичність пучка;

– довжина і радіус перетяжки;

– кутова розбіжність;

– центр мас пучка;

– діаметри пучка D_{ox} і D_{oy} ;

– параметр якості лазерного пучка M^2 .

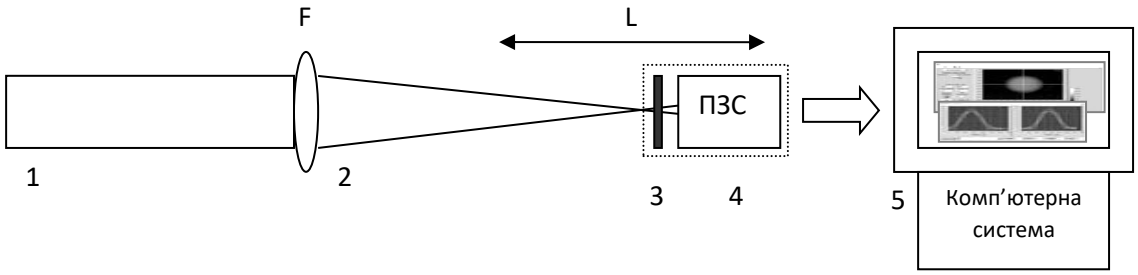


Рисунок 3.1 – Схема вимірювання профілю лазерного променя для напівпровідникової лазерної системи: 1 – лазерне випромінювання, 2 – фокусуюча лінза, 3 – система нейтральних фільтрів, 4 – ПЗС-камера, 5 – комп'ютерна система введення і обробки зображення.

Зупинимося більш детально на способі вимірювання кожного з параметрів.

Для контролю поперечної структури лазерного випромінювання вимірюється профіль інтенсивності (густини потужності) лазерного пучка (профілю), тобто в загальному випадку залежність $I(\vec{r})$, а в двовимірному випадку $I(x,y)$. Профіль гауссівського пучка буде являти собою залежність вигляду:

$$I(\vec{r}) = I(x,y) = I_0 \times \exp(-r^2 / r_0{}^2) = I_0 \times \exp(-\frac{x^2 + y^2}{r_0{}^2}), \qquad (3.1)$$

де I_0 – інтенсивність в центрі гауссівського пучка, r_0 – початковий радіус гауссівського пучка. Після того, як лазерний пучок потрапляє па ПЗС-матрицю Fireware-камери, на екрані комп'ютера формується зображення (рис. 3.2) що і є розподілом інтенсивності залежно від (x,y), тобто поперечний профіль лазерного пучка [6, 8, 17-19].

Переміщаючи камеру уздовж пучка, можна прописати перетяжку лазерного випромінювання, отримавши залежність діаметра пучка від відстані

уздовж його осі. Випромінювання якісного He-Ne лазера має майже дифракційну розбіжність при практично гауссівському профілю пучка. Випромінювання напівпровідникового лазера має істотно гіршу розбіжність і профіль. З рис. 3.3 і з аналізу профілю лазерного пучка в довільному перетині видно, що із зміною відстані z, змінюється і радіус пучка, відповідно. Область, де радіус пучка змінюється в межах від „$\sqrt{2}w_0$” до „w_0” називається „перетяжкою”, а довжина цієї області $2b$ – „довжиною перетяжки”, „радіусом перетяжки” називається радіус пучка в цій області „w_0” [6, 8, 17-19].

Рисунок 3.2 – Зображення пучка лазера

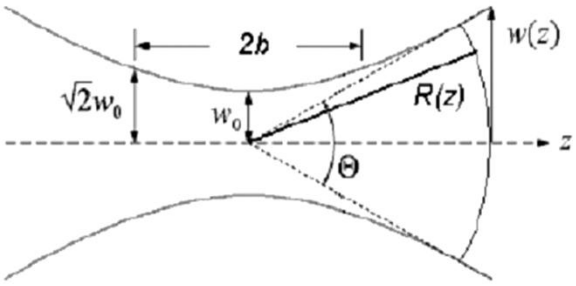


Рисунок 3.3 – Поздовжній профіль гауссівського пучка

Хвильова поверхня гауссівського пучка має кривизну хвильового фронту:

$$R(z) = z\left(1 + \left(\frac{b}{z}\right)^2\right) \tag{3.2}$$

звідки видно, що в перетяжці хвильовий фронт є плоским. На відстані радіус хвильової поверхні прямує до радіуса сферичної хвилі з центром в місці перетяжки: $R(z) = z$. Кутове розходження дорівнює: $\theta = \frac{2w_0}{b}$. Видно, що цю розбіжність можна розуміти як таку, що з'явилася в результаті дифракції на отворі з радіусом порядку w_0. Таким чином, при кожному положенні камери дуже зручно проводити вимірювання цих величин, включаючи і діаметри пучка D_{ox} і D_{oy} і еліптичність лазерного пучка, які визначаються, як $Q = D_{ox} \diagup D_{oy}$ [6, 8].

При цьому для напівпровідникових лазерів існує такий ефект, що до фокальної площини поперечний профіль пучка буде еліпсом, потім при наближенні з фокальною площиною він почне стискатися, досягнувши її перетвориться на коло, а після фокальної площини перетворюється з вертикально-орієнтованого у горизонтально-орієнтований. Останнє пояснюється тим, що пучок, зважаючи на конечність площин дзеркал лазера, розходиться. Через цю розбіжність хвиля виходить неплоска, але її можна описати, як плоску, що проходить через деяку апертуру. Тобто, пучок дифрагує, проходячи через цю віртуальну апертуру. А далі простіше: чим ширше початковий промінь тим вужча дифракційна картина, і навпаки. Тобто для найширшого перетину еліпса вийде найвужчий перетин зображення, а для найвужчого перетину еліпса буде найширший перетин зображення. По суті, еліпс „перекинеться”. Єдине потрібно відзначити, щоб цей ефект був помітний, апертура повинна бути маленькою, тобто хвиля повинна бути істотно неплоскою. Головне, що якщо цей ефект існує, то вимірюючи в автоматичному режимі еліптичність Q ми зможемо зрозуміти, де ми знаходимося по z – у фокальній площині, до або після неї. А знаючи основні параметри лазерної системи, легко визначити фокус і оптичну силу лінзи [2, 6, 8].

Визначення центру мас лазерного пучка також представляє інтерес, оскільки вимірювання цього параметра визначає вимірювання решти

вищеописаних величин, у тому числі і параметра якості пучка M^2. По аналогії з механічними системами, де центр мас системи визначається як середньозважене значення , тут можна записати, що [42]:

$$X^{n}_{ц.м.} = \frac{\rho(x_0)x_0 + \rho(x_1)x_1 + \rho(x_2)x_2 + ... + \rho(x_n)x_n}{\rho(x_0) + \rho(x_1) + \rho(x_2) + .. + \rho(x_n)} \text{ , де } \rho(x) = \sum_{y=0}^{\infty} I(x,y) \qquad (3.3)$$

$$Y^{n}_{ц.м.} = \frac{\rho(y_0)y_0 + \rho(y_1)y_1 + \rho(y_2)y_2 + ... + \rho(y_n)y_n}{\rho(y_0) + \rho(y_1) + \rho(y_2) + .. + \rho(y_n)} \text{ , де } \rho(y) = \sum_{x=0}^{\infty} I(x,y) \qquad (3.4)$$

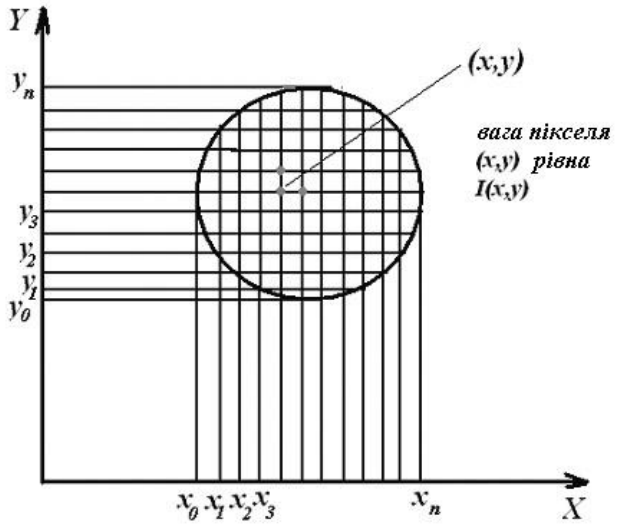


Рисунок 3.4 – Структурна модель розрахунку центру мас лазерного пучка

Якість лазерного пучка може оцінюватися різними методами. Одним з таких методів є оцінка ступеня близькості профілю досліджуваного пучка до профілю гауссівського пучка. Для цієї мети вводиться параметр M^2 який дорівнює відношенню діаметра перетяжки досліджуваного пучка (за рівнем інтенсивності 1/e) до діаметра перетяжки гауссівського пучка [3, 6, 8, 42].

Відповідно стандарту ISO 11146 алгоритм вимірювання параметра якості пучка M^2 припускає [3, 42]:

1) вимірювання діаметра пучка D (по рівню поля 1/e) в межах трьох довжин Релея (L_R= b/2, де b – довжина перетяжки) відносно положення перетяжки пучка,

2) апроксимацію отриманих даних гіперболічною залежністю,

3) розрахунок M^2.

Перетяжка створюється фокусуючою лінзою з фокусною відстанню, напр. 10 см. Основним критерієм оптимальних параметрів фокусуючої системи може бути обрано мінімальний розмір плями випромінювання на поверхні ПЗС-камери. Повноцінне вимірювання діаметра пучка можливо при реєстрації діаметра не менше 15 пікселями ПЗС-матриці відеокамери, яка на практиці може складати 7 мкм. Переміщення ПЗС-камери уздовж осі розповсюдження пучка може здійснюватися з використанням системи переміщення, наприклад, керованої контролером крокового двигуна.

Діаметр гауссівського пучка залежить від відстані за виразом [3, 17-19, 42]:

$$D^2(z) = D_0^2 + (z - z_0)^2 \Theta^2 / 2 \qquad (3.5)$$

де D_0 – ширина перетяжки пучка за рівнем 1/e інтенсивності, z_0 – положення перетяжки пучка, Θ – розбіжність випромінювання. Після апроксимації експериментально виміряної залежності *D(z)* вказаною вище функцією, розраховується параметр якості лазерного пучка M^2 за виразом [3, 42]:

$$M^2 = \frac{\pi\sqrt{2}}{4\lambda} \cdot D_0 \Theta . \qquad (3.6)$$

Для проведення досліджень розроблено апаратний макет інформаційно-вимірювальної системи для оброблення зображень профілю лазерного променя (рис. 3.5), що являє собою основу, розмірами 100×50×500 мм на кінцях якої

закріплено лазер (справа) та відеокамеру (зліва). Лазер тримається на двоплечовому шарнірі розмірами 50 та 65 мм. Фіксатором лазера є трубка зовнішнім діаметром 22 мм з прижимним гвинтом довжиною 16 мм. Зліва камера тримається також на двоплечовому шарнірі розмірами 50 мм кожен. Камера поставлена таким чином, що її можна обертати по осі X на 360 градусів. Принцип роботи макету – вихідний промінь лазера проекціюється на сенсор відеокамери, за допомогою програми відеозапису поведінка лазерного променя записується на жорсткий диск комп'ютерної системи для подальшого аналізу [1].

Рисунок 3.5 – Апаратний макет інформаційно-вимірювальної системи для оброблення профілю лазерного променя з класифікацією плямоподібних зображень

Загальні параметри:

1. Технічні параметри лазера:
   - маркування – HLDPM10-650-1;

- довжина хвилі – 650нм (червоний);
- потужність – 1 мВт;
- напруга – 3 В.

2. Технічні параметри відеокамери:

- маркування – Grand i-See 550;
- інтерфейс з'єднання з комп'ютером – USB1.1/USB2.0;
- режими роботи камери – 320×240 60 к/с;
  640×480 30 к/с;
  1280×1024 10 к/с;
- режим відео – RGB24;
- тип відео кристала – 1/6 CMOS VGA sensor;
- статична роздільна здатність – 640×480; 2560×2048; 3200×2400;
- операційна система – MS Windows.

При проведенні дослідження використовувалась потужність лазера 0,04 – 0,07 мВт, роздільна здатність відеокамери – 640×480 пікселів при 30 кадрах за секунду. Метод стиснення відеофайлів – MPEG-2. Далі відеофайл було розділено на 8-бітні зображення розмірністю 128x128 пікселів у форматі „bmp".

3.2 Узагальнена структурно-функціональна модель інформаційно-вимірювальної системи для оброблення профілю лазерного променя

В даному дослідженні процес оброблення зображень профілю лазерного променя з їх подальшою класифікацією складається з двох основних етапів: попереднього інтелектуального оброблення плямоподібного зображення та нейромережевої класифікації. Розглянемо послідовно обидва етапи.

Інтелектуальна передобробка вхідних зображень базується на: виділенні інформативної області зображення, топологічному аналізі отриманої робочої області та усередненні інтенсивності кольору по кожній зоні.

На вхід системи надходять 8-бітні зображення розмірністю 128x128 пікселів. Інтелектуальна передобробка призначена для відокремлення зображення інформативної частини («плями» зображення лазерного променя) від фонової частини зображення для подальшого аналізу і передбачає виконання наступних кроків:

1. Визначення центра максимальної інтенсивності методом знаходження центра мас зображення [1].

Ми працюємо з двовимірною ортогональною системою координат, тому для визначення кожної координати можна навести наступні вирази:

$$x = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}(\omega_{ij} - \omega_{bg})^k}{\sum_{i=1}^{n}\sum_{j=1}^{m} (\omega_{ij} - \omega_{bg})}; \qquad y = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}(\omega_{ij} - \omega_{bg})^k}{\sum_{i=1}^{n}\sum_{j=1}^{m} (\omega_{ij} - \omega_{bg})}; \tag{3.7}$$

де x – абсциса центра максимальної інтенсивності; y – ордината центра максимальної інтенсивності; x_{ij} – абсциса поточного пікселя з координатами *(i;j)*; y_{ij} – ордината поточного пікселя з координатами *(i;j)*; ω_{ij} – ваговий коефіцієнт поточного пікселя (фізичний зміст якого – інтенсивність, яскравість кольору); ω_{bg} – ваговий коефіцієнт фону, $\omega_{bg} \in [0;+\infty)$; k – коефіцієнт уточнення (виокремлення) градації яскравості.

Якщо обрати за початок відліку верхній лівий кут зображення, то вказані рівняння приймуть вигляд:

$$x = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} i(\omega_{ij} - \omega_{bg})^k}{\sum_{i=1}^{n}\sum_{j=1}^{m} (\omega_{ij} - \omega_{bg})}; \qquad y = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} j(\omega_{ij} - \omega_{bg})^k}{\sum_{i=1}^{n}\sum_{j=1}^{m} (\omega_{ij} - \omega_{bg})}. \tag{3.8}$$

В розроблюваній системі використовуються вхідні плямоподібні зображення лазерного променя в кольоровій моделі RGB, а оброблення зображення виконується посередництвом використання певних компонент моделі „CIE XYZ”. В зазначених 3-компонентних кольорових моделях вибір основних кольорів зумовлений особливостями фізіології сприйняття кольору сітківкою ока людини. Основна ж особливість „XYZ” моделі в тому, що будь-який фізично сприйманий колір представляється лише додатними величинами.

Для визначення інтенсивності кольору для кожного пікселя та для фонового кольору в RGB- зображенні необхідно виділити відповідні три компоненти „R”, „G” та „B”, і виконати лінійне перетворення в модель „XYZ” по Y-складовій.

Для представлення кольорів в моделі RGB в MS Windows використовується стандартний 4-байтний тип COLORREF. При визначенні будь-якого RGB кольору, значення змінної типу прийнято подавати в шістнадцятковому вигляді так:

0x00bbggrr,

де rr, gg, bb — значення інтенсивності червоної, зеленої та синьої складових кольору. Максимальне їх значення — 0xFF.

Для виділення окремої компоненти кольору необхідно виконати побітовий зсув на відповідну кількість розрядів.

$$R = \frac{\text{0x00FF0000 \& rgb}}{65536 * 255}; \qquad G = \frac{\text{0x0000FF00 \& rgb}}{256 * 255}; \qquad B = \frac{\text{0x000000FF \& rgb}}{255}. \tag{3.9}$$

Величина Y кольорової моделі XYZ є відносною яскравістю:

$$Y = 0.2126 * R + 0.7152 * G + 0.0722 * B. \tag{3.10}$$

В контексті даної задачі компонента Hue (відтінок) кольорової моделі „HSV(HSB)” взаємооднозначна з компонентою Y моделі XYZ і відображає функцію яскравості (luminosity function – зелений колір має більший вплив на сприйняття яскравості оком людини, червоний – менше, найменше – синій). Так як рівні яскравості плями зображення лазерного пучка представляються кольорами спектру, то ми фактично ми знайшли яскравість:

$$\omega_{ij} = Y = 0.2126 * R + 0.7152 * G + 0.0722 * B, \quad 0 \le \omega_{ij} \le 1 \tag{3.11}$$

При знаходженні зваженого центру максимальної інтенсивності ми враховуємо всі пікселі вхідного квадратного зображення, тому фонові пікселі мають вплив на визначення центру і дають похибку в результуючих координатах. Саме тому для більш точного визначення використовується коефіцієнт уточнення (виокремлення) градації яскравості k-ому степені.

2. Виділення вписаного в квадрат 128x128 пікселів кола максимального радіусу з центром в точці максимальної інтенсивності.

Знаходимо максимальний радіус кола з центром в точці максимальної інтенсивності, яке можна вписати в квадрат розмірністю 128x128 пікселів, і виокремлюємо його для подальшої обробки.

$$R_{\max} = Min((width - x), x, (height - y), y). \tag{3.12}$$

3. Уточнення інформативної частини зображення шляхом виокремлення фонового кольору.

Ітеративно аналізуємо середню інтенсивність кольору в кільці, утвореному колами з радіусом $R_{\max}$ та $R_{\max} - 2$, порівнюючи її з ω_{bg} та з кожною ітерацією

зменшуючи на 1 піксель $R_{\max}$ до певної межі. Коли знайдено інтенсивність більшу ніж ω_{bg}, вирізаємо зображення з поточним радіусом R .

Для опису початкового вхідного зображення заданої розмірності необхідно було б: 128x128x3=49152 байт пам'яті. Після застосування інтелектуальної передобробки робоче зображення має несуттєво менший розмір, тому необхідний обсяг пам'яті залишається не прийнятним для подальшого подання на входи нейронної мережі. Тому в даній системі застосовується сегментація (топологічний аналіз) зображення з подальшим усередненням кольору сегменту. Сегментація є відомим методом, що використовується для початкового аналізу зображень, але саме „оптимальна" сегментація багато в чому визначає правильність та ефективність вирішення задачі.

Проаналізувавши велику кількість плямових зображень лазерних відеорядів (14 відеорядів по 2044 зображення в кожному), області на зображенні було класифіковано за впливом на розташування енергетичного центру та загальну оцінку „правильності" форми плями. Зображення розбивається на 5 кілець інтенсивності і відповідно на 30 зон (рис. 3.6). По відношенню до радіусу виділеної інформативної частини зображення радіуси внутрішніх кілець розподілені таким чином:

$$R_0 < 0.4R;\ \ 0.4R \le R_1 < 0.6R;\ \ 0.6R \le R_2 < 0.7R;\ \ 0.4R \le R_3 < 0.9R;\ \ 0.9R \le R_4 \le R.$$

Центральна зона є найбільш важливою і має охоплювати частину зображення з максимальною яскравістю. Наступні кільця розділені на різну кількість секторів відповідно до свого номеру: друге кільце на 2^2 сектора, третє – на 2^3, четверте – на 2^4. Останнє кільце не розбивається на підзони, тому що воно є граничним між інформативною частиною зображення і фоном. Периферійні точки в реальних зображеннях найбільш сильно піддані флуктуаціям (це

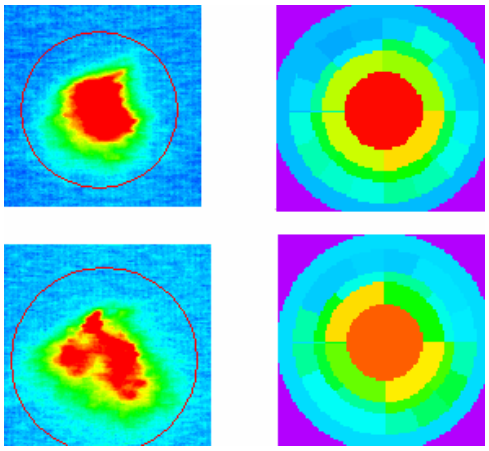
справедливо й відносно периферійних точок рівневих та розрядних зрізів зображення), що впливає на форму спектральних ліній. Тому в цій крайовій області доцільно усереднювати інтенсивність по всьому кільцю (рис. 3.6).

Рисунок 3.6 – Схема сегментації плямоподібного зображення профілю лазерного променя

Для визначення номера зони, в яку попадає піксель, що обробляється, спочатку будемо визначати радіус-вектор точки відносно системи координат з центром в точці знайденого енергетичного центру. Довжина радіус-вектора визначатиме номер зони-кільця. Далі розглядаємо точку в полярних координатах і знаходимо кут φ- між полярною віссю та радіус-вектором і приводимо його значення в додатну площину. Знаючи номер зони-кільця і полярний кут φ визначаємо фактичний номер зони. По кожній зоні усереднюється інтенсивність кольору.

В результаті виконання всіх етапів попередньої обробки отримуємо зображення у компактному вигляді, прийнятному для подачі на входи нейромережевої структури (рис. 3.7). Для опису плямоподібного зображення в

даному випадку потрібно 30 байт, так як його визначає 1-байтна дійсна величина середньої інтенсивності кожної з 30 зон.



а) б)

Рисунок 3.7 – Приклади плямоподібних зображень із різним ступенем спотворення: а) вихідні зображення; б) зображення після попередньої обробки

Таким чином, застосування попереднього інтелектуального оброблення плямоподібного зображення забезпечує економне використання ресурсів пам'яті комп'ютерної системи та спрощення структури нейронної мережі та, відповідно, підвищення швидкодії її роботи. Також, слід зазначити, що після попереднього оброблення ущільнене зображення неможливо відновити до первісного вигляду і ця властивість у перспективі може бути використана для захисту інформації в системі.

На другому етапі, для вирішення поставленої задачі нейромережевої класифікації застосовано багатошаровий персептрон, на основі якого зручно реалізовувати алгоритм навчання нейронної мережі зі вчителем. Дана нейронна мережа навчається на основі еталонних зображень за вдосконаленим ітеративним алгоритмом зворотного поширення похибки (Back Propagation). У випадку диференційованих функцій активації, які і доцільно обирати, визначення похідних за довільною вагою мережі задається ланцюговим правилом диференціювання. Метод зворотного розповсюдження похибки ефективно

використовує це правило. Обрано активаційну функцію – гіперболічний тангенс (thx). Функція і її похідна неперервні, похідна виражається через саму функцією. Гіперболічний тангенс симетричний відносно точки (0,0), це його перевага порівняно, наприклад, з сигмоїдою.

Нейронна мережа містить 30 входів, на які надходить інформація про середню інтенсивність кольору з 30 сегментів зображення (рис. 3.8).

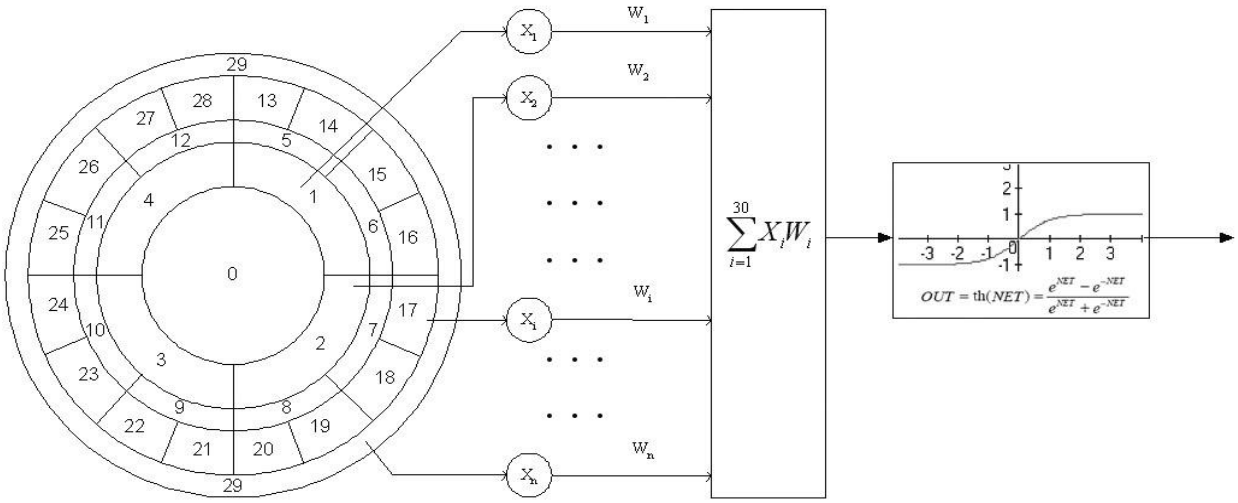


Рисунок 3.8 – Узагальнена схема процесу нейромережевої класифікації плямоподібних зображень лазерного променя

В процесі навчання нейронною мережею на кожній ітерації обирається один з елементів навчальної вибірки і ваги нейронів налаштовуються так, щоб сумарна помилка розпізнавання, тобто сума помилок розпізнавання по всім елементам вибірки, зменшилась. Принципово важливим є спосіб вибору наступного елемента навчальної вибірки для налаштування ваг нейронів. Наприклад, якщо вибирати наступний елемент по порядку, припустивши фіксований порядок елементів навчальної вибірки, то навчання мережі виявиться неефективним, тому що кожен елемент вибірки має різну складність для нейронної мережі.

Для оптимізації навчання нейронної мережі наступний елемент обиратиметься за евристичним правилом. На кожній ітерації похибки

розпізнавання кожного елемента вибірки змінюються, тому для покращення і прискорення процесу навчання пропонується обирати частіше ті елементи, похибка яких більша порівняно з похибкою інших елементів. Таким чином, ймовірність вибору конкретного елемента навчальної вибірки дорівнюватиме відношенню похибки розпізнавання даного елемента до поточної суми похибок розпізнавання всіх елементів вибірки: $p_i = \frac{E_i}{\sum_{i=1}^{30} E_i}$.

Для того, щоб мати більш повну інформацію про розпізнавання зображення нейронною мережею, ніж приналежність до певного класу, введемо поняття „коефіцієнта впевненості" класифікації нейронної мережі. Так як задача класифікації «плямових» зображень лазерного променя передбачає два класи і відповідно два еталонні виходи мережі, що приймають значення в діапазоні [-1;1]. Вихід мережі (1;-1) відповідає зображенню із невеликим ступенем спотворення або ж неспотворене зображення (профіль у межах норми), а вихід (-1;1) – великий ступінь спотворення, тобто спотворене зображення (профіль поза межами норми). При розпізнаванні конкретного зображення на виході формується значення не рівне еталонному виходу. Введемо функціонал похибки нейронної мережі при розпізнаванні зображення лазерного променя, вираз якого в загальному вигляді [32, 39]:

$$E = \frac{1}{4}\sum_{i=1}^{n}(Out_i - Out(k)_i)^2 \qquad (3.13)$$

де Out_i – значення *i*-го виходу мережі при розпізнаванні зображення; $Out(k)_i$ – значення *i*-го еталонного виходу мережі, що відповідає розпізнаному класу k; n – кількість виходів НМ (рівна кількості класів, в даному випадку n=2).

Вказана похибка приймає значення у діапазоні [0, n-1]. Таким чином, максимальне значення похибки $E_{\max} = n - 1 = 1$ і відповідає виходу мережі (1;1). Тоді коефіцієнт впевненості класифікації буде: $C = 1 - E/E_{\max}$.

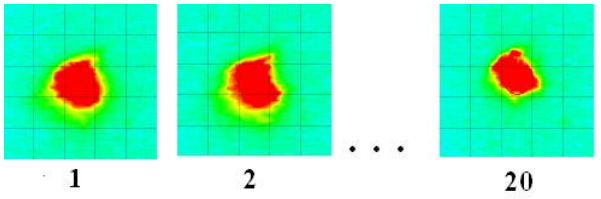


Рисунок 3.9 – Зображення навчальної вибірки із невеликим ступенем спотворення

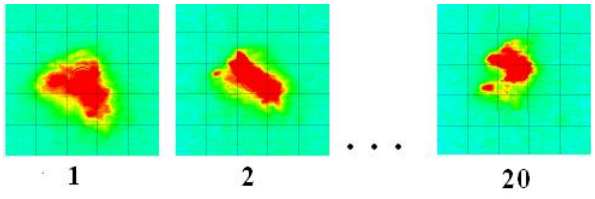


Рисунок 3.10 – Зображення навчальної вибірки із великим ступенем спотворення

На рис. 3.9 та рис. 3.10 наведено приклади зображень профілю лазерного променя із різними ступенями спотворення.

3.3 Програмна реалізація інформаційно-вимірювальної системи для оброблення профілю лазерного променя з класифікацією плямоподібних зображень

3.3.1 Обґрунтування вибору мови програмування

Мовою програмування була обрана об’єктно орієнтована мова високого рівня Java. Однієї з перших і головних переваг даної мови програмування є її платформонезалежність. Загалом, першочергово дана мова програмування створювалась для керування побутовими пристроями і повинна була розпоряджатись невеликими ресурсами. Тому першочергово мова програмування відповідала таким важливим критеріям як [43]:

– зручність використання;
– економія затрат на вивчення технології;
– зручність використання.

Від того часу багато змінилось, але багато яким критеріям Java і досі відповідає, що є її великою перевагою.

На сьогодні можемо відзначити такі переваги Java як [43]:

– орієнтованість на кросплатформність, що в першу чергу є зручним з точки зору можливості перенесення програми, написаній на даній мові програмування. Таким чином, один і той же код можна запускати під керуванням операційних систем Windows, Linux, FreeBSD, Apple Mac, тощо.

– потужні інструменти для роботи з усіма типами об’єктів, необхідних для правильного функціонування реалізуємого програмного забезпечення;

– великий попит на програмні продукти, реалізовані за допомогою саме даної мови програмування;

– можливість автоматизованої побудови діаграми класів розроблюваного проєкту;

Разом із тим, слід виділити такі недоліки [43]:

– не досить висока швидкодія та ефективність виконання програмного коду;

– необхідність попередньої підготовки у вивченні основних класів та структур мови. Синтаксис мови особливо багатий, тому потребує певних зусиль для його освоєння.

3.3.2 Розробка алгоритму головного програмного модуля

Сформулюємо загальну функціональність програмного продукту та перейдемо безпосередньо до розробки схеми алгоритму навчання, що стане основою програмного модуля реалізації нейронної мережі. Отже, програма буде сприймати плямове зображення профілю лазерного променя і намагатиметься класифікувати його відповідно до знань, які вона матиме про задані класи зображень внаслідок проходження циклів навчання. Окремий модуль відповідатиме за навчання нейронної мережі.

Вдалий вибір функції активації може зменшити час навчання в кілька раз. Нейронна мережа матиме функцію активації такого вигляду:

$$OUT = \text{th}(NET) = \frac{e^{NET} - e^{-NET}}{e^{NET} + e^{-NET}} \tag{3.14}$$

Функція симетрична відносно точки (0,0), це перевага порівняно з сигмоїдою. Графік функції активації зображено на рис. 3.11. Похідна функції також неперервна і виражається через саму функцію.

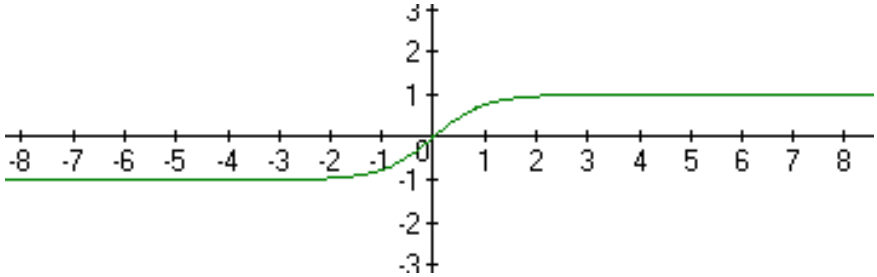


Рисунок 3.11 – Графік функції активації

Основою програми буде алгоритм навчання нейронної мережі, його схема представлена на рисунку 3.12.

Навчання нейронної мережі здійснюється в окремому процесі, який створюється за допомогою класу `public class LearningThread extends Thread`. Об’єкт цього класу має доступ до всіх змінних об’єкту класу `NeuralNet`. Функція `void startLearning(Vector data)` створює об'єкт класу `LearningThread` і запускає окремий процес, де виконується цикл навчання мережі. Вектор `data` є навчальною вибіркою і являє собою колекцію масивів типу `double`, в яких послідовно записані спочатку вхід мережі, а потім еталонний вихід. Щоб зупинити процес навчання, є функція `void stopLearning()`.

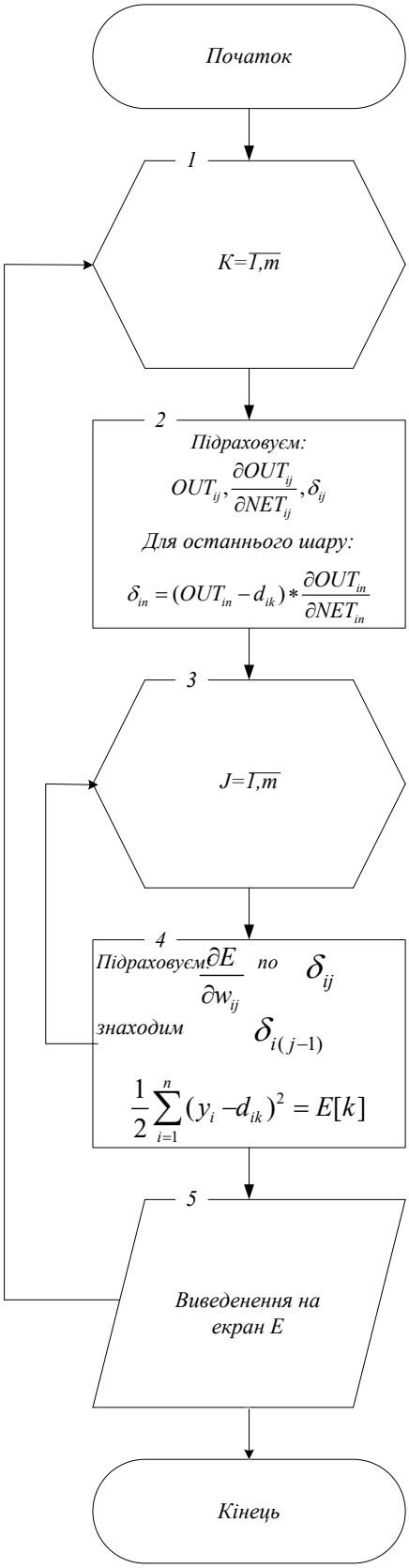


Рисунок 3.12 – Схема алгоритму навчання нейронної мережі

У процесі навчання використовується такий параметр як швидкість навчання. У алгоритмі зворотного розповсюдження помилки цей параметр позначений за ε і може приймати значення від 0 до 1. За допомогою функції `void setLearningSpeed(double)` задається цей параметр. За замовчуванням його значення є 0,1.

3.3.3 Тестовий приклад роботи та аналіз результатів

За результатами комп'ютерного моделювання (91,4 % коректно розпізнаних зображень тестової вибірки) на мові програмування Java здійснено програмну реалізацію інформаційно-вимірювальної системи для оброблення профілю лазерного променя з класифікацією плямоподібних зображень.

Комп'ютерна програма призначена для класифікації плямоподібних зображень у задачах попередньої обробки характеристик профілю лазерного променя. Для роботи комп'ютерної програми мінімально необхідно – наявність встановленої на комп'ютері операційної системи MS Windows; наявність JRE 6 або вище; тактова частота процесора не менше 1.8 GHz; наявність оперативної пам'яті обсягом не менше ніж 512 Mb. Файл програми займає біля 1 Mb дискового простору (разом з тим, JRE 6 займатиме орієнтовно 200 Mb). Для роботи з програмою необхідно запустити виконуваний JAR-файл (JAR-архів).

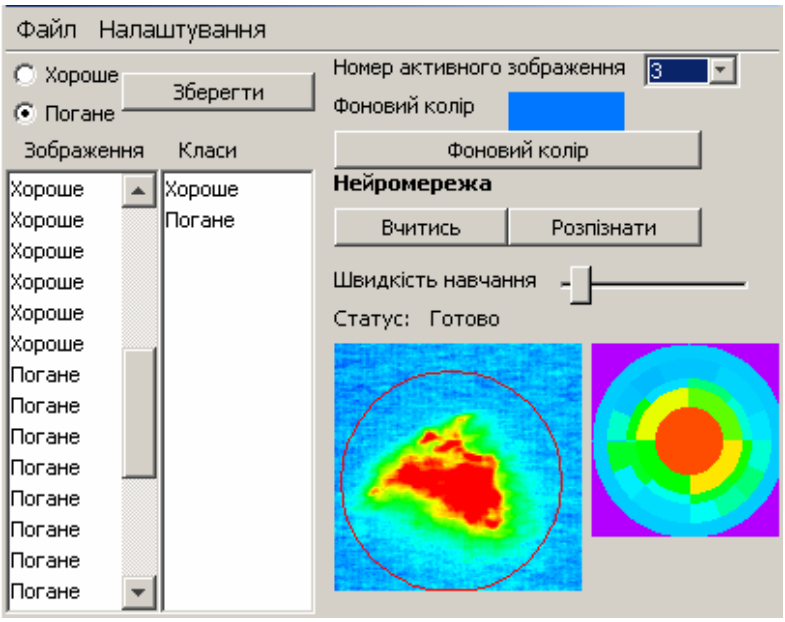


Рисунок 3.13 – Екранна форма головного програмного модуля

Програма реалізує функції: попередня обробка зображення, що базується на виділенні інформативної області зображення, топологічний аналіз отриманої робочої області та усередненні інтенсивності кольору по кожній зоні при сегментації зображення, а це в свою чергу обумовлює ущільнення оброблюваних даних та надає переваги для подальшої нейромережевої обробки та аналізу. На вхід системи можна подавати 8-бітні зображення розмірністю 128x128 пікселів лазерного відеоряду у форматі „BMP”.

Тестування проводилось за допомогою наявних наборів даних. Фрагмент бази даних одного відеоряду лазерних зображень показано на рис. 3.14.

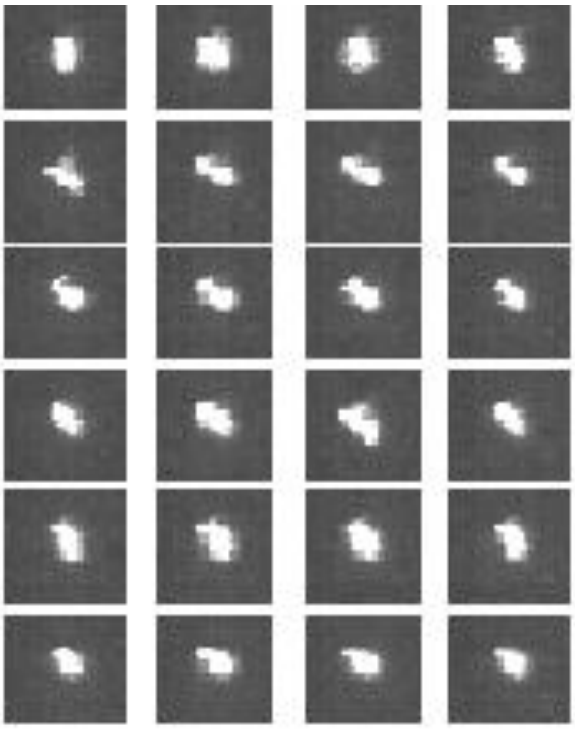


Рисунок 3.14 – Фрагмент відеоряду зображень профілю лазерного променя

Робота з програмою відбувається в межах таких основних етапів: завантаження навчальної вибірки плямових зображень (одиничний BMP-файл або обрана директорія, або з XML-файлу), збереження плямових зображень в XML-файлі – у такому вигляді для подання в нейронну мережу, вибір та встановлення фонового кольору, встановлення параметрів сегментації, задання параметрів для навчання нейронної мережі, навчання нейронної мережі, аналіз та класифікація зображень, відображення результатів класифікації зображень.

Комп’ютерну програму реалізовано в інтегрованому середовищі розробки NetBeans IDE на об’єктно-орієнтованій мові програмування Java. Java програми компілюються в байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Системні вимоги до програми в основному визначаються вимогами до виконавчого середовища Java Runtime Environment (JRE), яке має бути встановленим на комп’ютері для роботи з програмою.

Таблиця 3.1 – Результати тестування програмної реалізації нейромережевої класифікації плямоподібних зображень профілю лазерного променя

Параметр	**+**	**–**	**Всього**	**Параметр**	**+**	**–**	**Всього**
Вибірка 1	20	20	40	Вибірка 2	100	100	200
Класифіковано	19	18	37	Класифіковано	92	87	179
У відсотках, %	95	90	92,5	У відсотках, %	92	87	89,5

де "+" – клас зображень із невеликим ступенем спотворення ("хороші"), "–" - клас зображень із великим ступенем спотворення ("погані"), "Всього" – всього плямоподібних зображень.

Загалом, тестування програмної реалізації інформаційно-вимірювальної системи для оброблення профілю лазерного променя з класифікацією плямоподібних зображень по всій довжині відеоряду лазерних зображень (2044 зображення) в реальному часі показало результати, відповідно: 76% коректно класифікованих “хороших” зображень (із невеликим ступенем спотворення) та 64% – “поганих” (із великим ступенем спотворення). Результати тестування свідчать про працездатність та адекватність обраного підходу.

3.4 Висновки до розділу 3

Наведено аналіз та опис основних етапів процесу вимірювання динамічного профілю лазерного променя. Описано принцип роботи інформаційно-вимірювальної системи для нейромережевої класифікації плямоподібних зображень при обробленні профілю лазерного променя.

Розроблено узагальнену структурно-функціональну модель інформаційно-вимірювальної системи для оброблення профілю лазерного променя та описано принцип її роботи. Акцентовано увагу, що оброблення плямоподібних зображень відбувається в межах двох основних етапів: попередньої інтелектуальної обробки зображення та нейромережевої класифікації.

Наведено опис розробленого апаратного макета інформаційно-вимірювальної системи для оброблення профілю лазерного променя з класифікацією плямоподібних зображень. Відзначено, що при проведенні досліджень використовувалась потужність лазера 0,04 – 0,07 мВт, роздільна здатність відеокамери – 640×480 пікселів при 30 кадрах за секунду. Для наступного етапу оброблення відеофайл було розділено на 8-бітні зображення розмірністю 128×128 пікселів.

Обґрунтовано вибір мови програмування Java для програмної реалізації інформаційно-вимірювальної системи оброблення профілю лазерного променя з класифікацією плямоподібних зображень. Описано окремі особливості програмної реалізації та функціонування програмного коду. Розроблено та програмно реалізовано схему алгоритму навчання нейронної мережі – багатошарового персептрону на основі навчання зі вчителем за методом зворотного поширення похибки (Back Propagation). Здійснено тестування програмної реалізації інформаційно-вимірювальної системи для оброблення профілю лазерного променя з класифікацією плямоподібних зображень. Результати тестування свідчать про працездатність та адекватність обраного підходу.

ВИСНОВКИ

В ході даних досліджень відзначено актуальність задачі вимірювання динамічного профілю лазерного променя для різноманітних практично-прикладних застосувань. Здійснено аналіз предметної області оброблення профілю лазерного променя. Відзначено, що на даний час у сферах лазерної обробки матеріалів, лазерній локації, оптичному зв'язку, поліграфії й інших областях техніки відчувається гостра необхідність більш широкого впровадження оптико-електронних інтелектуальних систем з автоматичним коригуванням похибок формованого світлового випромінювання.

Здійснено класифікаційний аналіз методів оброблення та розпізнавання плямоподібних зображень, а також нейронних мереж та методів їх навчання. Здійснено комп’ютерне моделювання нейромережевого оброблення профілю лазерного променя з класифікацією плямоподібних зображень, що дозволило експериментально дослідити перспективи застосування різних типів нейронних мереж та методів їх навчання для вирішення задачі нейромережевої класифікації плямоподібних зображень.

Здійснено аналіз процесу вимірювання динамічного профілю лазерного променя. Описано принцип роботи інформаційно-вимірювальної системи для нейромережевої класифікації плямоподібних зображень при обробленні профілю лазерного променя. Розроблено узагальнену структурно-функціональну модель інформаційно-вимірювальної системи для оброблення профілю лазерного променя та наведено основні результати досліджень.

Наведено опис розробленого апаратного макета інформаційно-вимірювальної системи для оброблення профілю лазерного променя з класифікацією плямоподібних зображень.

Обґрунтовано вибір мови програмування Java для програмної реалізації інформаційно-вимірювальної системи оброблення профілю лазерного променя з

класифікацією плямоподібних зображень. Описано окремі особливості програмної реалізації та функціонування програмного коду. Розроблено та програмно реалізовано схему алгоритму навчання нейронної мережі – багатошарового персептрону на основі навчання зі вчителем за методом зворотного поширення похибки (Back Propagation).

Тестування інформаційно-вимірювальної системи для оброблення профілю лазерного променя з нейромережевою класифікацією плямоподібних зображень по всій довжині відеоряду лазерних зображень (2044 зображення) в реальному часі показало 76% коректно класифікованих “хороших” зображень (із невеликим ступенем спотворення), що свідчить про працездатність та адекватність обраного підходу. В результаті усі поставлені задачі виконано та досягнуто мету дослідження, а саме здійснено розширення функціональних можливостей інформаційно-вимірювальної системи для оброблення профілю лазерного променя шляхом застосування нейромережевої класифікації плямоподібних зображень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Заболотна Н. І., Ярова О. А. Інформаційно-вимірювальна система для оброблення зображень профілю лазерного променя : Збірник матеріалів Міжнародної науково-практичної конференції "Молодь в науці: дослідження, проблеми, перспективи (МН-2025)". В.: ВНТУ, 2025. С. 1-7. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/24739/20485
2. Пупань Л.І. Лазерні технології у машинобудуванні : навч. посіб. Харків: НТУ «ХПІ», 2020. 109 с.
3. Roundy, C.B. & Kirkham, K.D. Current technology of laser beam profile measurements. 2024. 463-524. 10.1201/b17140-13.
4. Pankaj J. An Introduction to Astronomy and Astrophysics. Boca Raton : CRC Press, 2015. 365 p.
5. Заболотна Н.І., Шолота В.В., Колівошко А.І. Аналіз методів та систем лазерної поляриметрії для відтворення анізотропних параметрів біологічних шарів. *Опт-ел. інф-енерг. техн*. 2019. №36, вип. 2. С. 60-71.
6. Тимчик Г.С., Богатирьова Г.В., Мамута М.С. Лазерні технології. Практикум. Навчальний посібник. Електронне мережне навчальне видання. Київ. КПІ ім. Ігоря Сікорського. 2022. 124 с.
7. Марков С.М., Ярова О.А. Лабораторний стенд для дослідження енергоефективності світлодіодів малої потужності для систем освітлення приміщень. – LIII Всеукраїнська науково-технічна конференція підрозділів ВНТУ, Ukraine, March. 2024. Available at: https://conferences.vntu.edu.ua/index.php/all-frtzp/all-frtzp-2024/paper/view/20548/16995
8. Тимчик Г. С., Сорока С. О., Філіппова М. В. Лазерні технології. Лабораторний практикум. Навчальний посібник. Київ. КПІ ім. Ігоря Сікорського. 2022. 86 с.

9. Висока потужність та інтелектуальні тенденції машин для лазерного різання. 2023. [Електронний ресурс] – Тип доступу: https://ua.hglasermachine.com/info/the-high-power-and-intelligent-trends-of-laser-81957683.html
10. Комп'ютерне моделювання систем та процесів. Методи обчислень. Частина 2 : навчальний посібник / Р. Н. Квєтний, І. В. Богач, О. Р. Бойко та ін.; за заг. ред. Р. Н. Квєтного. Вінниця : ВНТУ, 2013. 191 с.
11. Романюк О. Н., Найдюк В. І. Використання нейронних мереж для обробки та розпізнавання зображень. Тези доповідей XII Міжнародної науково-технічної конференції «Інформаційно-комп’ютерні технології – 2021 (ІКТ-2021)», м. Житомир, 01 - 03 квітня 2021 р. Житомир : Житомирська політехніка, 2021. С. 72-73.
12. Леонов М. С. Вирішення задач обробки зображень за допомогою згорткових нейронних мереж. *Теоретичні та прикладні аспекти побудови програмних систем* : праці 15 міжнародної науково-практичної конференції, Київ, 23-24 грудня 2024 р. / [за заг. ред.: М. М. Глибовця, Т. В. Панченка та ін. ; Факультет інформатики Національного університету "Києво-Могилянська академія" та ін.]. Київ : НаУКМА, 2024. С. 48-49.
13. Совин Я.Р., Пархуць Л.Т., Ракобовчук Л.М. Вплив атмосфери на поширення лазерного випромінювання [Електронний ресурс]. *Сучасний захист інформації.* 2024. № 1. С. 108-113.
14. Поплавський О.А., Поплавська А.А., Коротун І.А., Особливості організації передачі інформації лазером через атмосферу для розробки методів та програмно-апаратних засобів прогнозування характеристик зображень сигналу. *Опт-ел. інф-енерг. техн*. 2014. вип. 27, вип. 1. С. 206–209,
15. FMCW Лазерне вимірювання відстані. 2023. [Електронний ресурс] – Тип доступу: https://ua.mrj-marking.com/news/fmcw-laser-distance-measuring-speed-measuring-72615775.html

16. Coherent's Beam Profilers. [Електронний ресурс] – Тип доступу: https://www.coherent.com/search-results?query=profilers
17. Промінь Гауса проти пучка циліндрів: у чому різниця? [Електронний ресурс] – Тип доступу: https://hantencnc.com/uk/blog/gaussian-beam-vs-top-hat-beam-whats-the-difference
18. Чим гаусівський і плоский пучки відрізняються за розподілом енергії та ефективністю для лазерних застосувань? [Електронний ресурс] – Тип доступу: http://ua.oem-laser.com/info/how-do-gaussian-and-flat-top-beams-differ-in-e-96684974.html (дата звернення 12.06.2025).
19. Розуміння одномодового та багатомодового лазерного очищення. [Електронний ресурс] – Тип доступу: https://ua.mrj-marking.com/news/understanding-single-mode-and-multimode-in-las-82796659.html (дата звернення 12.06.2025).
20. Rüdiger Paschotta Beam Profilers. [Електронний ресурс] – Тип доступу: https://www.rp-photonics.com/beam_profilers.html (дата звернення 12.06.2025).
21. Laser Beam Analisys. [Електронний ресурс] – Тип доступу: https://msmacrosystem.nl/3d_laser_beam/laser_beam_profiler.html
22. Laser beam diagnostics. [Електронний ресурс] – Тип доступу: https://www.gentec-eo.com/laser-beam-diagnostics (дата звернення 12.06.2025).
23. Laser beam characterization. [Електронний ресурс] – Тип доступу: https://www.axiomoptics.com/blog/laser-beam-characterization/
24. Digital Laser Beam Analyzer with 3D/2D Software. [Електронний ресурс] – Тип доступу: http://www.sciencegl.com/3d_laser_beam/laser_beam_profiler.html
25. Laser Applications. LightMachinery. [Електронний ресурс] – Тип доступу: https://lightmachinery.com/lasers/laser-applications/
26. Lasers and laser systems for medicine. Інститут фізики напівпровідників імені В.Є. Лашкарьова НАНУ. 2025. [Електронний ресурс] – Тип доступу: https://isp.kiev.ua/index.php/uk/2014-09-23-11-47-26/5380-070

27. Заболотна Н. І. Застосування лазерів у діагностиці біологічних об`єктів. Лазер і здоров`я. Серія «Фізична і біомедична електроніка» : монографія / за заг. ред. А. В. Кіпенського. Харків : Комунальне підприємство «Міська друкарня», 2024. Розд. 6. С. 172–216.
28. Заболотна Н. І., Шолота В. В. Система двохвильової лазерної діагностики молочних залоз за поляризаційним картографуванням зображень плівок плазми крові. Інформаційні технології та комп`ютерна інженерія. 2024. № 1. С. 146-156.
29. Pavlov S. V., Zabolotna N. I., Shtofel D. Kh., Yang L., Komarova O. S., Kaduk O. V. Realization of a laser fiber-optical device for assessing tissue microcirculation. Оптико-електронні інформаційно-енергетичні технології. 2024. № 2 (48). С. 205–211.
30. Лазерна корекція методом ФРК. [Електронний ресурс] – Тип доступу: https://zrenie.dp.ua/uk/services/lazerna-korektsiia-metodom-frk
31. Теорія та практика нейронних мереж : навч. посіб. / Л. М. Добровська, І. А. Добровська. К. : НТУУ «КПІ» Вид-во «Політехніка», 2015. 396 с.
32. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. Житомир: Вид. О. О. Євенок, 2020. 184 с.
33. Кобилін О. А., Творошенко І. С. Методи цифрової обробки зображень : навч. посіб. ; Харків : ХНУРЕ, 2021. 124 с.
34. Вовк С.М., Гнатушенко В.В., Бондаренко М.В..Методи обробки зображень та комп’ютерний зір : навч. посіб. Д. : ЛІРА, 2016. 148 с.
35. Кутковецький В. Я. Розпізнавання образів : навч. посі. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2017. 420 с.
36. Основи інтелектуальних технологій. Частина 1. Технології розпізнавання : електронний навчальний посібник / Биков М. М., Ковтун В. В., Гаврилюк В. О. Вінниця : ВНТУ, 2023. 229 с.

37. Барабан М.В., Барабан С.В., Гармаш В.В. Розробка прогресивного веб-додатку зі згортковою нейронною мережою для розпізнавання зображень. *Інформаційні технології та комп`ютерна інженерія*. 2021. №1. С. 7-14.
38. Основи інтелектуальних технологій. Частина 2. Технології машинного навчання : електронний навчальний посібник / Биков М. М., Ковтун В. В., Грищук Т. В. Вінниця : ВНТУ, 2024. 153 с.
39. Multilayer Perceptron. [Електронний ресурс] – Тип доступу: https://itwiki.dev/data-science/ml-reference/ml-glossary/multilayer-perceptron
40. Використовуємо CNN для обробки зображень. Частина перша. [Електронний ресурс] – Тип доступу: https://dou.ua/forums/topic/48368/
41. Використовуємо CNN для обробки зображень. Частина друга. [Електронний ресурс] – Тип доступу: https://dou.ua/forums/topic/48429/?from=similar_posts_tech
42. Tymchenko, L., Petrovskiy, M., Kokryatskaya, N. et al. Modeling of the high-performance PLD-based sectioning method for classification of the shape of optical object images. SpringerPlus 2, 692 (2013). https://doi.org/10.1186/2193-1801-2-692
43. Васильєв О.М. Програмування мовою Java. Тернопіль : Навчальна книга Богдан, 2021. 696 с.

Додаток А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Інформаційно-вимірювальна система для оброблення профілю лазерного променя з класифікацією плямоподібних зображень

Тип роботи: бакалаврська кваліфікаційна робота

(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра БМІОЕС, факультет ІЕС, КОІС- 21б

(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 29,62 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Коваль Л.Г., зав. каф. БМІОЕС _____ (підпис)
(прізвище, ініціали, посада)

Заболотна Н.І., проф. каф. БМІОЕС _____ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку _____ (підпис) Тужанський С.Є.
(прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Заболотна Н.І., проф. каф. БМІОЕС
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Ярова О.А.

Додаток Б

(обов’язковий)

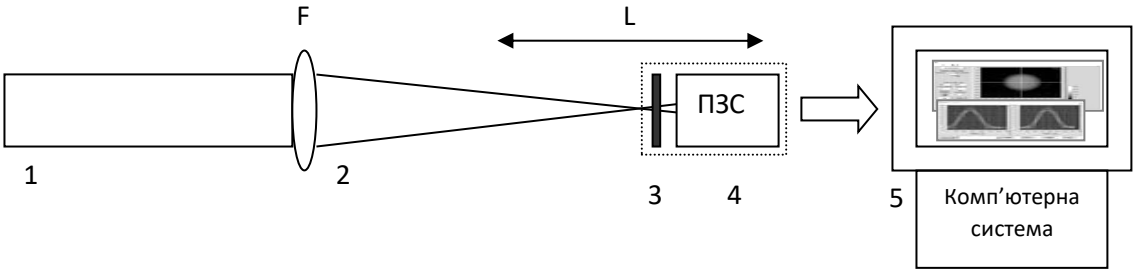
ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНА СИСТЕМА ДЛЯ ОБРОБЛЕННЯ ПРОФІЛЮ ЛАЗЕРНОГО ПРОМЕНЯ З КЛАСИФІКАЦІЄЮ ПЛЯМОПОДІБНИХ ЗОБРАЖЕНЬ

Додаток Б.1

(обов’язковий)

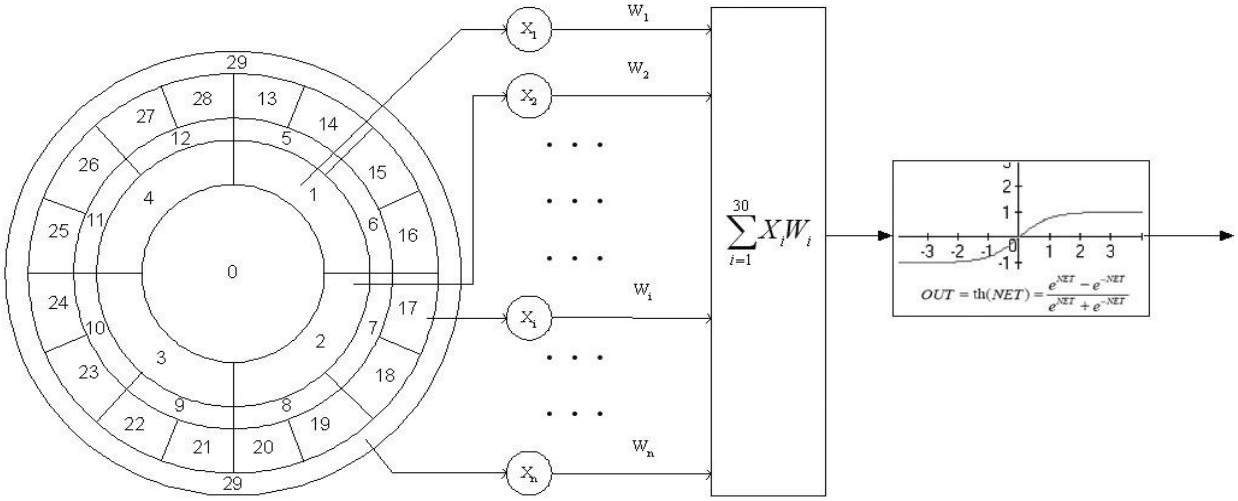
Схема вимірювання профілю лазерного променя для напівпровідникової лазерної системи



Додаток Б.2

(обов'язковий)

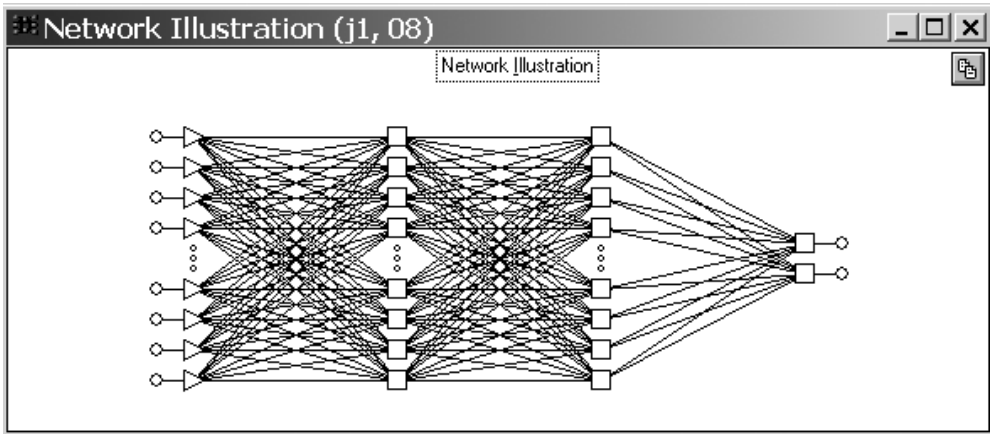
Узагальнена схема процесу нейромережевої класифікації плямоподібних зображень лазерного променя



Додаток Б.3

(обов'язковий)

Структурна організація обраної для моделювання НМ



Додаток Б.4

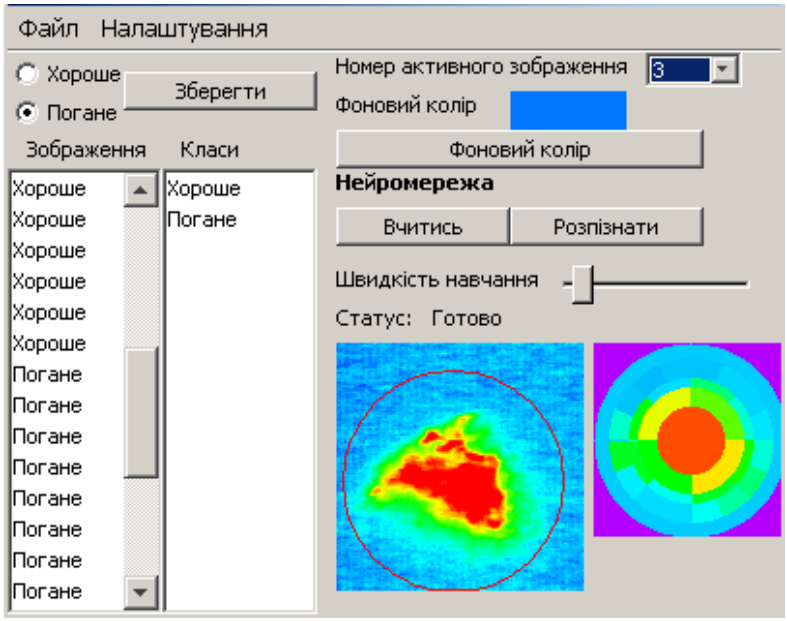
(обов'язковий)

Апаратний макет інформаційно-вимірювальної системи для оброблення профілю лазерного променя з класифікацією плямоподібних зображень

Додаток Б.5

(обов'язковий)

Екранна форма головного програмного модуля системи



Додаток В

(обов'язковий)

Лістинг програми

NeuralNet.java

```
/*Клас NeuralNet створює нейронну мережу, налаштовує її та виконує навчання нейронної мережі*/
package NeuralNet;

public class NeuralNet {
   private int numSensors; // кількість сенсорів (розмірність нульового шару)
   private Vector layers; // вектор (колекція) вагів для всіх шарів
   private Vector s; // колекція похідних активац.функції для кажного нейрону кожного шару
   private Vector y;// колекція виходів нейронів для кожного шару
   private LearningThread learningThread;
   private double learningSpeed;
   private double[] maxI;
   private double[] minI;
   private double[] maxO;
   private double[] minO;
   // Змінні опцій зупинки процесу навчання
   private boolean blLimitedIterations = false;
   private int maxIterations = 0;
   private boolean blMinError = false;
   private double minError = 0;

   /*Даний клас реалізує окремий процес для навчання мережі*/
   public class LearningThread extends Thread{
      private Vector data;
      private boolean run;
      LearningThread(){
         super();
         run = false;
      }
      /*Головний метод навчання нейромережі*/
      public void run(){
         if (data.size() == 0) return;
         run = true;
         double out[]; //вихід нейромережі
         double sensors[] = new double[numSensors]; // вхід нейромережі
         double dataEx[]; // масив, що буде містити один єлемент навчальної вибірки
         int k = 0; // номер елементу навчальної вибірки, що використовується для наступної епохи навчання
         /*Змінна, що містить сумарну похибку разпізнавання. На початку має достатньо велике значення рівне
кількості елементів навчальної вибірки*/
         double totalE = data.size();
         double[] E = new double[data.size()]; //масив похибок розпізнавання для кожного елементу вибірки
         for(int i = 0; i < data.size(); i++) E[i] = 1;
         double[] sEx;
         double[][] w; //масив ваг
         double[] newDelta = new double[0];
         int iteration = 0;

         while(run && (!blLimitedIterations || (iteration <= maxIterations)) && (!blMinError || (totalE > minError))){
            iteration ++;
            try{
               dataEx = (double[])data.get(k); // дістаєм елемент номер k з навчальної вибірки
```

```
            for(int i = 0; i < numSensors; i++) sensors[i] = dataEx[i]; // виокремлюємо вхід мережі для елементу
навчальної вибірки
            out = calculateWithoutNormalization(sensors); // підраховуємо вихід мережі для заданого входу
            // після виконання цієї функції маємо вектор S з похідними актив. функції для кожного шару
            double delta[] = new double[getNumLayerNeurons(layers.size())];
            sEx = (double[])s.get(layers.size() - 1); // маємо похідні активац.функцій останнього шару мережі
            // підраховуємо delta для останнього шару
            for(int i = 0; i < getNumLayerNeurons(layers.size()); i++){
               delta[i] = (out[i] - dataEx[i + numSensors]) * sEx[i];
            }
 /*Перераховуємо ваги нейронів для єлементу k навчальної вибірки, коефіцієнт нормалізації 0.25 дає наступне:
            якщо мережа розпізнає образ коректно, то E ~ 0
            якщо мережа не розпізнає образ, то       E ~ 1*/
               E[k] = 0;
               for(int i = 0; i < out.length; i++) E[k] += 0.25*(out[i] - dataEx[i + numSensors])*(out[i] - dataEx[i +
numSensors]);
               // цикл по всім шарам від останнього до першого, що перераховує ваги входів нейронів
               // згідно підрахованим параметрам delta. З самого початку маємо delta для останнього шару
               for(int j = layers.size() - 1; j >= 0; j--){
                     w = (double[][])layers.get(j); // масив вагів j-го шару
                    // підраховуємо параметри delta для попереднього шару
                    if(j > 0){
                    // якщо масив newDelta замалий, то збільшуємо його
                    if (newDelta.length < getNumLayerNeurons(j)) newDelta = new double[getNumLayerNeurons(j)];
                    // наступний цикл підраховує параметри delta для попереднього шару
                    for(int l = 0; l < getNumLayerNeurons(j); l++){
                       double a = 0;
                       for(int m = 0; m < w[0].length; m++) a += delta[m] * w[l + 1][m];
                       newDelta[l] = a * ((double[])s.get(j - 1))[l];
                    }
                 }
                 /*Перераховуємо ваги входу нейронів шару j по заданим параметрам delta для цього шару*/
                 for(int l = 0; l < w[0].length; l++){
                    for(int m = 0; m < w.length - 1; m++){
                       w[m + 1][l] -= learningSpeed * delta[l] * ((j == 0) ? sensors[m] : ((double[])y.get(j - 1))[m]);
                    }
                    w[0][l] -= learningSpeed * delta[l];
                 }
                 delta = newDelta;
               }
               /*Підраховуємо сумарну помилку розпізнавання*/
               totalE = 0;
               for(int i = 0; i < E.length; i++) totalE += E[i];
               double r = Math.random() * totalE;
               k = -1;
               while (r > 0){
                  k++;
                  r -= E[k];
               }
               outputObj.setText("Помилка = " + java.lang.String.valueOf(totalE));
            } catch(WrongParameterException ex) {
               run = false;
            } catch(Exception ex) {
               run = false;
            }
         }
         outputObj.setText("Навчання зупинено");
      }
```

```
        /*Метод, що задає навчальну вибірку*/
        public void setParams(Vector data){
            this.data = data;
            for(int i = 0; i < data.size(); i++){
            double[] d = (double[])data.get(i);
            for(int j = 0; j < numSensors; j++) d[j] = (2 * d[j] - minI[j] - maxI[j])/(maxI[j] - minI[j]);
            for(int j = numSensors; j < d.length; j++) d[j] = (2 * d[j] - minO[j - numSensors] - maxO[j -
numSensors])/(maxO[j - numSensors] - minO[j - numSensors]);
            }
        }
        public void stopLearning(){
            run = false;
        }
        public boolean isLearning(){
             return run;
             }
    }
    /*Даний метод створює нейронну мережу */
    public NeuralNet(int numSensors) {
        this.numSensors = numSensors;
        layers = new Vector();
        s = new Vector();
        y = new Vector();
        learningSpeed = .1;
        minI = new double[numSensors];
        maxI = new double[numSensors];
        for(int i = 0; i < numSensors; i++){
            minI[i] = -1;
            maxI[i] = 1;
        }
    }
    /*Даний метод встановлює мінімальні та максимальні значення входів та виходів нейронної мережі*/
    public void setMinMax(double[] minI, double[] maxI, double[] minO, double[] maxO){
        this.minI = minI;
        this.maxI = maxI;
        this.minO = minO;
        this.maxO = maxO;
    }
    public double[] getMinI(){ return minI; }
    public double[] getMaxI(){ return maxI; }
    public double[] getMinO(){ return minO; }
    public double[] getMaxO(){ return maxO; }
    /*Активаційнная функція гіперболічний тангенс*/
    private double activationFunc(double x){
        return (java.lang.Math.tanh(x));
    }
    /*Підрахунок похідної активаційної функції*/
    private double derivActivFunc(double x){
        double f = java.lang.Math.tanh(x);
        return (1 - f*f);
    }
    /*Даний метод додає шар в нейромережу з count нейронів у ньому*/
    public void addLayer(int count) throws WrongParameterException{
        if (count <= 0) throw new WrongParameterException();
        int prevCount;
        if (layers.size() == 0)
            prevCount = numSensors;
        else
```

```
            prevCount = ((double[][])layers.lastElement())[0].length;

        double weights[][] = new double[prevCount + 1][count];
        for(int i = 0; i < prevCount + 1; i++)
            for(int j = 0; j < count; j++)
                weights[i][j] = 2*java.lang.Math.random() - 1;
        layers.add(weights);
    }
    /*Даний метод завершує конфігурацію мережі*/
    public void configurationComplete(){
        minO = new double[getNumLayerNeurons(layers.size())];
        maxO = new double[getNumLayerNeurons(layers.size())];
        for(int i = 0; i < minO.length; i++){
            minO[i] = -1;
            maxO[i] = 1;
        }
    }
    /*Метод повертає шар з номером numLayaer, що являє собою масив з вагами нейронів у ньому*/
    public double[][] getLayer(int numLayer) throws WrongParameterException{
        if (numLayer > layers.size()) throw new WrongParameterException();
        return (double [][])layers.get(numLayer - 1);
    }
    /*Даний метод встановлює ваги для шару з номером numLayer*/
    public void setWeights(int numLayer, double weights[][]) throws WrongParameterException{
        double curWeights[][] = getLayer(numLayer);
        if (weights.length != curWeights.length) throw new WrongParameterException();
        if (weights[0].length != curWeights[0].length) throw new WrongParameterException();

        for(int i = 0; i < weights.length; i++)
            for(int j = 0; j < weights[0].length; j++)
                curWeights[i][j] = weights[i][j];
    }
    /*Метод повертає кількість сенсорів (входів мережі)*/
    public int getSensorCount(){ return numSensors;}
    /*Метод повертає кількість виходів мережі*/
    public int getOutCount(){
        return getNumLayerNeurons(layers.size());
    }
    /* Метод підраховує вихід мережі при заданому вході без нормалізації входу за параметрами maxI та
minI*/
    public double[] calculateWithoutNormalization(double sensors[]) throws WrongParameterException{
        double data[];
        data = sensors;
        s.clear();
        s = new Vector();
        y.clear();
        y = new Vector();
        for(int i = 1; i <= layers.size(); i++){
            double weights[][] = getLayer(i);
            if (weights.length != data.length + 1) throw new WrongParameterException();
            double newData[] = new double[weights[0].length];
            double newS[] = new double[weights[0].length];

            for(int j = 0; j < weights[0].length; j++){
                double w = weights[0][j];
                for(int k = 0; k < weights.length - 1; k++)
                    w += data[k] * weights[k + 1][j];
```

```
            newData[j] = activationFunc(w);
            newS[j] = derivActivFunc(w);
        }
        data = newData;
        s.add(newS);
        y.add(newData);
    }
    return data;
}
/*Метод підраховує вихід мережі при заданому входу з нормалізацією входу за параметрами maxI та minI*/
public double[] calculate(double sensors[]) throws WrongParameterException{
    double data[];
    data = sensors;
    for(int i = 0; i < numSensors; i++) data[i] = (2 * data[i] - minI[i] - maxI[i])/(maxI[i] - minI[i]);
    s.clear();
    s = new Vector();
    y.clear();
    y = new Vector();
    for(int i = 1; i <= layers.size(); i++){
        double weights[][] = getLayer(i);
        if (weights.length != data.length + 1) throw new WrongParameterException();
        double newData[] = new double[weights[0].length];
        double newS[] = new double[weights[0].length];
        //підрахунок ваг
        for(int j = 0; j < weights[0].length; j++){
            double w = weights[0][j];
            for(int k = 0; k < weights.length - 1; k++)
                w += data[k] * weights[k + 1][j];
            newData[j] = activationFunc(w);
            newS[j] = derivActivFunc(w);
        }
        data = newData;
        s.add(newS);
        y.add(newData);
    }
    /*Власне нормалізація виходу*/
    for(int i = 0; i < data.length; i++) data[i] = 0.5*((maxO[i] - minO[i]) * data[i] - minO[i] - maxO[i]);
    return data;
}
//Метод автоматично підраховує параметри нормалізації згідно навчальної вибірки
private void calculateMaxMin(Vector data){
    if (data.size() == 0) return;
    for(int i = 0; i < numSensors; i++){
        minI[i] = ((double[])data.get(0))[i];
        maxI[i] = ((double[])data.get(0))[i];
    }
    for(int j = 1; j < data.size(); j++){
        for(int i = 0; i < numSensors; i++){
            if(maxI[i] < ((double[])data.get(j))[i]) maxI[i] = ((double[])data.get(j))[i];
            if(minI[i] > ((double[])data.get(j))[i]) minI[i] = ((double[])data.get(j))[i];
        }
    }
}
/*Метод починає процес навчання, викликаюци алгоритм у окремому процессі (Thread)*/
public void startLearning(Vector data){
    learningThread = new LearningThread();
    learningThread.setParams(data);
    learningThread.start();
```

```
    }
    /*Метод зупиняє процес навчання*/
    public void stopLearning(){
        learningThread.stopLearning();
        while(learningThread.isLearning()){};
    }
    /*Метод повертає кількість шарів мережі*/
    public int getLayersCount(){
        return layers.size();
    }
    /*Метод повертає  кількість нейронів у і-му шарі*/
    public int getNumLayerNeurons(int i){
        return ((double[][])layers.get(i - 1))[0].length;
    }
    /*Метод повертає кількість входів для і-го шару, тобто кількість нейронів у шарі і-1*/
    public int getNumPrevLayerNeurons(int i){
        return ((double[][])layers.get(i - 1)).length - 1;
    }
      }
    /*Метод встановлює опції зупинки процессу навчання мережі*/
    public void setStopLearningOption(boolean blLimitedIterations, int maxIterations, boolean blMinError, double
minError){
        //максимальна кількість ітерацій
        if (blLimitedIterations){
            this.maxIterations = maxIterations;
        }
        this.blLimitedIterations = blLimitedIterations;
        //мінімальне значення похибки
        if (blMinError){
            this.minError = minError;
        }
        this.blMinError = blMinError;
    }
 }
```

ImageDB.java

```
/*Клас ImageDB реалізує всі операції по завантаженню, створенню, зберіганню навчальної вибірки чи
зображень для розпізнавання */

package lazerrecognizer;

public class ImageDB {
    private Vector ImageZones = new Vector(); //таблиця зон зображення
    private Vector ImageNames = new Vector(); //вектор імен зображень
    public static int NUMZONES = 30; //кількість зон(сегментів)
    public static int bgColor = 0x5A5A5A; //фоновий колір, встановлення фонового кольору
    public static double r1, r2, r3, r4; //радіуси кілець

    /*Конструктор класа ImageDB, в якому встановлюються відношення радіусів кіл для траси по замовчуванню*/
    ImageDB(){
        r1 = 0.35;
        r2 = 0.55;
        r3 = 0.65;
        r4 = 0.85;
    }
    /*Даний метод виконує додавання нового зображення до бази даних зображень*/
    public int add(BufferedImage im, String name){
        double Zones[] = ExtractZones(im);
```

```
        ImageZones.add(Zones);
        ImageNames.add(name);
        return (ImageZones.size() - 1);
    }
    /*Даний метод повертає кількість образів вибірки*/
    public int count(){
        return ImageZones.size();
    }
    /*Даний метод повертає ім'я зображення*/
    public String getImageName(int pos){
        return ((String)ImageNames.get(pos));
    }
    /*Даний метод отримує на вхід номер зображення з бази, а поветрає вектор інтенсивностей кожної зони зображення*/
    public double[] getZones(int pos){
        return ((double[]) ImageZones.get(pos));
    }
    /*Даний метод отримує номер зображення і повертає зображення у вигляді, який оброблений для сприйняття нейронною мережею*/
    public BufferedImage getImage(int pos, int D){
        BufferedImage im = new BufferedImage(D, D, java.awt.image.BufferedImage.TYPE_INT_RGB);
        Graphics g = im.getGraphics();
        double Zones[] = (double[]) ImageZones.get(pos);
        for (int i = 0; i < D; i++){
            for (int j = 0; j < D; j++){
                int k = Zone(i, j, D, D);
                if (k >= 0){
                    g.setColor(new Color((float)Zones[k], (float)Zones[k], (float)Zones[k]));
                }else{
                    g.setColor(Color.BLACK);
                }
                g.fillRect(i, j, 1, 1);
            }
        }
        return im;
    }
    /*Даний метод отримує номер зображення і повертає оригінальне зображення після сегментації для сприйняття нейронною мережею*/
    public static BufferedImage getImagePreview(BufferedImage originalIm, int D){
        BufferedImage im = new BufferedImage(D, D, java.awt.image.BufferedImage.TYPE_INT_RGB);
        Graphics g = im.getGraphics();
        double Zones[] = ExtractZones(originalIm);
        for (int i = 0; i < D; i++){
            for (int j = 0; j < D; j++){
                int k = Zone(i, j, D, D);
                if (k >= 0){
                    g.setColor(new Color((float)Zones[k], (float)Zones[k], (float)Zones[k]));
                }else{
                    g.setColor(Color.BLACK);
                }
                g.fillRect(i, j, 1, 1);
            }
        }
        return im;
    }
   /*Даний метод повертає вектор усереднених по зонам кольорів зображення im*/
    public static double[] ExtractZones(BufferedImage im){
        BufferedImage imSmpl;
```

```
    imSmpl = CropImage(im);
    int W = imSmpl.getWidth();
    int H = imSmpl.getHeight();
    int i, j;
    double Zones[] = new double[NUMZONES]; //масив усереднених кольорів зон
    int[] ZonesNum = new int[NUMZONES]; //масив кількостей пікселів зон
    /*Занулення масивів Zones та ZonesNum*/
    for (i = 0; i < NUMZONES; i++){
       Zones[i] = 0;
       ZonesNum[i] = 0;
    }
    int color, iZone;
    for (i = 0; i < W; i++){
       for (j = 0; j < H; j++){
          color = imSmpl.getRGB(i, j);
          iZone = Zone(i, j, W, H);
          /*Знаходження суми інтенсивності поточної зони*/
          if (iZone >= 0){
             Zones[iZone] +=((double)((0x00FF0000 & color)/65536)) / 255;
             ZonesNum[iZone]++;
          }
       }
    }
    /*Усереднення кольору зони*/
    for (i = 0; i < NUMZONES; i++){
       if (ZonesNum[i] > 0){
          Zones[i] /= ZonesNum[i];
       }
    }
    return Zones;
}
/*Даний метод визначає приналежність пікселя з координатами (x,y)  до певної зони*/
private static int Zone(double x, double y, double W, double H){
    double R = 2 * Math.hypot(x / W - 0.5, y / H - 0.5);
    if (R <= r1) return 0;
    double alpha;// кут від 0 to 2*Pi в полярних координатах
    alpha = Math.atan2(y / H - 0.5, x / W - 0.5) + Math.PI;
    if (R <= r2) return (1 + (int)Math.floor(alpha / (Math.PI / 2)));
    if (R <= r3) return (5 + (int)Math.floor(alpha / (Math.PI / 4)));
    if (R <= r4) return (13 + (int)Math.floor(alpha / (Math.PI / 8)));
    if (R <= 1) return 29;
    return -1;
}
/*Отримання компоненти r-кольору пікселя*/
public static int getR(int rgb){
    return ((0x00FF0000 & rgb) / 65536);
}
 /*Отримання компоненти g-кольору пікселя*/
public static int getG(int rgb){
    return ((0x0000FF00 & rgb) / 256);
}
 /*Отримання компоненти b-кольору пікселя*/
public static int getB(int rgb){
    return ((0x000000FF & rgb));
}
/*Обчислення функції освітленості для конкретного пікселя*/
private static double Luminance(int rgb){
    double r = ((double) getR(rgb)) / 255;
```

```
        double g = ((double) getG(rgb)) / 255;
        double b = ((double) getB(rgb)) / 255;
        return (0.2126 * r + 0.7152 * g + 0.0722 * b);
    }
    /*Даний метод обчислює середню яскравість області*/
    private static double AvgLumin(BufferedImage im, int x, int y, int rmin, int rmax){
        double l = 0, n = 0;
        for (int i = x - rmax; i < x + rmax; i++){
            for (int j = y - rmax; j < y + rmax; j++){
                if ((Math.hypot(i - x, j - y) <= rmax) && (Math.hypot(i - x, j - y) > rmin)) {
                    l += Luminance(im.getRGB(i, j));
                    n++;
                }
            }
        }
        return l / n;
    }
    /*Даний метод виділяє інформативну частину зображення*/
    public static int[] getCropArea(BufferedImage im){
        double x = 0, y = 0, s = 0;
        double min_l = Luminance(bgColor);//яскравість фонового кольору, що обрана за граничну
    /*Знаходження точки максимальної яскравості*/
        for (int i = 0; i < im.getWidth(); i++){
            for (int j = 0; j < im.getHeight(); j++){
                double l = Math.pow(Math.max(Luminance(im.getRGB(i, j)) - min_l, 0), 3);
                x += i * l;
                s += l;
                y += j * l;
            }
        }
        int cx, cy;//координати центра
        if (s > 0){
            cx = (int)(x / s);
            cy = (int)(y / s);
        }else{
            cx = im.getWidth() / 2;
            cy = im.getHeight() / 2;
        }
        /*Кінець блоку знаходження точки максимальної яскравості*/
        /*Знаходження вписаного кола з максимально можливим радіусом і центром в точці максимальної
яскравості*/
        int R = Math.min(im.getWidth(), im.getHeight());
        R = Math.min(cx, R);
        R = Math.min(cy, R);
        R = Math.min(im.getWidth() - cx, R);
        R = Math.min(im.getHeight() - cy, R);
        /*Уточнення виділеного кола шляхом відкидання фону*/
        if (R > 10){
            s = AvgLumin(im, cx, cy, R - 2, R);
            while ((s < Luminance(bgColor)) && (R > 10)){
                R -= 1;
                s = AvgLumin(im, cx, cy, R - 2, R);
            }
        }
        int ret[] = {cx, cy, R};
        return ret;
    }
    /*Даний метод реалізує Smart-Crop відповідно до обраного фонового кольору*/
```

```
    private static BufferedImage CropImage(BufferedImage im){
        int cropParam[] = getCropArea(im);
        int cx = cropParam[0];
        int cy = cropParam[1];
        int R  = cropParam[2];
        return im.getSubimage(cx - R, cy - R, 2*R, 2*R);
    }
}
```

LazerRecognizerView.java

```
/*Клас LazerRecognizerView є основним функціональним класом, що пов'язує всі інші та реалізує всі кроки
роботи програми від завантаження зображень до їх передобробки та розпізнавання*/

package lazerrecognizer;

public class LazerRecognizerView extends FrameView{
    /*Конструктор класу LazerRecognizerView, в якому створюється головне вікно додатку, панель, кнопки, меню,
списки, задаються основні кольори*/
    public LazerRecognizerView(SingleFrameApplication app) {
        super(app);
        /*Блок ініціалізації графіки*/
        initComponents();
        });
        learning = false;
        ImageCanvas.repaint();
        db = new ImageDB();
    }
    /*Даний метод повертає позицію елемента вибірки в списку*/
    private int ItemPos(JList lst, String item){
        ListModel lm = lst.getModel();
        int i;
        for(i = 0; i < lm.getSize(); i++){
            if (((String)lm.getElementAt(i)).equals(item)) break;
        }
        if (i == lm.getSize()){
            return -1;
        }else{
            return i;
        }
    }
    /*Даний метод зупиняє процес навчання по натисненню відповідної кнопки*/
    private void StopOptionButtonOKActionPerformed(java.awt.event.ActionEvent evt) {
        StopOptionButtonOK.setSelected(false);
        int n = 0;
        double e = 0;
        blMinError = CheckErrorLevel.isSelected();
        if (blMinError){
            e = Double.parseDouble(TextErrorLevel.getText());
        }
        blLimitedIterations = CheckLimitedIterations.isSelected();
        if (blLimitedIterations){
            n = Integer.parseInt(TextNumberIterations.getText());
        }
        if ((e < 0) || (n < 0)) return;
        minError = e;
        maxIterations = n;
```

```
        DialogStopOptions.setVisible(false);
    }
    /*Даний метод виконує перевірку опцій зупинки навчання*/
    private void CheckStopOptionStatus(){
        TextNumberIterations.setEnabled(CheckLimitedIterations.isSelected());
        TextErrorLevel.setEnabled(CheckErrorLevel.isSelected());
    }
    /*Даний метод виконує перерисування зображення у вигляді, як бачить його нейронна мережа*/
    private void RedrawImagePreview(){
        int i = Integer.valueOf((String)ComboImageNumber.getSelectedItem());
        ImageCanvas.basicImage = null;
        ImageCanvas.basicImage = (BufferedImage)Images.get(i - 1);
        int cropParam[] = ImageDB.getCropArea(ImageCanvas.basicImage);
        ImageCanvas.setCircle(cropParam[0], cropParam[1], cropParam[2]);
        ImageCanvas.repaint();
        ImageCopy.fillBackground(db.getImagePreview(ImageCanvas.basicImage, ImageCopy.getWidth()));
        ImageCopy.repaint();
    }
    /*Даний метод обробляє натиснення кнопки навчання мережі і запускає навчання*/
    private void ButtonLearnActionPerformed(java.awt.event.ActionEvent evt) {
        if (learning){
            net.stopLearning();
            ButtonLearn.setText("Вчитись");
            jStatusMessageLabel.setText("Готово");
            ButtonRecognize.setEnabled(true);
            learning = false;
        }else{
            if (ClassList.getModel().getSize() != 2){
                jStatusMessageLabel.setText("Недостатня кількість класів. Необхідно 2 класи.");
                return;
            }
            net = new NeuralNet(ImageDB.NUMZONES);
            try {
              //тут можна задавати кількість шарів мережі
              net.addLayer(ClassList.getModel().getSize());
            } catch (WrongParameterException ex) {
                errorHandler(ex, "Фатальна помилка при створенні нейронної мережі.");
            }
            net.setOutputObject(jStatusMessageLabel);
            double[] minI = new double[ImageDB.NUMZONES];
            double[] maxI = new double[ImageDB.NUMZONES];
            double[] minO = new double[ClassList.getModel().getSize()];
            double[] maxO = new double[ClassList.getModel().getSize()];
            for(int i = 0; i < ImageDB.NUMZONES; i++){
                minI[i] = 0;
                maxI[i] = 1;
            }
            for(int i = 0; i < ClassList.getModel().getSize(); i++){
                minO[i] = -1;
                maxO[i] = 1;
            }
            net.setMinMax(minI, maxI, minO, maxO);
            net.configurationComplete();
            net.setStopLearningOption(blLimitedIterations, maxIterations, blMinError, minError);
            ButtonLearn.setText("Зупинка навчання");
            jStatusMessageLabel.setText("Навчання");
            ButtonRecognize.setEnabled(false);
            net.startLearning(GenerateData());
```

```
            learning = true;
        }
        ButtonLearn.setSelected(false);
    }
    /*Даний метод виконує розпізнавання зображень*/
    private void ButtonRecognizeActionPerformed(java.awt.event.ActionEvent evt) {
        ImageCopy.fillBackground(ImageDB.getImagePreview(ImageCanvas.basicImage, ImageCopy.getWidth()));
        ImageCopy.repaint();
        int cropParam[] = ImageDB.getCropArea(ImageCanvas.basicImage);
        ImageCanvas.setCircle(cropParam[0], cropParam[1], cropParam[2]);
        ImageCanvas.repaint();
        try {
            DefaultTableModel imagesTM = ((DefaultTableModel)TableImageResults.getModel());
            DefaultTableModel classesTM = ((DefaultTableModel)TableClassResults.getModel());
            imagesTM.setRowCount(0);
            classesTM.setRowCount(ClassList.getModel().getSize());
            for(int i = 0; i < ClassList.getModel().getSize(); i++){
                classesTM.setValueAt((String)ClassList.getModel().getElementAt(i), i, 0);
                classesTM.setValueAt(0, i, 1);
            }
            for(int i = 0; i < Images.size(); i++){
                BufferedImage im = (BufferedImage)Images.get(i);
                double[] out = net.calculate(ImageDB.ExtractZones(im));
                int iMax = 0;
                for(int j = 1; j < out.length; j++){
                    if (out[j] > out[iMax]) iMax = j;
                }
                double certainty = 1;
                if (out.length > 1){
                    for(int j = 0; j < out.length; j++){
                        certainty -= 0.25 * Math.pow(out[j] - (iMax == j ? 1 : -1), 2) / (out.length - 1);
                    }
                }
                String sClass = (String)ClassList.getModel().getElementAt(iMax);
                Object[] rowData = {String.valueOf(i + 1), sClass, certainty};
                imagesTM.addRow(rowData);
                Integer n = (Integer)classesTM.getValueAt(iMax, 1);
                classesTM.setValueAt(n + 1, iMax, 1);
            }
            DialogRecognizeResult.setVisible(true);
        } catch (WrongParameterException ex) {
            errorHandler(ex, "Помилка обчислень нейронної мережі");
        }
        ButtonRecognize.setSelected(false);
    }
/*Даний метод округлює знак після коми при відображенні точності величин */
    private double TrunkPrecision(double v, int n){
        return java.lang.Math.rint(v * java.lang.Math.pow(10, n)) / java.lang.Math.pow(10, n);
    }
/*Даний метод реалізує редагуваня та встановлення параметрів сегментації*/
    private void SegmentationButtonOKActionPerformed(java.awt.event.ActionEvent evt) {
        SegmentationButtonOK.setSelected(false);
        double r1, r2, r3, r4, r5, r;
        r1 = Slider1.getValue();
        r2 = Slider2.getValue();
        r3 = Slider3.getValue();
        r4 = Slider4.getValue();
        r5 = Slider5.getValue();
```

```
        r = r1 + r2 + r3 + r4 + r5;
        ImageDB.r1 = r1 / r;
        ImageDB.r2 = (r1 + r2) / r;
        ImageDB.r3 = (r1 + r2 + r3) / r;
        ImageDB.r4 = (r1 + r2 + r3 + r4) / r;
        RedrawImagePreview();
        DialogSegmentation.setVisible(false);
}
/*Даний метод обробляє зміни радіусів сегментів при редагуванні*/
    private void showRadiuses(){
        double r1, r2, r3, r4, r5, r;
        r1 = Slider1.getValue();
        r2 = Slider2.getValue();
        r3 = Slider3.getValue();
        r4 = Slider4.getValue();
        r5 = Slider5.getValue();
        r = r1 + r2 + r3 + r4 + r5;
        r4 = Math.rint((r1 + r2 + r3 + r4) / r / 0.05) * 0.05;
        r3 = Math.rint((r1 + r2 + r3) / r / 0.05) * 0.05;
        r2 = Math.rint((r1 + r2) / r / 0.05) * 0.05;
        r1 = Math.rint(r1 / r / 0.05) * 0.05;
        TextR1.setText(String.valueOf(TrunkPrecision(r1, 2)));
        TextR2.setText(String.valueOf(TrunkPrecision(r2, 2)));
        TextR3.setText(String.valueOf(TrunkPrecision(r3, 2)));
        TextR4.setText(String.valueOf(TrunkPrecision(r4, 2)));
        TextR5.setText(String.valueOf(1));
    }
    /*Даний метод перетворює всю інформацію з бази даних (db) в специфічний формат для нейромережі, щоб
почати навчання*/
    private Vector GenerateData(){
        Vector vec = new Vector();
        double[] data;
        for(int i = 0; i < ImageList.getModel().getSize(); i++){
            data = new double[ImageDB.NUMZONES + ClassList.getModel().getSize()];
            for(int k = 0; k < ImageDB.NUMZONES; k++){
                data[k] = (db.getZones(i))[k];
            }
            for(int j = 0; j < ClassList.getModel().getSize(); j++) data[ImageDB.NUMZONES + j] = -1;
            data[ImageDB.NUMZONES + ItemPos(ClassList, (String)ImageList.getModel().getElementAt(i))] = 1;
            vec.add(data);
        }
        return vec;
    }
    /* Оголошення інтерфейсних змінних*/
    private boolean learning;
    private Vector Images = new Vector();
    private NeuralNet net;
    private ImageDB db;
    static String curDir = ".";
    /*Ініціалізація змінних умов зупинки навчання нейронної мережі*/
    private boolean blLimitedIterations = false;
    private int maxIterations = 0;
    private boolean blMinError = false;
    private double minError = 0;
    /*Присвоєння значення константі, що відповідає за розмір вхідних зображень*/
    private static int IMAGESIZE = 128;
}
```