**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**Радіотехнічний факультет**
**Кафедра прикладної радіоелектроніки**

«На правах рукопису»
УДК _____

До захисту допущено:
В. о. зав. кафедрою

Андрій МОВЧАНЮК
«___»_____2024 р

# Магістерська дисертація

**на здобуття ступеня магістра**

**за освітньо-професійною програмою «Інтелектуальні технології радіоелектронної техніки»**

**за спеціальністю 172 «Телекомунікації та радіотехніка» на тему: «Використання нейромережі для реально-часової оцінки та фільтрації шуму відеосигналу в каналі відеозв'язку»**

Виконав:

студент 2 курсу, групи РЕ-21мп

Герасимчук Олександр Володимирович _____
(підпис)

Керівник:

Доцент, к.т.н.,

Сушко Ірина Олександрівна _____
(підпис)

Рецензент

Доцент, к.т.н.,

Катін Павло Юрійович _____
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024 року

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Радіотехнічний факультет**

**Кафедра прикладної радіоелектроніки**

Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інтелектуальні технології радіоелектронної техніки»

В.о.зав. кафедрою

Андрій МОВЧАНЮК

«___»_____2023 р.

**ЗАВДАННЯ**

**на магістерську дисертацію студента**

**Герасимчук Олександр Володимирович**

1.Тема дисертації «Використання нейромережі для реально-часової оцінки та фільтрації шуму відеосигналу в каналі відеозв'язку»
науковий керівник дисертації Сушко Ірина Олександрівна, доцент, к.т.н.,
затверджені наказом по університету від «09» листопада 2023р.№ 5206-с

2. Термін подання студентом дисертації 11 січня 2024 року.
3. Об'єкт дослідження методи та алгоритми оброблення відеосигналів

4. Вихідні дані Відеосигнали з реальних каналів зв'язку, що містять різний рівень ураження шумами та перешкодами.

5. Перелік завдань, які потрібно розробити Аналіз сучасних підходів до обробки відеосигналів, вивчення принципів роботи нейромереж у контексті видалення шуму, створення технології для відеофільтрації шуму в режимі реального часу з використанням нейромереж

6. Орієнтовний перелік графічного (ілюстративного) матеріалу презентація

7.Орієнтовний перелік публікацій _____

8.Дата видачі завдання 01 вересня 2023 року.

Календарний план

| № з/п | Назва етапів виконання Магістерської дисертації | Термін виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Отримання теми магістерської дисертації | 01.09.2023 | Виконано |
| 2 | Розробка плану магістерської дисертації | 02.09.2023-10.09. 2023 | Виконано |
| 3 | Початок збору інформації для дослідження | 10.09. 2023-24.09.2023 | Виконано |
| 4 | Написання першого розділу | 25.09.2023 -15.10.2023 | Виконано |
| 5 | Написання другого розділу | 16.10.2023 - 01.11.2023 | Виконано |
| 6 | Створення нейромережевого алгоритму | 02.11.2023 - 30.11.2023 | Виконано |
| 7 | Написання третього розділу | 01.12.2023 - 14.12.2023 | Виконано |
| 8 | Написання стартап-проєкту | 15.12.2023 - 19.12.2023 | Виконано |
| 9 | Оформлення магістерської дисертації | 20.12.2023-09.01.2024 | Виконано |

Студент Олександр ГЕРАСИМЧУК

Науковий керівник Ірина СУШКО

# АНОТАЦІЯ

Метою магістерської дисертації є розробка та впровадження нейромережі для реально-часової оцінки та фільтрації шуму в відеосигналах, що передаються бездротовими мережами.

В епоху стрімкого розвитку інформаційних технологій, бездротова передача даних, зокрема відеопотоків, стає все більш поширеною. Це особливо актуально для таких систем, як безпілотні літальні апарати та сенсорні мережі. Однак, під час бездротової передачі відеопотік може бути уражений завадами, що може знизити якість зображення та ускладнити подальшу обробку. Це особливо актуально в умовах бойових дій, де відеопотоки, наприклад, з безпілотних літальних апаратів, можуть бути уражені засобами радіоелектронної боротьби, що негативно впливає на ефективність цих апаратів. Використання нейромереж для фільтрації шуму може значно покращити якість вихідного зображення.

В магістерському дослідженні було проведено аналіз сучасних методологій обробки відеосигналів, вивчено принципи роботи нейромереж у контексті фільтрації шуму та розроблено систему для реально-часової оцінки та фільтрації шуму в відеосигналах за допомогою нейромереж.

В результаті було створено систему, яка здатна оцінювати та фільтрувати шум у вихідних відеосигналах в режимі реального часу за допомогою нейромереж. Ця система була реалізована на Python за допомогою бібліотеки Pytorch.

*Перелік ключових слів*: згорткові нейронні мережі, глибоке навчання, алгоритми фільтрації, обробка зображень, шуми, навчання без вчителя.

Пояснювальна записка: має осбяг 103 сторінок, 32 ілюстрацій, 24 таблиць, 36 бібліографіних посилань, кількість додатків 1.

## ANNOTATION

Neural network development and implementation for real-time assessment and noise filtering in video signals transmitted through wireless networks are this master’s work purpose.

In the era of rapid development in information technology, wireless data transmission, particularly video streaming, becomes increasingly prevalent. This is especially relevant for unmanned aerial vehicles and sensor networks. However, during wireless video transmission, the video stream may be affected by interference, which can degrade image quality and complicate further processing. This is particularly crucial in combat situations, where video streams from unmanned aerial vehicles, for example, may be susceptible to electronic warfare, negatively impacting the efficiency of these devices. The use of neural networks for noise filtering can significantly enhance the quality of the output image.

The master's research involved an analysis of contemporary video signal processing methodologies, an examination of the basics of neural networks in the context of noise filtering, and the development of a system for real-time assessment and noise filtering in video streams using neural networks.

As a result, created system is capable to assess and filter noise in output video streams in real-time using neural networks. This system is implemented in Python using the Pytorch library.



*Keywords*: convolutional neural networks, deep learning, filtering algorithms, image processing, noise, unsupervised learning.


The thesis consists of 103 pages, including 32 illustrations, 24 tables, and 36 bibliographic references, number of applications 1.

## ПЕРЕЛІК СКОРОЧЕНЬ

CNN – Convolutional neural network, згорткова нейронна мережа

DNN – Deep neural networks, глибокі нейронні мережі

GAN – Generative adversarial network, генеративна змагальна мережа

GPU – Graphics processing unit, графічний процесор

CPU – Central processing unit, центральний процесор

PSNR – Peak signal to noise ratio, пікове відношення сигнал шум

MSSIM – Mean structural similarity, середній індекс структурної подібності

AWGN – Additive White Gaussian Noise, адитивний білий гауссовий шум

**ЗМІСТ**

# ВСТУП

**Актуальність теми дослідження.** З розвитком технологій, велика кількість даних генерується та передається в режимі реального часу бездротовими засобами. Одним з головних джерел таких даних є відеопотік, джерелами якого можуть бути камери спостереження, камери встановлені на безпілотних літальних чи наземних апаратах, супутникове телебачення.

Проблему відеопотоку, що передається бездротовими мережами, є шуми, які можуть знижувати якість зображення та ускладнювати подальшу обробку даних. Ці шуми можуть бути викликані нестабільністю підключення, зіткненням з іншими сигналами, перешкодами на шляху передачі сигналу, електромагнітними перешкодами, а також іншими факторами. Оскільки відеопотік є важливим джерелом інформації для багатьох галузей, включаючи безпеку, медицину, транспорт та багато інших, важливо мати ефективні методи фільтрації шумів у реальному часі, щоб забезпечити якість даних та точність аналізу. Актуальною практичною є покращення зображення, що передається від безпілотних літальних апаратів під час бойових дій в умовах постановки перешкод засобами радіоелектронної боротьби.

Використання нейромереж для фільтрації відеопотоку від шумів може допомогти покращити якість зображення та забезпечити можливість подальшої обробки даних без втрати інформації. Крім того, це може допомогти зменшити кількість помилок та неправильних рішень, які можуть бути прийняті на основі неправильної інформації, отриманої з відеопотоку.

Проблемою сучасних алгоритмів фільтрації відеопотоку від шумів є їх часові затрати та потреба в значних обчислювальних ресурсах. Алгоритм обробки відеопотоку може вимагати виконання численних операцій, таких як кореляція або згортка, з великими об'ємами даних на кожному кроці обробки. Це може призвести до затримок при обробці відеопотоку, що не допустимо для багатьох систем в режимі реального часу, оскільки відеопотік повинен оброблятися з максимально можливою швидкістю, обмеження з приводу часу

виконання може робити більшість алгоритмів непридатними для використання в таких системах.

Враховуючи обмеження сучасних алгоритмів фільтрації відеопотоку від шумів, виникає потреба в розробці нових алгоритмів, які можна буде ефективно використовувати в режимі реального часу.

***Взаємозв'язок дисертації з науковими дослідженнями.*** Зв'язок даної роботи з науковими програмами, планами та актуальними темами виявляється через актуальність та перспективність її досліджень в контексті сучасних вимог та викликів у галузі обробки відеоданих. Дисертація була розроблена відповідно до наукових досліджень, що проводяться на кафедрі прикладної радіоелектроніки КПІ ім. Ігоря Сікорського. Робота виконана в рамках наукової та навчально-методичної програми кафедри.

***Мета та завдання дисертаційної роботи.*** Метою даної роботи є розробка та налаштування нейромережевих методів для фільтрації відеопотоку, які б забезпечували покращення якості зображення та ефективну обробку даних у режимі реального часу.

Для досягнення цієї мети, в рамках магістерської роботи, передбачено наступні завдання:

1. Провести огляд існуючих методів та підходів до фільтрації відеопотоку та виявити їхні переваги та обмеження.
2. Розробити архітектуру нейронної мережі для фільтрації відеопотоку, враховуючи вимоги та характеристики роботи з відеоданими.
3. Реалізувати розроблену архітектуру на мові програмування Python з використанням бібліотеки PyTorch, забезпечивши її функціональність та ефективність у фільтрації відеопотоку в реальному часі.
4. Підготувати набір даних для тренування та тестування нейронної мережі, враховуючи різні види шуму та перешкод, які можуть виникнути в реальних умовах.
5. Провести навчання нейронної мережі на підготовленому наборі даних та налаштувати її для оптимальної фільтрації відеопотоку.

6. Провести експериментальне тестування розробленої системи фільтрації в реальному часі та оцінити її продуктивність та ефективність за різними умовами.

7. Зробити аналіз отриманих результатів та зробити висновки щодо можливостей та перспектив використання нейронних мереж для фільтрації відеопотоку в режимі реального часу на мові Python.

***Об'єкт дослідження.*** Методи та алгоритми фільтрації відеопотоку від шумів у режимі реального часу, зокрема з використанням нейронних мереж.

***Предмет дослідження.*** Методи та програмні засоби для фільтрації шуму в відеопотоках за допомогою нейромереж.

# Розділ 1. ВИДИ ШУМІВ. МЕТОДИ ОЦІНКИ ТА ПРОТИДІЇ ШУМАМ В ВІДЕОПОТОЦІ

Коли відеопотік чи зображення отримано через процес цифрового кодування, вони зазвичай містять шум, який може спотворити їх. Таким чином, обробка зображень, що має на меті покращити їх візуальне сприйняття людиною та вирішити завдання, пов’язані з машинним сприйняттям зображень, є важливим напрямком сучасної роботи. Це вимагає неперервного вдосконалення методів, які можуть бути застосовані для виконання цих завдань.

Шуми мають різну природу виникнення та вплив на відеопотік. Через це важливою задачею для створення систем боротьби з шумами є правильний вибір методів протидії. Так методи, які використовуються для покращення відео, знятого безпілотним літальним апаратом (БПЛА), можуть бути неефективними для відео, знятого в умовах низької освітленості. БПЛА часто використовуються в умовах, де є багато рухомих об’єктів знаходячись під впливом шумів від засобів радіоелектронної боротьби (РЕБ), тому методи фільтрації шуму для таких випадків повинні бути здатними ефективно обробляти цю динаміку та протидіяти шумам РЕБ.

Іншою сферою застосування таких алгоритмів є веб-камери для відеоконференцій. Вони часто працюють в умовах низької освітленості та мають обмежене апаратне забезпечення, що призводить до виникнення у відеопотоці термічного та Пуассонівського шумів. У цьому випадку потребуються методи фільтрації шуму, які можуть працювати ефективно при низькому рівні освітленості та обмежених обчислювальних ресурсах.

Серед поширених на сьогодні методів боротьби з шумами у відеопотоці можна виділити:

1) Фільтрація по простору та часу, їх комбінування

2) Фільтрація в частотній області

3) Використання нейронних мереж

Враховуючи вищеописане, постає важливість проаналізувати, як шуми впливають на зображення та методи оцінки шумів. Через різну природу впливу на відеопотік різних типів шумів та широкий спектр існуючих методик боротьби з ними, важливо обрати найбільш підходящий для конкретного сценарію застосування.

### 1.1. Види шумів в каналі відеозв'язку

Шум [1] у відеосигналі є невід'ємною частиною процесу передачі та обробки відеопотоку. Шум – це результат взаємодії різних фізичних та технічних факторів під час передачі, обробки та відтворення зображень. Серед основних таких факторів: недосконалість технічних засобів формування зображень, що викликає теплові шуми та оптичні завади, через передачу зображень по каналах зв'язку з завадами, які призводять до спотворення окремих пікселів та блоків інформації. Шум може виникати на будь-якому етапі передачі та обробки відеосигналу, від початкового захоплення до його відтворення чи перетворення. При цифровому кодуванні сигналу можуть виникати шуми дискретизації та квантування. Все це впливає на якість відео та може ускладнювати подальшу обробку та аналіз зображень. Шум може спотворити зображення, знизити якість відео та ускладнити подальшу обробку зображень, таку як сегментація та розпізнавання.

Перед розробкою методів фільтрації та усунення шуму важливо розрізнити основні типи шуму, які можуть виникати у каналах відеозв'язку. Кожен тип шуму має свої особливості та вимагає індивідуального підходу та методів фільтрації.

***Адитивний білий гаусівський шум*** [1,2] (Additive white Gaussian noise, AWGN) євидом білого шуму, що виникає в каналі передачі інформації. Білий шум — це сигнал, в якому відліки не корелюють один з одним. Білий

гаусівський шум, який виникає, зокрема, при низькій якості прийому сигналу і описується функцією густини розподілу амплітуд:

$$p(d) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}} \quad (1.1)$$

AWGN характеризується рівномірною спектральною щільністю, нормальним розподілом та адитивним впливом, тобто шум додається до корисного сигналу. Яскравість пікселів у зображенні, на яке накладається гаусівський шум, визначається стохастичною величиною, яка розподілена згідно за нормальним законом розподілу ймовірностей.

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(z-\mu)^2}{2\sigma^2}} \quad (1.2)$$

де z — яскравість пікселів на зображенні, μ — це середнє значення яскравості пікселів у цьому зображенні, σ — це стандартне відхилення яскравості пікселів у зображенні, $\sigma^2$ — це дисперсія яскравості пікселів.

Гаусівський шум є найбільш поширеним типом завад. У цифрових зображеннях він часто виникає внаслідок декількох основних факторів. Одним з таких джерел є сенсорний шум, який може виникати через погане освітлення або високу температуру під час фотографування або зйомки відео. Це може призводити до випадкових відхилень у значеннях яскравості пікселів у зображенні, що створює шумові ефекти. Приклад зображення ураженого AWGN шумом наведено на рисунку 1.1.

Рисунок 1.1 — Вплив гаусівського шуму на зображення

При обробці цих цифрових зображень гаусівський шум може бути пом'якшений за допомогою фільтрації. Однак при розмиванні зображення може виникнути небажаний результат — туманні межі та деталі зображення. Ці межі та деталі відповідають високочастотним складовим зображення, і фільтр може "видаляти" їх у процесі зниження шуму.

***Імпульсний шум.*** Імпульсний шум[1,2], відомий як "шум сіль і перець", це вид шуму, який виникає на цифровому зображенні через випадкові викиди (або імпульси) яскравості. Імпульсний шум призводить до того, що окремі пікселі стають дуже яскравими (солоними) або дуже темними (перцевими) порівняно з оточуючими пікселями. Часто такий тип шуму зустрічається при передачі зображень через аналогові канали зв'язку та в пристроях з помилковою комутацією.

В імпульсному (біполярному) шумі, процес додавання шуму полягає у тому, що значення яскравості кожної точки на зображенні з імовірністю $P_s$ (де $P_s = P_a + P_b \leq 1$) може бути замінено значенням шуму. При цьому яскравість пікселів визначається двома позитивними числами - $P_a$ та $P_b$. Яскравість будь-якого пікселя може бути замінена значенням "a" з імовірністю $P_a$, значенням "b" з імовірністю $P_b$ і залишитися без змін з імовірністю 1 - $P_a$ - $P_b$.

Розподіл імовірностей може бути представлений за допомогою дельта-функції Дірака у вигляді

$$p(z) = P_a\delta(\mathrm{z} - \mathrm{a}) + P_b * \delta(\mathrm{z} - \mathrm{b}) \qquad (1.3)$$

де δ — дельта-функція Дірака.

Якщо інтенсивність пікселя b перевищує a, то піксель з такою яскравістю b відображається як світла точка на зображенні, в той час як піксель з інтенсивністю a відображається як темна точка. Якщо імовірність $P_a$ або $P_b$ дорівнює нулю, то такий імпульсний шум визначається як уніполярний. Проте, якщо жодна з імовірностей не дорівнює нулю, то імпульсний шум має вигляд краплин солі та перцю, розсіяних по зображенні.Приклад такого зображення наведено на рисунку 1.2.


Рисунок 1.2 — Вплив імпульсного шуму на зображення

При перетворенні зображення в цифровий формат зазвичай обмежують діапазон значень яскравості. Тому зазвичай припускають, що значення a та b дорівнюють мінімальному та максимальному можливим значенням, які взагалі можуть бути на цифровому зображенні. Для 8-бітових зображень це означає, що a=0 (чорний, "перець"), b=255 (білий, "сіль").

***Дробовий шум (шум Пуассона).*** [1] Шум Пуассона отримав свою назву через те що описується за допомогою розподілу Пуассона. Цей шум характерний для детекції світла в цифровій обробці зображень.

Під час випромінювання світла лазером, фотони випускаються випадковим чином, та залежить від освітленості. На початковому етапі, кількість фотонів, необхідних для формування зображення, може бути значною, але з часом ця кількість поступово зменшується. У ситуації, коли інтенсивність лазера є низькою, кількість фотонів може стати настільки малою, що відносні коливання кількості фотонів і, відповідно, яскравості, стають помітними. Ці коливання розглядаються як дробовий шум. Приклад зображення ураженого шумом Пуассона наведено на рисунку.1.3.

Ймовірність кожного пікселя (m, n) спостереження y визначається наступним чином:

$$\forall m,n \qquad y(m,n) \sim P\big(x(m,n)\big). \tag{1.4}$$

де, P — розподіл Пуассона, що визначається як:

$$p(k) = \frac{\lambda^k}{k!} e^{-\lambda} \quad (1.5)$$

де, k — це кількість подій, λ — математичне очікування випадкової величини (середня кількість подій протягом фіксованого часового інтервалу).

Ця модель використовується в разі отримання дуже низької кількості фотонів, наприклад, у астрономії.


Рисунок 1.3 — Вплив дробового шуму на зображення

Важливо враховувати, що при значних значеннях середнього числа подій λ розподіл Пуассона наближається до гауссового розподілу. Це означає, що адитивний білий гауссівський шум стає хорошою моделлю шуму Пуассона при достатньому числі отриманих фотонів.

***Мультиплікативний шум*** [1] це вид шуму, який виникає на зображеннях і впливає на їх якість та чіткість шляхом зміни яскравості або контрастності пікселів. Під дією мультиплікаційного шуму оригінальні фрагменти зображення множаться на шумовий сигнал, що відрізняє його від адитивного шуму, де шум додається безпосередньо до значень пікселів. Приклад такого зображення наведено на рисунку 1.4.

Мультиплікативний шум виникає через випадкові зміни освітлення, невірний баланс білого, прозорі або матові поверхні об'єктів, помилки в оптиці зображувального пристрою тощо. Він проявляється як випадкові зміни яскравості чи контрасту областей зображення, що може зробити зображення менш чітким та менш інформативним.

Рисунок 1.3 — Вплив мультиплікативного шуму на зображення

Існує обмежена кількість моделей відновлення зображень, які можуть ефективно усунути мультиплікативний шум. Багато з них базуються на варіаційних методах, які можуть бути складними для реалізації. Також одним з поширених підходів до обробки мультиплікативного шуму є його перетворення в адитивний шум за допомогою логарифмічного перетворення. Проте, цей процес може призвести до додаткових складностей, таких як втрата деталей зображення виникнення “ефекту сходинок” або розмиття гострих країв.

### 1.2. Методи оцінки зашумленості відеопотоку

В контексті побудови алгоритмів боротьби з шумами у відеопотоці є оцінку їм рівня на вході та виході алгоритму. При застованні методів протидії на основі глибокого навчання використанні таких оцінок замість функції втрат дозволяє підвищити якість навчання мережі.

Зазвичай якість відео оцінюється за допомогою суб’єктивних та об’єктивних показників. Суб’єктивна оцінка якості базується на сприйнятті глядача і визначається шляхом експертної оцінки та обчислення середнього балу MOS (Mean Opinion Score). Об’єктивну якість можна виміряти за допомогою різних метрик які можна розділити на наступні класи:

- Еталонні (Full Reference, FR) передбачають наявність оригінального вихідного відеопотоку, який є опірним при порівнянні, оскільки він не має шуму і має ідеальну якість.

- Нееталонні (No Reference, NR) передбачають, що під час отримання оцінки якості відеопотоку опорний або еталонний потік відсутній. Ці метрики є найскладнішими у реалізації і часто спрямовані на конкретний вид спотворення.
- Псевдоеталонні (Reduced Reference, RR) передбачають, що деяка частина інформації про еталонний відеопотік присутня разом зі зашумленим, причому кількість цієї інформації значно менше, що потрібна для еталонного.

Для задач оцінки методів боротьби з шумами та глибокого навчання найкраще підходять FR метрики. Серед найпоширеніших FR метрик:

***Пікове відношення сигнал/шум*** Peak Signal to Noise Ratio (PSNR)[3] вказує на співвідношення між сигналом і супутнім шумом, який вводиться в його представлення. Розраховується як:

$$PSNR = 10log_{10}\frac{(2^n-1)^2}{\sqrt{MSE}} \quad (1.6)$$

де MSE визначається за формулою:

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{i,j} - y_{i,j})^2 \quad (1.7)$$

***Індекс структурної схожості*** [4] Structural Similarity Measure (SSIM) є одним із способів вимірювання подібності між двома зображеннями.

Індекс структурної схожості (IM-індекс) є методом повного порівняння, він вимірює якість на основі вихідного зображення. Таким чином, для оригінального зображення "x" і спотвореного зображення "y", IM-індекс розраховується як:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2)(\sigma_x^2+\sigma_y^2)} \quad (1.8)$$

де, $\mu(x)$ — середнє значення зображення x; $\mu(y)$ — середнє значення зображення y; $\sigma(x)$ та $\sigma(y)$ — середньоквадратичне відхилення для зображення x та y; $\sigma(x,y)$ — коваріація; $c_1$ и $c_2$ — поправочні коефіцієнти.

***VMAF (Video Multi-method Assessment Fusion)*** [5] вимірює рівень якості відеопотоку на основі схожості між оригінальним і спотвореним відео.

VMAF є метрикою повного порівняння, що враховує багато аспектів якості, зокрема:

- Індекс візуальної достовірності -побудова цього індексу ґрунтується на моделюванні джерела оригінального зображення, спотвореного зображення та візуальних спотворень, які сприймаються людиною. УЕ відображає втрату точності інформації
- Метрика втрати деталізації - вимірює втрату деталей, які відволікають увагу користувачів.
- Миттєва середня різниця освітленості MCPO — це середня величина відмінності освітленості між кадрами.
- Антишумове відношення сигнал шум — це відношення сигналу до шуму, що використовується для оцінки антишумових характеристик сигналу.

### 1.3. Методи фільтрації за простором

Просторова фільтрація використовує техніки зниження шуму зображення, які застосовуються до послідовних кадрів відео. Цей метод використовує розбиття зображення на прямокутні області –маски, для охоплення регіону зображення, де виконується ця попередньо визначена операція. Маска - це, матриця, яка використовується для обчислення нових інтенсивностей пікселів зображення з оригінального.

Просто згладити зображення для успішної фільтрації недостатньо. Більш сучасні методи не просто згладжують зображення, але й намагаються відновити втрачену інформацію, якщо це необхідно. Наприклад, у зображенні, отриманому за допомогою цифрової дзеркальної камери, нам часто потрібно зберегти чіткість та деталі, в той час як шум потрібно розмити.

Процес просторової фільтрації включає наступні кроки:

1) Встановлення центральної точки (x, y) маски.

2) Реалізація операції, що використовує тільки значення пікселів у заданому околі навколо центральної точки.

3) Встановлення результату цієї операції як “реакції” на процес, що відбувається в центральній точці маски.

4) Повторення цього процесу для кожної точки на зображенні.

Маска переміщується по зображенню, і значення кожного пікселя у результуючому зображенні обчислюються як сума добутків значень пікселів оригінального зображення та коефіцієнтів маски, що відповідають цим точкам. Процес фільтрації маскою зображено на рисунку 1.5.

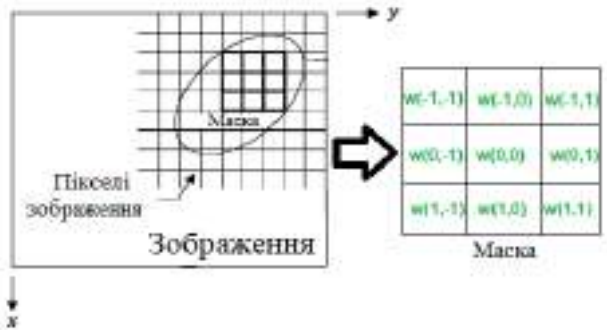


Рисунок 1.5 — Принцип застосування маски до зображення

Цей підхід може бути особливо ефективним для видалення імпульсного шуму, але через відсутність використання міжкадрової інформації, він не може ефективно протидіяти гауссовому або дробовому шуму. Серед основних методів просторової фільтрації можна визначити наступні:

***Фільтр Гауса,*** [6] є двовимірним оператором згладжування, найбільш використовуваним фільтром для розмиття зображень, видалення деталей та шуму. Основна ідея фільтра Гауса полягає в тому, що він модифікує вхідний сигнал шляхом згортки з гауссовою функцією. За принципом роботи схожий на фільтр усереднення, що замінює значення цільового пікселя на середнє, але на відміну від нього, усереднює значення пікселів за нормальним розподілом та має ядро, що описується формулою:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \quad (1.9)$$

де, $\sigma$ — стандартне відхилення нормального розподілу, $x$ — зсув від центру ядра згортки.

Застосування даного фільтру для фільтрації відеопотоку в режимі реального часу має свої недоліки, включаючи великі обчислювальні витрати, можливу втрату деталей, необхідність правильного підбору параметрів ядра, обмежену ефективність для інших видів шуму і вимогу до оптимізованої реалізації для забезпечення високої швидкості обробки. При використанні фільтра Гауса для обробки відеопотоку ураженого імпульсним шумом білі пікселі можуть залишитися, але їх яскравість може змінитися на сіру. Для ефективного видалення цього типу шуму частіше використовують медіанний фільтр, який дозволяє зберегти яскравість білих пікселів і відновити їх оригінальний вигляд.

***Медіанний фільтр,*** [6,7] представляє собою нелінійний метод цифрової фільтрації, на відміну від фільтру усереднення значення цільового пікселя замінюється медіанним. Для обчислення медіани всі значення входів в межах вікна спочатку сортуються за числовим порядком, а потім вхід, який розглядається, замінюється медіанним значенням з відсортованого списку вхідних значень. Цей процес повторюється для кожного входу на всьому сигналі, що призводить до створення сигналу з видаленим шумом, при цьому важливі особливості та деталі залишаються незмінними.

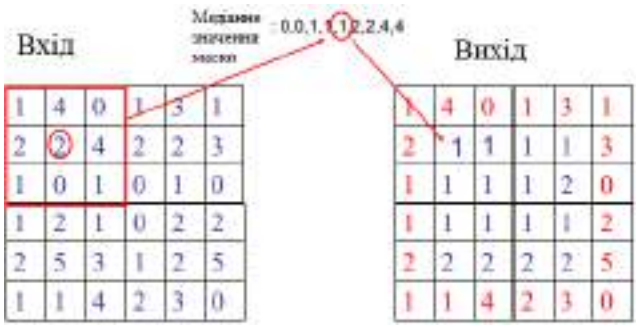


Рисунок 1.6 — Принцип роботи медіанного фільтра

Недоліком цього фільтра, особливо в контексті реально часової обробки відеосигналу є високі витрати обчислювальних ресурсів. Значних ресурсів при використанні медіанного фільтра витрачається на визначення медіани для кожної маски.

***Білатеральний фільтр*** [2] використовує два види подібності: інтенсивність пікселів та геометричну схожість пікселів, щоб покращити обробку зображень, зменшити шум та зберегти чіткість контурів об'єктів на зображенні. Він опирається на локальний аналіз зображення, що робить його відмінним і придатним для різних типів зображень та різних рівнів шуму.

Основна ідея білатерального фільтру полягає в тому, що він аналізує оточення кожного пікселя та враховує як його інтенсивність, так і геометричну схожість з іншими пікселями. Інтенсивність пікселів використовується для визначення вагового коефіцієнта, що впливає на результат фільтрації. Геометрична схожість визначає радіус околу кожного пікселя, який враховується під час обчислення вагових коефіцієнтів.

Використання білатерального фільтру призводить до зменшення шуму та збереження чіткості контурів об'єктів на зображенні. Математично він описується двома гаусовими функціями: просторовою (для врахування просторового розташування пікселів) та інтенсивність (для врахування різниці в інтенсивності пікселів). Цей підхід дозволяє збалансувати вплив обох видів подібності для досягнення оптимальних результатів під час фільтрації зображення.

Просторова гаусівська функція визначається наступним чином:

$$G_s(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1.10}$$

де, $G_s(x, y)$ — просторова гаусівська функція, $x, y$ — координати пікселя

Доменна гаусівська функція визначається наступним чином:

$$G_d(I_1, I_2) = e^{-(I_1-I_2)^2/(2\sigma^2)} \tag{1.11}$$

де, $G_d(I_1, I_2)$ — доменна гаусівська функція, $I_1, I_2$ — інтенсивності сусідніх пікселів.

Білатеральний фільтр можна описати за допомогою наступної математичної формули:

$$\mathrm{B}F(I)(x,y) = \frac{1}{W}\sum_{x',y'} I(x',y') * G_s(x-x',y-y')G_d(I(x,y)I(x',y') \quad (1.12)$$

де, $\mathrm{B}F(I)(x,y)$ —результат білатерального фільтра для пікселя з координатами (x,y), W — нормалізуючий коефіцієнт.

Білатеральний фільтр єефективним для покращення якості зображень та зменшення шуму, але має обмеження при обробці відеопотоку в режимі реального часу. Він вимагає значних обчислювальних ресурсів для ітераційного обчислення вагових коефіцієнтів для кожного пікселя, що може викликати затримки при великій роздільній здатності відео. Також він не враховує темпоральну стійкість між кадрами, що може призводити до артефактів при рухомих об'єктах або змінах в середовищі. Параметри, такі як стандартні відхилення, потрібно підібрати вручну для кожного відеопотоку. Таким чином, в застосуванні білатерального фільтра до відеопотоку в режимі реального часу потрібно розглядати компроміси між якістю обробки та продуктивністю, а також враховувати особливості даних в кожному конкретному випадку для досягнення оптимальних результатів.

### 1.4. Методи фільтрації за часом. Їх комбінування з просторовими методами.

У відеопотоці кадри зазвичай корелюють у часі. Основна суть часової (темпоральної) фільтрації полягає у використання інформації, отриманої з кореляції між послідовними кадрами відео. Замість обробки кожного кадру окремо, темпоральні фільтри використовують дані з попередніх та наступних кадрів для видалення шуму та покращення деталей на поточному кадрі.

Цей процес базується на припущенні, що багато елементів зображення залишаються стабільними або мало змінюються з кадру на кадр, оскільки багато об’єктів залишаються нерухомими або рухаються плавно. Таким чином, інформація з сусідніх кадрів може бути використана для виявлення та видалення шуму.

Цей процес може включати ряд методик, таких як усереднення значень пікселей протягом декількох кадрів або використання статистичних методик для ідентифікації та видалення шуму. Такий підхід допомагає зменшити шум, особливо у ділянках, де шум змінюється з кадру на кадр, та зберігає стабільність деталей та об’єктів на зображенні.

Найпростіший темпоральний фільтр - це Темпоральний Фільтр Арифметичного Середнього (Temporal Arithmetic Mean Filter, TAMF).

***TAMF*** [8,9] відфільтровує шуми шляхом заміни кожного значення пікселя середнім значенням з попередніх кадрів у визначеному вікні або інтервалі. Метод ефективно протидіє випадковому шуму або шуму, що змінюється швидко.

Недоліком цього методу є виникнення паразитного розмиття в областях відеопотоку де спостерігається рух об'єктів, тому цей спосіб зазвичай застосовують для статичних сцен. Тому важливим при застосування цього методу є правильний вибір розміру вікна. Іншим шляхом подолання цієї проблеми комбінування цього фільтру з просторовими фільтрами.


Рисунок 1.7 — Застосування до відеопотоку ураженого Гаусівським шумом TAMF фільтра з вікном довжиною n=9

Просторово-часове фільтрування використовує інформацію з сусідніх пікселів та міжкадрової кореляції для покращення якості зображення. Основні переваги просторово-часового фільтрування полягають у його здатності використовувати багато джерел інформації для обробки кожного пікселя, що дозволяє досягти більш точного і стабільного результату. В загальному випадку цей метод не вимагає виявлення руху, що спрощує його застосування.

Однак, просторово-часове фільтрування має і свої недоліки. Великий обсяг вхідної інформації, необхідний для обробки одного пікселя, означає значну обчислювальну складність. Це робить цей метод одним з найбільш обчислювально вимогливих, і часто його використовують для офлайн обробки. Крім того, деякі методи просторово-часового фільтрування можуть призводити до утворення артефактів на межах рухомих об'єктів.


Рисунок 1.8 — Результат фільтрації відеопотоку тривимірним векторним медіанним фільтром

### 1.5. Методи частотної фільтрації

Одним з основних методів обробки зображень є частотна фільтрація, яка полягає в переході від просторової області, де зображення представлене як матриця пікселів, до частотної області, де зображення представлене як сукупність гармонічних коливань різних частот. Для цього використовується перетворення Фур’є, для двовимірного зображення f(x, y), розміром M x N, визначається наступним чином:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi(\frac{ux}{M}-\frac{vy}{N})} \quad (1.13)$$

де F(u, v) — комплексна функція, яка називається спектром Фур'є зображення. Значення F(0, 0) відповідає середньому рівню яскравості всього зображення і називається постійною складовою спектра. Спектр Фур'є має симетричну структуру.

Спектр Фур'є складається з пікселів, які характеризуються амплітудою та фазою. Амплітуда показує силу частотної складової, а фаза – її зсув в просторовій області. Амплітуди пікселів спектра Фур'є можуть мати дуже великий динамічний діапазон, тому для їх візуалізації застосовується логарифмування. Частотна фільтрація зображення полягає в тому, що до спектра Фур'є застосовується певна функція, яка модифікує амплітуди та/або фази певних частотних складових. Ця функція називається частотною характеристикою або перехідною функцією фільтра. Потім до модифікованого спектра застосовується обернене перетворення Фур'є, яке повертає зображення в просторову область.

Частотна фільтрація відеопотоку від шумів передбачає наступні кроки:

1) Розбити вихідний відеопотік на окремі кадри. Для того, щоб спектр Фур'є зображення був симетрично розташований відносно центру, вхідне зображення піддається масштабуванню за допомогою виразу $(-1)^{x+y}$, який змінює знак кожного другого пікселя по горизонталі та вертикалі.

2) Для кожного фрейму обчислити пряме перетворення Фур'є

3) Помножити спектр Фур'є кожного кадру на функцію обраного частотного фільтру.

4) Перевести кожен фрейм назад з частотної області в просторову області за допомогою оберненого перетворення Фур'є.

5) Об'єднати всі кадри у вихідний відеопотік.

Частотні фільтри поділяються на:

***Низькочастотні фільтри*** [10] (Фільтри згладжування) – це вид частотної фільтрації зображень, який дозволяє пропустити низькі частоти та

послабити високі. Низькі частоти відповідають за яскравість на гладких ділянках зображення, а високі частоти - за контури та шуми. Цей процес можна представити як згортку зображення з функцією фільтра.

Ідеальний фільтр низьких частот описується функцією:

$$F(u,v) = \begin{cases} 1 \text{ якщо } D(u,v) \le D_0 \\ 0 \text{ якщо } D(u,v) > D_0 \end{cases}$$

$$D(u,v) = \left[(u - {}^{M}/_{2})^2 + (v - {}^{N}/_{2})^2\right]^{\frac{1}{2}} \quad (1.14)$$

де, $D_0$ — частота зрізу, додатня величина при якій H(u, v) перетворюється у 0 D(u, v) = відстань між точкою (u, v) у частотній області та центром частотного прямокутника.

У *ідеальному фільтрі* (ILPF) [11] пропускаються без затухання всі частоти, які знаходяться на або всередині кола радіусом Do, але всі частоти, які знаходяться за межами цього кола, повністю фільтруються. Фільтр має радіальну симетрію відносно початку координат, тому його можна описати за допомогою радіального перерізу. Ідеальний фільтр не може бути реалізований у практиці, тому що його імпульсна характеристика є нескінченною за тривалостю та некаузальною, тобто залежить від майбутніх значень сигналу . ILPF має різку грань між пропусканням та приглушенням частот, що призводить до появи коливань на зображенні після оберненого перетворення Фур'є. Це явище називається ефектом Гіббса .

*Фільтр Баттерворта* – вид фільтра, який проектується так, щоб мати максимально плоску амплітудно-частотну характеристику в області пропускання. Частотна характеристика фільтра Баттерворта плавно зменшується від одиниці в області пропускання до нуля в області затримки. Швидкість зменшення залежить від порядку фільтра. Функція передавання фільтра Баттерворта n-го порядка може бути задана наступним чином:

$$H(u,v) = \frac{1}{1 + ({}^{D(u,v)}/_{D_0})^{2n}} \quad (1.15)$$

Фільтр Баттерворта не має різкої межі між пропусканням та затримкою частот, що усуває ефект Гіббса на зображенні після оберненого перетворення Фур’є. Фільтр Баттерворта першого порядку не має коливань у просторовій області, які можуть бути непомітними для фільтрів другого порядку, але можуть стати значними для фільтрів вищих порядків. Фільтр Баттерворта двадцятого порядка має подібну характеристику до ідеального фільтра. Фільтр Баттерворта другого порядка є хорошим компромісом між ефективним низькочастотним фільтруванням та прийнятним коливанням

Згладжування низькочастотним фільтром допомагає підвищити видимість градієнтів і плавних змін в яскравості, особливо коли зображення містить шум. Також результатом фільтрації зображення стає менш деталізованим і знижує різкість змін у яскравості між сусідніми пікселями. Це може призвести до розмиття зображення, але такий ефект може бути корисним для приглушення високочастотного шуму та виділення більш плавних деталей на зображенні.

***Високочастотні фільтри*** [10] (Фільтри збільшення різкості) – це вид частотної фільтрації зображень, який дозволяє пропустити високі частоти та послабити низькі. Високі частоти відповідають за деталі, текстури і шуми на зображенні, а низькі частоти - за освітлення, контраст і кольори.

Підсилення високочастотним фільтром допомагає покращити різкість, виділити краї об’єктів. Також результатом фільтрації зображення стає більш деталізованим і підвищує контраст між сусідніми пікселями. Однак такі фільтри можуть підсилити високочастотний шум, зменшити різкість переходів між об'єктами, не ефективно враховувати великі масштабні зміни в яскравості.

### 1.6. Методи фільтрації на основі нейромереж.

У останні роки підходи, що базуються на нейронних мережах, показали високу ефективність у великій кількості завдань у сфері комп'ютерного зору, зокрема в знешумленні зображень та відеопотоків. Це стало можливим

завдяки постійному зростанню обчислювальної потужності пристроїв та впровадженню технологій, що дозволяють переносити стандартні обчислювальні операції з центрального процесора на графічний процесор. Такий підхід дозволяє будувати та тестувати великі моделі робочих мереж навіть на звичайних персональних комп'ютерах, а також передавати завдання обробки даних обчислювальним центрам через мережі.

***Згорткова нейронна мережа (CNN)*** [13] – це многошарова архітектура штучної нейронної мережі, що використовує процес згортки замість стандартного множення матриць. CNN розроблені для обробки даних з сітчастою структурою, наприклад, зображень або відео. Вони використовують фільтри або ядра для обробки вхідних даних і видобування відповідних характеристик. CNN є ідеальним вибором для обробки зображень, оскільки вони можуть автоматично вивчати важливі характеристики з вхідних даних, такі як краї, кути та текстури. Найбільш популярними архітектурами згорткових мереж є DnCNN та FFDNet.

DnCNN була розроблена [15] для боротьби з Гаусовим шумом при будь-якому розподілі та покращення різкості зображення. Її структуру наведено на рисунку1.9. Вона складається з трьох блоків:

- Початковий шар згортки має 64 фільтри розміром 3x3xC (де C відображає кількість кольорів у зображенні), а також використовує функцію активації ReLU.
- Послідовність згорткових шарів використовує фільтри розміром 3x3x64. Перед кожним сверточним шаром застосовується пакетна нормалізація, а також функція активації ReLU.
- Згортковий шар розміром 3x3x64, який відповідає за відновлення вихідних даних, знаходиться у кінці нейронної мережі.

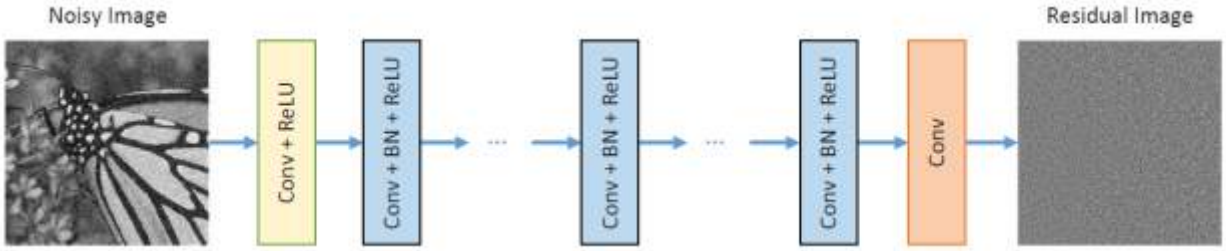


Рисунок 1.9 — Структура архітектури DnCNN

DnCNN погано протидіє просторово мінливому шуму, що робить цю модель погано застосованою на практиці. Враховуючи це було розроблено архітектуру FFDNet [16, 17], що базується на DnCNN. Структуру архітектури наведено на рисунку 1.10. Вхідний блок розподіляє вхідне зображення відповідно до його роздільної здатності. Це дозволяє зменшити складність алгоритму. Вихідний шар відновлює нелінійну вихідну інформацію до роздільної здатності зображення на момент входу. Це поліпшення призначене для пошуку оптимального розв'язку з точки зору складності архітектури та якості вихідних даних.

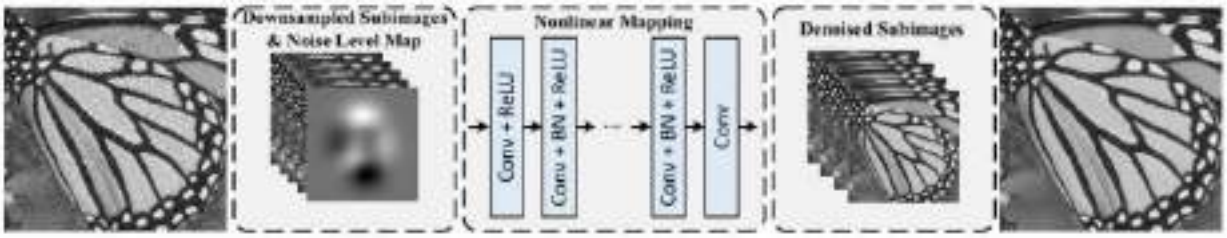


Рисунок 1.10 — Структура архітектури FFDNet

До недоліків сучасних систем фільтрації розроблених на архітектурі CNN [18-24] є необхідність створення датасетів з чистих та зашумлених відеопотоків для навчання нейромережі. Проте в деяких областях, таких як мікроскопія, часто неможливо отримати безшумні відео. Вирішення цієї проблеми було запропоновано в [25] архітектурі Unsupervised Deep Video Denoising (UDVD). UDVD пропонує архітектуру "сліпої плями", при якому нейромережа навчається оцінювати кожен шумний піксель з оточуючого просторового сусідства без включення самого пікселя. Це дозволяє UDVD ефективно обробляти послідовні дані, не маючи доступу до чистих вихідних відео під час тренування.


Рисунок 1.11 — Приклад знешумлення гауссового шума зі стандартним відхиленням 30 відносно інтенсивності архітектурою UDVD

***Рекурентні нейронні мережі (RNN)*** [13] представляють собою тип нейронних мереж, які використовують часові зв'язки для виявлення залежностей між послідовними проходами та з'єднаннями. Вони не ізольовані, але використовують інформацію з попереднього проходу для передачі даних не тільки від попереднього рівня, але й від самого нейрона. Це не дуже актуально при роботі з просторовими даними, такі як фото, але є корисним при роботі з часовими рядами зокрема послідовності кадрів відеопотоку. Експеримент [26] демонструє, що рекурентна структура, яка працює через часову область, здатна витягувати інформацію про рух, що сприяє видаленню шуму з відео. Тим часом, глибока структура має достатньо велику потужність для представлення відношення між пошкодженими відео як вхідними даними та чистими відео як результатом. Недоліки такої мережі схожі з класичними методами фільтрації за часом.

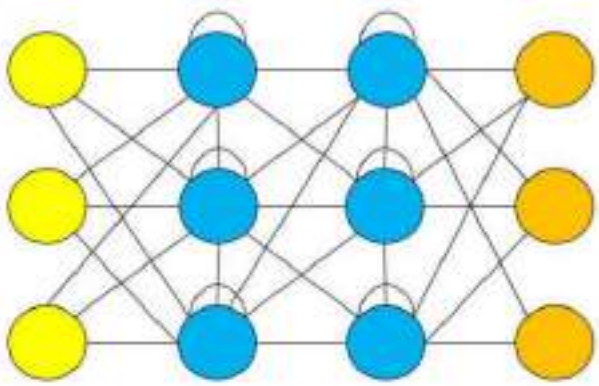

Рисунок 1.12 — Структура DRNN нейромережі

## 1.7. Висновки до розділу 1

Було проаналізовано основні види шумів, що діють на відеопотік та методи оцінки та боротьби з ними.

Було встановлено, що різні види шумів мають різний вплив на відеопотік, що необхідно враховувати при розробці методів боротьби з ними.

Також було виявлено основні проблеми, що виникають при розробці методів протидії шумам в відеопотоці в режимі реального часу:

1) Цифрова обробка відеопотоку в режимі реального часу вимагає значних обчислювальних потужностей. При цьому важливе значення в реально часовій обробці відеопотоку відіграють швидкість обробки та затримки при високому рівні пропускної здатності.

2) Обмеженість класичних методів фільтрації лише окремими видами шумами та їх повна непридатність для деяких видів шумів.

3) Виникнення артефактів при обробці відеопотоку. Особливо актуально це для динамічних сцен де присутній швидкий рух об’єктів.

Вищезазначені проблеми особливо актуальні для методів фільтрації [14]. В свою чергу основним недоліком нейромережевого методу є необхідність створення датасету з «чистими» та зашумленими відео для навчання нейромережі, що може стати проблемою для описаної задачі. Враховуючи вищезазначене для реалізації поставленої задачі обрано нейромережевий як такий, що має перспективи для розвитку. На основі проведеного аналізу для вирішення поставленої задачі планується розробка згорткової нейронної мережі, що буде використовувати підхід навчання без вчителя.

# Розділ 2. АНАЛІЗ ОСНОВНИХ НЕЙРОМЕРЕЖЕВИХ АЛГОРИТМІВ ДЛЯ ПРОТИДІЇ ШУМАМ

За останнє десятиріччя було проведено багато досліджень в сфері обробки зображень від шумів. Але як і класичні методи фільтрації більшість з цих алгоритмів базується на припущені, що шум наявний на зображенні є гауссовим чи імпульсним. Однак це припущення не завжди відповідає реальності, особливо коли мова йде про шум на фотографіях. Реальний шум або сліпий шум значно складніший і різноманітніший, через що більшість традиційних методів знешумлення не змогли ефективно впоратися з його видаленням з зображень. Ця проблема через методику, що використовується при навчанні таких мереж. Найбільш поширеним підходом навчання для алгоритмів знешумлення є навчання з вчителем. Зазвичай для навчання використовуються пари, що складаються з “зашумленого” та чистого зображень. Це значно спрощує процес навчання, але ускладнює збір навчальної вибірки для алгоритму, оскільки не для всіх об’єктів можна зробити знімки на великій витримці та з мінімальною кількістю шумів. Виходом з цієї ситуації для швидкої генерації датасету використовують накладення синтетичних шумів на відеопотік, але це призводить до вищезазначеною проблеми коли в результаті нейромережа може протидіяти лише цим синтетичним шумам. Виходом з цього може стати використання підходу навчання без вчителя та методів сліпої обробки, які не потребують попередньої інформації про джерело сигналу. У контексті навчання це означає, що для навчання мережі не потрібно мати пар «чистих» та «зашумленних» зображень.

Іншим основним недоліком таких мереж є складність архітектур, що містять багато згорткових шарів та потребують значної кількості обчислювальних ресурсів. Для спрощення розрахунків зображення, що потрапляє на вхід нейромережі стискають, використовують паралельну обробку та конвеєри даних. Але навіть цього буває недостатньо та багато з

існуючих алгоримів не можуть обробити тридцять кадрів на секунду, що робить їх використання для знешумлення відео в режимі реального часу неможливим.

В цьому розділі буде проведено порівняльний аналіз основних існуючих алгоритмів, що розроблені на основі методики навчання з вчителем та алгоритми, що використовують навчання без вчителя. Виявлено основні переваги та недоліки кожного з підходів.

## 2.1. Використання навчання з вчителем для створення алгоритмів знешумлення зображень

Штучна нейронна мережа – це структура, що складається з штучних нейронів, які з'єднані між собою та зовнішнім середовищем за допомогою зв'язків, кожен з яких має певний коефіцієнт, що називається ваговими коефіцієнтами.

Процес навчання нейронної мережі з вчителем полягає або ж керованого навчання полягає в тому, що мережа отримує набір навчальних даних, які складаються з вхідних і вихідних векторів. Вхідні вектори $x_i$ подаються на входи мережі, вони містять зашумленні зображення, а вихідні вектори $d_i$ відображають бажані реакції мережі на ці входи, тобто чисті зображення до яких повинна прагнути мережа. Кожна пара $x_i$ , $d_i$ називається навчальним прикладом, а їх сукупність - навчальною вибіркою. Метою навчання є знаходження таких вагових коефіцієнтів мережі, при яких вихідний сигнал мережі y буде якомога ближчим до цільового вектора d для всіх навчальних прикладів. Для цього використовуються різні алгоритми корекції ваг, які залежать від помилки δ, яка визначається як різниця між дійсним і бажаним вихідним сигналом мережі. Помилка вимірюється за допомогою певної функції, наприклад, квадратичної. Алгоритми корекції ваг змінюють ваги таким чином, щоб мінімізувати загальну помилку по всій навчальній вибірці. Навчання триває до тих пір, поки помилка не досягне заданого порогу або ж помилка на валідаційних даних не перестане зменшуватися. На рисунку 2.1

показано схематичне зображення процесу навчання нейронної мережі з вчителем.


Рисунок 2.1 — Процес навчання нейронної мережі

### 2.1.1. VNLB Non-local Bayesian Video Denoising

Алгоритм VNLB (Non-local Bayesian Video Denoising)[27] — це метод обробки відео, який використовує байєсівський підхід для зменшення шуму на відеозображеннях. Його сутність полягає у використанні байєсівської інференції для визначення ймовірностей різних станів пікселів відеоряду, а потім використанні цих ймовірностей для відновлення оригінального зображення з урахуванням шуму.

Суть методу зображена на рисунку 2.2. Метод передбачає наявність K зашумленних зображень, кожне з яких отримано незалежно додаванням білого гауссівського шуму до оригінального зображення. Далі для знешумлення зображення використовується метод емпіричної Байєсової оцінки. На першому етапі розраховується оцінка параметрів шуму моделі. Такий підхід передбачає, що амплітуди шуму розподілені за гауссовим законом та визначає шум за формулою

$$\theta = argmaxp(Y|X,\theta) \quad (2.1)$$

де Y — спостережувані дані (зашумлене зображення), X — невідоме чисте зображення, θ — параметри моделі шуму, і p(Y|X,θ) — ймовірність спостережуваних даних за умови невідомого чистого зображення та параметрів моделі шуму.

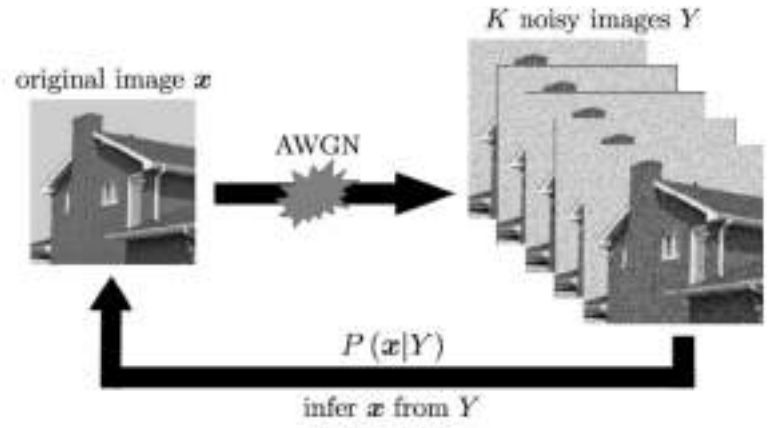


Рисунок 2.2 — Суть методу VNLB

Після чого відбувається оцінка значення пікселя за формулою:

$$\mathrm{X} = argmaxp(X|Y,\theta) \tag{2.2}$$

де p(X|Y,θ) — апостеріорна ймовірність невідомого чистого зображення за умови спостережуваних даних та оцінених параметрів моделі шуму.

Для алгоритму VNLB, який застосовується до відеозображень, цей процес виконується для кожного кадру відео, а результати можуть бути взяті до уваги в часовому контексті.

Роблячи це ітеративно та враховуючи не-локальний підхід, алгоритм VNLB намагається максимізувати ймовірність отримати оригінальне зображення, враховуючи як апріорне знання, так і актуальні спостереження.

Рисунок 2.3 — Результат Зліва направо: зашумлене зображення Y для σ = 20; зображення x після першого кроку знешумлення (29.57 PSNR), знешумлене зображення x (29.95 PSNR) та оригінальне зображення.

Недоліком цього методу є неможливість роботи в режимі реального часу. Обробка кольорового зображення розширенням 960 на 540 пікселів на процесорі займає близько 3 хвилин.

Іншим недоліком VNLB є припущення, що шум у відео є адитивним білим гауссовим шумом (AWGN). Якщо реальний шум відрізняється від цієї моделі, результати знешумлення можуть бути недостатньо ефективними.

Окрім цього хоча й не-локальний підхід може покращити якість знешумлення, він також може призвести до небажаних артефактів, якщо структура зображення значно відрізняється в різних областях.

### 2.1.2. V-BM4D Video denoising using separable 4-D nonlocal spatiotemporal transforms

Алгоритм V-BM4D (Video Block-Matching and 4D filtering) [28] представляє сучасний метод знешумлення відео, який демонструє значний прогрес порівняно з його попередником V-BM3D (Video Block-Matching 3D). V-BM3D вже використовував просторову та часову надлишковість відеосигналу, створюючи 3D просторово-часові об’єми, прослідковуючи блоки вздовж траєкторій, визначених векторами руху.

Алгоритм V-BM4D є еволюцією V-BM3D, розширюючи його можливості та досягаючи великої ефективності у видаленні шуму. В основі V-BM4D лежить використання просторово-часових перетворень, що дозволяють ефективніше аналізувати та враховувати як просторові, так і часові аспекти відеосигналу. Основна відмінність від V-BM3D полягає в тому, що V-BM4D не обмежується використанням лише одного кадру відео. Це дозволяє алгоритму ефективно використовувати просторову та часову інформацію, яка присутня в усій відеопослідовності.

Алгоритм V-BM4D (Video Block-Matching and 4D filtering) включає в себе три основні кроки: групування, колаборативне фільтрування та

агрегацію. Ці кроки виконуються для кожного просторово-часового об’єму відео.

Групування. На першому етапі визначається схожість просторово-часових блоків відео. Це досягається шляхом нелокального пошуку, що визначає інші блоки, схожі на обраний блок, і формує групу. Група представляє собою тривимірний об'єм, який враховує часові та просторові залежності.

Колаборативне фільтрування. Включає в себе застосування сепарабельного 4D просторово-часового перетворення до кожної сформованої групи. Це враховує локальні та нелокальні кореляції в часі та просторі, сприяючи зменшенню шуму. Методи порогового фільтрування використовуються для оптимізації обробки та видалення шумових складових.

Агрегація. Фінальний етап включає обернене просторово-часове перетворення оброблених груп, яке веде до отримання зменшених блоків. Потім відновлені блоки повертаються на відповідні місця у відеопослідовності та агрегуються, формуючи остаточну відновлену версію відео.

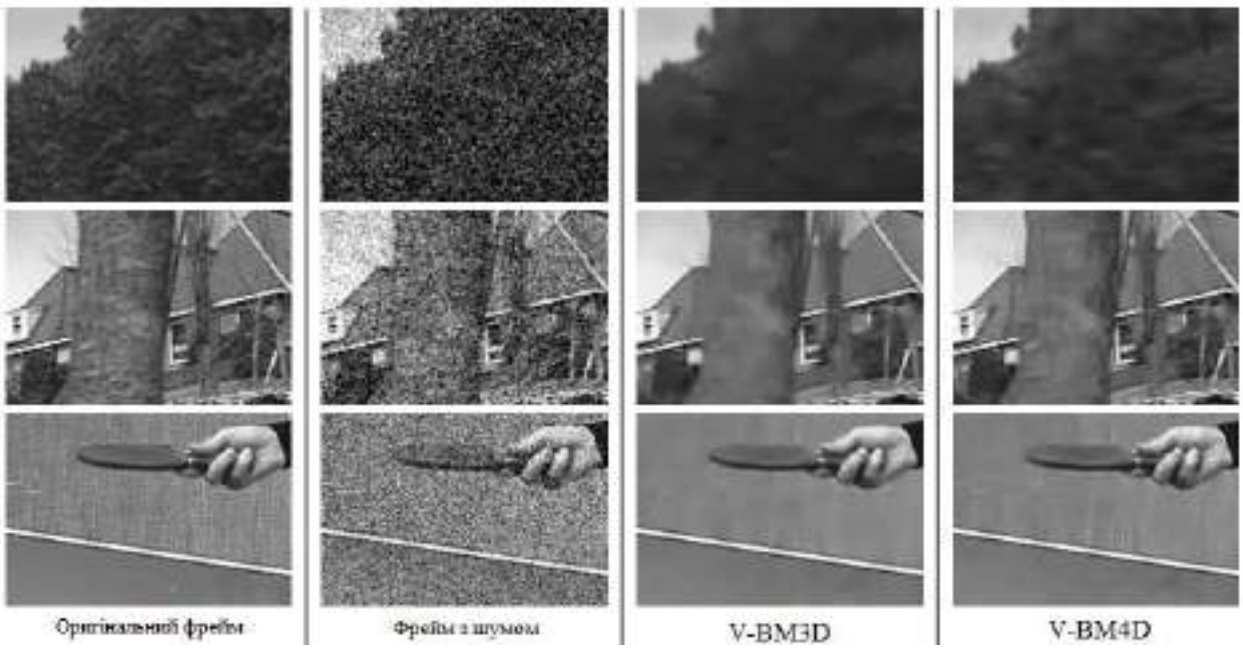


Рисунок 2.4 — Результати обробки відеофрейму ураженого білим гаусовим шумом зі стандартним відхиленням σ = 40.

Як видно з рисунку 2.4 обробка відео цим методом призводить до втрат дрібних детелей. Окрім цього алгоритму властиві недоліки аналогічні до VNLB. Зокрема обробка RGB зображення розширенням 960 на 540 пікселів на процесорі займає близько 1.5 хвилин. Також V-BM4D має проблеми з вирізненням між темпоральною та просторовою подібністю при групуванні блоків. Це може впливати на ефективність фільтрації, оскільки взаємна залежність цих аспектів може призводити до помилок та артефактів під час видалення шуму.

### 2.1.3. VNLnet Video denoising CNN with Non-locality information

VNLnet [29] — це алгоритм знешумлення відео, який використовує конволюційні нейронні мережі (CNN) для обробки відео та видалення шуму. Він використовує не-локальну інформацію, збираючи n найбільш схожих патчів для кожного патча вхідного зображення в прямокутному просторово-часовому вікні пошуку. Центральний піксель кожного схожого патча збирається в вектор ознак, який призначається кожному місцю зображення. Це призводить до зображення з n каналами, яке подається на вхід до CNN, навченої прогнозувати чисте зображення з цього високовимірного вектора. Мережа обробляє відео кадр за кадром, перед тим, як його подати на мережу, кожен кадр обробляється модулем пошуку не-локальних патчів, який обчислює не-локальний вектор ознак на кожній позиції зображення.

Архітектуру мережі наведено на рисунку 2.5. Вона складається з двох етапів: не-локального та локального. Не-локальний етап складається з чотирьох шарів згортки з розміром ядра 1 та 32 ознаками, які дозволяють мережі обчислювати особливості пікселів з сировинних не-локальних особливостей на вході. Другий етап отримує особливості, обчислені на першому етапі. Він складається з 15 шарів з 64 ознаками та розміром ядра 3x3, з нормалізацією та функцією активації Relu. Мережа виводить резидуальне

зображення, яке потрібно відняти від зашумленого зображення, щоб отримати знешумлене.

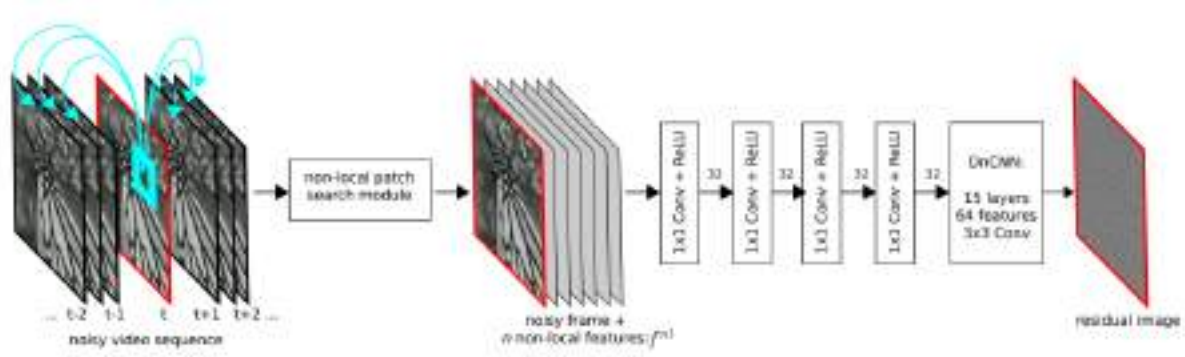


Рисунок 2.5 — Архітектура VNLnet

У порівняні з вищеописаними методами VNLnet для роботи використовує графічний процесор при цьому якщо порівнювати з вищеописаними алгоритмами обробка одного фрейму займає біля трьох секунд, що залишається все ще повільним для обробки в реальному часі.

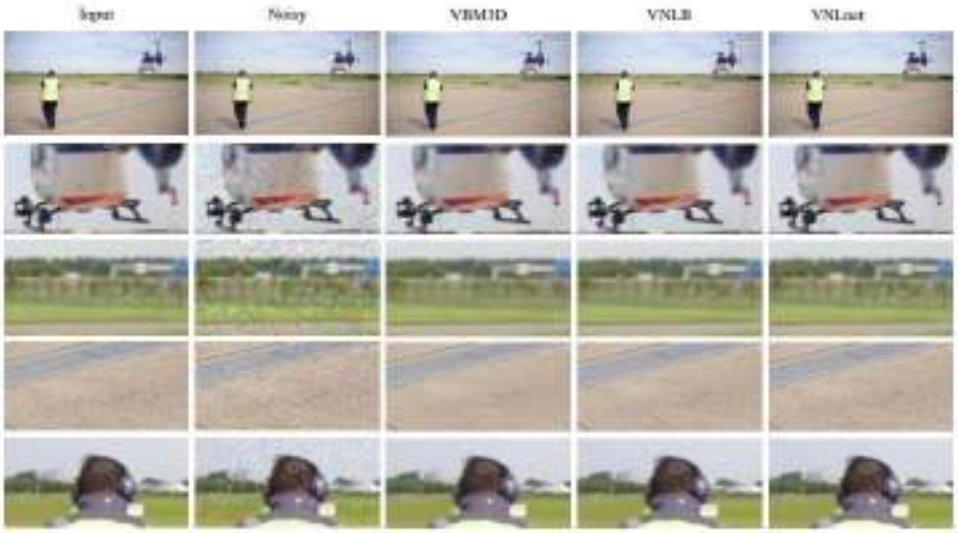


Рисунок 2.6 — Результати обробки відеофрейму ураженого білим гаусовим шумом зі стандартним відхиленням σ = 20.

### 2.1.4 ViDeNN Deep Blind Video Denoising

ViDeNN (Deep Blind Video Denoising) – це алгоритм знешумлення відео, який використовує конволюційні нейронні мережі (CNN) без попереднього знання про розподіл шуму.

Архітектуру метода зображено на рисунку 2.7. Вона також складається з просторової та темпоральної частин. На вхід просторового блоки мережа отримує цільовий кадр t, а також попередній t-1 та наступний до нього t+1. Аналогічно до VNLnet на виході просторового блоку присутній модуль залишкового шуму «noise residual». Вихід кожного блока віднімається від вхідного сигналу цього блока, що дозволяє отримати оцінку шуму12. Цей залишковий шум, або “noise residual”, представляє собою різницю між шумним вхідним сигналом та очищеним вихідним сигналом. Отримані просторові ознаки конкатенаціюються та потрапляють у блок темпорального знешумлення яким визначає часові ознаки у фреймах, на виході цього блоки знову присутній блок «noise residual» у результаті на виході нейромежі виходить знешумлений цільовий кадр t.

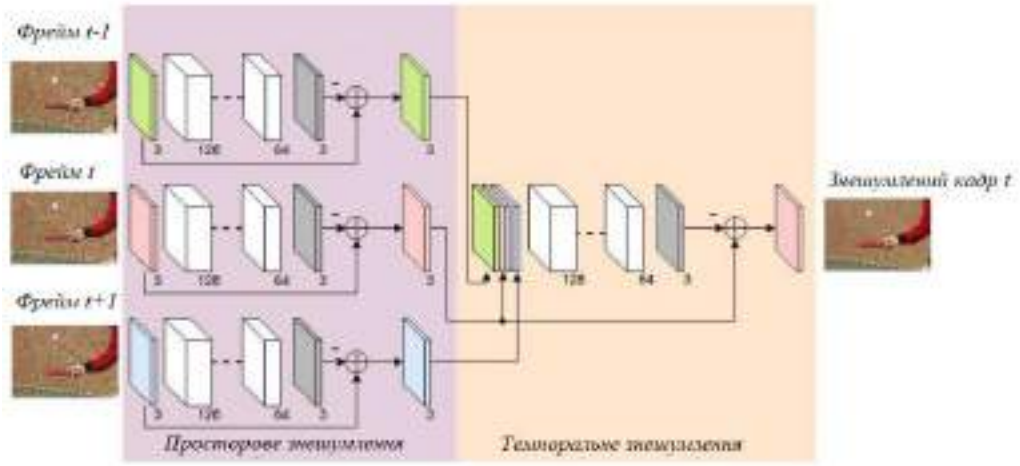


Рисунок 2.7 — Архітектура метода ViDeNN

Обробка RBG зображення на GPU розширенням 960 на 540 пікселів займає 1,2 секунди. Окрім цього для цієї архітектури також характерні типові недоліки методів знешумелння побудованих на методиці навчання з вчителем.

## 2.2. Використання навчання без вчителя для створення алгоритмів знешумлення зображень

Навчання без вчителя – метод машинного навчання, при якому система автономно здатна вирішувати завдання без заздалегідь визначених правильних

відповідей від експериментатора. Цей підхід ефективний у випадках, коли відомий опис множини об'єктів (навчальна вибірка), і головною метою є виявлення внутрішніх зв'язків, залежностей та закономірностей між об'єктами. Для боротьби з шумами на фото та відео, система повинна самостійно визначати структуру та закономірності, що характеризують шуми та корисну інформацію на цих зображеннях.

Відмінною рисою некерованого навчання є відсутність заданого для кожного об'єкта "правильного відповіді", як це характерно для керованого навчання. Замість цього, система взаємодіє з навчальними даними, коригуючи ваги мережі так, щоб отримані вихідні вектори були взаємно узгодженими. Цей процес визначає статистичні властивості обучального набору та групує схожі вектори в класи, що репрезентують внутрішні зв'язки в даних.

У випадку навчання без вчителя в контексті видалення шумів, навчальний набір складається лише з вхідних векторів, що містять зашумлені зображення одного і того самого об’єкту. Алгоритм навчання коригує ваги мережі з метою отримання узгоджених вихідних векторів, тобто таких, які відповідають схожим вхідним векторам. Це дозволяє системі визначати та видаляти шуми на зображеннях, розпізнавати патерни та встановлювати внутрішні зв'язки між пікселями чи кадрами відео. Це спрощує задачу створення датасету для таких мереж, що дозволяє навчати їх на реальних шумах.

### 2.2.1. Метод знешумлення без наявності чистих даних Noise2Noise

Метод Noise2Noise[30], розроблений компанією NVIDIA у 2018 році, представляє собою інноваційний підхід до вирішення проблеми шуму на зображеннях. На відміну від традиційних методів знешумлення, які використовують "чисті" зображення для порівняння із зшумленими, Noise2Noise навчали тільки на фотографіях, на яких вже був шум.

У процесі дослідження використовувалась архітектура згорткової нейромережі: U-Net. Датасет для навчання скалався з 50 тисяч зображень розміром 256 на 256 пікселів для тренування. Штучний шум був доданий до кожного зображення, і рівень шуму для кожної пари зображень був випадковим, що нейромережі мали враховувати при видаленні шуму. Додатково, алгоритми були навчені на рендерах приміщень, фотографіях з різнокольоровими написами та інших тренувальних об’єктах.

Метод Noise2Noise базується на припущенні, що хоча окремі кадри можуть бути зашумленими, сума зашумлених кадрів в цілому дає чисте зображення.

Крім застосування цієї технології в фотографії, наприклад, в астрофотографії або зйомці в темряві, розробники також розглядають можливість її використання для покращення якості знімків МРТ. Приклад використання алгоритму наведено на рис 2.8.

Рисунок 2.8 — Приклад використання алгоритму Noise2Noise Для гауссівського шуму з стандартним відхиленням (σ=25), Пуассона з середньою інтенсивністю(λ=30) та Бернуллі з дискретним розподілом ймовірностей (ρ=0.5)

Noise2Noise працює найкраще для шуму, що є незалежним від пікселів1. Якщо шум залежить від пікселів або має складну структуру, Noise2Noise може не забезпечити оптимальні результати. Noise2Noise краще працює з реальними шумами для яких неможливо створити датасет з пар чистих та зашумленних зображень ніж методи наведені в пункті 2.1. Але якщо вирішити проблему створення датасету то такий метод буде програвати у якості методам, що використовую методику навчання з вчителем.

Noise2Noise, як правило, використовується для обробки статичних зображень, а не відео в режимі реального часу. Теоретично, можливо застосувати Noise2Noise до послідовності кадрів відео, обробляючи кожен кадр окремо. Проте, з поточними обчислювальними ресурсами виконувати обробки в реальному часі цим алгоритмом неможливо.

### 2.2.2. Використання генеративно змагальних мереж для знешумлення зображень. Алгоритм SM-GAN

Генеративно-змагальні мережі (GANs) - це метод навчання без вчителя, що базується на використанні двох нейронних мереж, зазвичай згорткових. Прикладом такого алгоритму є SM-GAN[31] SMGAN, або Structurally-sensitive Multi-scale Deep Neural Network – це алгоритм, розроблений для зниження шуму на зображеннях, отриманих за допомогою комп’ютерної томографії (CT) з низькою дозою. Як і будь-яка GAN мережа вона скадається з двох підмереж. Перша мережа, це генератор, отримує на вхід зашумлене зображення та намагається видалити з цього шум. Друга мережа, дискримінатор, приймає результати, що генеруються першою мережею, і реальні дані, а потім намагається відрізнити реальні зразки від синтезованих.

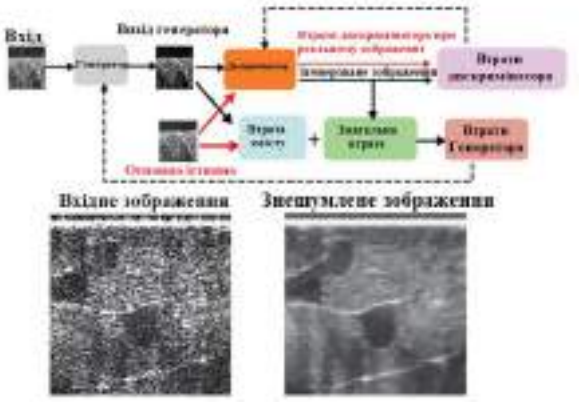


Рисунок 2.9 — Структура алгоритму SM-GAN

Основні труднощі при навчанні цієї моделі полягають в тому, що потрібно одночасно навчати дві нейронні мережі і правильно налаштувати баланс між ними. Генератор навчається створювати більш високоякісний вихід. У SM-GAN розрахунок втрат генератора складається з двох компонент: втрати змісту та чутливості, що передбачає розрахунок індексу подібності (SSIM) та середньої абсолютної помилки (MAE), між згенерованим та вхідним кадром та змагальними втратами дискримінатора- це середньоквадратична помилка (MSE) між вхідним та згенерованим зображенням, що показує наскільки добре дискрімінатор зміг виявити яке з зображень є згенерованим. Дискримінатор для навчання також використовує змагальні втрати. Проблемою GAN мереж є необхідність рівномірного тренування обох мереж. Якщо одна з мереж буде навчена краще, ніж інша, то це може призвести до того, що модель в цілому не буде працювати належним чином. Тому дискримінатор навчається у 5 разів частіше ніж генератор, що дозволяє надавати йому кращі градієнти для навчання генератора. Також він використовує техніку штраф за градієнт, або «gradient penalty» техніку, яка використовується в генеративно-змагальних мережах (GANs) для покращення стабільності навчання.

У контексті GANs, штраф за градієнт додається до функції втрат дискримінатора. Цей штраф обчислюється на основі градієнтів вихідних даних дискримінатора по відношенню до його вхідних даних12.

Мета штрафу за градієнт - забезпечити гладкість функції втрат, що допомагає уникнути проблеми з зниканням градієнтів, яка може виникнути, коли дискримінатор стає занадто сильним.

Недоліком цього алгоритму є його недоцільність його використання для обробки відео від шумів в режимі реального часу. Оскільки його основною задачею є знешумлення знімків комп’ютерної томографії.

### 2.3. Висновки до розділу 2

В розділі було виконано порівняльний аналіз актуальних методів, що використовуються для видалення шумів з фотознімків та відео. Основним недоліків усіх існуючих методів є висока потреба в обчислювальних ресурсах. Мінімумом продуктивності, якою повинна володіти система видалення шумів в режимі реального часу це 30 кадрів на секунду, що є недосяжним для більшості розглянутих алгоритмів. Виходом ситуації може стати стиснення відео, що потрапляє на вхід нейромережі, але це призводить до втрати малих деталей при обробці фреймів.

Методи навчання з вчителем при наявності датасету з пар «чистих» та «зашумлених» кадрів, що відповідають типу шумів яким повинна протидіяти нейромережа мають вищу продуктивність ніж методи некерованого навчання. При цьому у сферах де важко або неможливо отримати «чисті» кадри, наприклад, в астрофотографії  чи знімках МРТ краще себе показують методи навчання без вчителя.

# Розділ 3. РЕАЛІЗАЦІЯ АЛГОРИТМУ ДЛЯ ВИДАЛЕННЯ ШУМІВ В РЕАЛЬНОМУ ЧАСІ

## 3.1. Вибір засобів розробки програмного забезпечення

Протягом минулого десятиліття було створено велику кількість бібліотек, які включають в себе найпопулярніші алгоритми машинного навчання. Використання цих готових функцій, таких як метрики втрат та оптимізатори, спрощує процес розробки нейромережевих алгоритмів. Для цієї роботи було проведено аналіз різних мов програмування та відкритих розробок фреймворків та бібліотек, доступних для загального користування.

Для програмної реалізації моделі, було обрано мову програмування Python. Ця мова програмування відрізняється простотою і читабельністю, що знижує поріг входу у неї. ЇЇ було обрано для реалізації проекту через велику кількість доступних бібліотек, розроблених на її основі, які мають докладну документацію, дозволяють ефективно організовувати дані та використовувати алгоритми машинного навчання в зручному форматі. Згідно з статистикою Stack Overflow наведеною на Рисунку 3.1 червень 2017 року став першим , коли Python обійшов JavaScript і Java .

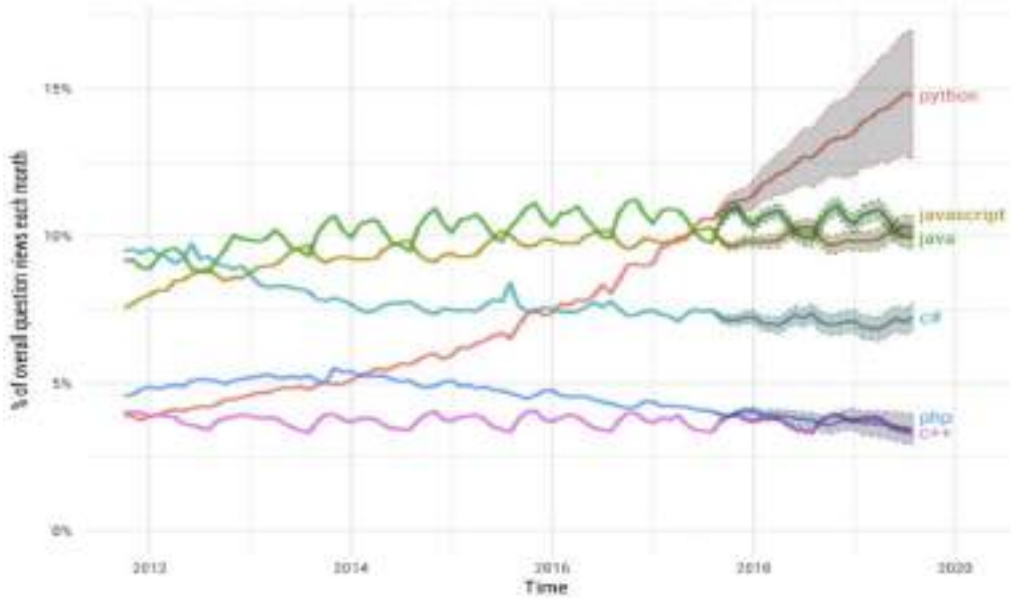


Рисунок 3.1 —Статистика використання мов програмування за даними stackoverflow

У процесі створення моделі нейронної мережі було вивчено кілька фреймворків, зокрема: mxNet, tensorflow, pytorch та keras. Ці фреймворки наразі є найбільш використовуваними для полегшення роботи з нейронними мережами, порівняння їх основних властивостей наведено в табл.3.1.

**Таблиця.3.1**. Порівняння фреймворків для розробки нейронних мереж

| | Переваги | Недоліки |
|---|---|---|
| Pytorch | -Дозволяє використовувати GPU для обчислень<br>-PyTorch є гнучкою бібліотекою для глибокого навчання, яка дозволяє створювати складні моделі з динамічною структурою<br>- Висока доля реалізованих проектів у сфері видалення шумів з фото та відео | - Може бути повільнишим ніж TensorFlow для деяких задач, особливо для великих моделей. |
| Keras | - Keras має простий, інтуїтивно зрозумілий API, що спрощує процес створення та налаштування моделей<br>- Keras має вбудовані функції для роботи з зображеннями та текстом | -Оскільки Keras є високорівневою бібліотекою, він може не мати такої гнучкості, як TensorFlow або PyTorch.<br>-Keras може бути повільнішим, ніж TensorFlow або PyTorch, особливо для більших, складніших моделей |
| Tensor Flow | - Дозволяє використовувати GPU для обчислень<br>-TensorFlow є потужною, гнучкою бібліотекою для глибокого навчання, яка дозволяє створювати складні моделі з багатьма шарами та компонентами | - Більш складний API<br>- TensorFlow може бути повільним для деяких задач, особливо для невеликих моделей або моделей, які вимагають багато ітерацій<br>- Невелика кількість реалізованих проектів у сфері видалення шумів з фото та відео |

У процесі порівняння було обрано Pythorch через можливість проводити розрахунки на GPU за допомогою бібліотеки CUDA від компанії NVIDIA. Це дозволило збільшити швидкість навчання та продуктивність на робочій станції, що використовувалась при розробці. Іншим важливим фактором стало те, що більшість проектів у цій сфері було реалізовано саме на Pythorch. PyTorch представляє собою відкритий інструментарій для машинного навчання, який активно застосовується в областях комп'ютерного зору та обробки природної мови. Первісно було розроблено командою Meta AI, але зараз він є частиною некомерційного консорціуму, що підтримує розвиток Linux [32].

Для попередньої обробки RGB зображень використовувалась бібліотека OpenCV. OpenCV представляє собою набір програмних функцій та алгоритмів, які в основному застосовуються для обробки зображень в режимі реального часу. Ця бібліотека є відкритим програмним забезпеченням, що доступне безкоштовно та може працювати на різних платформах. Крім того, якщо доступний графічний процесор, ця бібліотека дозволяє проводити обчислення на пристрої користувача. За допомогою OpenCV можна взаємодіяти з апаратним забезпеченням пристрою, наприклад, з камерою, і проводити первинне зчитування та обробку зображень.

Також використовувалась бібліотека NumPy, що є ключовим компонентом для більшості бібліотек, оскільки вона надає ефективні засоби для роботи з числовими даними, матрицями та таблицями, а також дозволяє легко виконувати багато типових операцій.

### 3.2. Підготовка даних для навчання нейромережі

У ході роботи проводилися, як дослідження на відео з FPV дронів, так і з синтетичними шумами. Оскільки під час дослідження не вдалося досягти задовільних результатів при роботі з реальними шумами остаточно формування датасету проводилося штучним зашумленням відеорядів з датасету DAVIS2017[34]. Цей датасет містить 87 відео розширенням 854x480 частотою кадрів 25кадрів/секунду. Зашумлення проводилось синтетичний AWGN шумом наближеним до реалістичного, що описується формулою

$$M = \sqrt{\frac{Ag*Dg}{N_{sat}*X} + Dg^2(\mathrm{Ag} * \mathrm{CT1} + \mathrm{CT2})^2} \quad (3.1)$$

де Ag — це підсилення сигналу датчика, Dg — цифрове підсилення, CT1 — константа моделі шуму 1.25e-4.та CT2 =1.11e-4, Nsat —параметр насичення шуму =7480, X — вхідне зображення.

Результуюче зашумлене зображення формується:

$$Y = \mathrm{X} + \mathrm{N} * \mathrm{M} \quad (3.2)$$

де Y — це зображення з шумом. X — це оригінальне зображення.
N — це шум, що генерується з нормального розподілу з середнім 0 та стандартним відхиленням 1, M - це стандартне відхилення шуму для кожного пікселя зображення, яке обчислюється за допомогою формули, наведеної вище.

Окрім цього також проводилось накладання гаусівського шуму. При цьому $\sigma$ для гаусівського шуму змінюється у межах 5-50, Ag та Dg у межах 0-0.8. Приклад зашумлення фрейму наведено на рисунку 3.2.


Рисунок 3.2 —.Приклад накладання штучного шуму на зображення

Для збільшення варіативності датасету та уникнення проблеми перенавчання було додатково використано аугментацію даних. Аугментація даних - це метод штучного збільшення обсягу набору даних за допомогою створення нових зображень з використанням різних перетворень. Зокрема було застосовано перевертання зображень на 180° та віддзеркалення зображення по вертикалі.

### 3.3. Архітектура розробленої нейронної мережі

Лістинг архітектури розробленої нейронної мережі наведено у додаткуА. Діаграму архітектури мережі наведено на рисунку 3.3. На вхід архітектури подаються цільовий кадр t, а також два попередніх t-1, t-2 та два наступних t+1,t+2. Для цих кадрів розраховується стандартне відхилення зображення за формулою

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - \mu)^2} \tag{3.3}$$

де N — загальна кількість пікселів в зображенні, $p_i$ — інтенсивність і-го пікселя, μ — середня інтенсивність пікселів в зображенні

На основі розрахованого стандартного відхилення будуються карти шуму, подібно до методу FFDNet [16,17], що також подаються на вхід нейромережі. Це дозволяє ефективно обробляти шум, що змінюється в

просторі. Карта шуму, яка подається на вхід, містить важливу інформацію про характеристики шуму на вході. Ця інформація представлена у вигляді очікуваного стандартного відхилення шуму для кожного пікселя. Наприклад, при обробці гауссового шуму, карта шуму буде мати постійне значення; у випадку шуму Пуассона, карта шуму буде залежати від інтенсивності зображення.

Модель включає в себе чотири блоки для обробки просторово-часових даних, які об’єднані в дворівневу структуру. Кожен з цих блоків представляє собою адаптовану версію моделі U-Net, яка обробляє три послідовних кадри. Всі чотири блоки використовують однакові ваги, що дозволяє зменшити вимоги до обсягу пам’яті і спрощує процес навчання мережі. При необхідності також можна навчати перший та другий блок окремо. Фрейми що отримуються на виході першого блоку об’єднуються з шумовими картами фреймів t-1 t t+1 відповідно та потрапляють у других блок знешумлення. У результаті на виході цього блоку виходить знешумленний фрейм t.

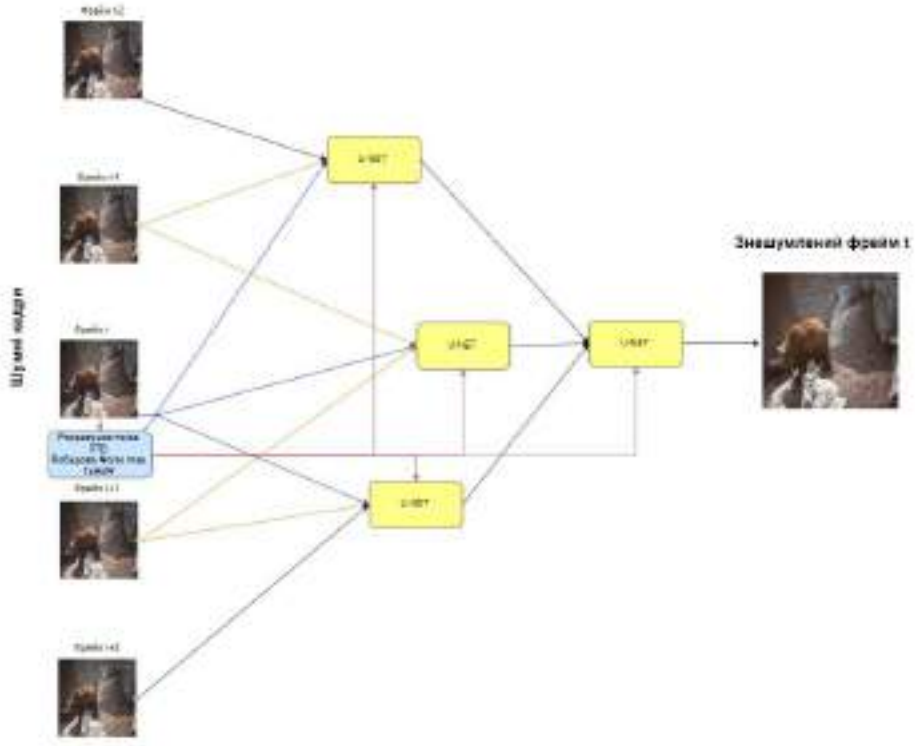


Рисунок 3.3 — Структура запропонованого алгоритму.

Розглянемо більш детально структуру блоків знешумлення U-Net. Структуру блоку наведено на рисунку 3.4. Цей блок є варіацією нейронної мережі U-Net - згорткової нейронної мережі, що була вперше введена у 2015 році командою з Фрайбурзького університету [35] з метою сегментації біомедичних зображень. Ця мережа є варіантом повнозв'язної згорткової мережі, але вона була адаптована для роботи з меншими наборами даних та для забезпечення більш точної сегментації.

На відміну від стандартного U-Net запропонований блок отримує 3 трьохканальні RGB зображення та 3 одноканальні карти шумів. Нам цими тензорами виконується операція конкатенації, яка об'єднує шість тензорів вздовж одного виміру. Це дозволяє врахувати не лише просторову темпоральну інформацію.

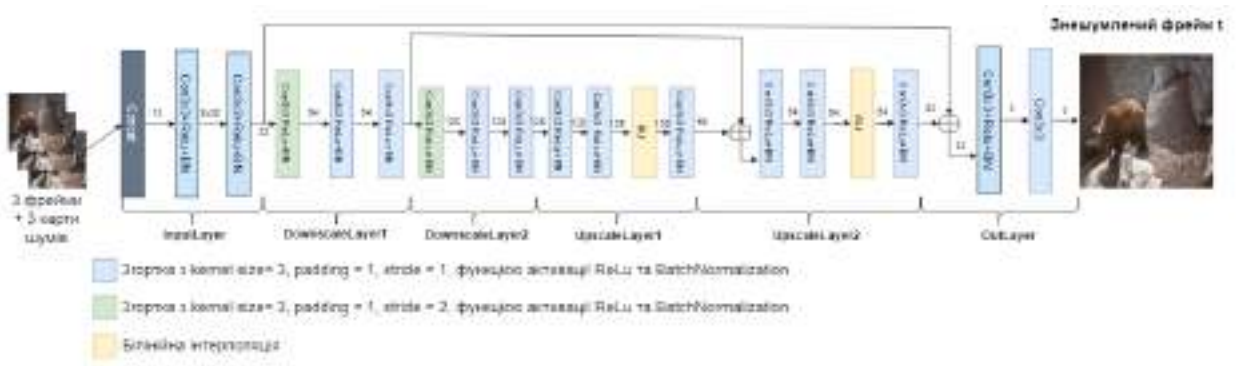


Рисунок 3.4 —. Структура блоку U-net

У якості функції активації використовувалась ReLU (Rectified Linear Unit). Це тип нелінійної функції активації, який широко застосовується в згорткових нейронних мережах та глибокому навчанні для всіх шарів, окрім останнього. Вона сприяє здатності моделі виявляти складні, не лінійні відносини в даних. ReLU функціонує шляхом заміни всіх негативних значень на нуль, що дозволяє моделі легше визначати, які нейрони мають бути активовані, а які - ні. Графік функції ReLu наведено на рисунку 3.4.

Рисунок 3.5 — Функція активації ReLu

Для оптимізації навчання Batch Normalization — це техніка, яка використовується для прискорення та стабілізації процесу навчання в штучних нейронних мережах. Вона досягає цього шляхом нормалізації вхідних даних в кожному шарі шляхом перецентрування та масштабування. Це поліпшує здатність моделі узагальнювати, прискорює процес навчання, дозволяє використовувати вищі швидкості навчання, зменшує вплив поганої ініціалізації та допомагає уникнути перенавчання.

Іншою особливістю U-Net, що дозволяє полегшити навчання та уникнути проблеми зниклого градієнта це skip connection. Skip connection (або залишкове з'єднання) - це компонент нейронних мереж, який дозволяє пропустити деякі шари нейронної мережі та подати вихід одного шару як вхід до наступних шарів

Ще однією відмінністю запропонованої мережі від класичної U-Net архітектури є використання замість операції деконволюції білінійної інтерполяції. Згідно [36], така заміна дозволяє уникнути так званих «шахових артефактів», що виникають через нерівномірне перекриття деконволюції.

### 3.4. Процес навчання запропонованої нейронної мережі.

Лістинг архітектури розробленої нейронної мережі наведено у додатку А. Алгоритм навчається за методикою навчання без вчителя. Для формування пар зображень використовується метод частково схожий на описаний в Noise2Noise [33]. У Noise2Noise на виході нейромережі використовується одне зображення двічі випадково зашумлене, або 2 зображення одного об'єкта у

разі якщо він нерухомий. У запропонованому методі для навчання достатньо мати лише одне зашумлене зображення.

На першому етапі відбувається попередня підготовка вхідних даних. Вхідні відео послідовності, що попередньо розбито на фрейми зчитуються за допомогою бібліотеки opencv. Коригується розмір вхідного фрейму після чого трьохканалальне зображення RGB зображення перетворюється у трьох канальний тезор, додатково значення пікселів у тензорі обмежуються у форматі 0 до 1 для уникнення артефактів.

На наступному етапі відбувається формування підвиборок для сліпого навчання шляхом розбиття зображення на фрагменти. Принцип формування навчальних підвиборок наведено на рисунку 3.6. Для роботи алгоритму зображення повинно бути однорідним. Спочатку формуються дві маски, для цього вхідне зображення розбивається на околи розміром 2 на 2 пікселі. З цього околу випадковим чином обираються 2 сусідні пікселі. Один з цих пікселів долучають до маски, що буде брати участь у формування фрагменту $g_1(y)$, а інший в маску для $g_2(y)$. З створених масок відповідно формуються суб зображення $g_1(y)$ $g_2(y)$ шляхом елементного множення зображення на маску. Результуючі суб зображення $g_1(y)$ $g_2(y)$ мають розміри вдвічі менший за зображення y. Отримані вибірки подаються на вхід нейромережі.

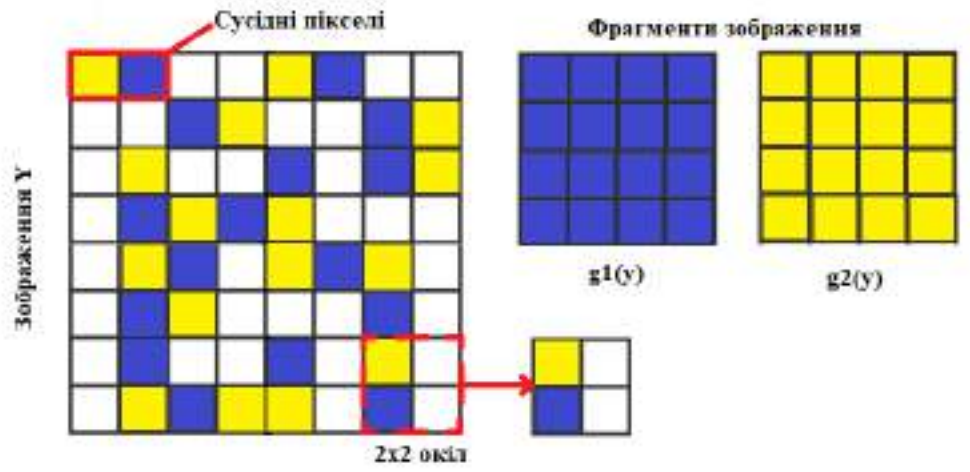


Рисунок 3.6 —.Створення навчальних підвиборок $g_1(y)$ та $g_2(y)$ з зображення Y. Пікселі, що використовуються для створення $g_1(y)$наведено синім, $g_2(y)$ жовтим

На третьому етапі відбувається розрахунок втрат. Загальні втрати нейронної мережі складаються з двох компонент. Перша це середньоквадратична помилка (Mean Squared Error, MSE) між вихідними даними моделі $g_1'(y)$та цільовим шумним зображенням $g_2(y)$. Вона обчислюється за формулою:

$$\text{loss1} = E_{x,y}[g1'(y) - g2(y)]^2 \quad (3.4)$$

де $g1'(y)$ — вихід моделі для підзображення $g_1(y)$, $g2(y)$ — цільове шумне підзображення

Другою компонентою втрат є середньоквадратична помилка, що обчислює різницю між вихідними даними моделі та цільовим шумним зображенням, а також очікувану різницю. Вона обчислюється за формулою:

$$\text{loss2} = E_{x,y}[(g1'(y) - g2(y)) - ((g1'(y) - g2'(y)))]^2 \quad (3.5)$$

де $g2'(y)$ —вихід моделі для цільового шумного зображення

Результуюча помилка знаходиться як сума компонент з врахуванням вагових коефіцієнтів:

$$\text{loss} = k * loss1 + j * loss2 \quad (3.6)$$

де k, j — вагові коефіцієнти складових втрат.

Після розрахунку втрат відбувається зворотне розповсюдження помилки та адаптивне підлаштування швидкості навчання за допомогою Оптимізатор Adam. Це алгоритм оптимізації, який використовується в глибокому навчанні. Він є розширенням стохастичного градієнтного спуску, який використовує відмінні швидкості навчання для різних параметрів. Adam оптимізатор дозволяє уникнути проблеми локальних мінімумів, коли алгоритм при навчанні запинається в мінімумі, який не є глобальним.

Важливим при перенавчанні є уникнення проблеми перенавчання, коли модель демонструє високу точність на тренувальному наборі даних, але погано справляється з новими, невідомими даними. Це відбувається, коли

модель “занадто добре” вивчає тренувальний набір даних до такої міри, що вона вивчає не тільки основні закономірності даних, але й шум, який присутній у тренувальному наборі даних. Для протидії цьому явищу в алгоритмі використовується метод “раннього зупинення” (Early Stopping). Він передбачає використання додаткового набору даних, відомого як валідаційний набір, для перевірки продуктивності моделі після кожної епохи навчання.

Якщо модель продемонструє покращення на валідаційному наборі, параметри моделі зберігаються. Якщо ж ні, то завантажуються ваги попередньо збереженої моделі.

### 3.5. Результати роботи алгоритму. Основні проблеми при розробці нейромережевих алгоритмів боротьби з шумами.

Для порівняння результатів роботи запропонованого алгоритму з іншими було використано тестову частину датасету DAVIS [34], на яку було накладено адитивний білий гауссівський шум з стандартним відхиленням 30, 40, 50. Для виміру якості роботи алгоритму використовувалась метрика відношення сигнал/шум PSNR. Результати наведено в таблиці 3.2. Приклади роботи запропонованого алгоритму наведено на рисунках 3.6-3.8.

***Таблиця 3.2*** Результати роботи алгоритму на датасеті DAVIS

| σ | Запропонований алгоритм, PSNR дБ | VNLB, PSNR дБ | VBM4D, PSNR дБ | VNLnet, PSNR дБ | DVDnet, PSNR дБ |
|---|---|---|---|---|---|
| 30 | 29.5 | 33.73 | 31.65 | - | 34.08 |
| 40 | 28.4 | 32.32 | 30.05 | 32.32 | 32.86 |
| 50 | 26.9 | 31.13 | 28.80 | 31.43 | 31.85 |

Було проведено тестування при сильному рівні AWGN для визначення порогового значення роботи алгоритму. На рисунку 3.10 зображено приклад роботи алгоритму для синтетичного шуму за формулою 3.2 з параметрами σ =175 Ag =1, Dg=1. Початкове значення PSNR між зашумленими та чистими

фреймами склало 9.3дБ, після застосування алгориму вдалося добитися 14.75дБ. При такому відношенні сигнал шум на відео присутні значні втрати деталей та викревлення початкових кольорів зображення через високу кількість втраченої інформації на початковому шумному фреймі, але при цьому застосування алгоритму дозволяє побачити основні контури об'єктів на зображенні.

Також було виконано порівняльний аналіз швидкості роботи алгоритму в порівнянні з існуючими рішеннями. Для цього на робочій станції, що використовувалась в дослідження було виконано тестовий запуск коду існуючих алгоритмів, що знаходяться у відкритому доступі [25,27-32]. Для тестування використовувались відео з розширенням 960 на 540 пікселів, для розрахунків використовувся GPU NVDIA RTX3060. За допомогою бібліотеки time було вимірано час на обробку одного фрейму. При цьому вимірювався лише час проходу фрейму через нейронну мережу, час попередньої та постобробки фреймів не враховувався. Результати наведено на рисунку 3.9. В середньому запропонований алгоритм обробляє фрейм при розширенні 960 на 540 без врахування додаткової обробки за 0.0057 секунди. Такої переваги вдалося досягти шляхом використання GPU при роботі нейромережі та спрощенної архітектури у порівнянні з конкуруючими алгоритмами.

Підсумовуючи дані наведені в таблицях 3.2 та рисунках 3.7-3.9 можна зазначити, що запропонований алгоритм має програш у продуктивності за метрикою PNSR в середньому на 1.1дБ, у порівнянні з DVDnet, що має найкращі результати. Причиною цього є більш проста архітектура, що призводить до неможливості вилучення настільки складних ознак з фрейму. При цьому через просту архітектуру запропонований алгоритм має перевагу у продуктивності перед DVDnet в 98.58%. Запропонованний алгоритм в основному орієнтованим на боротьбу з AWGN шумом, але при необхідності може бути навчений на роботу з імпульсним чи Пуассонівським шумом.

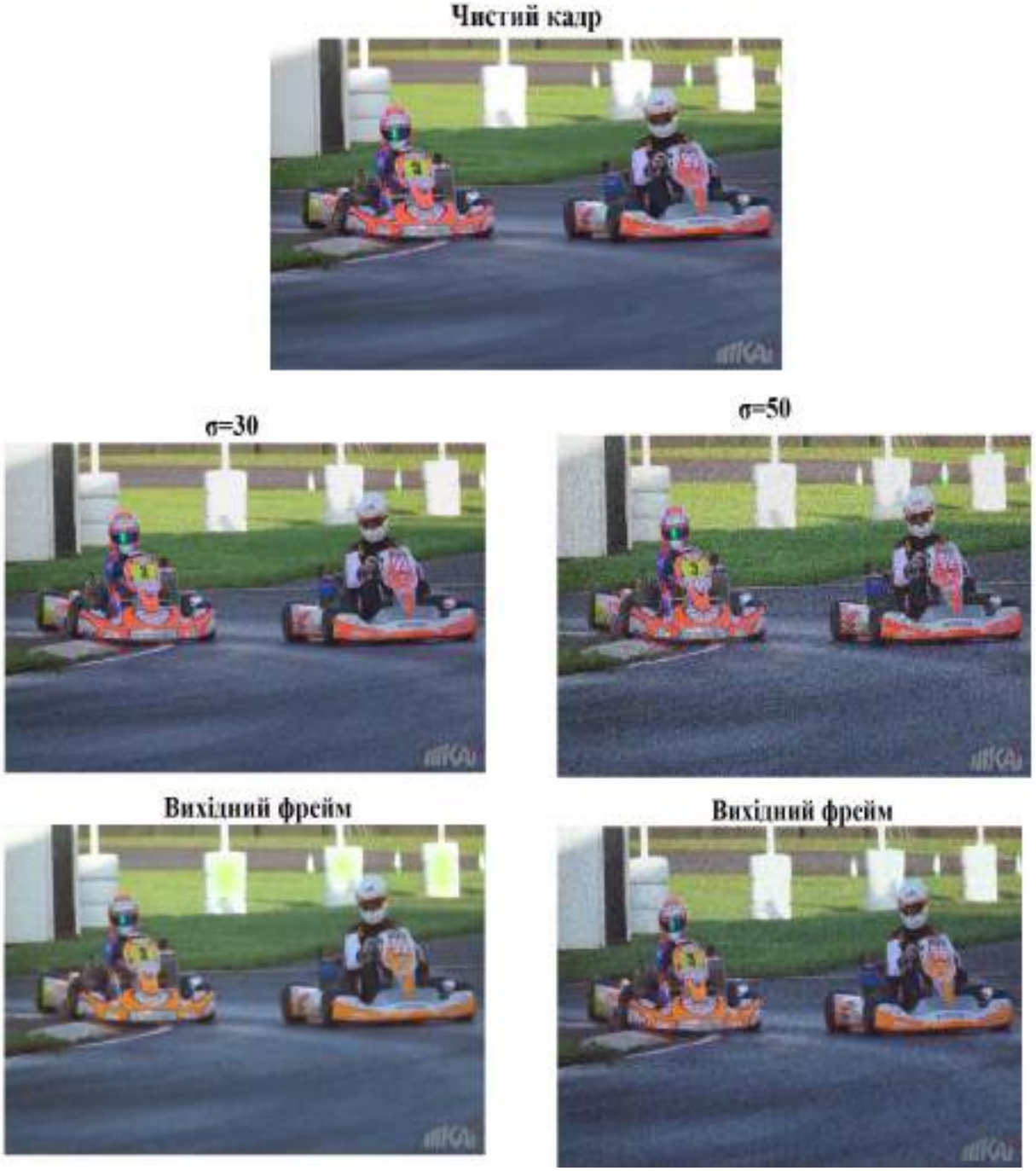


Рисунок 3.7 — Результати роботи алгоритму при AWGN σ=30 та 50


Рисунок 3.8 — Приклад очищенного відео зашумленного за формулою 3.2

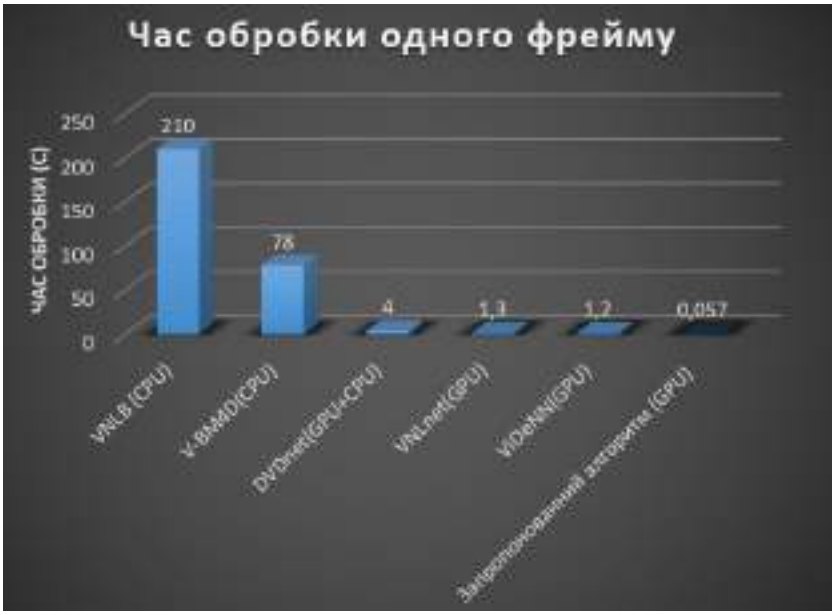


Рисунок 3.9 — Порівняльний результат продуктивності алгоритмів. Швидкість обробки одного фрейму розширенням 960 на 540 без врахування попередньої обробки

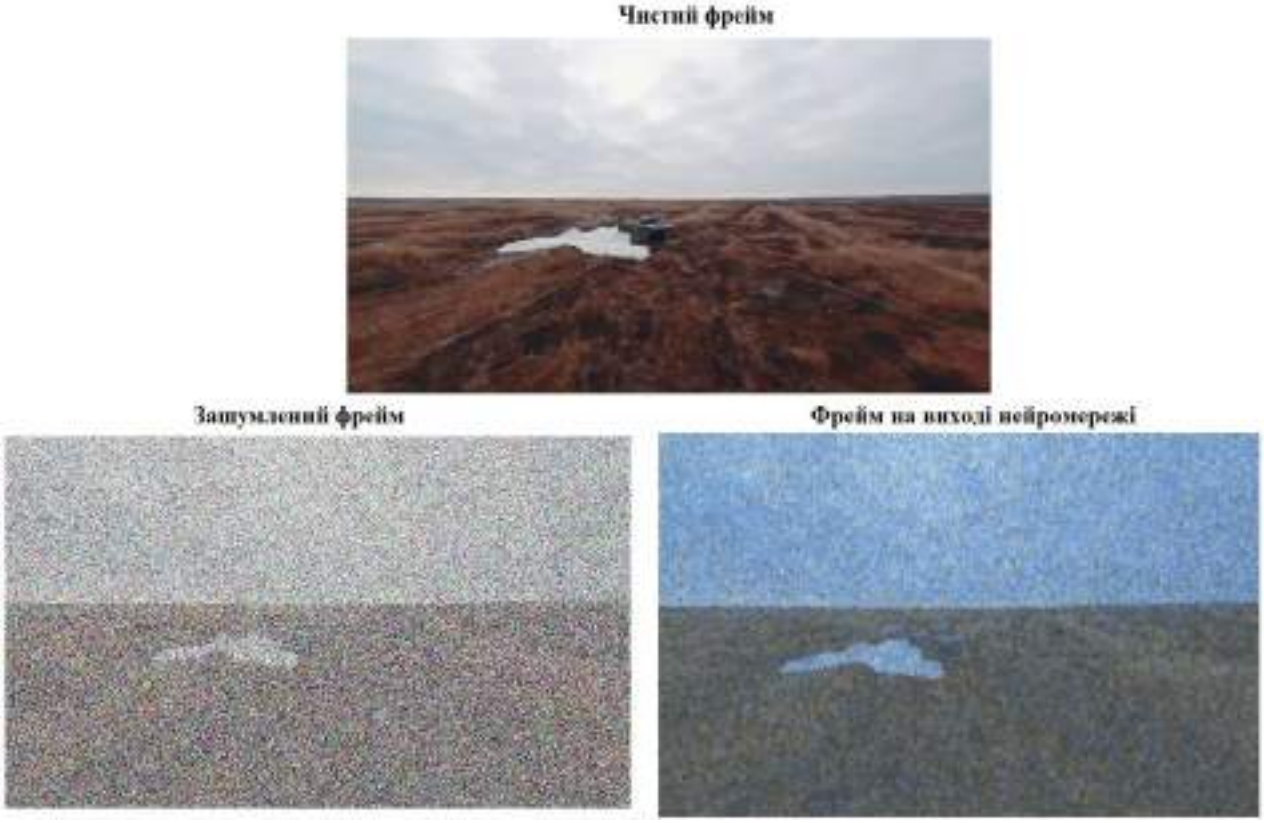


Рисунок 3.10 — Приклад очищенного з накладанням шуму за формулою 3.2 з параметрами σ =175 Ag =1, Dg=1.

Недоліком, який є характерним як для сучасних нейромережевих алгоритмів, так і для запропонованого алгоритму, є незадовільна здатність протистояти сильним реальним шумам, наприклад з FPV дрону. Такий фрейм окрім AWGN шуму також має сильний імпульсний шум, приклад такого фрейму наведено на рисунку 3.11. В окремих випадках коли шум на відео має схожу структуру з тим, на якому проводилось навчання нейромережа досягає задовільних результатів. Основною складністю обробки таких фреймів нейромережевим алгоритмом є високий рівень стохастичності таких шумів, кількість інформації, що втрачається через такі шуми та їх складна структура. Неможливість отримання пар «чистих» та «зашумлених» зображень ускладнює завдання використання методики навчання з вчителем для створення такої нейромережі. Створення синтетичних датасетів за такими шумами та використання методики некерованого навчання також не призводить до необхідного результату через стохастичність таких шумів.

Для вирішення проблеми протидії реальним шумам можна застосувати ансамблі нейромереж. Кожна нейромережа в ансамблі може бути навчена на різних типах шумів, з різними параметрами, що дозволяє ансамблю як цілому протистояти шуму. Недоліком такого методу є сильне зниження продуктивності. Тому використання такого методу в сучасних умовах для роботи в рельному часі є неможливим.

Ще одним варіантом протидії реальним шумам є використання однієї нейромережі, але з більш складною архітектурою, що містить більше фільтрів. Такі архітектури можуть навчатися та вилучати не лише прості ознаки даних, такі як краї, кути, кольорові плями та інші візуальні характеристики, а й більш більш високорівневі ознаки, такі як форми об'єктів або навіть цілі об'єкти. Недоліком цього підходу є також витрати значних обчислювальних ресурсів, та проблема перенавчання, що виникає через велику кількість нейронів.

Рисунок 3.11 — Приклад відеофрейму ураженого реальними шумами

Іншою основною проблемою сучасних алгоритмів є потреба у значних обчислювальних ресурсах, що робить неможливою роботу в реальному часі. Ця проблема може бути частково вирішена шляхом покращення обладнання, розробкою нових методів стиснення та вибору оптимальних параметрів масштабу для вхідного зображення, щоб дотримуватися балансу між якістю та продуктивністю. Важливими стратегіями для підвищення продуктивності є паралельна обробка та конвеєризація даних. Паралельна обробка дозволяє

одночасно виконувати кілька операцій, що значно прискорює процес обчислень. Конвеєризація даних - це процес, в якому вхідні дані проходять через послідовність етапів обробки, кожен з яких виконує певну частину загальної задачі. Кожен етап конвеєра може обробляти різні частини вхідних даних одночасно, що дозволяє ефективно використовувати обчислювальні ресурси. Ці методи можуть бути використані окремо або в комбінації для підвищення продуктивності нейромереж.

### 3.6. Висновки до розділу 3

1. Розроблено блоки знешумлення шляхом модифікації архітектури згорткової нейронної мережі U-Net.

2. На основі розроблених блоків побудовано нейромережевий алгоритм боротьби з шумами на мові Python з використанням бібліотеки Pythorch.

3. Розроблено та застосовано методику некерованого навчання для побудованої мережі.

4. Виконано порівняльний аналіз розробленого алгоритму.

# Розділ 4 РОЗРОБЛЕННЯ ПРОТОТИПУ СТАРТАП-ПРОЕКТУ

## 4.1. Опис концепції проекту

Цей розділ присвячено економічному обгрунтуванню впровадження стартап-проекту під назвою “Інтелектуальна система ідентифікації об’єктів на базі створеного ансамблю гібридних нейронних мереж”. Планується, що описана технологія буде втілена у форматі десктопного додатка, яким можна буде керувати з комп’ютера. Процес включатиме:

– впровадження технології, описаної в цій роботі;

– розробку стратегії для введення конкурентоспроможного продукту на ринок та подальше зростання стартапу.

***Таблиця 4.1*** Опис ідеї проекту

| Суть ідеї | Напрямки застосування | Переваги для користувача |
|---|---|---|
| Суть ідеї полягає в створенні додатки, що дозволяв би користувачу проводити знешумлення фото чи відео використовуючи нейромережу | 1.Використання алгоритму для знешумлення | Завантажений додаток, та матеріали з шумом подібним до того на якому навчався алгоритм |
| | 2.Використання алгоритму в сенсорних мережах, тощо для знешумлення в реальному часі | Наявність графічного процессора з підтримкою бібліотеки CUDA, встановлена бібліотека. Завантажений додаток |

В таблиці 4.1 представлено основну концепцію, потенційні області застосування та переваги для користувача стартап-проекту. Для втілення цього проекту будуть використані нейронні мережі для знешумлення фото та

відео. Цей програмний продукт матиме можливість працювати в режимі реального часу та зможе видаляти певні види шумів, якщо вони схожі до тих на яких навчалась нейронна мережа. Користувачі можуть самостійно проводити знешумлення, для робити в реальному часі алгоритм потребуватиму графічний процесор, з CUDA ядрами. Для втілення стартап-проекту необхідно визначити його основні характеристики (табл. 4.2).

Таблиця 4.1Опис концепції стартап-проекту

| № | Техніко-економічні характеристи-ки стартапу | Конкуренти | | | W (Слабка сторона) | N (Нейтральна сторона) | S (Сильна сторона) |
|---|---|---|---|---|---|---|---|
| | | Мій | Noise2Noise | ViDeNN | | | |
| 1 | Форма виконання | Дода-ток | Скрі-пт | Скр-іпт | | | + |
| 2 | Собівартість | Низь-ка | Низь-ка | Низь-ка | | + | |
| 3 | Можливість роботи в реальному часі | Так | Ні | Част ково | | | + |
| 4 | Крос-платформені сть | Так | Ні | Ні | | | + |

Визначення списку слабких, сильних та нейтральних атрибутів та властивостей потенційного продукту є основою для формування його конкурентоспроможності. Сильними сторонами цього проекту є його реалізація у вигляді десктопного додатку та можливість функціонування без потужного комп’ютера, але у такому випадку робота в режимі реального часу буде недоступна. Оскільки усі додатки не потребують значних фінансових

вкладень собівартість є нейтральною стороною Таким чином, цей проект можна вважати конкурентоспроможним.

### 4.2. Аудит технології проекту

У рамках цього підрозділу необхідно провести оцінку технології, яка дозволить реалізувати ідею проекту (технологія створення продукту).

***Таблиця 4.3*** Технологічна можливість реалізації проекту

| № | Концепція проекту | Технологія для реалізації | Наявність технологій | Доступність технологій |
|---|---|---|---|---|
| 1 | Розробка додатку | Нейромережа | Наявна | Безкоштовна, доступна |
| *Технологія, обрана для втілення концепції проекту*: для розробки додатку була вибрана технологія нейронних мереж, яка є вільною та відкритою для використання. | | | | |

Висновок: Таким чином, проект буде втілено за допомогою технології нейронних мереж, яка буде використовуватися для покращення якості зображень та відео. Вона є безкоштовною та доступною на ринку.

### 4.3. Аналіз ринкових перспектив запуску стартап-проекту.

Ідентифікація ринкових перспектив, які можуть бути застосовані при введенні проекту на ринок, та ринкових ризиків, які можуть перешкодити його успішному виконанню, допомагає визначати стратегії розвитку проекту, враховуючи ринкові умови, вимоги потенційних споживачів та пропозиції конкурентних проектів [32,33]. Розпочинаємо з аналізу попиту: присутність попиту, обсяг та тенденції розвитку ринку.

***Таблиця 4.4*** Основні характеристики потенційного ринку

| № | Показники ринку | Характеристики |
|---|---|---|
| 1 | Основні гравці | 3 |
| 2 | Загальні обсяги продаж, грн/ум.од | Дані відсутні |
| 3 | Динаміка ринку | Попит зростає |
| 4 | Обмеження для входу на ринок | Відсутні |
| 5 | Конкретні вимоги щодо стандартизації та сертифікації. | Відсутні |

Висновок: Дані про обсяги продажів у відповідній сфері не було знайдено у відкритому просторі. Проект не має обмежень для входу на ринок та специфічних вимог до стандартизації та сертифікації, оскільки він буде реалізований у вигляді десктопного додатку з необмеженим доступом. Наступним кроком є визначення потенційних груп клієнтів, їх характеристик, та формування приблизного списку вимог до продукту (табл. 4.5). Характеристики стартап-проекту визначено: основна потреба, що формує ринок - це десктопний додаток та мобільний додатки; основні цільові сегменти ринку включають охоронні компанії, та користувачі що знешумлюють особисті відео та фото матеріали. Поведінці різних потенційних цільових груп клієнтів включають отримання підвищення якості фото та відео для перегляду чи подальшої обробки нейромережами сегментації та класифікації; основні вимоги до споживачів підтверджено - десктопний додаток повинен бути зручним у користуванні, надійним та швидким.

Наступним кроком є створення таблиць факторів, які сприяють впровадженню проекту на ринок, та факторів, які можуть стати перешкодою для цього (табл. 4.6-4.7).

***Таблиця 4.5*** Основні характеристики потенційного ринку

| № | Потреби, ринку | Цільова группа | Відмінності у поведінці груп клієнтів | Вимоги цільової групи |
|---|---|---|---|---|
| 1 | Портативний додаток для знешумлення фото та відео | Охоронні компанії, користувачі, що видаляють шум з особистих матеріалів | Потреба у обробці матеріалів у реальному часі | Рішення має бути зручним у використанні, надійним та швидким |

***Таблиця 4.6*** Фактори загроз

| № | Фактор загрози | Зміст загрози | Відповідь компанії |
|---|---|---|---|
| 1 | Конкуренція | Вихід алгоритму, що має кращу продуктивність за запропонований алгоритм | - Модифікація алгоритму<br>- Вихід з ринку |
| 2 | Незаконне розповсюдження софту | Розповсюдження неліцензійних копій додатку | - Звернення до відповідних державних органів |

***Таблиця 4.7*** Фактори Можливостей

| № | Фактор можливості | Зміст можливості | Відповідь компанії |
|---|---|---|---|
| 1 | Підвищення попиту користувачів на подібні додатки | Зростання числа користувачів, що мають потребу у знешумленні візуальних матеріалів | - Проводити активну рекламну кампанію |
| 2 | Винайдення кращого алгоритму | Покращення можливостей алгоритму, що ще більше буде виділяти його на фоні конкурентів | - Проведення активної піар кампанії з демонстраціями можливостей алгоритму |

Найсерйознішими загрозами для проекту є конкуренція (вихід нового алгоритму кращого за характеристиками). Для усунення загроз необхідно постійно покращувати алгоритм підвищуючи продуктивність, збирати зворотній зв’язок з користувачів та відповідно цих відгуків покращувати інтерфейс та вводити нові особливості, яких потребують користувачі. Меншою загрозою є піратство, для боротьби з ним треба підвищувати захищеність додатку та контактувати з контролюючими органами.

Було визначено основні фактори, що сприяють ринковому впровадженню проекту: підвищення потреби користувачів у відповідному додатку, та як результат боротьби з конкуренцією розроблення алгоритму, що буде мати характеристики кращі від конкурентів. Основними реакціями компанії є активне проведення піар кампанії у соціальних мережах з демонстраціями можливостей алгоритму.

Наступним кроком є визначення загальних характеристик конкуренції на ринку (табл. 5.8).

***Таблиця 4.8*** Аналіз конкуренції на ринку

| № | Особливості середовища | Прояв характеристики | Вплив на проект та його реакція |
|---|---|---|---|
| 1 | Вказати тип конкуренції: - монополістична конкуренція | Існує багато невеликих алгоритмів на ринку | Продуктивність додатку повинна бути краща за товари конкуренів |
| 2 | За рівнем конкурентної боротьби: - міжнародний | Додатки поширюються у середовищі інтернет | Необхідність мати характеристики кращі за конкурентів. Надати користувачам перевагу у порівнянні з конкурентами |
| 3 | Галузева ознака: - міжгалузава | Алгоритми конкурують при обробки відео та фото у різних сферах | Створити алгоритм, що буде спроможний працювати з різними видами шумів |
| 4 | Конкуренція за видами товарів: товарно-видова | Товари є ідентичними, а саме - це програмне забезпечення. | Розробити додаток, враховуючи слабкі сторони конкурентів. |
| 5 | За характером конкурентних переваг: - нецінова | Розробка алгоритму не займає багато фінансових ресурсів. Головною характеристикою є продуктивність | Створити алгоритм з кращими характеристиками продуктивності, ніж у конкрументів. Також слід розробити простий та зручний інтерфейс для користувачів |
| 6 | За інтенсивністю: - не марочна | Бренди відсутні | - |

Було проведено дослідження конкурентного середовища на ринку, в ході якого було встановлено наступне: тип конкуренції - монополістична конкуренція; рівень конкурентної боротьби - міжнародний; міжгалузева ознака конкуренції; види

товарів, що беруть участь у конкуренції - товарно-видові; характер конкурентних переваг - неціновий; інтенсивність конкуренції - не марочна.

Було також запропоновано можливі стратегії для підвищення конкурентоспроможності компанії: підвищення характеристик продуктивності кращі за конкурентів, проведення аналізу та врахувнаня недоліків конкурентів, розробка зручного та простого інтерфейсу.

Наступним кроком є розробка переліку факторів конкурентоспроможності для ринку на основі аналізу складових моделі М. Портера. Результати наведено в табл. 4.9.

***Таблиця 4.9*** Дослідження конкуренції за М.Портером

| | | | | | |
|---|---|---|---|---|---|
| Складові аналізу | Прямі конкуренти в галузі | Потенційні конкуренти | Постачальники | Клієнти | Товари-замінники |
| | Визначити сильні та слабі сторони конкурентів | Дослідити бар'єри входу на ринок | Провести аналіз впливу постачальників | Встановити основні вимоги користувачів до продукту | Визначити вплив товарів замінників на конкуренцію |
| Висновки | Конкуренти переважно поширюють свої товари безкоштовно, але вони є складними у використанні | Більшість конкурентних додатків поширюються безкоштовно, що є завадою входу на ринок | Постачальники відсутні та не впливають на конкуренцію | Критичними показниками для користувачів є простота у використані та продуктивність | Товари замінники є складними у використанні тому не надають сильної конкуренції |

Враховуючи конкурентне середовище, можна стверджувати, що проект має потенціал для успішної роботи на ринку. Серед існуючих конкурентів немає таких, які б могли витіснити його, оскільки розроблене рішення спрямоване на спрощення та прискорення роботи фахівця. Основні переваги продукту, які можуть сприяти його конкурентоспроможності на ринку, включають надійність, зручність користувацького інтерфейсу, простоту використання, безпеку та застосування підходу з використанням нейромереж.

На основі вищенаведеного аналізу визначається та обгрунтовується список факторів конкурентоспроможності (табл. 4.10).

***Таблиця 4.10*** Фактори конкурентоспроможності

| № | Фактор и конкурентоспроможності | Обґрунтування |
|---|---|---|
| 1 | Застосування нейромережевого підходу | Дозволяє швидко проводити знешумлення на попередньо навченний мережах без додаткових налаштувань |
| 2 | Простота у використані | Алгоритм працює на пристрої користувача без необхідності додаткових налаштувань |
| 3 | Продуктивність | Можливість додатку працювати у режимі реального часу |

Було визначено ключові фактори конкурентоспроможності, які будуть представлені на ринку. На основі визначених факторів конкурентоспроможності проводиться аналіз переваг та недоліків запропонованого проекту(табл. 4.11).

***Таблиця 4.11*** Аналіз переваг та недоліків проекту

| № | Фактор конкурентопроможності | Бали | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | -3 | -2 | -1 | 0 | +1 | +2 | +3 |
| 1 | Застосування нейромережевого підходу | 10 | | | | + | | | |
| 2 | Простота у використані | 20 | + | | | | | | |
| 3 | Продуктивність | 20 | | | + | | | | |

Було проведено порівняльний аналіз переваг нашого проекту та проектів-конкурентів. Найбільше балів отримано за такі фактори: простота у використанні та продуктивні.

Наступним кроком є складання SWOT-аналізу (табл. 4.12).

***Таблиця 4.12*** SWOT аналіз проекту

| Сильні сторони (S): | Слабкі сторони (W): |
|---|---|
| - Простий та зрозумілий інтерфейс<br>- Надійність<br>- Продуктивність | - Необхідність мати графічний процессор з ядрами CUDA для обробці в реальному часі |
| Можливості(O): | Загрози (T): |
| -Зростання інтересу користувачів до сфери застосування додатку<br>- Покращення продуктивності додатку | - Конкуренція<br>- Неліцінзійне розповсюдження додатку |

Отже, внутрішні ресурси та можливості компанії щодо виведення продукту на ринок характеризуються наступними сильними та слабкими сторонами: сильні - простота користувацького інтерфейсу, надійність, продуктивність; слабкі - Необхідність мати графічний процессор з ядрами CUDA. Ринкові можливості та зовнішнє середовище компанії характеризуються наступними можливостями та загрозами: можливості - зростання інтересу користувачів, та різке покращення характеристик додатку, конкуренція та піратство.

На основі SWOT-аналізу формуються варіанти ринкового впровадження стартап-проекту (табл. 5.13).

У результаті було обрано шлях проведення піар кампанії через популярні соціальні мережі. Перевагами цього шляху є відсутність потреби у грошах для реалізації та середній час для проведення кампанії. Зокрема, рамках кампанії планується створення рекламних кліпів- бенчмарків, що демонструють роботу алгоритму на Youtube тощо.

***Таблиця 4.13*** Можливості ринкового запровадження проекту.

| № | Комплекс заходів ринкової поведінки | Отримання ресурсів | Терміни реалізації |
|---|---|---|---|
| 1 | Створення піар кампанії через соціальні мережі | Основний ресурс – час, цей ресурс наявний | 2 місяці |
| 2 | Презентація продукту на наукових конференціях, хакатонах, IT заходах | Основний ресурс – час та невелика кількість грошей для участі, цей ресурс наявний | 4 місяці |
| 3 | Замовлення рекламних послуг у медіа інфлюенсерів | Основний ресурс – велика кількість грошей, цей ресурс відсутній | 1 місяць |

## 4.4 Створення ринкової стратегії проекту

Першим етапом у розроблені ринкової стратегії є втановлення стратегії ринкового проникнення: представлення основний груп споживачів товару (табл. 414). На першому єтапі було обрано охоронні компанії та користувачів, що займаються знешумленням особистих матеріалів. Операторам дронів на першому етапі алгоритм не зможе надати необхідну якість. Також було визначено основні характеристики для задоволення цих сегментів

користувачів. Це робота в реальному часі, простота у використанні та надійність

***Таблиця 4.14*** Цільові групи споживачів проекту

| № | Цільова група потенційних клієнтів | Вірогідність долучення споживачів до продукту | Орієнтов-ний попит | Інтенсивність конкуренції | Параметри для входу у сегмент |
|---|---|---|---|---|---|
| 1 | Користувачі, що працюють з особистими данними | Висока | Високий | Висока | Простота у використанні |
| 2 | Охоронні компанії | Середня | Середній | Середня | Робота в реальному часі, надійність, безпека |
| 3 | Користувачі дронів | Низька | Високий | Низька | Робота в реальному часі, робота зі складними реальними шумами |
| У якості цільових груп на першому етапі було обрано охоронні компанії та користувачів, що займаються знешумленням особистих матеріалів | | | | | |

На наступному випадку визначаємо базову стратегію розвитку( табл 4.15)

Таким чином, було вирішено розвивати проект через створення додатку, який використовує технологію нейромереж. Була визначена основна стратегія розвитку - диференціація, засобом втілення якої є позиціонування на ринку. Це означає, що додаток матиме ключові відмінності від конкуруючих товарів, такі як швидкість, простота використання, безпека та надійність.

***Таблиця 4.15*** Основна стратегія розвитку проекту

| Стратегія розвитку | Стратегія з охоплення ринку | Основні конкурентні параметри відповідно до стратегії | Базова стратегія розвитку |
|---|---|---|---|
| Реалізація продукту з використанням нейромережевого алгоритму | Ринкове позиціонування | Робота в реальному часі, простота у використанні, безпека та надійність | Стратегія диференціації |

Далі обираємо стратегію конкурентних дій (табл. 4.16)

***Таблиця 4.16*** Базова стратегія конкурентних дій

| Чи є проект новаторським на ринку? | Чи планує компанія залучати нових клієнтів або відбирати існуючих у конкурентів? | Чи планує компанія копіювати ключові характеристики товару конкурента? | Тип стратегії |
|---|---|---|---|
| Ні, на ринку є аналогічні товари, але вони не мають деяких необхідних функцій. | Так, метою компанії є залучення нових клієнтів та, частково, відбір | Так, основним завданням є розробка додатку з використанням нейромережі простого інтерфейсу | Стратегія зайняття конкурентної ніші. |

| | | | |
|---|---|---|---|
| | існуючих у конкурентів | користувача та технічної підтримки. | |

Стратегія конкурентної дії полягає в зайнятті конкурентної ніші, оскільки програмний продукт буде цілеспрямований на конкретний ринковий сегмент - Інтернет речей. Він також матиме технічну підтримку у вигляді оновлень мобільного додатку, що є основною перевагою перед конкурентами і сприятиме формуванню довіри та лояльності споживачів. Наступним кроком є визначення стратегії позиціонування проекту (табл. 4.17).

***Таблиця 4.17*** Стратегія позиціонування

| Вимоги цільової аудиторії до товару | Базова стратегія розвитку | Конкурентні спроможності проекту | Асоціації, що формують комплексну позиції проекту |
|---|---|---|---|
| Простота використання інтерфейсу, швидкість роботи, надійність та безпека. | Диференціація | Простий у використанні інтерфейс дозволить користувачам без спеціального досвіду отримувати знешумлені матеріали, що будуть надійно захищені | Безпека, простота у використані, швидкість роботи |

Таким чином, було встановлено стратегію позиціонування, яка включає в себе основні вимоги цільової аудиторії до продукту: простоту інтерфейсу, швидкодію, надійність та безпеку. Базовою стратегією розвитку обрано диференціацію. Проект характеризується простим користувацьким інтерфейсом, який дозволяє здобувати важливі дані та слідкувати за подіями в режимі реального часу, а також забезпечує безпеку та надійність. Розроблено комплексу позицію проекту, що складається з швидкодії, безпеки та простоти у використанні.

## 4.5 Створення маркетингової стратегії для стартап-проекту

Початковим етапом є розробка маркетингової концепції продукту, який буде представлений споживачу. Для цього в таблиці 4.18 необхідно узагальнити результати конкурентоспроможності продукту.

***Таблиця 4.18*** Переваги проекту

| № | Параметр | Вигода, що пропонує проект | Ключова перевага перед конкурентами |
|---|---|---|---|
| 1 | Швидкодія | Додаток може працювати на мобільних пристроях, а при наявності GPU може працювати в режимі реального часу | Робота на мобільних пристроях, робота в режимі реального часу |
| 2 | Простота інтерфейсу | Простота роботи додатку | Користувач може працювати в додатку без особливих додаткових технічних знань |
| 3 | Безпека та надійність | Можливість шифрування відеопотоку | Додаток не збирає додаткові дані окрім відеопотоку, за необхідності шифруючи його |

Наступним етапом є розробка трирівневої маркетингової моделі товару (табл 4.19)

Таким чином, було викладено три рівні моделі продукту: продукт представлятиме собою десктопний додаток, ціллю якого є знешумлення фото та відео. Особливості проекту: комфорт та простота користувацького інтерфейсу, швидкість виконання, надійність, безпека згідно зі світовими стандартами. Додаток буде захищений від копіювання за допомогою патенту. Наступним етапом є встановлення цінових меж (табл. 4.20). Аналіз виконується за допомогою експертного методу.

***Таблиця 4.19*** Трирівнева модель товару

<table>
<tr><td>Рівень</td><td colspan="2">Складові</td></tr>
<tr><td>1. Задум товару</td><td colspan="2">Алгоритм для видалення шумів з фото та відео в режимі реального часу</td></tr>
<tr><td rowspan="4">2. Виконання товару</td><td>1. Зручність інтерфейсу<br>2. Швидкодія<br>3. Надійність<br>4. Відповідність протоколам безпеки</td><td>Швидкість фреймів/секунду</td></tr>
<tr><td colspan="2">Якість: застосування метрик PSNR, MSSIM</td></tr>
<tr><td colspan="2">Маркування відсутнє</td></tr>
<tr><td colspan="2">Моя компанія «Easy Denoise»</td></tr>
<tr><td rowspan="2">3. Підкріплення товару</td><td colspan="2">Безкоштовне встановлення, наявність реклами</td></tr>
<tr><td colspan="2">Постійне оновлення додатку додавання нових моделей</td></tr>
<tr><td colspan="3">Яким чином потенційний продукт буде відстежуватися від копіювання: за допомогою захисту інтелектуальної власності та патенту.</td></tr>
</table>

***Таблиця 4.20*** Визначення межі встановлення ціни

| Ціни на товари замінники, грн | Ціни на товари аналоги, грн | Доходи цільової групи споживачів, грн | Межі встановлення ціни на продукт, грн |
|---|---|---|---|
| 200 | Безкоштовно | 16000 | Безкоштовно+ реклама/40 грн |

Оскільки товари замінники поширюються безкоштовно, але при цьому для користування потребують спеціалізованих навичок пропонується поширювати продукт безкоштовно. У якості монетизації пропонується реклама для прибирання якої необхідно буде купити повну версію додатку за 40 грн.

Наступним етапом є встановлення найкращої системи збуту, в рамках якої буде прийнято рішення (табл. 4.21).

***Таблиця 4.21*** Створення системи збуту

| Збутова політика клієнтів | Функції, що покладаються на постачальника товару | Глибина каналу збуту | Оптимальна система збуту |
|---|---|---|---|
| Перегляд реклами, покупка повної версії без реклами | Поширення | Напряму, через одного посередника | Пряма та через посередників |

Було обрано систему збуту шляхом безкоштовного поширення додатку з переглядом рекламних повідомлень всередині додатку, що буде монетизовано, покупка повної версії додатку у якій цю рекламу буде відключено.

Розроблено концепція маркетингових комунікацій (табл. 4.22).

***Таблиця 4.22*** Концепція маркетингових комунікацій

| № | Поведінка цільової аудиторії | Канали збуту | Особливості позиціювання додатку | Мета рекламного повідомлення | Концепція рекламного повідомлення |
|---|---|---|---|---|---|
| 1 | Встановлення додатку | Інтернет | Швидкість роботи, простота використання, безпека та надійність. | Презентувати переваги додатку, включаючи його переваги перед конкурентами. Демонстраційний відеоролик про використання продукту. | Відео-бенчмарк, що демонструє ефективність |

Таким чином, була сформована концепція маркетингових комунікацій: мобільний додаток може бути завантажений з диску або через інтернет,

основний канал комунікації - інтернет. Ключові аспекти для позиціонування включають швидкість роботи, простоту використання, безпеку та надійність. Основна мета рекламного повідомлення - демонструвати переваги додатку, включаючи його переваги перед конкурентами. Концепція рекламного звернення буде представлена у вигляді демонстраційного відео про використання додатку.

### 4.6. Висновки до розділу 4

1. У цьому розділі було розглянуто ключові елементи введення на ринок додатку для знешумлення фото та відео матеріалів. Пропонований продукт важливий для користувачів, що прагнуть підвищити якість своїх фото та відео матеріалів для особистого використання чи подальшої обробки нейромережевими алгоритмам сегментації та класифікації.

2. У межах цього розділу було визначено список слабких, сильних та нейтральних характеристик та властивостей потенційного продукту для формування його конкурентоспроможності. Було обрано технологію реалізації проекту: для створення додатку було вибрано технологію нейромереж, яка є безкоштовною і з якою члени проекту мають досвід роботи.

3. На підставі проведених досліджень, проект має потенціал для комерційного впровадження на ринку. Існують перспективи для його впровадження, враховуючи цільові групи клієнтів, бар’єри входження не є великими, а проект має важливі переваги перед конкурентами: швидкість роботи, простота використання, безпека, надійність

4. На основі аналізу отриманих результатів, можна прийти до висновку, що подальше реалізація проекту є виправданою.

# ВИСНОВКИ

У даній магістерській дисертації для вирішення поставленої задачі по розробці нейромережевого алгоритму боротьби з шумами в режимі реального часу було виконано наступні завдання:

1. Досліджено вплив основних типів шуму на відеопотокік.

2. Проведено огляд основних методів фільтрації для боротьби з шумами. Встановлено їх переваги та недоліки.

3. Виконано порівняння методів глибокого навчання. Було порівняно підходи засновані на методі навчання з вчителем  та методів некерованого навчання. При наявності датасету складеного з пар «чистих» та «зашумлених» зображень пропонується використовувати метод навчання з вчителем. У випадках реальних шумів при неможливості створити такий датасет краще демонструє себе некероване навчання.

4. Розглянуто сучасні нейромережеві підходи для боротьби з шумами у сфері обробки зображень.

5. Розроблено нейромережевий алгоритм на мові Python для боротьби з шумами в режимі реального часу. Зокрема, було реалізовано методику для створення датасету, створено нейромережевий блок видалення шумів щляхом модифікації U-Net архітектури та розроблено методику некерованого навчання. Запропонований алгоритм у порівнянні на датасеті DAVIS в середньому програє у метриці PSNR на 1.1дБ перед найкращим варіантом UDVD, при цьому перевагою запропонованого алгоритму є продуктивність, що складає 98.58% порівнюючи з UDVD.

6. В процесі створення стартап-проекту було проведено дослідження ключових елементів введення в обіг додатку видалення шумів з фото та відео матеріалів. Зазначений продукт виявився відповідним для користувачів, що прагнуть очистити свої графічні матеріали від шумів. Було встановлено ряд

слабких, сильних та нейтральних характеристик і властивостей можливого продукту для формування його конкурентної переваги.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Шаховська, Н. Б., & Косар, О. І. (2018). Аналіз поширених методів накладання шуму на зображення. Науковий вісник НЛТУ України. 28(1), 145-149. DOI:10.15421/40280129

2. Фільтрація шумів на цифрових зображеннях з вимірювальною інформацією [Електронний ресурс] – Режим доступу до ресурсу: https://learn.ztu.edu.ua/mod/resource/view.php?id=109164

3. PSNR [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikidata.uk-ua.nina.az/PSNR.html

4. Наконечний, А. Й., & Верес, З. Є. (2018). Використання індексу структурної подібності та м’якого порогування для знаходження коефіцієнтів інтерполяційного фільтра з адаптивним кодуванням. Науковий вісник НЛТУ України, 28(1), 145-149. DOI:10.15421/40280129

5. VMAF - Video Multi-Method Assessment Fusion [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Netflix/vmaf

6. Фільтрація зображень засобами openCV [Електронний ресурс] – Режим доступу до ресурсу: https://ai-tern.in.ua/Filtration.html

7. Basics of Image Processing [Електронний ресурс] – Режим доступу до ресурсу: https://vincmazet.github.io/bip/restoration/denoising.html#denoising

8. Szczepanski, M. (2011). Spatio-Temporal Filters in Video Stream Processing. In Computer Recognition Systems 4 (pp. 421-430). DOI: 10.1007/978-3-642-20320-6_44

9. Yahya, A. A., Tan, J., & Li, L. (2015). Video Noise Reduction Method Using Adaptive Spatial-Temporal Filtering. Discrete Dynamics in Nature and Society, 2015, Article ID 351763. DOI:10.1155/2015/351763

10. Basics of Image Processing [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/dip-high-pass-vs-low-pass-filters

11. Ideal Lowpass Filter [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/matlab-ideal-lowpass-filter-in-image-processing/

12. Ideal Lowpass Filter [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/matlab-ideal-lowpass-filter-in-image-processing/

13. Линовський, А. О., & Мухін, В. Є. (2023). Засоби покращення якості та знешумлення зображень на основі застосування згорткових та рекурентних нейронних мереж. Телекомунікаційні та інформаційні технології, 1(78). DOI:10.31673/2412-4338.2023.018289

14. Федоров, Є. Є., & Уткіна, Т. Ю. (2021). Метод очищення від шуму візуальної біометричної інформації. Вісник Черкаського державного технологічного університету, 4, 5. DOI:10.24025/2306-4412.4.2021.247856

15. Ch, S., Jun, O., Xiring, G. Residual learning of deep convolutional neural networks for image denoising. Journal of Intelligent and Fuzzy Systems, 2019, Volume 37, Issue 2, P. 2809-2818. DOI:10.3233/JIFS-190017.

16. Lei, Z., Wangmeng, Z., Kai, Z. FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising. IEEE Transactions on Image Processing, 2018, Volume 27, Issue 9. DOI: 10.1109/TIP.2018.2839891.

17. Matias, T., Julie, D., Thomas, V. An Analysis and Implementation of the FFDNet Image Denoising Method. Image Processing On Line, 2019, Volume 9, P. 1-25. DOI: 10.5201/ipol.2019.231.

18. Matias Tassano, Julie Delon, and Thomas Veit. Dvdnet: A fast network for deep video denoising. In Proceedings of the IEEE International Conference on Image Processing, 2020, pages 1805–1809.

19. Matias Tassano, Julie Delon, and Thomas Veit. Fastdvdnet: Towards real-time deep video denoising without flow estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pages 1351–1360.

20. H. Yue, C. Cao, L. Liao, R. Chu, and J. Yang. Supervised raw video denoising with a benchmark dataset on dynamic scenes. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pages 2298–2307.

21. Thibaud Ehret, Axel Davy, Jean-Michel Morel, Gabriele Facciolo, and Pablo Arias. Model-blind video denoising via frame-to-frame training. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pages 11361–11370.

22. Axel Davy, Thibaud Ehret, Jean-Michel Morel, Pablo Arias, and Gabriele Facciolo. A non-local CNN for video denoising. In 2019 IEEE International Conference on Image Processing (ICIP), 2019, pages 2409–2413.

23. Valery Dewil, Jeremy Anger, Axel Davy, Thibaud Ehret, Gabriele Facciolo, and Pablo Arias. Self-supervised training for blind multi-frame video denoising. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pages 2724–2734.

24. Claus, M., Gemert, J. Videnn: Deep blind video denoising. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop, 2019, pages 1843–1852.

25. Sheth, Y., Mohan, S., Vincent, J. L., Manzorro, R., Crozier, P. A., Khapra, M. M., Simoncelli, E. P., Fernandez-Granda, C. Unsupervised Deep Video Denoising. Indian Institute of Technology Madras, 2016.

26. Applications of Digital Image Processing XXXIX; 99711T (2016) https://doi.org/10.1117/12.2239260. Event: SPIE Optical Engineering + Applications, 2016, San Diego, California, United States.

27. Arias, P., & Morel, J.-M. (2018). Video denoising via empirical Bayesian estimation of space-time patches. Journal of Mathematical Imaging and Vision.

28. Maggioni, M., Boracchi, G., Foi, A., & Egiazarian, K. (n.d.). Video denoising using separable 4-D nonlocal spatiotemporal transforms. Tampere University of Technology, Finland; Politecnico di Milano, Italy.

29. Davy, A., Ehret, T., Morel, J.-M., Arias, P., & Facciolo, G. (n.d.). Non-Local Video Denoising by CNN. CMLA, ENS Cachan, CNRS, Universite Paris-Saclay, 94235 Cachan, France.

30. Claus, M., & van Gemert, J. (n.d.). ViDeNN: Deep Blind Video Denoising. Computer Vision lab, Delft University of Technology.

31. Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., & Aila, T. (n.d.). Noise2Noise: Learning Image Restoration without Clean Data.

32. Chenyu Y, Qingsong Y, Hongming S, Lars G. (2018). Structurally-Sensitive Multi-Scale Deep Neural Network for Low-Dose CT Denoising.

33. PyTorch Foundation [Електронний ресурс] – Режим доступу до ресурсу https://pytorch.org/foundation

34. Davis Challenge [Електронний ресурс] – Режим доступу до ресурсу https://davischallenge.org

35. Ronneberger, O., Fischer, P., & Brox, T. (n.d.). U-Net: Convolutional Networks for Biomedical Image Segmentation. Computer Science Department and BIOSS Centre for Biological Signalling Studies.

36. Артефакти деконволюції та шахової дошки [Електронний ресурс] – Режим доступу до ресурсу https://distill.pub/2016/deconv-checkerboard/

## ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ

1. Файл model.py

```
import torch
import torch.nn as nn

class DoublConv(nn.Module):
    def __init__(self, inputChan, outChan):
        super(DoublConv, self).__init__()
        self.layers = nn.Sequential(
            nn.Conv2d(inputChan, outChan, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(outChan),
            nn.ReLU(inplace=True),
            nn.Conv2d(outChan, outChan, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(outChan),
            nn.ReLU(inplace=True)
        )

    def forward(self, x):
        return self.layers(x)

class InLayer(nn.Module):
    def __init__(self, num_in_frames, outChan):
        super(InLayer, self).__init__()
        self.intermediate_channels = 30
        self.layers = nn.Sequential(
            nn.Conv2d(num_in_frames * (3+1), num_in_frames *
self.intermediate_channels,
                      kernel_size=3, padding=1, groups=num_in_frames, bias=False),
            nn.BatchNorm2d(num_in_frames * self.intermediate_channels),
```

```
            nn.ReLU(inplace=True),
            nn.Conv2d(num_in_frames * self.intermediate_channels, outChan,
kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(outChan),
            nn.ReLU(inplace=True)
        )

    def forward(self, x):
        return self.layers(x)

class DownscaleLayer(nn.Module):
    def __init__(self, inputChan, outChan):
        super(DownscaleLayer, self).__init__()
        self.layers = nn.Sequential(
            nn.Conv2d(inputChan, outChan, kernel_size=3, padding=1, stride=2,
bias=False),
            nn.BatchNorm2d(outChan),
            nn.ReLU(inplace=True),
            DoublConv(outChan, outChan)
        )

    def forward(self, x):
        return self.layers(x)

class UpscaleLayer(nn.Module):
    def __init__(self, inputChan, outChan):
        super(UpscaleLayer, self).__init__()
        self.layers = nn.Sequential(
            DoublConv(inputChan, inputChan),
            nn.Upsample(scale_factor=2, mode='bilinear', align_corners=False),
```

```
            nn.Conv2d(inputChan, outChan, kernel_size=3, padding=1, bias=False)
        )

    def forward(self, x):
        return self.layers(x)

class OutLayer(nn.Module):
    def __init__(self, inputChan, outChan):
        super(OutLayer, self).__init__()
        self.layers = nn.Sequential(
            nn.Conv2d(inputChan, inputChan, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(inputChan),
            nn.ReLU(inplace=True),
            nn.Conv2d(inputChan, outChan, kernel_size=3, padding=1, bias=False)
        )

    def forward(self, x):
        return self.layers(x)

class Unet(nn.Module):
    def __init__(self, quantity_input_frames=3):
        super(Unet, self).__init__()
        self. unetchannels1 = 32
        self. unetchannels2 = 64
        self. unetchannels3 = 128

        self.input_block = InLayer(num_in_frames=quantity_input_frames,
outChan=self. unetchannels1)
        self.down_block0 = DownscaleLayer(inputChan=self. unetchannels1,
outChan=self. unetchannels2)
```

```
        self.down_block1 = DownscaleLayer(inputChan=self. unetchannels2,
outChan=self. unetchannels3)
        self.up_block2 = UpscaleLayer(inputChan=self. unetchannels3, outChan=self.
unetchannels2)
        self.up_block1 = UpscaleLayer(inputChan=self. unetchannels2, outChan=self.
unetchannels1)
        self.output_block = OutLayer(inputChan=self. unetchannels1, outChan=3)

        self.reset_params()

    @staticmethod
    def weight_init(m):
        if isinstance(m, nn.Conv2d):
            nn.init.kaiming_normal_(m.weight, nonlinearity='relu')

    def reset_params(self):
        for _, m in enumerate(self.modules()):
            self.weight_init(m)

    def forward(self, input0, input1, input2,noisemap):
        x0 = self.input_block(torch.cat((input0,noisemap, input1, noisemap,  input2,
noisemap), dim=1))
        x1 = self.down_block0(x0)
        x2 = self.down_block1(x1)
        x2 = self.up_block2(x2)
        x1 = self.up_block1(x1 + x2)
        x = self.output_block(x0 + x1)
        x = input1 - x
        return x
```

```
class QuatroUNET(nn.Module):
    def __init__(self, quantity_input_frames=5):
        super(QuatroUNET, self).__init__()
        self.quantity_input_frames = quantity_input_frames
        self.temp1 = Unet(quantity_input_frames=3)
        self.temp2 = Unet(quantity_input_frames=3)
        self.reset_params()

    def reset_params(self):
        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.kaiming_normal_(m.weight, nonlinearity='relu')
    def forward(self, *x):
        N, C, H, W = x[1].size()
        noise_std = [torch.std(i, dim=(1, 2, 3), keepdim=True) for i in x]
        noise_map = [i.expand((N, 1, H, W)) for i in noise_std]
        out1 = [self.temp1(x[i], x[i + 1], x[i + 2], noise_map[2]) for i in
                range(len(x) - 2)]
        noise_std_out = [torch.std(i) for i in out1]
        noise_map_out = [i.expand((N, 1, H, W)) for i in noise_std_out]

        out2 = self.temp2(out1[0], out1[1], out1[2], noise_map[2])
        return out2
```

2. Файл mask.py

```
import torch


def generate_mask_pair(img):
    n, c, h, w = img.shape
```

```
    mask1 = torch.zeros(size=(n, c, h, w), dtype=torch.bool, device=img.device)
    mask2 = torch.zeros(size=(n, c, h, w), dtype=torch.bool, device=img.device)

    # блоки 2x2
    rand_i1 = torch.randint(0, 2, (h // 2, w // 2), device=img.device)
    rand_j1 = torch.randint(0, 2, (h // 2, w // 2), device=img.device)
    rand_i2 = torch.randint(0, 2, (h // 2, w // 2), device=img.device)
    rand_j2 = torch.randint(0, 2, (h // 2, w // 2), device=img.device)

    # Переміщення пікселів у маску
    mask1[:, :, rand_i1 * 2, rand_j1 * 2] = True
    mask2[:, :, rand_i2 * 2, rand_j2 * 2] = True

    return mask1, mask2

def generate_subimages(img, mask1, mask2):
    subimage1 = img * mask1
    subimage2 = img * mask2

    subimage1 = torch.clamp(subimage1, min=0, max=1)
    subimage2 = torch.clamp(subimage2, min=0, max=1)

    return subimage1, subimage2
```

3. Файл train.py

```
import os
import glob
import cv2
from tqdm import tqdm
from torch.optim import Adam
from torch.nn import MSELoss, Module
```

```
from torch.autograd import Variable
from pytorch_msssim import ms_ssim
from model import QuatroUNET
import copy
import numpy as np
import random
from mask import *
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

def get_random_subfolder(path):
    subfolders = [f for f in os.listdir(path) if os.path.isdir(os.path.join(path, f))]
    if not subfolders:
        return None
    return os.path.join(path, random.choice(subfolders))

def get_random_videos_from_categories(dataset_path):
    sint_path = get_random_subfolder(os.path.join(dataset_path, 'Sint'))
    sint_path2 = get_random_subfolder(os.path.join(dataset_path, 'Sint'))
    paths = [sint_path,sint_path2]
    random.shuffle(paths)
    return paths
class PSNRLoss(Module):
    def __init__(self):
        super(PSNRLoss, self).__init__()
        self.mse = MSELoss()

    def forward(self, img1, img2):
        mse_result = self.mse(img1, img2)
        psnr = 10 * torch.log10(1 / mse_result)
        return psnr
```

```
def get_subdirectories(path):
    return [os.path.join(path, o) for o in os.listdir(path) if
os.path.isdir(os.path.join(path,o))]

def read_and_preprocess(frame_file, device):
    frame = cv2.imread(frame_file)
    frame = cv2.resize(frame, (1280, 720))  # Зміна розміру кадру до 640x480
    frame = (Variable(torch.from_numpy(frame).permute(2, 0,
1).float().unsqueeze(0))) / 255
    frame = frame.to(device)
    frame = torch.clamp(frame, min=0, max=1)
    return frame

def process(datasets_paths, model, loss_psnr, ms_ssim, type, device):
    total_loss = 0
    total_psnr = 0
    total_msssim = 0
    total_frames = 0
    pbar = tqdm(total=total_frames)
    iter=0
    num=0
    k,j=1,1
    output_path = 'Замінити шлях на той куди буде збережено результати
валідації'
    for dataset_path in datasets_paths:
        noisy_frames_path = os.path.join(dataset_path, 'noisy')
        clean_frames_path = os.path.join(dataset_path, 'clean')
        noisy_frame_files = sorted(glob.glob(noisy_frames_path + '/*.jpg'))
        clean_frame_files = sorted(glob.glob(clean_frames_path + '/*.jpg'))
```

```
        for i in range(len(noisy_frame_files) - 4):
            if type == 0:
                masks = [generate_mask_pair(noisy_frame) for noisy_frame in
noisy_frames]
                subimages = [generate_subimages(noisy_frames[k], masks[k][0],
masks[k][1]) for k in range(5)]

                out = model(*[subimages[k][0] for k in range(5)])
                out2 = model(*[subimages[k][1] for k in range(5)])
                out = torch.clamp(out, min=0, max=1)
                out2 = torch.clamp(out2, min=0, max=1)

                noisy_target = subimages[2][0]
                diff = out - noisy_target
                loss1 = torch.mean(diff ** 2)

                exp_diff = out - out2
                loss2 = torch.mean((diff - exp_diff) ** 2)
                loss = k * loss1 + j * loss2

                psnr = loss_psnr(clean_frame, noisy_frames[2])
                msssim = ms_ssim(clean_frame, noisy_frames[2])

                total_loss += loss.item()
                total_psnr += psnr.item()
                total_msssim += msssim.item()
                total_frames += 1
                pbar.update(1)
                loss.backward()
```

```
            optimizer.step()
            optimizer.zero_grad()

        elif type == 1:
            noisy_frames = [read_and_preprocess(noisy_frame_files[i + k], device)
for k in range(5)]
            clean_frame = read_and_preprocess(clean_frame_files[i + 2], device)

            out = model(*noisy_frames)
            out = torch.clamp(out, min=0, max=1)
            psnr = loss_psnr(clean_frame, out)
            msssim = ms_ssim(clean_frame, out)
            loss = -psnr

            total_loss += loss.item()
            total_psnr += psnr.item()
            total_msssim += msssim.item()
            total_frames += 1
            pbar.update(1)

            clean = (clean_frame * 255).cpu().squeeze(0).permute(1, 2,
0).detach().numpy().astype(np.uint8)
            noisy = (noisy_frames[2] * 255).cpu().squeeze(0).permute(1, 2,
0).detach().numpy().astype(np.uint8)
            out = (out * 255).cpu().squeeze(0).permute(1, 2,
0).detach().numpy().astype(np.uint8)
            cv2.imwrite(os.path.join(output_path, f'out{num}.png'), out)
            cv2.imwrite(os.path.join(output_path, f'clean{num}.png'), clean)
            cv2.imwrite(os.path.join(output_path, f'noisy{num}.png'), noisy)
            num += 1
```

```
        if type == 0:
            print(f'Обробка відео завершена[{iter + 1}/{15}]: {dataset_path}')
            iter += 1

        avg_loss = total_loss / total_frames
        avg_psnr = total_psnr / total_frames
        avg_msssim = total_msssim / total_frames
        if type == 0:
            print(f'Середні втрати для набору даних {avg_loss}')
            print(f'Середній PSNR для набору даних {avg_psnr}')
            print(f'Середній mssim для набору даних {avg_msssim}')
            print(
                f'Epoch [{epoch + 1}/{num_epochs}], Втрата PSNR: {avg_psnr},
mssim: {avg_msssim}, Загальна втрата: {avg_loss}')
        else:
            print(f'Середні втрати для набору даних {avg_loss}')
            print(f'Середній PSNR для набору даних {avg_psnr}')
            print(f'Середній mssim для набору даних {avg_msssim}')

        return avg_psnr

# Завантаження моделі, якщо вона існує
model = QuatroUNET()
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
optimizer = Adam(model.parameters(), lr=0.001)
best_loss = float('inf')

model_path = 'FASTDVD2.pth'
```

```
if os.path.exists(model_path):
    model.load_state_dict(torch.load(model_path))
    best_model_weights = copy.deepcopy(model.state_dict())
    print(f'Завантажена раніше навчена модель best_model_weights :
{best_model_weights}')
else:
   print("Модель не знайдена, навчаємо з нуля")
   best_loss = float(1000)

# Завантаження та підготовка даних
datasets_path = 'Замінити на шлях до тренувального датасету'
validation_dataset_path = ['Замінити на шлях до валідаційного датасету']

# Навчання моделі
num_epochs = 200
loss_psnr = PSNRLoss().cuda()
optimizer = Adam(model.parameters(), lr=0.001)

# Перша перевірка валідації
print(f'Перша валідація')
total_val_loss = 0
model.eval()
best_loss = process(validation_dataset_path, model, loss_psnr, ms_ssim,
type=1,device=device)
best_model_weights = model.state_dict()
model.train()
print(f'Починаємо навчання')

count=0
# Навчання
```

```
for epoch in range(num_epochs):
    random_folders = get_random_videos_from_categories(datasets_path)
    datasets_paths = [video_path for video_path in random_folders if video_path is
not None]
    print(f'_____Навчання[{epoch +
1}/{num_epochs}]_____')
    training= process(datasets_paths, model, loss_psnr,
ms_ssim,type=0,device=device)
    print(f'Загальна середня втрата на всіх відео {training}')

    # Процес валідації
    model.eval()
    print(f'Валідація')
    total_val_loss=process(validation_dataset_path, model, loss_psnr,
ms_ssim,type=1,device=device)
    if - total_val_loss < best_loss:
        best_loss = - total_val_loss
        best_model_weights = model.state_dict()
        torch.save(model.state_dict(), model_path)
        print(f'Модель збережена з втратою на валідації {best_loss}')
    elif -total_val_loss > best_loss and count>1:
        # Відновлення ваг попередньої ітерації, якщо результат гірший
        model.load_state_dict(best_model_weights)
        print(f'Відновлені ваги попередньої ітерації. Втрата: {best_loss}')
    model_path_save = f"It{epoch}{model_path}"
    torch.save(model.state_dict(), model_path_save)
    model.train()
    count+=1
```

4. Файл work.py

```
import cv2
```

```
import numpy as np
import torch
import tkinter as tk
from tkinter import filedialog
from model import QuatroUNET
import time
from concurrent.futures import ThreadPoolExecutor

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model = QuatroUNET()
model.load_state_dict(torch.load('QuatroUNET.pth'))
if torch.cuda.is_available():
    model = model.to(device)
model.eval()

# Визначення шляху для збереження даних
save_path = 'Замінити шлях'

def cutter(frame):
    height, width, _ = frame.shape
    if width % 2 != 0:
        width -= 1
    if height % 2 != 0:
        height -= 1
    if width > 640 or height > 480:
        frame = cv2.resize(frame, (960, 540))
    return frame

def process_frame(frame):
```

```
    return ((torch.from_numpy(cutter(frame)).permute(2, 0, 1).float().unsqueeze(0))
/ 255).to(device)

is_paused = False
screenshot_count = 0
root = tk.Tk()
root.withdraw()
file_path = filedialog.askopenfilename()
cap = cv2.VideoCapture(file_path)
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

frames = []
for i in range(total_frames - 4):
    ret, frame = cap.read()
    frames.append(frame)

with ThreadPoolExecutor() as executor:
    processed_frames = list(executor.map(process_frame, frames))

for i in range(len(processed_frames) - 4):
    frame0 = processed_frames[i]
    frame1 = processed_frames[i+1]
    frame2 = processed_frames[i+2]
    frame3 = processed_frames[i+3]
    frame4 = processed_frames[i+4]

    with torch.no_grad():
        denoised_frame = (torch.clamp(model(frame0, frame1, frame2, frame3,
frame4), min=0.0, max=1.0))
    denoised_frame = denoised_frame * 255
```

```
    denoised_frame = denoised_frame.detach().squeeze(0).permute(1, 2,
0).cpu().numpy().astype(np.uint8)
    cv2.imshow('NoisyOriginalFrame', frames[i+4])
    cv2.imshow('DenoisedFrame', denoised_frame)

    key = cv2.waitKey(1)
    if key == ord('p'):
        is_paused = not is_paused
    elif key == ord('s') and not is_paused:
        screenshot_count += 1
        cv2.imwrite(f'screenshot_clear{screenshot_count}.png', frames[i+4])
        cv2.imwrite(f'screenshot_denoise{screenshot_count}.png', denoised_frame)
    elif key == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```