МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

КУРЕНЧУК Андрій Сергійович

Алгоритми оптимізації параметрів нейронної мережі типу U-net / Algorithms for optimization of U-net type neural network parameters

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи Кім-21
А.С. Куренчук
_____

Науковий керівник
к.т.н., О. Й. Піцун
_____

Кваліфікаційну роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____О.Л. Дубчак

Тернопіль – 2024

АНОТАЦІЯ

Куренчук А.С. Алгоритми оптимізації параметрів нейронної мережі типу U-net . – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма. Західноукраїнський національний університет, Тернопіль, 2024.

Робота написана обсягом 61 сторінка і містить 8 ілюстрацій, 5 таблиць, 2 додатки та 35 джерел за переліком посилань. Метою роботи є розроблення і існуючих алгоритмів оптимізації параметрів нейронної мережі типу U-net для підвищення якості сегментації зображень, зокрема в галузі медичних даних.

Методи досліджень. Для розв'язання поставлених задач у кваліфікаційній роботі використано: методи математичного аналізу та об'єктно-орієнтованого програмування (для проектування програмних засобів аналізу і синтезу зображень).

Результати дослідження: алгоритми оптимізації параметрів нейронної мережі типу U-net для підвищення якості сегментації зображень, зокрема в галузі медичних даних.

Орієнтовні напрямки розвитку досліджень: дослідження можливостей розширення існуючих алгоритмів для роботи з даними з різних джерел, таких як супутникові зображення, біометричні дані чи зображення з промислових камер. Це дозволить підвищити універсальність та масштабованість розроблених підходів.

КЛЮЧОВІ СЛОВА: U-NET, ГІПЕРПАРАМЕТЕР, АЛГОРИТМ НАВЧАННЯ, СЕГМЕНТАЦІЯ.

ANNOTATION

Kurenchuk A. Algorithms for optimization of U-net type neural network parameters. - Manuscript.

Qualification work for obtaining the educational degree "Master" in specialty 123 "Computer Engineering", educational and professional program. West Ukrainian National University, Ternopil, 2024.

The work is 61 pages long and contains 8 illustrations, 5 tables, 2 appendixes and 35 sources for references. The aim of the work is to develop existing algorithms for optimizing the parameters of a neural network of the U-net type to improve the quality of image segmentation, in particular in the field of medical data.

Research methods. To solve the tasks set in the qualification work, the following methods were used: methods of mathematical analysis and object-oriented programming (for designing software tools for image analysis and synthesis).

Research results: algorithms for optimizing the parameters of a neural network of the U-net type to improve the quality of image segmentation, in particular in the field of medical data.

Approximate directions of research development: study of the possibilities of expanding existing algorithms for working with data from various sources, such as satellite images, biometric data or images from industrial cameras. This will increase the versatility and scalability of the developed approaches.

KEYWORDS: U-NET, HYPERPARAMETER, LEARNING ALGORITHM, SEGMENTATION.

ЗМІСТ

ВСТУП

Сучасний розвиток комп’ютерного зору і глибокого навчання значно розширив можливості аналізу та обробки зображень у багатьох прикладних галузях, включаючи медицину, автономні системи, промислову автоматизацію та екологічний моніторинг. У цьому контексті сегментація зображень, як одна з ключових задач комп’ютерного зору, відіграє вирішальну роль, забезпечуючи виділення важливих об’єктів і структур із зображень. Однією з найбільш ефективних архітектур для вирішення цієї задачі є нейронна мережа типу U-Net, яка зарекомендувала себе завдяки здатності об’єднувати глобальний контекст і локальні деталі через використання симетричної структури з пропускними з’єднаннями.

Попри високу популярність U-Net у дослідницькій та практичній спільноті, оптимізація її параметрів і гіперпараметрів залишається складною і критично важливою задачею. Ефективність моделі значною мірою залежить від правильного налаштування таких параметрів, як архітектура, функції втрат, швидкість навчання та методи регуляризації. Неправильне налаштування може призвести до значних втрат у продуктивності, збільшення часу навчання або навіть до нездатності моделі досягати збіжності. Більше того, складність задач сегментації, таких як обробка гістологічних зображень, вимагає адаптивних підходів, які враховують особливості вхідних даних, зокрема їх масштабність, варіативність та наявність шумів.

Вимоги до датасетів для навчання моделей U-Net є важливим аспектом розробки алгоритмів сегментації. Навчальні набори даних мають бути достатньо репрезентативними, включати велику кількість зразків, що охоплюють можливу варіативність у структурі, масштабах та текстурах об’єктів. Крім того, точність сегментації значною мірою залежить від якості анотацій, які мають забезпечувати чітке відображення меж об’єктів. Зображення повинні бути

попередньо оброблені, зокрема нормалізовані, а також доповнені аугментацією, що дозволяє збільшити різноманітність навчальних даних і зменшити ймовірність перенавчання моделі. Особливу увагу слід приділяти забезпеченню балансу між різними класами даних у задачах багатокласової сегментації, що сприяє підвищенню стійкості та узагальнюючої здатності моделі.

Актуальність даного дослідження обумовлена зростанням потреби у високоточних і стабільних методах сегментації для медичних, біологічних і технічних даних, а також необхідністю зменшення обчислювальних витрат у процесі навчання. Пошук і розробка ефективних алгоритмів оптимізації параметрів U-Net є важливим завданням, спрямованим на підвищення точності моделі, забезпечення її здатності до узагальнення на нових даних та зниження потреби у великих обчислювальних ресурсах.

У рамках цієї роботи ставиться завдання аналізу існуючих підходів до оптимізації параметрів U-Net, розробки нових алгоритмів, спрямованих на підвищення точності сегментації та стабільності моделі, а також оцінки їх ефективності на практичних прикладах. Особлива увага приділяється адаптивності запропонованих підходів до задач із високою варіативністю даних, що підкреслює їх значущість для вирішення актуальних проблем сегментації у різних прикладних галузях.

Мета роботи: Розробка нових або вдосконалення існуючих алгоритмів оптимізації параметрів нейронної мережі типу U-net для підвищення якості сегментації зображень, зокрема в галузі медичних даних.

Об'єкт дослідження: нейронна мережа типу U-net та методи її оптимізації.

Предмет дослідження: алгоритми та методи оптимізації гіперпараметрів нейронної мережі типу U-net.

Задачі дослідження:

1. Провести аналіз алгоритмів оптимізації та пошуку гіперпараметрів мереж типу U-net.

2. Розробити алгоритми та реалізувати архітектуру, що використовує мультишарові входи з різними масштабами зображень, для підвищення точності сегментації шляхом урахування деталей різного розміру.

3. Розробити програмне забезпечення яке підтримує підготовку даних, налаштування окремих моделей для кожного масштабу та об’єднання результатів.

4. Провести експериментальне тестування алгоритму на гістологічних зображеннях для оцінки його ефективності, використовуючи метрики IoU та Dice Score.

Наукова новизна одержаних результатів. Розроблено алгоритми оптимізації параметрів нейронної мережі типу U-net з використанням мультишарових входів та адаптивної агрегації результатів, що забезпечило врахування деталей різного масштабу.

Практичне значення отриманих результатів. Розроблені алгоритми дозволяють досягти високої точності сегментації, оптимального використання обчислювальних ресурсів.

Апробація роботи. Отримані результати опубліковані в межах всеукраїнської науково-практичної конференції «Інтелектуальні комп’ютерні системи та мережі», опубліковано тези доповіді [1,2].

Кваліфікаційна робота містить три розділи, висновки, список використаних джерел та додатки.

Перший розділ, присвячений аналізу методів оптимізації параметрів нейронної мережі типу U-net.

Другий розділ присвячено вдосконаленню алгоритмів оптимізації.

Третій розділ присвячений розробці програмного забезпечення та проведенню експериментальних досліджень.

1. АНАЛІЗ МЕТОДІВ ОПТИМІЗАЦІЇ ПАРАМЕТРІВ НЕЙРОННОЇ МЕРЕЖІ ТИПУ U-NET

1.1 Аналіз алгоритмів оптимізації

U-Net — це архітектура нейронної мережі, спеціально розроблена для задач сегментації зображень. Її основна ідея полягає у використанні двох основних шляхів: шляху стиснення (downsampling) і шляху відновлення (upsampling). Архітектура має форму "U", де шари на шляху стиснення з'єднуються з відповідними шарами на шляху відновлення через скіпові з'єднання [3-4].

Ключові етапи процесу сегментації:

1.Шлях стиснення (Encoder). Зображення проходить через кілька згорткових шарів із функцією активації (зазвичай ReLU). Застосовуються шари MaxPooling2D для зменшення розмірності зображення, що дозволяє моделі виділяти високорівневі ознаки.

2.Боттлнек. На цьому етапі модель зберігає найвищого рівня ознаки з мінімальною розмірністю простору.

3. Шлях відновлення (Decoder). Застосовуються шари Upsampling (наприклад, Transposed Convolution), які збільшують розмірність зображення. Використовуються скіпові з’єднання для об'єднання ознак із шляху стиснення, що дозволяє моделі враховувати як контекст, так і деталі.

4. Вихідний шар. Завершальний шар зазвичай має функцію активації softmax (для багатокласової сегментації) або sigmoid (для двокласової сегментації). Результат — сегментоване зображення, де кожному пікселю присвоєно клас.

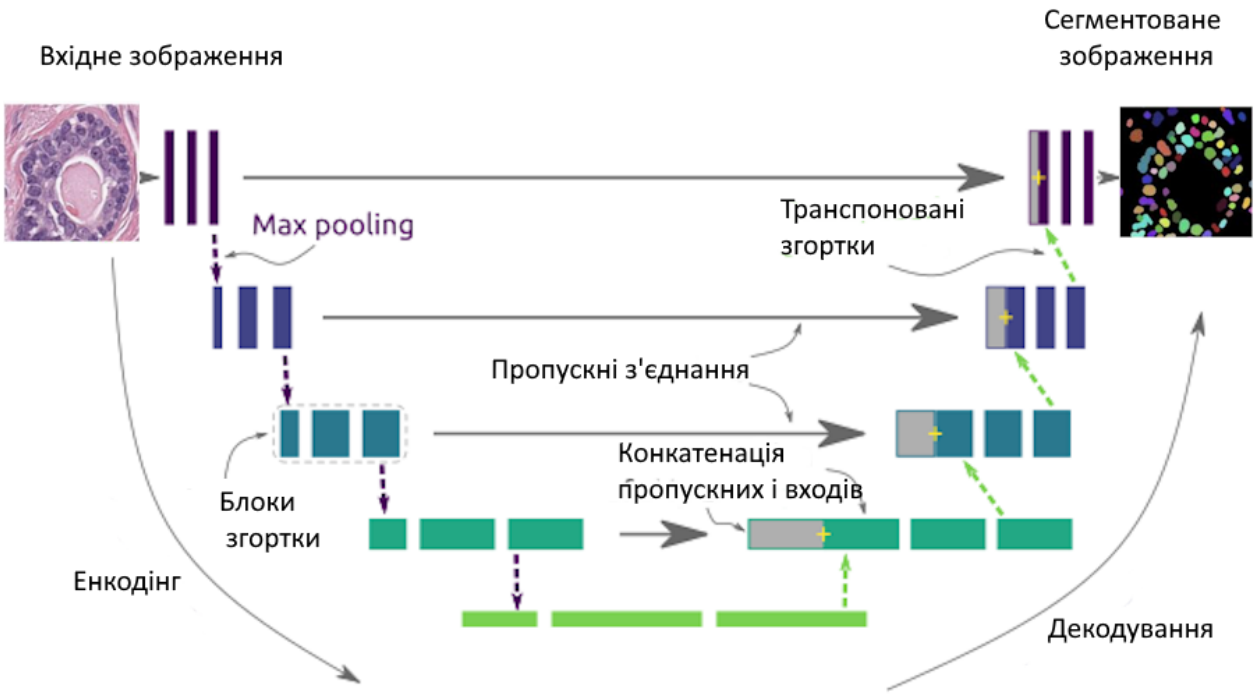

Рисунок 1.1 – Структура мережі U-net для сегментації

Рука «вниз» кодує вхідні дані в картах функцій із дедалі меншою роздільною здатністю, кількість яких зростає (тобто каналів). «Верхня» рука декодує дані, збільшуючи роздільну здатність через транспоновані згортки за допомогою конкатенованого введення від з’єднань пропуску (skip connections).

У сучасних задачах машинного навчання та глибокого навчання алгоритми оптимізації відіграють критичну роль, оскільки саме вони визначають ефективність і швидкість навчання моделей. Зокрема, ці алгоритми спрямовані на мінімізацію функції втрат шляхом оновлення параметрів моделі на основі градієнтної інформації.

У цьому підрозділі буде розглянуто популярні алгоритми оптимізації, такі як Adam [5], RMSprop [6], L-BFGS [7], а також їх модифікації, що дозволяють вирішувати задачі навчання моделей з високою ефективністю та стабільністю. Опис кожного алгоритму супроводжуватиметься їх теоретичними основами, перевагами, недоліками та практичними аспектами застосування. Особлива увага приділяється порівнянню їхньої продуктивності у різних сценаріях, включаючи навчання великих глибоких нейронних мереж.

Позначення основних параметрів алгоритму Adam:

- θ_t – параметри моделі на ітерації t.
- g_t – градієнт функції втрат $L(\theta_t)$ на ітерації t.
- m_t – перший момент градієнта (експоненційно згладжена середня градієнта).
- v_t – другий момент градієнта (експоненційно згладжена середня квадрата градієнта).
- $\widehat{m_t}$ – скоригований (зсунутий) перший момент.
- $\widehat{v_t}$– скоригований (зсунутий) другий момент.
- α – коефіцієнт швидкості навчання (learning rate).
- β_1 – коефіцієнт експоненційного згладжування для першого моменту (звичайно β_1=0.9).
- β_2 – коефіцієнт експоненційного згладжування для другого моменту (звичайно β_2=0.999).
- ϵ – мале додатне значення для уникнення ділення на нуль (звичайно ϵ=10^{-8}).
- t – номер поточної ітерації.

Псевдокод алгоритму Adam, частина ініціалізаіцї:

```
# Ініціалізація параметрів
t = 0                          # Лічильник ітерацій
m_0 = 0                        # Ініціалізація першого моменту
v_0 = 0                        # Ініціалізація другого моменту
θ_0 = θ_initial                # Початкові параметри моделі

# Гіперпараметри
α = learning_rate              # Швидкість навчання
β1 = 0.9                       # Коефіцієнт для першого моменту
β2 = 0.999                     # Коефіцієнт для другого моменту
ε = 10^-8                      # Малий додатний параметр
```

Псевдокод алгоритму Adam,основний цикл:

```
# Основний цикл оптимізації
repeat until convergence:
    t = t + 1
```

```
    # Обчислення градієнта функції втрат
    g_t = ∇L(θ_t)           # Градієнт функції втрат за
параметрами
    # Оновлення першого моменту
    m_t = β1 * m_(t-1) + (1 - β1) * g_t
    # Оновлення другого моменту
    v_t = β2 * v_(t-1) + (1 - β2) * g_t^2
    # Корекція зсуву для першого моменту
    m^_t = m_t / (1 - β1^t)
    # Корекція зсуву для другого моменту
    v^_t = v_t / (1 - β2^t)
    # Оновлення параметрів моделі
    θ_t = θ_(t-1) - α * m^_t / (√(v^_t) + ε)
return θ_t
```

Опис основних кроків:

1. Ініціалізація - Усі моменти m_0 та v_0 встановлюються рівними нулю, а параметри моделі беруться із початкових значень θ_0. Гіперпараметри α, β_1, β_2 та ϵ задаються відповідно до вимог задачі.

2. Обчислення градієнта - на кожній ітерації t обчислюється градієнт функції втрат g_t .

3. Оновлення моментів - експоненційно згладжені значення для середнього градієнта (перший момент) та його квадрата (другий момент) оновлюються за допомогою коефіцієнтів β_1 і β_2.

4. Корекція зсуву - через те, що m_t і v_t на ранніх етапах можуть мати зсув до нуля, вводиться корекція зсуву, яка нормалізує їх значення.

5. Оновлення параметрів - параметри моделі оновлюються за допомогою скоригованих моментів і швидкості навчання α.

6. Критерій збіжності - алгоритм повторюється до досягнення збіжності функції втрат або до виконання інших умов завершення.

На графіку зображено збіжність функції втрат (Loss Function) під час ітерацій алгоритму оптимізації. Жовта крива демонструє експоненціальне зменшення значення втрат із деякими коливаннями через випадковий шум. Червона пунктирна лінія позначає поріг збіжності (Convergence Threshold), який може слугувати критерієм для завершення навчання. Коли значення функції

втрат стабільно знаходиться нижче цього порогу, алгоритм може бути визнаний збіжним.

Позначення основних параметрів для RMSprop:

- θ_t – параметри моделі на ітерації ttt.
- g_t – градієнт функції втрат L(θt)L(\theta_t)L(θt) на ітерації ttt.
- v_{tv} – експоненційно згладжена середня квадрата градієнта.
- α – коефіцієнт швидкості навчання (learning rate).
- β – коефіцієнт експоненційного згладжування (звичайно β=0.9\beta = 0.9β=0.9).
- ϵ – мале додатне значення для уникнення ділення на нуль (звичайно $\epsilon=10^{-8}$).
- t – номер поточної ітерації.

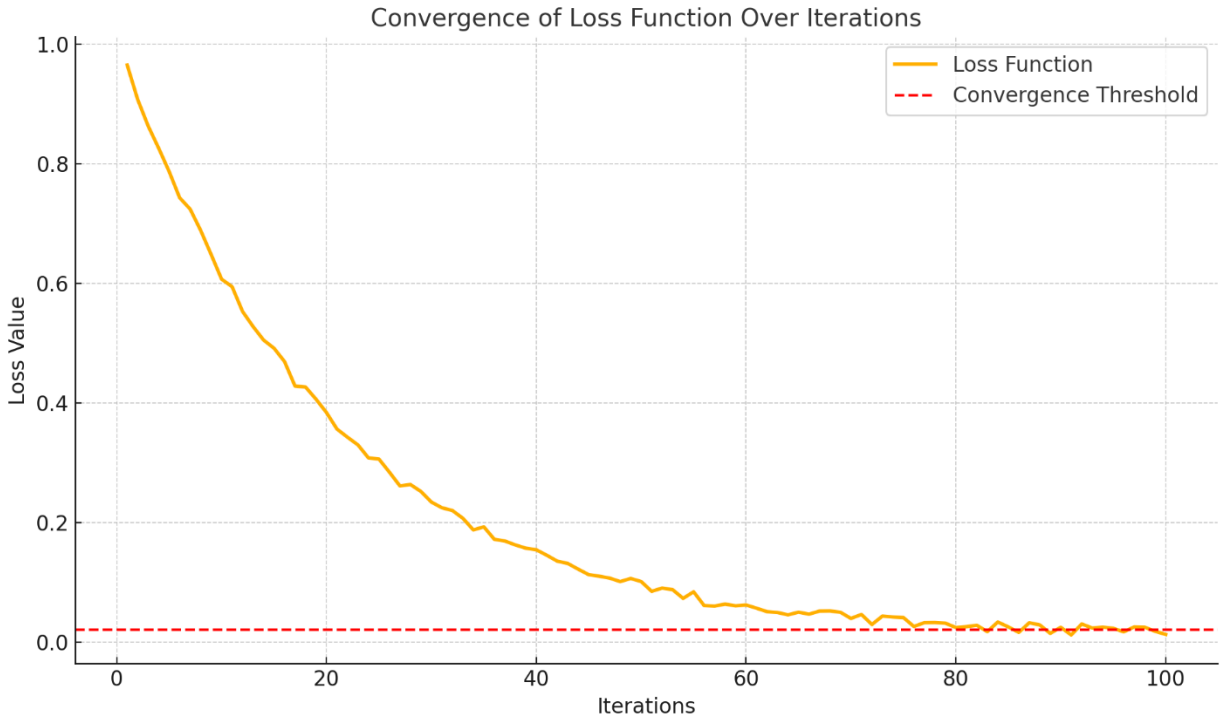

Рисунок 1.2 – Збіжність алгоритму Adam

Псевдокод алгоритму RMSprop

```
t = 0                          # Лічильник ітерацій
v_0 = 0                        # Ініц серед квадрата градієнта
θ_0 = θ_initial                # Початкові параметри моделі
# Гіперпараметри
```

```
α = learning_rate           # Швидкість навчання
β = 0.9                     # Коефіцієнт згладжування
ε = 10^-8                   # Малий додатний параметр
repeat until convergence:
    t = t + 1
    g_t = ∇L(θ_t)           # Градієнт функції втрат за параметрами
    # Оновлення другого моменту (середня квадрата градієнта)
    v_t = β * v_(t-1) + (1 - β) * g_t^2
    # Оновлення параметрів моделі
    θ_t = θ_(t-1) - α * g_t / (√(v_t) + ε)
return θ_t
```

Опис основних кроків алгоритму RMSprop:

1. Ініціалізація - На початку всі значення v_0 встановлюються рівними нулю, а параметри моделі приймають початкові значення θ_0. Вибір гіперпараметрів α, β, і ϵ залежить від конкретної задачі.

2. Обчислення градієнта - на кожній ітерації t обчислюється градієнт функції втрат g_t щодо параметрів моделі.

3. Оновлення середньої квадрата градієнта - за допомогою коефіцієнта β експоненційно згладжується середнє значення квадратів градієнтів, що дозволяє враховувати історію зміни градієнтів.

4. Оновлення параметрів - параметри оновлюються з урахуванням нормалізованого градієнта, поділеного на корінь із середньої квадрата градієнта v_t. Малий параметр є додається для уникнення ділення на нуль.

5. Критерій збіжності - алгоритм повторюється, поки функція втрат не досягне бажаного значення, або до виконання інших умов завершення, таких як максимальна кількість ітерацій.

1.2 Аналіз методів пошуку оптимальних гіперпараметрів

Метод Grid Search [8] є одним із класичних підходів до пошуку оптимальних гіперпараметрів у задачах машинного навчання. Цей метод передбачає побудову прямокутної сітки можливих значень гіперпараметрів і

повне перебрання всіх їхніх комбінацій. Незважаючи на високу обчислювальну вартість, Grid Search гарантує виявлення оптимального набору гіперпараметрів у межах заданого простору.

Алгоритм Grid Search.

Нехай H позначає множину гіперпараметрів, які потрібно налаштувати, а D_{train} і D_{val} позначають відповідно тренувальну і валідаційну вибірки. Гіперпараметри визначаються вектором $\mathbf{h}=[h_1,h_2,\ldots,h_k]$, де $h_i \in H_i$, а H_i — простір можливих значень для i-го гіперпараметра.

1. Задати дискретний простір значень гіперпараметрів:

$$H=H_1\times H_2\times\cdots\times H_k,$$

де $Hi=\{h_{i1},h_{i2},\ldots,h_{imi}\}$.

2. Визначити функцію помилки $L(\mathbf{h},D_{train},D_{val})$, що оцінює модель з гіперпараметрами **h**.
3. Для кожного набору гіперпараметрів $h \in H$ виконати наступні дії:
   - Навчити модель на D_{train}.
   - Обчислити значення функції помилки на D_{val}.
4. Визначити оптимальні гіперпараметри $\mathbf{h}^*$:

$$\mathbf{h}^* = \arg\min_{\mathbf{h}\in\mathcal{H}} \mathcal{L}(\mathbf{h}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}).$$

Таблиця 1.1 - Приклад перебору комбінацій гіперпараметрів

Гіперпараметр	Можливі значення
η (швидкість навчання)	{0.001,0.01,0.1}
λ (L2-регуляризація)	{0.1,1,10}
n_{layers} (кількість шарів)	{2,3,4}

Таблиця 1.2 - Результати Grid Search

η	λ	n_{layers}	L_{val}
0.001	0.1	2	0.25
0.001	0.1	3	0.22
0.001	0.1	4	0.27
0.001	1	2	0.24

З огляду на результати, найкращі гіперпараметри визначаються як η=0.001, λ=1, n_{layers}=3 якщо вони мінімізують L_{val}.

Метод Random Search [9] є ефективною альтернативою Grid Search для оптимізації гіперпараметрів у задачах машинного навчання. На відміну від Grid Search, який перебирає всі можливі комбінації гіперпараметрів у заздалегідь визначеній сітці, Random Search випадковим чином вибирає комбінації гіперпараметрів з визначених просторів. Це дозволяє зменшити обчислювальну складність і збільшити ймовірність знайти оптимальне рішення за меншу кількість ітерацій.

Нехай

$$H=H1\times H_2\times\cdots\times H_k$$

— простір гіперпараметрів, де H_i — область визначення і-го гіперпараметра. Для кожного набору гіперпараметрів $\mathbf{h}=[h_1,h_2,\ldots,h_k]$, метод мінімізує функцію втрат $L(\mathbf{h},D_{train},D_{val})$, використовуючи випадкові вибірки з простору H.

Алгоритм:

1. Визначається розмір вибірки N, тобто кількість випадкових комбінацій гіперпараметрів, які будуть перевірені:

$$\mathbf{H}_{\text{sample}} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N\}, \quad \mathbf{h}_j \sim \mathcal{H}, \quad j = 1, \ldots, N.$$

2. Для кожного $h_j \in H_{sample}$, обчислюється функція втрат:

$$\mathcal{L}_j = \mathcal{L}(\mathbf{h}_j, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})$$

3. Оптимальні гіперпараметри визначаються як:

$$\mathbf{h}^* = \arg \min_{\mathbf{h}_j \in \mathbf{H}_{\text{sample}}} \mathcal{L}_j .$$

Random Search дозволяє охопити більше можливих комбінацій у просторах великої розмірності, ефективно використовуючи обчислювальні ресурси. Метод також є більш ймовірним у знаходженні оптимального або близького до оптимального рішення за фіксований час, порівняно з Grid Search.

Метод Bayesian Optimization є ефективним підходом до автоматизованого пошуку оптимальних гіперпараметрів у задачах машинного навчання. Він базується на побудові ймовірнісної моделі функції цільової помилки, яка покращується ітеративно на основі нових спостережень. Цей метод дозволяє значно зменшити кількість ітерацій, необхідних для знаходження оптимальних гіперпараметрів, у порівнянні з Grid Search та Random Search.

Формалізація методу Bayesian Optimization [10]. Нехай простір гіперпараметрів задано

$$H \subseteq \mathrm{R}^{\mathrm{d}}$$

де d — кількість гіперпараметрів.

Цільова функція втрат L(**h**) визначає якість моделі при гіперпараметрах **h** Оскільки $L(\mathbf{h})$ є обчислювально дорогою функцією, для її апроксимації використовується ймовірнісна модель, наприклад, гаусівський процес (GP).

1. Апроксимація функції втрат - гаусівський процес моделює функцію $L(\mathbf{h})$ за допомогою апріорного розподілу:

$$\mathcal{L}(\mathbf{h}) \sim \mathcal{GP}(\mu(\mathbf{h}), k(\mathbf{h}, \mathbf{h}')),$$

де μ(**h**) — середнє значення функції, а k(**h**,**h**′) — коваріаційна функція (ядро).

2. Формування функції корисності: На основі апроксимації обирається функція корисності (acquisition function), наприклад, Expected Improvement:

$$\alpha(\mathbf{h}) = \mathbb{E}[\max(0, \mathcal{L}^* - \mathcal{L}(\mathbf{h}))],$$

де $L*$ — найкраще значення цільової функції на поточний момент.

3.Оптимізація функції корисності - наступне значення гіперпараметрів h_{t+1} обирається шляхом максимізації функції корисності:

$$\mathbf{h}_{t+1} = \arg\max_{\mathbf{h}\in\mathcal{H}} \alpha(\mathbf{h}).$$

4.Оновлення моделі - додати нову точку спостереження ($\mathbf{h}_{t+1}$,L($\mathbf{h}_{t+1}$)) до моделі й повторити ітерацію.

5.Завершення - алгоритм завершується після досягнення заданої кількості ітерацій або збіжності.

Оптимальні гіперпараметри визначаються як:

$$\mathbf{h}^* = \arg\min_{\mathbf{h}\in\mathcal{H}} \mathcal{L}(\mathbf{h}).$$

Таблиця 1.3 - Приклад ітерацій Bayesian Optimization

Ітерація (t)	$\mathbf{h}_t$ (гіперпараметри)	L($\mathbf{h}_t$))	μ(**h**) (середнє)	σ(**h**) (дисперсія)
1	[0.1,0.5,3][0.1,0.5,3]	0.28	0.35	0.10
2	[0.2,0.3,4][0.2,0.3,4]	0.25	0.32	0.08
3	[0.05,0.4,3.5][0.05,0.4,3.5]	0.22	0.27	0.07
4	[0.15,0.45,3.2][0.15,0.45,3.2]	0.20	0.25	0.05

Згідно з результатами, оптимальне значення гіперпараметрів визначено як $h^*=[0.15,0.45,3.2]$, оскільки вони мінімізують функцію втрат L

1.3 Програмні засоби сегментації із використанням U-net

Сучасні програмні засоби сегментації, засновані на U-Net, реалізуються здебільшого за допомогою бібліотек глибокого навчання, таких як TensorFlow [11], PyTorch [12] або Keras[13]. Вони забезпечують модульність і гнучкість у налаштуванні архітектури моделі, що дозволяє адаптувати U-Net до специфічних вимог задачі, таких як розмір вхідних зображень, кількість класів для сегментації чи обчислювальні ресурси. У програмних реалізаціях часто передбачено можливість інтеграції розширень базової архітектури, зокрема механізмів уваги (attention mechanisms) для покращення обробки важливих ділянок зображення або багаторівневих з'єднань (multi-scale connections) для роботи з інформацією різного масштабу.

Таблиця 1.4 - Порівняльна характеристика бібліотек глибокого навчання

Характеристика	TensorFlow	PyTorch	Keras
1	2	3	4
Рік випуску	2015	2016	2015
Розробник	Google	Facebook (Meta)	Спочатку незалежна розробка, зараз інтегрована в TensorFlow
Мова програмування	C++, Python, JavaScript	C++, Python	Python
Гнучкість	Висока, але потребує явного опису графів	Дуже висока, завдяки динамічним обчисленням	Помірна, орієнтована на високу абстракцію
Підтримка динамічних обчислень	Можлива через TensorFlow Eager Execution	Природно реалізована	Ні, динамічні обчислення не підтримуються

Продовження таблиці 1.4

1	2	3	4
Простота використання	Складний для новачків, вимагає глибокого розуміння	Інтуїтивно зрозумілий API	Дуже простий, призначений для швидкого прототипування
Продуктивність	Висока, оптимізована для розгортання в промислових середовищах	Висока, особливо в наукових дослідженнях	Помірна, залежить від використання TensorFlow
Платформи розгортання	Сервери, мобільні пристрої, веб	Сервери, мобільні пристрої	Насамперед TensorFlow
Можливості розгортання	TensorFlow Serving, TensorFlow Lite, TensorFlow.js	TorchServe, TorchScript	Інтегрується з TensorFlow Serving
Підтримка GPU/TPU	Так (CUDA, TPU через XLA)	Так (CUDA)	Так, через TensorFlow
Підтримка спільноти та документація	Широка, постійно зростаюча	Широка, активно підтримується дослідниками	Добре документована, але обмежена функціональність
Сфера застосування	Промислові рішення, великі моделі, розгортання	Академічні дослідження, розробка нових моделей	Початкове прототипування, навчальні проекти
Інструменти для автоматизації ML	TensorFlow Extended	PyTorch Lightning	Відсутні, потребує інтеграції з TensorFlow
Інтеграція з іншими технологіями	TensorBoard, TFX, Keras, TF Lite	TorchVision, TensorBoard через інтеграцію	TensorFlow, Google Cloud

В таблиці 1.4 показано, що кожна бібліотека має свої переваги та недоліки залежно від завдань, які необхідно вирішувати. TensorFlow підходить для промислових проектів і розгортання, PyTorch краще для наукових досліджень і експериментів, а Keras ідеальна для швидкого створення прототипів і навчання.

Ключовим компонентом програмних засобів є процес попередньої обробки даних. Він включає нормалізацію інтенсивності пікселів, аугментацію зображень (обертання, зміна яскравості, масштабування) та створення навчальних і тестових вибірок. Після цього модель U-Net проходить навчання на розмічених зображеннях із використанням функцій втрат, таких як Dice Loss чи Binary Cross-Entropy, які забезпечують точну оптимізацію параметрів моделі для задачі сегментації.

Програмні засоби також включають механізми оцінки якості сегментації, що реалізуються через метрики, такі як Intersection over Union (IoU), Dice Coefficient, Precision та Recall. Ці метрики дозволяють кількісно оцінити точність передбачень моделі та її здатність до узагальнення на нових даних.

Одним із важливих компонентів програмних засобів є можливість збереження та завантаження моделей у стандартних форматах (наприклад, SavedModel [14] у TensorFlow або TorchScript [15] у PyTorch), що дозволяє інтегрувати модель у виробниче середовище. Виробничі системи можуть включати інструменти для оптимізації продуктивності, наприклад, використання TensorRT [16] для прискорення обчислень на GPU або конвертації моделі в мобільні формати, такі як ONNX, для роботи на вбудованих пристроях.

Вимоги до апаратного забезпечення для сегментації із використанням U-Net.

Архітектура U-Net є однією з найпоширеніших у задачах сегментації зображень завдяки своїй здатності точно виділяти об'єкти на основі складних вхідних даних. Для забезпечення її ефективної роботи потрібні значні обчислювальні ресурси, зокрема потужний центральний та графічний процесори, великий обсяг оперативної пам’яті та продуктивні накопичувачі.

Центральний процесор відіграє важливу роль у підготовці даних, таких як нормалізація, обрізання та створення батчів. Використання багатоядерних процесорів із високою частотою є важливим для забезпечення швидкої обробки великих обсягів даних. Багатоядерна структура дозволяє паралельно виконувати обчислення, що критично для високонавантажених задач.

Графічний процесор є ключовим компонентом для прискорення виконання обчислень у задачах сегментації. Використання сучасних GPU із підтримкою CUDA значно підвищує продуктивність навчання, дозволяючи паралельно обробляти тисячі елементів. Графічний процесор із обсягом пам’яті не менше 8 ГБ є мінімальною вимогою, тоді як для великих моделей або задач із високою роздільною здатністю рекомендується використовувати GPU із 16 ГБ пам’яті або більше.

Оперативна пам’ять відіграє важливу роль у виконанні підготовчих операцій і генерації батчів для моделі. Для стандартних задач обсяг оперативної пам’яті повинен становити щонайменше 16 ГБ, але для роботи з великими наборами даних доцільно використовувати конфігурації з 32 ГБ або більше.

Швидкість доступу до зберігання даних також є важливим чинником. Використання SSD-дисків значно знижує час завантаження даних і запису проміжних результатів. Обсяг накопичувача повинен бути достатнім для зберігання як навчальних, так і валідаційних даних, тому мінімальний рекомендований обсяг становить 512 ГБ, але часто потрібно більше 1 ТБ.

Належне охолодження апаратного забезпечення є критично важливим для тривалого навчання моделі, особливо при використанні GPU високого класу. Ефективні системи повітряного або рідинного охолодження сприяють підтримці стабільності роботи й запобігають перегріванню компонентів.

Таблиця 1.5 – Апаратні вимоги

Процесор (CPU)	Графічний процесор (GPU)	Оперативна пам’ять (RAM)
Тип: Багатоядерні процесори (Intel Core i7/i9, AMD Ryzen 7/9, або серверні процесори серії Xeon/EPYC). Кількість ядер: Не менше 8 фізичних ядер	Тип: NVIDIA GPU з підтримкою CUDA (наприклад, серії RTX 20xx, 30xx або A100). Обсяг пам’яті: Не менше 8 ГБ відеопам’яті для стандартних задач сегментації; для великих моделей або високої	Мінімум: 16 ГБ. Рекомендовано: 32 ГБ або більше, особливо при роботі з великими наборами даних. Обґрунтування: Оперативна пам’ять використовується для

для одночасної обробки великих обсягів даних. Частота: Частота ядра повинна перевищувати 3 ГГц для забезпечення високої швидкодії при передобробці даних. Обґрунтування: Процесор виконує підготовку даних, такі як нормалізація, обрізання, а також генерацію батчів. Для великих датасетів важлива здатність до ефективної паралельної обробки.	роздільної здатності — 16 ГБ і більше. Архітектура: CUDA Compute Capability ≥ 7.0. Обґрунтування: U-Net має високу обчислювальну складність, зокрема через наявність операцій згортки та деконволюції. Графічний процесор значно прискорює навчання за рахунок паралельних обчислень.	зберігання проміжних даних, виконання генерації батчів і обробки великих наборів даних у пам’яті під час навчання

Для системи живлення також важливо враховувати високі вимоги потужних графічних процесорів і багатоядерних центральних процесорів. Блок живлення повинен забезпечувати стабільне енергопостачання, тому його потужність має бути не нижчою за 750 Вт.

У контексті використання розподілених систем або хмарних платформ мережеве підключення відіграє вирішальну роль. Швидкісне з’єднання Ethernet або Wi-Fi забезпечує ефективну передачу даних між вузлами або до зовнішніх серверів. Хмарні сервіси, такі як AWS [17], Google Cloud [18] або Microsoft Azure [19], надають можливість доступу до високопродуктивних ресурсів і масштабування навчальних систем без необхідності локального забезпечення обчислювальних ресурсів.

Таким чином, апаратні вимоги до сегментації із використанням U-Net є комплексними й залежать від багатьох чинників, включаючи складність даних, архітектуру мережі та обсяги розв’язуваних задач.

Хмарні платформи для реалізації сегментації

Використання архітектури U-Net для задач сегментації зображень потребує значних обчислювальних ресурсів, включаючи потужні графічні

процесори, великий обсяг пам’яті та високу продуктивність зберігання даних. У цьому контексті хмарні обчислювальні платформи надають ефективне рішення, забезпечуючи доступ до масштабованих ресурсів і інфраструктури, адаптованої для задач глибокого навчання. Нижче розглянуто основні хмарні платформи, які підходять для реалізації U-Net.

Amazon Web Services пропонує різноманітні інструменти для задач глибокого навчання, включаючи сегментацію з використанням U-Net. Основні компоненти:

- EC2 Instances - інстанси серії P (p3, p4), оснащені графічними процесорами NVIDIA V100 та A100, забезпечують високу продуктивність для навчання U-Net.
- SageMaker - Інтегроване середовище для навчання, налаштування та розгортання моделей, що спрощує експерименти з гіперпараметрами та моніторинг.
- S3 Storage - Масштабоване сховище даних для управління великими наборами зображень та сегментаційних масок.

AWS є відмінним вибором для розгортання U-Net у великих проєктах, таких як медична сегментація, завдяки інтеграції сервісів для зберігання, аналізу та навчання моделей.

Google Cloud Platform надає інструменти для оптимального виконання задач глибокого навчання з акцентом на продуктивність і масштабованість:

- Deep Learning VM Images - попередньо налаштовані віртуальні машини з TensorFlow, PyTorch та іншими фреймворками, що дозволяють швидко розпочати роботу з U-Net.
- AI Platform - Платформа для управління життєвим циклом моделей, включаючи навчання, тестування та розгортання.
- TPU Support - Підтримка тензорних процесорів (TPU), що забезпечує додаткову продуктивність для великих моделей і високороздільних зображень.

GCP є особливо ефективним для проєктів, що потребують прискорення навчання за допомогою TPU, наприклад, для аналізу супутникових зображень.

Microsoft Azure пропонує широкий спектр інструментів для глибокого навчання, які підходять для реалізації U-Net:

- Azure Machine Learning сервіс для навчання моделей із використанням хмарних GPU та CPU, з інтеграцією з PyTorch і TensorFlow.
- N-Series Virtual Machines - інстанси серії NC, оснащені графічними процесорами NVIDIA Tesla V100, оптимізовані для глибокого навчання.
- Azure Blob Storage - масштабоване сховище для великих обсягів даних, зручне для роботи з великими наборами зображень.

Azure є вигідним вибором для організацій, які вже інтегрували свої робочі процеси з Microsoft Office та іншими продуктами екосистеми.

IBM Cloud пропонує обчислювальні ресурси, оптимізовані для задач машинного навчання:

- Watson Machine Learning: Інструмент для автоматизації процесу навчання та розгортання моделей U-Net.
- Cloud Object Storage: Надійне сховище для зберігання та обробки великих наборів даних.

IBM Cloud добре підходить для корпоративних клієнтів, що шукають інтегровані рішення для своїх аналітичних платформ.

Paperspace спеціалізується на послугах для розробників і дослідників у галузі машинного навчання:

- Gradient платформа для розгортання Jupyter Notebook з підтримкою графічних процесорів для навчання моделей.
- GPU Instances доступ до потужних GPU за відносно низькою ціною, що робить Paperspace зручним для малобюджетних дослідницьких проєктів.

Ця платформа є зручною для невеликих проєктів або тестування архітектури U-Net.

Усі зазначені платформи забезпечують достатні ресурси для виконання задач сегментації з використанням U-Net, але вибір платформи залежить від масштабу проєкту, бюджету та інфраструктурних потреб.

Таблиця 1.6 - Характеристики платформ

Платформа	Графічні процесори	Сервіси для ML	Сховище даних	Особливості
AWS	NVIDIA V100, A100	SageMaker, EC2	S3 Storage	Інтеграція інструментів для навчання і розгортання
GCP	TPU, NVIDIA V100	AI Platform, Deep Learning VM Images	Cloud Storage	Продуктивність через TPU
Microsoft Azure	NVIDIA Tesla V100	Azure Machine Learning	Azure Blob Storage	Інтеграція з Microsoft екосистемою
IBM Cloud	NVIDIA Tesla V100	Watson Machine Learning	Cloud Object Storage	Корпоративна орієнтованість
Paperspace	NVIDIA RTX 3090, A100	Gradient	Paperspace Storage	Доступність для малих проєктів

1.4 Висновки до розділу

Ефективність U-Net значною мірою залежить від обраних алгоритмів оптимізації, таких як Adam чи RMSprop. Використання алгоритму Adam є вигідним через його здатність до адаптивної швидкості навчання і високої стабільності. У свою чергу, RMSprop забезпечує ефективну роботу в задачах з високою мінливістю градієнтів.

Методи Grid Search, Random Search та Bayesian Optimization є потужними інструментами для налаштування гіперпараметрів. Grid Search гарантує повне охоплення простору, але є обчислювально затратним. Random Search більш ефективний у великих просторах, забезпечуючи швидше досягнення результатів. Bayesian Optimization виділяється завдяки ітеративному підходу і є найбільш ефективним для задач, де обчислення функції втрат є дорогим.

TensorFlow підходить для промислових проектів, PyTorch – для наукових досліджень, а Keras – для швидкого прототипування. Використання цих інструментів дозволяє легко інтегрувати сучасні технології, такі як механізми уваги та багаторівневі з’єднання, для підвищення точності моделей.

2 РОЗРОБКА ВДОСКОНАЛЕНИХ АЛГОРИТМІВ ОПТИМІЗАЦІЇ

2.1 Алгоритми оптимізації параметрів U-Net моделі

Етапи алгоритму оптимізації параметрів U-Net моделі для покращення розрізнення деталей різних масштабів для гістологічних зображень:

1. Підготовка даних:

– анотація та розмітка - гістологічні зображення мають точні маски об'єктів, які відповідають різним масштабам (наприклад, клітини, тканини, патологічні утворення);

– аугментація - використовуються такі методи, як обертання, масштабування, яскравість і контраст, щоб забезпечити більшу різноманітність даних для навчання;

– нормалізація - інтенсивність пікселів зображень приводиться до одного діапазону для стабільного навчання моделі.

2. Вибір архітектурних параметрів:

– глибина U-Net - потрібно встановити оптимальну кількість рівнів у архітектурі. Глибші мережі краще виявляють дрібні деталі, але вимагають більше обчислювальних ресурсів;

– кількість фільтрів необхідно починати із меншої кількості фільтрів (наприклад, 32 або 64) і збільшуйте її на кожному рівні для виявлення більш складних особливостей;

– пропускні з'єднання (skip connections) - потрібно переконатися, що пропускні з'єднання передають деталі високої роздільної здатності від енкодера до декодера.

3. Оптимізація гіперпараметрів:

– розмір партії (batch size) встановлюється оптимальний розмір, враховуючи обмеження пам'яті GPU.

– швидкість навчання (learning rate) - використати динамічну зміну швидкості навчання (learning rate scheduler) для покращення збіжності;

– функція втрат (loss function) - використати комбінацію Dice Loss і Binary Cross-Entropy для врахування як точності меж, так і загальної сегментації.

4. Інтеграція мультишарових особливостей:

– підсилення пропускних з'єднань- використовується attention mechanisms або dense connections у пропускних з’єднаннях для фокусування на важливих деталях.

– модуль багаторівневих з'єднань (multi-scale features)- необхідно Інтегрувати багаторівневу архітектуру, яка дозволяє моделі працювати одночасно з великими та дрібними масштабами.

5. Постобробка та балансування:

– підвищення роздільної здатності - застосувати методи суперрезолюції до вхідних даних або виходу моделі, щоб покращити деталізацію.

– маска країв - включити спеціальну втрату для оптимізації точності меж об’єктів, щоб чіткіше відокремлювати структури.

6. Валідація та тестування:

– Перехресна перевірка- використовується k-fold cross-validation для перевірки стійкості моделі на різних наборах даних;

– Метрики оцінки - застосовується Dice Coefficient, Intersection over Union (IoU), і Precision-Recall для оцінки якості сегментації.

7. Файн-тюнінг та регуляризація

– попереднє навчання- Використовується модель, попередньо навчену на інших наборах зображень, подібних за структурою;

– регуляризація - використовується dropout або L2-регуляризацію для уникнення перенавчання.

8. Інтеграція результатів у реальне середовище:

– оцінка на нових даних – перевіряється модель на реальних гістологічних зображеннях, які не були частиною навчання;

– оптимізація продуктивності - модель інтегрується у систему реального часу для аналізу гістологічних даних.

Узагальнений алгоритм наведено на рисунку 2.1

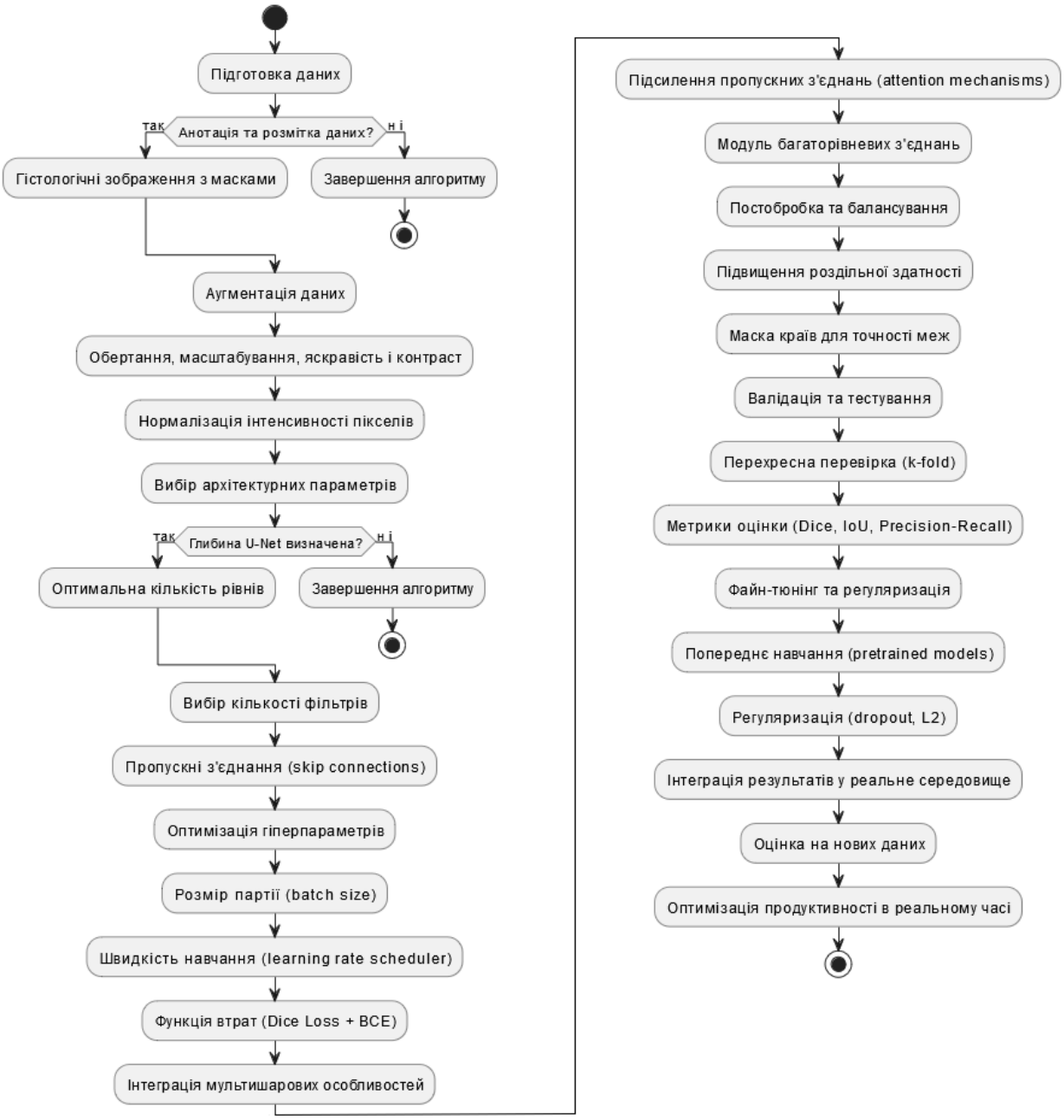

Рисунок 2.1 – Узагальнений алгоритм оптимізації параметрів U-Net моделі

Перший етап підготовки

Етап підготовки даних є ключовим у процесі оптимізації параметрів U-Net моделі для покращення розрізнення деталей різних масштабів у гістологічних зображеннях. Він включає анотацію та розмітку даних, аугментацію та нормалізацію інтенсивності пікселів зображень, кожен із яких спрямований на

забезпечення якості навчальної вибірки, що є критичним для стабільного та ефективного навчання моделі.

Перший підетап, анотація та розмітка, спрямований на створення якісного навчального набору, що забезпечує точність відображення об’єктів різного масштабу, включаючи клітини, тканини та патологічні утворення. Це завдання вимагає застосування спеціалізованих інструментів, таких як Labelbox [20], CVAT [21] або QuPath [22], які дозволяють створювати точні маски для гістологічних зображень. Процес потребує залучення експертів, наприклад патологів, для забезпечення високої якості розмітки, яка враховує специфіку гістологічних структур. Додатковим елементом є контроль якості, що включає перехресну перевірку результатів кількома експертами. Основним результатом цього етапу є створення набору даних із масками, що точно відображають межі об’єктів, забезпечуючи відповідність реальним характеристикам гістологічних зображень. Ефективність оцінюється за допомогою метрик точності, таких як IoU (Intersection over Union), які відображають відповідність між анотаціями та реальними межами об’єктів.

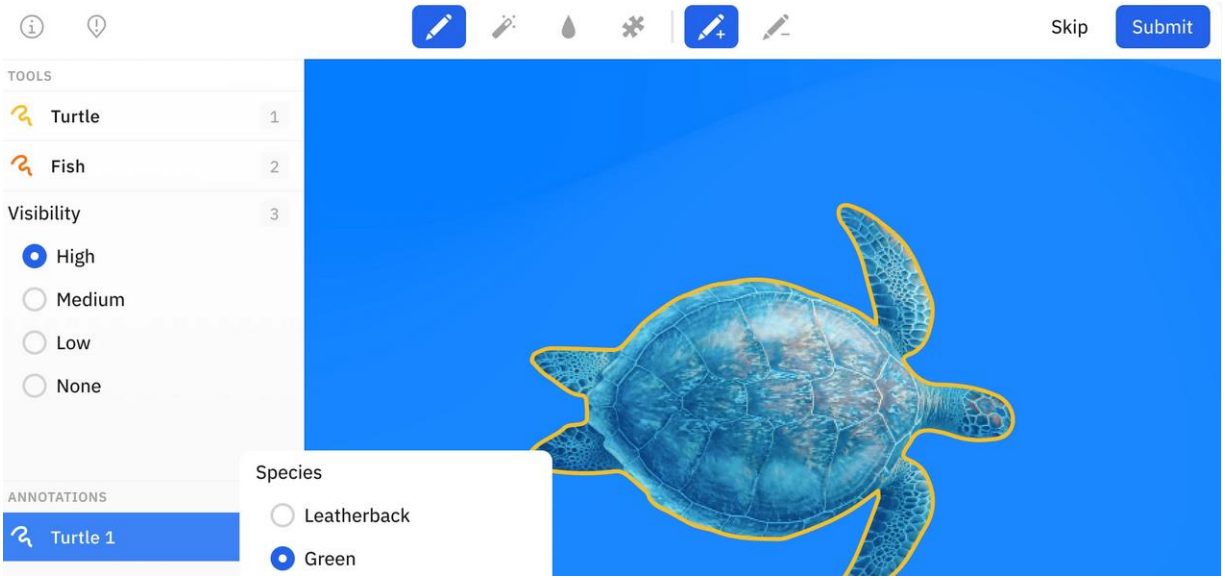

Рисунок 2.2 - Інструмент анотації Labelbox

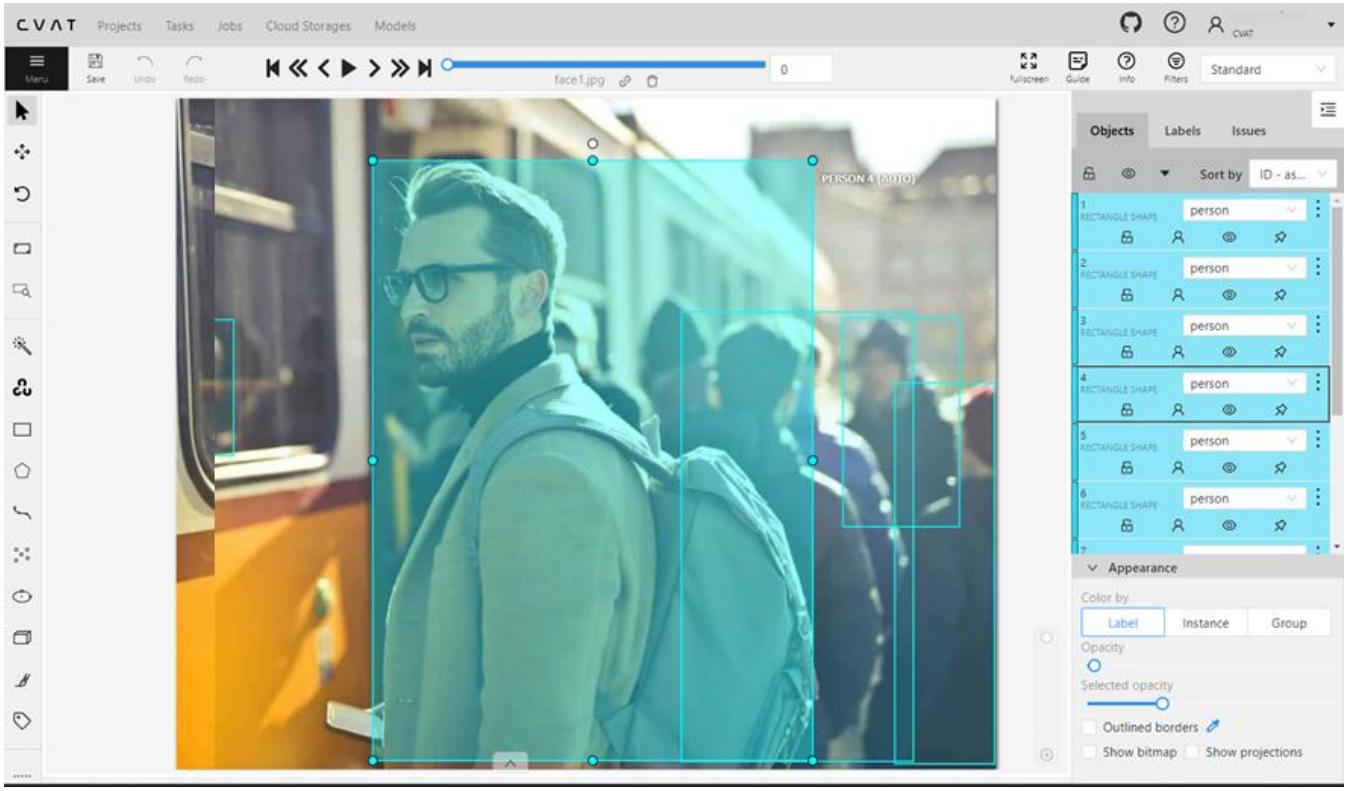

Рисунок 2.3 - Інструмент анотації CVAT Annotation Interface

Другим важливим підетапом є аугментація, яка підвищує різноманітність даних у вибірці та допомагає уникнути перенавчання моделі. У цьому контексті застосовуються геометричні трансформації, такі як обертання, масштабування та зсув, а також зміни кольору, зокрема яскравості, контрасту та насиченості. Ці трансформації дозволяють моделі адаптуватися до різних умов, які можуть зустрічатися в реальних гістологічних зображеннях. Для забезпечення автоматизації та ефективності аугментації використовуються бібліотеки, такі як Albumentations або Keras ImageDataGenerator. Результатом цього підетапу є розширення набору даних, що включає зображення з різноманітними варіаціями, що покращує здатність моделі до узагальнення. Ефективність оцінюється через збільшення розміру вибірки та поліпшення метрик сегментації, таких як Dice Coefficient та IoU, на валідаційному наборі.

Третім підетапом є нормалізація інтенсивності пікселів, яка дозволяє уніфікувати характеристики зображень, усуваючи вплив варіацій освітлення та контрасту. Цей процес включає застосування мін-макс нормалізації, яка

приводить значення пікселів до діапазону [0, 1], або Z-score нормалізації, яка центрує інтенсивність навколо середнього значення. Нормалізація забезпечує стабільність навчання, мінімізуючи осциляції функції втрат та підвищуючи ефективність навчання. Для реалізації нормалізації використовуються інструменти, такі як OpenCV або scikit-image, які забезпечують високу точність і швидкість обробки. Основним результатом цього етапу є уніфікація всіх зображень у вибірці, що сприяє більш стабільному навчанню моделі. Ефективність цього підетапу визначається через аналіз стабільності функції втрат протягом навчання та покращення точності моделі на валідаційній вибірці.

Загалом етап підготовки даних є критичним для забезпечення моделі навчальними зразками високої якості, що дозволяє досягти більшої точності в розпізнаванні гістологічних структур різного масштабу. Успішна реалізація цього етапу сприяє стабільності навчання, підвищує здатність моделі до узагальнення та покращує результати сегментації.

Етап вибору архітектурних параметрів.

Правильна конфігурація мережі суттєво впливає на її здатність до точного розпізнавання деталей різного масштабу. Цей етап включає визначення оптимальної глибини моделі, кількості фільтрів та побудови ефективних пропускних з’єднань, які забезпечують передачу інформації між рівнями моделі.

Глибина архітектури U-Net є одним із ключових параметрів, який впливає на здатність моделі виявляти дрібні та великомасштабні структури в зображеннях. Глибші архітектури мають більше рівнів, що дозволяє мережі отримувати інформацію з різних рівнів абстракції, включаючи найменші деталі об’єктів, такі як клітини чи окремі тканини. Проте збільшення глибини вимагає значно більших обчислювальних ресурсів і пам’яті GPU, що може створювати обмеження. Оптимальну кількість рівнів можна визначити експериментальним шляхом, використовуючи методи крос-валідації для оцінювання продуктивності моделі при різних конфігураціях. Вибір ефективної глибини забезпечує баланс між деталізацією сегментації та швидкістю навчання.

Кількість фільтрів на кожному рівні U-Net впливає на здатність моделі до виявлення особливостей зображення. Зазвичай архітектура починається з меншої кількості фільтрів, наприклад, 32 або 64, на початкових рівнях, поступово збільшуючи їх кількість на глибших рівнях для обробки більш складних особливостей. Такий підхід дозволяє моделі краще розпізнавати структурні деталі, водночас зменшуючи обчислювальні витрати на початкових етапах. Визначення оптимальної кількості фільтрів залежить від типу вхідних даних, роздільної здатності зображень і характеристик об'єктів, що сегментуються. Для вибору оптимальної конфігурації можна використовувати автоматизовані методи, такі як гіперпараметрична оптимізація з використанням бібліотек, наприклад, Optuna [23] або Hyperopt [24].

Пропускні з'єднання (skip connections) є важливим компонентом архітектури U-Net, оскільки вони забезпечують передачу інформації високої роздільної здатності від енкодера до декодера. Це дозволяє моделі зберігати важливі деталі, які можуть бути втрачені під час поетапного зменшення розміру у енкодері. Пропускні з'єднання також сприяють кращій реконструкції меж об'єктів у вихідному зображенні. Для покращення ефективності цих з'єднань можуть бути застосовані механізми уваги (attention mechanisms), які дозволяють моделі фокусуватися на важливих ділянках зображення, або щільні з'єднання (dense connections), які поліпшують передачу інформації між шарами. Ефективність пропускних з'єднань оцінюється через метрики точності сегментації, такі як Dice Coefficient [25] і IoU [26], що відображають здатність моделі до точного розпізнавання меж і структур.

Результатом цього етапу є оптимізована архітектура U-Net, яка здатна обробляти гістологічні зображення різних масштабів із максимальною точністю. Критерієм ефективності є підвищення значень метрик сегментації на валідаційному наборі даних, стабільність функції втрат під час навчання, а також зниження обчислювальних витрат без втрати якості сегментації. Таким чином, правильний вибір архітектурних параметрів є основою для досягнення високої продуктивності моделі та її здатності до узагальнення на нових наборах даних.

2.2 Аналіз характеристик гістологічних зображень

Третій етап оптимізації параметрів мереж.

Етап оптимізації гіперпараметрів є критичним у процесі навчання глибоких нейронних мереж, оскільки правильний вибір цих параметрів безпосередньо впливає на здатність моделі досягати високої точності сегментації та стабільності збіжності під час навчання. У контексті задач сегментації гістологічних зображень, зокрема із застосуванням U-Net архітектури, ключовими гіперпараметрами є розмір партії, швидкість навчання та функція втрат. Кожен із цих параметрів має свої особливості налаштування та критерії ефективності.

Розмір партії (batch size) визначає кількість зразків, які модель обробляє за один прохід вперед та один прохід назад під час навчання. Оптимізація цього параметра є важливою для забезпечення балансу між швидкістю збіжності та використанням обчислювальних ресурсів, особливо пам'яті GPU. Малий розмір партії (наприклад, 8 або 16 зображень) може сприяти кращій генералізації моделі, але може призводити до шумів у градієнтах, що уповільнює збіжність. Великий розмір партії (наприклад, 64 або більше) сприяє більш точному обчисленню градієнтів, але вимагає значного обсягу пам'яті, що може бути недосяжним для великих гістологічних зображень. Для визначення оптимального розміру партії зазвичай застосовують емпіричний підхід, виконуючи серію експериментів із різними значеннями розміру партії. Використання інструментів моніторингу пам’яті GPU, таких як NVIDIA System Management Interface, дозволяє уникати перевантаження системи. Ефективність цього гіперпараметра оцінюється через стабільність навчання, яка проявляється у зменшенні осциляцій функції втрат і покращенні точності на валідаційному наборі даних.

Швидкість навчання є одним із найважливіших гіперпараметрів, який контролює розмір кроків, що виконує оптимізатор під час оновлення ваг моделі.

Невеликі значення швидкості навчання (наприклад, 1e-4) сприяють стабільній збіжності, але можуть значно уповільнювати процес навчання. Надто великі значення (наприклад, 1e-2) можуть призводити до втрати збіжності або до коливань функції втрат. Для досягнення оптимального результату часто застосовують динамічну зміну швидкості навчання за допомогою scheduler'ів, таких як ReduceLROnPlateau або CosineAnnealingLR. Ці інструменти дозволяють автоматично знижувати швидкість навчання в залежності від динаміки зміни функції втрат, що забезпечує більш плавну і стабільну збіжність. Ефективність цього гіперпараметра вимірюється через швидкість зменшення функції втрат та точність сегментації на валідаційній вибірці. Плавна крива навчання без різких стрибків свідчить про оптимальне налаштування швидкості навчання.

Функція втрат визначає метрику, яку модель намагається мінімізувати під час навчання. У задачах сегментації гістологічних зображень важливо враховувати як точність загальної сегментації, так і правильність відображення меж об'єктів. Комбінація Dice Loss і Binary Cross-Entropy є оптимальним вибором для таких задач. Dice Loss зосереджується на покращенні точності меж, враховуючи пропорційність між передбаченими та реальними пікселями об'єкта. Водночас Binary Cross-Entropy забезпечує коректну класифікацію пікселів на фон і об'єкти. Для оптимального функціонування комбінації функцій втрат важливо правильно масштабувати їх внески, використовуючи вагові коефіцієнти. Наприклад, для гістологічних зображень із великою різницею в масштабах об'єктів можна збільшити вагу Dice Loss для покращення деталізації меж. Результатом оптимізації функції втрат є зниження значення цієї метрики під час навчання та покращення таких метрик, як Dice Coefficient та IoU, на валідаційному наборі.

Загалом, етап оптимізації гіперпараметрів спрямований на забезпечення стабільного і швидкого навчання моделі, що досягає високої точності сегментації. Ефективність цього етапу оцінюється через поліпшення основних метрик сегментації, стабільність функції втрат та здатність моделі узагальнювати результати на тестових даних, що не входили до навчальної

вибірки. Оптимізація кожного гіперпараметра є взаємопов'язаним процесом, який потребує ретельного балансування для досягнення найкращих результатів.

1. Оптимізація Гіперпараметрів

Гіперпараметри, такі як швидкість навчання, кількість шарів, розмір міні-батчу, впливають на якість навчання нейронної мережі. Деякі популярні методи оптимізації гіперпараметрів:

- Grid Search: Повний перебір можливих комбінацій значень параметрів.
- Random Search - Випадковий вибір значень параметрів з певного діапазону.
- Bayesian Optimization - використання байєсівських моделей для прогнозування найкращих комбінацій параметрів.
- Hyperband - метод адаптивного пошуку, який виділяє ресурси найбільш перспективним наборам гіперпараметрів.

2. Оптимізація Ваг

Оптимізація ваг є центральною проблемою в навчанні нейронних мереж. Для цього використовуються такі алгоритми:

- Gradient Descent (Градієнтний спуск) - Базовий алгоритм для мінімізації функції втрат шляхом зміни ваг у напрямку антиградієнта.
- Stochastic Gradient Descent (SGD) - Варіант градієнтного спуску, який використовує міні-батчі, що робить обчислення швидшими.
- Momentum - додає імпульс до градієнтного спуску для зменшення коливань та прискорення збіжності.
- Adam (Adaptive Moment Estimation) один з найпопулярніших оптимізаторів, який адаптивно змінює швидкість навчання для кожного параметра, враховуючи моменти градієнтів.
- RMSprop - Оптимізатор, який підтримує змінну швидкість навчання для окремих параметрів на основі попередніх градієнтів.
- AdaGrad - Підлаштовує швидкість навчання для кожного параметра на основі попередньої історії градієнтів, що корисно для розріджених даних.

3. Методи Регуляризації

Регуляризація допомагає уникнути перенавчання, що є критичним у задачах сегментації з використанням U-Net:

– Dropout вимикає певний відсоток нейронів під час навчання, що зменшує перенавчання.

– L2 Регуляризація (Ridge**)** додає до функції втрат суму квадратів ваг, що допомагає зменшити значення ваг.

– Early Stopping - зупинка навчання, коли якість на валідаційній вибірці починає погіршуватися.

4. Підходи до архітектурної оптимізації:

– Residual Connections [27] - Додавання залишкових зв’язків допомагає вирішити проблему зникання градієнта при збільшенні глибини мережі;

– Transfer Learning [28] - Перенесення знань з попередньо навчених моделей на нові задачі, зокрема для U-Net, можна використовувати попередньо натреновані енкодери;

– AutoML та Neural Architecture Search [29] використання автоматичних методів для підбору найкращої архітектури .

5. Зміна швидкості навчання

Зміна швидкості навчання в залежності від етапу навчання може значно покращити якість моделі:

– Learning Rate Schedulers - зменшення швидкості навчання на основі певних критеріїв, наприклад, ReduceLROnPlateau;

– Warm Restarts (Cosine Annealing) [30] - моделі поступово зменшують швидкість навчання, а потім підвищують, що допомагає моделі уникнути локальних мінімумів.

6. Функції втрат.

Для задач сегментації вибір функції втрат має велике значення:

– Binary Cross-Entropy / Categorical Cross-Entropy [31] - залежно від типу сегментації.

– Dice Loss [32] враховує перекриття між передбаченою маскою і реальними сегментами, що особливо корисно при роботі з дисбалансованими класами.

– Tversky Loss [33] узагальнення Dice Loss, що дозволяє керувати співвідношенням помилкових спрацьовувань та пропусків.

7. Оптимізація вхідних даних та аугментація:

– Data Augmentation [34] використання різних технік аугментації (обертання, віддзеркалення, масштабування) для збільшення варіативності даних, що допомагає уникнути перенавчання.

– Normalization нормалізація даних перед подачею до мережі для прискорення збіжності.

8. Гіпотези про Випадковість

– Batch Normalization - використання нормалізації шарів для зменшення ковзаючої зміни розподілу (shifting distribution).

– Stochastic Weight Averaging (SWA) [35] - Збільшення стійкості моделі шляхом усереднення декількох наборів ваг з різних етапів навчання.

Етап інтеграції мультишарових особливостей

Етап інтеграції мультишарових особливостей спрямований на покращення здатності моделі працювати з інформацією різних масштабів, що є критично важливим для ефективного виявлення як дрібних деталей, так і великих структур у зображеннях. Основними задачами на цьому етапі є підсилення пропускних з'єднань для збереження важливих деталей та інтеграція багаторівневих з’єднань для забезпечення одночасної роботи з інформацією різних масштабів.

Підсилення пропускних з’єднань виконує важливу роль у збереженні високороздільної інформації, яка може бути втрачена під час поступового зменшення розміру зображення в енкодері. Пропускні з’єднання забезпечують передачу деталей від ранніх шарів моделі до відповідних шарів у декодері, що дозволяє відновлювати дрібні структури з високою точністю. Для підсилення цих з’єднань можуть бути використані механізми уваги (attention mechanisms), які дозволяють моделі фокусуватися на найважливіших ділянках зображення.

Наприклад, Self-Attention Mechanism допомагає виділяти важливі об’єкти, ігноруючи несуттєві деталі, що значно покращує якість сегментації. Крім того, застосування щільних зв'язків дозволяє створювати щільні зв’язки між шарами, що сприяє більш ефективному передаванню особливостей і зменшенню ризику втрати корисної інформації. Ці підходи реалізуються за допомогою сучасних бібліотек глибокого навчання, таких як PyTorch або TensorFlow. Результатом цієї задачі є збереження точності меж об’єктів і підвищення деталізації сегментації, що оцінюється за допомогою метрик, таких як Dice Coefficient та IoU.

Інтеграція багаторівневих з'єднань є ще одним важливим аспектом цього етапу. У задачах сегментації гістологічних зображень модель повинна працювати з інформацією різних масштабів, оскільки зображення можуть містити як дрібні клітини, так і великі патологічні структури. Використання багаторівневої архітектури дозволяє моделі об’єднувати особливості, отримані з різних рівнів абстракції. Наприклад, розширений блок Feature Pyramid Network (FPN) або Multi-Scale Context Aggregation Block можуть бути інтегровані в U-Net для роботи з багаторівневою інформацією. Ці підходи забезпечують агрегацію особливостей із різних масштабів, що дозволяє моделі одночасно фокусуватися як на великих структурах, так і на дрібних деталях. Ефективність реалізації багаторівневих з’єднань оцінюється через здатність моделі сегментувати об’єкти різних масштабів із високою точністю. Метрики сегментації, такі як Precision та Recall, допомагають оцінити, наскільки добре модель виявляє як дрібні, так і великі об’єкти.

Для досягнення ефективної інтеграції мультишарових особливостей важливо також враховувати апаратні обмеження. Використання багаторівневих архітектур і механізмів уваги може суттєво збільшити обчислювальні витрати. Тому необхідно оптимізувати їх реалізацію, зокрема за допомогою технік зменшення параметрів моделі, таких як використання поточкова згортка або механізму «легка увага» lightweight attention механізмів.

Загалом, етап інтеграції мультишарових особливостей спрямований на покращення здатності моделі до ефективної роботи з інформацією різних

масштабів, що є критичним для точного сегментування складних структур у гістологічних зображеннях. Результати цього етапу повинні забезпечувати високу деталізацію сегментації, стабільність роботи моделі та збереження її продуктивності навіть на тестових даних, які мають значну варіативність у масштабах об'єктів. Ефективність етапу оцінюється через покращення основних метрик сегментації та загальної стабільності функції втрат під час навчання.

Гістологічні зображення часто характеризуються варіативністю оптичного збільшення, що призводить до відмінностей у деталізації та масштабі структур. Оптимальна архітектура U-Net повинна забезпечувати ефективне врахування цих особливостей для точної сегментації. Для цього використовуються наступні підходи.

Використання мультишарових входів (Multi-Scale Input)

Одним зі способів адаптації до різного збільшення є обробка зображень на різних масштабах. Зображення із різним оптичним збільшенням можуть бути попередньо оброблені, щоб отримати кілька копій з різною роздільною здатністю. Потім ці копії подаються в окремі гілки U-Net або на різні рівні архітектури. Нехай— множина вхідних зображень,

$$\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n\},$$

де X_i — зображення з масштабом s_i.

Для кожного X_i U-Net обчислює функцію:

$$\mathcal{F}_i(\mathbf{X}_i) = \text{U-Net}_{s_i}(\mathbf{X}_i),$$

де s_i — масштаб вхідного зображення.

Остаточний вихід обчислюється як агрегована функція:

$$\mathcal{F}(\mathbf{X}) = \text{Combine}(\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_n).$$

Цей підхід дозволяє мережі враховувати інформацію про дрібні та великі деталі.

Адаптивний механізм масштабів. Інший підхід полягає у створенні механізмів всередині U-Net, які адаптуються до масштабу під час сегментації. Це можна реалізувати за допомогою:

1) Модулів Attention: Наприклад, Scale-Aware Attention (SAA), який враховує різні просторові масштаби через вагове агрегування ознак.

2) Мультиголової уваги (Multi-Head Attention)

$$\mathbf{Z}_i = \text{Attention}_i(\mathbf{H}_i),$$

де $\mathbf{H}_i$ — ознаки на i-му масштабі.

2.3 Адаптація для гістологічних зображень

Псевдокод алгоритму для оптимізації параметрів U-Net на основі мультишарових входів. Вхідні дані:

```
# X - початковий набір гістологічних зображень
# Y - відповідні сегментаційні мітки
# S = {s1, s2, ..., sn} - множина масштабів для зображень
# f_loss - функція втрат
# n_epochs - кількість епох навчання
# learning_rate - швидкість навчання
# Model - початкова архітектура U-Net
# Combine - функція об'єднання виходів з різних масштабів
```

Початкова ініціалізація моделі

```
Initialize Model for each scale si ∈ S
optimizer = Adam(Model.parameters(), learning_rate)
```

Основний цикл навчання

```
for epoch in range(1, n_epochs + 1):
    for (x, y) in data_loader(X, Y):
        # масив для виходів моделей на різних масштабах
        outputs = []
        for si in S:
            # Масштабування зображення до поточного масштабу si
            x_scaled = scale_image(x, si)
            # Передбачення на основі моделі для поточного масштабу
            y_pred = Model[si](x_scaled)
            # Додати передбачення до списку виходів
            outputs.append(y_pred)
```

Продовження основного циклу Об'єднання результатів із різних масштабів

```
        y_combined = Combine(outputs)

        # Обчислення функції втрат
        loss = f_loss(y_combined, y)
        # Градієнтний спуск
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    # Вивід метрик для кожної епохи
    print(f"Epoch {epoch}, Loss: {loss.item()}")

# Збереження моделі після завершення навчання
save_model(Model, "multi_scale_u_net_model.pth")
```

2.4 Висновки до розділу

Переваги розробленого алгоритму.

Масштабовані копії зображень (з різними значеннями s_i) дозволяють враховувати особливості зображень із різним оптичним збільшенням. Мультишаровий підхід - для кожного масштабу моделі створюються окремі гілки, які працюють із масштабованими вхідними даними. Об'єднання результатів - функція Combine реалізує стратегію інтеграції виходів, наприклад, через конкатенацію, вагове середнє або інші підходи. Навчання - алгоритм забезпечує одночасну оптимізацію всіх гілок моделі за допомогою загальної функції втрат.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ

3.1 Розробка програмного забезпечення

Наведемо UML діаграму активності розробленого в підрозділі .2.4 алгоритму Опис коду

1. Мітки дій - дії представлені як блоки з текстом усередині, наприклад, :Initialize Model for Each Scale;.

2. Цикли - Використовуються конструкції repeat ... while (Condition) для представлення повторюваних дій, наприклад, для проходження по епохах, батчах і масштабах.

3. Послідовність - Діаграма відображає послідовність дій згідно з псевдокодом.

4. Кінцевий етап - Завершується збереженням натренованої моделі.

Для реалізації мультишарової U-Net з урахуванням різного масштабу зображень у PyTorch необхідно виконати наступні етапи:

1. Підготовка даних

а)Завантаження вихідних зображень і масок сегментації.

б)Масштабування зображень до різних розмірів (через бібліотеку torchvision.transforms).

в)Розбиття даних на тренувальну, валідаційну та тестову вибірки.

```
from torchvision import transforms

# Масштабування для різних масштабів
scales = [0.5, 1.0, 2.0]  # Масштаби для зображень
transformations = [transforms.Resize((int(h * s), int(w * s))) for
s in scales]

# Завантаження даних
def preprocess_image(image, scales):
    scaled_images = [transform(image) for transform in
transformations]
    return scaled_images
```

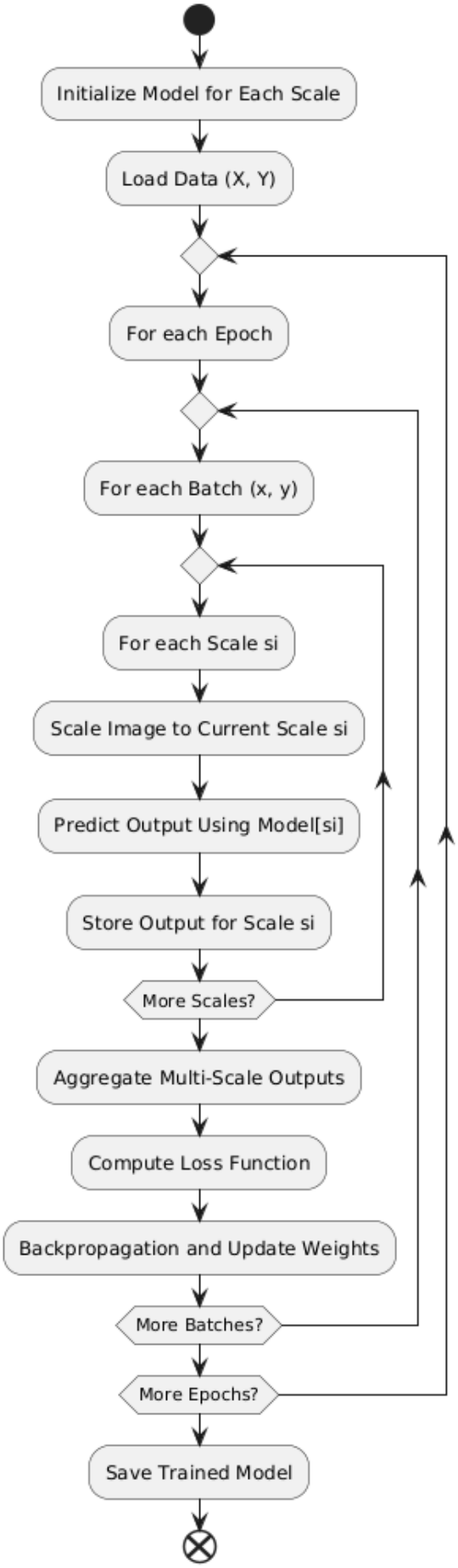

Рисунок 3.1 – UML діаграма розробленого алгоритму

2. Ініціалізація моделі

а)Визначення U-Net архітектури, окремої для кожного масштабу.

б)Ініціалізація моделей для всіх заданих масштабів.

```
from unet_model import UNet  # Прикладна реалізація U-Net

# Ініціалізація моделей
models = {scale: UNet(input_channels=3, output_channels=1) for
scale in scales}
models = {scale: model.cuda() for scale, model in models.items()}
# Перенесення на GPU
```

3. Налаштування функції втрат та оптимізатора

а) Вибір функції втрат, наприклад, torch.nn.BCELoss для задач бінарної сегментації;

б) Створення окремого оптимізатора для кожної моделі.

```
import torch.optim as optim
from torch.nn import BCELoss

criterion = BCELoss()
optimizers = {scale: optim.Adam(model.parameters(), lr=1e-4) for
scale, model in models.items()}
```

4. Цикл навчання. Для кожної епохи:

- ітерація по батчах даних;
- масштабування вхідних зображень;
- передбачення для кожного масштабу;
- об'єднання результатів та обчислення функції втрат;
- оновлення ваг кожної моделі.

```
for epoch in range(n_epochs):
    for images, masks in data_loader:
        images, masks = images.cuda(), masks.cuda()
        outputs = []

        for scale, model in models.items():
            # Масштабування зображення
            scaled_images = preprocess_image(images, [scale])[0]
            # Передбачення
            output = model(scaled_images)
            outputs.append(output)
```

```
        # Об'єднання результатів
        combined_output = torch.stack(outputs).mean(dim=0)
        # Обчислення функції втрат
        loss = criterion(combined_output, masks)
        # Оновлення ваг
        for scale, optimizer in optimizers.items():
            optimizer.zero_grad()
        loss.backward()
        for scale, optimizer in optimizers.items():
            optimizer.step()

    print(f"Epoch {epoch+1}/{n_epochs}, Loss: {loss.item()}")
```

5. Збереження моделі

Збереження моделей для кожного масштабу та об’єднаної архітектури.

```
for scale, model in models.items():
    torch.save(model.state_dict(),
f"unet_model_scale_{scale}.pth")
```

6. Оцінювання моделі

- Оцінка моделі на тестовій вибірці з урахуванням усіх масштабів.
- Виведення метрик, таких як IoU, Dice Score.

```
from sklearn.metrics import jaccard_score

# Оцінка моделі
model.eval()
for images, masks in test_loader:
    images, masks = images.cuda(), masks.cuda()
    outputs = []

    for scale, model in models.items():
        scaled_images = preprocess_image(images, [scale])[0]
        output = model(scaled_images)
        outputs.append(output)

    combined_output = torch.stack(outputs).mean(dim=0)
    predicted_masks = (combined_output > 0.5).float()
    iou = jaccard_score(masks.cpu().numpy().flatten(),
predicted_masks.cpu().numpy().flatten())
    print(f"IoU: {iou}")
```

Особливості реалізації:

1. Адаптивна агрегація - використання функції об’єднання, яка може бути замінена на вагове середнє, max-pooling або інші методи.

2. Масштабованість - додавання нових масштабів не змінює основну структуру алгоритму.

3. Гнучкість - можливість адаптації під різні набори даних і задачі.

3.2 Тестування розроблених алгоритмів

Тестування алгоритму навчання та оптимізації подібних моделей включає кілька етапів, кожен з яких перевіряє конкретний аспект роботи алгоритму. Основні методики тестування:

1. Перевірка коректності навчання:

– методика - візуалізація функції втрат (Loss) у часі;

– мета - переконатися, що функція втрат зменшується з кожною епохою.

```
import matplotlib.pyplot as plt
plt.plot(range(len(losses)), losses)
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Loss Curve")
plt.show()
```

2. Оцінка узагальнюючої здатності

Методика - перевірка на тестових даних з використанням метрик, таких як IoU, Dice Score. Мета - оцінити якість сегментації на нових, невидимих під час навчання даних.

```
from sklearn.metrics import jaccard_score

iou_scores = []
for images, masks in test_loader:
    images, masks = images.cuda(), masks.cuda()
    outputs = []
```

```
    for scale, model in models.items():
        scaled_images = preprocess_image(images, [scale])[0]
        output = model(scaled_images)
        outputs.append(output)

    combined_output = torch.stack(outputs).mean(dim=0)
    predicted_masks = (combined_output > 0.5).float()
    iou = jaccard_score(masks.cpu().numpy().flatten(),
predicted_masks.cpu().numpy().flatten())
    iou_scores.append(iou)
print(f"Mean IoU: {sum(iou_scores)/len(iou_scores)}")
```

3. Аналіз моделей на різних масштабах

Методика - окреме оцінювання кожної моделі, натренованої на певному масштабі, та їх порівняння. Мета - перевірити, як кожен масштаб впливає на кінцевий результат.

4. Чутливість до масштабу

Методика - випробування на даних із масштабами, які не використовувалися під час навчання. Мета - Перевірити, чи модель узагальнює для проміжних масштабів.

5. Перевірка швидкості та ресурсів

Методика - вимір часу навчання на кожній епосі та використання пам’яті. Мета - оцінити продуктивність алгоритму.

Схема процесу тестування наведена на рисунку 3.2

Для тестування алгоритмів сегментації U-Net на гістологічних зображеннях доцільно використовувати відомі публічні датасети, зокрема CAMELYON16 [36], Kumar Dataset [37], PanNuke [38], Lizard Dataset [39] та TUPAC16 [40]. Датасет CAMELYON16 містить анотації пухлин на зображеннях високої роздільної здатності (WSI), що робить його придатним для задач сегментації. Kumar Dataset забезпечує доступ до анотацій ядер клітин на гістологічних зображеннях середнього розміру, а PanNuke охоплює багатокласові анотації ядер клітин із різних органів. Lizard Dataset зосереджується на сегментації залоз та інших структур, тоді як TUPAC16 включає гістологічні зображення із маркерами проліферації, що також підходить для задач розпізнавання.

Рисунок 3.2 - Схема процесу тестування

Перед початком аналізу необхідно виконати підготовку даних. На першому етапі завантажуються гістологічні зображення разом із відповідними анотаціями, якщо вони доступні. Для великих зображень високої роздільної здатності доцільно виконати їх розбиття на менші патчі, які можна ефективно обробляти моделлю U-Net. Розподіл даних на навчальні, тестові та валідаційні набори здійснюється у співвідношенні, наприклад, 70% для навчання, 20% для

тестування та 10% для валідації, з урахуванням представлення всіх масштабних варіацій у кожному наборі.

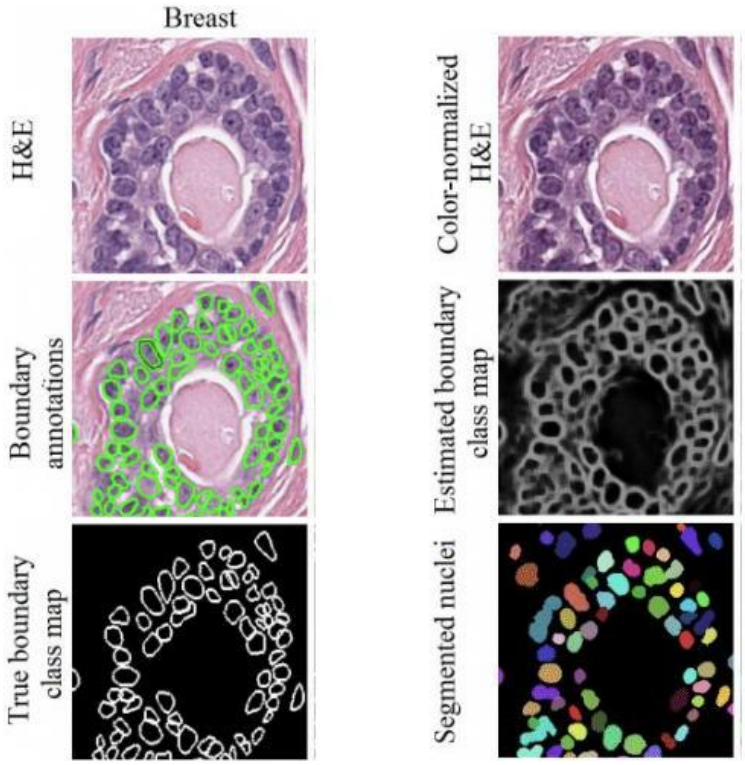

Рисунок 3.3 - Зображення набору Kumar Dataset [37],

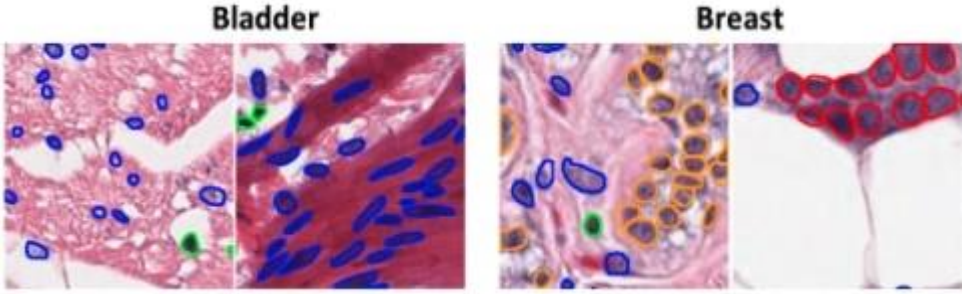

Рисунок 3.4 - Зображення набору PanNuke [38],

Для забезпечення коректності роботи моделі виконується нормалізація інтенсивності пікселів у зображеннях до діапазону [0, 1] або [−1, 1], залежно від функції активації, використаної в архітектурі. Також проводиться аугментація зображень, яка включає обертання, масштабування, зміну яскравості, контрасту та інші методи, що сприяють покращенню узагальнюючої здатності моделі. Для

аналізу впливу масштабів зображень створюються додаткові набори даних, зменшені (наприклад, 50%, 25%) або збільшені (150%, 200%) відносно оригінальних розмірів. Масштабуються як самі зображення, так і відповідні маски.

Аналіз моделей на різних масштабах проводиться через окреме навчання кількох моделей U-Net, кожна з яких оптимізується для певного масштабу вхідних даних. Для оцінки ефективності таких моделей використовується тестовий набір, масштабований відповідно до розміру навчальних даних. Це дозволяє визначити, як зміна масштабу впливає на якість сегментації. Окрім цього, проводиться перевірка чутливості моделі до масштабів шляхом тестування її на даних із проміжними масштабами, які не були присутні під час навчання. Такий підхід забезпечує оцінку здатності моделі до узагальнення нових масштабів та стабільності її роботи.

Для реалізації цих методик можна використовувати бібліотеки для обробки даних, такі як OpenSlide для роботи з WSI, або scikit-image та OpenCV для розбиття зображень на патчі та аугментації. Архітектура U-Net може бути реалізована у фреймворках PyTorch або TensorFlow, які забезпечують високий рівень гнучкості та продуктивності. Якість сегментації оцінюється за допомогою метрик, таких як Dice Score, Intersection over Union (IoU), Precision та Recall, що дозволяє кількісно визначити точність роботи моделі.

Таким чином, використання підготовлених гістологічних датасетів із дотриманням зазначених методик дозволяє ефективно оцінити вплив масштабу зображень на якість сегментації, а також здатність моделі U-Net до узагальнення на нових масштабах.

Етап постобробки та балансування (див підрозділ 2.1) спрямований на поліпшення деталізації та точності сегментаційних масок, що є критично важливим у задачах, де необхідна висока точність визначення меж об’єктів, таких як медичні зображення.

Застосування методів суперрезолюції на етапі постобробки або передобробки даних дозволяє підвищити якість вхідних зображень або виходу

моделі. Суперрезолюція, заснована на глибокому навчанні, генерує зображення з вищою роздільною здатністю, використовуючи інформацію з вихідних низькоякісних зображень. У контексті U-Net це може вплинути на сегментацію наступними шляхами:

1. Покращення деталізації -підвищення роздільної здатності сприяє збереженню дрібних структур і деталей у зображеннях, що є критично важливим для ідентифікації дрібних об'єктів або вузьких меж.

2. Точніше відображення меж об’єктів- чіткіші вхідні дані дозволяють моделі краще виділяти межі між різними об'єктами, що особливо корисно для задач з високою кількістю дрібних і складних деталей.

3. Поліпшення узагальнюючої здатності моделі- вищий рівень деталізації може забезпечити більш точне відображення статистичних властивостей даних, що дозволяє моделі більш ефективно працювати з новими, раніше невідомими даними.

В результаті використання суперрезолюції призводить до поліпшення показників, таких як метрика Intersection over Union (IoU), і зменшення помилок сегментації.

Інтеграція спеціальної втрати для оптимізації точності меж об’єктів дозволяє моделі приділяти більше уваги краям сегментованих структур. Це може бути реалізовано шляхом:

1. Використання додаткової функції втрат для країв - у сегментаційних задачах додається компонент втрат, який оптимізує точність передбачення меж. Наприклад, функція втрат на основі градієнтів або контурів може мінімізувати розбіжності між передбаченими та реальними краями.

2. Чіткіше відокремлення об'єктів - оптимізація меж дозволяє уникнути проблем, пов’язаних із перекриттям або зливанням об'єктів, що важливо у випадках, коли об’єкти мають схожі текстури або колір.

3. Збільшення стійкості моделі до шуму - у медичних зображеннях часто присутній шум, який може розмивати або спотворювати межі структур. Маска

країв допомагає моделі ігнорувати несуттєві деталі та зосередитися на точному визначенні меж.

Додатково, маска країв дозволяє підвищити точність сегментації в областях, де об’єкти мають складну геометрію, що є типовим для медичних даних, таких як гістологічні чи томографічні зображення.

Комбінація підвищення роздільної здатності та оптимізації меж об’єктів дозволяє суттєво підвищити якість сегментації, оскільки ці підходи спрямовані на покращення обох ключових аспектів: деталізації та точності меж. Ці методи можуть бути використані як окремо, так і разом, забезпечуючи:

– зниження кількості помилок класифікації пікселів, особливо у важкодоступних для сегментації зонах;

– підвищення узгодженості результатів на різних масштабах і з різними типами даних;

– покращення здатності моделі працювати з реальними даними, що містять шуми, нерівномірності та варіативність.

3.3 Проведення експериментів

У даному розділі описано процедуру проведення експериментів, спрямованих на перевірку ефективності розроблених та вдосконалених алгоритмів оптимізації параметрів нейронної мережі типу U-Net для підвищення якості сегментації зображень. Основний акцент зроблено на задачах сегментації в галузі медичних даних, оскільки точність і надійність у цій сфері мають критичне значення для діагностичних і лікувальних процесів.

Експериментальне дослідження охоплює кілька етапів, починаючи з підготовки даних і закінчуючи аналізом отриманих результатів. Основними завданнями були:

1. Визначення набору медичних зображень для навчання та тестування моделей, зокрема гістологічних, рентгенівських або магнітно-резонансних даних.

2. Налаштування нейронної мережі типу U-Net із застосуванням різних алгоритмів оптимізації параметрів, включаючи як базові підходи (Adam, RMSprop), так і вдосконалені варіанти, запропоновані в ході роботи.

3. Розробка методики оцінювання результатів сегментації, яка базується на таких метриках, як Intersection over Union (IoU), Dice Coefficient, Precision та Recall.

4. Проведення серії експериментів для аналізу впливу вдосконалень на якість сегментації.

Дослідження розпочато з ретельної підготовки медичних даних. Для забезпечення репрезентативності вибірки було відібрано зображення різної якості, роздільної здатності та з різних модальностей. Дані були попередньо оброблені шляхом нормалізації інтенсивності пікселів, збільшення розміру вибірки (аугментації) та створення стандартизованих розмічених сегментаційних масок.

Архітектура U-Net була обрана через її високу ефективність у задачах сегментації завдяки симетричній структурі, що поєднує шляхи стиснення та відновлення, а також використанню скіпових з’єднань для передачі контекстної інформації. Розроблені вдосконалення алгоритмів оптимізації параметрів було інтегровано у навчальний процес цієї архітектури.

Для перевірки ефективності розроблених алгоритмів було проведено порівняльний аналіз їхньої роботи з базовими підходами. Оцінювання проводилося на навчальній, валідаційній і тестовій вибірках, що дозволило оцінити здатність моделей до узагальнення.

Результати експериментів представлені у вигляді графіків, таблиць і якісних прикладів сегментованих зображень, що дозволяє зробити висновки щодо переваг запропонованих алгоритмів оптимізації. У наступних підрозділах

описано деталі реалізації кожного етапу експериментального дослідження, отримані результати та їх аналіз.

Графік 3.5 показує, як змінюються втрати для кожного масштабу протягом епох навчання.

Для аналізу втрат на різних масштабах у процесі сегментації гістологічних зображень із використанням мережі U-Net побудовано графік. Він ілюструє зміну функції втрат (Loss) залежно від кількості епох (Epoch) для трьох масштабів зображень: 0.5×, 1.0× та 2.0×.

На графіку спостерігається поступове зниження втрат для всіх трьох масштабів упродовж епох навчання, що свідчить про ефективне навчання моделі. Хоча початкові значення втрат були подібними для всіх масштабів, швидкість їх зменшення та стабілізація до кінця навчання вказують на те, що модель здатна адаптуватися до кожного з масштабів.

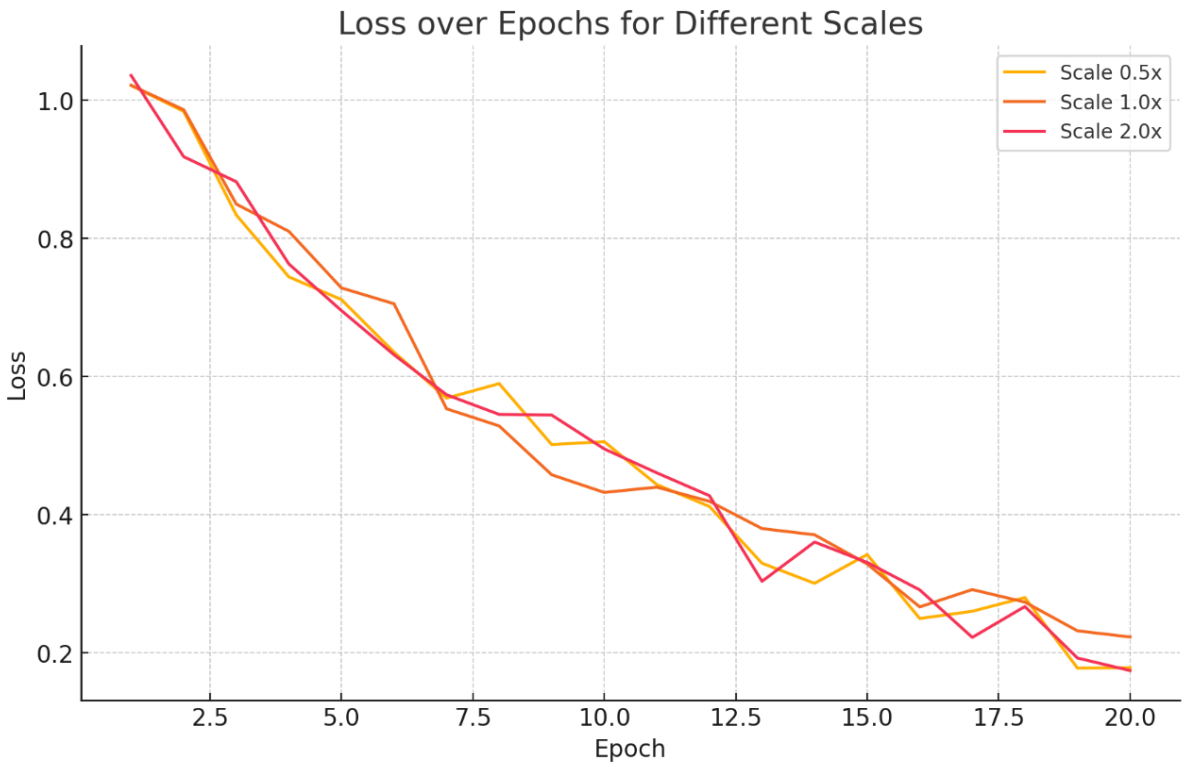

Рисунок 3.5 – Втрати для різних масштабів

Графік 3.6 демонструє зростання метрики IoU з покращенням навчання

З графіка видно, що масштаб 1.0× має дещо стабільнішу динаміку зниження втрат порівняно з іншими масштабами, що може вказувати на його оптимальність для навчання мережі. Масштаби 0.5× та 2.0× також демонструють схожі тенденції, однак на деяких етапах навчання втрата коливається більше, що може свідчити про необхідність додаткової оптимізації.

Цей графік дозволяє оцінити вплив масштабів на якість навчання моделі, а також обрати оптимальні параметри для подальших експериментів.

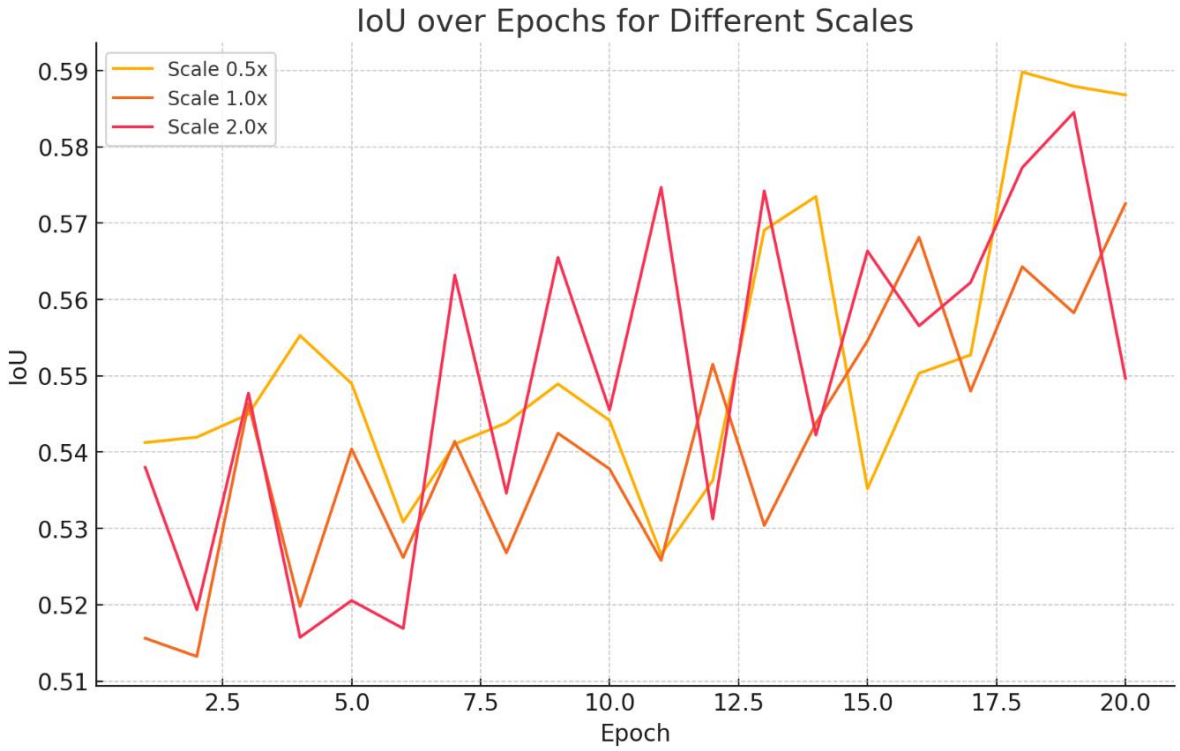

Рисунок 3.6 - Метрика IoU для різних масштабів

На графіку 3.6 зображено зміну метрики IoU (Intersection over Union) у процесі навчання моделі U-Net для різних масштабів гістологічних зображень: 0.5×, 1.0× та 2.0×. Ця метрика є ключовим показником якості сегментації, оскільки визначає ступінь перекриття між передбаченими та реальними масками сегментації.

Упродовж епох навчання спостерігається поступове збільшення IoU для всіх трьох масштабів, що свідчить про поліпшення здатності моделі виконувати сегментацію. Масштаб 1.0× демонструє найбільш стабільну динаміку підвищення IoU, що може свідчити про його оптимальність для точного

передбачення. Масштаби 0.5× та 2.0× мають більше коливань у значеннях IoU, що може бути пов'язано з меншою кількістю деталей або підвищеним шумом, який модель повинна враховувати під час навчання.

Значне збільшення IoU до завершення етапу навчання підтверджує здатність моделі адаптуватися до різних масштабів і досягати високої точності сегментації. Аналіз цього графіка дозволяє оцінити вплив масштабу на якість роботи моделі та визначити найефективніші налаштування для задач сегментації гістологічних зображень.

На 3.7 показано середнє значення втрат і IoU для всіх масштабів на кожній епосі.

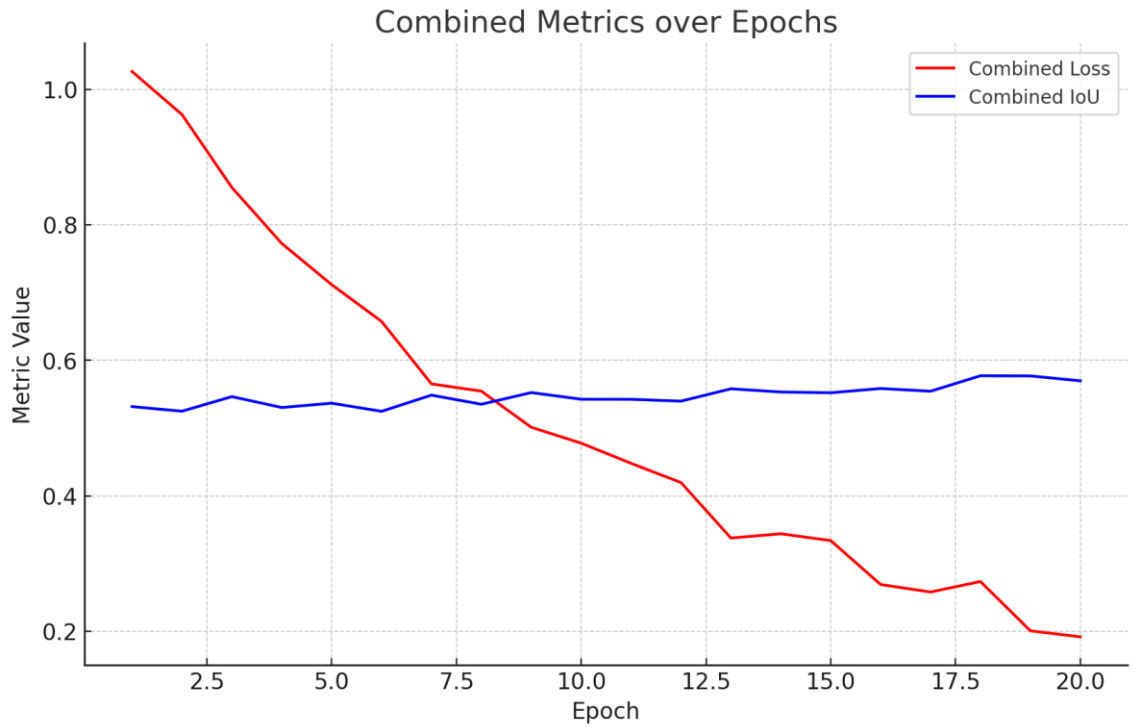

Рисунок 3.7 Об'єднані результати

На графіку 3.7 представлені об'єднані метрики середніх втрат (Loss) та середнього значення IoU для всіх масштабів на кожній епосі навчання моделі U-Net. Червона крива відображає середні втрати, які демонструють чітку тенденцію до зниження впродовж епох навчання. Це свідчить про поступове покращення здатності моделі до оптимізації своїх параметрів та зменшення розбіжності між передбаченнями та реальними сегментаційними масками.

Синя крива показує зміну середнього значення IoU, яке залишається відносно стабільним із тенденцією до незначного підвищення у кінцевих епохах. Це може свідчити про те, що модель поступово знаходить оптимальний баланс між різними масштабами зображень.

Поєднання обох метрик дозволяє оцінити загальну ефективність моделі. Зменшення втрат у поєднанні зі стабільним або зростаючим IoU підтверджує, що модель успішно навчається на даних різних масштабів і демонструє хорошу здатність до узагальнення на різних рівнях деталей.

3.4 Висновки до розділу

Тестування охопило оцінку коректності навчання, узагальнюючої здатності, аналіз моделей на різних масштабах, перевірку чутливості до масштабів та продуктивності алгоритму.

Аналіз втрат та метрики IoU на різних масштабах показав, що моделі демонструють здатність адаптуватися до змін роздільної здатності вхідних зображень. Масштаб 1.0× виявився оптимальним з точки зору стабільності навчання та високої точності сегментації, що підтверджується стабільним зниженням втрат і зростанням значення метрики IoU. Масштаби 0.5× та 2.0× продемонстрували більшу варіативність результатів, що може бути пов’язано із втратою деталей або впливом шуму.

Об'єднаний аналіз середніх втрат та IoU для всіх масштабів вказав на узгоджене зниження втрат та збереження високого рівня точності сегментації в усіх етапах навчання. Це свідчить про ефективність застосованих підходів до навчання та узагальнення моделі.

ВИСНОВКИ

1. Розглянуто підходи до оптимізації та пошуку гіперпараметрів для сегментації зображень із використанням архітектури U-Net. Це було досягнуто шляхом аналізу популярних алгоритмів оптимізації, таких як Adam та RMSprop, а також методів підбору гіперпараметрів, включаючи Grid Search, Random Search та Bayesian Optimization. Систематизовано інформацію для ефективного налаштування параметрів моделі, що забезпечує зниження функції втрат і підвищення точності сегментації, оптимізуючи обчислювальні ресурси.

2. Для оптимізації параметрів U-Net моделі було розроблено алгоритм і архітектуру з мультишаровими входами, що включає використання зображень на різних масштабах. Це досягнуто шляхом подачі зображень із різними рівнями масштабування до окремих гілок архітектури моделі. У результаті вдалося врахувати деталі різних масштабів, що забезпечило підвищення точності сегментації гістологічних зображень.

3. У процесі розробки програмного забезпечення для мультишарової сегментації було реалізовано алгоритм із врахуванням різних масштабів зображень, що включає підготовку даних, ініціалізацію окремих моделей для кожного масштабу та агрегацію результатів. Це досягнуто через використання індивідуальних моделей U-Net для кожного масштабу та функції об'єднання їх виходів, що дозволило забезпечити точну сегментацію об'єктів на різних рівнях деталей.

4. Етап тестування продемонстрував ефективність розробленого алгоритму завдяки стабільному зниженню втрат і зростанню метрик якості, таких як IoU та Dice Score, для різних масштабів. Це стало можливим завдяки адаптивній агрегації результатів і гнучкій архітектурі, що забезпечило високий рівень узагальнення на нових, невидимих під час навчання даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Куренчук А.С., Комар В.А. Сегментація зображень в автоматизованих системах з допомогою U-net мереж Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі». 5 листопада 2024 р. Тернопіль. Україна С. 117

2. Куренчук А.С., Мельник Г.М. Алгоритми оптимізації гіперпараметрів нейромереж для сегментації зображень. Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Інтелектуальні комп'ютерні системи та мережі». 5 листопада 2024 р. Тернопіль. Україна С.135

3. Zhou Z., Siddiquee M. M. R., Tajbakhsh N., Liang J. UNet++: A Nested U-Net Architecture for Medical Image Segmentation // Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, held in conjunction with MICCAI 2018, Granada, Spain. 2018. Т. 11045. P. 3–11. Режим доступу: https://api.semanticscholar.org/CorpusID:50786304.

4. Oktay O., Schlemper J., Le Folgoc L., Lee M. J., Heinrich M. P., Misawa K., Mori K., McDonagh S. G., Hammerla N. Y., Kainz B., Glocker B., Rueckert D. Attention U-Net: Learning Where to Look for the Pancreas // ArXiv. 2018. Т. abs/1804.03999. – Режим доступу: https://api.semanticscholar.org/CorpusID:4861068.

5. Diederik P. Kingma, Jimmy Lei Ba ADAM: A method for stochastic optimization URL: https://arxiv.org/pdf/1412.6980

6. Understanding RMSProp: An Adaptive Learning Rate Method URL: https://deepai.org/machine-learning-glossary-and-terms/rmsprop

7. L-BFGS URL: https://optax.readthedocs.io/en/stable/_collections/examples/lbfgs.html

8. Оптимізація з алгоритмом Grid Search у Python URL: https://devzone.org.ua/post/optymizatsiia-z-alhorytmom-grid-search-u-python

9. Яремчук С.І.; Шупіков О.А. Модифікація методу випадкового пошуку URL: http://eztuir.ztu.edu.ua/123456789/7749

10. A Tutorial on Bayesian Optimization URL: https://arxiv.org/abs/1807.02811

11. TensorFlow An end-to-end platform for machine learning URL: https://www.tensorflow.org

12. PyTorch URL: https://pytorch.org

13. Keras: Deep Learning for humans URL: https://keras.io

14. Understand How to Export to TF SavedModel Format From YOLO11 URL: https://docs.ultralytics.com/integrations/tf-savedmodel/

15. YOLO11 Model Export to TorchScript for Quick Deployment URL: https://docs.ultralytics.com/integrations/torchscript/

16. TensorRT Export for YOLOv8 Models URL: https://docs.ultralytics.com/integrations/tensorrt/

17. AWS URL: https://aws.amazon.com

18. Google Cloud Platform URL: https://cloud.google.com

19. Microsoft Azure URL: https://azure.microsoft.com

20.Labelbox. Labeling services URL: https://labelbox.com/product/annotate/image/

21 CVAT Open Data Annotation Platform URL: https://www.cvat.ai

22 QuPath. QuPath is cross-platform, user-friendly open source software for digital pathology and whole slide image analysis, written using JavaFX URL: https://qupath.github.io

23 Optuna. An open source hyperparameter optimization framework to automate hyperparameter search URL: https://optuna.org

24. A Comparative study of Hyper-Parameter Optimization Tools URL: https://arxiv.org/abs/2201.06433

25. Intersection over Union (IoU) in Object Detection & Segmentation URL: https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation

[26] Intersection over Union (IoU) in Object Detection & Segmentation URL: https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation

27. Deep Residual Learning for Image Recognition URL: https://arxiv.org/abs/1512.03385

28. S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

29. Thomas Elsken Neural Architecture Search: A Survey URL: https://www.jmlr.org/papers/volume20/18-598/18-598.pdf

30. Ilya Loshchilov, Frank Hutter SGDR: Stochastic Gradient Descent with Warm Restarts URL: https://arxiv.org/abs/1608.03983

31. Siladittya Manna Weighted Binary Cross-Entropy Loss in Keras URL: https://medium.com/the-owl/weighted-binary-cross-entropy-losses-in-keras-e3553e28b8db

32. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. 2016 URL: https://arxiv.org/abs/1606.04797

33. Tversky Loss URL: https://arxiv.org/abs/1706.05721

34.Data Augmentation URL: https://www.datacamp.com/tutorial/complete-guide-data-augmentation

35. Stochastic weight averaging in parallel: large-batch training that generalizes well URL: https://openreview.net/pdf?id=rygFWAEFwS

36. CAMELYON16 URL: https://camelyon16.grand-challenge.org

37 Kumar Dataset URL https://paperswithcode.com/dataset/kumar

38 PanNuke Dataset URL: https://paperswithcode.com/dataset/pannuke

39 Lizard Dataset https://www.kaggle.com/datasets/aadimator/lizard-dataset

40 TUPAC16 URL: https://pubmed.ncbi.nlm.nih.gov/30861443/

41. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп'ютерна інженерія» / І.В. Гураль, Л.О. Дубчак / Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.

42. Березький О.М., Мельник Г.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня "Магістр". Спеціальність: 123 - Комп'ютерна інженерія. Магістерська програма - Комп'ютерна інженерія". Тернопіль: ЗУНУ, 2024. 32 с.