НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК _004.89_

До захисту допущено:
Завідувач кафедри
_____Михайло Новотарський
«___»_____2025 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Інженерія програмного забезпечення комп'ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Розробка нейромережевої моделі визначення оптичного потоку для впровадження на мікрокомп’ютері»

Виконав:
студент ІІ курсу, групи ІМ-42мп
Рингач Руслан Валерійович _____

Науковий керівник:
к.т.н., доц. Шимкович В.М. _____

Консультант з нормоконтролю:
проф. каф. ОТ, д.т.н., професор,
Жабін Валерій Іванович _____

Рецензент:
проф. каф. ІПІ, д.т.н., проф. Стеценко Інна Вячеславівна _____

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних посилань.
Студент Рингач Руслан Валерійович___

Київ — 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет/ННІ інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

До захисту допущено:
Завідувач кафедри
_____Михайло Новотарський
«___» _____2025 р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Рингачу Руслану Валерійовичу

1. Тема дисертації *«Розробка нейромережевої моделі визначення оптичного потоку для впровадження на мікрокомп'ютері»*, керівник дисертації *Шимкович Володимир Миколайович, к. т. н., доц.*, затверджені наказом по університету від «06» листопада 2025 р., № 4841-с.

2. Термін подання студентом дисертації 15.12.2025 р.

3. Вихідні дані до проєкту: технічна документація мікрокомп'ютера MaixCAM на базі процесора Sophon CV181x, специфікація інструментального засобу TPU-MLIR для компіляції моделей, стандартні набори даних для обчислення оптичного потоку (FlyingChairs, MPI Sintel, KITTI 2015), наукові публікації щодо сучасних архітектур нейронних мереж для оптичного потоку, методології квантизації та оптимізації моделей для мікрокомп'ютерів та мобільних пристроїв._____

4. Зміст пояснювальної записки: аналіз предметної області та розгляд аналогів, огляд технологій та підходів для рішення технічного завдання, їхні переваги та недоліки, розробка програмного забезпечення, тестування та аналіз результатів.

5. Перелік завдань, які потрібно розробити: огляд існуючих програмних рішень розгляд, аналіз принципів роботи схожих систем, проектування та розробка системи, тестування та аналіз результатів.

6. Консультанти розділів проєкту:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1.	*доц. Шимкович В. М.*		
2.	*доц. Шимкович В. М.*		
3.	*доц. Шимкович В. М.*		
4.	*доц. Шимкович В. М.*		

7. Дата видачі завдання 01.09.2025 року

КАЛЕНДАНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітки
1	*Затвердження теми роботи*	*01.09.2025 - 07.09.2025*	
2	*Вивчення та аналіз завдання*	*22.09.2025 - 28.09.2025*	
3	*Проектування архітектури системи*	*29.09.2025 - 05.10.2025*	
4	*Програмна реалізація системи*	*06.10.2025 - 15.10.2025*	
5	*Налагодження системи*	*15.10.2025 - 01.11.2025*	
6	*Оформлення магістерської записки*	*02.11.2025 -15.11.2025*	
7	*Захист*	*25.12.2025*	

Студент Руслан РИНГАЧ (підпис)

Керівник Владимир ШИМКОВИЧ (підпис)

РЕФЕРАТ

Розробка нейромережевої моделі визначення оптичного потоку для впровадження на мікрокомп’ютері: 124 с., 26 табл., 10 рис., 37 джерел.

ОПТИЧНИЙ ПОТІК, ГЛИБОКЕ НАВЧАННЯ, ПЕРИФЕРІЙНІ ОБЧИСЛЕННЯ, КВАНТИЗАЦІЯ INT8, НЕЙРОННИЙ ПРОЦЕСОР, МІКРОКОМП'ЮТЕР, ЕНЕРГОЕФЕКТИВНІСТЬ, АВТОНОМНІ СИСТЕМИ, КОМП'ЮТЕРНИЙ ЗІР, TPU.

Актуальність роботи зумовлена потребою у ефективних методах обчислення оптичного потоку для автономних робототехнічних систем на базі периферійних обчислювальних пристроїв з обмеженими ресурсами. Існуючі нейромережеві рішення (RAFT, FlowFormer, FastFlowNet) вимагають графічних процесорів з продуктивністю 100+ TFLOPS та споживанням енергії понад 100 Вт, що робить їх непридатними для мікрокомп'ютерів з бюджетом енергоспоживання менше 1 Вт та обсягом пам'яті 256 МБ. Розроблення спеціалізованої архітектури для платформ наднизького енергоспоживання є актуальною науково-технічною проблемою для розширення застосування комп'ютерного зору на нові класи автономних пристроїв.

Метою роботи є підвищення ефективності обчислення оптичного потоку у реальному часі на мікрокомп'ютерах з обмеженими обчислювальними ресурсами за рахунок розробки нової нейромережевої архітектури, оптимізованої під апаратні обмеження тензорних процесорів.

Для досягнення мети вирішено завдання: аналіз існуючих архітектур та виявлення компонентів, що визначають баланс точність-ефективність; дослідження можливостей Sophon CV181x TPU (1 TOPS INT8, 256 MB RAM); розробка легковагої архітектури EdgeFlowNet з гібридною кореляційною схемою та статичним обчислювальним графом; формалізація методології двофазного навчання з curriculum learning та стійкими функціями втрат; експериментальна валідація на датасетах Sintel та KITTI з вимірюванням продуктивності на реальній апаратурі MaixCAM.

Об'єктом дослідження є процеси обчислення оптичного потоку з використанням методів глибокого навчання на периферійних пристроях з жорсткими обмеженнями продуктивності нейропроцесорів, доступної пам'яті та енергетичного бюджету.

Предметом дослідження є архітектура легковагової згорткової нейронної мережі EdgeFlowNet для обчислення щільного оптичного потоку, методи її оптимізації під апаратні обмеження Sophon CV181x TPU та техніки навчання для забезпечення балансу між точністю та ефективністю виведення.

Методи дослідження: аналіз предметної області для систематизації проблем існуючих підходів; математичне моделювання нейромережевих архітектур з використанням апарату згорткових мереж та теорії оптимізації; експериментальні дослідження на стандартних тестових наборах Sintel та KITTI для валідації точності; профілювання продуктивності на апаратурі MaixCAM для вимірювання швидкості інференсу та споживання пам'яті; порівняльний аналіз з існуючими легковагими методами FastFlowNet та RAFT-Small.

Наукова новизна роботи полягає у розробленні гібридної кореляційної архітектури, що поєднує глобальну подвійну одновимірну кореляцію на низьких роздільностях з локальною двовимірною кореляцією на високих, забезпечуючи економію пам'яті у 65 разів при збереженні діапазону пошуку до 96 пікселів; запропонованій архітектурі ітераційного уточнення без рекурентних блоків з фіксованою кількістю ітерацій та статичними розмірами тензорів для ефективної компіляції на TPU; розробленій методології архітектурної оптимізації під обмеження компілятора TPU-MLIR, що включає елімінацію операцій grid_sample та використання навчуваних обхідних з'єднань; експериментальному обґрунтуванні ефективності Post-Training Quantization з деградацією точності лише 7-8% при INT8 квантизації без Quantization-Aware Training.

Практичне значення результатів: розроблено функціональну систему для мікрокомп'ютера Sipeed MaixCAM (Sophon CV181x, 256MB RAM, ~$30) з продуктивністю 25,7 FPS на роздільності 320×256, точністю EPE 1,39 пікселя на

Sintel Clean та 4,67 пікселів на KITTI-15, розміром моделі 0,50M параметрів, енергоспоживанням менше 1 Вт. Безпосередніми застосуваннями є візуальна одометрія для мікродронів у середовищах без GPS, уникнення перешкод для мобільних роботів, відстеження руху для IoT камер. Методологія оптимізації узагальнюється на інші задачі комп'ютерного зору (виявлення об'єктів, семантична сегментація) та периферійні платформи (Google Coral, Intel Movidius).

ABSTRACT

Lightweight Neural Network Architecture for Optical Flow Computation on Edge Devices: 124 p., 26 tab., 10 fig., 37 sources.

OPTICAL FLOW, DEEP LEARNING, EDGE COMPUTING, INT8 QUANTIZATION, NEURAL PROCESSOR, MICROCOMPUTER, ENERGY EFFICIENCY, AUTONOMOUS SYSTEMS, COMPUTER VISION, TPU.

The relevance of this work stems from the need for efficient optical flow computation methods for autonomous robotic systems based on edge computing devices with limited resources. Existing neural network solutions (RAFT, FlowFormer, FastFlowNet) require GPUs with 100+ TFLOPS performance and power consumption exceeding 100W, making them unsuitable for microcomputers with power budgets below 1W and 256MB RAM. Developing a specialized architecture for ultra-low-power platforms represents an urgent scientific and technical challenge for expanding computer vision applications to new classes of autonomous devices.

The objective is to improve the efficiency of real-time optical flow computation on microcomputers with limited computational resources by developing a novel neural network architecture optimized for tensor processor hardware constraints.

To achieve this goal, the following tasks were addressed: analysis of existing architectures to identify components determining accuracy-efficiency balance; investigation of Sophon CV181x TPU capabilities (1 TOPS INT8, 256MB RAM); development of lightweight EdgeFlowNet architecture with hybrid correlation scheme and static computational graph; formalization of two-phase training methodology with curriculum learning and robust loss functions; experimental validation on Sintel and KITTI datasets with performance measurements on real MaixCAM hardware.

The research object encompasses optical flow computation processes using deep learning methods on edge devices with strict constraints on neural processor performance, available memory, and power budget.

The research subject is the architecture of lightweight convolutional neural network EdgeFlowNet for dense optical flow computation, methods for its optimization under Sophon CV181x TPU hardware constraints, and training techniques to balance accuracy and inference efficiency.

Research methods include: domain analysis to systematize existing approach limitations; mathematical modeling of neural network architectures using convolutional network theory and optimization; experimental studies on standard Sintel and KITTI benchmarks for accuracy validation; performance profiling on MaixCAM hardware to measure inference speed and memory consumption; comparative analysis with existing lightweight methods FastFlowNet and RAFT-Small.

Scientific novelty: developed hybrid correlation architecture combining global dual 1D correlation at low resolutions with local 2D correlation at high resolutions, achieving 65× memory savings while maintaining search range up to 96 pixels; proposed iterative refinement architecture without recurrent blocks featuring fixed iteration count and static tensor sizes for efficient TPU compilation; developed architectural optimization methodology for TPU-MLIR compiler constraints including grid_sample operation elimination and learnable skip connections; experimentally validated Post-Training Quantization efficiency with only 7-8% accuracy degradation under INT8 quantization without Quantization-Aware Training.

Practical significance: developed functional system for Sipeed MaixCAM microcomputer (Sophon CV181x, 256MB RAM, ~$30) achieving 25.7 FPS at 320×256 resolution, EPE 1.39 pixels on Sintel Clean and 4.67 pixels on KITTI-15, model size 0.50M parameters, power consumption below 1W. Direct applications include visual odometry for micro-drones in GPS-denied environments, obstacle avoidance for mobile robots, motion tracking for IoT cameras. Optimization

methodology generalizes to other computer vision tasks (object detection, semantic segmentation) and edge platforms (Google Coral, Intel Movidius).

ЗМІСТ

ВСТУП

Швидкий розвиток робототехніки та автономних систем зумовлює необхідність створення ефективних методів візуального сприйняття для периферійних обчислювальних пристроїв. Оптичний потік — векторне поле швидкостей руху точок зображення між послідовними кадрами — є фундаментальною задачею комп'ютерного зору, яка знаходить застосування у візуальній одометрії [17], уникненні перешкод, розпізнаванні жестів та автономному керуванні транспортними засобами [1; 2].

Традиційні класичні методи обчислення оптичного потоку, такі як Lucas-Kanade [8] та Horn-Schunck [7], хоча й заклали математичну основу задачі, демонструють обмежену точність та не можуть ефективно обробляти складні сценарії з великими зміщеннями, оклюзіями та динамічними змінами освітлення. Впровадження методів глибокого навчання суттєво покращило якість обчислення оптичного потоку — сучасні нейромережеві методи, такі як RAFT [3] та FlowFormer [4], досягають середньої похибки кінцевої точки (EPE) 1–2 пікселі на стандартних тестових наборах даних, що на 60–70% краще порівняно з класичними підходами.

Однак існуючі нейромережеві архітектури розроблялися переважно для потужних настільних графічних процесорів (GPU) з обчислювальною потужністю понад 100 TFLOPS та обсягом пам'яті 24–80 ГБ. Їхнє впровадження на мікрокомп'ютерах з обмеженими ресурсами стикається із значними обмеженнями. Периферійні пристрої, такі як MaixCAM на базі Sophon CV181x [13], продуктивність нейронного процесора (NPU) становить лише 1 TOPS (INT8), обсяг оперативної пам'яті (RAM) обмежено 256 МБ, енергетичний бюджет становить менше 1 Вт для забезпечення автономної роботи від батареї.

Це створює різницю у 20–200 разів за обчислювальною потужністю та у 100–300 разів за доступною пам'яттю порівняно з серверними рішеннями.

Наявні легковагі архітектури, такі як FastFlowNet [5] (1,37 мільйона параметрів, 12,2 GFLOPS) та RAFT-Small (0,99 мільйона параметрів, 47,66

GFLOPS), хоча й зменшують обчислювальну складність порівняно з оригінальними версіями, все ще залишаються занадто ресурсоємними для мікрокомп’ютерів наднизького енергоспоживання. Крім того, ці архітектури проектувалися для графічних процесорів NVIDIA та використовують операції (деформація ознак через grid_sample, рекурентні блоки GRU, динамічні розміри тензорів), які погано підтримуються або є неефективними на спеціалізованих NPU прискорювачах (напр., Sophon CV181x [14]).

Таким чином, розроблення спеціалізованої архітектури нейронної мережі для обчислення оптичного потоку, оптимізованої під жорсткі обмеження ресурсів мікрокомп’ютерів з нейронним процесором Sophon CV181x, становить актуальну науково-технічну проблему, вирішення якої дозволить розширити застосування робототехнічних систем комп’ютерного зору на нові класи автономних пристроїв з низьким енергоспоживанням та собівартістю.

Метою роботи є підвищення ефективності обчислення оптичного потоку у реальному часі на мікрокомп’ютерах з обмеженими обчислювальними ресурсами та енергетичним бюджетом за рахунок розробки нової нейромережевої моделі.

Для досягнення поставленої мети визначено такі завдання:

1. проаналізувати існуючі архітектури нейронних мереж для обчислення оптичного потоку та виявити ключові компоненти, що визначають баланс між точністю та обчислювальною ефективністю;
2. дослідити апаратні можливості та обмеження мікрокомп’ютерів на базі Sophon CV181x (MaixCAM), включаючи продуктивність нейронного процесора, підтримувані операції, особливості компілятора TPU-MLIR [14] та оптимізації квантизації INT8 [23; 24];
3. розробити легковагу архітектуру EdgeFlowNet з використанням групових роздільних згорток [21], гібридної кореляційної схеми та спрощеного модуля уточнення, оптимізовану для статичних розмірів тензорів та інференсу INT8;

4. розробити методологію двофазного навчання з поступовим ускладненням [15], інтенсивною аугментацією даних та стійкими функціями втрат [25] для забезпечення високої точності компактної моделі;
5. провести експериментальне дослідження точності на стандартних тестових наборах даних Sintel [18] та KITTI [19], продуктивності на апаратурі MaixCAM, та порівняти результати з існуючими легкими архітектурами.

Об’єктом дослідження є процеси обчислення оптичного потоку з використанням методів глибокого навчання на периферійних обчислювальних пристроях з жорсткими обмеженнями ресурсів, включаючи продуктивність нейропроцесорних прискорювачів, доступну оперативну пам’ять, енергетичний бюджет та підтримувані обчислювальні операції.

Предметом дослідження є архітектура легковагої згорткової нейронної мережі EdgeFlowNet для обчислення щільного оптичного потоку, методи її оптимізації під апаратні обмеження мікрокомп’ютерів Sophon CV181x, та техніки навчання для забезпечення прийнятного балансу між точністю, швидкістю інференсу та енергоефективністю.

Наукові та практичні результати.

Розроблено гібридну кореляційну архітектуру для обчислення оптичного потоку на надмалопотужних пристроях, що поєднує глобальну подвійну одновимірну кореляцію (Dual 1D Global Correlation) на низьких роздільностях з локальною двовимірною кореляцією на високих. Запропонований підхід забезпечує економію пам’яті у 65 разів порівняно з повнопарною 4D кореляцією при збереженні діапазону пошуку до 96 пікселів на роздільності 320×256, що критично для роботи на пристроях з 256MB RAM.

Запропоновано архітектуру ітераційного уточнення без рекурентних блоків з фіксованою кількістю ітерацій та статичними розмірами тензорів. На відміну від RAFT та IRR-PWC, які використовують ConvGRU з динамічним hidden state, розроблена архітектура забезпечує повністю статичний

обчислювальний граф для ефективної компіляції на тензорні процесори з обмеженою підтримкою динамічних операцій.

Розроблено методологію архітектурної оптимізації під обмеження компілятора TPU-MLIR, що включає: елімінацію операцій grid_sample через явну кореляцію на основі циклів, використання навчуваних обхідних з'єднань замість concatenation для економії пам'яті, обмеження глибини мереж для уникнення stack overflow при компіляції. Методологія забезпечує успішну компіляцію моделі на Sophon CV181x TPU з продуктивністю 1 TOPS (INT8).

Експериментально обґрунтовано ефективність Post-Training Quantization (PTQ) для архітектур оптичного потоку на платформах з екстремальними обмеженнями. Показано, що використання LeakyReLU замість ReLU, Batch Normalization замість Instance Normalization, та мілкі декодувальні мережі забезпечує деградацію точності лише 7-8% при INT8 квантизації без Quantization-Aware Training, що значно краще типових 15-25% деградації для стандартних архітектур.

Розроблено функціональну систему обчислення оптичного потоку на мікрокомп'ютері Sipeed MaixCAM (Sophon CV181x TPU, 256MB RAM, вартість ~$30), що забезпечує: - Продуктивність 25,7 FPS на роздільності 320×256 у режимі INT8; - Точність EPE 1,39 пікселя на Sintel Clean та 4,67 пікселя на KITTI-15 (оцінка на downsampled версіях датасетів); - Модель з 0,50M параметрів та 1,48 GFLOPS обчислювальної складності; - Споживання пам'яті 43,9 МБ (17,1% від доступних 256 МБ); - Енергоспоживання <1W, що дозволяє автономну роботу від батареї 6-8 годин.

Безпосередніми застосуваннями є візуальна одометрія та уникнення перешкод для мікродронів у середовищах без GPS, відстеження руху для носимих пристроїв, розпізнавання жестів для IoT камер. Низька вартість платформи (~$30) дозволяє масове впровадження у освітню робототехніку та прототипування стартапів.

Методологія оптимізації може бути узагальнена на інші задачі комп'ютерного зору (object detection, semantic segmentation, depth estimation) та

платформи з подібними обмеженнями (Google Coral Edge TPU, Intel Movidius, ARM Ethos-N77).

Результати демонструють можливість досягнення точності конкурентоспроможної з моделями у 2,7-10× більшими при радикальному зменшенні обчислювальних вимог, що розширює межі застосування deep learning на edge devices та створює передумови для нового покоління доступних автономних систем.

РОЗДІЛ 1
МЕТОДИ ОБЧИСЛЕННЯ ОПТИЧНОГО ПОТОКУ

1.1. Теоретичні основи предметної області

1.1.1. Концепція оптичного потоку в комп'ютерному зорі

Оптичний потік є фундаментальною задачею комп'ютерного зору, яка полягає у визначенні векторного поля швидкостей руху точок зображення між послідовними кадрами відеопослідовності. По суті, оптичний потік описує видиме переміщення кожного пікселя між двома моментами часу у вигляді двовимірного векторного поля.

Фізична інтерпретація полягає у тому, що коли камера або об'єкти в сцені рухаються, пікселі на зображенні змінюють своє положення. Оптичний потік фіксує цей рух у вигляді вектора зміщення для кожного пікселя. При зміщенні об'єкта вправо на 5 пікселів та вниз на 3 пікселі між двома кадрами, відповідний вектор оптичного потоку буде (5, 3).

Класична теорія оптичного потоку базується на припущенні сталості яскравості — інтенсивність певної точки на поверхні об'єкта залишається незмінною при русі, тобто яскравість пікселя зберігається між кадрами навіть при зміні його положення. Математично це записується як:

$$I(x, y, t) = I(x + u, y + v, t + \Delta t) \tag{1.1}$$

де I — інтенсивність пікселя в точці (x, y) у момент часу t. u, v — компоненти вектора оптичного потоку.

Використовуючи наближення малих зміщень, отримуємо базове рівняння градієнтного обмеження:

$$I_x \cdot u + I_y \cdot v + I_t = 0 \tag{1.2}$$

де I_x, I_y — просторові градієнти, I_t — часова похідна.

Це рівняння має одну формулу з двома невідомими u, v для кожного пікселя, що створює апертурну проблему — неможливість однозначно визначити рух лише з локальної інформації.

Розв’язання апертурної проблеми здійснюється через додаткові припущення. Метод Horn-Schunck додає припущення про плавність — оптичний потік змінюється поступово в просторі без різких стрибків [7]. Метод Lucas-Kanade припускає постійність потоку в межах невеликої локальної області [8].

1.1.2. Класичні підходи до обчислення оптичного потоку

Історично методи обчислення оптичного потоку можна класифікувати на кілька основних категорій.

Методи на основі градієнтів використовують просторові та часові похідні інтенсивності зображення. Найвідоміший представник — метод Lucas-Kanade — розв’язує систему рівнянь для невеликого вікна навколо кожного пікселя [8]. Основна перевага: швидкість обчислень та простота реалізації. Недоліки: обмеженість малими зміщеннями та чутливість до шуму.

Методи на основі енергії формулюють задачу як проблему мінімізації енергетичного функціоналу. Класичний метод Horn-Schunck мінімізує комбінацію члена відповідності даних та члена гладкості [7]. Перевага: гладкі поля потоку без розривів. Недоліки: розмиття меж об’єктів та високі обчислювальні витрати.

Методи на основі відповідності блоків працюють за простим принципом - для кожного блоку пікселів у першому зображенні шукається найбільш схожий блок у другому зображенні. Міра схожості обчислюється через суму абсолютних різниць, суму квадратів різниць або нормалізовану крос-кореляцію. Перевагами є простота та стійкість до шуму. Недоліками є дискретність результатів та втрата точності на межах об’єктів.

Ієрархічні методи використовують пірамідальне представлення зображень для обробки великих зміщень. На найнижчій роздільності обчислюється грубий оптичний потік, який потім масштабується та уточнюється на кожному вищому рівні [9]. Ця ідея стала основою для більшості сучасних нейромережевих архітектур.

1.1.3. Глибоке навчання в задачах оптичного потоку

Поява глибокого навчання суттєво змінила підхід до обчислення оптичного потоку. Перші успішні спроби застосування згорткових нейронних мереж продемонстрували моделі FlowNet та FlowNet 2.0, які навчалися наскрізно на синтетичних даних [1; 2].

Принципова зміна підходу полягає у тому, що замість ручного проектування алгоритмів нейронна мережа автоматично навчається виділяти релевантні ознаки та встановлювати відповідності між кадрами на основі великої кількості прикладів.

Нейромережеві методи мають характерні архітектурні особливості. Архітектура кодувальник-декодувальник стала основою більшості сучасних підходів. Кодувальник послідовно зменшує просторову роздільність вхідних зображень, витягуючи багаторівневі ознаки — від простих країв та текстур на нижніх рівнях до семантичної інформації про форми об’єктів на вищих рівнях. Декодувальник відновлює поле оптичного потоку до вихідної роздільності, поступово збільшуючи розмір та додаючи деталі на кожному кроці.

Кореляційні шари відіграють ключову роль у встановленні відповідностей між ознаками двох зображень. Ці спеціалізовані шари обчислюють міру схожості між кожною позицією у першому зображенні та можливими відповідниками у другому, створюючи тривимірний тензор кореляції. Основні типи кореляції:

All-pairs кореляція обчислює схожість кожної позиції з усіма можливими відповідниками, створюючи повний простір пошуку. Для зображення 64×64 отримуємо тензор кореляції розміром 64×64×4096.

Локальна кореляція обмежує пошук невеликою областю навколо кожної позиції, зменшуючи обчислювальну складність. Для вікна радіусом 4 пікселі отримуємо тензор лише 64×64×81.

Ітераційне уточнення дозволяє поступове покращення оцінки оптичного потоку. Замість одноразового обчислення кінцевого результату, мережа робить початкову грубу оцінку, яка потім проходить через серію блоків уточнення. Кожна ітерація аналізує поточну оцінку потоку, обчислює залишкову помилку та додає корекцію.

1.1.4. Піраміда ознак та багатомасштабна обробка

Піраміда ознак є критично важливим компонентом сучасних архітектур оптичного потоку, що дозволяє обробляти рухи різних масштабів — від малих зміщень окремих деталей до великих рухів цілих об’єктів або камери.

Концепція полягає у тому, що вхідні зображення обробляються на кількох рівнях роздільності одночасно. Типова піраміда має 4-6 рівнів з коефіцієнтом зменшення 2× між рівнями.

Багатомасштабна обробка має такі переваги. Нижні рівні (низька роздільність 16×20 або 32×40) захоплюють великі рухи та глобальний контекст сцени без надмірних обчислювальних витрат. Вищі рівні (висока роздільність 128×160 або 256×320) додають точні деталі руху окремих об’єктів та країв. Ієрархічна структура дозволяє обробляти зміщення у десятки або сотні пікселів, які були б неможливі для одномасштабних методів.

Модель PWC-Net (Pyramid, Warping, and Cost Volume Network) запропонувала класичну схему багатомасштабної обробки [10]: на кожному рівні піраміди обчислюється тензор вартості через кореляційний шар,

оцінюється оптичний потік, результат масштабується та використовується для деформації ознак наступного рівня. Цей підхід став стандартом у галузі.

1.1.5. Архітектура RAFT та ітераційне уточнення

Модель RAFT (Recurrent All-Pairs Field Transforms) представила радикально новий підхід до обчислення оптичного потоку [3]. Замість традиційної пірамідальної схеми RAFT використовує рекурентну архітектуру з повторюваним уточненням на одному масштабі.

RAFT складається з таких ключових компонентів. Екстрактор ознак витягує багаторівневі ознаки з обох вхідних зображень. Архітектура використовує залишкові блоки для створення щільного представлення. All-pairs кореляція обчислюється один раз для всіх можливих відповідностей між ознаками двох зображень, створюючи чотиривимірний тензор кореляції. Цей тензор зберігається в пам’яті та використовується на всіх ітераціях уточнення. Блок GRU виконує ітераційне оновлення поля потоку. На кожній ітерації блок отримує поточну оцінку потоку, витягує відповідні кореляційні ознаки через оператор пошуку, та обчислює оновлення.

Процес уточнення полягає у тому, що RAFT виконує 12-32 ітерації оновлення на етапі навчання та 12-24 ітерації під час інференсу. Кожна ітерація покращує точність на долі пікселя, дозволяючи досягти субпіксельної точності.

Перевагою RAFT є те, що архітектура демонструє видатну точність на стандартних бенчмарках — покращення на 30-40% порівняно з попередніми методами. RAFT добре узагальнюється на нові домени завдяки простій та елегантній структурі. Ітераційний характер дозволяє компроміс між точністю та швидкістю через регулювання кількості ітерацій.

Обмеженнями для периферійних пристроїв є те, що all-pairs кореляція вимагає великих обсягів пам’яті. Рекурентний характер GRU ускладнює оптимізацію та розгортання на апаратних прискорювачах. Повторювані оператори пошуку створюють додаткові накладні витрати.

1.1.6. Трансформери в оптичному потоці

У роботі FlowFormer вперше успішно адаптовано архітектуру трансформера для задач оптичного потоку [4]. Основна ідея полягає у використанні механізму уваги для встановлення відповідностей між ознаками двох зображень.

Ключовими інноваціями FlowFormer є Cost Volume Transformer — спеціалізований трансформер для обробки тензорів кореляції. Замість традиційної обробки кореляційного тензора через згортки, FlowFormer токенізує його та застосовує механізм само-уваги. Alternate-Group Transformer (AGT) є ефективною архітектурою для зменшення обчислювальної складності механізму уваги. Рекурентний декодувальник трансформера — це рекурентний декодер з динамічними позиційними запитами для ітераційного уточнення потоку.

Результати показали, що FlowFormer досягає нових рекордів точності на бенчмарках Sintel та KITTI, покращуючи показники RAFT на 15-20%. Особливо вражаючі результати на складних сценах з оклюзіями та швидкими рухами.

Обмеженнями є те, що висока обчислювальна складність (~150+ GFLOPS) робить FlowFormer непридатним для мобільних пристроїв. Механізм уваги вимагає значних обсягів пам’яті та складний для оптимізації на спеціалізованих прискорювачах.

1.1.7. Надлегкі архітектури для периферійних пристроїв

З розвитком автономних мобільних систем та IoT пристроїв виникла потреба у надлегких моделях оптичного потоку, здатних працювати на мікрокомп’ютерах з екстремально обмеженими ресурсами.

NanoFlowNet (TU Delft, 2022) є першою моделлю, що досягла інференсу оптичного потоку в реальному часі на мікроконтролері GAP8 (8-ядерний мікроконтролер, 250 МГц) [26]. Архітектура базується на мережі семантичної

сегментації STDC-Seg та використовує модифіковані блоки STDC з роздільними згортками по глибині та оптимізованим розташуванням операцій для зменшення MAC операцій на 50% на першому етапі. Тренувальна техніка з використанням розмітки меж руху покращує деталізацію країв об'єктів без впливу на латентність інференсу через використання допоміжної голови тільки під час навчання. Модель використовує низьку роздільність 112×160 пікселів (qqVGA) та вхід в градаціях сірого для економії 66% пам'яті.

Модель містить лише 0,17M параметрів (приблизно 1,0 GFLOPS) та досягає 5,57 кадрів за секунду на GAP8 при споживанні менше 1 Вт. Точність становить EPE 7,12 на Sintel Clean (навчальна вибірка). NanoFlowNet було успішно продемонстровано на нанодроні Bitcraze Crazyflie (34 грами) для уникнення перешкод у реальних умовах.

Обмеженнями є дуже низька роздільність (112×160) та відносно висока похибка (EPE 7,12), що обмежує застосування задачами грубого виявлення руху. Архітектура оптимізована для IoT мікроконтролерів, а не для спеціалізованих TPU/NPU прискорювачів.

CompactFlowNet (Picsart AI, 2024) є найновішою моделлю, що досягла інференсу в реальному часі на мобільних пристроях (iPhone 8+) [27]. Архітектура базується на PWC-Net з критичними модифікаціями. Заміна послідовного з'єднання на послідовні з'єднання у модулі оцінки потоку зменшила латентність зі 131 мілісекунди до приблизно 40 мілісекунд на iPhone 8 (512×512). Використовується сучасний мобільний екстрактор ознак MobileNetV3 (3,12M параметрів) замість оригінального енкодера PWC-Net. Навчання з учителем (важка PWC-Net + MobileNetV3) дає 8% покращення EPE порівняно з прямим навчанням. Роздільні згортки по глибині використовуються у модулі оцінки потоку та уточнення для зменшення обчислювальної складності. Покращена процедура навчання включає використання датасету Autoflow, циклічну швидкість навчання OneCycle та обмеження градієнтів.

Модель містить менше 5M параметрів та досягає 25 кадрів за секунду на iPhone 8 (512×512), 77 кадрів за секунду на iPhone 14 Pro при споживанні

пам’яті 118 МБ. Точність становить EPE 4,43 на Sintel Clean тестовій вибірці, 5,55 на Sintel Final тестовій вибірці. CompactFlowNet підтримує роздільності до 1920×1080 (FullHD) з 10 кадрами за секунду на iPhone 14 Pro.

Перевагами є готова до промислового використання реалізація з TFLite для iOS/Android, найнижче споживання пам’яті серед легковагих моделей (118 МБ для 512×512), детальний бенчмарк на 4 мобільних пристроях. Обмеженнями є значно більший розмір (приблизно 5M параметрів) порівняно з наднадлегкими моделями, відносно низька точність (EPE 4,43) порівняно з моделями середнього розміру.

Таблиця 1.1

Порівняння надлегкі архітектур оптичного потоку

Модель	**Параметри**	**Платформа**	**FPS (edge)**	**Роздільність**	**EPE (Sintel Clean)**	**Рік**
NanoFlowNet	0,17M	GAP8 MCU	5,6	112×160	7,12 (train)	2022
CompactFlowNet	<5M	iPhone 8+	25	512×512	4,43 (test)	2024
EdgeFlowNet (наш)	0,50M	CV181x TPU	16,9	256×320	1,39 (train)	2024

Аналіз показує, що існує компроміс між розміром моделі, точністю та цільовою платформою. NanoFlowNet оптимізований для IoT MCU з мінімальним розміром, CompactFlowNet — для мобільних GPU з балансом швидкості та точності. Запропонована робота EdgeFlowNet займає проміжну позицію, досягаючи найкращої точності серед надлегкі моделей при роздільності 256×320 на спеціалізованому TPU прискорювачі.

1.2. Порівняльний аналіз існуючих архітектур

1.2.1. Еволюція точності та ефективності

Розвиток методів оптичного потоку демонструє постійний компроміс між точністю та обчислювальною ефективністю. FlowNet 2.0 (~162M параметрів) показав можливість наскрізного навчання, але мав великий розмір. PWC-Net (~8.7M параметрів) значно зменшив кількість параметрів через пірамідальну архітектуру. RAFT (~5.3M параметрів) досягнув прориву в точності через ітераційне уточнення. FlowFormer (~15.1M параметрів) встановив нові рекорди точності через трансформери, але повернувся до великих моделей.

Паралельно розвивалася гілка надлегких моделей для edge пристроїв: NanoFlowNet (2022) [26] довів можливість інференсу на IoT MCU з 0,17M параметрів, CompactFlowNet (2024) [27] досяг real-time на мобільних телефонах з production-ready реалізацією. Запропонована робота EdgeFlowNet синтезує досягнення обох напрямків, поєднуючи компактність надлегких моделей (0,50M параметрів) з точністю, порівнянною із значно більшими архітектурами.

Таблиця 1.2

Детальне порівняння архітектур оптичного потоку

Модель	**Params**	**FLOPS**	**Platform**	**EPE (Sintel Clean)**	**FPS**	**Ключова інновація**
FlowNet2.0 [1]	162M	150G	GPU	3.96 (test)	8	Каскадна архітектура
PWC-Net [10]	8.7M	90G	GPU	2.15 (train)	35	Pyramid + Cost Volume
RAFT [3]	5.3M	211G	GPU	1.43 (test)	10	All-pairs correlation + GRU
FlowFormer [4]	15.1M	150G	GPU	1.01 (test)	5	Transformer attention
FastFlowNet [5]	1.37M	12.2G	GPU	2.90 (train)	58	Head Pooling Pyramid
NanoFlowNet [26]	0.17M	~1.0G	GAP8 MCU	7.12 (train)	5.6	Motion boundary guidance
CompactFlowNet [27]	<5M	~15G	iPhone 8+	4.43 (test)	25	Sequential estimator + Distillation
EdgeFlowNet	0.50M	1.48G	CV181x TPU	1.39 (train)	16.9	Dual 1D Correlation

Аналіз таблиці демонструє, що EdgeFlowNet досягає найкращого балансу між точністю та ефективністю для надлегкої категорії:

— 5× точніший за NanoFlowNet при 3× більшій кількості параметрів;

— 3× точніший за CompactFlowNet при 10× меншій кількості параметрів;

— лише на 0,04 пікселя гірший за RAFT при 10× меншій кількості параметрів та 142× меншій обчислювальній складності.

1.2.2. Апаратні платформи для периферійних пристроїв

Таблиця 1.3

Порівняння апаратних платформ для реалізації оптичного потоку

Платформа	**Продуктивність (INT8)**	**Пам'ять (МБ)**	**Енергоспоживання (Вт)**	**Вартість (USD)**	**Особливості**
MaixCAM (CV181x/SG2002)	1,0 TOPS	256	~0,5	~30	Дуже низька вартість, TPU спеціалізація
NVIDIA Jetson Orin Nano	40 TOPS	8192	7-25	~250	Висока продуктивність, ARM Cortex-A78AE
Raspberry Pi 4 + Coral TPU	4 TOPS	8192	~6	~135	Модульність, Edge TPU прискорювач
Coral Dev Board	4 TOPS	4096	2-4	~150	Інтегроване рішення, Edge TPU

1.2.3. Аналіз обмежень існуючих методів

Обчислювальна складність сучасних архітектур становить від 12 до 211 GFLOPS, що значно перевищує можливості мікрокомп'ютерів (1-4 TOPS INT8). PWC-Net потребує ~90.8 GFLOPS, RAFT — ~211 GFLOPS, що робить їх непридатними для виконання у реальному часі на edge пристроях.

Споживання пам'яті: All-pairs кореляція у RAFT створює тензори розміром до 100+ ГБ для високих роздільностей. Навіть з оптимізацією споживання пам'яті залишається на рівні 4-8 ГБ, що неприйнятно для пристроїв з 256 МБ RAM.

Сумісність з TPU/NPU: більшість архітектур використовують операції, що погано підтримуються спеціалізованими прискорювачами: динамічні операції

reshape, деформацію через grid_sample, рекурентні блоки GRU/LSTM. Це ускладнює або унеможливлює ефективну компіляцію для TPU.

1.3. Методи оптимізації для периферійних пристроїв

1.3.1. Квантизація моделей

INT8 квантизація зменшує розмір моделі у 4 рази та прискорює інференс у 2-4 рази на спеціалізованих прискорювачах.

Post-training quantization — квантизація після навчання з калібруванням на невеликому наборі даних.

Quantization-aware training (QAT) — навчання з урахуванням квантизації, що мінімізує втрату точності.

Викликами квантизації для optical flow є те, що активації оптичного потоку мають широкий динамічний діапазон (від -100 до +100 пікселів), що ускладнює квантизацію. Кореляційні тензори мають нерівномірний розподіл значень з великою кількістю нулів.

1.3.2. Архітектурні оптимізації

Групові роздільні згортки (Depthwise Separable Convolutions) зменшують обчислювальну складність у 7-8 разів порівняно зі стандартними згортками:

$$\text{Зменшення } FLOPS = \frac{C_{out}}{K^2} \text{ для } K \times K \text{ ядра} \quad (1.3)$$

Дистиляція знань дозволяє передати знання від великої моделі-вчителя до компактної моделі-учня, покращуючи точність на 3-5% без збільшення розміру.

Обрізання видаляє найменш важливі з’єднання або канали, зменшуючи розмір моделі на 50-90% з мінімальною втратою точності.

1.3.3. Функції втрат та методи навчання

Для забезпечення ефективного навчання моделі на об'єктах із широким діапазоном розмірів зміщень та покращення збіжності градієнтного спуску, застосовується багатомасштабний функціонал втрат L. Він являє собою вагове сумування локальних втрат L_i, обчислених на різних рівнях роздільності (масштабу) піраміди оптичного потоку:

$$L = \sum_i w_i \cdot L_i \tag{1.4}$$

де w_i — ваги для масштабів $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ (наприклад, $1.0, 0.5, 0.25, 0.125$).

З метою підвищення стійкості процесу навчання до аномальних значень, які можуть виникати в областях оклюзій або різких змін освітлення, замість стандартної квадратичної норми використовується функціонал втрат Шарбоньє (Charbonnier loss). Цей функціонал ефективно обмежує вплив великих помилок:

$$L_{char} = \sqrt{\left(flow_{pred} - flow_{gt}\right)^2 + \epsilon^2} \tag{1.5}$$

де $flow_{pred}$ — прогнозований оптичний потік, $flow_{gt}$ — еталонний потік, а $\epsilon = 0.001$ — що забезпечує чисельну стабільність при диференціюванні.

Для забезпечення просторової когерентності та гладкості обчисленого векторного поля оптичного потоку вводиться регуляризаційний член гладкості L_{smooth}. Цей член є адаптивним, тобто він згладжує потік усередині однорідних областей, але мінімізує згладжування на межах об'єктів (краях), де градієнт зображення великий:

$$L_{smooth} = \sum_{x,y} |\nabla u| e^{-\alpha|\nabla I|} + |\nabla v| e^{-\alpha|\nabla I|} \tag{1.6}$$

де ∇I — градієнт зображення, α — параметр чутливості.

1.4. Аналіз застосувань оптичного потоку у робототехніці

Оптичний потік є фундаментальним інструментом візуального сприйняття, що має широке застосування в автономних та робототехнічних системах [1; 2]. Він є ключовим компонентом для Візуальної Одометрії (VO) та Одночасної Локалізації та Картографування (SLAM), забезпечуючи первинну інформацію для оцінки траєкторії руху камери, локалізації об'єктів та підвищення точності оцінки глибини сцени [17]. Критичними вимогами для систем VO/SLAM є висока точність (EPE < 1 піксель) та функціонування у реальному часі (10–30 кадрів/с).

Для мобільних роботів та дронів оптичний потік критично важливий для уникнення перешкод і планування траєкторій, оскільки він дозволяє оцінювати час до зіткнення (за дивергентним потоком) та визначати напрямок руху камери через Фокус Розширення (FoE). У цьому застосуванні необхідна висока продуктивність (30+ кадрів/с) та низька затримка.

У системах взаємодії людина-робот (HRI) оптичний потік використовується для розпізнавання жестів шляхом траєкторного аналізу та детекції руху, що допомагає відокремити рухомі частини тіла від фону. Нарешті, у системах автономного водіння він застосовується для компенсації власного руху транспортного засобу, розділяючи рух автомобіля та зовнішніх об'єктів, а також для оцінки відносної дистанції до об’єктів, що є необхідним для надійного сприйняття дорожнього середовища. В усіх цих доменах ключовими вимогами є висока продуктивність (30+ кадрів/с), мінімальна латентність та надійність в різних експлуатаційних та погодних умовах.

Таблиця 1.4

Вимоги до оптичного потоку для різних застосувань

Застосування	**Точність (EPE)**	**Частота (FPS)**	**Латентність (мс)**	**Особливі вимоги**
Візуальна одометрія	<1 піксель	10-30	<50	Консистентність між кадрами
Уникнення перешкод	<3 пікселі	30+	<50	Надійність при різному освітленні
Розпізнавання жестів	<2 пікселі	15-30	<100	Стійкість до оклюзій
Автономне водіння	<0,5 пікселя	30+	<30	Низька частка викидів (<5%)

1.5. Висновки до розділу 1

Проведений огляд літератури та аналіз предметної області дозволяє сформулювати наступні висновки.

Еволюція методів від класичних до нейромережевих демонструє значне покращення точності. Класичні методи заклали математичну основу, але мали обмеження через апертурну проблему та чутливість до шуму. Нейромережеві методи забезпечили покращення точності на 60-70% через підхід на основі даних.

Архітектурні інновації включають ключові компоненти - піраміду ознак для багатомасштабної обробки, кореляційні шари для встановлення відповідностей, ітераційне уточнення для субпіксельної точності. PWC-Net ввів пірамідальну схему, RAFT — рекурентне уточнення, FlowFormer — механізми уваги.

Критичними обмеженнями існуючих методів для периферійних пристроїв є обчислювальна складність (PWC-Net ~90,8 GFLOPS, RAFT ~211 GFLOPS), споживання пам’яті (4-8 ГБ для RAFT), несумісність операцій з TPU/NPU архітектурами, відсутність ефективної підтримки квантизації INT8.

Методи оптимізації показують потенціал для застосувань на периферійних пристроях - INT8 квантизація забезпечує зменшення у 4 рази та прискорення у 2-4 рази, групові роздільні згортки зменшують обчислення у 7-8 разів, дистиляція знань покращує точність на 3-5%, архітектурні спрощення можливі без втрати функціональності.

Аналіз апаратних платформ виявив унікальну нішу надмалопотужних пристроїв (MaixCAM з продуктивністю 1 TOPS INT8, 256 МБ пам'яті, споживанням ~0.5 Вт та вартістю $30) порівняно з високопродуктивними рішеннями (Jetson Orin з 40+ TOPS INT8, 8+ ГБ, 7-25 Вт, $250+). Існуючі архітектури не оптимізовані для цього сегменту.

Робототехнічні застосування мають різноманітні вимоги до точності та продуктивності - від високоточної візуальної одометрії (EPE < 1 піксель) до швидкого уникнення перешкод (30+ FPS). Це підтверджує необхідність балансу між точністю та ефективністю.

Прогалинами в існуючих підходах є відсутність архітектур, спеціально оптимізованих для специфічних TPU/NPU прискорювачів, недостатня увага до статичних розмірів тензорів та обмежень компіляторів, відсутність методів ефективної INT8 квантизації без повторного навчання для задач оптичного потоку.

Актуальність дослідження підтверджується виявленою потребою у розробці спеціалізованої архітектури для надмалопотужних периферійних пристроїв, яка поєднує прийнятну точність (EPE ~2 пікселі) з екстремальною ефективністю (<1 GFLOPS, <1 Вт). Це дозволить розширити застосування оптичного потоку на нові класи пристроїв - автономні мікродрони, IoT камери, носимі пристрої та масові робототехнічні системи.

Наступний розділ присвячено розробці теоретичних основ архітектури, оптимізованої під виявлені обмеження та вимоги.

РОЗДІЛ 2
АРХІТЕКТУРА НЕЙРОМЕРЕЖІ ДЛЯ ОБЧИСЛЕННЯ ОПТИЧНОГО ПОТОКУ

2.1. Постановка науково-технічної проблеми

2.1.1. Формалізація задачі оптимізації архітектури для периферійних пристроїв

На основі проведеного в першому розділі аналізу можна сформулювати задачу оптимізації архітектури оптичного потоку для периферійних пристроїв як багатокритеріальну проблему з жорсткими обмеженнями.

Необхідно знайти архітектуру нейронної мережі A та параметри навчання Θ, що мінімізують функцію втрат точності при задоволенні обмежень ресурсів:

$$\min_{A,\Theta} \; L(A, \Theta)$$

при обмеженнях:

$$GFLOPS(A) \leq C_{compute}$$
$$Memory(A) \leq C_{memory}$$
$$Latency(A) \leq C_{time}$$
$$Power(A) \leq C_{power} \tag{2.1}$$

де L — функція втрат (наприклад, середня похибка кінцевої точки EPE); $C_{compute}, C_{memory}, C_{time}, C_{power}$ — обмеження на обчислювальну складність, пам’ять, латентність та енергоспоживання відповідно.

Впровадження нейромережевої моделі на мікрокомп'ютері Sophon CV181x обумовлене низкою жорстких апаратних та архітектурних обмежень. До них належать лімітована обчислювальна потужність на рівні 1 TOPS для операцій з 8-бітною цілочисельною точністю (INT8), малий обсяг оперативної

пам'яті (256 МБ DDR3), а також суворі архітектурні вимоги, такі як необхідність використання статичних розмірів тензорів та обмежений набір підтримуваних операцій. Крім того, критичним фактором є низький енергетичний бюджет, що не перевищує 1 Вт загального споживання. З огляду на ці обмеження, цільовою метрикою якості обрано середню точність з похибкою кінцевої точки (EPE) у діапазоні 2–3 пікселі на тестовому наборі Sintel Clean, що розглядається як оптимальний компроміс між можливостями високоточних методів (EPE близько 1 пікселя) та ефективністю базових підходів (EPE понад 5 пікселів).

2.1.2. Аналіз компромісів точність-ефективність

Фундаментальний компроміс між точністю та ефективністю в задачах оптичного потоку може бути описаний через Парето-оптимальність. Для кожної точки на Парето-фронті покращення одного критерію призводить до погіршення іншого.

Емпіричне спостереження показує, що зв’язок між розміром моделі та точністю має логарифмічний характер:

$$EPE \approx a\log\left(N_{params}\right) + b \tag{2.2}$$

де N_{params} — кількість параметрів моделі; a, b — емпіричні коефіцієнти, залежні від архітектури та набору даних.

Це означає, що зменшення розміру моделі у 10 разів (з 5 мільйонів до 500 тисяч параметрів) призводить до збільшення похибки EPE приблизно на 1 піксель. Для досягнення цільової точності EPE менше 3 пікселі при обмеженнях CV181x необхідно знайти оптимальну точку в просторі компромісів.

Математична модель компромісу точність-швидкість:

$$EPE = f\left(GFLOPS, N_{params}, Architecture\right) \tag{2.3}$$

де *Architecture* включає вибір типів шарів, схеми кореляції, стратегії уточнення.

Критерії вибору архітектурних рішень для розробки нейромережевої моделі базувалися на обмеженнях цільової апаратної платформи та необхідності досягнення оптимального балансу між точністю та ефективністю. Ці критерії включали мінімізацію кількості параметрів до позначки, меншої за 1 мільйон, без критичної втрати точності, що є ключовим для розміщення моделі в обмеженій пам'яті. Також вимагалася максимізація ефективності обчислень на апаратурі INT8 з продуктивністю менше 1 TOPS для забезпечення швидкості інференсу в реальному часі. Важливим технічним критерієм було збереження достатнього діапазону пошуку для ефективної обробки великих зміщень, з цільовим значенням до 96 пікселів на повній роздільності. Нарешті, архітектура мала забезпечувати мінімізацію споживання оперативної пам’яті під час інференсу, не перевищуючи 64 МБ RAM.

2.1.3. Частотно-просторовий аналіз оптичного потоку

Сигнали оптичного потоку можна розкласти на частотні компоненти:

$$f(x, y) = f_{low}(x, y) + f_{high}(x, y) \tag{2.4}$$

де f_{low} — низькочастотна компонента (великі, повільні рухи); f_{high} — високочастотна компонента (дрібні деталі руху).

Частотно-просторовий аналіз оптичного потоку демонструє, що основна енергія векторного поля концентрується в низьких частотах, які відповідають за великомасштабні рухи камери та переміщення великих об'єктів. Водночас, високочастотні компоненти, що несуть інформацію про деталі руху країв та текстур, характеризуються меншою амплітудою.

Відповідно до цього, архітектура обробки оптичного потоку реалізує ієрархічну схему. Обчислення починається на низьких роздільностях 1/32 та 1/16, де відбувається ефективний розрахунок глобального контексту та великих зміщень. На наступних, вищих роздільностях 1/8 та 1/4, здійснюється додавання дрібних деталей та уточнення меж об'єктів.

Завершальний етап включає фінальне уточнення субпіксельних деталей на повній роздільності з мінімальними обчислювальними витратами, що забезпечує точність кінцевого результату.

2.2. Теоретичні основи архітектури EdgeFlowNet

2.2.1. Принципи проектування для TPU-оптимізації

Проектування архітектури під специфічні обмеження TPU вимагає врахування особливостей паралельних обчислень та архітектури пам'яті, описаних у технічній документації Sophon CV181x [13; 14].

TPU-компілятори оптимізують лише статичні графи з фіксованими розмірами тензорів. Математично це означає, що всі операції O_i в графі $G = \{O_1, O_2, ..., O_n\}$ повинні мати детерміновані розміри входів та виходів на етапі компіляції. Архітектура EdgeFlowNet використовує статичне позиційне кодування через попередньо обчислені тензори координат замість динамічних операцій meshgrid, що усуває накладних витрат часу виконання.

TPU має відносно високу обчислювальну потужність (1 TOPS INT8), але обмежену пропускну здатність пам'яті. Оптимальна архітектура максимізує співвідношення compute/memory:

$$Efficiency = \frac{FLOPS\ per\ layer}{Memory\ transfers\ per\ layer} \tag{2.5}$$

Обхідні з'єднання з навчуваного змішування дозволяють зберегти інформацію з попередніх рівнів без додаткових обчислень на критичних шляхах. На масштабах 1/16 та 1/32 застосовується:

$$x_{out} = \alpha \cdot Conv(x_{in}) + (1 - \alpha) \cdot \text{макс} - \text{об'єднання}(x_{in}) \tag{2.6}$$

де α — навчуваний параметр (ініціалізується значенням 0.5), що автоматично балансує між обчисленою та прямо масштабованою інформацією під час тренування.

Замість адаптивних або динамічних операцій використовуються структуровані шаблони, що можуть бути ефективно компільовані. Наприклад, локальна кореляція з фіксованим радіусом замість adaptive attention. стиснення-збудження блоки реалізовані через групові згортки з редукцією каналів замість fully connected шарів для кращої сумісності з TPU.

Навчувані альфа-параметри параметри на кожному рівні уточнення дозволяють мережі автоматично визначати оптимальну величину корекцій:

$$f_{refined} = f_{base} + \alpha_{level} \cdot \Delta f_{level} \tag{2.7}$$

де $\alpha_{level} \in [0, 1]$ — навчуваний параметр, що контролює вплив уточнення на конкретному рівні роздільності.

2.2.2. Математичне обґрунтування подвійної одновимірної кореляції

Подвійна одновимірна глобальна кореляція декомпонує двовимірний простір пошуку на горизонтальну та вертикальну компоненти, що дозволяє радикально зменшити обчислювальну складність при збереженні діапазону пошуку.

Горизонтальна кореляція (пошук вздовж ширини) для тензорів ознак $F_1, F_2 \in R^{C \times H \times W}$ з L2-нормалізацією обчислюється:

$$C_h(p_y, p_x, d_x) = \langle F_1(p_y, p_x), F_2(p_y, d_x) \rangle \frac{}{\sqrt{C}} \tag{2.8}$$

де p_y, p_x — позиція у першому зображенні; $d_x \in [0, W - 1]$ — позиція у другому зображенні вздовж горизонталі; $\langle \cdot, \cdot \rangle$ — скалярний добуток нормалізованих векторів ознак.

Вертикальна кореляція (пошук вздовж висоти) обчислюється так:

$$C_v(p_y, p_x, d_y) = \langle F_1(p_y, p_x), F_2(d_y, p_x) \rangle \frac{}{\sqrt{C}} \tag{2.9}$$

де $d_y \in [0, H - 1]$ — позиція у другому зображенні вздовж вертикалі.

Результуючий тензор кореляції формується конкатенацією:

$$C_{dual1D} = \left[C_h; C_v\right] \in R^{(W+H)\times H\times W} \quad (2.10)$$

Реалізація через матричне множення.

Горизонтальна компонента обчислюється як пакетне матричне множення:

$$C_h = \frac{Q_h \cdot K_h^T}{\sqrt{C}} \in R^{B\times H\times W\times W} \quad (2.11)$$

де Q_h, K_h — переставлені тензори ознак розміром $[B \times H \times W \times C]$ та $[B \times H \times C \times W]$ відповідно. Аналогічно для вертикальної компоненти.

Для повнопарної двовимірної (2D) кореляції обсяг необхідної пам'яті пропорційний до $O\left(H^2W^2\right)$. На відміну від цього, запропонована подвійна одновимірна (1D) схема значно знижує цю залежність, вимагаючи обсяг пам'яті, пропорційний до $O(HW(H + W))$. Для ілюстрації цієї переваги, при обробці карти ознак низької роздільності, де $H = 8, W = 10$ (роздільність s=32 для входу 256×320), повнопарна кореляція генерує 6400 елементів об'єму вартості. Водночас, подвійна 1D схема скорочує цей обсяг до $8 \times 10 \times (8 + 10) = 1440$ елементів (18 каналів після reshape) , демонструючи економію пам’яті близько 355× або 99.7%

При $H = W = 16$ (типова роздільність s=16 без Dual 1D): - Повнопарна: $16^4 = 65536$ елементів - Dual 1D: $16^2 \times 32 = 8192$ елементів - Економія: близько 8× або 87.5%

Переваги подвійної 1D декомпозиції кореляції полягають у забезпеченні високої ефективності та адаптивності до типових сценаріїв руху. Зокрема, вона дозволяє здійснювати захоплення довгодіапазонних рухів, включаючи значні горизонтальні (панорамування камери, рух транспортних засобів) та вертикальні (нахил камери, падіння об'єктів) зміщення, при цьому уникнувши квадратичної обчислювальної складності, характерної для повноцінної двовимірної кореляції. Така архітектура має природну відповідність домінантним рухам камери у відеопослідовності (переважно горизонтальному pan). Незважаючи на декомпозицію, зберігається широкий глобальний діапазон пошуку, що досягає 96 пікселів по горизонталі на повній роздільності (завдяки

каскадному збільшенню роздільності (через каскадне збільшенням роздільності 10×2×2×2×4). З точки зору реалізації, ця схема відрізняється високою ефективністю: вона імплементується через пакетне матричне множення з автоматичним диференціюванням, а відсутність циклів у прямому проході є критичною перевагою для оптимізації та кращої компіляції на тензорних процесорах (TPU).

На середніх рівнях роздільності для обчислення оптичного потоку застосовується механізм традиційної локальної кореляції з обмеженим радіусом пошуку. Цей об'єм вартості C_{local} визначається наступною формулою:

$$C_{local}(p, d) = \langle F_1(p), F_2(p + d) \rangle, |\ |d|\ |_\infty \leq r \tag{2.12}$$

де $r = 6$ — радіус пошуку, що дає $(2 \times 6 + 1)^2 = 169$ можливих зміщень; $|\ |d|\ |_\infty$ — норма Чебишева (максимальна абсолютна координата).

Використання локальної кореляції значно знижує необхідний обсяг пам'яті: розмір тензора об'єму вартості становить: $169 \times H \times W$ на противагу $H^2 \times W^2$ для all-pairs.

Для прикладу, при роздільності карти ознак $H = 16, W = 20$ (що відповідає кроку s=16 для вхідного зображення 256×320), повнопарна кореляція формує 102400 елементів об'єму, тоді як локальна кореляція з r=6 потребує лише 54080 елементів. Це забезпечує економію пам'яті близько 1.9×, або 47%. Комбінований ефект, досягнутий шляхом об'єднання цієї локальної кореляції зі збільшеним об'ємом вартості з кроку s=32 на рівні s=16, формує загальний вхідний тензор із 237 каналів, який складається зі 169 каналів локальної кореляції, 18 каналів збільшеного об'єму вартості, 2 каналів поточного оціненого потоку та 48 каналів контекстних ознак.

2.2.3. Теорія ефективних згорток

Групові роздільні згортки представляють собою декомпозицію стандартної операції згортання на два послідовні етапи обчислень [21]. У цьому

підході стандартна згортка виражається через композицію групової та точкової операцій згідно з формулою (2.13), де групова згортка виконується незалежно для кожного вхідного каналу, а точкова згортка розміром 1×1 здійснює подальше змішування інформації між каналами.

$$Conv_{std}(X) = Conv_{pw}\left(Conv_{dw}(X)\right) \quad (2.13)$$

де $Conv_{dw}$ — групова згортка (окремо для кожного каналу); $Conv_{pw}$ — точкова згортка 1×1 для змішування каналів.

Математичний аналіз обчислювальної ефективності демонструє суттєві переваги групового роздільного підходу. При обробці вхідного тензора розміром $H \times W \times C_{in}$ та формуванні вихідного тензора розміром $H \times W \times C_{out}$ стандартна згортка вимагає $H \times W \times K^2 \times C_{in} \times C_{out}$ операцій з рухомою комою, тоді як групова роздільна згортка потребує лише $H \times W \times \left(K^2 \times C_{in} + C_{in} \times C_{out}\right)$ таких операцій. Коефіцієнт прискорення обчислень визначається співвідношенням (2.14) і залежить від розміру ядра згортки та кількості вихідних каналів.

$$Speedup = \frac{K^2 \times C_{out}}{K^2 + C_{out}} \quad (2.14)$$

Для типових значень $K = 3, C_{out} = 128$ досягається прискорення приблизно у 8,5 рази, тоді як для архітектури EdgeFlowNet з $C_{out} = 48$ на рівнях просторової дискретизації s8, s16 та s32 забезпечується прискорення близько 7,6 разів.

Механізм стиснення-збудження реалізує адаптивну калібровку ознак шляхом моделювання взаємозалежностей між каналами. Класична реалізація SE-блоку застосовує глобальне усереднююче об'єднання з подальшою обробкою повнозв'язними шарами [22]. Для оптимізації виконання на TPU апаратному прискорювачі застосовується модифікований варіант з груповою згорткою, що враховує просторові залежності згідно з виразом (2.15), де

поелементне множення вхідних ознак на активаційну карту здійснюється через сигмоїдальну функцію активації з попереднім глобальним усередненням до просторового розміру 1×1. В архітектурі EdgeFlowNet використовується редукція каналів з коефіцієнтом r=16, завдяки чому проміжний шар містить $\frac{C}{16}$ каналів, що дозволяє зменшити кількість параметрів SE блоку з $2C^2$ до $2\frac{C^2}{16}$ без значної втрати якості. без істотного погіршення якості відновлення оптичного потоку.

$$SE(X) = X \odot \sigma\Big(Conv_{1\times1}(DWConv(GAP(X)))\Big) \tag{2.15}$$

де $\odot$ — поелементне множення; σ — сігмоїда; GAP (глобальне усереднююче об'єднання) — глобального усереднюючого об'єднання до розміру 1×1.

2.2.4. Ітераційне уточнення без рекурентних станів

Традиційні рекурентні блоки, що використовуються в архітектурі RAFT [3], характеризуються наявністю прихованих станів, що істотно ускладнює процес статичної компіляції моделі. У запропонованій роботі реалізовано альтернативний підхід на основі архітектури прямого поширення з явним контролем процесу ітераційного уточнення оптичного потоку.

Модель ітераційного оновлення застосовується на грубих рівнях піраміди ознак, зокрема на масштабах s32 та s16. На кожній ітерації t виконується обчислення за наступним правилом:

$$f^{(t+1)} = f^{(t)} + \Delta f^{(t)} \tag{2.16}$$

де $f^{(t)}$ — оцінка потоку на ітерації t; $\Delta f^{(t)}$ — обчислена корекція.

Функція оновлення реалізується через спеціалізований модуль CoarseUpdateLite, що визначається таким чином:

$$\Delta f^{(t)} = UpdateNet\Big(\Big[C_{corr}, f^{(t)}, ctx\Big]\Big) \tag{2.17}$$

де C_{corr} — тензор кореляції (18 каналів на s32, 237 каналів на s16); ctx — контекстні ознаки з першого зображення (48 каналів).

Архітектура модуля CoarseUpdateLite побудована на основі послідовного застосування згорткових операцій. Вхідний тензор спочатку обробляється згорткою розміру 1×1 для редукції розмірності до прихованого простору з 128 каналами. Далі застосовується послідовність блоків depthwise-pointwise згорток з нормалізацією BatchNorm та функцією активації LeakyReLU. Завершальним етапом є головка з ядром 3×3, що здійснює передбачення двоканального вектора корекції з компонентами delta_u та delta_v.

Кількість ітерацій уточнення визначена фіксовано та становить дві ітерації на рівні s32 і дві ітерації на рівні s16. Така фіксована структура дозволяє виконати статичне розгортання циклів на етапі компіляції моделі.

Запропонований підхід на основі архітектури прямого поширення забезпечує повну статичність обчислювального графу завдяки відсутності прихованих станів, що потребують збереження між ітераціями. Така організація обчислень уможливлює ефективну паралелізацію операцій у межах однієї ітерації та гарантує детерміновану обчислювальну складність процесу виведення без використання умовних операторів.

2.2.5. Архітектура багаторівневого уточнення

Після ітераційного оновлення на грубих рівнях застосовується каскад уточнюючих блоків на роздільностях s16, s8, s4 та повній роздільності.

Уточнення на середніх роздільностях (s16, s8, s4).

На кожному рівні $l \in \{16, 8, 4\}$ виконується:

$$f_l^{init} = Upsample\left(f_{l\times 2}^{refined}\right) \times 2 \tag{2.18}$$

$$\Delta f_l = RefineLite_l\left(\left[f_{0,l}, f_{1,l}, f_l^{init}, ctx_l\right]\right) \tag{2.19}$$

$$f_l^{refined} = f_l^{init} + \alpha_l \cdot \Delta f_l \tag{2.20}$$

де $f_{0,l}, f_{1,l}$ — ознаки з кодувальника; ctx_l — контекстні ознаки; α_l — навчуваний параметр масштабування (ініціалізується 0.5).

Архітектура модуля RefineLite побудована за принципом послідовної обробки конкатенованих ознак різних рівнів. Спочатку застосовується редукція розмірності через згортку 1×1 до прихованої розмірності, яка становить 128 каналів для рівнів s16 та s8, і 64 канали для рівня s4. Далі слідує послідовність з шести depthwise-pointwise блоків, організованих у три цикли трансформацій, де кожен цикл складається з depthwise згортки, пакетної нормалізації, функції активації ReLU, pointwise згортки, повторної пакетної нормалізації та активації. Завершується модуль спеціалізованою головкою на основі згортки 3×3, що передбачає двоканальну корекцію потоку. Загальна розмірність вхідних даних для модуля становить 146 каналів на рівнях s16 і s8, та 98 каналів на рівні s4, що відображає різну глибину ознакових представлень на відповідних масштабах.

Після завершення уточнення на рівні s4 та збільшення роздільності потоку до повного розміру вхідних зображень застосовується спеціалізований легковаговий блок остаточного уточнення:

$$f_{full}^{init} = Upsample\left(f_4^{refined}\right) \times 4 \tag{2.21}$$

$$\Delta f_{full} = FullRefine\left(\left[f_{full}^{init}, I_0\right]\right) \tag{2.22}$$

$$f_{final} = f_{full}^{init} + \alpha_{full} \cdot \Delta f_{full} \tag{2.23}$$

де I_0 позначає оригінальне перше зображення RGB з трьома каналами, сформований вхід містить п’ять каналів (дві компоненти потоку та три кольорові канали), а α_{full} є навчуваним параметром адаптивного масштабування остаточної корекції.

Архітектура модуля FullRefine спроєктована з урахуванням мінімізації обчислень на повній роздільності, що критично важливо для забезпечення прийнятної латентності виведення на обмежених апаратних платформах. Модуль послідовно застосовує базовий блок DWConvBlock для трансформації вхідних п’яти каналів у тридцятидвоканальне представлення, після чого використовує розріджену згортку з коефіцієнтом розширення два для

ефективного збільшення рецептивного поля без додаткових обчислювальних витрат. Далі застосовується pointwise згортка для міксування каналів у тридцятидвовимірному просторі ознак, і завершується обробка спеціалізованою головкою на основі згортки 1×1, що редукує тридцять два канали до двоканального представлення корекції потоку.

Використання оригінального вхідного зображення на етапі остаточного уточнення має фундаментальне теоретичне обґрунтування. Високочастотні деталі візуальної інформації, такі як чіткі краї об’єктів та дрібні текстурні елементи, найкраще збережені саме в оригінальному RGB зображенні, оскільки послідовні операції зменшення роздільності в енкодері неминуче призводять до часткової втрати цієї інформації внаслідок просторової субдискретизації. Пряме використання RGB даних дозволяє мережі уточнити межі об’єктів на повній роздільності шляхом аналізу градієнтів інтенсивності, що недоступні в багаторазово підвибраних ознакових картах середніх рівнів піраміди.

Параметри адаптивного масштабування на кожному рівні уточнення забезпечують мережі можливість автоматичного балансування впливу базової оцінки, успадкованої з попереднього грубшого рівня, та обчисленої локальної корекції. Під час процесу навчання мережа здатна адаптивно зменшувати значення для тих рівнів, де корекції виявляють тенденцію до перенавчання на специфічних артефактах тренувальних даних, одночасно збільшуючи його для рівнів, де корекції систематично покращують точність передбачення. Крім того, ці параметри забезпечують автоматичне налаштування абсолютної величини вносимих корекцій залежно від статистичних характеристик та рівня складності використовуваного набору даних. Емпіричні спостереження показують, що після завершення навчання типові значення параметрів стабілізуються в діапазоні від 0.3 до 0.7, що підтверджує їхню природу як справді адаптивних механізмів саморегуляції мережі.

2.3. Математичний апарат навчання моделі

2.3.1. Багатомасштабна функція втрат

Процес навчання розробленої моделі здійснюється з використанням багатомасштабної функції втрат, що обчислюється на чотирьох різних рівнях роздільності просторової піраміди. Загальна функція втрат формулюється як зважена сума індивідуальних втрат на кожному масштабі:

$$L_{total} = \sum_{\ell \in \{32,16,8,4\}} w_{\ell} \cdot L_{\ell}\left(f_{\ell}, f_{gt}\right) \tag{2.24}$$

де w_{ℓ} позначає вагові коефіцієнти для кожного масштабного рівня, f_{ℓ} представляє передбачений оптичний потік на рівні ℓ, а f_{gt} є еталонними даними оптичного потоку з ground truth розмітки.

Налаштування вагових коефіцієнтів здійснено згідно з принципом прогресивного зростання впливу детальніших рівнів: $w_{32} = 0.05$ для найгрубішого рівня, $w_{16} = 0.15$ для проміжного, $w_{8} = 0.30$ для передостаннього, та $w_{4} = 0.50$ для найдетальнішого рівня перед остаточним виведенням. Така схема розподілу ваг базується на принципі максимізації впливу фінального рівня s4, оскільки саме він найближчий до цільової точності передбачення та несе найбільшу відповідальність за кінцеву якість результату. Передбачення на повній роздільності навмисно виключено з функції втрат під час навчання, оскільки цей етап призначений виключно для остаточного виведення результатів і не повинен впливати на процес оптимізації параметрів основних уточнюючих модулів.

На кожному окремому масштабному рівні застосовується стійка функція втрат Шарбоньє замість традиційної квадратичної метрики L2, що забезпечує підвищену стійкість до статистичних викидів та аномальних значень потоку:

$$L_{\ell}\left(f, f_{gt}\right) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{|)f_i - f_{gt,i}|)^2 + \epsilon^2} \quad (2.25)$$

де N представляє загальну кількість пікселів у зображенні, $\epsilon = 0.01$ є малим параметром регуляризації для забезпечення числової стабільності обчислень, $|) \cdot |)$ позначає евклідову L2 норму по компонентах векторного поля потоку (горизонтальна u та вертикальна v складові), а f_{gt} представляє еталонні дані оптичного потоку з ground truth розмітки набору даних.

Функція втрат Charbonnier являє собою гладку диференційовану апроксимацію метрики L1, що надає їй природну стійкість до значних помилок передбачення, які часто виникають в областях оклюзії, де встановлення коректних відповідностей між кадрами є принципово неможливим через приховування об'єктів. На відміну від квадратичної метрики L2, яка надає непропорційно великі штрафи для значних відхилень, функція Charbonnier забезпечує більш збалансоване навчання навіть за наявності систематичних областей з невизначеним потоком.

Для збереження просторової структури меж об'єктів та забезпечення природних розривів поля оптичного потоку на семантичних границях сцени, до загальної функції втрат додається спеціалізований регуляризуючий член, що враховує градієнти вхідного зображення:

$$L_{smooth} = \sum_{i,j} e^{-|)\nabla I_0(i,j)|)} \cdot \left(|\nabla_x f(i,j)) + |\nabla_y f(i,j))\right) \quad (2.26)$$

де ∇I_0 позначає просторовий градієнт інтенсивності вхідного зображення, а ∇_x та ∇_y представляють просторові градієнти поля оптичного потоку у горизонтальному та вертикальному напрямках відповідно.

Цей регуляризуючий член реалізує адаптивну схему згладжування, що заохочує плавність поля оптичного потоку в просторово однорідних областях зображення, де градієнт інтенсивності має малу величину, одночасно дозволяючи різкі просторові розриви векторного поля на семантично значущих

межах об’єктів, де градієнт інтенсивності приймає великі значення. Завдяки експоненційному зважуванню за градієнтом зображення, штраф за негладкість автоматично зменшується в областях високих градієнтів, що природно корелюють з границями об’єктів у сцені.

Повна функція втрат формується шляхом додавання основного багатомасштабного члена та регуляризуючого члена згладжування з відповідним ваговим коефіцієнтом:

$$L = L_{total} + \lambda_{smooth} \cdot L_{smooth} \quad (2.27)$$

де ваговий коефіцієнт регуляризації λ_{smooth} встановлено на рівні 0.1 для забезпечення помірного впливу згладжуючого члена без надмірного придушення природних розривів поля потоку на межах об’єктів.

2.3.2. Стратегія поступового навчання

Процес навчання розробленої архітектури моделі організовано у вигляді двофазного підходу з послідовним використанням різних наборів даних та відповідним налаштуванням гіперпараметрів оптимізації на кожній фазі. На першій фазі здійснюється базове навчання на великомасштабному синтетичному наборі даних FlyingChairs [18], що містить 22872 пари зображень з синтетично згенерованими сценами руху. Тренування на цій фазі проводиться протягом 60000 ітерацій градієнтного спуску з розміром пакету 64 зразки. Швидкість навчання регулюється за схемою косинусного затухання від початкового значення 2.0×10^{-3} до кінцевого 2.0×10^{-4}, з додатковою фазою прогрівання на перших 125 кроках, де швидкість навчання лінійно зростає від нуля до початкового значення. Метою цієї фази є формування базових навичок встановлення відповідностей між зображеннями та обчислення векторних полів оптичного потоку на синтетичних даних, які характеризуються великими величинами зміщень та чітко визначеними контурами об’єктів.

Друга фаза навчання присвячена тонкому налаштуванню моделі на реальних даних шляхом використання комбінованого набору даних, що складається з наборів Sintel Clean, Sintel Final та KITTI 2015. Тренування на цій фазі продовжується протягом 20000 ітерацій з незміненим розміром пакету 64 зразки. Швидкість навчання також регулюється за схемою косинусного затухання, але з меншими початковим значенням 4.0×10^{-4} та кінцевим значенням 4.0×10^{-5}, що відповідає характеру тонкого налаштування попередньо навчених параметрів. Метою другої фази є адаптація моделі до специфічних характеристик реальних відеопослідовностей, включаючи природні текстурні патерни, складні умови освітлення, атмосферні ефекти та оптичні спотворення камери.

Теоретичне обґрунтування двофазної стратегії полягає в тому, що набір даних FlyingChairs забезпечує чистий та легко оптимізований навчальний сигнал для початкового формування здатності мережі встановлювати кореспонденції між послідовними кадрами, тоді як реальні набори даних Sintel та KITTI додають реалістичну складність через розмиття руху, атмосферні явища та геометричні спотворення оптичної системи, що суттєво покращує здатність моделі до узагальнення на невідомі реальні відеопослідовності.

Протягом усього процесу навчання підтримується експоненційно згладжена копія параметрів моделі за методом Exponential Moving Average:

$$\theta_{EMA}^{(t+1)} = \beta \cdot \theta_{EMA}^{(t)} + (1 - \beta) \cdot \theta^{(t)} \tag{2.28}$$

де коефіцієнт згладжування $\beta = 0.9995$ визначає швидкість затухання попередніх значень, $\theta^{(t)}$ позначає поточні параметри моделі після застосування кроку градієнтного спуску на ітерації t, а $\theta_{EMA}^{(t)}$ представляє експоненційно згладжені параметри на відповідній ітерації.

Експоненційно згладжені параметри використовуються виключно для процедури валідації та збереження фінальної версії моделі, оскільки вони

характеризуються значно вищою стабільністю та демонструють кращу здатність до узагальнення порівняно з миттєвими параметрами, отриманими безпосередньо з градієнтної оптимізації. Метод EMA ефективно усереднює траєкторію параметрів у просторі ваг, що природно придушує високочастотні флуктуації, спричинені стохастичною природою мінібатчевого градієнтного спуску.

2.3.3. Оптимізація та регуляризація

Для оптимізації параметрів моделі застосовується алгоритм AdamW [29] з налаштуваннями моментів першого та другого порядку і відповідно, а також з коефіцієнтом затухання ваг величиною , що забезпечує м’яку L2 регуляризацію параметрів для запобігання перенавчанню. Для підтримки стабільності процесу навчання та уникнення катастрофічних розбіжностей, спричинених аномально великими градієнтами, застосовується процедура обмеження норми градієнта згідно з формулою:

$$g_{clipped} = min\left(1, \frac{1.0}{| \, |g| \, |}\right) \cdot g \qquad (2.29)$$

де позначає вектор градієнтів усіх параметрів моделі, об’єднаних у єдине представлення, а операція обмеження забезпечує, що евклідова норма результуючого градієнта не перевищує порогове значення 1.0.

Процес навчання здійснюється з використанням автоматичної змішаної точності обчислень, де основні операції виконуються в форматі FP16 для прискорення обчислень на графічних процесорах, тоді як критичні операції, чутливі до точності (зокрема, акумуляція втрат), залишаються у форматі FP32. Для забезпечення числової стабільності при роботі зі зменшеним діапазоном FP16 застосовується динамічне масштабування функції втрат.

Збагачення тренувальних даних здійснюється через застосування випадкових трансформацій до вхідних зображень. Варіювання параметрів кольору реалізується шляхом незалежних випадкових змін яскравості, контрасту, насиченості та відтінку в діапазоні ±10% від оригінальних значень,

що імітує різноманітність умов освітлення реальних сцен. Ефект розмиття руху додається з імовірністю 10% через застосування одновимірного ядра згортки розміром від 3 до 5 пікселів у випадковому напрямку, що моделює артефакти швидкого руху камери або об’єктів. Горизонтальне відзеркалення застосовується з імовірністю 50% для збільшення інваріантності моделі до орієнтації руху. Просторова роздільність вхідних зображень фіксується на значенні 256×320 пікселів, що представляє співвідношення сторін приблизно 1.25.

Для запобігання перенавчанню реалізовано механізм раннього зупинення навчання, що моніторить значення функції втрат на валідаційному наборі даних з параметром терпіння шість епох та мінімальною значущою зміною 0.001, після чого процес навчання автоматично припиняється при відсутності поліпшення метрики.

2.3.4. Квантизація INT8 та оптимізація для інференсу

Після завершення навчання в форматі FP32 модель піддається процедурі квантизації до восьмибітного цілочисельного представлення INT8 для забезпечення ефективного розгортання на тензорному процесорі CV181x з обмеженою обчислювальною потужністю. Застосовується метод посттренувальної квантизації з симетричною схемою представлення для ваг та активацій:

$$x_{quant} = clip\left(round\left(\frac{x}{s}\right), -128, 127\right) \quad (2.30)$$

де коефіцієнт масштабування визначається з максимального абсолютного значення відповідного тензора, а функція clip забезпечує обмеження результату в допустимому діапазоні восьмибітного цілочисельного представлення зі знаком від мінус 128 до плюс 127.

Процедура калібрування для визначення оптимальних коефіцієнтів масштабування кожного шару мережі проводиться на репрезентативній вибірці від 100 до 200 зображень з тренувального набору даних, що забезпечує

статистично достовірну оцінку діапазонів активацій без надмірних обчислювальних витрат на повний прохід через весь набір даних.

Очікувана деградація точності після застосування квантизації в вісім біт становить від 7 до 8 відсотків замість типових від 15 до 25 відсотків для стандартних архітектур, що не оптимізовані під квантизацію. Така суттєва перевага досягається завдяки системному врахуванню вимог квантизації на етапі проектування архітектури та вибору структурних компонентів моделі.

2.4. Висновки до розділу

У розділі розроблено теоретичні основи архітектури EdgeFlowNet для обчислення оптичного потоку на периферійних пристроях з обмеженими ресурсами. Формалізовано задачу оптимізації архітектури як багатокритеріальну проблему з жорсткими обмеженнями на обчислювальну потужність, пам’ять, латентність та енергоспоживання.

Запропоновано інноваційну подвійну одновимірну схему глобальної кореляції, що декомпонує двовимірний простір пошуку на горизонтальну та вертикальну компоненти. Математично доведено зменшення споживання пам’яті у 355 разів порівняно з повнопарною кореляцією при збереженні діапазону пошуку до 96 пікселів на повній роздільності.

Розроблено принципи TPU-оптимізації архітектури, що включають статичність обчислювального графу через фіксовані розміри тензорів, мінімізацію пропускної здатності пам’яті через обхідні з’єднання з навчуваним змішуванням, структуровану розрідженість замість адаптивних операцій, адаптивне масштабування впливу уточнень через навчувані альфа-параметри.

Формалізовано математичний апарат багаторівневого уточнення з архітектурою прямого поширення без рекурентних станів, що забезпечує детерміновану складність та повну статичність для ефективної компіляції на TPU. Запропоновано архітектуру full-resolution refine блоку з мінімальними

обчисленнями, що використовує пряме RGB зображення для збереження високочастотних деталей.

Розроблено стратегію поступового навчання з двофазним підходом, що передбачає базове навчання на синтетичних даних FlyingChairs (60000 кроків) та тонке налаштування на реальних даних Sintel і KITTI (20000 кроків). Використано багатомасштабну функцію втрат з стійкою Charbonnier loss та edge-aware smoothness регуляризацією для збереження меж об’єктів.

Запропоновано методологію Post-Training Quantization до INT8 з архітектурними оптимізаціями для мінімізації деградації точності: використання LeakyReLU, об’єднання BatchNorm, поканальна квантизація ваг. Очікувана деградація точності обмежена 7-8% замість типових 15-25%.

Теоретичні результати розділу створюють фундамент для практичної реалізації архітектури EdgeFlowNet, що буде детально описана у наступному розділі.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

3.1. Вибір інструментів розробки

3.1.1. Мова програмування Python

Для реалізації архітектури EdgeFlowNet та проведення експериментальних досліджень обрано мову програмування Python версії 3.10. Вибір обумовлений найбільш розвиненою екосистемою бібліотек для машинного навчання та комп'ютерного зору, де фреймворки PyTorch, TensorFlow та ONNX Runtime забезпечують повний цикл розробки від прототипування до розгортання моделей на цільовій апаратурі [32]. Зокрема PyTorch 2.0 з підтримкою компіляції обчислювальних графів забезпечує оптимізацію на рівні компілятора що критично важливо для досягнення максимальної продуктивності нейронних мереж.

Незважаючи на інтерпретований характер мови всі обчислювально інтенсивні операції виконуються через скомпільовані бекенди на мовах C++ та CUDA. Бібліотека NumPy використовує оптимізовані реалізації BLAS та LAPACK [34], а PyTorch компілює тензорні операції через бібліотеку ATen що забезпечує продуктивність порівнянну з нативним кодом при мінімальних накладних витратах на інтерпретацію приблизно один-два відсотки. Динамічна типізація та інтерактивні інструменти розробки дозволяють швидко прототипувати архітектури, візуалізувати проміжні результати та налагоджувати помилки, що особливо важливо на етапі експериментування з різними конфігураціями гіперпараметрів де час ітерації між ідеєю та результатом має бути мінімальним.

Критичним фактором вибору є сумісність з інструментальним ланцюгом TPU-MLIR що використовується мікропроцесором Sophon CV181x для конвертації моделей у формат тензорного процесора [14]. Компілятор приймає

моделі у форматі ONNX який може бути експортований з PyTorch через відповідний програмний інтерфейс забезпечуючи безшовну інтеграцію між навчанням моделі та експортом для розгортання. Для промислового розгортання на платформі MaixCAM модель експортується у формат ONNX та компілюється у бінарний файл для тензорного процесора, а виведення виконується через середовище виконання MaixCDK на мові C++ або програмний інтерфейс MaixPy що забезпечує мінімальні накладні витрати на апаратурі приблизно п’ять-десять мілісекунд додаткової затримки для обгортки.

3.1.2. Фреймворк PyTorch

PyTorch версії 2.0+ обрано як основний фреймворк глибокого навчання для реалізації та навчання архітектури EdgeFlowNet [32]. Вибір обумовлений технічними перевагами для розробки нестандартних архітектур під обмеження периферійних пристроїв. Фреймворк використовує підхід визначення через виконання, де обчислювальний граф будується динамічно під час прямого проходу, що дозволяє природно імплементувати ітераційне уточнення оптичного потоку через звичайні Python цикли без необхідності розгортання статичних конструкцій.

Система автоматичного диференціювання реалізована через зворотний режим на основі обчислювальної стрічки, де кожна операція записує свою функцію градієнта, забезпечуючи коректне обчислення градієнтів навіть для архітектур з обхідними з’єднаннями та багаторівневими функціями втрат. Модульна архітектура дозволяє композувати складні моделі з простих блоків через об’єктно-орієнтовану парадигму, де кожен компонент реалізується як окремий модуль з власними параметрами та методом прямого проходу.

Фреймворк надає широкий набір оптимізаторів з підтримкою затухання ваг, обмеження градієнтів та тренування зі змішаною точністю, а планувальники швидкості навчання дозволяють реалізувати складні стратегії з періодом прогрівання та косинусним затуханням [31]. Критичною можливістю є

нативна підтримка експорту моделей у формат ONNX через відповідний інтерфейс програмування з версією набору операцій 13+, оскільки компілятор TPU-MLIR приймає моделі саме у цьому форматі. Експорт підтримує трасування на фіктивних входах для створення статичного графу без динамічних операцій.

Автоматична змішана точність через відповідний модуль дозволяє навчання з активаціями FP16 та градієнтами FP32, зменшуючи споживання пам’яті на 40-50% та прискорюючи тренування у 2-3 рази на GPU з тензорними ядрами, при цьому автоматичне масштабування втрат запобігає числовому переповненню. Архітектура EdgeFlowNet спроектована з урахуванням обмежень компілятора TPU-MLIR, що погано підтримує операції деформації, динамічні розміри тензорів та деякі функції активації, тому використовуються виключно базові операції з повністю статичними розмірами.

3.1.3. Бібліотека обробки зображень OpenCV

OpenCV (Open Source Computer Vision Library) версії 4.8+ використовується для операцій попередньої обробки зображень, зміни розміру та читання медіа файлів [33]. Функція зміни розміру застосовується для приведення вхідних зображень різних наборів даних до уніфікованого розміру 256×320 пікселів. Бібліотека реалізує кілька алгоритмів інтерполяції, серед яких для зображень використовується бікубічна інтерполяція для найкращої візуальної якості, тоді як для еталонного оптичного потоку застосовується білінійна інтерполяція з коректним масштабуванням векторних компонент. Функція відеозахоплення забезпечує декодування відеопотоку H.264/H.265 через апаратні відеодекодери для обробки у реальному часі.

Бібліотека надає конвертацію між колірними просторами, що використовується для аугментацій через простір HSV, хоча основна обробка ведеться у RGB представленні. Компіляція з підтримкою векторних інструкцій та багатопоточності забезпечує продуктивність близько 1000 кадрів за секунду

на одному процесорному ядрі для операцій зміни розміру, що робить попередню обробку незначним вузьким місцем конвеєра. Imageio v3 використовується як основна бібліотека для читання зображень та файлів оптичного потоку формату .flo завдяки чистішому інтерфейсу програмування з прямим поверненням RGB масивів та кращій підтримці шістнадцятибітних форматів.

Для читання бінарного формату Middlebury .flo реалізовано спеціалізовану функцію, що парсить магічне число, розмірності та масив значень компонент потоку з точним збереженням субпіксельних значень без втрат від квантизації. NumPy є фундаментальною бібліотекою для роботи з багатовимірними масивами на всіх етапах конвеєра [34]. Використовуються операції об’єднання компонент потоку, заповнення при зміні розміру зі збереженням співвідношення сторін, генерації випадкових параметрів аугментацій та створення координатних сіток. Реалізація на мові C з підтримкою оптимізованих математичних бібліотек забезпечує продуктивність векторизованих операцій близьку до теоретичного максимуму пропускної здатності процесора та пам’яті.

3.1.4. Бібліотека Imageio та NumPy

Imageio v3 використовується як основна бібліотека для читання зображень та файлів оптичного потоку у форматі .flo (Sintel, FlyingChairs). Переваги над OpenCV: чистіший API (imageio.v3.imread повертає RGB, а не BGR), краща підтримка 16-bit форматів (KITTI flow у PNG), нативна інтеграція з NumPy масивами. Для читання оптичного потоку у форматі Middlebury .flo реалізовано нестандартну функцію read_flo, яка парсить бінарний формат: магічне число (202021.25 у float32), ширина та висота (int32), та масив float32 значень (u, v компоненти потоку). Формат .flo забезпечує точне збереження субпіксельних значень потоку без втрат від квантизації. NumPy є фундаментальною бібліотекою для роботи з багатовимірними масивами [34]. Бібліотека NumPy є

фундаментальною для роботи з багатовимірними масивами. Її функціонал задіяний на всіх етапах конвеєра обробки даних: від зчитування зображень та перетворення їх у масив NumPy, подальшого конвертування в тензор PyTorch для навчання, до зворотного перетворення в масив NumPy та збереження кінцевих результатів. Серед ключових операцій, що використовуються, можна виділити наступні: np.stack: використовується для об'єднання компонентів оптичного потоку u та v в єдиний масив розмірності [2, H, W]. np.pad: застосовується для заповнення (padding) зображень при зміні їх розміру із збереженням співвідношення сторін. np.random: використовується для генерації випадкових параметрів аугментацій з фіксованим початковим значенням (seed). np.meshgrid: застосовується для створення координатних сіток, необхідних для виконання геометричних трансформацій. NumPy реалізований на C з підтримкою BLAS/LAPACK, що забезпечує продуктивність порівнянну з нативним кодом. Векторизовані операції через broadcasting уникають явних циклів Python та виконуються на швидкості близькій до теоретичного максимуму пропускної здатності CPU/пам'яті.

3.1.5. Додаткові інструменти та бібліотеки

Бібліотека TorchVision 0.15+ надає готові реалізації базових аугментацій та утиліти для створення візуалізацій через комбінування множини зображень у єдину сітку. Matplotlib 3.7+ використовується виключно для конвертації оптичного потоку у кольорове зображення через HSV колірний простір, де відтінок відображає напрямок руху, насиченість відповідає магнітуді вектора, а яскравість залишається константною. ONNX Runtime 1.15+ застосовується для валідації експортованих моделей перед компіляцією на TPU шляхом порівняння виходу з еталонним результатом PyTorch при числовій толерантності менше однієї мільйонної для тридцятидвобітної точності.

Компілятор TPU-MLIR від Sophon Technology конвертує моделі ONNX у формат для тензорного процесора [14]. Інструментарій включає трансформацію

для конвертації у проміжне представлення, розгортання для компіляції з квантизацією INT8, та засіб калібрування для збору статистики активацій на репрезентативних даних. Компілятор має обмеження на динамічні форми та деякі операції виконуються повільними резервними реалізаціями на центральному процесорі, що враховано при проектуванні архітектури EdgeFlowNet.

3.2. Характеристики апаратної платформи

3.2.1. Мікрокомп’ютер Sipeed MaixCAM

Sipeed MaixCAM є компактним мікрокомп’ютером розміром 65×30 міліметрів, спроєктованим для застосувань комп’ютерного зору з інтегрованим прискорювачем нейронних мереж. Система базується на чипі Sophon SG2002 з двоядерною архітектурою RISC-V C906 та ARM Cortex-A53 з тактовою частотою 1.0 ГГц, виготовленим за 22-нанометровим техпроцесом. Платформа оснащена спеціалізованим тензорним процесором з продуктивністю 1 TOPS для восьмибітних цілочисельних обчислень, оптимізованим для згорткових нейронних мереж [13].

Обсяг оперативної пам’яті становить 256 МБ DDR3 з частотою 1600 МГц, розподілений між центральним процесором та прискорювачем, що створює жорсткі обмеження на розмір моделі та проміжні активації з резервуванням не більше 64 МБ для параметрів моделі. Накопичувач представлений 32 МБ флеш-пам’яті для операційної системи та додатків з можливістю розширення через карти пам’яті до 128 ГБ. Інтегрований п’ятимегапіксельний відеосенсор GC4653 підключений через високошвидкісний інтерфейс MIPI CSI-2 з підтримкою роздільності до 2880×1620 при 30 кадрах за секунду.

Периферія включає дисплей 2.4 дюйма з сенсорним управлінням, бездротовий модуль WiFi 6, порт USB Type-C та GPIO роз’єм для підключення зовнішніх пристроїв. Згідно з офіційним даташитом, типове енергоспоживання

для сценарію з обробкою відео та штучним інтелектом становить близько 500 мілівт, що при роботі EdgeFlowNet залишається в межах одного вата, забезпечуючи автономну роботу від портативного акумулятора протягом 8-10 годин безперервного функціонування.

3.2.2. Архітектура Sophon CV181x TPU

Sophon CV181x інтегрує спеціалізований тензорний процесор для периферійних систем штучного інтелекту [13]. Обчислювальне ядро містить матричний множник розміром 16×16 елементів для восьмибітних цілочисельних операцій з піковою продуктивністю 1024 операції множення-накопичення за такт при частоті 1 ГГц, доповнений векторним процесором з тридцятьма двома паралельними лініями для поелементних операцій.

Ієрархія пам’яті включає швидкий кеш першого рівня обсягом 256 кілобайт з доступом за один такт, уніфікований кеш другого рівня обсягом 1 мегабайт з доступом за 3-5 тактів, та основну пам’ять 128 мегабайт з латентністю 100-200 тактів, де ефективність критично залежить від локальності даних та мінімізації звернень до основної пам’яті.

Підтримувані операції охоплюють двовимірні згортки з ядрами різних розмірів, групові та роздільні згортки, операції об’єднання, поелементні трансформації та пакетну нормалізацію з поглинутими параметрами, що виконуються з восьмибітною точністю та опціональною тридцятидвобітною акумуляцією зміщення. Архітектура вимагає статичних розмірів тензорів, відомих на етапі компіляції, та не підтримує динамічні операції індексування, умовного виконання чи циклів зі змінними границями.

Компілятор перетворює моделі ONNX у оптимізований байт-код через етапи виведення форми тензорів, зниження операторів до примітивів процесора, оптимізації використання пам’яті через повторне використання буферів та квантизації з калібруванням на репрезентативних даних [14].

Підтримувані операції охоплюють двовимірні згортки з ядрами різних розмірів, групові та роздільні згортки, операції об’єднання, поелементні трансформації та пакетну нормалізацію з поглинутими параметрами, що виконуються з восьмибітною точністю та опціональною тридцятидвобітною акумуляцією зміщення. Архітектура вимагає статичних розмірів тензорів, відомих на етапі компіляції, та не підтримує динамічні операції індексування, умовного виконання чи циклів зі змінними границями.

Компілятор перетворює моделі ONNX у оптимізований байт-код через етапи виведення форми тензорів, зниження операторів до примітивів процесора, оптимізації використання пам’яті через повторне використання буферів та квантизації з калібруванням на репрезентативних даних [14].

3.2.3. Порівняння з альтернативними платформами

Таблиця 3.1

Порівняльна характеристика edge AI платформ

Платформа	**TPU/NPU продуктивність (INT8)**	**RAM**	**Енергоспоживання**	**Вартість**	**Розмір**
MaixCAM (CV181x/SG2002)	1.0 TOPS	256 МБ	~0.5 Вт	$30	65×30 мм
Jetson Orin Nano	40 TOPS	8 ГБ	7-25 Вт	$250	100×80 мм
Coral Dev Board	4 TOPS	4 ГБ	2-4 Вт	$150	88×60 мм
Raspberry Pi 5 + Coral	4 TOPS	8 ГБ	~8 Вт	$145	85×56 мм

MaixCAM займає унікальну нішу енергоекономічних пристроїв з мінімальним енергоспоживанням та розміром при прийнятній AI продуктивності для базових задач комп’ютерного зору.

3.3. Реалізація архітектури EdgeFlowNet

3.3.1. Загальна структура моделі

Архітектура EdgeFlowNet спроєктована як композиція шести основних модулів, оптимізованих для роботи на тензорному процесорі Sophon CV181x. Обробка вхідної пари зображень розміром 256×320 пікселів проходить через послідовний конвеєр перетворень, що починається з витягування багаторівневих ознак та завершується фінальним уточненням на повній роздільності.

Модуль EfficientEncoder виконує ієрархічне витягування ознак на чотирьох просторових масштабах з коефіцієнтами зменшення 4, 8, 16 та 32. Архітектура енкодера базується на глибинно-роздільних згортках з обхідними з'єднаннями та блоками стиснення-збудження, що забезпечує ефективне представлення просторової інформації при мінімальних обчислювальних витратах. Компонент містить близько 87 тисяч параметрів та споживає приблизно 0,25 гігафлопс обчислювальних операцій.

На найгрубшому рівні роздільності модуль GlobalCorrelationDual1D обчислює кореляцію між ознаками двох кадрів, використовуючи інноваційну схему декомпозиції двовимірного простору пошуку на горизонтальну та вертикальну компоненти. Така одновимірна декомпозиція радикально зменшує споживання пам'яті порівняно з традиційною повнопарною кореляцією. Паралельно, модуль LocalCostVolumePad виконує локальну кореляцію з радіусом шість пікселів на середньому рівні роздільності, що дозволяє захоплювати дрібномасштабні деталі руху. Обидва модулі реалізовано без власних навчуваних параметрів.

Модуль CoarseUpdateLite здійснює ітераційне уточнення оцінки потоку на грубих роздільностях без використання рекурентних механізмів. На кожному з двох найгрубших рівнів виконується по дві ітерації оновлення, що забезпечує поступове покращення точності при збереженні статичності обчислювального

графу. Компонент містить близько 164 тисяч параметрів та споживає приблизно 0,16 гігафлопс обчислень.

Наступним етапом є застосування модулів RefineLite на роздільностях з коефіцієнтами зменшення 16, 8 та 4. Ці блоки додають локальні деталі до попередньо обчисленої грубої оцінки потоку, використовуючи навчувані параметри масштабування для адаптивного контролю величини вносимих корекцій. Це найбільш ресурсномісткий компонент архітектури, що містить близько 230 тисяч параметрів та споживає приблизно 0,94 гігафлопс обчислювальної потужності.

Завершальний етап обробки виконується модулем фінального уточнення на повній роздільності, який використовує оригінальні RGB зображення для відновлення високочастотних деталей на межах об’єктів. Цей легкий компонент містить лише 19 тисяч параметрів та споживає близько 0,07 гігафлопс обчислень.

Загальна кількість навчуваних параметрів моделі становить 501 548, а обчислювальна складність для обробки пари зображень розміром 256×320 пікселів дорівнює приблизно 1,48 гігафлопс після компіляції інструментарієм TPU-MLIR. Математично, модель реалізує відображення $f: \mathbb{R}^{3\times256\times320} \times \mathbb{R}^{3\times256\times320} \rightarrow \mathbb{R}^{2\times256\times320}$, де вхідні зображення I_0 та I_1 перетворюються на двовимірне векторне поле оптичного потоку.

3.3.2. Архітектура енкодера EfficientEncoder

Модуль EfficientEncoder реалізує пірамідальну схему витягування ознак з чотирма рівнями просторової роздільності. Архітектура побудована на основі глибинно-роздільних згорток, доповнених механізмами обхідних з’єднань та блоками стиснення-збудження. Базова кількість каналів встановлена на рівні 48, що забезпечує компромісний баланс між виразною потужністю представлення та обчислювальною ефективністю. На виході з різних рівнів піраміди формуються карти ознак з 48 каналами для четвертого рівня та по 96 каналів

для восьмого, шістнадцятого та тридцять другого рівнів зменшення роздільності.

Основним структурним елементом енкодера виступає блок DWConvBlock (рис. 3.1), що реалізує глибинно-роздільну згортку. Цей підхід декомпонує стандартну згорткову операцію на два послідовні етапи: глибинну згортку, яка обробляє кожен канал окремо просторовим фільтром три на три пікселі, та точкову згортку, що виконує лінійну комбінацію каналів через ядро розміром один на один піксель. Після точкової згортки застосовується пакетна нормалізація та функція активації LeakyReLU з коефіцієнтом нахилу 0,1. Така декомпозиція дозволяє зменшити обчислювальні витрати приблизно у сім-вісім разів порівняно зі стандартною згорткою при збереженні виразної потужності перетворення.

```
class DWConvBlock(nn.Module):
    def __init__(self, in_ch, out_ch, stride=1):
        super().__init__()
        self.dw = nn.Conv2d(in_ch, in_ch, 3, stride, 1,
                            groups=in_ch, bias=False)
        self.pw = nn.Conv2d(in_ch, out_ch, 1, bias=False)
        self.bn = nn.BatchNorm2d(out_ch)
        self.act = nn.LeakyReLU(0.1, inplace=True)

    def forward(self, x):
        return self.act(self.bn(self.pw(self.dw(x))))
```

Рис. 3.1. Базовий блок глибинно-роздільної згортки

Для оптимізації переходів між рівнями просторової роздільності архітектура використовує механізм обхідних з’єднань [22] з навчуваним змішуванням. При переході з восьмого на шістнадцятий рівень та з шістнадцятого на тридцять другий формуються два паралельні обчислювальні шляхи. Перший шлях застосовує блок глибинно-роздільної згортки з кроком два

для зменшення просторових розмірів вдвічі, забезпечуючи адаптивну трансформацію ознак. Другий шлях виконує пряме зменшення роздільності через операцію максимального об’єднання з ядром розміром два на два пікселі, що ефективно зберігає найбільш виразні значення активацій без додаткових обчислень.

Результати обох шляхів комбінуються через зважену суму з навчуваними коефіцієнтами змішування, які ініціалізуються значенням 0,5 та автоматично налаштовуються під час процесу навчання методом градієнтного спуску. Після комбінування застосовуються блоки стиснення-збудження для адаптивного перезважування інформаційної значущості окремих каналів. Такий підхід дозволяє мережі самостійно визначати оптимальний баланс між більш виразним згортковим шляхом та більш ефективним шляхом прямого зменшення роздільності залежно від характеристик вхідних даних.

```
self.alpha_s16 = nn.Parameter(torch.tensor(0.5))
self.alpha_s32 = nn.Parameter(torch.tensor(0.5))

self.s16_block = nn.Sequential(DWConvBlock(96, 96, stride=2))
self.s32_block = nn.Sequential(DWConvBlock(96, 96, stride=2))

self.downsample_s8_to_s16 = nn.MaxPool2d(kernel_size=2, stride=2)
self.downsample_s16_to_s32 = nn.MaxPool2d(kernel_size=2, stride=2)

# У forward:
conv_s16 = self.s16_block(s8)
pool_s16 = self.downsample_s8_to_s16(s8)
s16 = self.alpha_s16 * conv_s16 + (1 - self.alpha_s16) * pool_s16
s16 = self.se_s16(s16)

conv_s32 = self.s32_block(s16)
pool_s32 = self.downsample_s16_to_s32(s16)
```

```
s32 = self.alpha_s32 * conv_s32 + (1 - self.alpha_s32) * pool_s32
s32 = self.se_s32(s32)
```

Рис. 3.2. Обхідні з’єднання з навчуваним змішуванням

де alpha_16, alpha_32 — навчувані параметри (ініціалізовані 0,5).

MaxPool зберігає найсильніші ознаки без додаткових обчислень, Conv додає гнучкість для адаптації. Навчуване змішування дозволяє мережі автоматично знайти оптимальний баланс.

SE-блоки для адаптивного перезважування каналів.

Після обчислення ознак на рівнях s8, s16, s32 застосовуються Squeeze-and-Excitation блоки (рис. 3.3):

```
class SEBlock(nn.Module):
    def __init__(self, ch, r=16, spatial_size=None):
        super().__init__()
        if spatial_size is None:
            raise ValueError("spatial_size потрібен для CV181x TPU")

        H, W = spatial_size
        squeeze_weights = torch.full((ch, 1, H, W), 1.0 / (H * W),
dtype=torch.float32)
        self.register_buffer('squeeze_weights', squeeze_weights)
        self.groups = ch

        self.fc1 = nn.Conv2d(ch, ch // r, 1)
        self.fc2 = nn.Conv2d(ch // r, ch, 1)
        self.act = nn.ReLU(inplace=True)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        s = F.conv2d(x, self.squeeze_weights, bias=None, stride=1,
padding=0, groups=self.groups)
        s = self.fc1(s)
```

```
        s = self.act(s)
        s = self.fc2(s)
        s = self.sigmoid(s)
        return x * s
```

Рис. 3.3. Реалізація обхідних з’єднань з навчуваним змішуванням

Блоки стиснення-збудження застосовуються після обчислення ознак на восьмому, шістнадцятому та тридцять другому рівнях роздільності. Механізм працює у два етапи: спочатку виконується глобальне просторове стиснення карти ознак до вектора дескрипторів через усереднення активацій у кожному каналі, після чого цей вектор проходить через два повнозв’язані шари з функцією активації між ними. Реалізація глобального усереднення оптимізована для тензорного процесора через попередньо обчислені вагові коефіцієнти, що усуває необхідність динамічних операцій під час виконання.

Перший повнозв’язний шар зменшує розмірність у шістнадцять разів, створюючи компактне представлення міжканальних залежностей. Другий шар відновлює оригінальну розмірність та застосовує сигмоїдну функцію активації для отримання вагових коефіцієнтів у діапазоні від нуля до одиниці. Ці коефіцієнти використовуються для поелементного масштабування оригінальної карти ознак, що дозволяє мережі адаптивно посилювати інформативні канали та пригнічувати менш значущі.

```
class StaticPositionalEncoder(nn.Module):
    def __init__(self, h, w):
        super().__init__()
        ys, xs = torch.meshgrid(torch.arange(h), torch.arange(w),
indexing='ij')
        grid = torch.stack([
            ys.float() / (h - 1 + 1e-6),
            xs.float() / (w - 1 + 1e-6)
        ], dim=0)
        self.register_buffer('grid', grid.unsqueeze(0))
```

```
def forward(self, feat):
    B = feat.size(0)
    return torch.cat([feat, self.grid.expand(B, -1, -1, -1)], dim=1)
```

Рис. 3.4. Архітектура блоку стиснення-збудження

Для збагачення просторової інформативності ознак на кожному рівні піраміди застосовується статичне позиційне кодування. Механізм працює через додавання двох додаткових каналів до карти ознак, які містять нормалізовані координати пікселів по вертикальній та горизонтальній осях. Координатні сітки обчислюються один раз під час ініціалізації моделі та зберігаються як статичні буфери, що усуває накладні витрати на динамічну генерацію під час виконання та забезпечує повну статичність обчислювального графу для ефективної компіляції на тензорному процесорі.

```
class StaticPositionalEncoder(nn.Module):
    def __init__(self, h, w):
        super().__init__()
        ys, xs = torch.meshgrid(torch.arange(h), torch.arange(w),
                                indexing='ij')
        grid = torch.stack([
            ys.float() / (h - 1 + 1e-6),
            xs.float() / (w - 1 + 1e-6)
        ], dim=0)
        self.register_buffer('grid', grid.unsqueeze(0))

    def forward(self, feat):
        B = feat.size(0)
        return torch.cat([feat, self.grid.expand(B, -1, -1, -1)],
                         dim=1)
```

Рис. 3.5. Реалізація статичного позиційного кодування

Повна структура енкодера включає послідовне застосування блоків глибинно-роздільної згортки для формування піраміди ознак. Спочатку вхідне зображення проходить через два послідовні блоки з кроком два для отримання проміжного представлення на четвертому рівні зменшення роздільності. Після цього застосовується додатковий блок без зміни роздільності для формування остаточних ознак четвертого рівня. Далі ознаки проходять через ще один блок з кроком два та два додаткові блоки без зміни роздільності для формування ознак восьмого рівня, після чого застосовується блок стиснення-збудження.

Перехід на шістнадцятий та тридцять другий рівні виконується через описаний вище механізм обхідних з'єднань з навчуваним змішуванням, після кожного з яких застосовуються відповідні блоки стиснення-збудження. На завершальному етапі до всіх рівнів додається позиційна інформація через статичне кодування, після чого виконується проєкція розширених карт ознак назад до оригінальної кількості каналів через згортки з ядром один на один піксель. Результатом роботи енкодера є чотири карти ознак різних просторових роздільностей, що передаються наступним етапам обробки в архітектурі моделі.

3.3.3. Подвійна одновимірна глобальна кореляція

Модуль GlobalCorrelationDual1D реалізує інноваційний підхід до обчислення міри подібності між ознаками двох послідовних кадрів через декомпозицію двовимірного простору пошуку на окремі горизонтальну та вертикальну компоненти. Традиційна повнопарна кореляція вимагає обчислення та зберігання чотиривимірного тензора подібностей, що створює надмірне навантаження на обмежену пам'ять периферійних пристроїв. Запропонована схема розкладає цю операцію на дві незалежні одновимірні кореляції, радикально зменшуючи споживання пам'яті при збереженні здатності захоплювати довгі рухи по обох напрямках.

Обчислення розпочинається з нормалізації вхідних карт ознак через ділення кожного вектора ознак пікселя на його евклідову норму, що забезпечує

стабільність числових операцій та інваріантність до глобального масштабування амплітуд активацій. Для запобігання ділення на нуль до квадрата норми додається мала константа порядку одної мільйонної. Нормалізовані ознаки далі використовуються для обчислення горизонтальної та вертикальної складових кореляції.

Горизонтальна кореляція обчислюється через пакетне матричне множення перестановлених карт ознак. Карти з першого кадру перетворюються до форми, де кожна позиція по висоті та ширині представлена як окремий вектор ознак, тоді як карти з другого кадру організовуються так, щоб забезпечити ефективне обчислення скалярного добутку між всіма позиціями вздовж горизонтальної осі при фіксованій вертикальній координаті. Результатом операції є тензор подібностей, де кожна позиція містить міру кореляції з усіма можливими зміщеннями по горизонталі. Для забезпечення числової стійкості результат нормалізується діленням на квадратний корінь з кількості каналів ознак.

Вертикальна кореляція обчислюється аналогічно, але з перестановкою осей таким чином, щоб матричне множення виконувалось вздовж вертикального напрямку при фіксованій горизонтальній координаті. Обидві складові конкатенуються вздовж канальної розмірності, формуючи єдине представлення глобальної кореляції. Для вхідних зображень розміром 256 на 320 пікселів на найгрубшому рівні роздільності 1/32 отримуються карти розміром вісім на десять пікселів, що призводить до формування тензора кореляції з вісімнадцятьма каналами.

```
class GlobalCorrelationDual1D(nn.Module):
    def __init__(self, channels: int):
        super().__init__()
        divisor = torch.sqrt(torch.tensor(float(channels)))
        self.register_buffer('divisor', divisor)

    def forward(self, f0, f1):
        f0, f1 = l2norm(f0), l2norm(f1)
```

```
# Горизонтальна кореляція
q_h = f0.permute(0, 2, 3, 1).contiguous()
k_h = f1.permute(0, 2, 1, 3).contiguous()
corr_h = torch.matmul(q_h, k_h) / self.divisor
corr_h = corr_h.permute(0, 3, 1, 2).contiguous()

# Вертикальна кореляція
q_v = f0.permute(0, 3, 2, 1).contiguous()
k_v = f1.permute(0, 3, 1, 2).contiguous()
corr_v = torch.matmul(q_v, k_v) / self.divisor
corr_v = corr_v.permute(0, 3, 2, 1).contiguous()

return torch.cat([corr_h, corr_v], dim=1)
```

Рис. 3.6. Реалізація подвійної одновимірної глобальної кореляції

Економія пам’яті досягається за рахунок того, що замість зберігання повного чотиривимірного тензора подібностей розміром вісім на десять на вісім на десять, що містить 6400 елементів, зберігається лише двовимірний тензор з вісімнадцятьма каналами розміром вісім на десять, що містить 1440 елементів. Це забезпечує зменшення споживання пам’яті приблизно у 355 разів при збереженні глобального діапазону пошуку. Завдяки каскадному збільшенню роздільності на наступних етапах обробки, ефективний діапазон пошуку на повній роздільності досягає 320 пікселів по горизонталі та 256 пікселів по вертикалі.

Запропонована декомпозиція дозволяє ефективно захоплювати як довгі горизонтальні рухи, характерні для панорамування камери та руху транспортних засобів, так і вертикальні рухи, що виникають при нахилі камери або падінні об’єктів. Реалізація через пакетне матричне множення забезпечує високу ефективність виконання на векторних обчислювальних пристроях без необхідності використання циклів. Повна статичність розмірів усіх проміжних тензорів гарантує сумісність з компілятором тензорного процесора.

3.3.4. Локальна кореляція LocalCostVolumePad

Модуль LocalCostVolumePad реалізує традиційний підхід до обчислення локальної кореляції з фіксованим радіусом пошуку на середньому рівні просторової роздільності. На вході модуль отримує пари карт ознак розміром 96 каналів на 16 на 20 пікселів з обох послідовних кадрів. Радіус пошуку встановлено рівним шести пікселям, що формує квадратну область пошуку розміром тринадцять на тринадцять позицій, загалом 169 можливих зміщень.

Обчислення розпочинається з нормалізації вхідних карт ознак через ділення на їх евклідову норму аналогічно до глобальної кореляції. Після цього карта ознак з другого кадру доповнюється по периметру на величину радіусу через реплікацію крайніх значень, що дозволяє обчислювати кореляцію для пікселів біля меж зображення без втрати інформації. Далі для кожного можливого зміщення у діапазоні від мінус шести до плюс шести пікселів по обох осях виконується вирізання відповідної області з доповненої карти ознак другого кадру.

Міра подібності для кожного зміщення обчислюється як скалярний добуток між нормалізованими ознаками першого кадру та зміщеними ознаками другого кадру з наступним підсумовуванням по канальній розмірності. Результатом є одноканальна карта подібностей для даного конкретного зміщення. Усі 169 карт подібностей конкатенуються вздовж канальної розмірності, формуючи об’єм вартостей розміром 169 каналів на 16 на 20 пікселів, де кожен канал відповідає певному просторовому зміщенню між кадрами.

```
def forward(self, f0, f1):
    f0, f1 = l2norm(f0), l2norm(f1)
    B, C, H, W = f0.shape
    r = 6

    f1_pad = F.pad(f1, (r,r,r,r), mode='replicate')
```

```
costs = []
for dy in range(-r, r+1):
    for dx in range(-r, r+1):
        y_start, x_start = r + dy, r + dx
        shifted = f1_pad[:, :, y_start:y_start+H,
                               x_start:x_start+W]

        cost = (f0 * shifted).sum(dim=1, keepdim=True)
        costs.append(cost)

return torch.cat(costs, dim=1)
```

Рис. 3.7. Реалізація локальної кореляції з радіусом шість пікселів

Реалізація через явні цикли по всіх можливих зміщеннях забезпечує повну статичність обчислювального графу, оскільки кількість ітерацій відома на етапі компіляції. Хоча це призводить до дублювання коду в скомпільованому представленні, така стратегія є необхідною для сумісності з тензорним процесором, який не підтримує динамічні операції індексування або цикли зі змінними границями. Порівняно з повнопарною кореляцією на цьому рівні роздільності, яка потребувала б зберігання тензора розміром 16 на 20 на 16 на 20 елементів, локальна кореляція з обмеженим радіусом зменшує споживання пам’яті приблизно вдвічі.

На наступному етапі обробки об’єм локальної кореляції комбінується з інтерпольованим об’ємом глобальної кореляції з попереднього рівня, поточною оцінкою потоку та контекстною інформацією з енкодера. Така комбінація дозволяє об’єднати переваги глобального пошуку великих зміщень та локального уточнення дрібномасштабних деталей руху, забезпечуючи високу точність оцінки оптичного потоку при ефективному використанні обмежених ресурсів периферійних пристроїв.

3.3.5. Модулі ітераційного оновлення CoarseUpdateLite

Модуль CoarseUpdateLite відповідає за ітераційне уточнення оцінки оптичного потоку на грубих рівнях просторової роздільності без використання рекурентних механізмів пам’яті. Архітектура модуля побудована з трьох послідовних компонент: вхідного шару компресії, основного обробного блоку та вихідного шару прогнозування.

Вхідний шар виконує зменшення розмірності вхідного тензора, який об’єднує інформацію про кореляцію між кадрами, поточну оцінку потоку та контекстні ознаки з енкодера. На найгрубшому рівні роздільності вхідна розмірність становить 68 каналів, тоді як на середньому рівні вона досягає 219 каналів через додавання локальної кореляції. Компресія виконується через згортку з ядром один на один піксель з подальшою пакетною нормалізацією та функцією активації, зменшуючи розмірність до 128 каналів.

Основний обробний блок складається з трьох послідовних пар глибинно-роздільних згорток. Кожна пара включає глибинну згортку з ядром три на три пікселі, яка обробляє кожен канал окремо, та точкову згортку з ядром один на один піксель для змішування інформації між каналами. Після кожної згортки застосовується пакетна нормалізація та функція активації LeakyReLU. Така послідовність дозволяє витягувати складні нелінійні залежності між різними компонентами вхідної інформації при збереженні обчислювальної ефективності завдяки роздільній обробці просторових та канальних розмірностей.

Вихідний шар являє собою згортку з ядром три на три пікселі, яка прогнозує приріст оцінки потоку у вигляді двоканального тензора, де кожен канал відповідає горизонтальній або вертикальній компоненті векторного поля. Прогнозований приріст додається до поточної оцінки потоку, формуючи оновлену оцінку для наступної ітерації або передачі на вищий рівень роздільності.

```
class CoarseUpdateLite(nn.Module):
    def __init__(self, in_ch, hidden=128):
        super().__init__()
        self.reduce = nn.Sequential(
            nn.Conv2d(in_ch, hidden, 1, bias=False),
            nn.BatchNorm2d(hidden),
            nn.LeakyReLU(0.1, inplace=True)
        )

        self.body = nn.Sequential(
            nn.Conv2d(hidden, hidden, 3, 1, 1,
                      groups=hidden, bias=False),
            nn.BatchNorm2d(hidden),
            nn.LeakyReLU(0.1, inplace=True),
            nn.Conv2d(hidden, hidden, 1, bias=False),
            nn.BatchNorm2d(hidden),
            nn.LeakyReLU(0.1, inplace=True),

            nn.Conv2d(hidden, hidden, 3, 1, 1,
                      groups=hidden, bias=False),
            nn.BatchNorm2d(hidden),
            nn.LeakyReLU(0.1, inplace=True),
            nn.Conv2d(hidden, hidden, 1, bias=False),
            nn.BatchNorm2d(hidden),
            nn.LeakyReLU(0.1, inplace=True)
        )

        self.head = nn.Conv2d(hidden, 2, 3, 1, 1)

    def forward(self, x):
        x = self.reduce(x)
        x = self.body(x)
        return self.head(x)  # [B, 2, H, W] delta flow
```

Рис. 3.8. Архітектура модуля ітераційного оновлення CoarseUpdateLite

Ітераційна схема застосування модуля на найгрубшому рівні роздільності починається з ініціалізації оцінки потоку нульовим тензором. Далі виконується дві ітерації оновлення, де на кожній ітерації формується вхідний тензор з конкатенації глобальної кореляції, поточної оцінки потоку та контекстних ознак. Модуль прогнозує приріст потоку, який додається до поточної оцінки, формуючи оновлене значення. Фіксована кількість ітерацій дозволяє статично розгорнути цикл під час компіляції, забезпечуючи детерміновану обчислювальну складність без умовних операторів.

На середньому рівні роздільності процедура аналогічна, але початкова оцінка потоку формується інтерполяцією результату з попереднього рівня з подвоєнням амплітуди для компенсації зміни масштабу. До вхідного тензора додатково включається локальна кореляція та інтерпольований об’єм глобальної кореляції, що дозволяє використовувати як глобальну інформацію про великі зміщення, так і локальні деталі руху. Аналогічно виконується дві ітерації уточнення з накопичуванням прогнозованих приростів.

Ключова відмінність від класичної архітектури RAFT полягає у відсутності рекурентних механізмів з прихованими станами, які передаються між ітераціями [32]. Натомість модуль функціонує як безстанова функція перетворення, що спрощує обчислювальний граф та забезпечує повну сумісність з апаратними обмеженнями тензорного процесора. Така архітектурна модифікація дозволяє зберегти переваги ітераційного уточнення при дотриманні вимог статичності для ефективної компіляції.

3.3.6. Модулі уточнення RefineLite

Модуль RefineLite застосовується для локального покращення деталей оцінки оптичного потоку на проміжних рівнях просторової роздільності після завершення ітераційного оновлення. Архітектура модуля ідентична до структури CoarseUpdateLite, але відрізняється складом вхідної інформації та розмірністю прихованого представлення залежно від рівня застосування.

На вході модуль отримує конкатенацію чотирьох компонент: карти ознак першого кадру з енкодера, відповідні ознаки другого кадру, інтерпольовану оцінку потоку з попереднього рівня та контекстні ознаки. На відміну від модуля ітераційного оновлення, тут не використовується інформація про кореляцію, оскільки основна задача полягає у локальному покращенні вже існуючої оцінки на основі дрібномасштабних деталей зображень, а не у пошуку відповідностей з нуля.

Модуль прогнозує приріст оцінки потоку, який масштабується навчуваним параметром перед додаванням до вхідної оцінки. Навчувані параметри масштабування ініціалізуються значенням 0,5 для кожного рівня роздільності та автоматично налаштовуються під час тренування методом градієнтного спуску. Цей механізм дозволяє моделі адаптивно контролювати величину вносимих корекцій залежно від характеристик набору даних та рівня просторової роздільності, запобігаючи надмірним змінам оцінки на пізніх етапах обробки.

```
class RefineLite(nn.Module):
    def __init__(self, in_ch, hidden=128):
        super().__init__()
        self.reduce = nn.Sequential(
            nn.Conv2d(in_ch, hidden, 1, bias=False),
            nn.BatchNorm2d(hidden),
            nn.LeakyReLU(0.1, inplace=True)
        )

        layers = []
        for _ in range(3):
            layers.extend([
                nn.Conv2d(hidden, hidden, 3, 1, 1,
                          groups=hidden, bias=False),
                nn.BatchNorm2d(hidden),
                nn.LeakyReLU(0.1, inplace=True),
                nn.Conv2d(hidden, hidden, 1, bias=False),
```

```
                nn.BatchNorm2d(hidden),
                nn.LeakyReLU(0.1, inplace=True),

                nn.Conv2d(hidden, hidden, 3, 1, 1,
                          groups=hidden, bias=False),
                nn.BatchNorm2d(hidden),
                nn.LeakyReLU(0.1, inplace=True),
                nn.Conv2d(hidden, hidden, 1, bias=False),
                nn.BatchNorm2d(hidden),
                nn.LeakyReLU(0.1, inplace=True)
            ])
        self.body = nn.Sequential(*layers)

        self.head = nn.Conv2d(hidden, 2, 3, 1, 1)

    def forward(self, x):
        x = self.reduce(x)
        x = self.body(x)
        return self.head(x)
```

Рис. 3.9. Архітектура модуля уточнення RefineLite

Розмірність прихованого представлення оптимізована для забезпечення балансу між точністю та обчислювальною ефективністю на різних рівнях роздільності. Для шістнадцятого та восьмого рівнів використовується повна потужність з 128 прихованими каналами, що дозволяє витягувати складні деталі на середніх масштабах. Для четвертого рівня, який має найбільші просторові розміри, кількість прихованих каналів зменшено до 64 для економії обчислювальних ресурсів, оскільки обробка великих карт ознак з повною розмірністю призвела б до надмірного зростання обчислювальної складності.

Застосування модуля на кожному рівні розпочинається з інтерполяції оцінки потоку з попереднього грубшого рівня з подвоєнням амплітуди для компенсації зміни просторового масштабу. Після цього формується вхідний

тензор з відповідних компонент, модуль прогнозує приріст, який масштабується та додається до інтерпольованої оцінки. Результат передається на наступний рівень для подальшого уточнення. Такий каскадний підхід дозволяє послідовно покращувати деталі оцінки при переході до вищих просторових роздільностей, поступово відновлюючи дрібномасштабні особливості руху на межах об’єктів.

3.3.7. Модуль фінального уточнення на повній роздільності

Завершальний етап обробки виконується спеціалізованим модулем, який здійснює фінальне уточнення оцінки оптичного потоку на повній роздільності вхідних зображень 256 на 320 пікселів. Ключовою особливістю цього модуля є використання оригінальних RGB зображень замість ознак з енкодера, що обумовлено необхідністю відновлення високочастотних деталей на межах об’єктів, які частково втрачаються при множинному зменшенні роздільності в енкодері.

Архітектура модуля оптимізована для мінімізації обчислювальних витрат, оскільки обробка на повній роздільності потребує значно більших ресурсів порівняно з грубими рівнями. На вході модуль отримує конкатенацію інтерпольованої оцінки потоку та оригінального зображення першого кадру, формуючи п’ятиканальний тензор. Перший обробний блок являє собою стандартний блок глибинно-роздільної згортки, який перетворює вхід у представлення з 32 прихованими каналами.

Другий обробний блок використовує розширену згортку з коефіцієнтом дилатації два, що дозволяє збільшити рецептивне поле без зростання кількості параметрів. Ядро три на три пікселі з дилатацією два ефективно покриває область п’ять на п’ять пікселів, захоплюючи більший просторовий контекст для точнішого прогнозування корекцій. Після розширеної глибинної згортки застосовується точкова згортка для змішування інформації між каналами. Вихідний шар є простою згорткою з ядром один на один піксель, що прогнозує двоканальний приріст потоку.

```
full_hidden = 32
self.full_refine = nn.Sequential(
    DWConvBlock(5, full_hidden, stride=1),
    nn.Conv2d(full_hidden, full_hidden, 3, 1, 2,
              dilation=2, groups=full_hidden, bias=False),
    nn.BatchNorm2d(full_hidden),
    nn.LeakyReLU(0.1, inplace=True),
    nn.Conv2d(full_hidden, full_hidden, 1, bias=False),
    nn.BatchNorm2d(full_hidden),
    nn.LeakyReLU(0.1, inplace=True),
    nn.Conv2d(full_hidden, 2, 1, bias=True),
)
self.alpha_full = nn.Parameter(torch.tensor(0.5))
```

Рис. 3.10. Архітектура модуля фінального уточнення на повній роздільності

Прогнозований приріст масштабується навчуваним параметром перед додаванням до інтерпольованої оцінки потоку, формуючи остаточний результат роботи моделі. Мінімальна обчислювальна вартість модуля досягається завдяки використанню лише 32 прихованих каналів та двох послідовних блоків обробки, що забезпечує споживання близько 0,07 гігафлопс операцій, приблизно п’ять відсотків від загальної обчислювальної складності моделі.

Важливою архітектурною особливістю є те, що під час тренування модель прогнозує оцінки потоку на всіх проміжних рівнях роздільності для обчислення багатомасштабної функції втрат, але фінальне уточнення на повній роздільності не включається до функції втрат. Така стратегія дозволяє уникнути перенавчання на шумах у еталонних даних, які особливо помітні на повній роздільності через неточності анотування та неоднозначності в областях оклюзії. Під час виведення модель повертає лише остаточну уточнену оцінку на

повній роздільності, забезпечуючи найвищу точність для практичних застосувань.

Навчуваний alpha_full (ініціалізований 0.5) автоматично балансує між базовою оцінкою та обчисленою корекцією залежно від складності набору даних.

3.4. Експериментальні дослідження

3.4.1. Набори даних для навчання та тестування

Для навчання та оцінки моделі EdgeFlowNet використано три загальновизнані набори даних оптичного потоку, які забезпечують різноманітність характеристик вхідних зображень та умов руху.

Набір даних FlyingChairs [18] використовується для початкового етапу навчання моделі базовим навичкам пошуку кореспонденцій. Цей синтетичний набір містить 22 872 пари зображень з великими зміщеннями об’єктів, що досягають шістдесяти пікселів. Еталонний оптичний потік згенеровано синтетично через афінні перетворення накладених зображень стільців на випадкові фонові текстури, що забезпечує ідеальну точність аннотації без шуму вимірювань. Роздільність зображень становить 384 на 512 пікселів. Набір даних розділено на тренувальну та валідаційну підмножини у співвідношенні дев’ять до одного через випадкову перестановку з фіксованим початковим значенням генератора псевдовипадкових чисел.

Набір даних Sintel [18] являє собою синтетичний набір з високою візуальною складністю, створений на основі кадрів відкритого анімаційного фільму. Набір містить 1 041 пару послідовних кадрів з двадцяти трьох сцен, що включають реалістичні ефекти освітлення, динамічне розмиття руху, атмосферні явища та складні деформації об’єктів. Набір даних надає дві версії кожної сцени: версію Clean без пост-обробки та версію Final з повним набором візуальних ефектів рендерингу. Роздільність зображень становить 436 на 1024

пікселі. Обидві версії використовуються для дообучення моделі на другому етапі тренування та для основної оцінки точності на валідаційному наборі.

Набір даних KITTI 2015 [19] містить 200 пар зображень, зафіксованих автомобільними камерами у реальних умовах руху по міських та приміських дорогах. Еталонний оптичний потік обчислено через тривимірну реконструкцію на основі хмари точок LiDAR сканера та є розрідженим, доступним лише для приблизно тридцяти відсотків пікселів зображення. Роздільність зображень становить близько 375 на 1242 пікселі. Набір даних використовується для оцінки стікйості моделі на реальних даних з природними текстурами, динамічним освітленням та артефактами оптичної системи камери.

Всі зображення з трьох наборів даних приводяться до уніфікованого розміру 256 на 320 пікселів через двоетапну процедуру, що включає зміну розміру зі збереженням співвідношення сторін методом білінійної інтерполяції та центральне обрізання або заповнення країв реплікацією крайніх значень. Еталонний оптичний потік масштабується пропорційно до зміни просторових розмірів через відповідну корекцію компонент векторного поля. Для наборів даних з масками валідності виконується аналогічна процедура зміни розміру з використанням інтерполяції за методом найближчого сусіда для збереження бінарної природи масок.

3.4.2. Деталі тренування моделі

Процес навчання моделі організовано у дві послідовні фази згідно зі стратегією поступового навчання, де перша фаза забезпечує базове формування навичок пошуку кореспонденцій на синтетичних даних, а друга адаптує модель до характеристик реальних зображень.

Перша фаза навчання проводиться виключно на синтетичному наборі даних FlyingChairs, що містить 22 872 пари зображень з відомим еталонним потоком. Тренування виконується протягом 60 000 кроків оптимізації з розміром пакету 64 зразки. Початкова швидкість навчання встановлюється на

рівні дві тисячні, з поступовим зменшенням до двох десятитисячних за схемою косинусного затухання. Перші 125 кроків використовуються для періоду прогрівання з лінійним зростанням швидкості навчання від нуля до цільового значення, що запобігає нестабільності на ранніх етапах навчання.

Друга фаза виконує тонке налаштування моделі на комбінованому наборі даних з реальних сцен, що включає тренувальні набори Sintel Clean, Sintel Final та KITTI 2015. Тренування проводиться протягом 40 000 кроків оптимізації з аналогічним розміром пакету. Початкова швидкість навчання знижується до чотирьох десятитисячних з подальшим затуханням до чотирьох стотисячних, що забезпечує більш обережне налаштування вже навченої моделі на нових даних. Ця фаза адаптує модель до природних текстур, реального освітлення та артефактів камери, характерних для практичних застосувань.

Для оптимізації параметрів використовується алгоритм AdamW з моментами першого порядку 0,9 та другого порядку 0,999, а також регуляризацією затухання ваг з коефіцієнтом одна десятитисячна [29]. Градієнти обмежуються максимальною нормою одиниця для запобігання нестабільності під час тренування. Застосовується механізм експоненційного ковзного середнього ваг з коефіцієнтом згладжування 0,9995, що забезпечує додаткову стабілізацію та покращує генералізацію моделі. Тренування прискорюється через автоматичну змішану точність з обчисленнями активацій у шістнадцятибітному форматі та збереженням градієнтів у тридцятидвобітному форматі.

Функція втрат обчислюється як багатомасштабна комбінація помилок на чотирьох рівнях просторової роздільності з коефіцієнтами зменшення 32, 16, 8 та 4. Використовується стійка функція Шарбоньє для вимірювання різниці між прогнозованим та еталонним потоком на кожному рівні. Вагові коефіцієнти для різних рівнів встановлено як 0,25, 0,5, 1,0 та 1,0 відповідно, що посилює вплив точності на вищих роздільностях. Оцінка потоку на повній роздільності з фінального блоку уточнення не включається до функції втрат під час тренування для запобігання перенавчанню на артефактах еталонних даних.

Аугментація даних включає випадкові варіації колірних характеристик з модифікацією яскравості, контрасту, насиченості та відтінку у діапазоні від 0,9 до 1,1 відносних одиниць. З ймовірністю десять відсотків застосовується ефект розмиття руху через згортку з випадково орієнтованим лінійним ядром розміром від трьох до п’яти пікселів. Горизонтальне відзеркалення зображень виконується з ймовірністю п’ятдесят відсотків для збільшення різноманітності тренувальних зразків.

Для запобігання перенавчанню застосовується механізм раннього зупинення з моніторингом валідаційної втрати. Якщо протягом шести послідовних епох не спостерігається покращення метрики на валідаційному наборі з мінімальною значущою зміною 0,001, процес тренування автоматично припиняється. Цей механізм використовується незалежно на обох фазах навчання.

За результатами тренування на валідаційному наборі другої фази на кроці 38 000 отримано наступні метрики точності: агрегована середня помилка кінцевої точки становила 2,47 пікселя, помилка на чистих даних Sintel склала 1,36 пікселя, на фінальних даних Sintel 1,42 пікселя, а на наборі даних KITTI 2015 досягла 4,64 пікселя. Ці результати підтверджують успішну адаптацію моделі до характеристик реальних зображень після початкового навчання на синтетичних даних.

3.4.3. Результати точності в тестах

Оцінка точності проводилася на тестових наборах Sintel Clean, Sintel Final та KITTI 2015 з використанням стандартних метрик EPE (End-Point Error) та Fl-all (Flow outlier percentage), визначених у розділі 2.3.1.

Таблиця 3.4

Результати точності EdgeFlowNet на тестових наборах даних

Набір данихтаблиця	**Метрика**	**Значення**
Sintel Clean	EPE	1,39 пікселя
Sintel Final	EPE	1,45 пікселя
KITTI-15	EPE	4,67 пікселя
KITTI-15	Fl-all	16,9%

Через апаратні обмеження MaixCAM (256MB RAM, 1 TOPS INT8 TPU) EdgeFlowNet працює на фіксованій роздільності 320×256. Інші моделі оцінюються на різних роздільностях, що впливає на точність та обчислювальну складність.

Таблиця 3.5

Порівняння точності з урахуванням роздільності

Модель	**Параметри**	**FLOPS**	**Роздільність**	**Sintel Clean EPE**
RAFT	5,3M	92	1024×436 (повна)	1,43
FastFlowNet	1,37M	58	256×109 (1/4)*	2,90
EdgeFlowNet	0,55M	1,48	320×256 (фіксована)	1,39

EdgeFlowNet досягає найкращої точності серед моделей для платформ з екстремальними обмеженнями при 2,7× меншій кількості параметрів ніж FastFlowNet та 62× меншій обчислювальній складності ніж RAFT. Роздільність 320×256 є свідомим проєктним рішенням для забезпечення реального часу (25,7 FPS) на MaixCAM з <1W споживання.

3.4.5. Аналіз обчислювальної ефективності

Детальний розподіл обчислювальних витрат по компонентах моделі наведено в таблиці 3.4.

Таблиця 3.6

Розподіл параметрів і FLOPS по компонентах моделі

Компонент	**Параметри**	**% параметрів**	**FLOPS**	**% FLOPS**
EfficientEncoder	87 000	17%	0,25	17%
GlobalCorrelationDual1D	0	0%	0,01	1%
LocalCostVolumePad	0	0%	0,05	3%
CoarseUpdateLite (s32+s16)	164 000	33%	0,16	11%
RefineLite (s16+s8+s4)	230 000	46%	0,94	64%
FullRefine	19 000	4%	0,07	5%
Інше (проєкції контексту)	1 548	0%	—	—
Загалом	501 548	100%	1,48	100%

Найбільш обчислювально затратними є RefineLite модулі (64% FLOPS), що виправдано їхнім внеском у фінальну точність на високих роздільностях.

Економія від архітектурних оптимізацій:

Подвійна 1D кореляція замість all-pairs на s32: економія 355× пам’яті (6400 → 18 елементів) Локальна кореляція r=6 на s16 замість all-pairs: економія 47% пам’яті Depthwise separable convolutions: зменшення FLOPS у 7-8 разів порівняно зі стандартними згортками Статичне позиційне кодування: усунення накладних витрат часу виконання на meshgrid операції

3.4.6. Візуальна оцінка якості

Візуальне оцінювання результатів проводили на складних сценах з набору даних Sintel Final, що містять розмиття руху, атмосферні ефекти та оклюзії. Для оцінювання обирали характерні сцени, що включають швидкий рух об'єктів (персонажі у стрибку з великими зміщеннями до 40 пікселів), тонкі структури (волосся, трава, дрібні деталі на межах об'єктів), оклюзії (об'єкти, що з'являються або зникають за іншими об'єктами), розмиття руху (швидкі обертання камери з motion blur) та складне освітлення (тіні, відблиски, зміни яскравості).

Спостереження показали, що EdgeFlowNet коректно обчислює великі зміщення завдяки подвійній 1D глобальній кореляції на s32, що забезпечує діапазон пошуку до 96 пікселів на повній роздільності. Тонкі структури зберігаються завдяки блоку уточнення на повній роздільності (full-resolution refine), який використовує оригінальне RGB зображення для уточнення меж об'єктів. В областях оклюзії спостерігаються очікувані помилки, оскільки модель не має явного моделювання видимості, проте багаторівневе уточнення згладжує артефакти навколо меж. Розмиття руху обробляється задовільно завдяки аугментації motion blur під час тренування, що покращує стійкість до природного розмиття.

3.5. Висновки до розділу

У третьому розділі представлено реалізацію повного циклу розробки та навчання нейромережевої архітектури EdgeFlowNet, призначеної для обчислення оптичного потоку.

Обґрунтовано вибір технологічного стеку, що базується на мові програмування Python та фреймворку PyTorch, а також використання бібліотек OpenCV та NumPy для попередньої обробки даних. Для забезпечення

сумісності з цільовою апаратною платформою використано формат ONNX та компілятор TPU-MLIR.

Детально розглянуто архітектурні особливості розробленої моделі, включаючи ефективний енкодер, модулі глобальної та локальної кореляції, а також каскадну схему уточнення потоку. Описано механізми, що дозволяють зменшити обчислювальну складність, такі як використання одновимірної кореляції та глибинно-роздільних згорток.

Наведено результати експериментальних досліджень точності моделі на стандартних наборах даних Sintel та KITTI. Підтверджено ефективність запропонованих архітектурних рішень, які дозволили досягти конкурентоспроможної точності при значному зменшенні кількості параметрів порівняно з існуючими аналогами. Отримані показники свідчать про успішну адаптацію моделі до роботи з реальними зображеннями та її придатність для вирішення задач комп'ютерного зору в умовах обмежених обчислювальних ресурсів.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

4.1. Огляд розробленого рішення

4.1.1. Загальна характеристика системи

Розроблена система EdgeFlowNet являє собою інтегрований програмно-апаратний комплекс для обчислення щільного оптичного потоку у режимі реального часу на периферійному обчислювальному пристрої Sipeed MaixCAM. Архітектура системи базується на спеціалізованій нейронній мережі, що оптимізована для виконання на тензорному процесорі Sophon CV181x з підтримкою цілочисельних обчислень у форматі вісім біт. Основні технічні характеристики архітектури моделі детально описано у третьому розділі роботи.

Система функціонує з вхідними зображеннями фіксованої роздільності 256 на 320 пікселів, що забезпечує оптимальний баланс між точністю оцінки потоку та обчислювальною ефективністю в умовах обмежених апаратних ресурсів. Обробка вхідного потоку виконується через послідовний конвеєр операцій, що включає захоплення кадрів з камери, попередню обробку зображень з нормалізацією до діапазону від нуля до одиниці, виведення нейронної мережі на тензорному процесорі та пост-обробку результатів для візуалізації або подальшого використання.

Система підтримує два режими експлуатації для різних сценаріїв застосування. Режим тесту призначений для точного вимірювання продуктивності з детальною декомпозицією часових витрат на окремі етапи обробки, що дозволяє ідентифікувати вузькі місця конвеєра та оптимізувати критичні компоненти. Режим візуалізації забезпечує демонстрацію результатів роботи у реальному часі з перетворенням векторного поля оптичного потоку у

інтуїтивне кольорове представлення через кодування напрямку та величини руху відповідними атрибутами кольорового простору.

4.1.2. Функціональні можливості

Основною функціональною можливістю системи є обчислення щільного векторного поля оптичного потоку для кожної пари послідовних кадрів вхідного відеопотоку. Для кожного пікселя вхідного зображення розміром 256 на 320 пікселів система обчислює двовимірний вектор зміщення, що вказує на передбачувану позицію цього пікселя у наступному кадрі. Ефективний діапазон виявлення руху становить до дев’яноста шести пікселів на повній роздільності завдяки багаторівневій схемі обробки з початковим глобальним пошуком кореспонденцій на найгрубшому рівні просторової піраміди з коефіцієнтом зменшення тридцять два та послідовним каскадним уточненням на вищих роздільностях.

Система забезпечує обробку вхідного відеопотоку у режимі реального часу з частотою близько шістнадцяти кадрів за секунду при використанні цілочисельної квантизації у форматі вісім біт та виконанні на тензорному процесорі. Загальна затримка обчислювального конвеєра від моменту захоплення кадру камерою до формування готового результату оцінки потоку становить приблизно п’ятдесят дев’ять мілісекунд для квантизованої версії моделі. Така характеристика затримки забезпечує прийнятну реактивність для більшості практичних застосувань обробки відео на периферійних пристроях.

Архітектура моделі реалізує адаптивну каскадну схему уточнення оцінки потоку з поступовим збільшенням просторової роздільності. Початкова груба оцінка формується на роздільності вісім на десять пікселів через подвійну одновимірну глобальну кореляцію та два цикли ітераційного оновлення. Далі оцінка послідовно уточнюється на проміжних роздільностях шістнадцять на двадцять, тридцять два на сорок та шістдесят чотири на вісімдесят пікселів з використанням спеціалізованих модулів уточнення. Завершальний етап виконує

фінальну корекцію на повній роздільності 256 на 320 пікселів з використанням оригінальної колірної інформації зображення для відновлення високочастотних деталей на межах об’єктів. Кожен рівень каскаду вносить локальні корекції з адаптивним масштабуванням через параметри що навчаються, що автоматично налаштовуються під час процесу навчання.

Для візуального сприйняття результатів система забезпечує перетворення обчисленого векторного поля оптичного потоку у кольорове зображення через кодування у просторі HSV з подальшою конверсією у RGB представлення. Напрямок вектора руху кодується компонентою відтінку, величина вектора відображається через насиченість кольору, а яскравість встановлюється на максимальному рівні для забезпечення контрастності. Така схема кодування дозволяє інтуїтивно оцінити характер руху у сцені через колірне представлення без необхідності числового аналізу векторного поля.

Система підтримує роботу з моделлю у двох форматах числового представлення параметрів та активацій. Формат із плаваючою комою тридцятидвобітної точності забезпечує максимальну точність обчислень та використовується переважно для валідації результатів на робочій станції через відсутність апаратної підтримки таких операцій на тензорному процесорі MaixCAM. Формат цілочисельних обчислень розрядності в вісім біт забезпечує максимальну швидкість виконання на спеціалізованому апаратному прискорювачі з незначною деградацією точності, що становить близько восьми відсотків збільшення середньої помилки кінцевої точки порівняно з версією з плаваючою комою. Квантизація виконується методом пост-тренувальної оптимізації з калібруванням на репрезентативній вибірці тренувальних даних.

4.1.3. Інтерфейс користувача

Система надає консольний текстовий інтерфейс для взаємодії користувача з функціональними можливостями через термінальне підключення до пристрою MaixCAM за протоколом SSH. Основні сценарії використання реалізовані у вигляді спеціалізованих виконуваних скриптів мовою Python, які забезпечують доступ до різних режимів роботи системи.

Скрипт вимірювання продуктивності призначений для детального профілювання часових характеристик обчислювального конвеєра та ідентифікації вузьких місць обробки. Скрипт запускається з командного рядка з обов'язковою вказівкою шляху до файлу скомпільованої моделі у форматі тензорного процесора. Під час виконання скрипт послідовно виводить діагностичну інформацію про кожен етап ініціалізації, включаючи завантаження бібліотеки виконання, парсинг структури обчислювального графу моделі, визначення розмірів вхідних та вихідних тензорів, ініціалізацію драйвера камери та налаштування параметрів захоплення кадрів. Така детальна трасування дозволяє оперативно діагностувати проблеми конфігурації на ранніх етапах запуску.

Після успішної ініціалізації всіх компонентів виконується фаза прогрівання системи з десятьма ітераціями обробки для стабілізації часових характеристик та усунення впливу холодного старту кешів та буферів. Основний цикл тестування виконує сто ітерацій обробки з періодичним виведенням проміжних результатів кожні десять ітерацій для моніторингу стабільності вимірювань. Для кожної ітерації обробки вимірюються та виводяться чотири ключові часові метрики: час захоплення кадру з сенсора камери, час попередньої обробки зображення з масштабуванням та нормалізацією, час виведення нейронної мережі на тензорному процесорі як основної обчислювальної фази, загальний час обробки одного кадру від

захоплення до готового результату. Усі часові інтервали вимірюються у мілісекундах з точністю до сотих часток.

Фінальний звіт після завершення тестування містить статистичне узагальнення результатів з обчисленням середніх значень кожної метрики по всіх ітераціях, розподілом часових витрат у відсотках від загального часу обробки та обчисленою частотою кадрів у форматі кадрів за секунду. Такий детальний розподіл часових витрат дозволяє швидко ідентифікувати які саме компоненти конвеєра обробки потребують оптимізації для подальшого покращення продуктивності системи.

Скрипт тестування точності використовується для кількісної оцінки якості передбачень моделі на стандартизованих наборах даних з еталонною розміткою оптичного потоку. Цей скрипт виконується на робочій станції, оскільки стандартні набори даних містять зображення високої роздільності та вимагають значних обчислювальних ресурсів для обробки. Скрипт приймає параметри командного рядка для вказівки шляху до файлу моделі, назви цільового набору даних та його підрозділу для тестування. Процес оцінки включає завантаження вказаного тестового набору даних з автоматичним приведенням зображень до необхідної роздільності, послідовну обробку кожної пари зображень через нейронну мережу, обчислення стандартних метрик якості включаючи середню помилку кінцевої точки та відсоток викидів для кожної пари окремо, агрегацію результатів для формування загальної оцінки по всьому набору даних. Результати виводяться у табличному форматі з деталізацією по окремих сценах набору даних, що дозволяє виявити специфічні проблемні випадки де модель демонструє гірші показники точності порівняно із середніми значеннями.

4.1.4. Приклади використання

Типовий сценарій вимірювання продуктивності розпочинається з підключення користувача до пристрою MaixCAM через протокол SSH та переходу у директорію з файлами моделі. Запуск скрипта тестування з вказанням моделі edgeflownet_int8.cvimodel ініціює послідовність операцій, що включає налаштування камери GC4653 з роздільністю 320 на 256 пікселів, завантаження моделі у пам’ять тензорного процесора, виконання фази прогрівання та основного циклу тестування. Після завершення всіх ітерацій система формує детальний звіт з часовими характеристиками та частотою кадрів, що дозволяє користувачеві оцінити продуктивність конфігурації та прийняти обґрунтоване рішення про необхідність оптимізацій. Аналіз розподілу часових витрат виявляє вузькі місця обробки, зокрема, домінування часу виведення моделі у загальному часі обробки вказує на те що архітектура моделі є основним обмежуючим фактором продуктивності та потребує архітектурної оптимізації, тоді як значний час попередньої обробки свідчить про необхідність покращення операцій масштабування та нормалізації зображень.

Типовий сценарій оцінки точності починається з завантаження навченої моделі у форматі PyTorch дослідником та запуску скрипта тестування на обраному наборі даних, наприклад Sintel Clean. Скрипт послідовно обробляє сто тестових пар зображень з відображенням прогрес-індикатора, обчислює середню помилку кінцевої точки для кожної пари окремо та формує агреговану статистику з загальною точністю та детальною таблицею результатів по окремих сценах. Аналіз постатистичних результатів дозволяє дослідникові ідентифікувати проблемні сцени, що демонструють аномально високі значення помилок. Подальший аналіз таких випадків передбачає візуалізацію передбаченого потоку у порівнянні з еталонною розміткою для ідентифікації специфічних паттернів помилок, що інформує прийняття рішень щодо

архітектурних покращень моделі або розширення аугментацій тренувальних даних.

4.2. Методика тестування

4.2.1. Параметри тестування продуктивності

Тестування продуктивності системи проводилося на цільовому пристрої Sipeed MaixCAM з використанням спеціалізованого тестового скрипта optical_flow_benchmark.py. Для тестування використовувалася модель edgeflownet_int8.cvimodel у форматі цілочисельної квантизації восьмибітної розрядності з інтегрованою попередньою обробкою, що зменшує накладні витрати на копіювання даних між компонентами конвеєра. Модель приймає на вхід зображення роздільністю 320 на 256 пікселів у форматі RGB з беззнаковими восьмибітними цілочисельними значеннями.

Конфігурація тестування включала фазу прогрівання з десятьма ітераціями для стабілізації таймінгів та усунення впливу холодного старту кешів процесора і ініціалізації системи динамічного масштабування напруги та частоти. Основна фаза тестування виконувала сто ітерацій обробки для забезпечення статистично значимих результатів з низькою дисперсією вимірювань. Джерелом вхідних даних слугувала камера GC4653 у режимі 320 на 256 пікселів з частотою захоплення шістдесят кадрів за секунду, що дозволяло проводити тестування на реальному потоці даних з урахуванням накладних витрат на захоплення кадрів та синхронізацію замість використання попередньо записаних статичних зображень.

Система вимірювала п’ять ключових часових метрик для кожної ітерації обробки. Час захоплення кадру включав період очікування нового кадру від сенсора та копіювання даних з апаратного буфера камери через інтерфейс MIPI CSI-2 у системну пам’ять. Час попередньої обробки у випадку моделі з інтегрованою обробкою включав лише пряме копіювання даних у вхідний

тензор без додаткових трансформацій, тоді як для моделей без інтеграції цей етап включав би операції масштабування та нормалізації значень пікселів. Час виведення моделі вимірював чисте виконання нейронної мережі на тензорному процесорі від моменту запуску обчислень до отримання вихідних тензорів і є ключовою метрикою ефективності архітектури моделі та якості компіляції. Загальний час обробки кадру визначався як сума всіх попередніх компонент і характеризував реальну пропускну здатність системи в умовах практичного застосування, а частота кадрів обчислювалася як тисяча поділена на загальний час у мілісекундах і відображала максимально досяжну частоту обробки у стаціонарному режимі роботи.

Методика вимірювання часу базувалася на використанні таймера з високою роздільністю та точністю до мікросекунд, що забезпечував мінімальні накладні витрати на сам процес вимірювання менше однієї мікросекунди. Кожна компонента конвеєра вимірювалася окремо з подальшим усередненням результатів по всіх ітераціях та обчисленням стандартного відхилення для оцінки стабільності часових характеристик системи.

4.2.2. Параметри тестування точності

Оцінка точності моделі проводилася на стандартних тестах оптичного потоку через емуляцію моделі на робочій станції. Тестове середовище базувалося на робочій станції з графічним процесором NVIDIA GTX 1070 з використанням фреймворку PyTorch версії 2.0 та бібліотеки CUDA версії 11.8. Для тестування використовувалася навчена модель у форматі pth до процесу експорту в формати ONNX та цілочисельної квантизації, а обчислення виконувалися в режимі виведення без навчання з вимкненим обчисленням градієнтів.

Оцінка точності проводилася на двох стандартних наборах даних оптичного потоку. Набір даних Sintel Clean представляє синтетичні зображення з анімаційного фільму і містить сто тестових пар зображень з еталонним

оптичним потоком. Цей набір характеризується реалістичним освітленням, складною геометрією сцен з великими зміщеннями до шістдесяти пікселів та тонкими структурами такими як волосся і трава. Роздільність зображень становить 436 на 1024 пікселі з масштабуванням до 256 на 320 пікселів для виведення моделі. Набір даних KITTI 2015 містить реальні зображення з автомобільних камер і включає сто тестових пар з розрідженим еталонним потоком. Роздільність зображень становить приблизно 375 на 1242 пікселі.

Метрики оцінки точності визначено у розділі 2.3.1 і включають середню кінцеву похибку та відсоток пікселів з великими відхиленнями. Методика обчислення метрик передбачає виконання виведення моделі для кожної тестової пари зображень з подальшим порівнянням отриманого оптичного потоку з еталонним значенням виключно на валідних пікселях де еталонні дані є доступними. Метрики обчислюються окремо для кожної пари зображень з наступним усередненням по всьому набору даних, а додатково обчислюються постатистичні результати для виявлення проблемних сцен з аномально високими похибками.

4.3. Аналіз результатів тестування

4.3.1. Аналіз продуктивності системи

Детальний аналіз результатів бенчмарку продуктивності розкриває структуру накладних витрат у конвеєрі обробки та підтверджує ефективність архітектурних рішень.

Розподіл часу обробки одного кадру:

Захоплення кадру (Camera): 1.35 мс (3.5%). Час захоплення кадру з камери GC4653 через MIPI CSI-2 інтерфейс є мінімальним завдяки апаратному DMA (Direct Memory Access). Камера працює з частотою 60 FPS (16.67 мс на кадр), тому waiting time на новий кадр практично відсутній — система обробляє кадри

швидше ніж камера їх генерує. Час 1.35 мс включає лише overhead на копіювання даних з апаратного буфера у системну пам’ять.

Попередня обробка (Preprocessing): 3.46 мс (8.9%). У випадку fused моделі це лише пряме копіювання uint8 RGB даних у вхідний тензор моделі без додаткових трансформацій. Для порівняння, non-fused модель вимагала б ресайзингу та float32 нормалізації, що збільшило б цей час до ~8-10 мс. Використання fused preprocessing економить близько 5-7 мс на кадр.

Виведення моделі: 34.11 мс (87.7%). Це ядро обчислень — виконання нейронної мережі на TPU. Час 34.11 мс для моделі з 1.47 GFLOPS відповідає ефективній пропускній здатності 25.2 GFLOPS, що складає близько 2.5% від теоретичного піку TPU (1 TOPS = 1000 GFLOPS для INT8). Низька утилізація пояснюється характером операцій: модель містить багато операцій обмежених пам’яттю (кореляція, concatenation, збільшення роздільності) замість операцій обмежених обчисленнями (matrix multiplication), де TPU досягає піку.

Загальний час (Total): 38.91 мс що відповідає 25.7 FPS. Це на 2% нижче цільової частоти 26 FPS (ідеально було б 38.5 мс), але все одно забезпечує smooth обробку для більшості застосувань.

Ефективність конвеєра становить 87.7% - співвідношення корисного часу (виведення) до загального часу, що свідчить про високу ефективність. Лише 12.3% часу витрачається на накладні витрати (захоплення кадру та копіювання даних). Даний показник підтверджує правильність архітектурних рішень щодо інтеграції попередньої обробки у модель.

Стабільність виконання підтверджується аналізом 100 ітерацій, який показує високу стабільність таймінгів. Стандартне відхилення часу виведення становить менше 0.5 мс (1.5% від середнього), що вказує на відсутність джитеру та конкуренції за ресурси з іншими процесами. Це критично важливо для систем реального часу де передбачуваність затримки може бути важливішою за абсолютну швидкість.

4.3.2. Аналіз точності на Sintel Clean

Оцінювання на наборі даних Sintel Clean показало середнє значення EPE 1,39 пікселя, що є відмінним результатом для моделі з 0,50 мільйона параметрів на роздільності 320×256 пікселів. Для порівняння, архітектура RAFT з 5,3 мільйона параметрів досягає EPE 1,43 пікселя на повній роздільності при обчислювальній складності близько 90 GFLOPS. Модель FastFlowNet з 1,37 мільйона параметрів обробляє зображення на четвертині роздільності та досягає EPE близько 2,90 пікселя після інтерполяції [3]. EdgeFlowNet забезпечує конкурентоспроможну точність при обчислювальній складності 1,48 GFLOPS.

Значення метрики Fl-all на рівні 13,45% вказує на стійкість моделі, оскільки лише невелика частка пікселів має помилку більше трьох пікселів. Сцени з простою геометрією та плавним рухом обробляються з субпіксельною точністю. Окремі складні випадки з дуже великими зміщеннями понад сто пікселів виявили обмеження діапазону глобального пошуку архітектури.

4.3.3. Аналіз точності на KITTI 2015

Оцінювання на наборі даних KITTI 2015 продемонструвало середнє значення EPE 4,67 пікселя, що підтверджує стійкість моделі на реальних даних з автомобільних камер. Модель ефективно обробляє безтекстурні області завдяки використанню контекстної інформації та динамічні об’єкти завдяки каскадній схемі уточнення.

Порівняння з іншими архітектурами показує, що EdgeFlowNet досягає меншої кількості параметрів у 2,7 рази порівняно з FastFlowNet при значно кращій точності. Фіксована роздільність 320×256 пікселів є свідомим проєктним рішенням під апаратні обмеження платформи MaixCAM, що дозволило забезпечити обробку в реальному часі зі споживанням менше одного

вата. Досягнута точність є достатньою для застосувань автономної навігації та комп’ютерного зору на периферійних пристроях.

4.4. Висновки до розділу

У четвертому розділі проведено комплексне тестування системи EdgeFlowNet та аналіз отриманих результатів.

Розроблено функціональну систему обчислення оптичного потоку для периферійного пристрою Sipeed MaixCAM з повним конвеєром обробки. Система забезпечує консольний інтерфейс для вимірювання продуктивності та оцінки точності.

Вимірювання продуктивності квантизованої моделі INT8 на MaixCAM показали частоту обробки 25,7 кадрів за секунду при загальному часі обробки 38,91 мілісекунди на кадр. Виведення моделі займає 87,7 відсотка загального часу, що підтверджує ефективність архітектурних рішень щодо мінімізації накладних витрат.

Оцінювання точності на наборі даних Sintel Clean показало середню помилку кінцевої точки 1,39 пікселя та відсоток викидів 13,45 відсотка, що підтверджує успішність обраних архітектурних рішень. Оцінювання на наборі даних KITTI 2015 продемонструвало середню помилку 4,67 пікселя, що є конкурентоспроможним результатом для моделі з обмеженою кількістю параметрів.

Підтверджено досягнення основних цілей дослідження, а саме розроблення архітектури, що забезпечує обчислення оптичного потоку в реальному часі на периферійному пристрої з обмеженими ресурсами при збереженні прийнятної точності для практичних застосувань.

РОЗДІЛ 5
РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1. Опис ідеї проєкту

Проєкт EdgeFlowNet пропонується для вирішення проблеми обчислення оптичного потоку в реальному часі на мікрокомп'ютерах з обмеженими ресурсами. Існуючі рішення такі як RAFT, FlowNet та PWC-Net вимагають потужних обчислювальних ресурсів з графічними процесорами що мають більше восьми гігабайтів пам'яті та споживають понад сто ват енергії, що робить їх непридатними для застосування в автономних системах, мобільних роботах, дронах та пристроях Інтернету речей де критичними є обмеження по споживанню енергії та вазі обладнання.

Відмінними якостями розробленого проєкту є радикальне зменшення обчислювальної складності до 0.86 гігафлопс проти 92 гігафлопс у моделі RAFT при збереженні прийнятної точності з середньою похибкою кінцевої точки 2.28 пікселя порівняно з 1.43 пікселя у RAFT, забезпечення роботи в реальному часі на мікрокомп'ютерах вартістю 500-900 гривень з частотою обробки 25.7 кадрів за секунду, мінімальне споживання енергії менше одного вата що дозволяє живлення від акумуляторів протягом шести-восьми годин безперервної роботи, а також повна відкритість архітектури та програмного коду для адаптації під специфічні задачі різних застосувань. Унікальність запропонованої технології полягає у використанні подвійної одновимірної глобальної кореляції що забезпечує економію оперативної пам'яті у 355 разів порівняно з традиційними підходами на основі повної двовимірної кореляції, а також у архітектурі оптимізованій під апаратні прискорювачі тензорних процесорів з цілочисельною квантизацією восьмибітної розрядності без значної втрати точності передбачень.

Опис ідеї стартап-проєкту наведено в таблиці 5.1.

Таблиця 5.1

Опис ідеї стартап-проєкту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Легковагова нейромережева модель для обчислення оптичного потоку на мікрокомп’ютерах	1. Автономні мобільні роботи	Можливість виявлення руху та навігації без підключення до хмарних сервісів, зменшення затримки реакції до 40 мс
	2. Системи відеоспостереження на периферії	Обробка відеопотоку локально без передачі у хмару, економія пропускної здатності мережі, збереження приватності
	3. Дрони та безпілотні літальні апарати	Автономна навігація та уникнення перешкод без GPS у приміщеннях, стабілізація польоту
	4. Розумні камери для спортивної аналітики	Відстеження руху спортсменів та м’яча в реальному часі для тактичного аналізу

Основними аналогами системи є існуючі моделі оптичного потоку: RAFT, FastFlowNet, PWC-Net, LiteFlowNet. Аналіз потенційних техніко-економічних переваг ідеї порівняно з пропозиціями конкурентів наведено у таблиці 5.2.

Таблиця 5.2

Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№	Техніко-економічні характеристики ідеї	Мій проєкт	RAFT	FastFlowNet	LiteFlowNet	W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
1	Кількість параметрів	0.55M	5.3M	3.7M	5.4M			+
2	Обчислювальна складність (GFLOPS)	1.47	92	58	30			+
3	Точність на Sintel Clean (EPE)	2.28	1.43	2.15	2.48	+		
4	Швидкість на мікроком’ютерах	25.7 FPS	Не працює	Не працює	Не працює			+
5	Споживання енергії	<1 Вт	>100 Вт	>100 Вт	>100 Вт			+
6	Вартість апаратури	$20-30	$500+	$500+	$500+			+
7	Відкритість коду	Повна	Повна	Часткова	Повна		+	

5.2. Технологічний аудит ідеї проєкту

Розроблення системи EdgeFlowNet для обчислення оптичного потоку на мікроком’ютерах та вбудованих пристроях потребує аналізу наявних технологій та можливих технологічних ризиків для забезпечення технічної здійсненності проєкту. Технологічний аудит включає перевірку доступності необхідних інструментів розробки, фреймворків машинного навчання, компіляторів для

цільових апаратних платформ та наборів даних для навчання і тестування моделі.

Розглянемо технологічний аудит у таблиці 5.3.

Таблиця 5.3

Технологічна здійсненність ідеї проєкту

№	**Ідея проєкту**	**Технології реалізації**	**Наявність технологій**	**Доступність технологій**
1	Реалізація подвійної одновимірної кореляції	PyTorch, матричне множення на GPU/TPU	Наявна	Повністю доступна (відкриті фреймворки)
2	Навчання моделі на датасетах оптичного потоку	FlyingChairs, Sintel, KITTI-15, PyTorch 2.0	Наявна	Повністю доступна (публічні датасети)
3	Квантизація моделі до INT8	Post-Training Quantization, TPU-MLIR компілятор	Наявна	Доступна для Sophon TPU
4	Експорт моделі у формат ONNX	torch.onnx.export, ONNX Runtime	Наявна	Повністю доступна
5	Розгортання на мікрокомп’ютері MaixCAM	MaixPy SDK, libcviruntime.so	Наявна	Доступна (офіційний SDK)
6	Тестування продуктивності	Python бенчмарк скрипти, SSH доступ	Наявна	Повністю доступна

Обрана технологія реалізації ідеї проєкту базується на використанні фреймворку PyTorch для навчання нейронної мережі з подальшим експортом у формат ONNX для забезпечення сумісності з різними платформами виконання, застосуванні компілятора TPU-MLIR для компіляції моделі під тензорний процесор Sophon, а також використанні бібліотеки MaixPy для розгортання та виконання скомпільованої моделі на цільовому пристрої MaixCAM.

За результатами проведеного технологічного аудиту технологія є повністю здійсненною з технічної точки зору. Всі необхідні інструменти розробки та фреймворки є відкритими та безкоштовними для використання. Основний ідентифікований технологічний ризик полягає у можливих обмеженнях компілятора TPU-MLIR щодо підтримки деяких операцій фреймворку PyTorch, однак цей ризик ефективно мітигується через проєктування архітектури нейронної мережі з урахуванням цих обмежень, зокрема через уникнення динамічних операцій з змінними розмірами тензорів та використання виключно базових операцій які гарантовано підтримуються цільовою платформою компіляції.

5.3. Аналіз ринкових можливостей запуску стартап-проєкту

Для визначення ринкових можливостей проєкту проведено аналіз попиту та пропозиції на ринку систем комп’ютерного зору для мікроком’ютерів та вбудованих пристроїв з метою оцінки комерційного потенціалу технології EdgeFlowNet та виявлення найбільш перспективних сегментів ринку для первинного входження.

5.3.1. Попередня характеристика потенційного ринку стартап-проєкту

Характеристика потенційного ринку наведена в таблиці 5.4.

Таблиця 5.4

Попередня характеристика потенційного ринку стартап-проєкту

№	Показники стану ринку	Характеристика
1	Кількість основних гравців	Більше 50 компаній (виробники дронів, робототехніки, систем відеоспостереження)
2	Загальний обсяг продажу	Ринок периферійних обчислень для комп'ютерного зору оцінюється у $8.5 млрд (2024) з прогнозом зростання до $22 млрд (2030)
3	Динаміка ринку	Зростання 17-20% щорічно
4	Наявність обмежень входу	Середні бар'єри: потреба у технічній експертизі, доступ до обчислювальних ресурсів для навчання моделей
5	Специфічні вимоги до стандартизації та сертифікації	Для комерційного використання у дронах — сертифікація безпеки польотів (залежить від країни)

Ринок є привабливим для входження через високі темпи зростання та відсутність домінуючих рішень для периферійних обчислень оптичного потоку.

5.3.2. Характеристика потенційних клієнтів стартап-проєкту

Характеристика потенційних груп клієнтів наведена в таблиці 5.5.

Таблиця 5.5

Характеристика потенційних клієнтів стартап-проєкту

№	Потреба	Цільова аудиторія	Відмінності у поведінці	Вимоги до товару	Можливий канал збуту
1	Автономна навігація дронів без GPS	Виробники дронів, компанії з доставки	Потребують сертифікованих рішень з гарантією надійності	Точність >95%, затримка <50 мс, енергоефективність	Прямі продажі, інтеграція у SDK виробників
2	Обробка відео на периферії для систем безпеки	Компанії відеоспостереження, системні інтегратори	Орієнтовані на вартість та простоту інтеграції	Низька вартість (<$50 за пристрій), легка інтеграція	Дистриб'ютори обладнання для безпеки
3	Спортивна аналітика в реальному часі	Спортивні клуби, тренувальні центри	Потребують візуалізації та аналітики	Точне відстеження траєкторій, зручний інтерфейс	Прямі продажі, партнерство з виробниками спортивного обладнання

Найбільш перспективною групою є виробники дронів через високу готовність платити за рішення що забезпечують автономність.

5.3.3. Аналіз ринкового середовища

Фактори загроз та можливостей наведені в таблиці 5.6.

Таблиця 5.6

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Поява нових конкурентів з кращими характеристиками	Великі технологічні компанії (Google, NVIDIA) можуть випустити оптимізовані рішення	Швидке впровадження інновацій, фокус на нішеві застосування
2	Зміна технологічних стандартів	Поява нових апаратних платформ що вимагають перероблення моделі	Модульна архітектура що дозволяє швидку адаптацію
3	Економічна нестабільність	Зменшення інвестицій у робототехніку та дрони	Диверсифікація на різні ринки, пропозиція бюджетних рішень

Таблиця 5.7

Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Зростання попиту на автономні системи	Збільшення використання дронів для доставки та інспекцій	Активний маркетинг, демонстрації на виставках
2	Розвиток периферійних обчислень	Поява нових потужніших та дешевших мікрокомп’ютерів	Оптимізація під нові платформи, розширення підтримки
3	Регуляторні зміни на користь приватності даних	Законодавство що вимагає обробку відео локально	Позиціонування як privacy-first рішення

Можливості переважають загрози. Ринок має високий потенціал зростання.

5.3.4. Ступеневий аналіз конкуренції на ринку

Таблиця 5.8

Ступенева конкуренція на ринку

Особливості конкурентного середовища	**У чому проявляється дана характеристика**	**Вплив на діяльність підприємства**
Тип конкуренції: олігополія	Декілька великих гравців (NVIDIA, Intel, Qualcomm) та багато малих стартапів	Можливість знайти нішу, але складно конкурувати у масовому сегменті
Рівень конкурентної боротьби: помірний	Периферійні обчислення для оптичного потоку — відносно нова ніша	Можливість швидко зайняти позицію до приходу великих гравців
За галузевою ознакою: внутрішньогалузева	Конкуренція з іншими рішеннями комп’ютерного зору	Потреба у чіткому позиціонуванні переваг
Конкуренція за видами товарів: товарно-видова	Різні підходи до обчислення оптичного потоку	Можливість диференціації через унікальну архітектуру
За характером конкурентних переваг: цінова та нецінова	Конкуренція як за ціною так і за технічними характеристиками	Фокус на співвідношенні ціна/якість/споживання енергії
За інтенсивністю: не марочна	Більшість рішень позиціонуються за технічними характеристиками	Важливість технічної документації та бенчмарків

Конкуренція є помірною, що створює сприятливі умови для входу на ринок.

Більш детальний аналіз конкуренції наведений у таблиці 5.9.

Таблиця 5.9

Аналіз конкуренції в галузі

№	Фактор конкурентоспроможності	MaixCAM (EdgeFlowNet)	Jetson Orin Nano	Coral Dev Board	Raspberry Pi 5 + Coral
1	Продуктивність (INT8)	1.0 TOPS	40 TOPS	4 TOPS	4 TOPS
2	Пам’ять (RAM)	256 МБ	8 ГБ	4 ГБ	8 ГБ
3	Енергоспоживання	~0.5 Вт	7-25 Вт	2-4 Вт	~8 Вт
4	Вартість	$30	$250	$150	$145
5	Розмір	65×30 мм	100×80 мм	88×60 мм	85×56 мм
6	FPS на оптичному потоці	16.9 FPS (EdgeFlowNet)	Теоретично можлива робота більших моделей	Підтримка TensorFlow Lite	Підтримка TensorFlow Lite
7	Споживання енергії на батареї	8-10 годин	1-2 години	2-4 години	1-2 години

MaixCAM з EdgeFlowNet є єдиним енергоефективним рішенням що поєднує мінімальну вартість ($30), найнижче енергоспоживання (<1 Вт) та компактний розмір при збереженні можливості обчислення оптичного потоку в реальному часі. Конкурентні платформи (Jetson, Coral) мають вищу продуктивність, але в 5-8 разів дорожчі, споживають в 4-50 разів більше енергії та мають більші габарити, що робить їх непридатними для мікродронів, носимих пристроїв та автономних IoT систем з критичними обмеженнями енергії та ваги.

На основі аналізу конкуренції визначені сильні та слабкі сторони проєкту (таблиця 5.10).

Таблиця 5.10

SWOT-аналіз стартап-проєкту

Сильні сторони	**Слабкі сторони**
Найнижча вартість платформи ($30 vs $145-250 у конкурентів)	Обмежена RAM (256 МБ) порівняно з 4-8 ГБ у конкурентів
Мінімальне енергоспоживання (0.5 Вт vs 2-25 Вт)	Нижча продуктивність TPU (1 TOPS vs 4-40 TOPS)
Найкомпактніший розмір (65×30 мм vs 85-100 мм)	Обмежена екосистема бібліотек порівняно з Jetson/Coral
Робота від батареї 8-10 годин vs 1-4 години	Потреба у спеціалізованій оптимізації моделей
Відкритий код EdgeFlowNet та повна документація	Відсутність бренду та репутації на ринку
Можливості	Загрози
Зростання ринку малопотужних AI (17-20% щорічно)	NVIDIA може випустити дешевший Jetson для цього сегменту
Попит на автономні дрони з довгим часом польоту	Google Coral може зменшити ціни та енергоспоживання
Розширення на IoT камери, носимі пристрої, робототехніку	Поява нових китайських платформ з кращими характеристиками
Партнерства з виробниками дронів та IoT пристроїв	Економічна нестабільність що зменшує інвестиції в R&D

5.4. Розроблення ринкової стратегії проєкту

5.4.1. Визначення стратегії охоплення ринку

Для виходу на ринок обрана стратегія концентрованого маркетингу з фокусом на нішу автономних дронів та мобільних роботів.

Таблиця 5.11

Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи	Готовність споживачів сприйняти продукт	Орієнтовний попит	Інтенсивність конкуренції	Простота входу
1	Виробники дронів (DJI, Parrot, стартапи)	Висока (активно шукають рішення для автономної навігації)	500-1000 компаній світових	Низька (мало готових рішень)	Середня (потреба у демонстрації)
2	Компанії відеоспостереження	Середня (консервативний ринок)	5000+ компаній	Висока (багато існуючих рішень)	Складна (довгі цикли продажів)
3	Дослідницькі лабораторії та університети	Висока (потреба у відкритих рішеннях)	1000+ лабораторій	Низька	Легка (наукові публікації)

Обрані цільові групи: виробники дронів (пріоритет 1) та дослідницькі лабораторії (пріоритет 2).

Стратегія охоплення ринку: концентрований маркетинг на нішу автономних дронів з поступовим розширенням на суміжні ринки.

5.4.2. Базова стратегія розвитку

Таблиця 5.12

Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції	Базова стратегія розвитку
1	Фокус на автономні дрони та мобільні роботи	Концентрований маркетинг	Єдине рішення що працює на периферії в реальному часі з низькою вартістю	Стратегія диференціації через технологічні інновації

Стратегія диференціації дозволяє уникнути цінової конкуренції та позиціонувати продукт як унікальне технологічне рішення з характеристиками недоступними у конкурентів.

5.4.3. Стратегія конкурентної поведінки

Таблиця 5.13

Визначення базової стратегії конкурентної поведінки

№	Чи є проєкт першопрохідцем?	Чи буде захищена інтелектуальна власність?	Чи можливий швидкий вихід на ринок?	Чи має потенціал прибутковості?	Стратегія конкурентної поведінки
1	Ні (є конкуренти у загальній сфері, але не на периферії)	Частково (патент на Dual 1D кореляцію можливий, але код відкритий)	Так (продукт готовий до демонстрації)	Так (низька вартість виробництва, готовність платити)	Стратегія виклику лідера через технологічні інновації

Проєкт кидає виклик існуючим “важким” моделям через радикально інший підхід — жертвуємо 30% точності заради 100× зменшення обчислень. Позиціонування: “Не найточніший, але єдиний що працює на периферії”.

5.5. Розроблення маркетингової програми стартап-проєкту

5.5.1. Визначення ключових переваг концепції потенційного товару

Таблиця 5.14

Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Обчислення оптичного потоку в реальному часі на дронах	Автономна навігація без GPS, уникнення перешкод, стабілізація	Єдине рішення що працює на мікрокомп’ютерах $20-30 з частотою 25 FPS
2	Тривала робота від акумулятора	Споживання <1 Вт дозволяє 6-8 годин роботи	Конкуренти споживають >100 Вт, не придатні для автономних систем
3	Легка інтеграція у існуючі системи	Стандартний формат ONNX, Python/C++ API	Відкритий код, детальна документація, приклади інтеграції

5.5.2. Опис трьох рівнів моделі товару

Таблиця 5.15

Опис трьох рівнів моделі товару

Рівні товару	**Сутність та складові**
I. Товар за задумом	Програмно-апаратне рішення для обчислення оптичного потоку на периферійних пристроях
II. Товар у реальному виконанні	Властивості/характеристики: - Швидкодія: 25.7 FPS (Технічна/Технологічна/Ергономічна) - Точність: EPE 2.28 пікселя (Технічна/Технологічна) - Ефективність: 1,47 GFLOPS (Технічна) - Споживання: <1 Вт (Технологічна/Екологічна) - Простота інтеграції: ONNX формат, Python SDK (Ергономічна) Якість: навчена модель, протестована на стандартних бенчмарках Пакування: GitHub репозиторій з документацією та прикладами Марка: EdgeFlowNet
III. Товар із підкріпленням	Додаткові послуги: - Технічна підтримка інтеграції - Навчання моделі на кастомних даних клієнта - Адаптація під нові апаратні платформи - Консультації з оптимізації

Захист від копіювання: патент на метод подвійної одновимірної кореляції, торгова марка EdgeFlowNet, експертиза у оптимізації під TPU.

5.5.3. Визначення цінової політики

Таблиця 5.16

Визначення меж встановлення ціни

№	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи	Верхня та нижня межі встановлення ціни
1	Відсутні (немає рішень що працюють на периферії)	Ліцензії на RAFT/FastFlowNet: безкоштовні (open source), комерційна підтримка: $50,000-200,000/рік	Виробники дронів: $1M-100M оборот, готові платити $10,000-50,000 за критичну технологію	Нижня: $5,000 (ліцензія на використання), Верхня: $30,000 (+ підтримка та адаптація)

Модель монетизації базується на диференційованому ліцензуванні з чотирма рівнями доступу до технології. Базова ліцензія вартістю п’ять тисяч доларів США надає право використання моделі у комерційних продуктах клієнта без додаткових сервісів підтримки. Преміум ліцензія вартістю п’ятнадцять тисяч доларів додатково включає технічну підтримку протягом одного року та регулярні оновлення моделі з покращеннями архітектури та точності. Корпоративна ліцензія вартістю тридцять тисяч доларів забезпечує повний спектр послуг включаючи навчання моделі на кастомних наборах даних клієнта та адаптацію під нові апаратні платформи які можуть з’явитися у майбутньому. Open Source версія надається безкоштовно для некомерційного використання у дослідницьких та освітніх цілях що сприяє формуванню спільноти користувачів та підвищенню впізнаваності технології.

5.5.4. Формування системи збуту

Таблиця 5.17

Формування системи збуту

№	Специфіка закупівельної поведінки	Функції збуту	Глибина каналу	Оптимальна система збуту
1	Виробники дронів: довгий цикл прийняття рішень (3-6 місяців), потреба у технічній демонстрації	Демонстрація, технічна підтримка інтеграції, навчання команди клієнта	Пряма (без посередників)	Прямі продажі через відділ розробки бізнесу, демонстрації на профільних виставках (CES, InterDrone)
2	Дослідницькі лабораторії: швидке прийняття рішень (1-2 місяці), орієнтація на наукову цінність	Публікації, документація, приклади коду	Пряма	GitHub, наукові публікації, партнерства з університетами

Канали збуту:

Власний веб-сайт з технічною документацією та демо - GitHub репозиторій для open source версії

Прямі продажі через LinkedIn та email campaigns до виробників дронів - Участь у виставках робототехніки та дронів

Партнерство з дистриб’юторами периферійних обчислювальних платформ (Sipeed, NVIDIA Jetson)

5.5.5. Концепція маркетингових комунікацій

Таблиця 5.18

Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій	Ключові позиції позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Виробники дронів: читають технічні блоги, відвідують виставки, активні на LinkedIn	Технічні блоги (Medium, Towards Data Science), LinkedIn, виставки (CES, InterDrone), YouTube (технічні демо)	Єдине рішення для обчислення оптичного потоку на периферії що працює в реальному часі	Показати конкретні цифри: 25.7 FPS на $20 апаратурі, <1 Вт споживання. Демонстрація роботи на реальному дроні	“Оптичний потік на периферії: вперше можливо. EdgeFlowNet — 100× ефективніший за RAFT, працює на мікрокомп’ютерах ”
2	Дослідники: читають arXiv, GitHub, відвідують конференції (CVPR, ICCV)	Наукові публікації (arXiv, конференції), GitHub, Twitter (AI community), Reddit (r/computervision)	Відкрите інноваційне рішення з новою архітектурою	Залучити до експериментів , отримати citations, побудувати спільноту	“Open source архітектура для edge optical flow. Подвійна 1D кореляція: 355× економія пам’яті. Код та моделі на GitHub”

Рекламні матеріали: 1. Технічне відео (YouTube): демонстрація роботи EdgeFlowNet на дроні в режимі реального часу з порівнянням метрик 2. Наукова стаття: публікація архітектури на arXiv та подання на CVPR/ICCV 3.

Інтерактивне демо: веб-додаток де можна завантажити своє відео та побачити результат 4. Case study: історія успіху інтеграції EdgeFlowNet у дрон-стартап

5.6. Висновки до розділу

У п’ятому розділі дипломної роботи розроблено стартап-проєкт для комерціалізації технології EdgeFlowNet.

Визначена актуальність проєкту полягає у тому, що існуючі рішення для обчислення оптичного потоку (RAFT, FastFlowNet) вимагають потужних GPU та споживають більше 100 Вт енергії, що робить їх непридатними для автономних дронів, мобільних роботів та пристроїв Інтернету речей. EdgeFlowNet вирішує цю проблему через радикальну оптимізацію архітектури.

Проведено технологічний аудит що підтвердив повну здійсненність проєкту. Всі необхідні технології є доступними та відкритими (PyTorch, ONNX, TPU-MLIR, MaixPy SDK).

Проаналізовано ринкові можливості. Ринок периферійних обчислень для комп’ютерного зору оцінюється у $8.5 млрд (2024) з прогнозом зростання до $22 млрд (2030), темп зростання 17-20% щорічно. Виявлено що EdgeFlowNet має унікальні конкурентні переваги як єдине рішення що працює в реальному часі на мікрокомп’ютерах вартістю $20-30 з споживанням енергії менше 1 Вт.

Визначені цільові групи споживачів - пріоритетом є виробники дронів (500-1000 компаній світових), додатково — дослідницькі лабораторії (1000+ установ). Обрана стратегія концентрованого маркетингу з фокусом на нішу автономних дронів.

Розроблена цінова політика: базова ліцензія $5,000, преміум $15,000, корпоративна $30,000, з безкоштовною open source версією для некомерційного використання. Така модель дозволяє залучити спільноту дослідників (маркетинг через публікації) та монетизувати комерційне використання.

Визначена система збуту: прямі продажі через технічні демонстрації та участь у профільних виставках (CES, InterDrone), онлайн присутність через GitHub та технічні блоги, партнерства з виробниками периферійних платформ.

Розроблена концепція маркетингових комунікацій з фокусом на конкретні технічні переваги (25.7 FPS, <1 Вт, 100× ефективніший) та демонстрацію роботи на реальних пристроях.

Проведений SWOT-аналіз показав що можливості (зростаючий ринок, попит на локальну обробку) значно переважають загрози (поява конкурентів від великих компаній). Основна слабка сторона — нижча точність порівняно з найкращими моделями — компенсується можливістю роботи на периферії що є критичним для цільових застосувань.

Отже, стартап-проєкт EdgeFlowNet є технологічно здійсненним, має чітко визначену цільову аудиторію з готовністю платити, працює на зростаючому ринку з помірною конкуренцією. Проєкт готовий до запуску після завершення патентування методу подвійної одновимірної кореляції та підготовки маркетингових матеріалів.

ВИСНОВКИ

У результаті проведеного дослідження розроблено та експериментально досліджено легковагу архітектуру нейронної мережі EdgeFlowNet для обчислення оптичного потоку у реальному часі на мікрокомп'ютерах з обмеженими обчислювальними ресурсами та енергетичним бюджетом.

Проведено комплексний аналіз предметної області та систематизовано проблеми існуючих методів обчислення оптичного потоку для мікрокомп’ютерів. Встановлено, що сучасні нейромережеві архітектури, такі як RAFT, FlowFormer та PWC-Net, мають критичні обмеження для розгортання на мікрокомп’ютерах через високу обчислювальну складність від 90 до 211 гігафлопс, великий розмір моделей від п'яти до сорока мільйонів параметрів та несумісність операцій з архітектурами тензорних процесорів.

Розроблено спеціалізовану архітектуру EdgeFlowNet, оптимізовану під апаратні обмеження мікрокомп'ютера Sophon CV181x. Модель містить 0.55 мільйона параметрів з обчислювальною складністю 1.48 гігафлопс, що є радикальним зменшенням порівняно з існуючими архітектурами. Архітектура включає ефективний кодувальник на основі роздільних згорток по глибині, гібридну кореляційну схему з економією пам'яті до 99.2 відсотка та конвеєр ітераційного уточнення без рекурентних блоків.

Розроблено методологію оптимізації архітектури під специфічні обмеження компілятора TPU-MLIR, що включає забезпечення статичних розмірів тензорів, мінімізацію операцій інтерполяції та використання виключно базових операцій з гарантованою підтримкою нейронними процесорами. Запропоновано техніки для ефективної цілочисельної квантизації восьмибітної розрядності із збереженням точності передбачень.

Формалізовано математичний апарат навчання компактної моделі, що базується на багатомасштабній функції втрат з стійкою функцією Шарбоньє, регуляризації з урахуванням меж об'єктів та двофазної стратегії навчання з поступовим ускладненням.

Проведено експериментальну валідацію розробленої архітектури на стандартних тестових наборах даних та реальній апаратурі цільової платформи. Досягнуто середню похибку кінцевої точки 2.84 пікселя на наборі даних Sintel Clean та 5.47 пікселів на наборі даних KITTI 2015. Продуктивність системи становить 25.7 кадрів за секунду з латентністю 38 мілісекунд на роздільності 256 на 320 пікселів у режимі цілочисельної квантизації восьмибітної розрядності. Споживання оперативної пам'яті становить 43.9 мегабайта, що складає 17.1 відсотка від доступної пам'яті платформи. Цілочисельна квантизація забезпечує прискорення обчислень у 3.8 рази порівняно з режимом з плаваючою комою при деградації точності лише на 7-8 відсотків.

Продемонстровано практичне значення отриманих результатів для застосування у мікродронах, камерах Інтернету речей, носимих пристроях та робототехнічних системах з обмеженим бюджетом на апаратне забезпечення.

Напрямки подальших досліджень включають покращення точності передбачень через архітектурні модифікації з механізмами уваги, розширення принципів оптимізації на інші задачі комп'ютерного зору, портування архітектури на інші периферійні платформи та мультимодальну інтеграцію з даними від інших сенсорів.

Розроблена архітектура EdgeFlowNet успішно розв'язує фундаментальну проблему впровадження високоякісного обчислення оптичного потоку на пристроях з екстремально обмеженими обчислювальними ресурсами, досягаючи оптимального балансу між точністю передбачень, швидкістю обробки та енергоефективністю роботи системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dosovitskiy A., Fischer P., Ilg E., Häusser P., Hazırbaş C., Golkov V., van der Smagt P., Cremers D., Brox T. FlowNet: Learning Optical Flow with Convolutional Networks. IEEE International Conference on Computer Vision (ICCV). 2015. P. 2758-2766. URL: https://arxiv.org/abs/1504.06852
2. Ilg E., Mayer N., Saikia T., Keuper M., Dosovitskiy A., Brox T. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. P. 2462-2470. URL: https://arxiv.org/abs/1612.01925
3. Teed Z., Deng J. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. European Conference on Computer Vision (ECCV). 2020. P. 402-419. URL: https://arxiv.org/abs/2003.12039
4. Huang Z., Shi X., Zhang C., Wang Q., Cheung K.C., Qin H., Dai J., Li H. FlowFormer: A Transformer Architecture for Optical Flow. European Conference on Computer Vision (ECCV). 2022. P. 668-685. URL: https://arxiv.org/abs/2203.16194
5. Kong L., Shen C., Yang J. FastFlowNet: A Lightweight Network for Fast Optical Flow Estimation. IEEE International Conference on Robotics and Automation (ICRA). 2021. P. 7855-7861. URL: https://arxiv.org/abs/2103.04524
6. Zhao S., Zhao Y., Zhang K., Xu Y. Global Optical Flow Estimation with Cross-Attention. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2022. P. 8090-8099. URL: https://openaccess.thecvf.com/content/CVPR2022/papers/Zhao_Global_Optical_Flow_Estimation_With_Cross-Attention_CVPR_2022_paper.pdf
7. Horn B.K.P., Schunck B.G. Determining optical flow. Artificial Intelligence. 1981. Vol. 17. No. 1-3. P. 185-203. URL: https://doi.org/10.1016/0004-3702(81)90024-2

8. Lucas B.D., Kanade T. An iterative image registration technique with an application to stereo vision. International Joint Conference on Artificial Intelligence (IJCAI). 1981. P. 674-679. URL: https://www.ri.cmu.edu/pub_files/pub3/lucas_bruce_d_1981_2/lucas_bruce_d_1981_2.pdf
9. Anandan P. A computational framework and an algorithm for the measurement of visual motion. International Journal of Computer Vision. 1989. Vol. 2. No. 3. P. 283-310. URL: https://doi.org/10.1007/BF00158168
10. Sun D., Yang X., Liu M.Y., Kautz J. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018. P. 8934-8943. URL: https://arxiv.org/abs/1709.02371
11. Xu H., Zhang J., Cai J., Rezatofighi H., Tao D. GMFlow: Learning Optical Flow via Global Matching. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2022. P. 8121-8130. URL: https://arxiv.org/abs/2111.13680
12. Lu Y., Valmadre J., Wang H., Kannala J., Harandi M., Torr P.H.S. HD³: Hierarchical Decomposition-Driven Optical Flow Estimation with Spatial-Temporal Consistency. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2021. P. 16257-16267. URL: https://arxiv.org/abs/2105.11744
13. Sophon Technology. SG2002 System-on-Chip Technical Reference Manual. Version 1.0. Sophon Technology Co., Ltd., 2023. 156 p. URL: https://github.com/sophgo/sophon-doc
14. Sophon Technology. TPU-MLIR Compiler User Guide. Version 2.1. Sophon Technology Co., Ltd., 2023. 89 p. URL: https://github.com/sophgo/tpu-mlir
15. Bengio Y., Louradour J., Collobert R., Weston J. Curriculum learning. International Conference on Machine Learning (ICML). 2009. P. 41-48. URL: https://doi.org/10.1145/1553374.1553380

16. Meister S., Hur J., Roth S. UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss. AAAI Conference on Artificial Intelligence. 2018. P. 7251-7259. URL: https://arxiv.org/abs/1711.07837
17. Nistér D., Naroditsky O., Bergen J. Visual odometry. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2004. Vol. 1. P. 652-659. URL: https://doi.org/10.1109/CVPR.2004.1315094
18. Butler D.J., Wulff J., Stanley G.B., Black M.J. A naturalistic open source movie for optical flow evaluation. European Conference on Computer Vision (ECCV). 2012. P. 611-625. URL: https://doi.org/10.1007/978-3-642-33783-3_44
19. Geiger A., Lenz P., Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2012. P. 3354-3361. URL: https://doi.org/10.1109/CVPR.2012.6248074
20. Mayer N., Ilg E., Häusser P., Fischer P., Cremers D., Dosovitskiy A., Brox T. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 4040-4048. URL: https://arxiv.org/abs/1512.02134
21. Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861. 2017. URL: https://arxiv.org/abs/1704.04861
22. Hu J., Shen L., Sun G. Squeeze-and-Excitation Networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018. P. 7132-7141. URL: https://arxiv.org/abs/1709.01507
23. Jacob B., Kligys S., Chen B., Zhu M., Tang M., Howard A., Adam H., Kalenichenko D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. IEEE Conference on Computer Vision and

Pattern Recognition (CVPR). 2018. P. 2704-2713. URL: https://arxiv.org/abs/1712.05877

24. Gholami A., Kim S., Dong Z., Yao Z., Mahoney M.W., Keutzer K. A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv preprint arXiv:2103.13630. 2021. URL: https://arxiv.org/abs/2103.13630
25. Charbonnier P., Blanc-Feraud L., Aubert G., Barlaud M. Deterministic edge-preserving regularization in computed imaging. IEEE Transactions on Image Processing. 1997. Vol. 6. No. 2. P. 298-311. URL: https://doi.org/10.1109/83.551699
26. Bouwmeester R.J., Paredes-Vallés F., de Croon G.C.H.E. NanoFlowNet: Real-time Dense Optical Flow on a Nano Quadcopter. arXiv preprint arXiv:2209.06918. 2022. URL: https://arxiv.org/abs/2209.06918
27. Znobishchev A., Filev V., Kudashev O., Orlov N., Shi H. CompactFlowNet: Efficient Real-time Optical Flow Estimation on Mobile Devices. arXiv preprint arXiv:2412.13273. 2024. URL: https://arxiv.org/abs/2412.13273
28. Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018. P. 4510-4520. URL: https://arxiv.org/abs/1801.04381
29. Hinton G., Vinyals O., Dean J. Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531. 2015. URL: https://arxiv.org/abs/1503.02531
30. Polyak B.T., Juditsky A.B. Acceleration of Stochastic Approximation by Averaging. SIAM Journal on Control and Optimization. 1992. Vol. 30. No. 4. P. 838-855. URL: https://doi.org/10.1137/0330046
31. Loshchilov I., Hutter F. SGDR: Stochastic Gradient Descent with Warm Restarts. International Conference on Learning Representations (ICLR). 2017. URL: https://arxiv.org/abs/1608.03983
32. Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L., Desmaison A., Kopf A., Yang E., DeVito Z., Raison M., Tejani A., Chilamkurthy S., Steiner B., Fang L., Bai J., Chintala S.

PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems (NeurIPS). 2019. P. 8024-8035. URL: https://arxiv.org/abs/1912.01703

33. Bradski G. The OpenCV Library. Dr. Dobb's Journal of Software Tools. 2000. Vol. 25. No. 11. P. 120-125. URL: https://opencv.org
34. Harris C.R., Millman K.J., van der Walt S.J., Gommers R., Virtanen P., Cournapeau D., Wieser E., Taylor J., Berg S., Smith N.J., Kern R., Picus M., Hoyer S., van Kerkwijk M.H., Brett M., Haldane A., del Río J.F., Wiebe M., Peterson P., Gérard-Marchant P., Sheppard K., Reddy T., Weckesser W., Abbasi H., Gohlke C., Oliphant T.E. Array Programming with NumPy. Nature. 2020. Vol. 585. P. 357-362. URL: https://doi.org/10.1038/s41586-020-2649-2
35. Scaramuzza D., Fraundorfer F. Visual Odometry: Part I: The First 30 Years and Fundamentals. IEEE Robotics & Automation Magazine. 2011. Vol. 18. No. 4. P. 80-92. URL: https://doi.org/10.1109/MRA.2011.943233
36. Zhang R., Isola P., Efros A.A., Shechtman E., Wang O. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018. P. 586-595. URL: https://arxiv.org/abs/1801.03924
37. Sun D., Roth S., Black M.J. A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles Behind Them. International Journal of Computer Vision. 2014. Vol. 106. No. 2. P. 115-137. URL: https://doi.org/10.1007/s10559-025-00837-0
38. Shymkovych, V. M., Doroshenko, A. Yu., Kravets, P. I., & Yatsenko, O. A. (2025). FPGA-Based Method for Automated Design of Neural Network Controllers. Cybernetics and Systems Analysis. https://doi.org/10.1007/s10559-025-00837-0
39. Shymkovych V., Doroshenko A., Mamedov T., Yatsenko O. (2022). Automated design of an artificial neuron for field-programmable gate arrays based on an Algebra-Algorithmic approach. International Scientific Technical

Journal “Problems of Control and Informatics,” 67(5), 61–72. https://doi.org/10.34229/2786-6505-2022-5-6

40. Shymkovych, V., Telenyk, S., & Kravets, P. (2021). Hardware implementation of radial-basis neural networks with Gaussian activation functions on FPGA. Neural Computing and Applications, 33(15), 9467–9479. https://doi.org/10.1007/s00521-021-05706-3

ДОДАТОК А

Репозиторій EdgeFlowNet проєкту

Репозиторій EdgeFlowNet проєкту доступний за посиланням https://github.com/ryngach/EdgeFlowNet. Крім того, доступ забезпечено за QR-кодом (рис. А.1).

Рис. А.1. QR-код з посиланням на репозиторій EdgeFlowNet проекту