ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНИВЕРСИТЕТ

ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ФАКУЛЬТЕТ ЕКОНОМІЧНОЇ ІНФОРМАТИКИ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

Пояснювальна записка

до дипломної роботи

МАГІСТРА

на тему: «Моделювання впливу режимів Distributed Tensorflow на продуктивність вирішення задач сегментації зображень»

Виконав:
студент 2 року навчання
за освітнім ступенем «магістр»
зі спеціальності 122 «Комп'ютерні науки»
Максим НОВІКОВ
Керівник: д.т.н., професор
Сергій МІНУХІН

Харків – 2020 рік

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНИВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

Факультет	Економічної інформатики
Кафедра	Інформаційних систем
Освітній ступінь	Магістр
Спеціальність	122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем
_____ к.е.н., Ірина УШАКОВА
«11» вересня 2020 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Максиму НОВІКОВУ

1. Тема роботи «Моделювання впливу режимів Distributed Tensorflow на продуктивність вирішення задач сегментації зображень»

Керівник роботи: Сергій МІНУХІН , д.т.н., професор

затверджені наказом ректора від « 05 » вересня 2020 року № 838-С

2. Строк подання студентом роботи: « 27 » листопада 2020 року

3. Вихідні дані до роботи: ДСТУ щодо обробки інформації, нормативно правові та законодавчі акти України, фахові періодичні видання з комп'ютерних наук, науково-методичні розробки та результати, оприлюднені в працях вітчизняних та зарубіжних авторів, літературні джерела.

4. Зміст розрахунково-пояснювальної записки:

Розділ 1. Аналіз методів та моделей сегментації зображень

Розділ 2. Аналіз сучасних моделей сегментації зображень та використання фреймворка Tensorflow

Розділ 3. Експериментальні дослідження сегментації зображень на прикладі ультразвукових зображень плечового нерву

5. Перелік графічного матеріалу:схеми та діаграми результатів дослідження

6. Дата видачі завдання: 11 вересня 2020 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану дипломної роботи, ознайомлення з літературними джерелами за темою	12.09.2020	
2.	Аналіз існуючих методів розв'язку поставлених завдань, підготовка теоретичної частини дипломної роботи	28.09.2020	
3.	Розроблення методу (моделі) розв'язку поставлених завдань з тестовим прикладом його застосування	21.10.2020	
4.	Проведення експериментів з описом, обґрунтуванням та аналізом отриманих результатів	14.11.2020	
5.	Оформлення та перевірка чернетки дипломної роботи та внесення змін до неї керівником	17.11.2020	
6.	Перевірка пояснювальної записки дипломної роботи у системі «Антиплагіат»	19.11.2020	
7.	Оформлення пояснювальної записки дипломної роботи	21.11.2020	
8.	Підготовка презентації та доповіді	25.11.2020	
9.	Попередній захист роботи	25.11.2020	
10.	Рецензування роботи	26.11.2020	
11.	Допуск до захисту дипломної роботи в екзаменаційній комісії	27.11.2020	
12.	Подання дипломної роботи в екзаменаційну комісію для захисту	27.11.2020	

Студент _____ Максим НОВІКОВ
(підпис)

Керівник роботи _____ Сергій МІНУХІН
(підпис)

РЕФЕРАТ

Пояснювальна записка до магістерської дипломної роботи містить: 69 с., 35 рис., 5 табл., 63 джерела.

Метою роботи є отримання якнайбільш точного результату сегментації медичного зображення та аналіз ефективності розподіленого навчання з використанням штучної нейронної мережі U-Net.

Об'єктом дослідження є сегментація зображень з використанням нейронної мережі U-Net та розподіленого режиму TensorFlow.

Предметом дослідження є навчання мережі U-Net на ультразвукових зображеннях нервової структури шиї з використанням різних функцій оптимізації та мстодів розподілсння навчання.

Методами дослідження є статистичний аналіз точності передбачень вивчених моделей сегментації зображень та порівняння загального часу витраченого на тренування.

Отримані результати: статистика ефективності різних конфігурацій штучної нейронної мережі UNET та вплив використання розподіленого навчання у порівнянні з навчанням у локальному режимі.

Сегментація зображень - це швидко зростаюча технологія, яка сьогодні застосовується в різних аспектах науки, техніки, менеджменту, бізнесу та повсякденної діяльності, і також є ключовим напрямком досліджень. Наприклад, технологія сегментації зображень використовується в такій області як обробка медичних зображень, оскільки вона дозволяє виділяти область інтересу за допомогою напівавтоматичного або автоматичного процесу. Зі збільшенням використання комп'ютерної томографії (КТ) та магнітно-резонансної томографії (МРТ) для діагностики, планування лікування та клінічних досліджень стало майже обов'язковим використання комп'ютерів для допомоги рентгенологічним експертам у клінічній діагностиці та плануванні лікування.

МАШИННЕ НАВЧАННЯ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ, СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ, TENSORFLOW, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, UNET, РОЗПОДІЛЕНЕ НАВЧАННЯ.

ABSTRACT

Explanetory nore to the master's thesis contains: 69 p., 35 fig., 5 tables, 63 sources .

The aim of this work is to achieve the highest possible prediction accuracy in image segmentation task using artifiial neural network U-Net and to analyze how distributed training can affect the result.

The object of research is an image segmentation with use of artificial neural network U-Net and Tensorflow's distributed learning mode.

The subject of research is training of U-Net image segmentation model on ultrasound nerve image dataset using several different loss functons and training distribution.

The research methods are statistical analysis of trained image segmentation model's prediction accuracy and raw comparison of time required for model training.

The results of the research are an effectiveness statistic of severl different U-Net model configurations and an analysis of distributed training impact on the model compared to local training.

Image segmentation is a rapidly growing technology, which today can be applied in many cases science, engineering, management, business or even daily tasks and is still a case of many studies. For a example, image segmentation technology is used in medical image processing, because it allows to mark a region of interest in an image with a full or semi automatic process. With increased use of computed tomography (CT) and magnetic resonance imaging (MRI) for diagnosis, treatment planning, and clinical trials, the use of computers to assist radiologists in clinical diagnosis and treatment planning has become almost mandatory

MACHINE LEARNING, ARTIFICIAL NEURAL NETWORKS, DISTRIBUTED COMPUTATIONS, IMAGE SEGMENTATION, TENSORFLOW, CONVOLUTIONAL NEURAL NETWORK, UNET, DISTRIBUTED TRAINING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 8
ВСТУП 9
1. АНАЛІЗ МЕТОДІВ ТА МОДЕЛЕЙ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ 10
1.1 Класифікація методів сегментації зображень 11
1.2 Методи сегментації зображень 12
1.2.1 Метод порогового значення 12
1.2.2 Метод виявлення країв 13
1.2.3 Регіональний метод 13
1.2.4 Метод кластеризації 14
1.2.5 Методи, що базуються на вододілі 15
1.2.6 Метод сегментації на основі часткових диференціальних рівнянь 15
1.2.7 Метод сегментації на основі штучної нейронної мережі 16
1.3 Використання згорткових нейронних мереж для сегментації зображень 16
1.3.1 Будівельні блоки архітектури CNN 17
1.3.2 Згортковий шар 18
1.3.3 Функція нелінійної активації 19
1.3.4 Об'єднуючий шар 20
1.3.5 Повністю зв'язаний шар 21
1.3.6 Функція активації останнього шару 21
1.3.7 Функція оптимізації 22
1.3.7.1 Градієнтний спуск 22
1.3.7.2 Перехресна ентропія 23
1.3.7.3 Індекс Соренсена 23
1.3.7.4 Міра Жаккара 24
1.4 Висовки з аналізу методів та моделей сегментації зображень 25
2. АНАЛІЗ СУЧАСНИХ МОДЕЛЕЙ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ ТА ВИКОРИСТАННЯ ФРЕЙМВОРКА TENSORFLOW 26
2.1 Архітектура мережі Mask R-CNN 26
2.2 Архітектура нейронної мережі U-Net 29
2.3 Архітектура мережі DeepLab 33
2.4 TensorFlow 36
2.4.1 Модель виконання TensorFlow 37
2.4.1.1 Графи 37
2.4.1.2 Охоче виконання 37

2.4.2 Розподілене навчання TensorFlow 38
2.4.3 Розподілений TensorFlow за допомогою tf.distribute.Strategy 38
2.4.3.1 MirroredStrategy 39
2.4.3.2 TPUS стратегія 39
2.4.3.3 MultiWorkerMirroredStrategy 40
2.4.3.4 CentralStorageStrategy 40
2.4.3.5 ParameterServerStrategy 40
2.5 План проведення експерименту 41
3. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ НА ПРИКЛАДІ УЛЬТРАЗВУКОВИХ ЗОБРАЖЕНЬ ПЛЕЧОВОГО НЕРВУ 43
3.1 Конфігурація сервера 43
3.2 Навчання у локальному режимі з використанням двійкової перехресної ентропії 43
3.3 Навчання у локальному режимі з використанням індекса Соренсена 44
3.4 Навчання у локальному режимі з використанням міри Жаккара 45
3.5 Порівняльний аналіз результатів використання різних функцій оптимізації для тренуванні моделі у локальному режимі 46
3.6 Тренування у розподіленому режимі з використанням міри Жаккара 46
3.7 Аналіз результатів використання псевдо-розподіленого режиму для тренування моделі 47
ВИСНОВКИ 48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 49
ДОДАТКИ 54
ДОДАТОК А 55
ДОДАТОК Б 59
ДОДАТОК В 66

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CNN	Convolutional neural network
КТ	Компьʼютерна томографія
МРТ	Магнітно-резонансна томографія
ReLU	Rectified linear unit
FC	Fully-convoluted
FCN	Fully-convolutional network
SGD	Stochastic gradient descent

ВСТУП

Сегментація зображень - один з основних розділів комп'ютерного зору. Техніка сегментації зображень використовується для поділу картинки на значущі частини, які мають схожі риси і властивості. основна ціль сегментації - спрощення, тобто представлення зображення в осмисленому, легко згадуваному вигляді.

Сегментація зображень сьогодні застосовується в різних аспектах науки, техніки, менеджменту, бізнесу та повсякденної діяльності, і також є ключовим напрямком досліджень. Наприклад, технологія сегментації зображень використовується в такій області як обробка медичних зображень, оскільки вона дозволяє виділяти область інтересу за допомогою напівавтоматичного або автоматичного процесу.

В даний час зростає попит на медичну візуалізацію в радіотерапії, керованій зображеннями, хірургії, керованій зображеннями, інтервенційній терапії та навігації, керованій зображеннями, що сприяє дослідженням та розвитку медичної технології візуалізації та технології обробки зображень. Точність та ефективність хірургічного втручання набагато вища, ніж у традиційних хірургічних процедур, і це може зменшити ризик хірургічного втручання.

Медична сегментація зображень - ключова технологія наведення зображень. Переваги та недоліки сегментації зображень відіграють важливу роль в хірургії, керованій зображеннями. Медична сегментація зображень - це складний та важливий крок у галузі обробки та аналізу медичних зображень. Його метою є якомога точніше зобразити анатомічну структуру, що цікавить, або область певної тканини.

1. АНАЛІЗ МЕТОДІВ ТА МОДЕЛЕЙ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

Сегментація зображень - це техніка поділу або розподілу зображення на частини, які називаються сегментами. Це в основному корисно для таких програм, як стиснення зображень або розпізнавання об'єктів, оскільки для таких типів програм неефективно обробляти ціле зображення. Отже, сегментація зображення використовується для сегментації деталей із зображення для подальшої обробки. Існує кілька методів сегментації зображень, які розподіляють зображення на кілька частин на основі певних особливостей зображення, таких як значення інтенсивності пікселів, колір, текстура тощо. Усі ці методи класифікуються на основі використовуваного методу сегментації. У цьому розділі розглядаються різні методи сегментації зображень і наводиться порівняння їх переваг та недоліків.

Цифрова обробка зображень - це використання комп'ютерних алгоритмів для обробки зображень на цифрових зображеннях. Сегментація зображень є важливим та складним процесом обробки зображень. Техніка сегментації зображень використовується для розподілу зображення на значущі частини, що мають подібні риси та властивості. Головною метою сегментації є спрощення, тобто представлення зображення значущим та легко аналізованим способом. Сегментація зображень є першим кроком в аналізі зображень. Мета сегментації зображень - розділити зображення на кілька частин/сегментів, що мають подібні ознаки або атрибути. Основними програмами сегментації зображень є: отримання зображень на основі вмісту, медична візуалізація, завдання виявлення та розпізнавання об'єктів, автоматичні системи контролю дорожнього руху та відеоспостереження тощо. Сегментацію зображень можна класифікувати на два основних типи: Локальна сегментація (стосується конкретних частина або область зображення) та глобальна сегментація (стосується сегментування всього зображення, що складається з великої кількості пікселів).

Підходи до сегментації зображень можна класифікувати на два типи на основі властивостей зображення:

1. Підхід на основі виявлення розривів - це підхід, при якому зображення сегментується на регіони на основі розриву. Сегментація, заснована на виявленні країв, відноситься до цієї категорії, в якій краї, утворені внаслідок розриву інтенсивності, виявляються і пов'язуються з межами областей утворення [1].
2. Підхід на основі виявлення подібності - це підхід, при якому зображення сегментується на регіони на основі подібності. Методи, які

підпадають під цей підхід, це: техніки встановлення порогів, методи вирощування регіонів та розділення та злиття регіонів. Всі вони ділять зображення на регіони, що мають подібний набір пікселів. Методи кластеризації також використовують цю методологію. Вони поділяють зображення на набір кластерів, що мають схожі риси на основі деяких заздалегідь визначених критеріїв [1, 2]. Іншими словами, ми також можемо сказати, що до сегментації зображень можна підходити з трьох перспектив: регіональний підхід, крайовий підхід та кластеризація даних. Регіональний підхід підпадає під виявлення подібності, а виявлення країв і границь - під виявлення розривів. Методи кластеризації також перебувають під виявленням подібності

1.1 Класифікація методів сегментації зображень

Існує кілька існуючих методів, які використовуються для сегментації зображень. Усі ці методи мають своє значення. До всіх цих методів можна підходити з двох основних підходів сегментації, тобто підходів, що базуються на регіонах або країв. Кожну техніку можна застосовувати на різних зображеннях для виконання необхідної сегментації. Ці всі методи також можна класифікувати на три категорії [3, 4]:

1. Методи структурної сегментації - це ті методи сегментації зображень, які спираються на інформацію про структуру необхідної частини зображення, тобто необхідної області, яка повинна бути сегментована.
2. Стохастичні прийоми сегментації - це ті прийоми сегментації зображення, які працюють на дискретних значеннях пікселів зображення замість структурної інформації регіону.
3. Гібридні техніки - це ті методи сегментації зображень, які використовують концепції обох вищезазначених методів, тобто вони використовують дискретну піксельну та структурну інформацію разом [5].

Популярними методами, що використовуються для сегментації зображень, є: метод порогового значення, техніка, що базується на виявленні країв, техніка, що базується на областях, техніка на основі кластеризації, техніка, що базується на вододілі, техніка, що базується на диференціальному рівнянні, техніка на основі штучної нейронної мережі тощо. щодо методу, що використовується ними для сегментації.

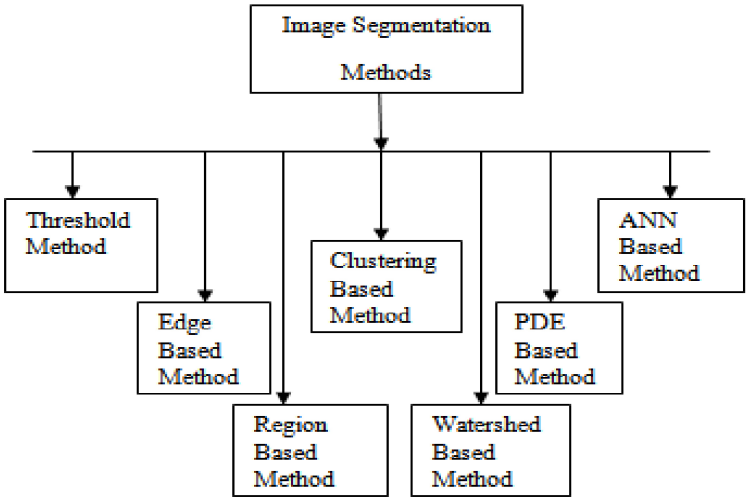

Рис. 1.1 Класифікація методів сегментації зображень

1.2 Методи сегментації зображень

1.2.1 Метод порогового значення

Порогові методи - це найпростіші методи сегментації зображень. Ці методи ділять пікселі зображення за рівнем їх інтенсивності. Ці методи використовуються для зображень, які мають світліші предмети, ніж фон. Вибір цих методів може бути ручним або автоматичним, тобто може базуватися на попередніх знаннях або інформації про особливості зображення. В основному існує три типи порогових значень [16, 20]:

1. Глобальне порогове значення. Це робиться за допомогою будь-якого відповідного порогового значення T. Це значення T буде постійним для всього зображення. На основі T вихідне зображення $q(x,y)$ можна отримати з вхідного зображення $q(x,y)$ як:

$$q(x,y) = \begin{cases} 1, if\ p(x,y) > T \\ 0, if\ p(x,y) \le T \end{cases}$$

2. Змінне порогове значення. При цьому типі порогового значення значення T може змінюватися залежно від зображення. Крім того, це може бути двох типів:

Місцевий поріг: при цьому значення T залежить від сусідства x та y.

Адаптивний поріг: Значення T є функцією x та y.

3. Кілька порогових значень. У цьому типі порогових значень існує кілька порогових значень, таких як T_0 і T_1. За допомогою цих вихідних зображень можна обчислити як

$$q(x,y) = \begin{cases} m, if\ p(x,y) > T_1 \\ n, if\ p(x,y) \le T_1 \\ o, if\ p(x,y) \le T_0 \end{cases}$$

Значення порогових значень можна обчислити за допомогою піків гістограм зображення. Для їх обчислення також можна створити прості алгоритми.

1.2.2 Метод виявлення країв

Методи виявлення країв - це добре розроблені методи обробки зображень самостійно. Методи сегментації на основі ребра засновані на швидкій зміні значення інтенсивності зображення, оскільки одне значення інтенсивності не дає хорошої інформації про ребра. Методи виявлення країв знаходять ребра, де або перша похідна інтенсивності більша, ніж певний поріг, або друга похідна має нульові перехоли. У методах сегментації, заснованих на ребрах, перш за все ребра виявляються, а потім з'єднуються між собою, утворюючи межі об'єкта для сегментації необхідних областей. Основними двома методами сегментації на основі ребер є: сірі гістограми та методи на основі градієнта. Для виявлення ребер може бути використаний один з основних методів виявлення ребер, такий як оператор собеля, оператор canny та оператор Роберта тощо. Результатом цих методів є в основному двійковий образ. Це структурні методи, засновані на виявленні розривів [11].

1.2.3 Регіональний метод

Методи сегментації на основі регіону - це методи сегментації зображення на різні регіони, що мають схожі характеристики. Існують дві основні методики, засновані на цьому методі [3, 8, 26].

1. Методи вирощування регіонів. Методи сегментації на основі вирощування регіонів - це методи сегментації зображення на різні регіони на основі вирощування насіння (початкові пікселі). Ці насіння можна відібрати вручну (на основі попередніх знань) або автоматично (на основі конкретного застосування). Тоді вирощування насіння контролюється за допомогою зв’язку між пікселями, і за допомогою попереднього знання проблеми це можна зупинити. Основними етапами алгоритму (заснованого на 8 підключеннях) для методу вирощування регіону є:

Якщо $p(x,y)$ - це оригінальне зображення, яке потрібно сегментувати, $s(x,y)$ - це двійкове зображення. Нехай T - будь-який предикат, який перевіряється для кожного (x,y) місця.

1.Перш за все, всі сполучені компоненти s розмиваються.

2.Обчисліть двійкове зображення P_T. Де $P_T(x,y) = 1$, якщо $T(x,y) = True$.

3.Обчисліть двійкове зображення q, де $q(x,y) = 1$, якщо $P_T(x,y) = 1$ та (x,y) - це 8-приєднане значення в s.

Ці сполучені компоненти в q є сегментованими областями.

2. Методи розділення та об'єднання регіонів. Методи сегментації регіонів, що розбиваються та об'єднуються, використовують дві основні техніки, тобто розділення та об'єднання для сегментації зображення на різні регіони. Розбиття означає ітеративний розподіл зображення на області, що мають подібні характеристики, та об’єднання сприяє об’єднанню сусідніх подібних областей. На наступній діаграмі показано розподіл на основі чотирикутника. Основними кроками алгоритму росту та злиття регіону є [22].

Нехай p - оригінальне зображення, а T - конкретний предикат.

1.Перш за все R_1 дорівнює p.

2.Кожна область ділиться на квадранти, для яких $T(R_i) = False$.

3.Якщо для кожної області $T(R_j) = True$, то зливається з сусідніх областей R_i та R_j, що $T(R_i \cup R_j) = True$.

4.Повторюється крок 3, доки об’єднання неможливе

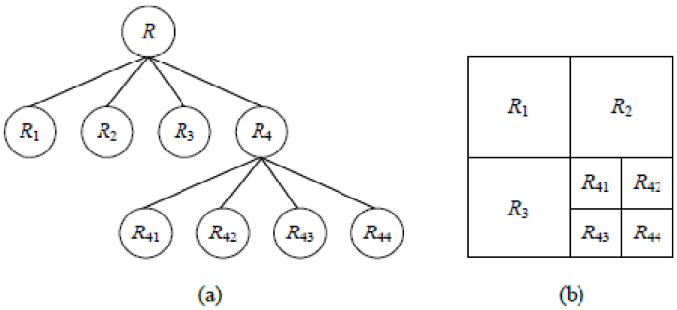

Рис. 1.2 Поділ регіонів за квадратичним деревом [27]

1.2.4 Метод кластеризації

Методи, засновані на кластеризації, - це методи, які сегментують зображення на кластери, що мають пікселі з подібними характеристиками. Кластеризація даних - це метод, який розділяє елементи даних на кластери так, що елементи в одному кластері більш схожі один на одного, ніж інші. Існує дві основні категорії методів кластеризації: ієрархічний метод та метод, заснований на розділах. Ієрархічні методи базуються на концепції дерев. При цьому корінь дерева представляє всю базу даних, а внутрішні вузли представляють кластери. З іншого боку, методи, що базуються на розділах, використовують методи оптимізації ітеративно для мінімізації цільової функції. Між цими двома методами є різні алгоритми пошуку кластерів. Існує два основних типи кластеризації [13, 24].

1. Жорстке кластеризування - це проста техніка кластеризації, яка ділить зображення на набір кластерів таким чином, що один піксель може належати лише одному кластеру. Іншими словами, можна сказати, що кожен піксель може належати рівно одному кластеру. Ці методи використовують функції належності, що мають значення 1 або 0, тобто один або певний піксель може належати певному кластеру чи ні. Прикладом техніки, що базується на жорсткій кластеризації, є одна з методів кластеризації на основі k-середніх, відома як HCM. У цій техніці спочатку обчислюються центри, потім кожен піксель призначається найближчому центру. Він наголошує на максимізації внутрішньої кластерної подібності, а також мінімізації міжкластерної рівності.

2. М'якс кластеризування є більш природним видом кластеризації, оскільки в реальному житті неможливий точний поділ через наявність шуму. Таким чином, методи м'якої кластеризації є найбільш корисними для сегментації зображень, де поділ не є суворим. Прикладом такого типу техніки є нечітка кластеризація середніх значень. У цій техніці пікселі поділяються на кластери на основі часткового членства, тобто один піксель може належати до кількох кластерів, і цей ступінь належності описується значеннями членства. Цей прийом є більш гнучким, ніж інші прийоми [13].

1.2.5 Методи, що базуються на вододілі

Методи, що базуються на вододілі, використовують концепцію топологічної інтерпретації. При цьому інтенсивність представляє басейни, що мають отвір у мінімумах, звідки вода розливається. Коли вода досягає межі басейну, сусідні басейни зливаються між собою. Для підтримання розділеності між басейнами потрібні дамби, які є межами ділянки сегментації. Ці дамби єпобудований з використанням дилатації. Методи вододілу розглядають градієнт зображення як топографічну поверхню. Пікселі, що мають більший градієнт, представлені у вигляді меж, які є безперервними [15].

1.2.6 Метод сегментації на основі часткових диференціальних рівнянь

Методи, що базуються на рівнянні з частковими похідними, є швидкими методами сегментації. Вони підходять для часових застосувань. Існують два основні методи PDE: нелінійний ізотропний дифузійний фільтр (використовується для посилення країв) та опукла неквадратична варіація відновлення (використовується для видалення шуму). Результатами методу PDE є розмиті краї та межі, які можна зміщувати за допомогою близьких операторів. Метод PDE четвертого порядку використовується для зменшення

шуму від зображення, а метод PDE другого порядку використовується для кращого виявлення країв та меж [13].

1.2.7 Метод сегментації на основі штучної нейронної мережі

Методи сегментації на основі штучної нейронної мережі імітують стратегії навчання людського мозку з метою прийняття рішень. Зараз цей метод в основному використовується для сегментації медичних зображень. Він використовується для відокремлення потрібного зображення від фону. Нейронна мережа складається з великої кількості підключених вузлів, і кожне з'єднання має певну вагу. Цей метод не залежить від PDE. У цьому проблема перетворюється на проблеми, які вирішуються за допомогою нейронної мережі. Цей метод має два основних етапи: вилучення функцій та сегментацію за допомогою нейронної мережі [8].

1.3 Використання згорткових нейронних мереж для сегментації зображень

Згорткова нейронна мережа (convolutional neural network/CNN) - це тип моделі глибокого навчання для обробки даних, що має сітчастий шаблон, такий як зображення, який натхненний організацією зорової кори кори тварин [40, 41] і призначений для автоматичного та адаптивного вивчення просторових ієрархій особливостей, від низьких до моделей високого рівня. CNN - це математична конструкція, яка, як правило, складається з трьох типів шарів (або будівельних блоків): згортки, об'єднання та повністю пов'язаних шарів. Перші два, згортання та об'єднання шарів, виконують видобуток особливостей, тоді як третій, повністю пов'язаний шар, відображає вилучені ознаки в кінцевий результат, такий як класифікація. Рівень згортки відіграє ключову роль у CNN, який складається з набору математичних операцій, таких як згортка, спеціалізований тип лінійних операцій. На цифрових зображеннях значення пікселів зберігаються у двовимірній (2D) сітці, тобто масиві чисел (Рис. 1.4), а невелика сітка параметрів, яка називається ядром, оптимізується функцією екстрактора, застосовується в кожному положенні зображення, що робить CNN високоефективними для обробки зображень, оскільки особливість може траплятися де завгодно на зображенні. Оскільки один шар подає свої вихідні дані в наступний шар, витягнуті функції можуть ієрархічно і поступово ускладнюватися. Процес оптимізації таких параметрів, як ядра, називається навчанням, яке виконується таким чином, щоб мінімізувати різницю між

виходами та основними мітками істини за допомогою алгоритму оптимізації, званого зворотним розповсюдженням та зменшенням градієнта, серед іншого.

В останніх радіомічних дослідженнях використовуються власноруч виготовлені прийоми вилучення особливостей, такі як аналіз текстур, а потім традиційні класифікатори машинного навчання, такі як випадкові ліси та опорні векторні машини [42, 43]. Існує кілька відмінностей між цими методами та CNN. По-перше, CNN не вимагає видобування функцій, виготовлених вручну. По-друге, архітектури CNN не обов'язково вимагають сегментації пухлин або органів фахівцями-людьми. По-третє, CNN набагато голодніший через свої мільйони параметрів, що піддаються вивченню, і, таким чином, є більш важким для обчислювання, що призводить до необхідності навчання з використанням графічних модулів обробки даних (GPU).

1.3.1 Будівельні блоки архітектури CNN

Архітектура CNN включає кілька будівельних блоків, такі шари згортки, шари об'єднання та повністю пов'язані шари. Типова архітектура складається з повторень накопичення декількох згорткових шарів та шару об'єднання, за яким слідують один або кілька повністю з'єднаних шарів. Етап, на якому вхідні дані перетворюються у вихідні через ці шари, називається простим поширенням (Рис. 1.3). Хоча операції згортки та об'єднання, описані в цьому розділі, стосуються 2D-CNN, подібні операції також можуть виконуватися для тривимірних (3D) -CNN.

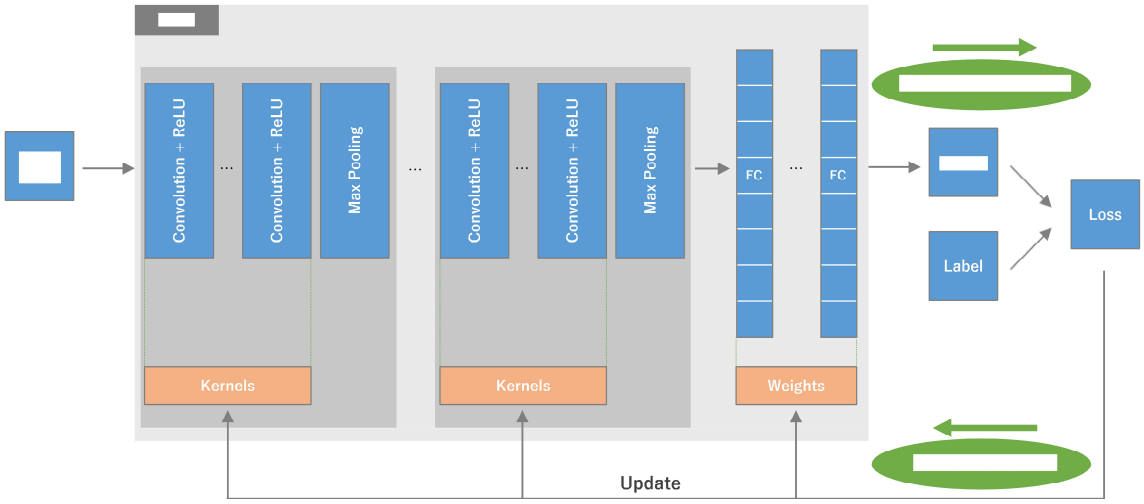

Рис. 1.3 Огляд архітектури згорткової нейронної мережі (CNN) та навчального процесу. CNN складається з шарів згортки, шарів об'єднання та повністю з'єднаних (FC) шарів

1.3.2 Згортковий шар

Шар згортки є фундаментальним компонентом архітектури CNN, який виконує вилучення ознак, який, як правило, складається з комбінації лінійних та нелінійних операцій, тобто операції згортки та функції активації.

Згортка- це спеціалізований тип лінійних операцій, що використовується для вилучення особливостей, де на вхід подається невеликий масив чисел, званий акернелом, який є масивом чисел, що називається тензором. Елементний добуток між елементом ядра та вхідним тензором обчислюється при кожному розташуванні тензора і підсумовується для отримання вихідного значення у відповідному положенні вихідного тензора, що називається карта об’єктів. Ця процедура повторюється із застосуванням декількох ядер для формування довільної кількості карт об'єктів, які представляють різні характеристики вхідних тензорів; різні ядра, таким чином, можна розглядати як різні екстрактори ознак. Два ключових гіперпараметри, що визначають операцію згортання, - це розмір і кількість ядер. Розмір зазвичай складає 3 × 3, але іноді 5 × 5 або 7 × 7. Кількість ядер є довільною і визначає глибину вихідних карт функцій.

Описана вище операція згортки не дозволяє центру кожного ядра перекривати крайній елемент вхідного тензора і зменшує висоту та ширину вихідної карти особливостей порівняно з вхідним тензором. Доповнення, як правило, нульове заповнення, - це техніка вирішення цієї проблеми, коли рядки та стовпці нулів додаються з кожного боку вхідного тензора, щоб відповісти центру ядра на самому зовнішньому елементі і зберегти той самий розмір у площині через операція згортки (рис. 1.4). Сучасні архітектури CNN зазвичай використовують нульове заповнення, щоб зберегти площинні розміри, щоб нанести більше шарів. Без нульового заповнення кожна послідовна карта функцій стане менше після операції згортки. Відстань між двома послідовними позиціями ядра називається кроком, що також визначає роботу згортки. Загальним вибором кроку є 1; однак іноді використовується вершина більша за 1 для того, щоб домогтися дискретизації карт об'єктів. Альтернативною технікою виконання демпсемплінгу є операція об’єднання, як описано нижче. Ключовою особливістю операції згортки є розподіл ваги: ядра розподіляються по всіх позиціях зображення. Спільне використання ваги створює наступні характеристики операцій згортки:

1. Дозволяючи локальним шаблонам особливостей, вилученим перекладом ядер, перекладати b, інваріантно, коли ядра пересуваються по всіх позиціях зображення і виявляти вивчені локальні шаблони.

2. Вивчення просторових ієрархій шаблонів об’єктів шляхом зменшення вибірки в поєднанні з операцією об’єднання, призводить до захоплення дедалі більшого поля зору.

3. Підвищення ефективності за рахунок зменшення кількості параметрів для вивчення порівняння з повністю підключеними нейронними мережами.

Процес навчання моделі CNN щодо рівня згортки полягає у визначенні ядер, які найкраще працюють для даного завдання на основі заданого набору даних навчання. Ядра - це єдині параметри, які автоматично засвоюються під час тренувального процесу на рівні згортки; з іншого боку, розмір ядер, кількість ядер, прокладки та крок - це гіперпараметри, які потрібно встановити перед початком навчального процесу.

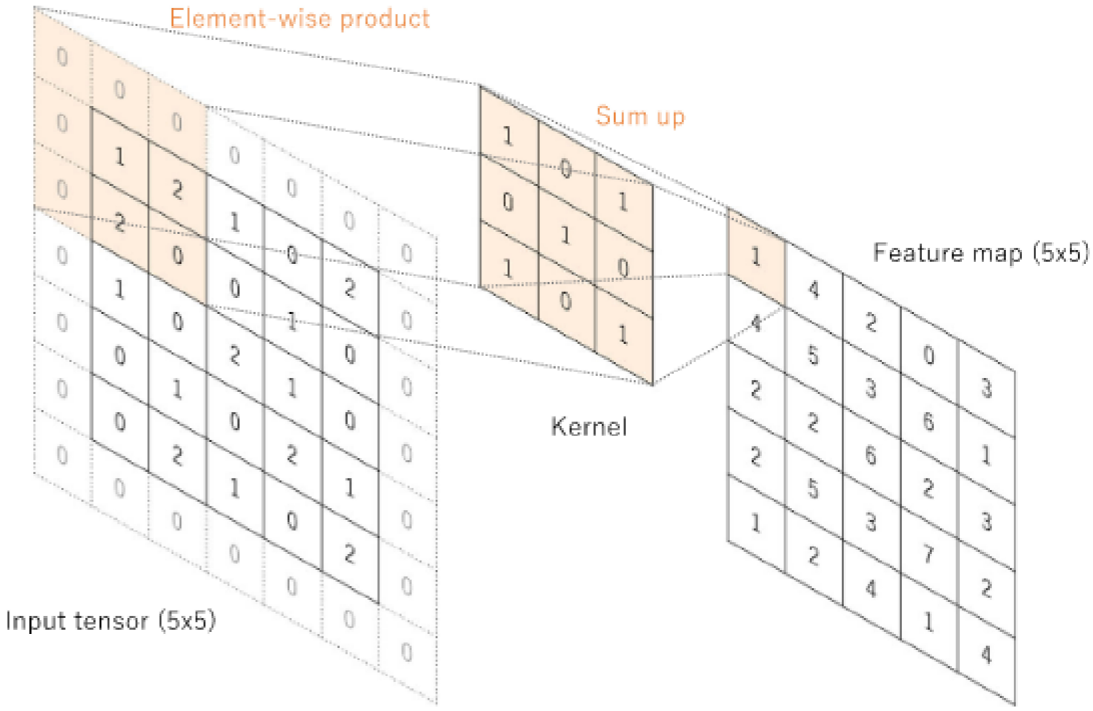

Рис. 1.4 Операція згортки з нульовим заповненням, щоб зберегти розміри в площині. Вхідний розмір 5 × 5 зберігається на карті вихідних характеристик. У цьому прикладі розмір ядра та крок встановлюються відповідно 3 × 3 та 1

1.3.3 Функція нелінійної активації

Результати лінійної операції, такі як згортка, передаються через нелінійну функцію активації. Хоча раніше використовувались плавні нелінійні функції, такі як сигмоїдна або гіперболічна дотична (tanh), оскільки вони є математичними зображеннями поведінки біологічного нейрона, найпоширеніша функція нелінійної активації, що використовується в даний час,

є випрямлячем (ReLU) (рис. 1.5.(a)), яка просто обчислює функцію: $f(x) = max(0,x)$ [28, 31, 40–42]

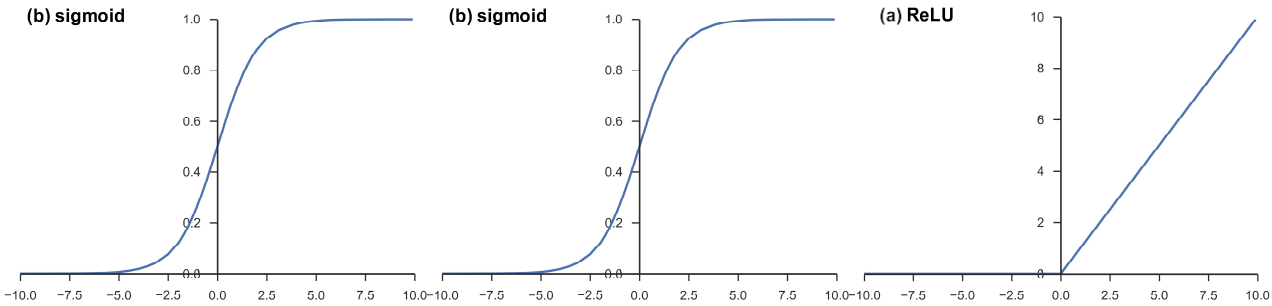

Рис. 1.5 Функції активації, які зазвичай застосовуються до нейронних мереж: (a) випрямляч (ReLU), (b) сигмоїдна та (c) гіперболічна (tanh)

1.3.4 Об’єднуючий шар

Шар об'єднання забезпечує типову операцію зменшення дискретизації, яка зменшує розмірність площини карт об'єктів для того, щоб внести незмінність трансляції до невеликих зсувів та спотворень та зменшити кількість наступних параметрів, що піддаються вивченню. Слід зазначити, що в будь-якому з шарів об’єднання немає параметрів, яких можна дізнатись, тоді як розмір фільтра, крок та відступ - це гіперпараметри в операціях об’єднання, подібно до операцій згортання.

Найпопулярнішою формою операції об'єднання є максимальне об'єднання, яке витягує патчі з вхідних карт функцій, виводить максимальне значення в кожному патчі та відкидає всі інші значення (рис.6). На практиці зазвичай використовується макс. Це зменшує вибірку в площині розмірності карт об'єктів у 2 рази. На відміну від висоти та ширини, розмір глибини карт об'єктів залишається незмінним.

Ще однією операцією об’єднання, на яку варто звернути увагу, є глобальне об’єднання середніх віків [47]. У середньому глобальному об’єднанні виконується екстремальний тип зменшення дискретизації, коли карта об’єктів із розміром висоти × ширини зменшується в масив 1 × 1, просто беручи середнє значення всіх елементів на кожній особливості карти, тоді як глибина карт об’єктів зберігається. Ця операція, як правило, застосовується лише один раз перед повністю з'єднаними шарами. Переваги застосування загального середнього пулу полягають у наступному:

1. Зменшує кількість засвоюваних параметрів
2. Дозволяє CNN приймати входи змінного розміру.

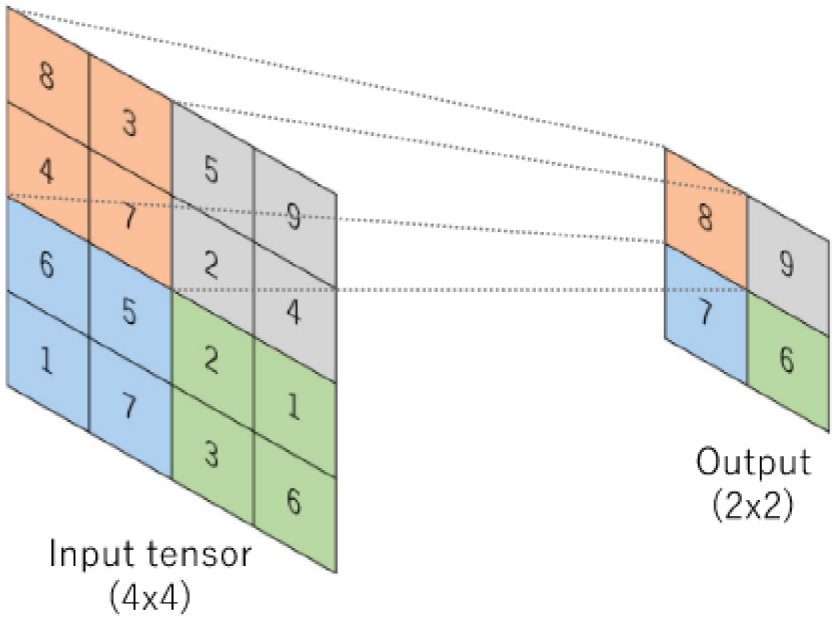

Рис. 1.6 Приклад операції максимального об'єднання з розміром фільтра 2 × 2, без заповнення та кроком 2, який витягує 2 × 2 патчі з вхідних сенсорів, виводить максимальне значення в кожному патчі та відкидає всі інші значення, в результаті чого зменшення вибірки площинної розмірності вхідного тензору в 2 рази.

1.3.5 Повністю зв’язаний шар

Карти вихідних особливостей шару остаточної згортки або об'єднання, як правило, сплющуються, тобто перетворюються в одновимірний (1D) масив чисел (або вектор) і з'єднуються з одним або декількома повністю пов'язаними шарами, також відомими як щільні шари, в яких кожен вхід приєднується до кожного виходу вагою, яку можна дізнатись. Після того, як елементи, виділені шарами згортки та зменшеними за допомогою шарів об'єднання, створюються, вони підсвічуються підмножиною повністю зв'язаних шарів з кінцевими результатами роботи мережі, такими як ймовірності для кожного класу в задачах класифікації. Кінцевий повністю підключений шар зазвичай має таку ж кількість вихідних вузлів, як і кількість класів. За кожним повністю зв’язаним шаром слідує нелінійна функція, така як ReLU, як описано вище.

1.3.6 Функція активації останнього шару

Функція активації, застосована до останнього повністю підключеного шару, зазвичай відрізняється від інших. Відповідно до кожної задачі потрібно вибрати відповідну функцію активації. Функція активації, застосована до багатокласової класифікаційної задачі, є функцією softmax, яка нормалізує

реальні значення з останнього повністю підключеного шару до ймовірностей цільового класу, де кожне значення коливається від 0 до 1, а всі значення складають до 1. Типовий вибір функції активації останнього шару для різних типів завдань узагальнений у таблиці 1.1.

Таблиця 1.1

Часто застосовувані функціїї активації останнього шару для різних завдань

Завдання	Функція активації
Бінарна класифікація	Sigmoid
Однокласна класифікація	Softmax
Багатокласова класифікація	Sigmoid
Регресія до безперервних значень	Identity

1.3.7 Функція оптимізації

Ефективність моделі під конкретними ядрами та вагами обчислюється функцією оптимізації шляхом прямого розповсюдження на навчальному наборі даних, а доступні параметри, а саме ядра та ваги, оновлюються відповідно до значення втрат за допомогою оптимізаційного алгоритму, званого зворотним розповсюдженням та градієнтом спуск, серед інших (рис. 1.3).

Функція оптимізації, яка також називається функцією втрат, вимірює сумісність між вихідними прогнозами мережі через пряме розповсюдження та заданими основними мітками істинності. Часто використована функція втрат для багатокласової класифікації - це перехресна ентропія, тоді як середня квадратична помилка зазвичай застосовується для регресії до безперервних значень. Тип функції збитків є одним із гіперпараметрів, і його потрібно визначати відповідно до заданих завдань.

1.3.7.1 Градієнтний спуск

Градієнтний спуск зазвичай використовується як алгоритм оптимізації, який ітеративно оновлює вивчувані параметри мережі, тобто ядра та ваги мережі, щоб мінімізувати втрати. швидкість збільшення, і кожен зрозумілий параметр оновлюється в негативному напрямку градієнта з довільним розміром кроку, визначеним на основі гіперпараметра, який називається швидкістю навчання. Математично градієнт є частковою похідною втрати щодо кожного вивчуваного параметра, а одне оновлення параметра формулюється наступним чином:

$$w := w - \alpha * \frac{\delta L}{\delta w}$$

де w означає кожен параметр, що піддається вивченню, α означає швидкість навчання та L - функцію втрат. Слід зазначити, що на практиці швидкість навчання є одним з найважливіших гіперпараметрів, який слід встановити перед початком навчання. На практиці з таких причин, як обмеження пам'яті, градієнти функції втрати з урахуванням параметрів обчислюються за допомогою підмножини навчального набору даних, що називається міні-пакет, і застосовуються до оновлення параметрів. Цей метод називається міні-пакетним градієнтним спуском, який також часто називають стохастичним градієнтним спуском (SGD), а розмір міні-партії також є гіперпараметром. Крім того, було запропоновано та широко використано багато вдосконалень алгоритму градієнтного спуску, таких як SGD з імпульсом, RMSprop та Адам [48–50].

1.3.7.2 Перехресна ентропія

Перехресна ентропія вимірює ефективність класифікаційної моделі, вихід якої становить значення ймовірності від 0 до 1. Втрати перехресних ентропій зростають, коли передбачувана ймовірність відхиляється від фактичної мітки. Отже, прогнозування ймовірності 0,012, коли фактична мітка спостереження дорівнює 1, буде поганим і призведе до великого значення втрат. Ідеальна модель мала б втрату 0.

У двійковій класифікації, де кількість класів M дорівнює 2, перехресну ентропію можна обчислити як:

$$CE = -\left(ylog\left(p\right) + (1-y)log(1-p)\right)$$

Якщо $M > 2$ (тобто багатокласова класифікація), ми розраховуємо окремі втрати для кожної мітки класу за спостереження та підсумовуємо результат.

$$CE = -\sum_{c=i}^{M} y_{o,c}\ log(p_{o,c})$$

де M - кількість класів, log - натуральний логарифм, y - двійковий показник (0 або 1), якщо мітка класу c є правильною класифікацією для спостереження o та p - прогнозоване спостереження за ймовірністю o класа c.

1.3.7.3 Індекс Соренсена

Інша популярна функція втрат для завдань сегментації зображень заснована на коефіцієнті Соренсена , який, по суті, є мірою перекриття двох зразків. Цей показник коливається від 0 до 1, де коефіцієнт 1 позначає ідеальне

та повне перекриття. Коефіцієнт Dice був спочатку розроблений для двійкових даних і може бути розрахований як:

$$Dice = \frac{2\,|A \cap B|}{|A| + |B|}$$

де $|A \cap B|$ представляє загальні елементи між множинами A і B, а $|A|$ представляє кількість елементів у множині A (і також для множини B).

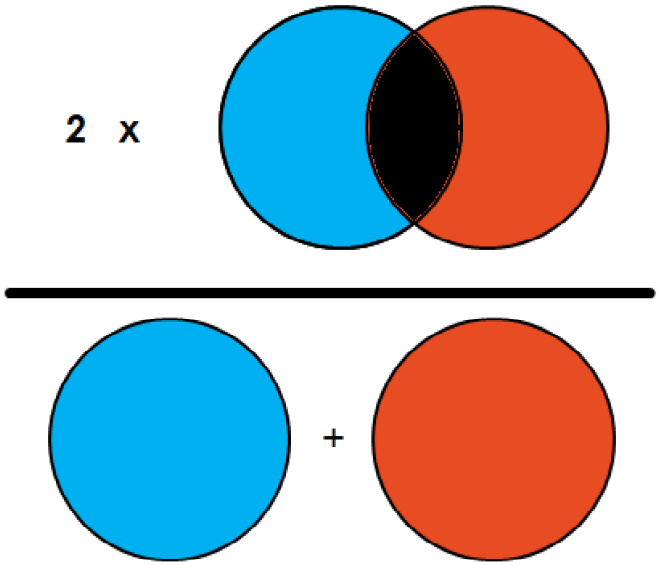

Рис. 1.7 Розрахунок коефіцієнта Соренсена

1.3.7.4 Міра Жаккара

Відомо, що перетин над об'єднанням (IoU) чи міра Жаккара є хорошою метрикою для вимірювання перекриття двох обмежувальних коробок або масок і може бути розрахована як:

$$IoU = \frac{|T \cap P|}{|T \cup P|}$$

де T означає справжнє зображення мітки, а P - спрогнозоване вихідне зображення. Якщо прогноз повністю вірний, $IoU = 1$. Чим нижчий IoU, тим гірший результат прогнозування.

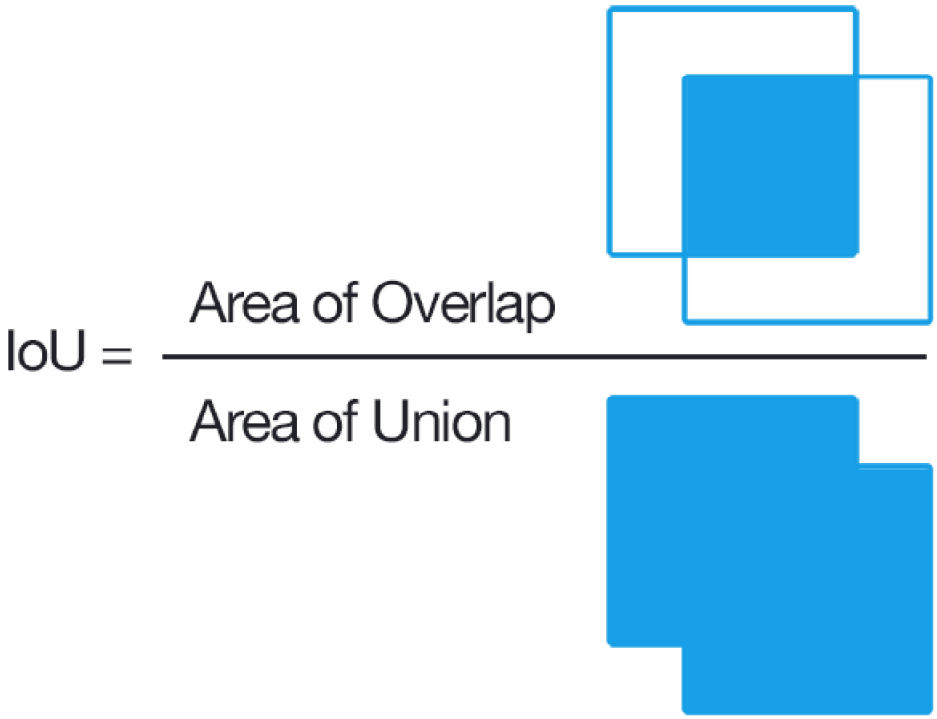

Рис. 1.8 Розрахунок міри Жаккара

1.4 Висовки з аналізу методів та моделей сегментації зображень

У цьому розділі розглянуто основні підходи до вирішення задачі сегментації зображень. Далі в експерементальному дослідженні було вирішено застосувати саме метод сегментації із використанням штучних нейронних мереж, бо на сьогоднішній день вони продовжують швидко розвиватися та можуть бути застосовані до багатьох предметних областей де використання інших методів є проблематичним [23]. У наступному розділі приведено декілька сучасних моделей сегментації зображень, що базуються на використанні згорткових шарів (с. 18) та їх модифікаціях.

2. АНАЛІЗ СУЧАСНИХ МОДЕЛЕЙ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ ТА ВИКОРИСТАННЯ ФРЕЙМВОРКА TENSORFLOW

2.1 Архітектура мережі Mask R-CNN

Регіональний підхід CNN (R-CNN) [55] до виявлення обмежуючого ящика полягає у відстеженні керованої кількості кандидатних обласних областей та оцінці згорткових мереж , незалежно від кожного регіону інтереса. R-CNN був розширений, щоб дозволити відмічати регіони інтересу на картах функцій за допомогою басейну регіонів інтересу, що призводить до швидкої швидкості та кращої точності. Faster R-CNN [56] вдосконалив цей алгоритм, використавши механізм уваги за допомогою регіональної мережі пропозицій (RPP). Швидший R-CNN є гнучким та суттєвим для багатьох подальших удосконалень і є сучасною провідною структурою.

Керуючись ефективністю R-CNN, багато підходів до сегментації екземплярів базуються на пропозиціях сегментів. Раніше методи пересортувались до сегментів знизу вгору а наступні роботи вчаться пропонувати сегментовані кандидати, які потім класифікуються Faster R-CNN. У цих методах сегментація передуєвизнанню, яке є повільним і менш точним. Так само, Dai et al. [57] запропонував комплексний багатоступеневий каскад, який передбачає пропозиції сегментів із пропозицій обмеження, з подальшою класифікацією. Натомість метод Mask R-CNN заснований на паралельному передбаченні регіонів міток та їх належності до класу, що є більш простим та гнучким. Загальна ідея полягає в тому, щоб передбачити набір позиційно чутливих вихідних каналів повністю згорнуто. Ці канали одночасно звертаються до класів об'єктів, блоків та масок, що робить систему швидкою.

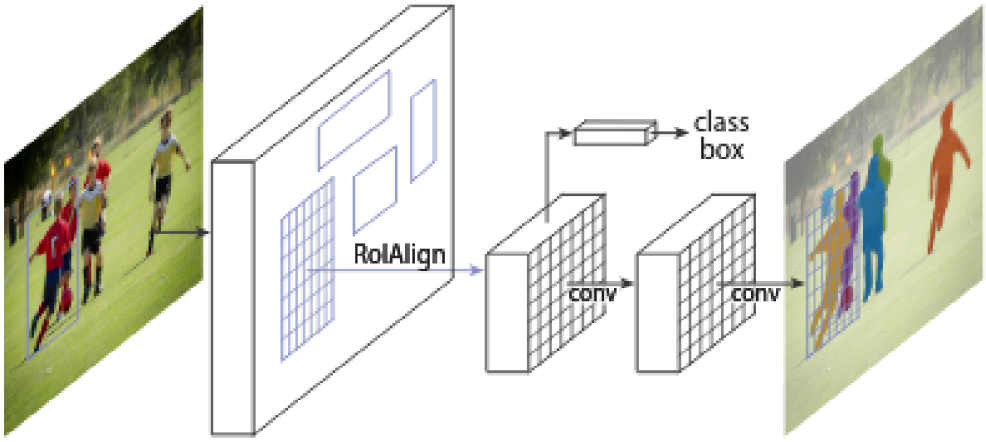

Рис. 2.1 Архітектура мережі Mask R-CNN для сешментації зображень

Маска R-CNN концептуально проста: Faster R-CNN має два виходи для кожного об'єкта-кандидата, мітку класу та зміщення ряду; до цього додається

третя гілка, яка виводить маску об’єкта. Таким чином, Mask R-CNN є природною та інтуїтивною ідеєю. Але додатковий вивід маски відрізняється від виходів класу та вікна, що вимагає вилучення набагато більш просторого макета об'єкта. Далі ми представлені ключові елементи Mask R-CNN, включаючи вирівнювання між пікселями, яке є основним відсутнім фрагментом Faster R-CNN.

Faster R-CNN складається з двох етапів. Перший етап, який називається мережею регіональних пропозицій (RPN), пропонує поля, що обмежують об'єкт-кандидат. Другий етап, який, по суті, є Faster R-CNN, витягує особливості за допомогою басейну регіона інтересів з кожного поля кандидатів та виконує класифікацію та регрессію обмежувальної коробки. Функція, яку використовують обидва етапи, може бути спільною для швидшого висновку.

Mask R-CNN приймає ту саму двоступеневу процедуру, з однаковою першою стадією (яка є RPN). На другому етапі, паралельно для прогнозування класу та зсуву, Mask R-CNN також видас двійкову маску для кожного регіону інтереса. Це на відміну від найновіших систем, де передбачення маски залежить від передбачення, слідує духу Faster R-CNN, який застосовує обмежувальну класифікацію та регресію паралельно (що, як виявилося, значно спростило багатоступеневий алгоритм оригінального R-CNN).

Формально під час навчання ми визначаємо багатозадачну втрату кожного вибіркового регіона інтересу як:

$$L = L_{cls} + L_{box} + L_{mask}$$

де L_{cls} - втрата класифікації та L_{box} - втрата обмежувального ящика ідентичні, як визначено в [58]. Гілка маски має а Km^2 - розмірний вивід для кожного регіону інтереса, який кодує K двійкових масок з роздільною здатністю $m \times m$, по одній для кожного з класів K. Для цього застосовується сигмоїд на піксель та визначається L_{mask} - середня двійкова втрата перехресної ентропії. Для регіону інтереса, асоційованого з класом маски k, L_{mask} визначається лише на k-й масці (інші виходи маски не сприяють втраті).

Таке визначення L_{mask} дозволяє мережі створювати маски для кожного класу без конкуренції між класами; ми покладаємось на спеціальну гілку класифікації, щоб передбачити мітку класу, яка використовується для вибору вихідної маски. Це відрізняється від загальноприйнятої практики при застосуванні FCN до семантичної сегментації, яка зазвичай використовує втрати на піксель softmax і multinominal перехресну ентропію. У такому випадку маски між класами змагаються; в нашому випадку, при піксельних

сигмоїдах та бінарних втратах вони цього не роблять. Експериментально показано, що ця формула є ключовою для хороших результатів сегментації примірників [59].

Маска кодує просторовий макет вхідного об'єкта. Таким чином, на відміну від міток класів або зсувів ящиків, які неминуче згортаються на короткі вихідні вектори, з'єднані між собою шарами (FC), вилучення просторової структури масок може бути вирішене природним чином за допомогою відповідності пікселів до пікселів, що забезпечується згортаннями. від кожного RoI, використовуючи FCN. Це дозволяє кожному шару в гілці маски підтримувати просторовий елемент $m \times m$, не згортаючи його у векторне представлення, в якому відсутні просторові розміри. На відміну від попередніх методів, які здійснюють пересортування до програвачів для прогнозування масок, це повністю згорткове представлення вимагає меншої кількості параметрів і є більш точним. Щоб достовірно зберегти явну просторову кореспонденцію на пікселі був розроблений наступний RoIAlign шар, який відіграє ключову роль у прогнозуванні маски.

RoIPool - це стандартна операція для вилучення невеликої карти функцій (наприклад, 7 × 7) з кожного регіону інтереса. Спочатку RoIPool квантує регіони з плаваючим числом RoI до дискретної деталізації карти об'єктів, цей квантований RoI потім поділяється на просторові групи , які самі квантуються, і, нарешті, значення характеристик, охоплені кожною групою, агрегуються (як правило, шляхом максимального об'єднання). Квантування виконується, наприклад, на безперервній координатіхпри обчисленні $[\frac{x}{16}]$, де 16 - крок карти об'єктів, а квантування проводиться при поділі на групи (наприклад, 7 × 7). Ці квантування вносять розбіжності між RoI та вилученими ознаками. Хоча це може не вплинути на класифікацію, яка є надійною для невеликих груп, вона викликає тривожний негативний вплив на прогнозування піксельних масок.

Для вирішення цієї проблеми був запропонован RoIAlign шар, який переміщує жорстке квантування RoIPool, правильно вирівнюючи витягнуті функції з вхідними даними. Запропонована зміна проста: ми уникаємо будь-якого квантування меж або груп RoI. Використовується білінійна інтерполяцію для обчислення точних значень вхідних характеристик у чотирьох регулярно відібраних місцях у кожній RoI групі та агрегування результату (використовуючи максимальне або середнє значення, див. Рис. 2.2). Було визначено, що результати не є чутливими до точних місць відбору проб або скільки точок відбираються, якщо не проводиться квантування.

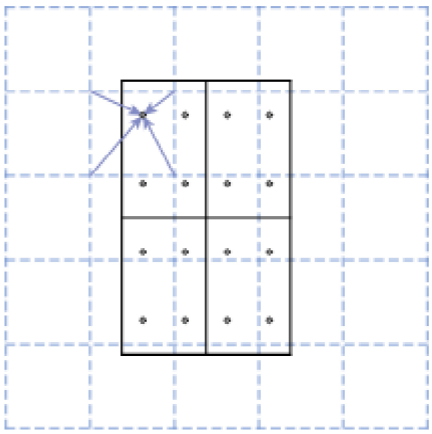

Рис. 2.2 RoiAlign операція. Пунктирна сітка повторює карту об’єктів, суцільні лінії RoI (із 2 × 2 бункерами у цьому прикладі) та крапки 4 точки відбору проб у кожній групі. RoIAlign обчислює значення кожної точки вибірки шляхом білінійної інтерполяції із сусідніх точок сітки на карті об’єктів. Жодне квантування не виконується за будь-якими координатами, задіяними в RoI, його групі або точках відбору проб.

2.2 Архітектура нейронної мережі U-Net

Нейронна мережа U-Net була розроблена для сегментації біомедичних зображень у відділенні Computer Science Фрайбургського університету. U-Net є підвидом згорткових нейронних мереж, відомим як повнозгорткові мережі [51]. Цей вид згорткових мереж відрізняється від традиційних тим, що просторова інформація (тобто розташування кожного фрагменту згортки) враховується на кожному шарі завдяки відсутності повноз’єднаних шарів у мережі. Це також дозволяє мережам даного типу оперувати зображеннями будь-якого розміру, що є неможливим з традиційними мережами оскільки розмір вхідного зображення визначається кількістю вхідних нейронів у першому повноз’єднаному шарі.

Архітектура мережі показана на рис. 2.3. Вона складається зі зменшувального шляху (ліва частина) та збільшувального шляху (права частина). Зменшувальний шлях має собою типову згорткову мережу та складається з повторного виконання двох 3×3 згорток, за кожною з яких слідує шар ReLU та 2×2 агрегування з функцією максимуму та кроком 2 для зменшення розміру зображення. На кожному кроці зменшення відбувається подвоєння кількості каналів ознак. Кожний крок збільшувального шляху складається зі збільшення карти ознак, за якою слідує 2×2 згортка, що ділить навпіл кількість каналів ознак, конкатенація з обрізаною картою ознак з відповідного кроку зменшувального шляху, та двома 3×3 згортками з шарами

ReLU. Обрізка карти ознак необхідна через те, що кожна згортка призводить до втрати кутових пікселів. На останньому кроці 1×1 згортка перетворює 64-елементний вектор ознак у бажану кількість класів. Всього мережа має 23 згорткових шара.

Для того, щоб дозволити безшовне покриття вихідної карти сегментації, потрібно вибрати такий розмір вхідного сегмента, щоб усі 2×2 максимум-пулінг операції були виконані над шаром з парним x- та y-розміром.

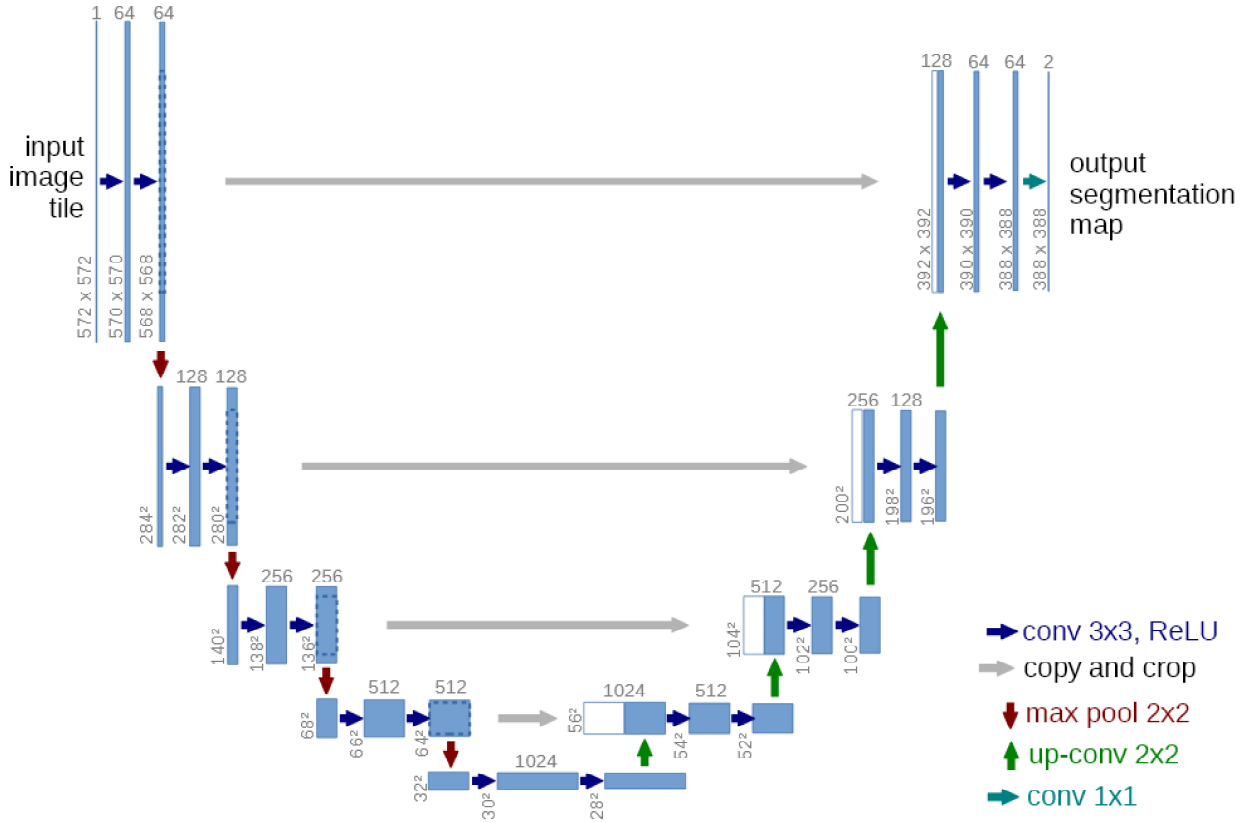

Рис. 2.3 Архітектура мережі U-Net. Числа над прямокутниками позначають кількість каналів ознак, числа у нижньому лівому куті позначають розмір зображення

Однією з важливих особливостей архітектури U-Net є те, що збільшувальний шлях має велику кількість каналів ознак, що дозволяє передавати контекстну інформацію до шарів більшого розміру зображення. Таким чином, збільшувальний шлях є більш-менш симетричним до зменшувального шляху, через що схема архітектури має U-подібну форму. Як було відмічено раніше, мережа не має повноз'єднаних шарів та використовує тільки вагому частину кожної згортки, тобто карта сегментації містить тільки пікселі, для яких присутній повний контекст у вхідному зображенні. Даний підхід дозволяє виконувати безшовну сегментацію довільно великих зображень

за допомогою стратегії перекриття-покриття (рис. 2.4.). Для передбачення пікселів у прикордонному регіоні зображення, відсутній контекст екстраполюється за допомогою дзеркальної трансформації вхідного зображення. Ця стратегія покриття необхідна для використання мережі з великими зображеннями, бо в іншому випадку розмір буде обмежений доступним об'ємом пам'яті.

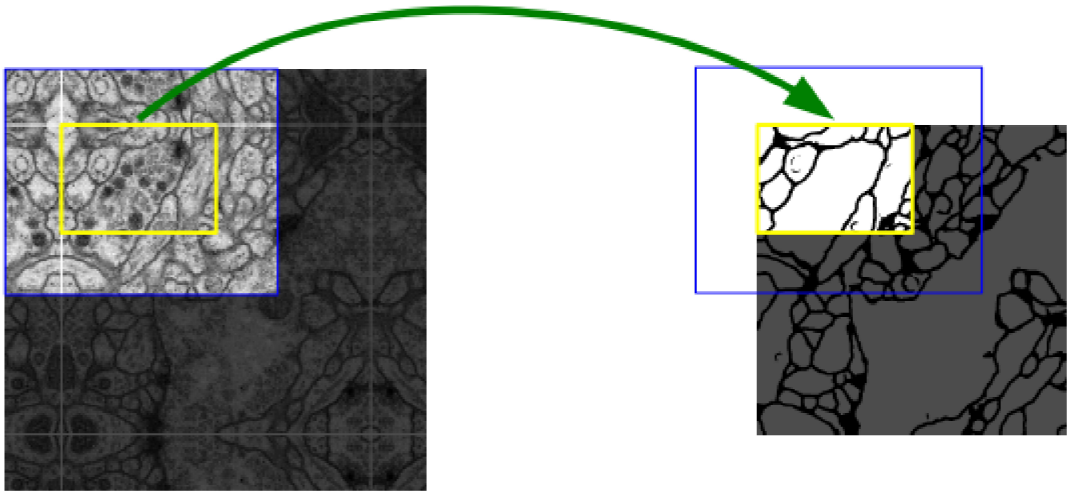

Рис. 2.4 Стратегія перекриття-покриття для безшовної сегментації довільно великих зображень. Передбачення сегментації у жовтій області потребує вхідні дані у синій області. Відсутні дані екстраполюються дзеркальним відображенням

Так як часто даних для навчання замало, було запропоновано використовувати надмірне збільшення об'єму даних за допомогою еластичної деформації доступних зображень. Це дозволяє мережі навчитися ігнорувати такий тип деформації без зовнішнього втручання. Це особливо важливо у таких областях, як сегментація біомедичних зображень, де такий тип деформації є найбільш поширеним та може бути ефективно змодельованим. Значення збільшення об'єму даних для навчання інваріантності показано у роботі [52] щодо навчання без учитля.

Вхідні зображення та відповідні карти сегментації використовуються для навчання мережі за допомогою реалізації метода стохастичного градієнтного спуску через використання згорток без доповнення, вихідне зображення має менший розмір, ніж вхідне на константанту величину. Щоб мінімізувати надмірне використання пам'яті було вирішено віддати пріоритет більшому розміру сегментів покриття та зменшити розмір пакету до одного зображення. Також використовується висока інерція (0.99) оптимізатора, що забезпечує

використання великої кількості попередньо оброблених зображень для обчислення зміни до поточного кроку оптимізації.

Функція енергії обчислюється по-піксельним виконанням операції soft-max над кінцевою картою ознак, об'єднаною з функцією втрат на основі перехресної ентропії. Функція softmax задана наступним чином:

$$p_k(x) = \frac{\exp\left(a_k(x)\right)}{\sum_{k'=1}^{K} \exp\left(a_{k'}(x)\right)},$$

де $a_k(x)$ означає активацію у каналі ознак k у позиції пікселя $x \in \Omega$, де $\Omega \subset \boldsymbol{Z}^2$, K – кількість класів, та $p_k(x)$ – наближене значення функції максимуму. Тобто, $p_k(x) \approx 1$ для k з максимальною активацією $a_k(x)$, та $p_k(x) \approx 0$ для усіх інших k. Перехресна ентропія обчислюється для кожної позиції наступним чином:

$$E = \sum_{x \in \Omega} w(x) \log\left(\mathrm{p}_{\ell(x)}(x)\right),$$

де $\ell : \Omega \rightarrow \{1, ..., K\}$ – істинний клас кожного пікселя, та $w : \Omega \rightarrow \boldsymbol{R}$ – карта вагових коефіцієнтів, що дозволяє надати деяким пікселям більшу важливість при навчанні.

Карта вагових коефіцієнтів зазвичай обчислюється зарані для кожного набору навчальних даних, наприклад для компенсації різних частот пікселів деяких класів, або для того, щоб мережа навчалася створювати розділювальні межі між незалежними об'єктами, що торкаються один одного.

Для обчислювання карти вагових коефіцієнтів, що враховують обидва вище приведених приклади можна використовувати наступну формулу:

$$w(x) = w_c(x) + w_0 \cdot \exp\left(-\frac{\left(\mathrm{d}_1(x) + d_2(x)\right)^2}{2\sigma^2}\right),$$

де $w_c : \Omega \rightarrow \boldsymbol{R}$ – карта коефіцієнтів для балансування частот класів, $d_1 : \Omega \rightarrow \boldsymbol{R}$ – відстань до межі найближчого об'єкта, $d_2 : \Omega \rightarrow \boldsymbol{R}$ – відстань до межі другого ближчого об'єкта, а w_0 та σ – константи, що підбираються експериментальним шляхом для кожного набору даних.

У мерсжах глибинного навчання з багатьма згортковими шарами та різними шляхами даних через мережу дуже важливою є ініціалізація вагових коефіцієнтів, бо в іншому випадку одні частини мережі можуть давати надмірні активації, а інші частини – не мати жодного ефекту. В ідеальному випадку початкові ваги мають бути такими, що кожна карта ознак має приблизно одиничну дисперсію. Для мережі з архітектурою що розглядається (згортки та ReLU шари, що чергуються), це може бути досягнене шляхом отримання початкових коефіцієнтів з розподілу Гауса зі стандартним відхиленням $\sqrt{2/N}$,

де N позначає кількість вхідних вузлів одного нейрона [11]. Наприклад, для 3×3 згортки та 64 каналів ознак у попередньому шарі, $N = 9 \cdot 64 = 576$.

2.3 Архітектура мережі DeepLab

Показано, що використання глибоких згорткових нейронних мереж (Deep Convolutional Neural Networks чи DCNN) для семантичної сегментації або інших завдань з щільним прогнозуванням просто та успішно вирішується шляхом розгортання DCNN повністю згорнуто [61], [62]. Однак багаторазове поєднання об'єднання максимумів та кроків на послідовних шарах цих мереж значно зменшує просторову роздільну здатність отриманих карт об'єктів, як правило, у 32 рази по кожному напрямку в останніх DCNN. Частковим засобом є використання "деконволюційних" шарів, як у [14], що, проте, вимагає додаткової пам'яті та часу.

Натомість було запропановано використання «atrous» згортки, спочатку розробленої для ефективного обчислення недециміруваного вейвлет-перетворення в схемі «алгоритму Троуса» [63] і раніше використовуваної в контексті DCNN. Цей алгоритм дозволяє нам обчислювати відповіді будь-якого шару з будь-яким бажаним дозволом. Його можна застосовувати як тільки мережа пройшла навчання, але також може бути легко інтегрована з навчанням.

Розглядаючи спочатку одновимірні сигнали, вихідний сигнал $y[i]$ «Atrous» згортки 1-D вхідного сигналу x[i] з фільтром $w[k]$ довжини K визначається як:

$$y[\mathrm{i}] = \sum_{\mathrm{k}=1}^{\mathrm{K}} x[i + r \cdot \mathrm{k}] w[k]$$

Параметр r відповідає кроку, за допомогою якого ми відбираємо вхідний сигнал. Стандартна згортка є особливим випадком для кроку $r = 1$. (див. рис. 2.5.)

Роботу алгоритму в 2-D можна проілюструвати на прикладі рис. 2.5: Отримавши зображення, ми припускаємо, що спочатку ми виконуємо операцію зменшення дискретизації, яка зменшує роздільну здатність у 2 рази, а потім виконуємо згортку з ядром - тут вертикальна Похідна Гауса. Якщо хтось імплантує результуючу карту об’єктів у вихідні координати зображення, ми розуміємо, що отримали відповіді лише на 1/4 позицій зображення. Натомість ми можемо обчислити відповіді в усіх положеннях зображення, якщо згорнути зображення з повною роздільною здатністю за допомогою фільтра «з дірками», в якому ми відбираємо вихідний фільтр у 2 рази та вводимо нульові значення

між значеннями фільтра. Хоча ефективний розмір фільтра збільшується, нам потрібно брати до уваги лише ненульові значення фільтра, отже, і кількість параметрів фільтра, і кількість операцій на позицію залишаються постійними. Схема результатів дозволяє нам легко і чітко контролювати просторову роздільну здатність реакцій нейронної мережі.

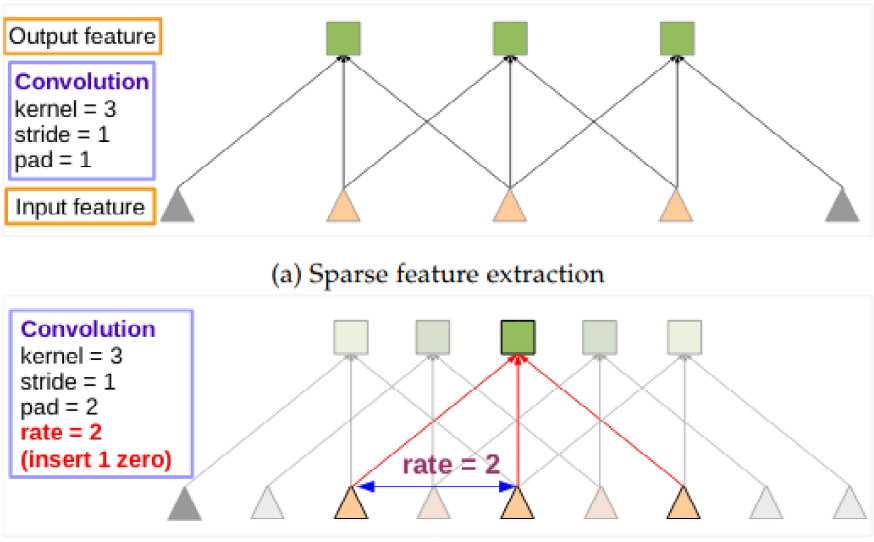

Рис. 2.5 Ілюстрація «Atrous» згортки в 1-D. (a) Виділення рідкісних особливостей зі стандартною звивиною на карті вхідних характеристик з низькою роздільною здатністю. (b) Вилучення щільних ознак із звивистою згорткою з показником кроку = 2, застосованим на карті вхідних ознак з високою роздільною здатністю.

У контексті DCNN можна використати «Atrous» згортку в ланцюзі шарів, що ефективно дозволяє нам обчислити кінцеві відповіді мережі DCNN із довільно високою роздільною здатністю. Наприклад, щоб подвоїти просторову щільність обчислюваних характеристик відповідей у мережах VGG-16 або ResNet-101, ми знаходимо останній пул об'єднання або згортковий шар, який зменшує роздільну здатність ('pool5' або 'conv51' відповідно), встановіть його крок на 1 до уникайте децимації сигналу та замінюйте всі наступні згорткові шари наїдливими згортковими шарами, що мають рейтинг = 2. Просунення цього підходу по всій мережі може дозволити обчислювати відповіді на функції з вихідною роздільною здатністю зображення, але це в кінцевому підсумку занадто дорого. Натомість було застосували гібридний підхід, який вражає хорошим компромісом між ефективністю та точністю, використовуючи «Atrous» згортку, щоб збільшити в 4 рази щільність обчислюваних карт об’єктів, а потім швидку дволінійну інтерполяцію додатковим коефіцієнтом 8

для відновлення карт об’єктів на вихідному зображенні дозвіл. Білінійна інтерполяція є достатньою в цьому налаштуванні, оскільки карти оцінок класів (що відповідають log-ймовірностям) є досить плавними. На відміну від деконволюційного підходу, прийнятого в [62], запропонований підхід перетворює мережі класифікації зображень на щільні екстрактори об'єктів без необхідності вивчати будь-які додаткові параметри, що призводить до швидшого навчання на практиці за допомогою програми DNNN.

Згортання «Atrous» також дозволяє збільшувати поля видимості фільтрів на будь-якому рівні DCNN. Сучасні DCNN зазвичай використовують просторово невеликі згорткові ядра (як правило, 3×3), щоб зберегти як обчислення, так і кількість параметрів, що містяться. Звітна зв’язка з показником введенняr – 1 нулів між послідовними значеннями фільтра, ефективно збільшуючи розмір ядра з $k \times k$ на $k_e = k + (k - 1)(r - 1)$ без збільшення кількості параметрів або обсягу обчислень. Таким чином, він пропонує ефективний механізм контролю поля зору та знаходить найкращий компроміс між точною локалізацією (мале поле зору) та асиміляцією контексту (велике поле зору).

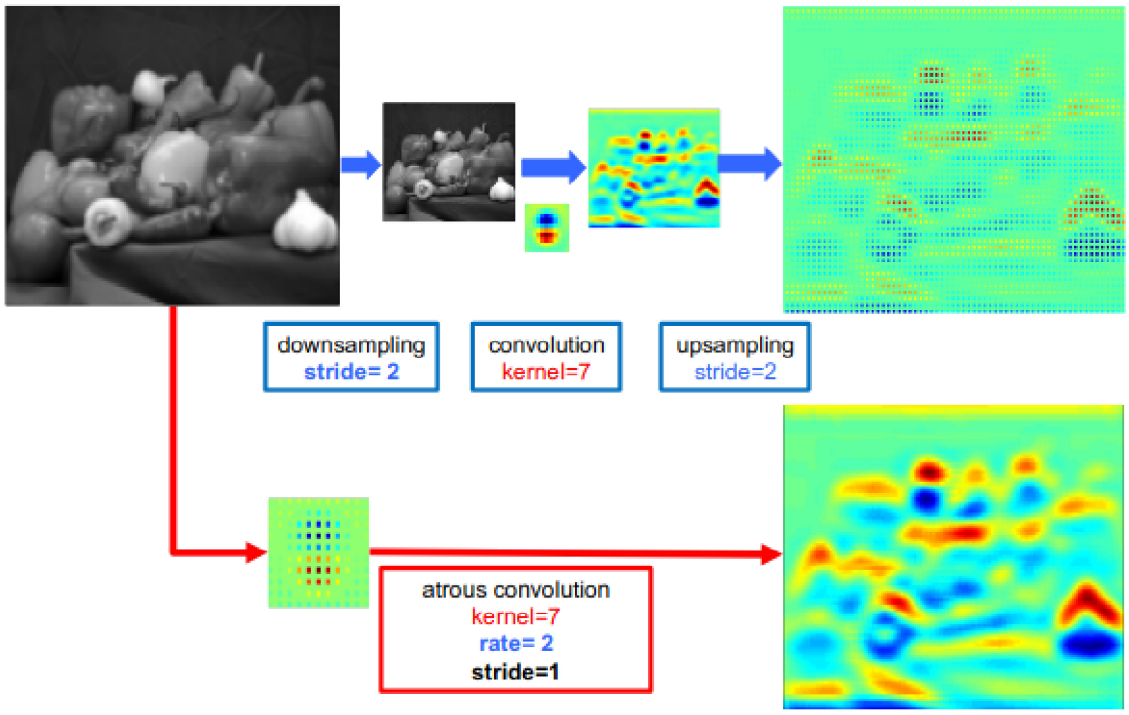

Рис. 2.6 Ілюстрація «Atrous» згортки в 2-D. Верхній рядок: розрідженс вилучення об’єкта зі стандартною згорткою на карті об’єктів вводу з низькою роздільною здатністю. Нижній рядок: Щільна екстракція особливостей з

атрусною звивиною з оцінкою = 2, застосована на карті вхідних характеристик високої роздільної здатності.

2.4 TensorFlow

TensorFlow - це бібліотека програмного забезпечення з відкритим кодом для чисельних обчислень за допомогою графіків потоків даних. Спочатку вона була розроблена командою Google Brain в рамках дослідницької організації Google Machine Machine Intelligence для машинного навчання та дослідження глибоких нейронних мереж, але система є достатньо загальною, щоб бути застосовною і в багатьох інших сферах. У лютому 2017 року він досяг версії 1.0 і продовжує стрімко розвиватися, наразі налічуючи 21 000+ комітів, багато з яких внесли внески. Ця стаття представляє TensorFlow, його спільноту з відкритим кодом та екосистему, та висвітлює деякі цікаві моделі TensorFlow з відкритим кодом.

TensorFlow є крос-платформним. Він працює майже на всьому: графічні процесори та центральні процесори, включаючи мобільні та вбудовані платформи, і навіть блоки обробки тензорів (TPU), які є спеціалізованим обладнанням для обчислення тензорної математики.

Розподілений механізм виконання TensorFlow абстрагує від багатьох підтримуваних пристроїв та забезпечує ядро високої продуктивності, реалізоване на C ++ для платформи TensorFlow.

Зверху цього сидять інтерфейси Python та C ++ (їх буде більше). API Layers забезпечує простіший інтерфейс для часто використовуваних шарів у моделях глибокого навчання. Крім того, сидять API вищого рівня, включаючи Keras (детальніше на сайті Keras.io) та Estimator API, що полегшує навчання та оцінку розподілених моделей.

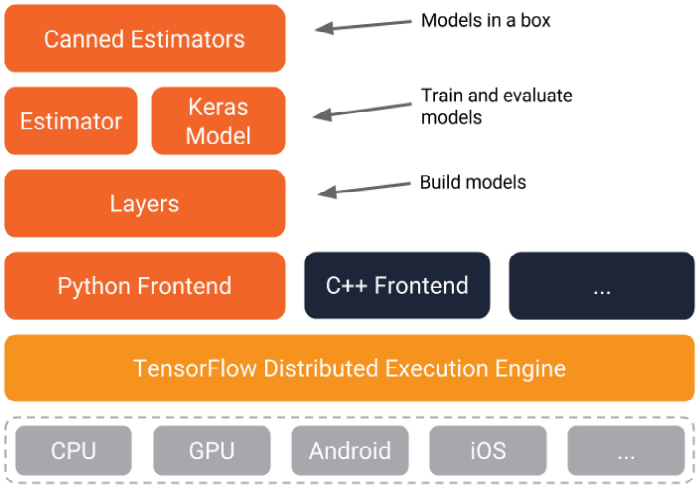

Рис. 2.7 Складові частини Tensorflow

2.4.1 Модель виконання TensorFlow

2.4.1.1 Графи

Машинне навчання може швидко ускладнитися, а моделі глибокого навчання можуть стати великими. Для багатьох моделей графів потрібно розподілене навчання, щоб мати можливість ітерації протягом розумного періоду часу. І, як правило, ви хочете, щоб розроблені вами моделі розгорталися на декількох платформах.

У поточній версії TensorFlow ви пишете код для побудови графіку обчислень, а потім виконуєте його. Граф - це структура даних, яка повністю описує обчислення, які ви хочете виконати. Це має масу переваг:

1. Він портативний, оскільки граф може бути негайно виконаний або збережений для подальшого використання, і він може працювати на декількох платформах: центральних процесорах, графічних процесорах, TPU, мобільних, вбудованих. Крім того, його можна розгорнути у виробництві, не залежачи від будь-якого коду, який побудував граф, а лише часу виконання, необхідного для його виконання.
2. Це трансформується та оптимізується, оскільки граф може бути трансформований, щоб отримати більш оптимальну версію для даної платформи. Крім того, можна виконати оптимізацію пам’яті або обчислень та зробити компроміси між ними. Це корисно, наприклад, для підтримки швидшого мобільного висновку після тренувань на більших машинах.
3. Підтримка розподіленого виконання

Високорівневі API TensorFlow у поєднанні з графами обчислень забезпечують багате та гнучке середовище розробки та потужні виробничі можливості в одних і тих же рамках.

2.4.1.2 Охоче виконання

Охоче виконання TensorFlow - це імперативне середовище програмування, яке оцінює операції негайно, без побудови графів: операції повертають конкретні значення, замість того, щоб створювати обчислювальний граф для подальшого запуску. Це полегшує початок роботи з моделями TensorFlow та налагодження, а також зменшує обсяг шаблонного коду.

Охоче виконання - це гнучка платформа машинного навчання для досліджень та експериментів, що забезпечує:

1. Інтуїтивно зрозумілий інтерфейс - дозволяє структурувати свій код природним чином і використовуйте структури даних Python. Зручний для швидкого переглядання невеликих моделей з невеликими обсягами даних.

2. Простіше налагодження - можливі безпосередні визови операцій для перевірки запущених моделей та тестування змін. Використовуйте стандартні засоби налагодження Python для негайного повідомлення про помилки.

3. Природний потік управління - використовуйте керуючий потік Python замість потоку керування графіком, спрощуючи специфікацію динамічних моделей.

2.4.2 Розподілене навчання TensorFlow

Для запуску великих моделей глибокого навчання або великої кількості експериментів потрібно буде розподілити їх між кількома процессорами, графічними процесорами або машинами. TensorFlow може допомогти розподілити тренування, розділивши моделі на багато пристроїв і проводячи тренування паралельно на пристроях.

Є два основні методи реалізації, які можна використовувати для розповсюдження навчання:

1. Модельний паралелізм - модель сегментована на різні частини, щоб ви могли запускати її паралельно. Ви можете запускати кожну деталь на одних і тих самих даних у різних вузлах. Цей підхід може зменшити потребу у спілкуванні між працівниками, оскільки працівникам потрібно лише синхронізувати загальні параметри.Паралельність моделі також добре працює для графічних процесорів на одному сервері, який використовує високошвидкісну шину, і з більшими моделями, оскільки апаратні обмеження на вузол не є обмеженням.

2. Паралельність даних - у цьому режимі дані навчання поділяються на кілька підмножин. Запускається кожна підмножину на одній і тій же реплікованій моделі в іншому вузлі. Параметри моделі повинні бути синхронізовані в кінці пакетного обчислення, щоб переконатися, що вони навчають послідовну модель, оскільки помилки передбачення обчислюються незалежно на кожному комп'ютері. Отже, кожен пристрій повинен надсилати повідомлення про всі зміни на всі моделі всіх інших пристроїв.

2.4.3 Розподілений TensorFlow за допомогою tf.distribute.Strategy

API DistributionStrategy - це простий спосіб розподілити навчальні навантаження на декількох машинах. API DistributionStrategy призначений для надання користувачам доступу до існуючих моделей та коду. Користувачам

потрібно лише мінімально змінити моделі та код, щоб забезпечити розподілене навчання.

Основні переваги використання tf.distribute.Strategy:

1. Просте переклчення між синхронним та асинхронним навчанням. Це два поширені способи розподілу навчання з паралелізмом даних. В процесі синхронного навчання всі працівники синхронно тренуються над різними зрізами вхідних даних та агрегують градієнти на кожному кроці. Під час асинхронного навчання всі працівники самостійно навчаються над вхідними даними та асинхронно оновлюють змінні.

2. Незалежність від апаратної платформи. Коли є потреба масштабувати своє навчання на декількох графічних процессорах на одній машині, або кількох машинах у мережі (з 0 або більше графічними процесорами кожен), або на хмарних TPU.

2.4.3.1 MirroredStrategy

Підтримує синхронне розподілене навчання на декількох графічних процесорах на одній машині. Він створює одну копію на пристрій графічного процесора. Кожна змінна в моделі відображається у всіх репліках. Разом ці змінні утворюють єдину концептуальну змінну, яка називається MirroredVariable. Ці змінні синхронізуються між собою, застосовуючи однакові оновлення.Для передачі змінних оновлень на всі пристрої використовуються ефективні алгоритми "all-reduce". Повністю зменшити тензори агрегатів на всіх пристроях, додаючи їх, і робить доступними на кожному пристрої. Це злитий алгоритм, який є дуже ефективним і може значно зменшити накладні витрати на синхронізацію.

Доступно багато алгоритмів і реалізацій, що дозволяють зменшити кількість, залежно від типу зв'язку між пристроями. За замовчуванням він використовує NVIDIA NCCL як всебічну реалізацію. Ви можете вибрати з кількох інших варіантів або написати свій власний.

2.4.3.2 TPUS стратегія

Дозволяє запускати тренінг TensorFlow на блоках обробки тензорів (TPU). TPU - це спеціалізовані ASIC від Google, призначені для різкого прискорення навантажень на машинне навчання. Вони доступні на Google Colab, TensorFlow Research Cloud та Cloud TPU. Що стосується розподіленої навчальної архітектури, TPUStrategy - це та сама MirroredStrategy - вона реалізує синхронне розподілене навчання.

ТПУ забезпечують власну реалізацію ефективних операцій "все скорочення" та інших колективних операцій на декількох ядрах ТПУ, які використовуються в TPUStrategy.

2.4.3.3 MultiWorkerMirroredStrategy

Дуже схожий на MirroredStrategy. Він реалізує синхронне розподілене навчання між кількома працівниками, кожен з яких може мати кілька графічних процесорів. Подібно MirroredStrategy, він створює копії всіх змінних у моделі на кожному пристрої для всіх працівників. Він використовує CollectiveOps як багатосторонній метод комунікації, який використовується для синхронізації змінних. Колективна операція - це одна операція на графу TensorFlow, яка може автоматично вибирати алгоритм з усім зменшенням у середовищі виконання TensorFlow відповідно до обладнання, топології мережі та розмірів тензора. Він також реалізує додаткові оптимізації продуктивності. Наприклад, він включає статичну оптимізацію, яка перетворює багаторазові всі скорочення на малих тензорах у меншу кількість усіх скорочень на більших тензорах.

На даний момент MultiWorkerMirroredStrategy дозволяє вибирати між двома різними реалізаціями колективних операцій. CollectiveCommunication.RING реалізує кільцеві колективи, використовуючи gRPC як рівень зв'язку. CollectiveCommunication.NCCL використовує NCCLI NCCL для реалізації колективів. CollectiveCommunication.AUTO призначає вибір часу виконання. Найкращий вибір колективної реалізації залежить від кількості та виду графічних процесорів та мережевого взаємозв’язку в кластері.

2.4.3.4 CentralStorageStrategy

CentralStorageStrategy також виконує синхронне навчання. Змінні не віддзеркалюються, натомість вони розміщуються на центральному процесорі, а операції копіюються на всіх локальних графічних процесорах. Якщо є лише один графічний процесор, усі змінні та операції будуть розміщені на цьому графічному процесорі.

2.4.3.5 ParameterServerStrategy

У TF1 ParameterServerStrategy доступний лише за допомогою оцінювача через символ tf.compat.v1.distribute.experimental.ParameterServerStrategy, і використання в TF2 розробляється для більш пізнього випуску. Для цієї стратегії деякі машини позначаються як робочі, а деякі як сервери параметрів.

Кожна змінна моделі розміщується на одному сервері параметрів. Обчислення повторюються у всіх робочих машинах кластера.

2.5 План проведення експерименту

Для експерименту було обрано датасет Kaggle Ultrasound Nerve Segmentation [53]. Вибір цього набору даних обґрунтований актуальністю сегментації медичних зображень, а мережа U-Net була обрана через те, що вона початково була розроблена для сегментації біомедичних зображень. Використаний набір даних представляє собою ультразвукові знімки плечового нервового сплетіння людини та попередньо створені бінарні маски, що виділяють групу нервів на знімку (рис. 2.8.).

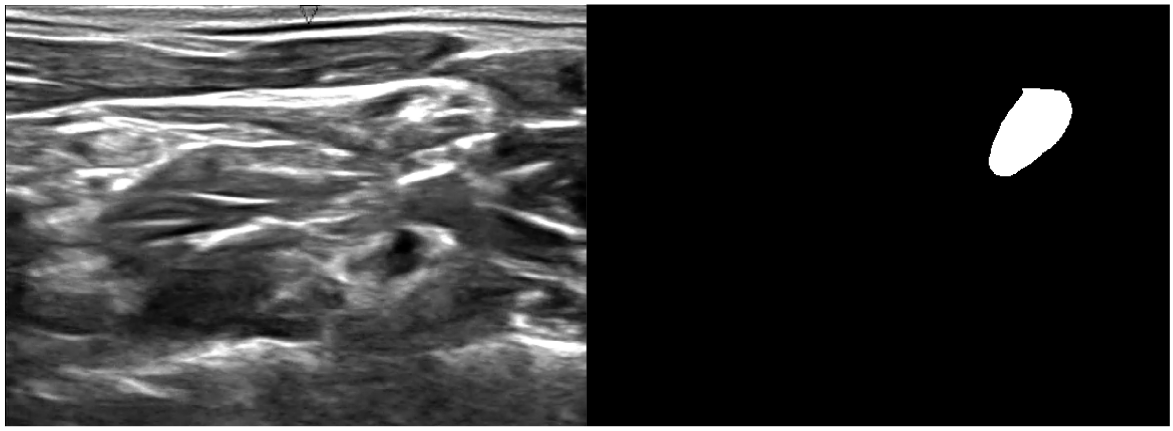

Рис. 2.8 Зображення та відповідна маска з використаного набору даних

Усього цей набір даних містить 5508 зображень і 2323 з них мають масками для навчання. Ці зображення мають розмір 580×420 пікселів, що є завеликим для використання на кластері без GPU. Тому для значного зменшення часу навчання усі зображення та маски було зменшено розміру 256×256.

Для коректної валідації результатів навчання моделі штучної нейронної мережі датасет було розділено на 3 частини: 80% зображень безпосередньо використовуються у тренуванні; 10% було виділено для валідації прогресу після каждої епохи; останні 10% зображень використано виключно для оцінювання натренованої моделі. Таким чином, якщо модель запам’ятає знімки, то це не вплине на фінальну оцінку точності моделі.

У процесі навчання моделі відстежуються 4 основні метрики: загальна піксельна точність, двійкова перехресна ентропія (с. 22), індекс Соренсена (с. 22) та міра Жаккара (с. 23).

Процесс навчання зупиняється автомтично, коли значення функції оптимізації не змінюється протягом 10 кроків.

Перед тим, як почати дослідження було виділено декілька етапі:

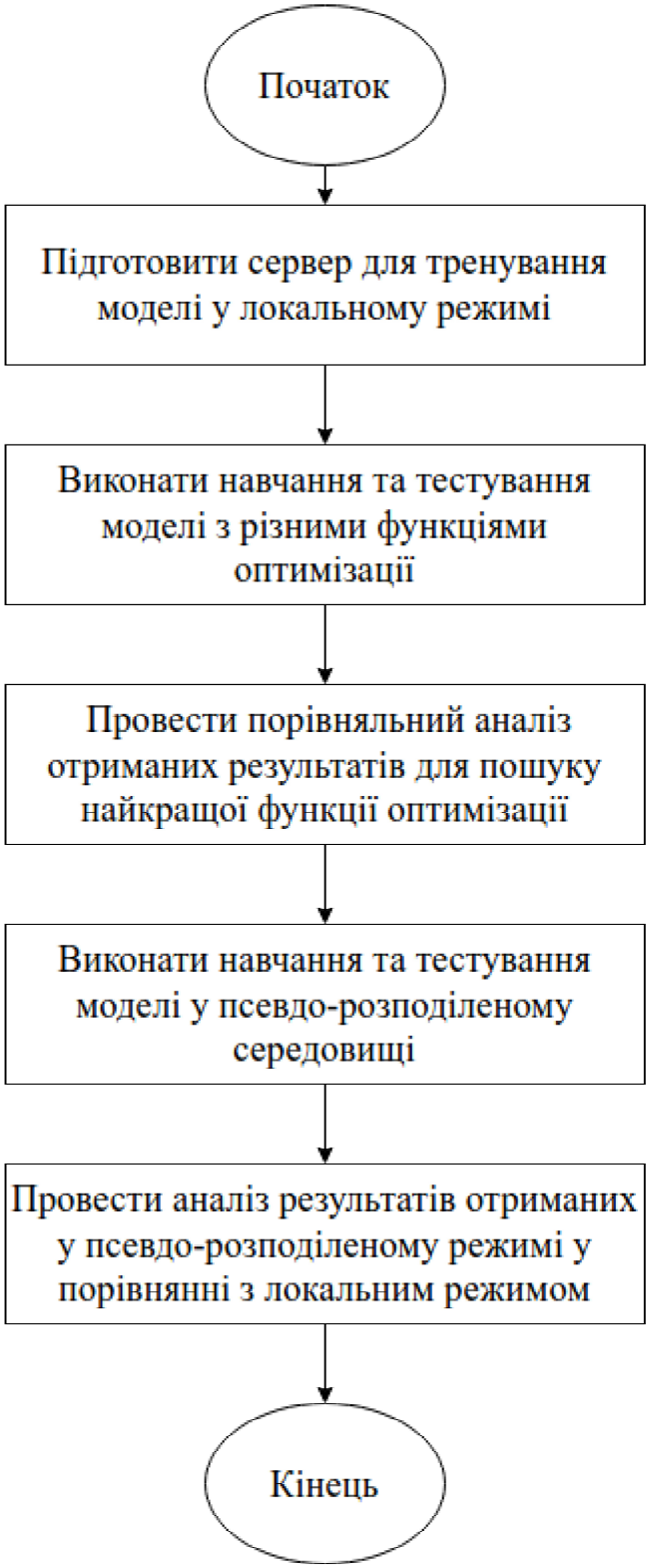

Рис. 2.9 Схема алгоритму проведення дослідження

3. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ НА ПРИКЛАДІ УЛЬТРАЗВУКОВИХ ЗОБРАЖЕНЬ ПЛЕЧОВОГО НЕРВУ

3.1 Конфігурація сервера

Всі есперименталні випробування було проведено на обчислювальному сервері з використанням Tensorflow версії 2.3.1

Апаратна конфігурація сервера:

CPU: Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz

Кількість ядер: 12

Кількість потоків: 24

L1 ксш: 32 KiB

L2 кеш: 256 KiB

L3 кеш: 15360 KiB

RAM: 64 GB

Операційна система: Ubuntu 18.04.2 LTS

3.2 Навчання у локальному режимі з використанням двійкової перехресної ентропії

При навчанні з використанням двійкової перехресної ентропії було отримано наступні результати:

Мінімальне значення фукції втрати було досягнуто на 14ту епоху за 1 годину 46 хвилини 35 секунд (6395 секунд) і становить 0.033. Піксельна точність становить 98,75%.

Таблиця 3.1

Результат тестування моделі, вивченої з використнням двійкової перехресної ентропії

Метріка	Середньє значення
Піксельна точність	0.9865
Двійкової перехресна ентропія	0.0459
Індекс Соренсена	0.2745
Міра Жаккара	0.3987

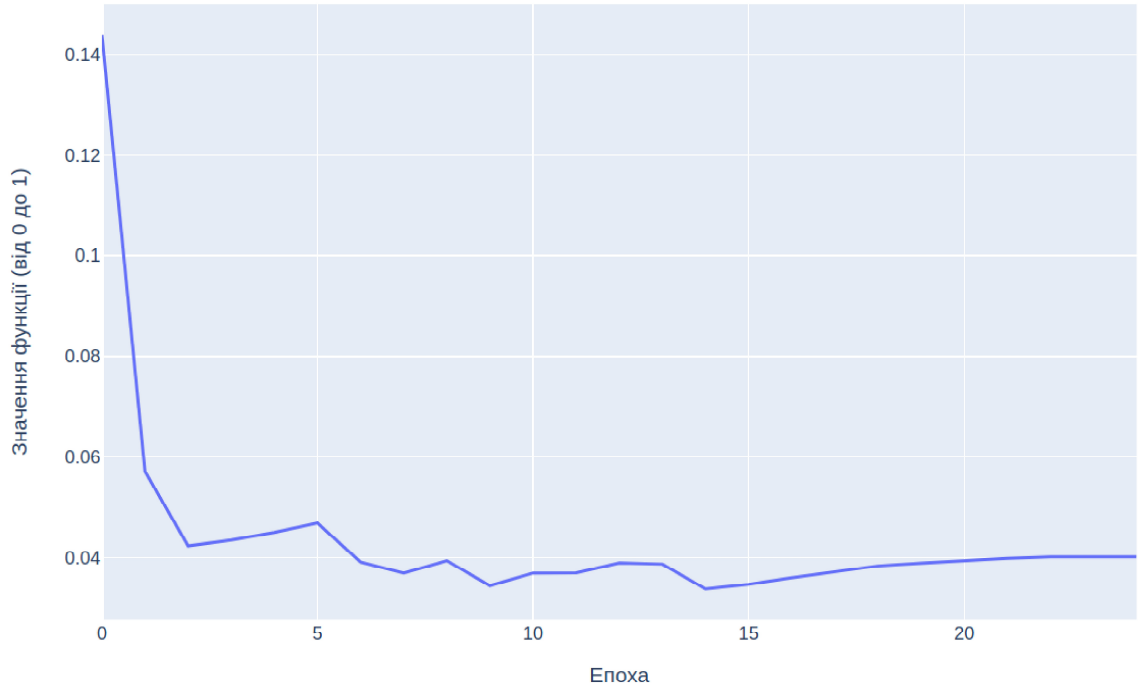

Рис. 3.2 Залежність значення функції двійкової перехресної ентропії від кількості навчальних епох

3.3 Навчання у локальному режимі з використанням індекса Соренсена

При навчанні з використанням індекса Соренсена було отримано наступні результати:

Мінімальне значення фукціїї втрати було досягнуто на 11ту епоху за 1 годину 23 хвилини 7 секунд (4987 секунд) і становить 0.142. Піксельна точність становить 98,73%.

Таблиця 3.2

Результат тестування моделі, вивченої з використнням індекса Соренсена

Метріка	Середньє значення
Піксельна точність	0.9867
Двійкової перехресна ентропія	0.1735
Індекс Соренсена	0.2307
Міра Жаккара	0.3399

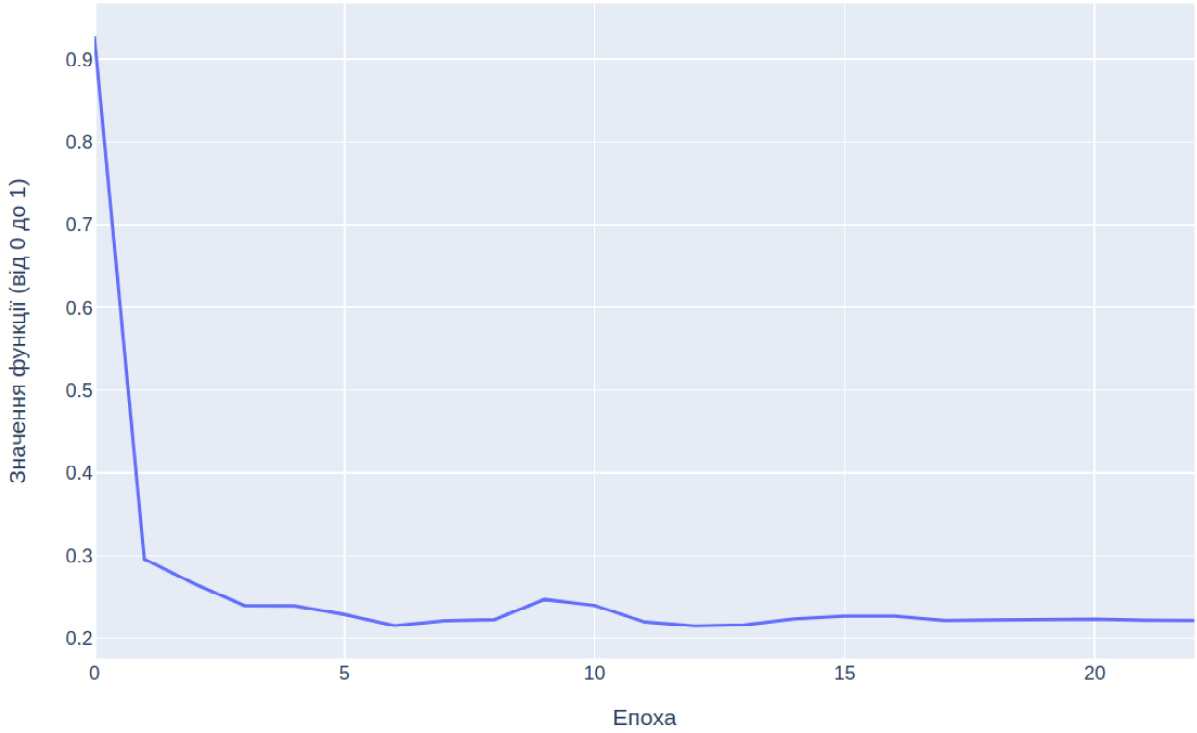

Рис. 3.3 Залежність значення індекса Соренсена від кількості навчальних епох

3.4 Навчання у локальному режимі з використанням міри Жаккара

При навчанні з використанням міри Жаккара було отримано наступні результати:

Мінімальне значення фукціїї втрати було досягнуто на 19ту епоху за 2 години 24 хвилини 31 секунду(8671 секунда) і становить 0.3248. Піксельна точність становить 98,75%.

Таблиця 3.3

Результат тестування моделі, вивченої з використнням міри Жаккара

Метріка	Середньє значення
Піксельна точність	0.9872
Двійкової перехресна ентропія	0.1699
Індекс Соренсена	0.2124
Міра Жаккара	0.3209

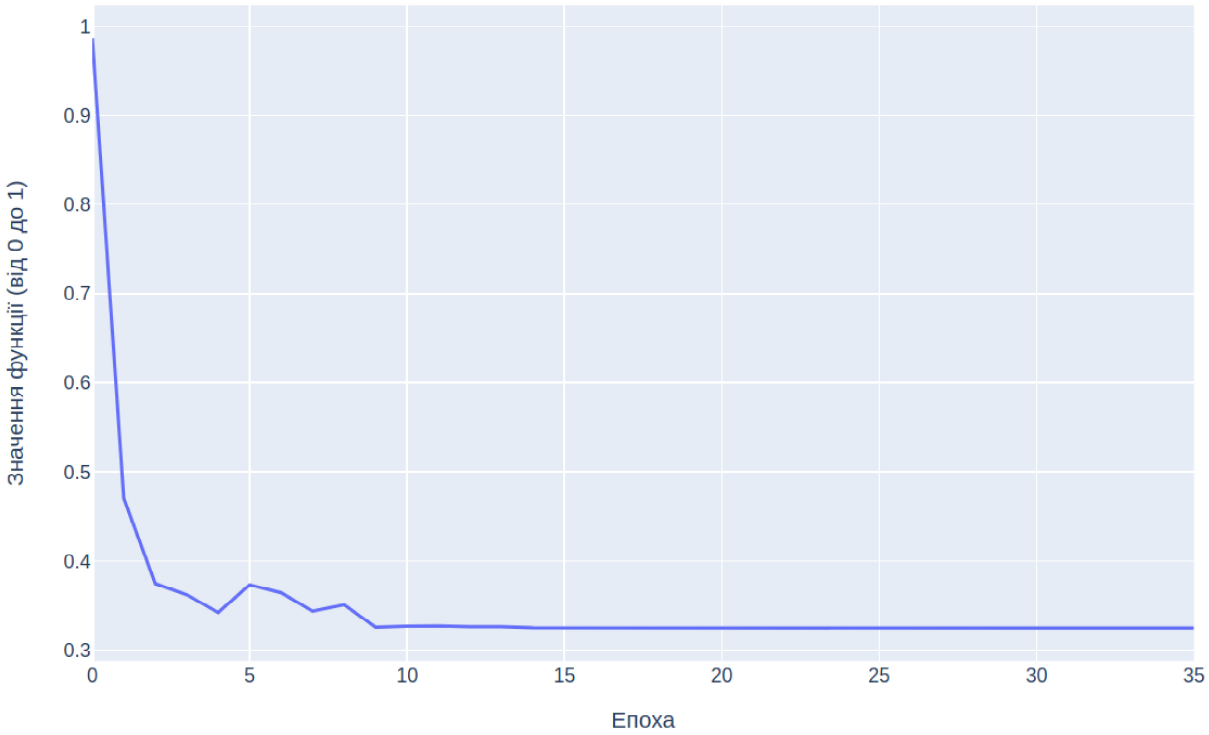

Рис. 3.4 Залежність значення міри Жаккара від кількості навчальних епох

3.5 Порівняльний аналіз результатів використання різних функцій оптимізації для тренуванні моделі у локальному режимі

В ході тестування 3х функцій втрат було з'ясовано, що використання міри Жаккара в ролі функції збитків дозволяє отримати найбільшу піксельну точніть 0.9872 (98.72%), але потребує більше епох навчання і часу. Для отримання максимальної точності також і в експерименті в псевдо-розподіленому середовищі, далі буде використано саме міру Жаккара як функцію оптимізації.

3.6 Тренування у розподіленому режимі з використанням міри Жаккара

При навчанні в розподіленому режимі з двома працівниками було отримано наступні результати:

Мінімальне значення фукціїї втрати було досягнуто на 18ту епоху за 1 годину 39 хвилин 33 секунди (5973 секунди) і становить 0.3317. Піксельна точність становить 98,73%.

Таблиця 3.4

Результат тестування моделі, тренованої у розподіленому режимі

Метріка	Середньє значення
Піксельна точність	0.9867
Двійкової перехресна ентропія	0.2025
Індекс Соренсена	0.2318
Міра Жаккара	0.3414

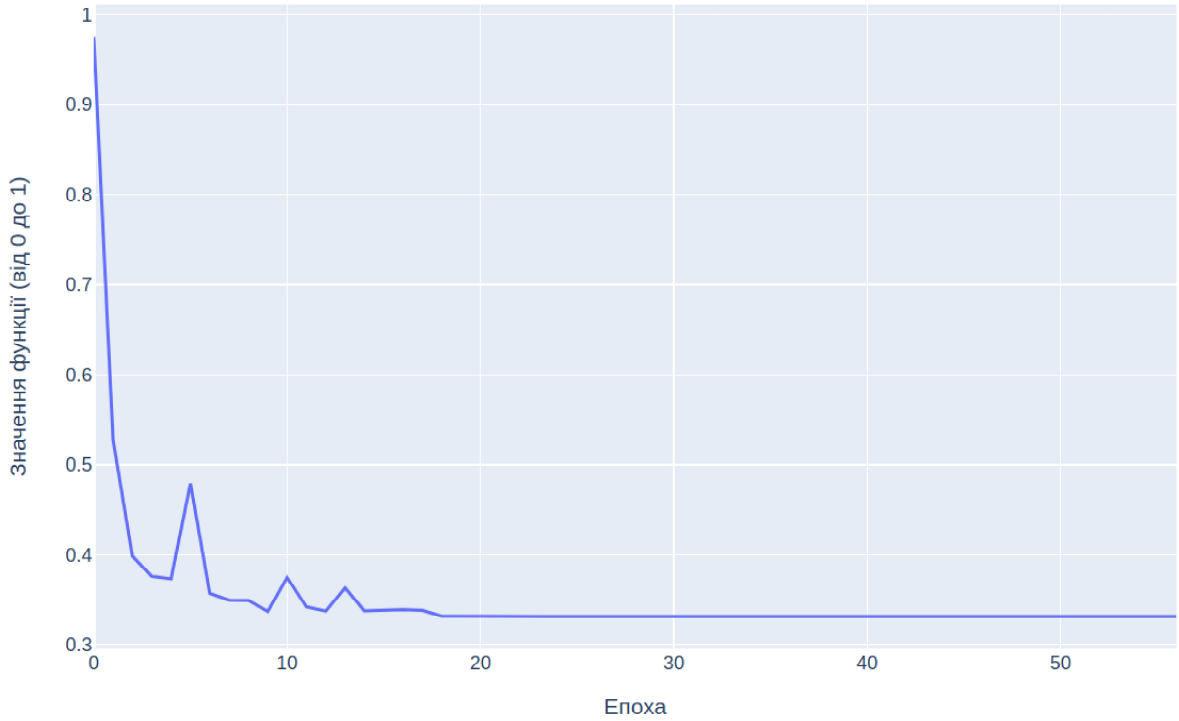

Рис. 3.5 Залежність значення міри Жаккара від кількості навчальних епох у розподіленому режимі

3.7 Аналіз результатів використання псевдо-розподіленого режиму для тренування моделі

Графіки на рис. 3.4 та 3.5 разом із результатами тестування у таблицях 3.3 та 3.4 містять дані про два ключових випробування для порівняння ефективності використання розподіленого режиму навчання. Видно, що точність сегментації для локального режиму (98.72%) загалом трохи більша, ніж при навчанні у розподіленому режимі (98.67%). З іншого боку, загальний час навчання моделі у розподіленому режимі становить 5973 секунди проти 8671 секунд у локальному режимі, що є прискоренням на 31.11%. Вірогідніше всього цей результат можна побачити через різницю в ефективній кількості навчальних даних, що обробляються за раз (тобто «ефективний розмір пакету»). Таким чинов встановлено, що можна майже завжди рекомендувати до використання розподілений режим тренування моделей штучних нейронних мереж, за виключенням випадків, коли точність передбачень є занадто критичною.

ВИСНОВКИ

Під час виконання магістреської дипломної роботи було проведено аналіз існуючих методів сегментації зображень та можливостей фреймворку машинного навчання Tensorflow для роботи в локальному та розподілених середовищах.

Для проведення експериментальнго дослідження сегментації зображень був обран датасет ультразвукових знімків плечового нерву. Сегментація медичних зображень була виконана за допомогою реалізації архітектури штучної нейронної мережі U-Net.

Перед тим як проводити експерименти з розподіленням навчання було спершс вивчсно та протсстовано моделі з дскількома різними функціями оптимізації. У результаті було встановлено, що міра Жаккара дає найкращий результат в даному сценацрії.

Далі на сервері було підготовлено та проведено навчання моделі U-Net з використанням міри Жаккара як функції оптимізації у псевдо-розподіленому середовищі.

Отримані результати показали що, використання розподіленого режиму може значно прискорити процесс навчання штучної нейронної мережі за рахунок досить невеликої втрати точності. Але, звісно, це не може бути рекомендовано у випадках, коли точність моделі є занадто критичною для досягнення встановленої мети.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gonzalez R. C. Digital Image Processing / R. C. Gonzalez, R. E. Woods. - Beijing: Publishing House of Electronics Industry. - 2007. - 2nd ed. - 244 p.

2. Image Segmentation: A Review / T. Shraddha, K. Krishna, B. K. Singhand, R. P. Singh // International Journal of Computer Science and Management Research. - 2012. - Vol. 1. - 383 p.

3. Khokher M. R. Image segmentation using multilevel graph cuts and graph development using fuzzy rule-based system / M. R. Khokher, A. Ghafoor, A. M. Siddiqui // IET image processing. - 2012. - 74 p.

4. Dey V. A review on image segmentation techniques with Remote sensing perspective / V. Dey, Y. Zhang, M. Zhong // ISPRS. - Vienna, Austria, 2010. - Vol. XXXVIII. - 204 p.

5. Inderpaland. S. A Review on Different Image SegmentationTechniques / S. Inderpaland, K. Dinesh // IJAR. - 2014. - Vol. 4. - 88 p.

6. Seerah G. K. Review on Recent Image Segmentation Techniques / G. K. Seerah, K. Rajneet // International Journal on Computer Science and Engineering (IJCSE). - 2013. - Vol. 5. - 310 p.

7. Dabhiand N. Review on Outdoor Scene image Segmentation / N. Dabhiand, H. Mewada // International Journal of Computer Science and Information Technologies(IJCSIT). - 2012. - Vol. 3. - 178 p.

8. Senthilkumaranand N. Edge Detection Techniques for ImageSegmentation - A Survey of Soft ComputingApproaches / N. Senthilkumaranand, R. Rajesh // International Journal of Recent Trends in Engineering. - 2009. - Vol. 1. - 359 p.

9. Pal N. R. A review on image segmentation techniques / N. R. Pal, S. K. Pal // Pattern recognition society. - 1993. - Vol. 29. - 244 p.

10. Dass R. Image Segmentation Techniques / R. Dass, Priyanka, S. Devi // IJECT. - 2012. - Vol. 3. - 331 p.

11. Saleh S. Image segmentation by using edge detection / S. Saleh, N. V. Kalyankar, S. Khamitkar // (IJCSE) International Journal on Computer Science and Engineering. - 2010. - Vol. 2. - 177 p.

12. Rawat G. S. Proposed Method for Image Segmentation UsingSimilarity Based Region Merging Techniques / G. S. Rawat, J. Bhattacharjee, R. Soni // (IJCSIT) International Journal of Computer Science and Information Technologies. - 2012. - Vol. 3. - 171 p.

13. Yambaland M. Image Segmentation using Fuzzy C Means Clustering: A survey / M. Yambaland, H. Gupta // International Journal of Advanced Research in Computer and Communication Engineering. - 2013. - Vol. 2. - 236 p.

14. A Novel Recursive Bayesian Learning-Based Method for the Efficient and Accurate Segmentation of Video With Dynamic Background / Q. Zhu, Z. Song, Y. Xie, L. Wang // IEEE transactions on image processing. - 2012. - Vol. 21. - 353 p.

15. The Comparative Research on Image Segmentation Algorithms : IEEE Conference on ETCS / W.X. Kang, Q. Q. Yang, R. R. Liang. - 2009. - pp. 703-707.

16. An Overview of Image and Video Segmentation in the last 40 years : Proceedings of the 6th International Symposium on Signal Processing and Its Applications / Y. J. Zhang. - 2001. - pp. 144-151.

17. Pham D. L. A survey on current methods in medical image segmentation / D. L. Pham, C. Xu, J.L. Princo // Annual Review of Biomedical Engineering. - 2000. - Vol. 2. - 256 p.

18. Marian W. An Automated Modified Region Growing Technique for Prostate Segmentation in Trans-Rectal Ultrasound Images / W. Marian. - Master‟s Thesis, Department of Electrical and Computer Engineering, University of Waterloo. - Waterloo, Ontario, Canada, 2008. - 96 p.

19. Zhang H. Image Segmentation Evaluation: A Survey of unsupervised methods / H. Zhang, J. E. Fritts, S. A. Goldman // Computer Vision and Image Understanding. - 2008. - pp. 260-280.

20. Lindeberg M. Segmentation and classification of edges using minimum description length approximation and complementary junction cues / M. Lindeberg , X. Li // Computer Vision and Image Understanding. - 1997. - Vol. 67. - 162 p.

21. An Experimental Comparison of Modern Methods of Segmentation : IEEE 8th International Symposium on SAMI / P. Karch, I. Zolotova. - 2010. - pp. 247-252.

22. Kaganami H. G. Region Based Detection versus Edge Detection / H. G. Kaganami, Z. Beij. - IEEE Transactions on Intelligent information hiding and multimedia signal processing. - 2009. - pp. 1217-1221.

23. Aurdal L. Image Segmentation beyond thresholding / L. Aurdal. - Norsk Regnescentral. - 2006. - 80 p.

24. Clustering of Image Data Set Using K-Means and Fuzzy K-Means Algorithms : International conference on CICN / V. K. Dehariya, S. K. Shrivastava, R. C. Jain. - 2010. - pp. 386-391.

25. Kettaf F. Z. A Comparison Study of Image Segmentation by Clustering Technique / F. Z. Kettaf. - Proceedings of ICSP. - 1996. - pp. 1280-1282.

26. Image Segmentation Based On Genetic Algorithm forRegion Growth and Region Merging : International Conference on Computing, Electronics and Electrical Technologies (ICCEET) / S. Angelina, L. Padma, S. H. Sureshand, .K. Veni. - 2012.

27. Narkhede H. P. Review on Image Segmentation Techniques / H.P. Narkhede // International Journal of Science and Modern Engineering. - 2013. - Vol 1. - 342 p.

28. LeCun Y. Deep learning / Y. LeCun, Y. Bengio, G. Hinton // Nature. - 2015. - 26 p.

29. Russakovsky O. ImageNet Large Scale Visual Recognition Challenge / O. Russakovsky, J. Deng, H. Su // International Journal of Computer Vision. - 2015. - 31 p.

30. Krizhevsky A. ImageNet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. E. Hinton. - Advances in Neural Information Processing Systems. - 2012. - 79 p.

31. Gulshan V. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs / V. Gulshan, L. Peng, M. Coram // JAMA. - 2016. - 250 p.

32. Esteva A. Dermatologist-level classification of skin cancer with deep neural networks / A. Esteva, B. Kuprel, R. A. Novoa // Nature. - 2017. - 115 p.

33. Bejnordi B. E. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer / B. E. Bejnordi, M. Veta, J. van Diest // JAMA. - 2017. - 486 p.

34. Lakhani P. Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks / P. Lakhani, B. Sundaram // Radiology. - 2017. - 338 p.

35. Deep learning with convolutional neural network for differentiation of liver masses at dynamic contrast-enhanced CT: a preliminary study / K. Yasaka, H. Akai, O. Abe, S. Kiryu // Radiology. - 2018. - 231p.

36. Automatic liver and lesion segmentation in CT using cascaded fully convolutional neural networks and 3D conditional random fields. Proceedings of Medical image computing and computer-assisted intervention / P. F. Christ, M. E. A. Elshaer, F. Ettlinger, S. Ourselin, L. Joskowicz, M. Sabuncu, G. Unal, W. Wells. - MICCAI. - 2016. - 366 p.

37. Kim K. H. Improving arterial spin labeling by using deep learning / K. H. Kim, S. H. Choi, S. H. Park // Radiology. - 2018. - 199 p.

38. Deep learning MR imaging-based attenuation correction for PET/MR imaging / F. Liu, H. Jang, R. Kijowski, T. Bradshaw, A. B. McMillan // Radiology. - 2018. - 224 p.

39. Chen M. C. Deep learning to classify radiology free-text reports / M. C. Chen, R. L. Ball, L. Yang // Radiology. - 2018. - 366 p.

40. Hubel D. H. Receptive fields and functional architecture of monkey striate cortex / D. H. Hubel, T. N. Wiesel // Physiol. - 1968. - 317 p.

41. Fukushima K. Neocognitron: a self organizing neural net- work model for a mechanism of pattern recognition unaffected by shift in position / K. Fukushima // Biol Cybern. - 1980.

42. Aerts H. J. Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach / H. J. Aerts, E. R. Velazquez, R. T. Leijenaar // Nat Commun. - 2014. - 268 p.

43. Lambin P. Radiomics: extracting more information from medical images using advanced feature analysis / P. Lambin, E. Rios-Velazquez, R. Leijenaar // Eur J Cancer. - 2012. - 363 p.

44. Rectified linear units improve restricted Boltzmann machines : Proceedings of the 27th International Conference on Machine Learning / V. Nair, G. E. Hinton. - 2010. - 342 p.

45. Ramachandran P. Searching for activation functions / P. Ramachandran, B. Zoph, Q. V. Le. - 2017.

46. Glorot X. Deep sparse rectifier neural networks / X. Glorot, A. Bordes, Y. Bengio. - Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. - 2011. - Vol 15. - pp 315–323.

47. Lin M. Network in network / Lin M., Q. Chen, S. Yan. - 2013. - 134 p.

48. Qian N. On the momentum term in gradient descent learning algorithms / Qian N. // Neural Netw. - 1999.

49. Kingma D. P. Adam: a method for stochastic optimization / D. P. Kingma, J. Ba. - 2014. - 269 p.

50. Ruder S. An overview of gradient descent optimization algorithms / Ruder S. - 2016. - 199 p.

51. Image Segmentation: A Review / T. Shraddha, K. Krishna, B. K. Singh, R. P. Singh // International Journal of Computer Science and Management Research. - 2012. - Vol. 1. - №4. - 296 p.

52. Scherer D. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition : Artificial Neural Networks (ICANN), 20th International Conference / D. Scherer, A. C. Müller, S. Behnke - 2010. - pp. 92–101.

53. Ultrasound Nerve Segmentation dataset [Електронний ресурс]. - Режим доступу.: https://www.kaggle.com/c/ultrasound-nerve-segmentation/data

54. Experimental research of optimizing the Apache Spark tuning: RDD vs Data Frames / Sergii Minukhin, Maksym Novikov, Natalia Brynza, Dmytro Sitnikov // SCOPUS. - 2020. - 17 с.

55. Rich feature hierarchies for accurate object detection and semanticsegmentation / R. Girshick, J. Donahue, T. Darrell, J. Malik. // CVPR, 2014. - 15 p.

56. Ren .S. Faster R-CNN: Towards realtime object detection with region proposal networks / S. Ren, K. He, R. Girshick, and J. Sun // InNIPS. - 2015. - 17 p.

57. Dai J. Instance-aware semantic segmen-tation via multi-task network cascades / J. Dai, K. He, J. Sun // CVPR, 2016. - 21 p.

58. Girshick R. Fast R-CNN / R. Girshick // ICCV, 2015. 30 p.

59. Mask R-CNN / K He, G. Gkioxari, P. Dollar, R. Girshick. - 2018. - 27 p.

60. Методичні рекомендації до виконання магістерської дипломної роботи для студентів ОПП «Компьютерні науки» другого (магістерського) рівня / Мінухін С.В., Ушакова І.О., Голубничий Д.Ю., Щербаков О.В. - ХНЕУ ім. С. Кузнеця. - Харків, 2021. - 45 с.

61. Overfeat: Integrated recognition, localization anddetection using convolutional networks / P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun. - 2013. - 86 p.

62. Shelhamer E. Fully convolutional net-works for semantic segmentation / E. Shelhamer, T. Darrell // CVPR. - 2015. - 42 p.

63. A real-time algorithm for signal analysis with the help of the wavelet transform / M. Holschneider, R. Kronland-Martinet, J. Morlet, P. Tchamitchian. // Wavelets: Time-Frequency Methods and Phase Space. - 1989. - pp. 289–297.

ДОДАТКИ

ДОДАТОК А

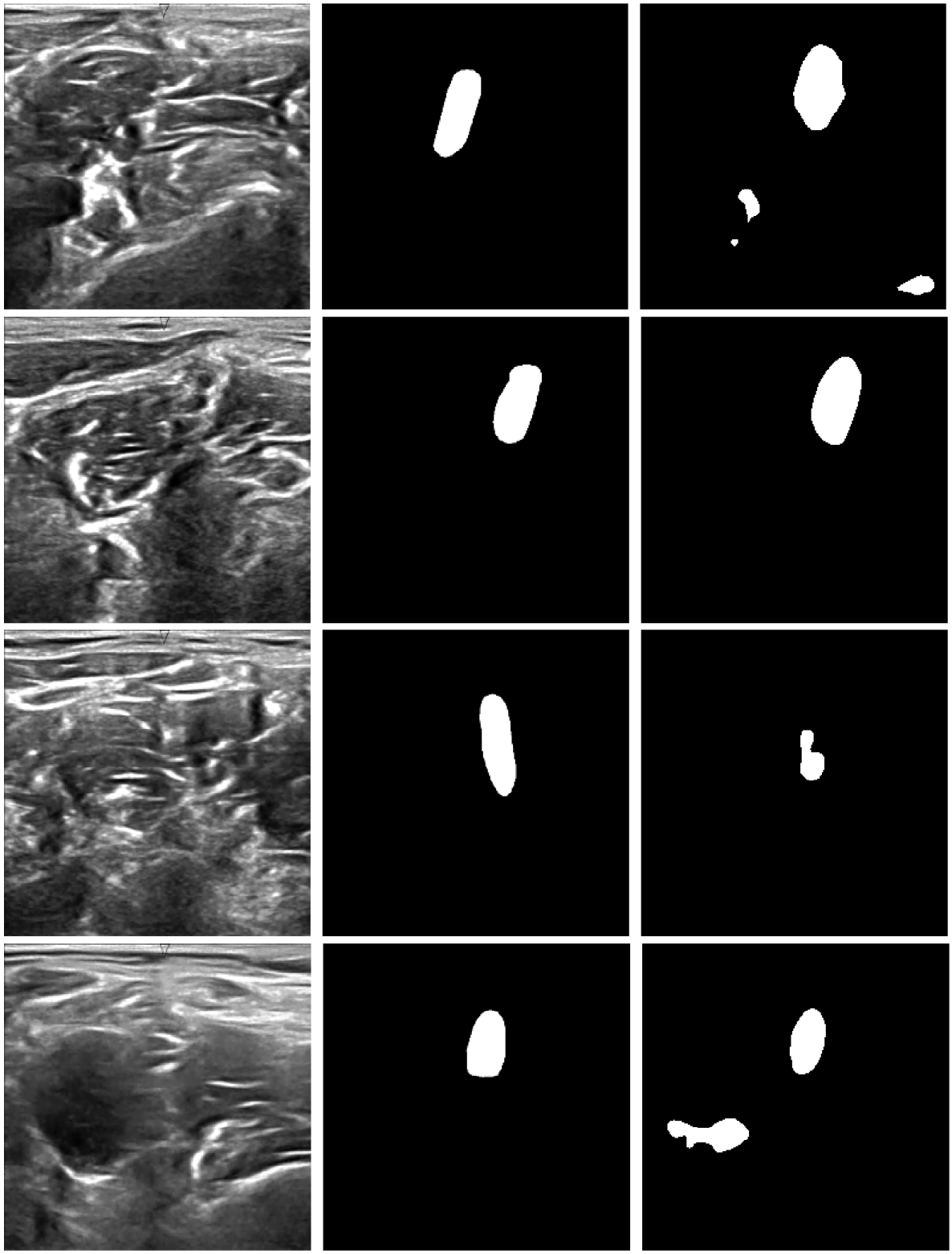

Рис. А.1-А.4 Приклади тестових передбачень моделі U-Net вивченої з використанням міри Жаккара - 1 епоха (зображення - маска - передбачення)

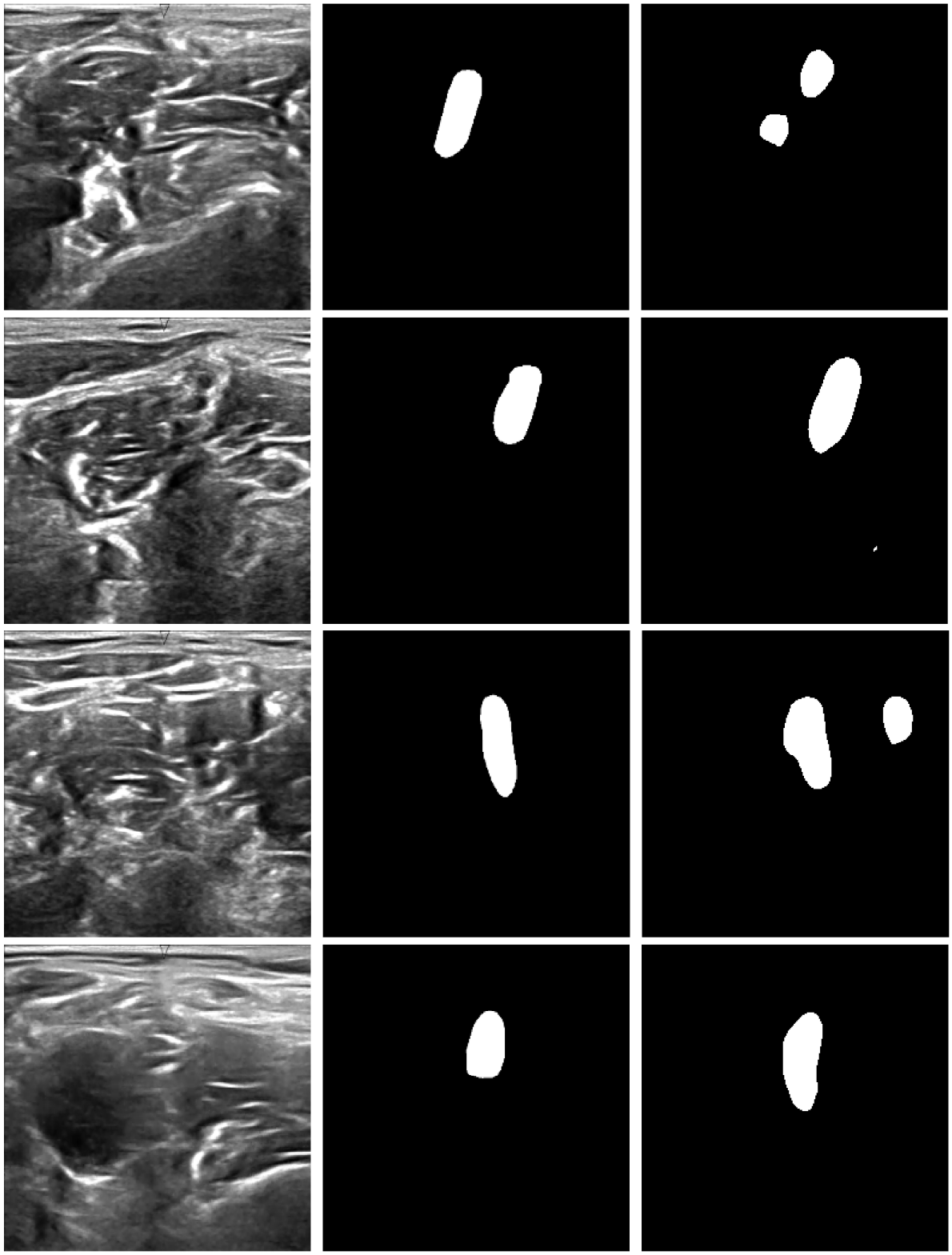

Рис. А.5-А.8 Приклади тестових передбачень моделі U-Net вивченої з використанням міри Жаккара - 5 споха (зображення - маска - передбачення)

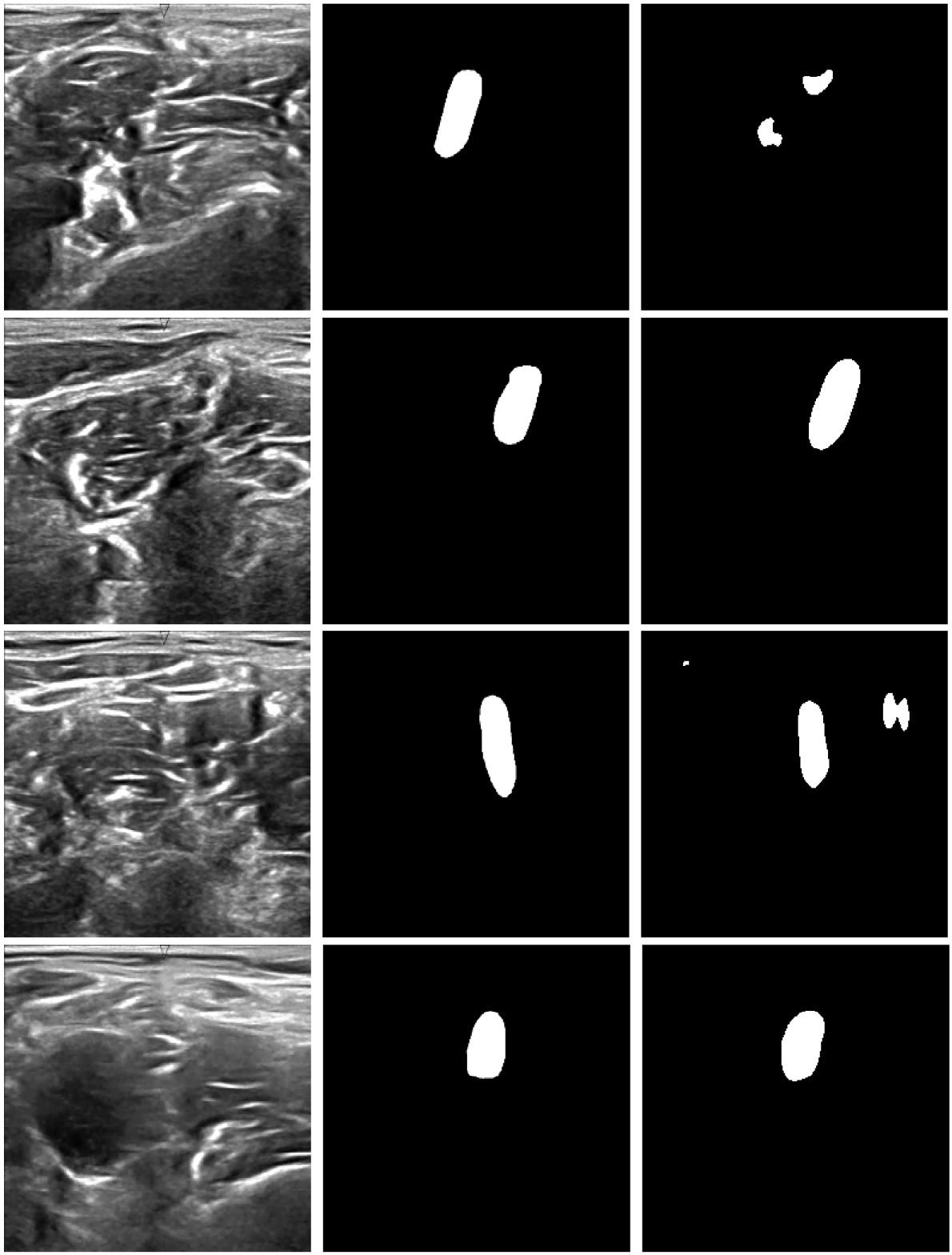

Рис. А.9-А.12 Приклади тестових передбачень моделі U-Net вивченої з використанням міри Жаккара - 10 епоха (зображення - маска - передбачення)

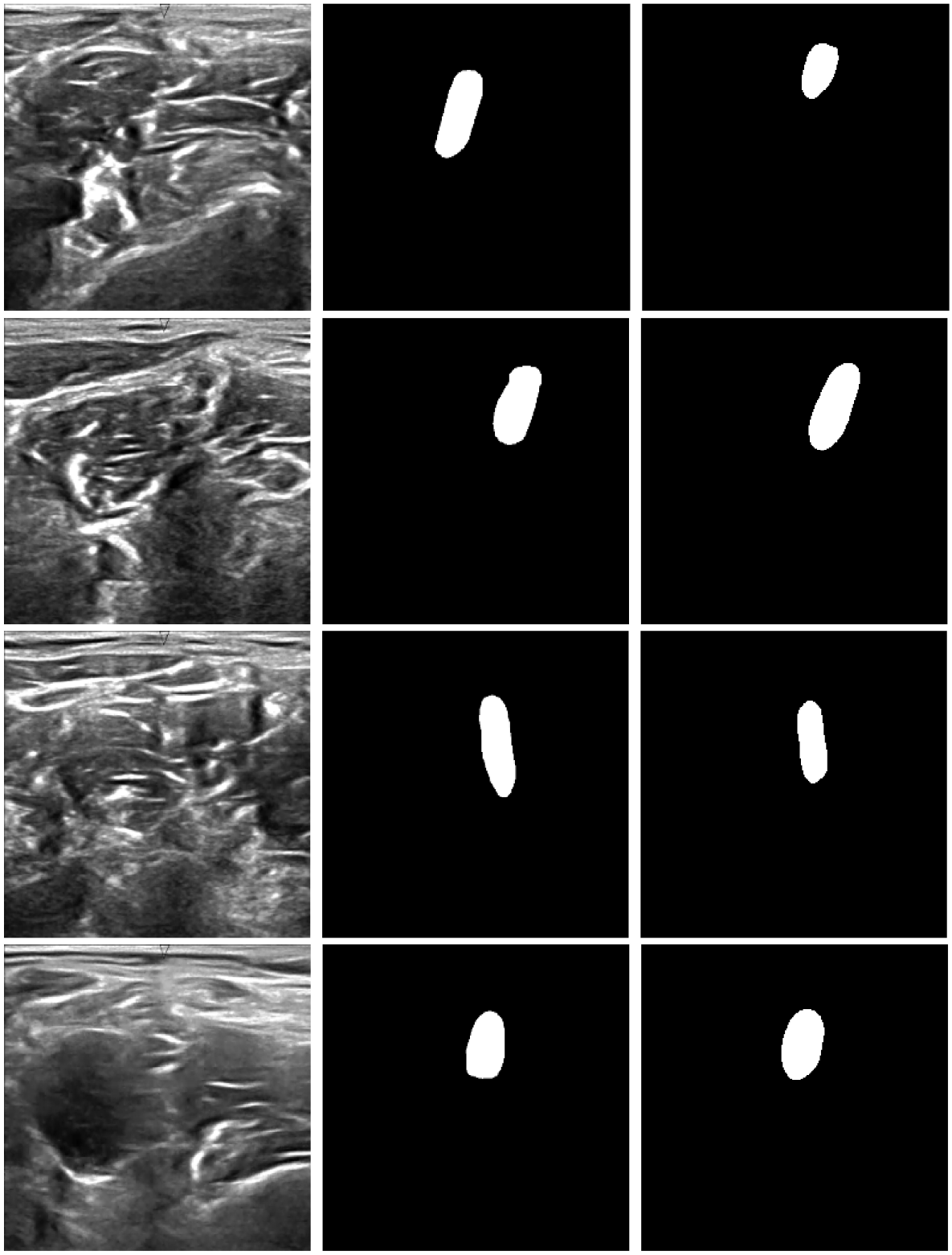

Рис. А.13-А.16 Приклади тестових передбачень моделі U-Net вивченої з використанням міри Жаккара - 19 епоха (зображення - маска - передбачення)

ДОДАТОК Б

Вихідний код програми для локального режиму навчання:

Файл main.py:

```
import argparse
import os
import numpy as np
import cv2
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau, CSVLogger, TensorBoard
from datetime import datetime
from tqdm import tqdm
from data import load_data, tf_dataset, read_image_test
from model import build_model


def dice(y_true, y_pred, smooth=1):
    intersection = tf.reduce_sum(y_true * y_pred, axis=[1, 2, 3])
    union = tf.reduce_sum(y_true, axis=[1, 2, 3]) + tf.reduce_sum(y_pred, axis=[1, 2, 3])
    dice = tf.reduce_mean((2. * intersection + smooth) / (union + smooth), axis=0)
    return 1 - dice


def iou(y_true, y_pred):
    y_true = tf.cast(y_true, tf.dtypes.float64)
    y_pred = tf.cast(y_pred, tf.dtypes.float64)
    I = tf.reduce_sum(y_pred * y_true, axis=(1, 2))
    U = tf.reduce_sum(y_pred + y_true, axis=(1, 2)) - I
    return 1 - tf.reduce_mean(I / U)


def calculate_steps(dataset, batch_size):
    steps = len(dataset) // batch_size
    if len(dataset) % batch_size != 0:
```

```
            steps += 1
        return steps


    def get_loss_function(func_name):
        return {
            'binary_crossentropy': 'binary_crossentropy',
            'iou': iou,
            'dice': dice
        }[func_name]


    if __name__ == '__main__':

        print(f"Tensorflow ver. {tf.__version__}")

        ap = argparse.ArgumentParser()

        ap.add_argument("--epochs", required=False, help="number of epochs",
type=int, default=50)
        ap.add_argument("--learning_rate", required=False, help="learning rate",
type=float, default=1e-3)
        ap.add_argument("--batch_size", required=False, help="batch size",
type=int, default=8)
        ap.add_argument("--output_dir", required=False, help="output directory",
type=str, default='results')
        ap.add_argument("--loss_function", required=False, help="loss function",
type=str, default='iou')
        args = vars(ap.parse_args())

        epochs = args['epochs']
        learning_rate = args['learning_rate']
        batch_size = args['batch_size']
        output_dir = args['output_dir']
        loss_function = args['loss_function']

        ROOT_DIR = os.path.abspath(os.curdir)
```

```
        OUTPUT_DIR = os.path.join(ROOT_DIR, output_dir, datetime.now().strftime("%d-%m-%YT%H-%M-%S"))
        PREDICTIONS_DIR = os.path.join(OUTPUT_DIR, 'predictions')

        if not os.path.exists(OUTPUT_DIR):
            os.makedirs(OUTPUT_DIR)
        if not os.path.exists(PREDICTIONS_DIR):
            os.makedirs(PREDICTIONS_DIR)

        (train_x, train_y), (valid_x, valid_y), (test_x, test_y) = load_data()

        train_dataset = tf_dataset(train_x, train_y, batch=batch_size)
        valid_dataset = tf_dataset(valid_x, valid_y, batch=batch_size)
        test_dataset = tf_dataset(test_x, test_y, batch=batch_size)

        model = build_model()
        opt = tf.keras.optimizers.Adam(learning_rate)
        metrics = ['acc', 'binary_crossentropy', iou, dice]
        model.compile(loss=get_loss_function(loss_function), optimizer=opt, metrics=metrics)

        callbacks = [
            ModelCheckpoint(os.path.join(OUTPUT_DIR, 'model.h5')),
            ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=4),
            CSVLogger(os.path.join(OUTPUT_DIR, 'data.csv')),
            TensorBoard(log_dir=OUTPUT_DIR),
            EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=False)
        ]

        train_steps = calculate_steps(train_x, batch_size)
        valid_steps = calculate_steps(valid_x, batch_size)
        test_steps = calculate_steps(test_x, batch_size)

        model.fit(train_dataset,
                validation_data=valid_dataset,
                epochs=epochs,
```

```
            steps_per_epoch=train_steps,
            validation_steps=valid_steps,
            callbacks=callbacks)

    model.evaluate(test_dataset, steps=test_steps)

    for i, (x, y) in tqdm(enumerate(zip(test_x, test_y)), total=len(test_x)):
        x = read_image_test(x)
        y = read_image_test(y)
        y_pred = model.predict(np.expand_dims(x, axis=0))[0] > 0.5
        h, w, _ = x.shape
        white_line = np.ones((h, 10, 1)) * 255.0
        all_images = [
            x * 255.0, white_line,
            y * 255.0, white_line,
            y_pred * 255.0
        ]
        image = np.concatenate(all_images, axis=1)

        if not cv2.imwrite(os.path.join(PREDICTIONS_DIR, f'{i}.png'), image):
            raise Exception('Could not write image')
```

Файл data.py:

```
import numpy as np
import cv2
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split

SEED = 42

def load_data():
    all_data = pd.read_csv("kaggle/train_masks.csv")
    all_usable_data = all_data[all_data.pixels.notnull()]

    train_dataset_info, val_dataset_info = train_test_split(all_usable_data,
test_size=0.2, random_state=SEED)
```

```
    val_dataset_info, test_dataset_info = train_test_split(val_dataset_info, test_size=0.5, random_state=SEED)

    training = get_image_names(train_dataset_info)
    validation = get_image_names(val_dataset_info)
    test = get_image_names(test_dataset_info)

    return training, validation, test



def get_image_names(dataset_info: list):
    x = []
    y = []
    for index, row in dataset_info.iterrows():
        x.append(f'kaggle/train/{row["subject"]}_{row["img"]}.tif')
        y.append(f'kaggle/train/{row["subject"]}_{row["img"]}_mask.tif')

    return x, y

def read_image(path):
    path = path.decode()
    x = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    x = cv2.resize(x, (256, 256))
    x = x / 255.0
    x = np.expand_dims(x, axis=-1)
    return x

def read_image_test(path):
    x = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    x = cv2.resize(x, (256, 256))
    x = x / 255.0
    x = np.expand_dims(x, axis=-1)
    return x

def tf_parse(x, y):
    def _parse(x, y):
```

```
        x = read_image(x)
        y = read_image(y)
        return x, y

    x, y = tf.numpy_function(_parse, [x, y], [tf.float64, tf.float64])
    x.set_shape([256, 256, 1])
    y.set_shape([256, 256, 1])
    return x, y


def tf_dataset(x, y, batch=8):
    dataset = tf.data.Dataset.from_tensor_slices((x, y))
    dataset = dataset.map(tf_parse)
    dataset = dataset.batch(batch)
    dataset = dataset.repeat()
    return dataset
```

Файл model.py:

```
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model

def conv_block(x, num_filters):
    x = Conv2D(num_filters, (3, 3), padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)

    x = Conv2D(num_filters, (3, 3), padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)

    return x


def build_model():
    size = 256
    num_filters = [16, 32, 48, 64]
```

```
inputs = Input((size, size, 1))

skip_x = []
x = inputs

## Encoder
for f in num_filters:
    x = conv_block(x, f)
    skip_x.append(x)
    x = MaxPool2D((2, 2))(x)

## Bridge
x = conv_block(x, num_filters[-1])

num_filters.reverse()
skip_x.reverse()

## Decoder
for i, f in enumerate(num_filters):
    x = UpSampling2D((2, 2))(x)
    xs = skip_x[i]
    x = Concatenate()([x, xs])
    x = conv_block(x, f)

## Output
x = Conv2D(1, (1, 1), padding="same")(x)
x = Activation("sigmoid")(x)

return Model(inputs, x)
```

ДОДАТОК В

Вихідний код програми для розподіленого режиму навчання:

Відрізняється від програми для локального режему тільки файлом main.py.

Файл main.py:

```
import argparse
import os
import numpy as np
import cv2
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau, CSVLogger, TensorBoard
from tensorflow.python.keras.utils.generic_utils import CustomObjectScope
from datetime import datetime
from tqdm import tqdm
from data import load_data, tf_dataset, read_image_test
from model import build_model


def dice(y_true, y_pred, smooth=1):
    intersection = tf.reduce_sum(y_true * y_pred, axis=[1, 2, 3])
    union = tf.reduce_sum(y_true, axis=[1, 2, 3]) + tf.reduce_sum(y_pred, axis=[1, 2, 3])
    dice = tf.reduce_mean((2. * intersection + smooth) / (union + smooth), axis=0)
    return 1 - dice


def iou(y_true, y_pred):
    y_true = tf.cast(y_true, tf.dtypes.float64)
    y_pred = tf.cast(y_pred, tf.dtypes.float64)
    I = tf.reduce_sum(y_pred * y_true, axis=(1, 2))
    U = tf.reduce_sum(y_pred + y_true, axis=(1, 2)) - I
    return 1 - tf.reduce_mean(I / U)


def calculate_steps(dataset, batch_size):
```

```
        steps = len(dataset) // batch_size
        if len(dataset) % batch_size != 0:
            steps += 1
        return steps


    def get_loss_function(func_name):
        return {
            'binary_crossentropy': 'binary_crossentropy',
            'iou': iou,
            'dice': dice
        }[func_name]


    if __name__ == '__main__':

        print(f"Tensorflow ver. {tf.__version__}")
        strategy = tf.distribute.experimental.MultiWorkerMirroredStrategy(
            tf.distribute.experimental.CollectiveCommunication.RING)

        ap = argparse.ArgumentParser()

        ap.add_argument("--epochs", required=False, help="number of epochs", type=int, default=50)
        ap.add_argument("--learning_rate", required=False, help="learning rate", type=float, default=1e-3)
        ap.add_argument("--batch_size", required=False, help="batch size", type=int, default=8)
        ap.add_argument("--output_dir", required=False, help="output directory", type=str, default='results')
        ap.add_argument("--loss_function", required=False, help="loss function", type=str, default='iou')
        args = vars(ap.parse_args())

        epochs = args['epochs']
        learning_rate = args['learning_rate']
        batch_size = args['batch_size']
```

```
        output_dir = args['output_dir']
        loss_function = args['loss_function']

        ROOT_DIR = os.path.abspath(os.curdir)
        OUTPUT_DIR = os.path.join(ROOT_DIR, output_dir, datetime.now().strftime("%d-%m-%YT%H-%M-%S"))
        PREDICTIONS_DIR = os.path.join(OUTPUT_DIR, 'predictions')

        if not os.path.exists(OUTPUT_DIR):
            os.makedirs(OUTPUT_DIR)
        if not os.path.exists(PREDICTIONS_DIR):
            os.makedirs(PREDICTIONS_DIR)

        (train_x, train_y), (valid_x, valid_y), (test_x, test_y) = load_data()

        train_dataset = tf_dataset(train_x, train_y, batch=batch_size)
        valid_dataset = tf_dataset(valid_x, valid_y, batch=batch_size)
        test_dataset = tf_dataset(test_x, test_y, batch=batch_size)

        with strategy.scope():
            model = build_model()
            opt = tf.keras.optimizers.Adam(learning_rate)
            metrics = ['acc', 'binary_crossentropy', iou, dice]
            model.compile(loss=get_loss_function(loss_function), optimizer=opt, metrics=metrics)

        callbacks = [
            ModelCheckpoint(os.path.join(OUTPUT_DIR, 'model.h5')),
            ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=4),
            CSVLogger(os.path.join(OUTPUT_DIR, 'data.csv')),
            TensorBoard(log_dir=OUTPUT_DIR),
            EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=False)
        ]

        train_steps = calculate_steps(train_x, batch_size)
        valid_steps = calculate_steps(valid_x, batch_size)
```

```
test_steps = calculate_steps(test_x, batch_size)

model.fit(train_dataset,
        validation_data=valid_dataset,
        epochs=epochs,
        steps_per_epoch=train_steps,
        validation_steps=valid_steps,
        callbacks=callbacks)

model.evaluate(test_dataset, steps=test_steps)

for i, (x, y) in tqdm(enumerate(zip(test_x, test_y)), total=len(test_x)):
    x = read_image_test(x)
    y = read_image_test(y)
    y_pred = model.predict(np.expand_dims(x, axis=0))[0] > 0.5
    h, w, _ = x.shape
    white_line = np.ones((h, 10, 1)) * 255.0
    all_images = [
        x * 255.0, white_line,
        y * 255.0, white_line,
        y_pred * 255.0
    ]
    image = np.concatenate(all_images, axis=1)

    if not cv2.imwrite(os.path.join(PREDICTIONS_DIR, f'{i}.png'), image):
        raise Exception('Could not write image')
```