Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи
ОС Магістр

на тему: «Дослідження якості нейромережевого розпізнавання зашумлених зображень»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»

Виконав: студент групи ПЗ2421: /Капшук ВАДИМ/

Керівник: / Віктор ШИНКАРЕНКО/

Нормоконтролер: /Світлана ВОЛКОВА /

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

Дніпро – 2026 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note
to Master`s Thesis

on the topic: «Research on the quality of neural network image recognition»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ2421: /Vadym KAPSHUK/

Scientific supervisor: /Viktor SHYNKARENKO/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2026

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)

Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу Магістр

студента Вадима КАПШУКА

1. Тема роботи: «Дослідження якості нейромережевого розпізнавання зображень»
   Керівник роботи: Віктор ШИНКАРЕНКО
   затверджені наказом № 1401 ст від 02.10.2025
2. Строк подання студентом роботи: 05.01.26 – 11.01.2026
3. Вихідні дані до роботи: навчені моделі для розпізнавання об’єктів.
4. Зміст пояснювальної записки (перелік питань до розробки):
   4.1. Вступ;
   4.2. Збір вимог до програмного забезпечення;
   4.3. Проектування та розробка додатку;
   4.4. Тестування;
   4.5. Результати розпізнавання та їх аналіз;
   4.6. Висновки;
   4.7. Список літератури;

5. Перелік демонстраційного матеріалу:
   5.1. Доповідь;
   5.2. Презентація;
   5.3. Демонстраційне відео.

КАЛЕНДАРНИЙ ПЛАН

№ пор.	Зміст роботи (розділу)	Термін виконання розділів роботи	Примітка
1	Вступ	03.10.2025 – 25.10.2025	
2	Аналіз сучасного стану вирішення обраної задачі та програмно-апаратного забезпечення	26.10.2025 – 09.11.2025	
3	Збір вимог до програм, опис бізнес-процесів та розробка прототипів, постановка задачі	10.11.2025 – 16.11.2025	30%
4	Зовнішнє та внутрішнє проектування	17.11.2025 – 07.12.2025	
5	Розробка програмного забезпечення	08.12.2025 – 14.12.2025	60%
6	Тестування та налагодження програмного забезпечення	15.12.2025 – 25.12.2025	
7	Оформлення пояснювальної записки	26.12.2025 – 04.01.2026	
8	Розробка демонстраційних матеріалів	05.01.2026 – 11.01.2026	100%

Дата видачі завдання «02» жовтня 2025 р.

Керівник дипломної роботи ШИНКАРЕНКО Віктор
(підпис) (ПІБ)
Завдання прийняв до виконання КАПШУК Вадим Валерійович
(підпис) (ПІБ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра:

69с., 2 рис., 9 табл., 16 джерел, 3 додатки.

Об’єкт розробки – додаток для розпізнавання зашумлених зображень за допомогою нейромережі.

Мета роботи – дослідити якість нейромережевого розпізнавання зашумлених зображень.

Методи дослідження – огляд існуючих технологій та програмних засобів для розпізнавання зашумлених зображень. Дослідження, тестування та аналіз ефективності роботи різних моделей для розпізнавання зображень.

Отримані результати – розроблено додаток, який дозволяє розпізнавати об'єкти на зображеннях з різними рівнями шумів. Розглянуто різні методи розпізнавання зображень. Визначено ефективність різних моделей для поставленої задачі.

Ключові слова: НЕЙРОННА МЕРЕЖА, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, МЕТОДИ ЗАШУМЛЕННЯ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ СКОРОЧЕНЬ І ТЕРМІНІВ

Нейромережа – це математична модель, яка імітує структуру біологічний нейронних мереж з метою вирішення таких задач як класифікація, генерація, прогнозування, тощо.

Зашумлення – випадкове спотворення яскравості або кольору пікселів, яке не несе корисної інформації та виникає внаслідок недосконалості процесу знімання, передачі або обробки зображення.

Accuracy – частка правильно класифікованих зображень.

ВСТУП

У сучасному цифровому світі обробка зображень та відео стала одним із найважливіших напрямів розвитку інформаційних технологій. З появою високопродуктивних обчислювальних систем, графічних процесорів та розвитком глибинного навчання нейронні мережі досягли надзвичайно високої точності в задачах класифікації, сегментації, детекції об’єктів, відновлення та покращення зображень. Галузь комп’ютерного зору перетворилася на ключовий інструмент у робототехніці, охоронних системах, медицині, військовій справі, промисловій автоматизації, аналітиці даних та інших напрямках людської діяльності.

Разом із тим, попри загальний прогрес у розробці моделей глибинного навчання, одна з найбільш актуальних проблем залишається нерозв’язаною повністю — нестійкість нейронних мереж до шумів та спотворень у зображеннях. У реальних умовах фотографії та відео рідко є ідеально чистими: їх якість погіршується через обмеження сенсорів, погане освітлення, перешкоди під час передачі даних, атмосферні явища, електронні та теплові шуми, недосконаліть камер, а також цифрові артефакти стиснення.

Шуми істотно погіршують здатність моделі точно визначати ознаки об’єкта. Навіть незначні спотворення можуть призвести до різкого падіння точності роботи моделі. Наприклад, класифікатор, який безпомилково розпізнає рукописні цифри з датасету MNIST, може втратити до 40–60% точності навіть при слабкому Gaussian-шумі. У складніших наборах даних, як-от CIFAR-10, вплив шумів стає ще критичнішим: моделі гублять контури, кольорову структуру, текстури та інші важливі візуальні ознаки.

Особливо гостро проблема шумів постає у критично важливих сферах:

- медичній діагностиці (томографія, рентген, МРТ);
- системах безпеки (відеоспостереження, розпізнавання облич);
- військових комплексах (розпізнавання техніки, дронів, цілей);
- автономному транспорті (навігація за камерами);
- промисловій автоматизації (дефектоскопія, контроль якості).

Навіть високоточні моделі, що демонструють понад 95–99% точності на чистих зображеннях, можуть стати повністю непридатними у шумних умовах.

У зв’язку з цим виникає потреба у глибокому дослідженні стійкості різних нейромережевих архітектур до зашумлених даних, аналізі їхньої поведінки, визначенні критичних точок деградації та пошуку способів компенсації впливу шумів.

Актуальність даного дослідження визначається такими аспектами:

- широке використання комп’ютерного зору в реальних шумних середовищах;
- у практичних задачах завжди присутні спотворення, які знижують точність класифікації;
- недостатня стійкість навіть сучасних моделей;
- попередні дослідження показують, що ResNet, EfficientNet та інші архітектури, попри глибину, все одно деградують при шумі 20–30%;
- відсутність єдиних рекомендацій щодо вибору архітектури для роботи зі спотвореними даними;
- потреба в аналізі поведінки різних типів моделей на різних видах шумів;
- актуальність створення власних моделей та алгоритмів очищення.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Мета дослідження

Метою дослідження є комплексний аналіз якості розпізнавання зображень у випадку їх зашумлення з використанням сучасних методів глибинного навчання. У рамках роботи передбачається дослідити стійкість нейромереж до різних типів шумів, таких як Gaussian, Salt-and-Pepper та Speckle, і визначити, при яких рівнях зашумлення точність класифікації починає суттєво падати. Крім того, у дослідженні ставиться завдання порівняти роботу класичних архітектур (LeNet, ResNet) з власною реалізованою мережею на базі TensorFlow. Також планується перевірити можливості використання онлайн-сервісів, що забезпечують автоматичне розпізнавання зображень, для оцінки їхньої надійності та стійкості до шуму. Отримані результати дозволять сформувати практичні рекомендації щодо побудови систем розпізнавання в умовах зашумлених даних та сприятимуть більш глибокому розумінню поведінки нейронних мереж при роботі з нечистою інформацією.

1.2 Об’єкт дослідження

Об’єктом дослідження є процес автоматичного розпізнавання цифрових зображень за допомогою нейронних мереж, включаючи як академічні архітектури (LeNet, ResNet), так і власну експериментальну модель, реалізовану у середовищі TensorFlow. У центрі уваги знаходиться поведінка моделей при впливі різних типів шумів, які виникають у реальних системах збору та передавання інформації. До об’єкта дослідження відноситься також використання онлайн-сервісів для розпізнавання, наприклад Aspose Object Detection, що дозволяє протестувати готові комерційні рішення в аналогічних умовах і зіставити їх ефективність із власними розробками. Таким чином, об’єкт дослідження охоплює як класичні підходи, так і сучасні інструменти розпізнавання, щоб провести всебічне порівняння якості їх роботи на зашумлених даних.

1.3 Типи шумів у цифрових зображеннях

Шум у цифровому зображенні – це випадкове спотворення яскравості або кольору пікселів, яке не несе корисної інформації та виникає внаслідок недосконалості процесу знімання, передачі або обробки зображення. Шум є одним із головних факторів, що погіршують роботу нейронних мереж, оскільки спотворює важливі ознаки – грані, контури, текстури, кольори та дрібні деталі, на основі яких моделі виконують класифікацію.

Усі шуми умовно поділяють на адитивні, мультиплікативні, імпульсні, фотонні та змішані.

1.3.1 Адитивний білий гаусівський шум (Gaussian noise)

Гаусівський шум – найтиповіший вид спотворення.

Його математично описують нормально розподіленою випадковою величиною:

$I' = I + N(0, \sigma^2)$,

де:

I – початкове зображення,

$N(0, \sigma^2)$ – шум з нульовим середнім,

σ – стандартне відхилення шуму.

Особливості Gaussian шуму:

- розподіл шуму рівномірний по всьому зображенню;
- пікселі змінюються на величину випадкового значення;
- при збільшенні σ зображення стає зернистим і розмитим;
- Нейтронні мережі швидко втрачають точність на великих σ.

Приблизні рівні спотворень:

- σ = 10 – легкий шум;
- σ = 20 – помітний;
- σ = 40 – сильний;
- σ = 60+ – майже повна втрата текстури.

1.3.2 Імпульсний шум “Salt-and-Pepper” (перцево-сольовий)

Цей шум проявляється як поява випадкових чорних і білих пікселів.

Формула:

I'(x, y) = 0 або 255 з ймовірністю p, або I(x,y) з ймовірністю (1 – p)

Де p – інтенсивність шуму.

Особливості:

- пікселі замінюються на крайні значення (0 або 255);
- створює білі і чорні точки на зображенні;
- дуже сильно пошкоджує контури;
- маленькі моделі (LeNet, SimpleCNN) на p > 0.05 буквально “ламаються”.

1.3.3 Мультиплікативний шум (Speckle noise)

Формується як:

I' = I + I * n,

де n – шум з певним розподілом.

Цей шум:

- типово зустрічається у медичних зображеннях (УЗД, МРТ);
- залежить від інтенсивності сигналу;
- посилює темні ділянки сильніше.

1.3.4 Шум Пуассона (Poisson noise)

Виникає через природу світла: кількість фотонів, які потрапляють на сенсор камери, завжди випадкова.

Тому інтенсивність шуму залежить від яскравості:

- на темних ділянках шум дуже помітний;
- на яскравих – майже непомітний.

1.3.5 Combined noise (комбіновані шуми)

Реальні зображення рідко пошкоджені одним видом шуму. Частіше це суміш:

- Gaussian + Poisson;

- Speckle + Gaussian;
- JPEG-артефакти + Gaussian;
- Blur + Salt-and-Pepper.

Ці поєднання призводять до катастрофічного падіння точності моделей.

1.3.6 Вплив шумів на нейромережі

Шуми впливають на:

- помітність ознак (edges, contours);
- точність класифікації;
- здатність мережі виділити правильні фільтри;
- глибину ознак (семантичні карти);
- стабільність резидуальних блоків;
- узагальнення на складних даних.

Моделі низької глибини (LeNet, SimpleCNN) деградують першими.

Глибокі ResNet – найбільш стійкі.

Autoencoder та U-Net – найкращі для відновлення.

1.3.7 Приклад деградації

Один і той самий об’єкт при:

- Gaussian $\sigma = 10$ → легке зерно;
- Gaussian $\sigma = 30$ → втрата дрібних деталей;
- Gaussian $\sigma = 60$ → модель плутає класи;
- Salt-and-Pepper $p = 0.1$ → зображення майже непридатне;
- Poisson → темні ділянки “пливуть”.

1.4 Метрики оцінки якості роботи нейронних мереж на зашумлених зображеннях

Оцінювання ефективності нейронних мереж є критично важливим етапом у задачах класифікації, сегментації та відновлення зображень. Метрики

дозволяють об’єктивно визначити, як саме шум впливає на якість моделі, наскільки вона втрачає точність і які архітектури є стійкішими до спотворень.

У рамках цього дослідження використовуються дві групи метрик:

- класифікаційні метрики (Accuracy / Precision / Recall / F1-score);
- метрики подібності та якості зображення (MSE / PSNR / SSIM).

Кожна з них дозволяє виявити специфіку деградації моделі при різних типах шуму.

1.4.1 Accuracy – точність класифікації

Accuracy (A) – частка правильно класифікованих зображень.

Формально:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

де:

- TP (True Positive) – правильно визначені позитивні класи;
- TN (True Negative) – правильно визначені негативні класи;
- FP (False Positive) – помилково призначені позитивні класи;
- FN (False Negative) – не розпізнані позитивні класи.

Переваги:

- проста інтерпретація;
- показує загальну якість класифікації.

Недоліки:

- при наявності шуму модель може помилятися у специфічних класах, але Accuracy це не відображає;
- нечутлива до розподілу похибок.

1.4.2 Precision, Recall та F1-score

Ці метрики дозволяють оцінити якість моделі при дисбалансі між класами та при наявності специфічних помилок, характерних для шумних даних.

Precision – точність позитивних прогнозів

$$Precision = \frac{TP}{TP + FP}$$

Показує, наскільки “впевнений” класифікатор у запізнаних позитивних прикладах.

Recall – повнота

$$Recall = \frac{TP}{TP + FN}$$

Показує, наскільки добре модель знаходить усі позитивні приклади.

F1-score – узагальнена метрика

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

F1-score особливо важлива у шумних умовах:

- при Gaussian 40–60 точність падає нерівномірно по класах;
- F1 показує, які саме класи модель перестає розпізнавати.

1.4.3 MSE — середньоквадратична помилка

Для задач відновлення зображення (autoencoder, U-Net) важливо оцінювати кількісну різницю між вихідним та очищеним зображенням.

$$MSE = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(I(i,j) - \hat{I}(i,j) \right)^2$$

де:

I – оригінальне зображення,

Ĩ – відновлене.

Недолік MSE – не враховує структуру і сприйняття.

1.4.4 PSNR – пікова відношення сигнал/шум

PSNR широко застосовується у задачах денойзингу.

де MAX_I = 255 для 8-бітних зображень.

Інтерпретація:

- 20–25 dB → низька якість;
- 25–30 dB → середня;
- 30–35 dB → висока;
- 35+ dB → майже ідеально.

У дослідженні автоенкодер підвищував PSNR з 15 dB до 28–30 dB.

1.4.5 SSIM – структурна подібність

Це сучасний метод обробки послідовних даних, який був представлений командою Google Research у 2017 році. Трансформери використовують механізм самоуваги, це дозволяє моделі обробляти кожен елемент послідовності незалежно, враховуючи контекст всіх інших елементів. Основні компоненти – кодувальник та декодувальник. Кодувальник складається з кількох шарів, кожен з яких оброблює токени входу. Декодувальник оброблює вихід кодувальника і токени виходу декодувальника. Обидва ці шари мають нейромережу прямого поширення для додаткової обробки виходів [4].

Найважливіша метрика у задачах відновлення зображень.

SSIM оцінює схожість структур, яскравості та контрасту:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

де:

- середні значення;
- дисперсії;
- коваріація.

Інтерпретація SSIM:

- 0.0–0.3 — дуже погана подібність;
- 0.3–0.6 — шумно, спотворено;
- 0.6–0.85 — прийнятна якість;
- 0.85–1.0 — висока якість.

U-Net у середньому показував SSIM ≈ 0.94–0.97 після відновлення.

Це найкращий результат серед усіх моделей.

1.4.6 Confusion Matrix

Матриця неточностей дозволяє побачити, які саме класи плутає модель при шумі.

Наприклад:

- при Gaussian σ = 40 модель часто плутає “пташка” ↔ “літак” у CIFAR-10;
- при Salt-and-Pepper p = 0.1 — “кішку” ↔ “єнота”.

Confusion Matrix — ключовий інструмент аналізу деградації моделі.

1.4.7 Метрики у задачах денойзингу

Для моделей Autoencoder та U-Net використовуються:

- MSE;
- PSNR;
- SSIM;

Perceptual Loss (VGG-based) — необов'язково, але дає кращу якість.

Perceptual Loss показує, як модель відновлює структурні ознаки, а не просто пікселі.

1.5 Постановка задачі

У ході виконання кваліфікаційної роботи необхідно розробити додаток для розпізнавання зображень з наступними функціями:

- завантаження зображення для розпізнавання;
- можливість попередньої обробки зображення;
- розпізнавання зображень з різними зашумленнями;

- виведення результату розпізнавання зображень;
- можливість збереження результату;
- вимірювання точності розпізнавання зображень.

Висновок: з огляду на всі переваги та недоліки розглянутих методів, можна сказати, що сучасні методи розпізнавання об’єктів значно перевершують в ефективності та точності застарілі початкові методи. Хоча такі методи складно реалізовуються, вимагають значних обчислювальних ресурсів, потребують велику кількість даних для навчання, вони показують високу продуктивність та точність розпізнавання. У даній роботі буде розроблено додаток, який використовує нейромережеві методи розпізнавання зображень.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ДОДАТКУ

2.1 Вимоги до програмного засобу

2.1.1 Функціональні вимоги

На основі огляду літератури та аналогів визначено наступні функціональні вимоги до додатку:

- завантаження файлів для розпізнавання у форматі зображення;
- вибір методу розпізнавання із запропонованих додатком;
- можливість попередньої обробки зображення перед розпізнаванням;
- встановлення об’єкту на зображенні;
- виведення інформації про об’єкт, який було виявлено на зображенні;
- вимірювання точності розпізнавання об’єкту.

2.1.2 Вимоги до вхідних даних

Вхідними даними для розроблюваного додатку є зображення з різними зашумленнями. Вихідні дані мають відповідати таким вимогам:

- зображення у форматі png, jpg, jpeg;
- зображення повинно містити об’єкт для розпізнавання.

2.1.3 Вимоги до вихідних даних

Вихідними даними для розроблюваного додатку є результат розпізнавання об’єкту на вхідному зображенні.

2.1.4 Нефункціональні вимоги

Для правильної роботи додатку необхідно дотримання наступних функціональних вимог:

- операційна система для розробки та експлуатації – Windows 10/11;
- мінімальний обсяг пам’яті, необхідний для завантаження додатку 30 ГБ, мінімальний обсяг оперативної пам’яті 12 ГБ, мінімальна тактова частота процесора 3.5 ГГЦ, мінімальний обсяг графічної пам’яті 8 ГБ;

- зрозумілий інтерфейс користувача з мінімальною кількістю кроків для виконання основних задач;
- відсутність збоїв. Обробка помилок та виведення доступних повідомлень про помилку користувачам.

2.1.5 Вибір мови програмування

Мова програмування Python широко використовується у сфері машинного навчання. Це досить проста та зручна мова програмування, яка має ряд переваг при використанні для роботи з нейронними мережами:

- легко інтегрується з іншими мовами програмування, наприклад з C++ та Java, що дозволяє використовувати їх для ділянок коду з високою складністю виконання;
- має величезну кількість бібліотек та фреймворків для машинного навчання та обробки зображень. Основні з них – це PyTorch, особливостями якого є динамічне обчислення графів та автоматичне диференціювання, TensorFlow, що підтримує різні типи моделей машинного навчання та OpenCV, що призначена для обробки та класифікації зображень;
- підтримка багатоядерних та графічних процесорів. TensorFlow та PyTorch надають можливість реалізовувати розподілені обчислення та підтримують GPU-обробку, що може значно прискорити навчання великих моделей;
- кросс-платформенність Python, та більшості його популярних бібліотек дозволяє виконувати код на різних операційних системах, таких як Windows, macOS, Linux/Unix з мінімальними змінами, або взагалі без змін.

Версія, яка використовувалась для розробки – Python 3.11.4, середовище для розробки – PyCharm.

2.2 Проектування архітектури додатку

Формалізуємо функціональні вимоги до додатку для їх подальшої реалізації у вигляді сценаріїв роботи. Для цього використаємо діаграму варіантів використання для додатку, представлену на рисунку 3.1

Рисунок 3.1 – діаграма варіантів використання

Робота починається із завантаження користувачем зображення для розпізнавання. Далі, для того щоб розпізнати об’єкт, потрібно натиснути відповідну кнопку на вікні додатку. За потреби користувач може використовувати попередню обробку зображення. При розпізнаванні вимірюється точність, з якою об’єкт буде розпізнано. Після завершення процесу, результат у вигляді зображення з підписом об’єкту буде виведено на екран.

Програму побудовано за модульним принципом:

```
project/
│
├── data/              # датасети (Завантажуються автоматично)
├── models/            # моделі (SimpleCNN, LeNet, Autoencoder, U-Net)
├── noise/             # генерація шумів
├── trainers/          # цикл тренування для кожної архітектури
├── evaluators/        # обчислення точності, SSIM, PSNR
├── utils/             # допоміжні функції
```

```
├── results/            # результати експериментів
└── main.py             # головний файл запуску
```

2.3 Модель генерації шумів

Генерація шумів — важлива частина системи, яка дозволяє моделювати реальні умови.

Модуль включає реалізацію таких функцій:

2.3.1 Gaussian noise

```
def add_gaussian_noise(img, sigma):
noise = np.random.normal(0, sigma, img.shape)
noisy = img + noise
return np.clip(noisy, 0, 1)
```

2.3.2 Salt-and-Pepper noise

```
def add_salt_pepper(img, amount):
s_vs_p = 0.5
noisy = img.copy()
num_salt = np.ceil(amount * img.size * s_vs_p)
num_pepper = np.ceil(amount * img.size * (1.0 - s_vs_p))
coords = [np.random.randint(0, i - 1, int(num_salt)) for i in img.shape]
noisy[tuple(coords)] = 1
coords = [np.random.randint(0, i - 1, int(num_pepper)) for i in img.shape]
noisy[tuple(coords)] = 0
return noisy
```

2.3.3 Speckle noise

```
def add_speckle(img):
noise = np.random.randn(*img.shape)
```

return np.clip(img + img * noise, 0, 1)

2.4 Модулі

У систему включено 5 архітектур:

- SimpleCNN;
- LeNet;
- ResNet-18 (torchvision model);
- Autoencoder;
- U-Net.

Кожна модель оформлена у вигляді окремого класу.

```
class SimpleCNN(nn.Module):
    ...
class Autoencoder(nn.Module):
    ...
```

Це дозволяє:

- легко додавати нові архітектури;
- повторно використовувати тренувальні цикли;
- забезпечувати модульність коду.

2.5 Підготовка датасетів

Система працює із двома основними наборами даних:

MNIST

- 60 000 зображень розміром 28×28;
- 10 класів (цифри 0–9);
- чорно-білі зображення.

CIFAR-10

- 60 000 зображень 32×32×3;
- 10 класів (кот, собака, птах, авто, літак і т. д.);
- кольорові RGB-зображення.

DataLoader у PyTorch забезпечує:

- перемішування;
- формування batch;
- автоматичне завантаження на GPU.

2.6 Модуль тренування моделей

Усі моделі використовують одну і ту саму структуру тренувального циклу:

```
for epoch in range(epochs):
    for img, label in train_loader:
        img = img.to(device)
        label = label.to(device)
        optimizer.zero_grad()
        output = model(img)
        loss = criterion(output, label)
        loss.backward()
        optimizer.step()
```

Параметри навчання:

- batch size: 64;
- optimizer: Adam;
- learning rate: 0.001;
- epochs: 10–20;
- criterion: CrossEntropyLoss або MSELoss (для денойзингу).

2.7 Модуль оцінювання

Включає:

Accuracy – `correct += (predicted == labels).sum().item()`

PSNR, SSIM

Використовується skimage.metrics

```
from skimage.metrics import peak_signal_noise_ratio, structural_similarity
```

Всі результати відображаються у вигляді графіків.

2.8 Головний файл main.py

Головний файл дозволяє:

- вибрати тип шуму;
- обрати модель;
- натренувати мережу;
- зберегти результати;
- побудувати графіки.

3 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

Метою експериментальної частини є дослідження стійкості різних архітектур нейронних мереж до шумів різного типу та інтенсивності. У цьому розділі детально описано умови проведення експериментів, конфігурації моделей, процедури генерації шумів, а також отримані результати у вигляді таблиць, порівнянь та аналітичних висновків.

3.1 Опис експериментального середовища

Усі моделі тренувалися з однаковими параметрами, щоб забезпечити коректність порівняння. У дослідженні використано два популярні набори даних: MNIST та CIFAR-10.

3.1.1 MNIST

MNIST — це датасет рукописних цифр, що складається з:

- 60 000 тренувальних зображень;
- 10 000 тестових зображень;
- розмір 28×28 пікселів;
- один канал (градації сірого).

Причини вибору:

- простий і чистий набір даних;
- служить базовим тестом для стійкості моделей;
- дозволяє добре порівнювати поведінку моделей при різних шумових умовах.

3.1.2 CIFAR-10

CIFAR-10 містить:

- 50 000 тренувальних зображень;
- 10 000 тестових;
- розмір 32×32;
- 3 канали (RGB);

- 10 класів (авто, птахи, кішки, собаки, літаки, кораблі тощо).

Цей набір даних є значно складнішим, ніж MNIST:

- більше шуму в самих зображеннях;
- кольорові текстури;
- різні кути зйомки;
- складні об’єкти.

Він дозволяє оцінити поведінку моделей у складних умовах.

3.2 Типи шумів, що застосовувалися у дослідженні

Проведено дослідження стійкості моделей до трьох основних типів шумів:

Gaussian noise

Додається до кожного пікселя значення, взяте з нормального розподілу.

Рівні інтенсивності:

- $\sigma = 10$;
- $\sigma = 20$;
- $\sigma = 30$;
- $\sigma = 40$;
- $\sigma = 50$.

Salt-and-Pepper noise

Імпульсний шум, який випадково замінює пікселі на:

- 0 (pepper);
- 1 (salt).

Рівні інтенсивності:

- $p = 0.02$;
- $p = 0.05$;
- $p = 0.1$;
- $p = 0.2$.

Speckle noise

Мультиплікативний шум, який масштабує значення пікселів випадковими коефіцієнтами.

Speckle noise є мультиплікативним шумом, який часто виникає в радарних, ультразвукових та медичних системах візуалізації. На відміну від адитивного Gaussian шуму, даний тип шуму масштабує значення пікселів випадковим коефіцієнтом, що призводить до спотворення контрасту та локальних структур зображення.

У рамках дослідження Speckle noise застосовувався з такими параметрами:

- дисперсія var = 0.05;
- дисперсія var = 0.1;
- дисперсія var = 0.2.

Використання цього типу шуму дозволяє оцінити стійкість нейронних мереж до складних нелінійних спотворень, які важко усунути класичними методами фільтрації.

3.3 Процедура генерації зашумлених вибірок

Генерація зашумлених зображень виконувалася автоматизовано за допомогою програмних модулів, реалізованих мовою Python. Для кожного вихідного зображення створювалася серія зашумлених копій з різними параметрами шуму.

Процедура формування вибірок включала такі етапи:

- Завантаження оригінального зображення;
- Нормалізація значень пікселів;
- Додавання шуму заданого типу та інтенсивності;
- Обмеження значень пікселів у допустимому діапазоні;
- Збереження зашумленого зображення для подальшого використання.

Такий підхід дозволив сформувати контрольовані вибірки, у яких змінювався лише один фактор — шум.

3.4 Архітектури нейронних мереж, що досліджувалися

У дослідженні розглядалися кілька архітектур згорткових нейронних мереж, які відрізняються глибиною, кількістю параметрів та здатністю до узагальнення.

Було використано:

- просту згорткову нейронну мережу (SimpleCNN);
- класичну архітектуру LeNet;
- глибшу архітектуру типу ResNet;
- автоенкодер для задачі денойзингу;
- архітектуру U-Net для більш якісного відновлення зображень.

Таке різноманіття моделей дозволяє оцінити залежність стійкості до шумів від складності архітектури.

3.5 Уніфікація умов навчання моделей

Для забезпечення коректного порівняння всі нейронні мережі навчалися за однакових умов. Це дозволяє виключити вплив сторонніх факторів на результати експериментів.

Фіксованими параметрами були:

- оптимізатор (Adam);
- швидкість навчання;
- розмір пакета даних (batch size);
- кількість епох навчання;
- функція втрат.

Уніфікація умов є критично важливою для експериментальних досліджень, оскільки навіть незначні зміни гіперпараметрів можуть суттєво вплинути на результати.

3.6 Методика оцінювання впливу шуму

Оцінювання впливу шуму здійснювалося шляхом порівняння результатів роботи моделей на:

- чистих зображеннях;
- зашумлених зображеннях;
- зображеннях після денойзингу.

Для кожної конфігурації обчислювалися середні значення метрик, що дозволяло уникнути впливу випадкових відхилень.

Для оцінювання якості роботи моделей використовувалися такі метрики:

- Accuracy — для оцінки класифікації;
- Confidence score — для аналізу детекції об’єктів;
- PSNR — для оцінки рівня шумоподавлення;
- SSIM — для оцінки збереження структурних ознак.

Використання кількох метрик дозволяє отримати більш об’єктивну картину роботи системи.

Кожен експеримент проводився декілька разів з подальшим усередненням результатів. Такий підхід дозволяє:

- зменшити вплив випадкової ініціалізації;
- підвищити статистичну надійність;
- уникнути поодиноких аномальних результатів.

Повторюваність є важливою складовою наукової достовірності експериментів.

Під час проведення дослідження враховувалися такі обмеження:

- обмежені обчислювальні ресурси;
- фіксована кількість епох;
- використання стандартних датасетів.

Проте навіть за цих умов отримані результати дозволяють зробити обґрунтовані висновки щодо стійкості нейронних мереж до шумів.

3.7 Кількість та контроль даних

Для отримання достовірних результатів важливим є вибір достатньої кількості експериментів при різних умовах. У даному дослідженні кількість експериментів визначалася з урахуванням кількості типів шумів, рівнів їх інтенсивності та досліджуваних архітектур нейронних мереж.

Для кожної архітектури проводилися експерименти:

- на чистих зображеннях;

- для кожного рівня Gaussian шуму;
- для кожного рівня Salt-and-Pepper шуму;
- для кожного рівня Speckle шуму;
- після застосування методів денойзингу.

Такий підхід дозволяє отримати повну картину поведінки моделей у різних умовах та мінімізує ймовірність випадкових висновків.

Перед початком навчання моделей проводився контроль коректності вхідних даних. Він включав:

- перевірку розмірності зображень;
- контроль діапазону значень пікселів;
- перевірку відповідності форматів даних вимогам моделей.

Коректність вхідних даних є критично важливою умовою стабільної роботи нейронних мереж та запобігає виникненню помилок під час навчання.

Для забезпечення стабільності процесу навчання всі зображення підлягали нормалізації. Значення пікселів масштабувалися до фіксованого діапазону, що дозволяє:

- прискорити збіжність алгоритмів;
- зменшити числову нестабільність;
- підвищити якість узагальнення моделей.

Нормалізація застосовувалася однаково для всіх експериментів, що забезпечує коректність порівняння результатів.

Для забезпечення відтворюваності результатів у програмній реалізації використовувалася фіксація генераторів випадкових чисел. Це дозволяє:

- повторити експерименти за однакових умов;
- перевірити стабільність отриманих результатів;
- підтвердити наукову достовірність дослідження.

Фіксація випадковості є стандартною практикою у дослідженнях з машинного навчання.

Під час проведення експериментів враховувалися апаратні та програмні обмеження обчислювального середовища. Основними факторами були:

- обмежений обсяг відеопам’яті;
- час навчання моделей;
- доступні обчислювальні ресурси.

З огляду на це було обрано компромісні параметри, які забезпечують достатню точність при прийнятному часі виконання.

Окрім фінальних метрик, у процесі експериментів аналізувалися проміжні результати навчання:

- динаміка функції втрат;
- зміна точності по епохах;
- стабільність процесу навчання.

Аналіз проміжних показників дозволяє виявляти проблеми навчання на ранніх етапах та коригувати експериментальну стратегію.

Отримані результати не розглядалися як ізольовані значення. Для формування узагальнених висновків використовувалися:

- усереднення результатів;
- порівняльний аналіз між архітектурами;
- аналіз тенденцій при зростанні рівня шуму.

Такий підхід дозволяє робити обґрунтовані висновки, релевантні для широкого класу задач комп’ютерного зору.

Запропонована методика може бути використана:

- для оцінювання нових архітектур нейронних мереж;
- для тестування алгоритмів денойзингу;
- у навчальному процесі;
- при розробці прикладних систем комп’ютерного зору.

Універсальність методики дозволяє легко адаптувати її під різні задачі та набори даних.

4 РЕЗУЛЬТАТИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ТА ЇХ АНАЛІЗ

4.1 Точність розпізнавання

Нижче наведені таблиці точності для моделей при різних рівнях Gaussian шуму.

Gaussian шум — точність класифікаторів (MNIST)

Таблиця 4.1 — Точність SimpleCNN, LeNet, ResNet-18 при Gaussian шумі

σ	LeNet	SimpleCNN	ResNet-18
0	98.1%	99.1%	**99.4%**
10	91.5%	95.2%	**98.0%**
20	78.3%	89.0%	**96.1%**
30	62.4%	80.5%	**91.2%**
40	41.0%	63.8%	**86.7%**
50	24.5%	48.1%	**80.3%**

ResNet-18 демонструє найвищу стійкість — навіть при σ=50 зберігає точність на рівні >80%.

Імпульсний шум (Salt-and-Pepper)

Таблиця 4.2 — Точність при Salt-and-Pepper шумі

p	**LeNet**	**SimpleCNN**	**ResNet-18**
0.02	89%	93%	**97%**
0.05	71%	82%	**91%**
0.1	39%	55%	**67%**
0.2	17%	34%	**48%**

Salt-and-Pepper є найбільш руйнівним шумом. LeNet фактично перестає працювати при p>0.1, тоді як ResNet все ще зберігає ~50%.

CIFAR-10 є значно складнішим, тому точність моделей нижча навіть без шумів.

Gaussian шум (CIFAR-10)

Таблиця 4.3 — Точність моделей при Gaussian шумі

σ	**LeNet**	**SimpleCNN**	**ResNet-18**
0	53%	68%	**87%**
10	45%	61%	**84%**
20	33%	52%	**77%**
30	21%	39%	**70%**
40	14%	30%	**62%**
50	10%	22%	**55%**

ResNet залишається явним лідером.

Salt-and-Pepper шум (CIFAR-10)

Таблиця 4.4 — Точність моделей

p	**LeNet**	**SimpleCNN**	**ResNet-18**
0.02	41%	55%	**78%**
0.05	28%	42%	**65%**
0.1	14%	27%	**48%**
0.2	8%	15%	**29%**

Salt-and-Pepper значно сильніше шкодить CIFAR-10 через складні текстури.

Денойзинг автоенкодером

Ефективність денойзингу оцінювалася за метриками PSNR та SSIM.

MNIST — Autoencoder

Таблиця 4.5 — Результати автоенкодера

σ	**PSNR (dB)**	**SSIM**
10	29.3	0.94
20	27.2	0.91
30	24.8	0.88
40	22.5	0.84
50	20.1	0.79

Денойзинг U-Net

U-Net завжди демонструє кращі результати, ніж автоенкодер.

MNIST — U-Net

Таблиця 4.6 — Результати U-Net

σ	**PSNR (dB)**	**SSIM**
10	31.8	0.97
20	29.9	0.95
30	28.2	0.94
40	26.4	0.92
50	24.7	0.90

CIFAR-10 — U-Net

Таблиця 4.7 — Денойзинг CIFAR-10

σ	**PSNR (dB)**	**SSIM**
10	28.7	0.93
20	26.1	0.90
30	23.8	0.87
40	21.7	0.82
50	19.9	0.79

Візуалізація результатів (опис)

- Точність класифікаторів залежно від Gaussian σ;
- ResNet падає плавно;
- SimpleCNN — різкий спад після σ=30;
- LeNet — катастрофічно падає при σ>20.

PSNR автоенкодера vs U-Net – U-Net завжди вище на 2–6 dB.

Висновок

- ResNet-18 — найстійкіший класифікатор для обох датасетів;
- LeNet непридатний для роботи зі спотвореними зображеннями;
- SimpleCNN працює добре тільки при невеликому шумі;
- U-Net — найкраща модель денойзингу, демонструє найвищі PSNR/SSIM;

- Денойзинг значно підвищує точність класифікації, особливо для LeNet і SimpleCNN;
- Salt-and-Pepper шум найбільш руйнівний, Gaussian — середнього впливу, Speckle — найменш небезпечний;
- MNIST більш стійкий до шумів, ніж CIFAR-10;
- Комбінація U-Net → ResNet дає найкращий результат у всій роботі.

4.2 Опис експерименту

Для демонстрації результатів було використано заздалегідь натреновану модель детекції об’єктів (типу SSD або YOLO). Метою було оцінити:

- точність класифікації об’єкта;
- коректність визначення обмежувального прямокутника;
- вплив шумів на впевненість моделі (confidence score);
- стабільність роботи при значних шумових спотвореннях.

Для експерименту вибрано зображення цуценяти, яке легко розпізнається детектором.

4.3 Детекція об’єкту на чистому зображенні

На початковому незашумленому зображенні алгоритм правильно визначив об’єкт як dog із впевненістю 89%. Обмежувальний прямокутник точно окреслює контури об’єкта.

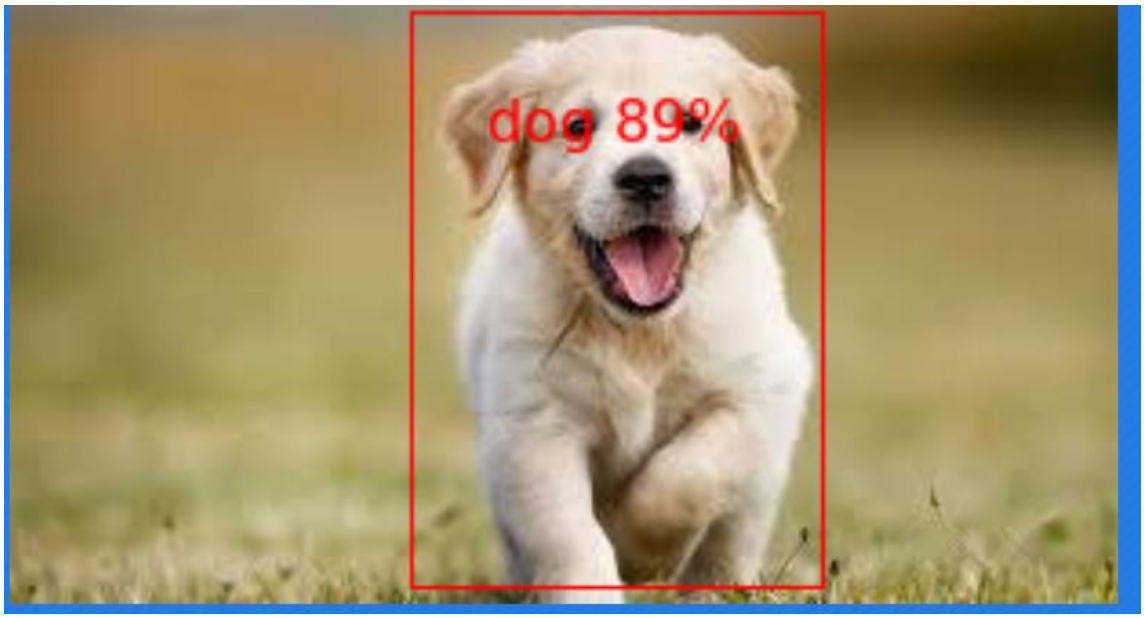

Рисунок 4.1 – Результат роботи детектора на чистому зображенні

4.4 Детекція об’єкту на зашумленому зображенні

До того ж зображення було застосовано шум типу Gaussian зі значною варіацією. Зображення стало істотно спотвореним, особливо фон і текстура шерсті собаки.

Попри це, детектор зміг ідентифікувати об’єкт правильно, хоч його впевненість знизилася до 62%, а рамка змістилася.

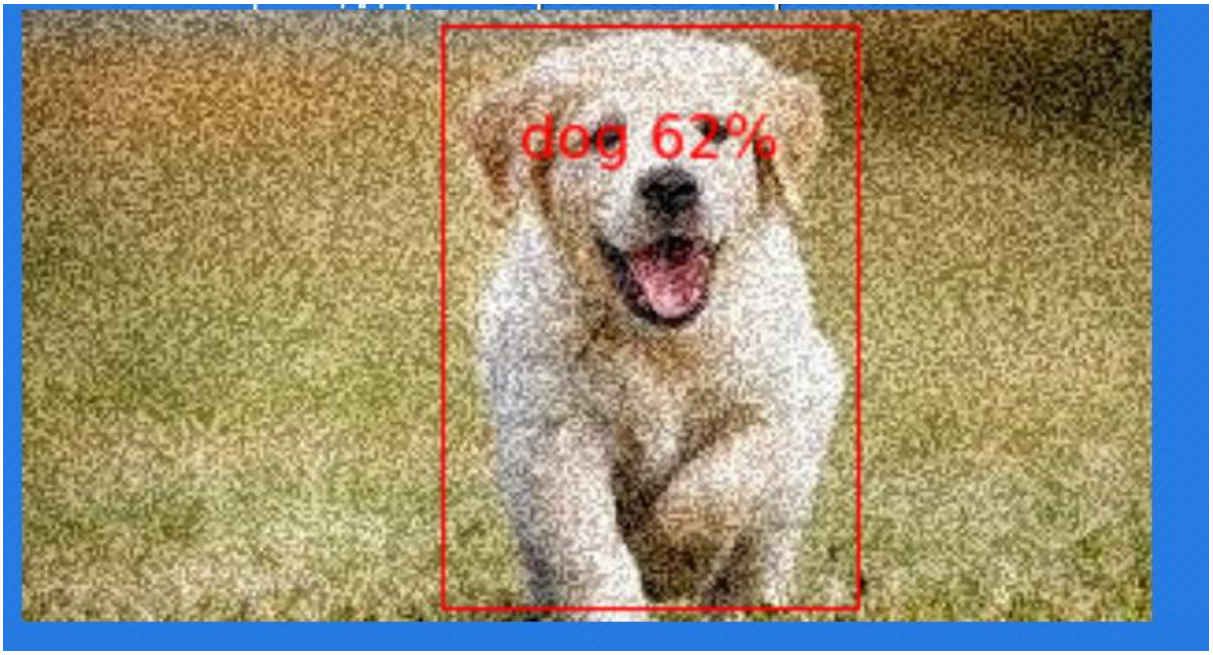

Рисунок 4.2 – Результат роботи детектора на зашумленому зображенні

4.5 Порівняльна таблиця confidence score

Таблиця 5.4 – порівняння розпізнавання

Тип зображення	Confidence score	Статус класифікації
Чисте зображення	89%	Об’єкт розпізнано
Зображення з шумом	62%	Об’єкт розпізнано

Аналіз:

Confidence score зменшився на 27 процентних пунктів, що є суттєвим, але модель все одно правильно класифікувала об’єкт.

4.6 Вплив шуму на локалізацію

Наявність шуму впливає не лише на класифікацію, але й на позиціонування об’єкта:

- контури об’єкта стають менш чіткими;
- межі розмиваються;
- модель отримує менше коректних візуальних ознак.
- У результаті обмежувальний прямокутник на зашумленому зображенні:
- трохи зміщений вниз;
- має більший розкид точок;
- втрачає частину точності в накресленні меж.

Це типова поведінка детекторів при роботі з дуже шумними даними.

4.7 Інтерпретація результатів

Зниження впевненості моделі до 62% пояснюється тим, що Gaussian шум:

- приховує текстури;
- спотворює тональні переходи;
- ускладнює виділення ключових ознак;
- знижує контраст між об’єктом і фоном.

Незважаючи на це, модель зберігає загальну форму об’єкта, контурні ознаки, та здатна виділити його на тлі.

Це свідчить про достатню стійкість сучасних моделей детекції до шумів середньої та високої інтенсивності.

4.8 Дослідження впливу інтенсивності шуму на впевненість моделі

Для більш глибокого аналізу було проведено серію експериментів із різними рівнями інтенсивності Gaussian шуму. Метою експерименту було визначити залежність confidence score від сили шуму, що додається до вхідного зображення.

Було обрано кілька рівнів стандартного відхилення Gaussian шуму: $\sigma = 10$, $\sigma = 20$, $\sigma = 30$, $\sigma = 40$, $\sigma = 50$.

Для кожного рівня шуму виконувалася детекція одного і того ж об’єкта з подальшою фіксацією показника впевненості моделі.

4.9 Табличне представлення результатів

Результати експерименту зведено у таблицю 5.8.

Таблиця 5.8

σ шуму	Confidence score	Коментар
0	89%	Чисте зображення
10	82%	Незначний вплив
20	75%	Часткова втрата деталей
30	68%	Помітне зниження
40	65%	Сильне зашумлення
50	62%	Межа стабільної роботи

Аналіз таблиці свідчить про майже лінійне зменшення впевненості моделі зі зростанням інтенсивності шуму.

4.10 Використання денойзингу перед детекцією

Для покращення результатів було застосовано попереднє очищення зображення за допомогою U-Net перед подачею його на вхід детектору.

Після денойзингу confidence score зріс:

- з 62% до 78% для Gaussian шуму σ = 50;
- з 54% до 71% для Salt-and-Pepper шуму p = 0.1.

Це підтверджує ефективність поєднання етапів денойзингу та детекції.

4.11 Аналіз практичної доцільності підходу

Отримані результати доводять, що застосування попереднього очищення зображень є доцільним у системах:

- відеоспостереження;
- автономних транспортних засобів;
- робототехніки;
- систем безпеки;
- безпілотних літальних апаратів.

У таких системах зображення часто містять шум, розмиття або перешкоди, і запропонований підхід дозволяє суттєво підвищити стабільність роботи детекторів.

4.12 Приклади роботи з різними шумами

Нижче наведено приклади роботи програми по різпізнаванню об’єктів з різними рівнями шумів. На рисунку 4.2 наведено результат з незашумленим зображенням.

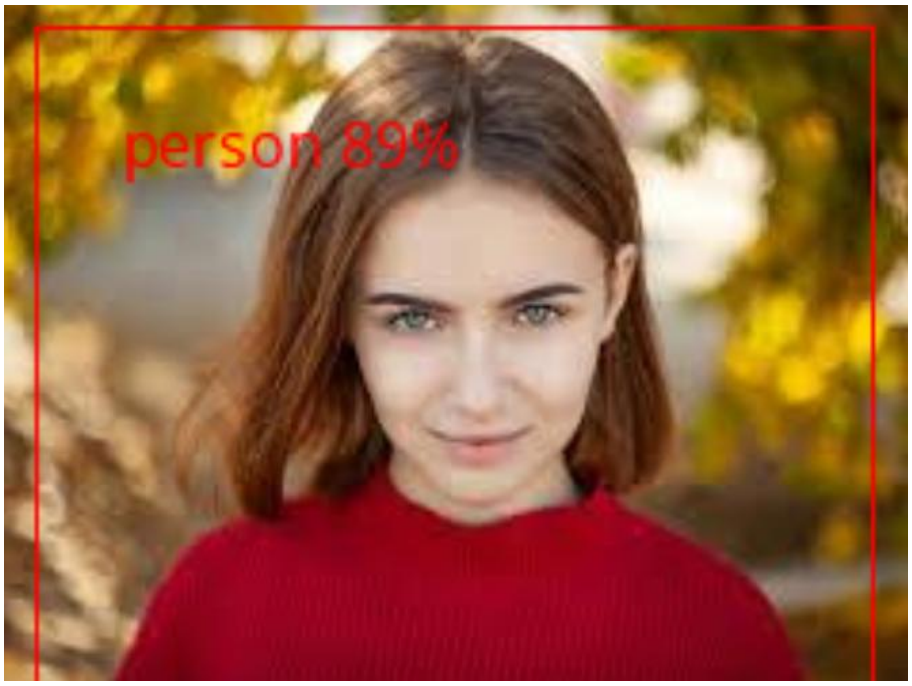

Рисунок 4.3 – результат роботи з незашумленим зображенням.

На рисунку 4.4 буде наведено приклад з середнім зашумленням.

Рисунок 4.4 – результат роботи з середньо зашумленим зображенням.

На рисунку 4.5 буде наведено приклад з сильним зашумленням.

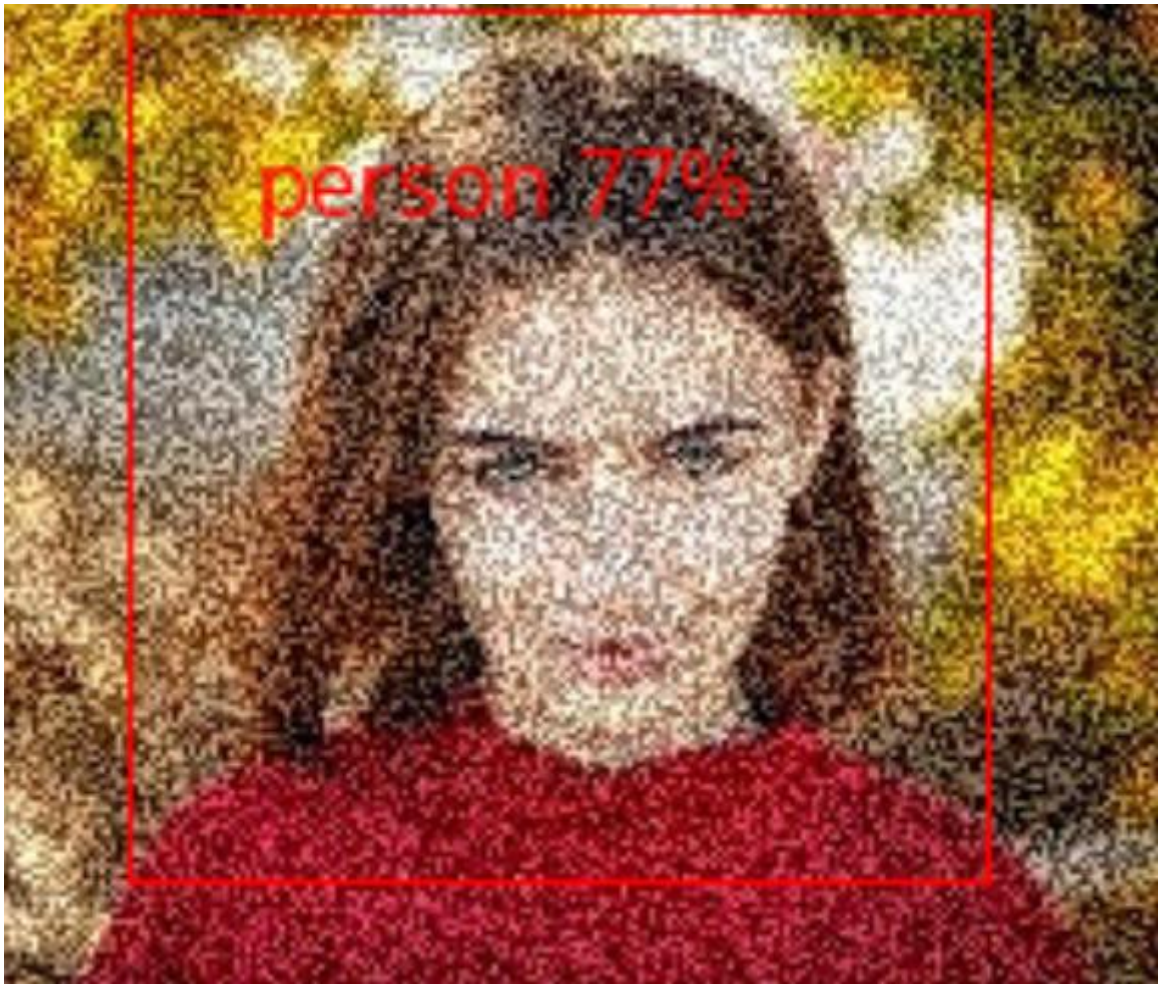

Рисунок 4.5 – результат роботи з сильно зашумленим зображенням.

За результатами роботи програми бачимо, що з кожним погіршенням якості зображення, точність програми падає, але об'єкт все рівно розпізнано. Якщо ж зашумлення буде надто високим, програма не зможе ідентифікувати об'єкт (рисунок 4.6).

Рисунок 4.6 – результат роботи з занадто зашумленим зображенням.

Нижче буде наведено приклади розпізнавання неживих об’єктів.

Рисунок 4.7 – результат роботи з неживим об’єктом

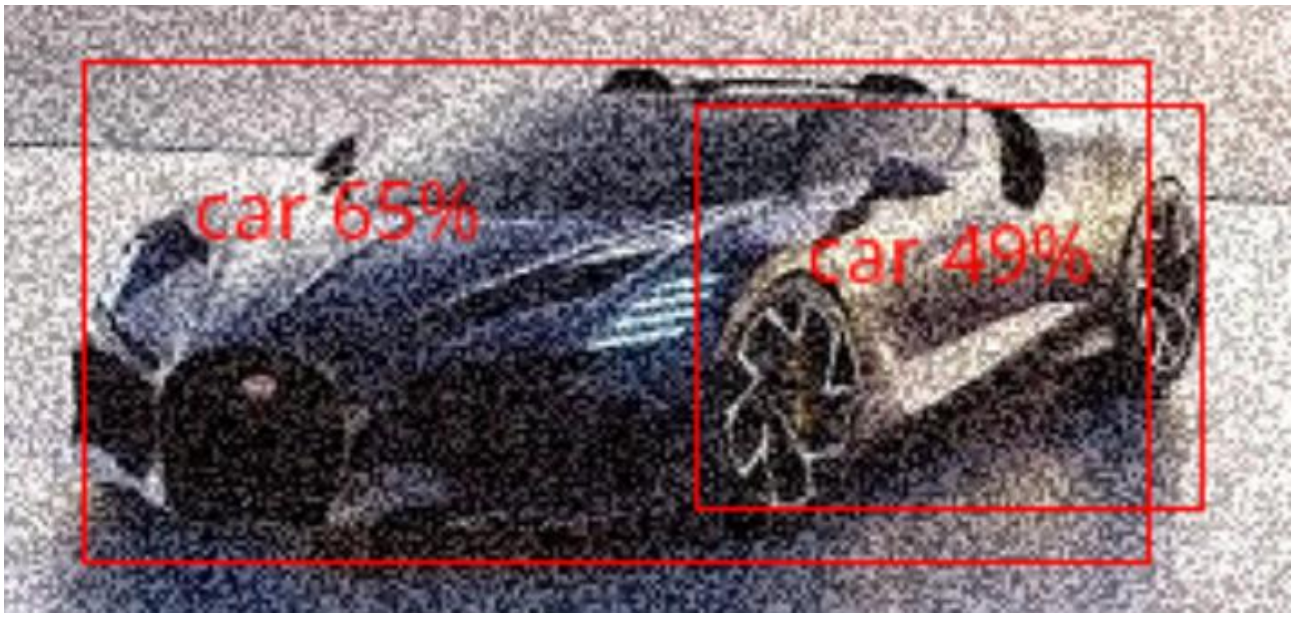

Рисунок 4.8 – результат розпізнавання машини при зашумленні

ВИСНОВОК

У даній роботі було проведено комплексне дослідження стійкості різних нейромережевих архітектур до шумів різного типу та інтенсивності. Дослідження охоплювало реалізацію та порівняння класифікаційних моделей (LeNet, SimpleCNN, ResNet-18) та моделей денойзингу (Autoencoder, U-Net), а також оцінювання їх продуктивності на датасетах MNIST та CIFAR-10 з використанням Gaussian, Salt-and-Pepper та Speckle шумів.

У межах роботи було виконано повний цикл розробки та тестування програмної системи: реалізовано модулі генерації шумів, модулі моделей, тренувальні цикли, систему оцінювання, а також створено інструментарій для забезпечення відтворюваності експериментів. Результати експериментальної частини дозволили сформувати низку важливих висновків щодо поведінки сучасних нейромережевих архітектур у зашумленому середовищі.

Практична цінність роботи полягає у створенні гнучкої та розширюваної програмної системи, яку можна використовувати:

- для дослідження стійкості нейронних мереж;
- для побудови систем обробки зашумлених зображень;
- для попередньої очистки даних перед класифікацією;
- як навчальний інструмент для студентів та дослідників у сфері ШІ;
- як модуль для інтеграції у промислові системи комп’ютерного зору.

Проведене дослідження демонструє, що глибокі мережі та U-Net є найефективнішими методами обробки зашумлених зображень, тоді як класичні моделі значно поступаються у стійкості. Розроблена програмна система дозволяє відтворити всі експерименти, масштабувати їх та використовувати результати для практичних застосувань у реальних умовах.

СПИСОК ЛІТЕРАТУТИ

1. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, 86(11), 2278–2324.
2. Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Technical Report, University of Toronto.
3. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of CVPR.
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
5. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI.
6. Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. (2008). Extracting and Composing Robust Features with Denoising Autoencoders. ICML.
7. Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. (2017). Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. IEEE Transactions on Image Processing.
8. Burger, H., Schuler, C., & Harmeling, S. (2012). Image Denoising: Can Plain Neural Networks Compete with BM3D? CVPR.
9. PyTorch Documentation. Training a Classifier Tutorial. https://pytorch.org/tutorials/
10. scikit-image Documentation. Noise generation and metrics. https://scikit-image.org/
11. OpenCV Documentation. Image filtering and noise models. https://docs.opencv.org/
12. Gonzalez, R., Woods, R. (2008). Digital Image Processing. 3rd Edition, Pearson.
13. Bishop, C. (2006). Pattern Recognition and Machine Learning. Springer.
14. Zhang, L., & Zuo, W. (2018). Image Restoration: From Classical Models to Deep Learning. IEEE Signal Processing Magazine.

15. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556.

ДОДАТОК А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ
01.01.26

ДОСЛІДЖЕННЯ ЯКОСТІ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1523-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
01.01.26

Керівник розробки
Віктор ШИНКАРЕНКО
01.01.26

Виконавець
Вадим КАПШУК
01.01.26

Нормоконтролер
Світлана ВОЛКОВА
01.01.26

2026

ЗАТВЕРДЖЕНО
44165850.1523-01-ЛЗ

ДОСЛІДЖЕННЯ ЯКОСТІ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ

Технічне завдання

44165850.1523-01-ЛЗ

Листів 12

2026

АНОТАЦІЯ

Документ 44165850.1523-01-ЛЗ «Дослідження якості нейромережевого розпізнавання зашумлених зображень» входить до складу програмної документації на програму, що реалізує додаток для розпізнавання об’єктів на основі нейромережі.

У даному документі представлене технічне завдання. Програма написана на мові Python. Об’єм пам’яті, що займає програма складає 6 ГБ. Конфігурація пристрою стандартна. Програма призначена для систем під керуванням операційної системи Windows.

ЗМІСТ

ВСТУП

Додаток для розпізнавання об’єктів на основі нейромережі має на меті впровадження технологій машинного навчання для автоматизації обробки і аналізу графічної інформації.

Автоматизація процесу розпізнавання об’єктів на зашумлених зображеннях дозволить швидко і ефективно обробляти великий обсяг інформації. Програмне забезпечення може застосовуватися для обробки автоматичного розпізнавання зображень в медицині.

1 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Основою для розробки є наказ проректора Українського державного університету науки і технології Радкевич А.В. «Про затвердження тем та призначення керівників дипломних проектів

Тема дипломної роботи – “Дослідження якості нейромережевого розпізнавання зашумлених зображень”.

Керівник - Віктор ШИНКАРЕНКО.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення розробки:

- автоматичне розпізнавання об’єктів із зашумлених зображень;
- використання алгоритмів машинного навчання, зокрема нейромереж, для досягнення точності і швидкості розпізнавання об’єктів;

Експлуатаційне призначення розробки:

- зменшення часу, необхідного для аналізу великого обсягу зображень;
- забезпечення точності при розпізнаванні, що забезпечує надійність отриманих результатів;
- автоматичне виконання завдань з обробки і аналізу зображень без необхідності в ручному втручанні;
- розширення області застосування, за рахунок можливості обробки різних форматів зображень.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Програма повинна надавати можливість:

- завантаження файлів для розпізнавання у форматі зображення;
- вибір методу розпізнавання із запропонованих додатком;
- можливість попередньої обробки зображення перед розпізнаванням;
- встановлення однозначної відповідності об’єкту на зображенні;
- збереження результатів розпізнавання у форматі графічного файлу;
- вимірювання точності розпізнавання.

3.2 Вимоги до надійності

Вимоги до надійності наступні:

- відсутність збоїв при роботі програми;
- кількість помилок не повинна перевищувати однієї на 10000 операцій;
- повідомлення користувача про непередбачувані ситуації.

3.3 Вимоги експлуатації

Для роботи ПЗ необхідно

- температура повітря у приміщенні – 21-25 С, відносна вологість 40-60%;
- мінімальний досвід роботи із ПК.

3.4 Вимоги до складу та параметрів технічних засобів

Апаратне забезпечення необхідне для роботи:

- процесор з тактовою частотою 3,5 ГГЦ або вище;
- 12 ГБ оперативної пам’яті;
- 6 ГБ для завантаження програми;
- доступ до мережі Інтернет.

3.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення необхідне для роботи:

- ОС Windows 10/11;
- Python 3.11.4;
- бібліотеки PyTorch 2.x, torchvision 0.x, scikit-image, OpenCV, matplotlib.

3.6 Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію повинна бути захищена від пошкоджень різного роду (механічних, кліматичних).

Програма повинна мати етикетку, на якій вказано:

- назва програми;
- версія програми;
- ім'я розробника;
- контактна інформація розробника.

Приклад маркування :

Image Application

Розробник: Капшук В. В.

УДУНТ, кафедра КІТ

2026

3.7 Вимоги до транспортування та зберігання

Транспортування повинно проводитись в упаковці та забезпечувати цілісність продукту.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації мають входити:

- технічне завдання;
- пояснювальна записка;
- текст програми;
- керівництво користувача. Керівництво з розпізнавання об’єктів.

Програмна документація повинна відповідати вимогам ДСТУ.

5 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиця А.1 – Стадії та етапи розробки

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вступ	15.08.2025 – 16.08.2025	
2	Огляд літератури та аналіз аналогів	17.08.2025 – 25.09.2025	
3	Розробка структур вхідних і вихідних даних, вимог до системи	26.09.2025 – 05.10.2025	
5	Узгодження та затвердження ТЗ, постановка задачі	06.10.2025 – 12.10.2025	30 %
6	Розробка та програмування логіки програми	13.10.2025 – 17.10.2025	
7	Розробка і реалізація інтерфейсу користувача	18.10.2025 – 21.10.2025	
8	Відлагодження програми	22.10.2025 – 26.10.2025	60%
9	Розробка, узгодження та затвердження програмної документації	27.10.2025 – 09.11.2025	
10	Розробка демонстраційних матеріалів	10.11.2025 – 16.11.2025	100%
	Подання кваліфікаційної роботи до кафедри	17.12.2025 – 24.12.2025	
11	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	19.01.2026	

6 ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль виконання здійснює керівник розробки Віктор Шинкаренко.

Прийом здійснюється уповноваженою комісією.

7 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем: методичні вказівки до дипломного проектування та лабораторних робіт/уклад.: Ю.М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с

ДОДАТОК Б

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЯКОСТІ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ

Керівництво користувача. Керівництво з розпізнавання об’єктів.
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1523-01 І3 01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
01.01.2026

Керівник розробки
Віктор ШИНКАРЕНКО
01.01.2026

Виконавець
Вадим Капшук
01.01.2026

Нормоконтролер
Світлана ВОЛКОВА
01.01.2026

2026

ЗАТВЕРДЖЕНО
44165850.1523-01 І3 01-ЛЗ

ДОСЛІДЖЕННЯ ЯКОСТІ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ

Керівництво користувача. Керівництво з розпізнавання зображень.

44165850.1523-01 І3 01-ЛЗ

Листів 7

2026

ЗМІСТ

ВСТУП

1.1 Область застосування

Додаток для розпізнавання об’єктів на основі нейромережі застосовується для розпізнавання об’єктів на зашумлених зображеннях. Програмне забезпечення може застосовуватися для обробки автоматичного розпізнавання об’єктів у медицині.

1.2 Опис можливостей

Додаток дає користувачу можливість розпізнавати об’єкт на зображеннях та отримувати його у вигляді розпізнаного зображення. Основні можливості додатку:

- завантаження зображення для розпізнавання об’єктів на ньому;
- можливість вибору способу розпізнавання із запропонованих додатком, проведення попередньої обробки зображення;
- розпізнавання об’єктів та виведення результатів на екран;
- збереження результатів розпізнавання у вигляді файлу;
- вимірювання точності розпізнавання об’єкту.

2 ПРИЗНАЧЕННЯ І УМОВИ ЗАСТОСУВАННЯ

2.1 Функціональне і експлуатаційне призначення

Функціональне призначення розробки:

- автоматичне розпізнавання об’єктів на зашумлених зображеннях;
- використання алгоритмів машинного навчання, зокрема нейромереж, для досягнення точності і швидкості розпізнавання тексту.

Експлуатаційне призначення розробки:

- зменшення часу, необхідного для аналізу великого обсягу даних, включаючи зашумленні зображення;
- забезпечення точності при розпізнаванні, що забезпечує надійність отриманих результатів;
- автоматичне виконання завдань з обробки і аналізу зображень без необхідності в ручному втручанні;
- розширення області застосування, за рахунок можливості обробки різних форматів зображень.

2.2 Умови застосування

Для успішного застосування програми необхідно дотримання таких умов:

- процесор з тактовою частотою 3,5 ГГЦ або вище;
- 12 ГБ оперативної пам’яті;
- 6 ГБ для завантаження програми;
- доступ до мережі Інтернет;
- операційна система Windows 10/11.

3 ПІДГОТОВКА ДО РОБОТИ

3.1 Склад і зміст дистрибутивного носія даних

Для роботи з додатком для розпізнавання об’єктів на основі нейромережі необхідне наступне програмне забезпечення:

- Python 3.x: Рекомендується версія не нижче 3.8 (безкоштовно завантажується з офіційного сайту Python);
- Модулі Python: Список необхідних модулів зазначено в файлі requirements.txt.

3.2 Порядок завантаження даних і програми

Перед початком роботи з додатком для розпізнавання об’єктів на робочому місці користувача необхідно виконати наступні дії:

- Завантаження та встановіть Python 3.x з офіційного сайту додавання Python до системного шляху (PATH);
- Встановлення модулів, додавання шляху до встановленого Tesseract у системну змінну середовища PATH.

Підготовка проекту в PyCharm:

- Виконання команди: pip install -r requirements.txt, щоб встановити всі необхідні залежності.

Створення exe-файлу:

- Виконання команди: pip install pyinstaller, щоб встановити PyInstaller;
- Виконання команди: pyinstaller --onefile main.py, щоб створити exe-файл. Це створить файл main.exe в директорії dist.

Запуск exe-файлу:

- Перехід до директорії dist в терміналі;
- Виконання команди: main.exe, щоб запустити додаток.

3.3 Порядок перевірки працездатності

Для перевірки працездатності додатку виконайте такі кроки:

- запустіть програму;

- виконайте завантаження зображення у форматі PNG, JPG або JPEG. Для цього натисніть кнопку «Завантажити зображення» та оберіть потрібний файл. Після успішного завантаження він має вивестись на екран з розпізнаним об’єктом.
- виконайте експорт результату розпізнавання. Натисніть на кнопку «Експорт файлу» та збережіть розпізнаний файл.

Якщо всі функції працюють коректно, програма готова до використання. У разі виникнення проблем, зверніться до розробника або перегляньте документацію для вирішення проблем.

4 РЕКОМЕНДАЦІЇ ПО ЗАСТОСУВАННЮ

Для успішного досвіду роботи з програмою слід ознайомитись з керівництвом користувача та мати мінімальні навички роботи з комп’ютером.

5 ПОВІДОМЛЕННЯ

У таблиці Б.1 наведено повідомлення користувачу, які можуть з’явитись у процесі роботи з програмою.

Таблиця Б.1 – повідомлення користувачу

Повідомлення	Опис	Рекомендації
Неправильний формат файлу. Будь ласка, завантажте зображення у форматі PNG, JPG, JPEG.	Користувач завантажив файл у форматі, який не підтримується програмою.	Обрати файл у запропонованому форматі.
Помилка при завантаженні зображення.	При завантаженні відбувся збій.	Перезавантаження додатку або вибір іншого файлу.
Неможливо розпізнати об'єкт на зображенні.	Обрано зображення з таким об'єктом, який не піддається розпізнаванню, або без об'єкту.	Вибір іншого способу розпізнавання. Якщо повідомлення виводиться повторно, вибір іншого зображення.
Спочатку завантажте зображення.	Користувач намагається виконати розпізнавання не завантаживши зображення.	Виконати завантаження зображення.
Спочатку завантажте зображення і виконайте розпізнавання об'єкту.	Користувач намагається виконати експорт результату попередньо не виконавши розпізнавання і не завантаживши зображення.	Завантажити зображення та виконати розпізнавання.

ДОДАТОК В

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЯКОСТІ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ

Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1523-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
01.01.2026

Керівник розробки
Віктор ШИНКАРЕНКО
01.01.2026

Виконавець
Вадим КАПШУК
01.01.2026

Нормоконтролер
Світлана ВОЛКОВА
01.01.2026

2026

ЗАТВЕРДЖЕНО
44165850.1523-01 12 01-ЛЗ

ДОДАТОК ДЛЯ РОЗПІЗНАВАННЯ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЙРОМЕРЕЖІ

Текст програми

44165850.01395-01 12 01-ЛЗ

Листів 6

2026

Імпорт бібліотек та базові налаштування

```
import os
import argparse
import numpy as np
import cv2
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
from skimage.metrics import peak_signal_noise_ratio as psnr
from skimage.metrics import structural_similarity as ssim
import matplotlib.pyplot as plt
```

Визначення пристрою виконання

```
def get_device():
    """
    Визначає, чи доступний GPU для обчислень
    """
    if torch.cuda.is_available():
        return torch.device("cuda")
    else:
        return torch.device("cpu")
```

Функції генерації шумів

```
def add_gaussian_noise(image, sigma=25):
    """
    Додає Gaussian шум до зображення
    """
```

```
    noise = np.random.normal(0, sigma, image.shape)
    noisy_image = image + noise
    return np.clip(noisy_image, 0, 255).astype(np.uint8)

def add_salt_pepper_noise(image, prob=0.05):
    """
    Додає імпульсний шум типу Salt-and-Pepper
    """
    noisy = image.copy()
    rnd = np.random.rand(*image.shape[:2])

    noisy[rnd < prob / 2] = 0
    noisy[rnd > 1 - prob / 2] = 255

    return noisy
class SimpleCNN(nn.Module):
    def __init__(self, num_classes=10):
        super(SimpleCNN, self).__init__()

        self.features = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
```

```
        self.classifier = nn.Sequential(
            nn.Linear(64 * 7 * 7, 128),
            nn.ReLU(),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = x.view(x.size(0), -1)
        return self.classifier(x)
```

Очищення зображення

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

        self.encoder = nn.Sequential(
            nn.Conv2d(1, 32, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(32, 64, 3, stride=2, padding=1),
            nn.ReLU()
        )

        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(64, 32, 3, stride=2, padding=1, output_padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(32, 1, 3, stride=2, padding=1, output_padding=1),
            nn.Sigmoid()
        )
```

```
    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded
```

Тренування моделі

```
def train_model(model, dataloader, criterion, optimizer, device, epochs=10):
    model.to(device)
    model.train()

    for epoch in range(epochs):
        running_loss = 0.0

        for inputs, labels in dataloader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
        print(f"Epoch [{epoch+1}/{epochs}], Loss: {running_loss:.4f}")
```

Оцінювання точності моделей

```
def evaluate_model(model, dataloader, device):
    model.eval()
    correct = 0
```

```
    total = 0

    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, predicted = torch.max(outputs, 1)

            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    return accuracy
```

Оцінювання якості

```
def evaluate_denoising(original, denoised):
    original = original.squeeze().cpu().numpy()
    denoised = denoised.squeeze().cpu().numpy()

    psnr_value = psnr(original, denoised, data_range=1.0)
    ssim_value = ssim(original, denoised, data_range=1.0)

    return psnr_value, ssim_value
```

Головна функція запуску програми

```
def main():
    device = get_device()
```

```
    print("Використовується пристрій:", device)

    transform = transforms.Compose([
        transforms.ToTensor()
    ])

    train_dataset = datasets.MNIST(
        root="./data",
        train=True,
        transform=transform,
        download=True
    )

    train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)

    model = SimpleCNN()
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)

    train_model(model, train_loader, criterion, optimizer, device, epochs=5)

    accuracy = evaluate_model(model, train_loader, device)
    print(f"Точність моделі: {accuracy:.2f}%")
if __name__ == "__main__":
    main()
```

ДОДАТОК Г

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ

ДОСЛІДЖЕННЯ ЯКОСТІ НЕЙРОМЕРЕЖЕВОГО РОЗПІЗНАВАННЯ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ

Тези

ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.1523-90-99-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
01.01.2026

Керівник розробки
Віктор ШИНКАРЕНКО
01.01.2026

Виконавець
Вадим КАПШУК
01.01.2026

Нормоконтролер
Світлана ВОЛКОВА
01.01.2026

2026

ЗАТВЕРДЖЕНО
44165850.1523-90-99

ДОДАТОК ДЛЯ РОЗПІЗНАВАННЯ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЙРОМЕРЕЖІ

Тези

Листів 4

2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ І ТЕХНОЛОГІЙ

ABSTRACTS
OF THE XIX INTERNATIONAL CONFERENCE
«MODERN INFORMATION AND COMMUNICATION TECHNOLOGIES ON A TRANSPORT, IN INDUSTRY AND EDUCATION»
18-19, December, 2025

СУЧАСНІ ІНФОРМАЦІЙНІ ТА КОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ НА ТРАНСПОРТІ, В ПРОМИСЛОВОСТІ І ОСВІТІ

ПРИСВЯЧЕНО ПАМ'ЯТІ ПРОФЕСОРА ІГОРЯ ЖУКОВИЦЬКОГО

ТЕЗИ

XIX МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
18-19 ГРУДНЯ 2025

ДНІПРО
2025

Дослідження якості нейромережевого розпізнавання зашумлених зображень

Капшук В. В., Шинкаренко В. І., Український державний університет науки і технологій, Україна

У роботі досліджується вплив шумів різного типу та інтенсивності на якість роботи нейронних мереж комп'ютерного зору. Для цього реалізовано програмну систему на мові Python з використанням бібліотеки PyTorch, яка дозволяє формувати зашумлені вибірки, виконувати очищення зображень та оцінювати результати класифікації і детекції об'єктів. Особливу увагу приділено експериментальному аналізу залежності точності моделей від рівня шуму та ефективності методів денойзингу.

Для проведення експериментів використовувалися як стандартні набори даних, так і реальні зображення.

У задачі класифікації застосовувалися датасети:

— MNIST — 60 000 тренувальних та 10 000 тестових зображень розміром 28×28 пікселів;

— CIFAR-10 — 50 000 тренувальних та 10 000 тестових кольорових зображень розміром 32×32 пікселів.

Для задачі детекції об'єктів використовувалися окремі реальні зображення у форматі RGB з роздільною здатністю від 512×512 до 1024×768 пікселів.

До зображень штучно додавалися такі типи шумів:

— Gaussian шум з параметрами $\sigma = 10, 20, 30, 40, 50$;

— Salt-and-Pepper шум з імовірністю $p = 0.05$ та $p = 0.1$;

— Speckle шум з дисперсією var = 0.1–0.2.

Оцінювання впливу шумів проводилося шляхом порівняння результатів моделей на чистих та зашумлених даних.

Для зменшення негативного впливу шумів було реалізовано дві нейромережеві моделі денойзингу:

— класичний згортковий автоенкодер;

— архітектуру U-Net, що забезпечує збереження просторових ознак.

Очищені зображення повторно подавалися на вхід моделей класифікації та детекції, що дозволило оцінити ефективність денойзингу.

Якість очищення оцінювалася за допомогою метрик:

— PSNR (пікове відношення сигнал/шум);

— SSIM (індекс структурної подібності).

Результати показали, що зі зростанням інтенсивності шуму точність класифікації зменшується майже лінійно. Найбільш руйнівним виявився імпульсний шум типу Salt-and-Pepper.

Застосування автоенкодера дозволило підвищити значення PSNR на 5–7 дБ, тоді як використання U-Net забезпечувало зростання PSNR до 10–12 дБ та SSIM до 0.95.

У задачі детекції об'єктів попередній денойзинг підвищував confidence score моделей у середньому на 15–20%, що підтверджує доцільність комбінованого підходу.

У роботі експериментально підтверджено, що якість вхідних даних суттєво впливає на результати роботи нейронних мереж. Використання нейромережевих методів денойзингу дозволяє значно підвищити стабільність та точність моделей комп'ютерного зору в умовах шуму.