

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ**

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

Пояснювальна записка

до дипломної роботи

МАГІСТРА

на тему: "МОДЕЛЮВАННЯ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ З
ЗАСТОСУВАННЯМ НЕЙРОННОЇ МЕРЕЖІ FASTFCN"

Виконав:

студент 2 року навчання

групи 8.04.122.010.20.1

спеціальності 122 "Комп'ютерні науки"

Сиса Артем Сергійович

Керівник:

д.т.н., проф. Мінухін Сергій Володимирович

Харків – 2021 рік

РЕФЕРАТ

Пояснювальна записка до магістерської дипломної роботи містить: 74 стор., 27 рис., 1 табл., 5 додатків, 47 джерел.

Метою роботи є дослідження методу сегментації зображень з використанням нейронної мережі FastFCN, визначення ефективності його використання для роботи з медичними зображеннями.

Об'єктом дослідження є процес обробки медичних зображень для інтерпретації в системах технічного зору.

Предметом дослідження є алгоритм та метод сегментації зображень FastFCN.

Результатами дослідження є метод, алгоритми та програмне забезпечення проведення семантичної сегментації медичних МРТ-зображень. Програмне забезпечення реалізована на мові Python з використанням фреймворку PyTorch.

МОДЕЛЬ, НЕЙРОННА МЕРЕЖА, FAST FCN, МРТ, JOINT PYRAMID UPSAMPLING, СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ, PYTORCH, TENSORFLOW.

ABSTRACT

Explanatory note to the master's thesis contains:

74 p., 27 fig., 1 table., 5 app, 47 sources.

The aim of the work is to study the method of image segmentation using the Fast FCN neural network, to determine the effectiveness of its use for working with medical images.

The object of research is the process of processing medical images for interpretation in technical vision systems.

The subject of the study is the algorithm and method of image segmentation Fast FCN.

The results of the study are the method, algorithms and software for semantic segmentation of medical images in the form of MRI. The software is implemented in Python using the PyTorch frequency.

MODEL, NEURAL NETWORK, FAST FCN, MRI, JOINT PYRAMID UPSAMPLING, IMAGE SEGMENTATION, PYTORCH, TENSORFLOW.

ЗМІСТ

ВСТУП.....	8
1. ОГЛЯД МЕТОДІВ СЕГМЕНТАЦІЇ ДАНИХ МРТ-ОБРАЖЕНЬ	10
1.1 Структурні методи.....	10
1.1.1 Граничні методи (3D-методики виявлення країв)	10
1.1.2 Морфологічні методи (морфологічні прийоми)	10
1.1.3 Алгоритми пошуку графа.....	11
1.1.4 Моделі активного контуру (активні контурні моделі)	12
1.1.5 Динамічні моделі, що деформуються (dynamic deformable models) ...	13
1.1.6 Імовірнісні деформовані моделі (probabilistic deformable models).....	14
1.1.7 Активний контур та потік вектора градієнта (active contours and gradient vector flow).	14
1.2 Стохастичні методи.	16
1.2.1 Порогова класифікація (thresholding approaches).....	16
1.2.2 Класифікатори	16
1.2.3 Кластерний аналіз (clustering algorithms)	18
1.2.4 Випадкове поле Маркова (Markov random fields)	22
1.3 Змішані методи	23
1.3.1 Нарощування області (region growing)	23
1.3.2 Метод "розділити та об'єднати" (split and merge)	24
1.3.3 Методи, засновані на атласах (atlas-guided approaches).....	25
1.3.4 Штучні нейронні мережі	25
1.3.5 Модель LEGION	26
2 НЕЙРОННА МЕРЕЖА FAST FCN. АРХІТЕКТУРА ТА ПРИНЦИПИ ФУНЦІОНУВАННЯ	29
2.1 Огляд методу	29
2.2 Пов'язані роботи	33
2.2.1 Семантична сегментація.....	33
2.2.2 DilatedFCN	33
2.2.3 EncoderDecoder.	33

2.2.4 Підвищення вибірки	34
2.2.5 Joint Upsampling	34
2.3 Функціонування FastFCN.....	34
2.3.1 DilatedFCN	35
2.3.2 Основа методу.....	35
2.3.3 Joint Pyramid Upsampling.....	36
2.3.4 Зміна до Joint Upsampling.....	39
3. ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ ТА АНАЛІЗ РЕЗУЛЬТАТІВ.....	42
3.1 Налаштування середовища експерименту.	43
3.2 Підготовка даних до навчання.	43
3.3 Організація та проведення експериментів	50
3.4 Збір результатів.	51
3.5 Обробка результатів.	51
3.6 Порівняння результатів експериментів на ПК з різними архітектурами ..	56
ВИСНОВКИ	60
ДОДАТКИ	65
ДОДАТОК А. Вихідний код програми перетворення NifTy файлу до PNG-файлу	65
ДОДАТОК Б. Вихідний код навчання нейронної мережі.....	67
ДОДАТОК В. Вихідний код скрипту тестування нейронної мережі	71
ДОДАТОК Г. Вихідний код тестування параметру FPS.....	73
ДОДАТОК Д. Вихідний код файлів Dockerfile та docker-compose для запуску програми на Linux	74

ВСТУП

Завдання аналізу зображень, відео та інших даних є дуже актуальною в даний час, особливо в медицині. Просторова сегментація – важлива складова медичних програм з діагностики та аналізу анатомічних даних. Разом зі стрімким розвитком обчислювальної техніки гостро постає питання про розробку методів і підходів до обробки медичних зображень. Найчастіше постановка правильного діагнозу хвороби ґрунтується на інформації, отриманої за допомогою магнітно-резонансного дослідження, позитронно-емісійного дослідження, рентгена та інших. У зв'язку з цим виникає необхідність в розробці ефективних алгоритмів сегментації і, зокрема, просторової сегментації. Ручна обробка тривимірних зображень вимагає великих зусиль і часто загрожує помилками. Інтерпретація ж тривимірного медичного зображення вимагає від лікарів перебудови способу мислення і може призводити до великих різночитань. Алгоритми, що автоматизують цей процес, можуть допомогти медикам впоратися з великим обсягом даних, забезпечивши їм при цьому надійну підтримку для діагностування і лікування. У використується велика кількість складних приладів і апаратно-програмних комплексів. Однак, незважаючи на всю різноманітність технічних засобів, доступних медикам, все ще залишається відкритим ряд проблем.

Однією з таких проблем є обробка медичних зображень. Складність аналізу та обсяги даних бувають дуже великими. Це тягне за собою складність обробки і високу ймовірність помилки при відсутності будь-яких засобів автоматизації. У зв'язку з цим проблема автоматизації процесу аналізу медичних зображень і сегментації, зокрема, є актуальною. Сегментація медичних зображень створює широкий спектр можливостей по дослідженню анатомічних структур без безпосереднього контакту з пацієнтом. Так, наприклад, дані, отримані за допомогою МРТ або КТ, дають можливість створювати віртуальні карти кровоносних судин, кишечника, досліджувати області інтересу (тканини, органи, пухлини, інші ділянки). На сьогоднішній день існує велика різноманітність алгоритмів сегментації тривимірних зображень, що дають оптимальні результати. Однак не існує одного універсального алгоритму, придатного для будь-якого класу даних. Крім того, різні алгоритми ведуть себе по-різному при обробці досить великих обсягів даних. У зв'язку з цим доцільно розробити або обрати алгоритм, який міг би одночасно вирішувати обчислювально складні завдання за прийнятний час і був би стійкий до шумів і артефактів.

Об'єкт дослідження – процес обробки медичних зображень для інтерпретації в системах технічного зору.

Предмет дослідження – метод та алгоритм сегментації зображень Fast FCN.

Методи досліджень. В даній роботі використовуються метод досліджень, такі як опис, аналіз, практичний експеримент.

Мета і завдання дослідження:

- підвищення ефективності обробки медичних МРТ-зображень.

Досягнення поставленої мети вимагає вирішення наступних завдань:

- дослідити методи обробки МРТ-зображень із застосування FastFCN;
- дослідити особливості, характеристики медичних зображень мозку людини;

- експериментально підтвердити ефективність досліджуваного методу на наборах реальних МРТ-зображень мозку пацієнтів, що знаходяться на різних стадіях захворювання;

- на базі досліджуваного методу і моделей розробити програмне забезпечення для обробки МРТ-зображень мозку, яка дозволяє підвищити швидкість і точність сегментації зображень.

Практичне значення отриманих результатів. Досліджуваний метод та алгоритм сегментації зображень може успішно застосовуватися в областях науки і техніки, пов'язаних з обробкою візуальної інформації для контекстної інтерпретації інформації, зокрема в медичних комплексах, які пов'язані з обробкою зображень, а також в системах обробки графічної інформації. Аналіз експериментальних досліджень показує, що застосування розробленого методу та алгоритмів сегментації зображень забезпечує вирішення цілого класу задач обробки зображень та їх контекстної інтерпретації.

1. ОГЛЯД МЕТОДІВ СЕГМЕНТАЦІЇ ДАНИХ МРТ-ОБРАЖЕНЬ

1.1 Структурні методи

Структурні методи, засновані на використанні інформації про структуру сегментованої області (об'єкта).

1.1.1 Граничні методи (3D-методики виявлення країв)

При перетині областей з різною інтенсивністю утворюються лінії перетину (у двовірному випадку) або ж поверхні перетину (у трьохвірному випадку) [28]. Граничні методи направлені на виявлення цих ліній/поверхонь.

Граничний метод працює в два етапи. Спочатку шляхом диференціювання знаходяться всі локальні границі. Далі ці границі групуються разом для отримання граничного контуру.

Метод успішно застосовується з високою контрастністю зображення між різними областями. Однак метод виявляє всі границі, і буває дуже складно знайти кореляцію між областю яка нас цікавить та отриманими границями. Крім того, метод сприятливий до шуму.

Граничні методи самі по собі не є методами сегментації. Як правило, вони використовуються в парі з яким-небудь іншим алгоритмом сегментації для вирішення часткової задачі

1.1.2 Морфологічні методи (морфологічні прийоми)

Математична морфологія призначена для дослідження структури і форми безлічі однотипних об'єктів [32]. Будь-яке зображення в комп'ютерній графіці можна представити у вигляді набору пікселів. Тому операції математичної морфології можуть бути виконані і до зображення.

Розглянемо приклад двійкової морфології. Нехай є подвійне зображення, представлене у вигляді упорядкованого набору чорно-білих точок, або «0» і «1». Під областю зображення зазвичай розуміється безліч єдиних зображень. Кожна операція математичної морфології є перетворенням цієї множини. В якості вихідних даних приймаються двійкове зображення та деякий структурний елемент, що кодує інформацію про просту форму. Структурний елемент може бути випадковою формою і структурою. Найчастіше використовуються симетричні елементи. В кожному елементі виділяється певна точка, яка називається початковою. Ця точка може бути розташована в будь-якому місці структурного елемента. У симетричних елементах це, як правило, центральний піксель.

Спочатку все зображення заповнюється нулями, що формують фон. Потім здійснюється сканування зображення структурним елементом у кожній точці. Для цього структурного елемента накладається зображення таким чином, щоб поєднати початкову і точку з точкою що сканується. Далі перевіряється друга умова відповідності пікселів структурного елемента та зображення «під ним». Якщо умова виконується, то на зображенні ставиться «1» (іноді можуть додати всі єдині елементи зі структурного елемента).

По розглянутій вище схеми виконуються базові операції математичної морфології. Такими операціями є розширення (розширення) і звуження (ерозія). Виробничі операції – це деяка комбінація базових операцій, що виконуються послідовно. Основними з них є відкриття і закриття. Таким чином, перетворене зображення є функція розподілу структурного елемента в цілому зображенні.

Морфологічні операції прості для розуміння та виконання, однак вимагають деякого зовнішнього критерію для контролю над ними. Ці операції можуть змінити морфологію вихідних даних. Тому слід використовувати подібні алгоритми, коли від самого початку важлива точність.

Морфологічні методи самі по собі не є алгоритмами сегментації. Вони використовуються в якості проміжного етапу в процесі сегментації [11, 23].

1.1.3 Алгоритми пошуку графа

У цих алгоритмах [33] зображення об'єкта представляється у вигляді графа, вершинами якого є вугли пікселів, а ребрами – сторонами пікселів. За допомогою функції градієнта, або будь-якого іншого оператора виявлення, границі кожному ребру присвоюється вага (cost). Ті ребра, які розділяють схожість за інтенсивністю пікселів, мають велике вагове значення, тоді як «граничні» ребра – маленьке вагове значення (lowest-cost path).

Вага ребра, що розділяє два пікселі p і q , визначається так:

$$c(p,q)=I_{\max}-|I(p)-I(q)|, \quad (1.1)$$

де $I(p)$ та $I(q)$ – значення інтенсивності в точках p та q відповідно,

I_{\max} – максимальне значення інтенсивності на цьому зображенні.

Основними перевагами даного алгоритму є те, що він добре працює навіть при нечіткому розподілі між сегментованими областями. У той же час алгоритм вимагає представлення зображень у вигляді графа, що може виявитися складною задачею. Ще одним недоліком (з точки зору об'ємної візуалізації) є те, що

алгоритм пов'язаний з поверхнею об'єкта, тому необхідний інший метод для воксельного представлення.

1.1.4 Моделі активного контуру (активні контурні моделі)

Модель активного контуру [34] – це плоска крива, параметрично задана в області зображення. Всі властивості кривої, а також її поведінка визначаються через функцію, звану функціоналом енергії (по аналогії з фізичною системою). Динамічне рівняння у частинних похідних, що описує рух кривої, змушує її деформуватися таким чином, щоб мінімізувати функціональну енергію. Фізичний аналог можна розширити, розглянути рух кривої, як рух під дією зовнішніх і внутрішніх сил.

Припустимо, крива рухається по зображенню, представленого мапою границь (edge map). На криві діють внутрішня і зовнішні сили, змушуючи її деформуватися в пошуку рівноважного стану, якому відповідає мінімум енергії. Внутрішня сила повідомляє кривій як себе вести (зберігати початкову форму, утворювати кут і т.д.), тоді як зовнішня сила керує напрямком руху (куди рухатися). Крива повинна примкнути до кордону об'єкта, який може бути розпізнаний за значенням масштабної функції потенціалу. Кінцева форма кривої буде відповідати мінімуму енергії.

Функціонал енергії, який необхідно мінімізувати, визначався таким чином:

$$E(v) = E_{\text{внутр}}(v(s)) + E_{\text{зовн}}(v(s)), \quad (1.2)$$

де $v(s) = (x(s), y(s))$, $s \in [0;]$ – є параметричне завдання кривої на площині;

$E_{\text{внутр}}$ – внутрішня енергія кривої за рахунок вигинів,

$E_{\text{зовн}}$ – зовнішня сила, що штовхає контур до меж об'єкта

$$E_{\text{внутр}}(v(s)) = \int_0^1 \left(\alpha(s) \left(\frac{\partial v(s)}{\partial s} \right)^2 + \beta(s) \left(\frac{\partial^2 v(s)}{\partial s^2} \right)^2 \right) ds. \quad (1.3)$$

Перша похідна дає швидкість зміни довжини кривої або поздовжнього згортання. Коефіцієнт $\alpha(s)$, який називається коефіцієнтом гнучкості, дозволяє кривій мати різну ступінь стиснення. Великі значення $\alpha(s)$ позначають сильне стиснення контуру в напрямленні сили. Друга похідна дає швидкість згинання кривої, або кривизну. Коефіцієнт $\beta(s)$ регулює кривизну контуру в напрямленні норми до його границі. Якщо значення $\beta(s)$ велике, то крива є жорсткою і

пручається вигинам. Тоді як при малих значеннях $\beta(s)$ крива може утворювати вугли.

Шляхом узгодження цих двох коефіцієнтів активного контуру дає відповідну пружність і здатний охопити сегментований об'єкт.

$$E_{\text{зовн}}(v(s)) = \int_0^1 P(v(s)) ds, \quad (1.4)$$

де $P(v(s))$ – скалярна функція потенціалу, визначена на площині зображення.

Ця функція задається таким чином, щоб її локальний мінімум співпадав з кордоном об'єкта або якими або іншими характеристиками зображення.

Так, наприклад, при виборі потенційного виду $P(x, y) = -c[\Delta(G_\sigma * I(x, y))]$ крива буде йти до границі об'єкта; де коефіцієнт «с» контролює величину потенціалу; $G_\sigma * I(x, y)$ дає згортку гауссіана (згладжуючий фільтр) та зображення; ширина σ регулює просторове розширення локального мінімуму $P(x, y)$.

Кінцева форма кривої $v(s)$, що мінімізує функціонал енергії $E(v)$, повинна задовольняти рівняння Ейлера-Лагранжа:

$$-\frac{\partial}{\partial s} \left(\frac{\alpha(s)(\partial v(s))}{\partial s} \right) + \frac{\partial^2}{\partial s^2} \left(\beta(s) \frac{\partial^2 v(s)}{\partial s^2} \right) + \nabla P(v(s, t)) = 0. \quad (1.5)$$

Це рівняння виражає баланс внутрішніх та зовнішніх сил, у якому крива $v(s)$ залишається у рівновазі. Підхід до вирішення рівняння 1.5 ґрунтується на застосуванні чисельних методів [6, 4, 26].

1.1.5 Динамічні моделі, що деформуються (dynamic deformable models)

Проблема мінімізації функціоналу енергії може бути розглянута з точки зору динамічної системи, яка, будучи керованою функціоналом, розвивається до рівноваги (динамічні ДМ). Такий підхід дозволяє об'єднати опис форми і руху, уможливаючи кількісно визначити як статичну форму, а й розвиток форми у часі.

Простим прикладом динамічної ДМ є крива $v(s, t) = (x(s, t), y(s, t))$ із щільністю мас $\mu(s)$ та щільністю загасання $\gamma(s)$. Рух такої кривої можна описати за допомогою рівняння Лагранжа:

$$\mu(s) \frac{\partial^2 v(s,t)}{\partial t^2} + \partial v \frac{(s,t)}{\partial t} - \frac{\partial}{\partial s} \left(\alpha(s) \frac{\partial v(s,t)}{\partial s} \right) + \frac{\partial^2}{\partial s^2} \left(\beta(s) \frac{\partial^2 v(s,t)}{\partial s^2} \right) = -\nabla P(v(s, t)). \quad (1.6)$$

Перші два члени в лівій частині рівняння є інерційною (inertial) і демпфуючою (damping) силою. Інші два члени – внутрішня розтягуюча (stretching) та згинальна (bending) сили. Права частина рівняння виражає зовнішні сили. Рівновага в системі настає при балансі між внутрішніми та зовнішніми силами, коли крива приходить до стану спокою, що еквівалентно умові рівноваги (1.5).

Основним недоліком моделей активного контуру та динамічних ДМ є топологічне обмеження: крива не може мати розривів для опису меж багатозв'язкової області [16, 25].

1.1.6 Імовірнісні деформовані моделі (probabilistic deformable models).

Альтернативний варіант моделей, що деформуються, виникає при розгляді процесу деформації з точки зору моделі Байєса (імовірнісні ДМ) [8]. Такий підхід дозволяє об'єднати характеристики об'єкта, що деформується, з параметрами деформованої моделі, а також оцінити міру невизначеності знайдених параметрів.

Модель Байєса є статистичним описом завдання оцінювання, що складається з двох окремих частин: апріорної моделі і сенсорної моделі. Апріорна модель, $p(u)$, є ймовірнісний опис об'єкта до того, як були отримані будь-які відомості про нього.

Сенсорна модель, $p(d|u)$, є імовірнісним описом стохастичних процесів (шуму), що відносяться до початкового (невідомого) стану об'єкта u для вступного зображення d . Ці дві моделі можуть бути об'єднані для отримання постеріорної моделі, $p(u|d)$, яка є ймовірним описом поточної оцінки u фіксованого зображення d

1.1.7 Активний контур та потік вектора градієнта (active contours and gradient vector flow).

Традиційна модель активного контуру [34] вимагає завдання початкової кривої в безпосередній близькості від меж сегментованого об'єкта. Крім того, така модель не здатна охопити U-подібні ділянки кордону. Ці проблеми були вирішені за допомогою моделі активного контуру Ху [1]. Модель полягає в використанні іншого типу зовнішніх сил, названих потоком вектора градієнта.

Сили обчислюються із узагальненого рівняння дифузії, застосованого до функції градієнта зображення. Саме за рахунок дії цих сил ця модель активного контуру суттєво відрізняється від попередніх методів.

Потік вектора градієнта має тенденцію простягатися за межі об'єкта. Це розширює "смугу захоплення", і активний контур може виявляти об'єкти, які знаходяться далеко від місця розташування початкової кривої. Ці сили здатні затягнути активний контур у увігнуті області. Основна їх особливість на відміну традиційних сил у тому, що вони є чисто безвихровими. Потік вектора градієнта поєднує в собі і безвихрову, та соленоїдальні компоненти. Тому його не можна отримати шляхом мінімізації функціоналу енергії, характерної для традиційної моделі активного контуру.

Деформацію активного контуру можна описати за допомогою динамічного рівняння:

$$\mu(s) \frac{\partial^2 v(s,t)}{\partial t^2} + \gamma(s) \frac{\partial v(s,t)}{\partial t} = F_{\text{внутр}} + F_{\text{зовн}}, \quad (1.7)$$

де $v(s, t) = (x(s, t), y(s, t))$ – параметричне завдання кривої на площині в момент часу t ; $\mu(s)$ і $\gamma(s)$ – параметри, що представляють відповідно щільність мас та щільність згасання моделі,

$F_{\text{внутр}}, F_{\text{зовн}}$ – внутрішня та зовнішня сили.

Рух кривої відбувається відповідно до величини та напрямку внутрішньої та зовнішньої сил:

$$F_{\text{внутр}} = \frac{\partial}{\partial s} \left(\alpha(s) \frac{v(s, t)}{\partial s} \right) - \frac{\partial^2}{\partial s^2} \left(\beta(s) \frac{\partial^2 v(s, t)}{\partial s^2} \right) \quad (1.8)$$

$$F_{\text{внутр}} = -\nabla E_{\text{зовн}}.$$

В роботі [1] Хи запропонував замінити поле зовнішніх сил зображення $\nabla E_{\text{зовн}}$ на полі потоку вектора градієнта $G(x, y)$:

$$\begin{aligned} & \mu(s) \frac{\partial^2 v(s,t)}{\partial t^2} + \gamma(s) \frac{\partial v(s,t)}{\partial t} = \\ & = \frac{\partial}{\partial s} \left(\alpha(s) \frac{\partial v(s,t)}{\partial s} \right) - \frac{\partial^2}{\partial s^2} \left(\beta(s) \frac{\partial^2 v(s,t)}{\partial s^2} \right), \end{aligned} \quad (1.9)$$

де $G(x, y) = (c(x, y), d(x, y))$ є векторне поле, яке мінімізує функціонал енергії E :

$$E = \iint \lambda \{(c_x^2 + c_y^2 + d_x^2 + d_y^2) + |\nabla f|^2 |G - \nabla f|^2\} dx dy, \quad (1.10)$$

де $f(x, y) = |\nabla I(x, y)|$.

За малих значень градієнта $|\nabla f|$ функціонал енергії визначається першим доданком, що є сумою квадратів приватних похідних векторного поля G . Це дає повільно змінюється поле. За більших значень $|\nabla f|$ інтеграл визначається другим доданком та мінімізується шляхом встановлення $G = |\nabla f|$. Таким чином, отримуємо, що векторне поле G приблизно дорівнює градієнту зображення, коли градієнт великий, і прагне повільно змінюється полю в однорідних областях. Параметр λ регулює співвідношення між першим та другим доданком [2, 9].

1.2 Стохастичні методи.

Стохастичні методи ґрунтуються на статистичному аналізі зображень

1.2.1 Порогова класифікація (thresholding approaches)

Порогова класифікація [10] сегментує зображення шляхом розбиття всіх вокселів на 2 групи: група вокселів з інтенсивністю вище за встановлений поріг і група вокселів з інтенсивністю нижче цього порога. Поріг визначається, виходячи з профілю гістограми спектра потужності для даного зображення. Якщо потрібно сегментувати кілька типів тканин, може бути введено кілька порогів (багатопорогова класифікація).

Незважаючи на простоту виконання, метод дуже ефективний при сегментації з високим контрастом між собою. Основним недоліком методу є залежність результатів сегментації від вибраної величини порога. Інший недолік – підвищена чутливість до шуму та неоднорідностей в інтенсивності. Тому гранична класифікація використовується лише як один з етапів (зазвичай перший або останній) у процедурі обробки МРТ-даних [12].

1.2.2 Класифікатори

Класифікатори [15] є методами розпізнавання образів, які шукають поділу простору характеристик, отриманого з обсягу, за допомогою даних з відомими маркерами. Простір характеристик є діапазон N -мірного вектора, складеного з

характеристик у кожному вокселі. Характеристики можуть містити інтенсивність вокселя, градієнт у точці його координати, відстань вокселя від межі об'єкта і так далі. Математично простір характеристик може бути область визначення будь-якої функції.

Класифікатори відносяться до категорії методів, що контролюються, які вимагають попередньо сегментованого навчального набору даних, (сегментація проводиться або вручну, або за допомогою якогось іншого методу сегментації). Далі отримані дані використовуються як система посилань виконання автоматичної сегментації на новому наборі даних.

Одним із найпростіших класифікаторів є метод найближчого сусіда, де кожен воксель відноситься до того класу, що і воксель з близькою інтенсивністю з навчального набору даних. Метод k-найближчих сусідів є узагальненням цього підходу, де кожен воксель відноситься до того класу, що більшість з K-найближчих сусідів. Іншим прикладом простого класифікатора може бути вікно Parzen, де класифікація проводиться у відповідності з більшістю голосів усередині певного вікна простору характеристик, центрованого в немаркованому вокселі. Обидва класифікатори є параметричними, т.к. вони не роблять припущення про статистичну структуру даних.

Іншим загальноприйнятим класифікатором є метод максимальної ймовірності, або класифікатор Байєса. Метод робить припущення, що інтенсивності вокселів є незалежними вибірками із сумарного накладання всіх розподілів ймовірності. Таке накладення, назване кінцевою змішаною моделлю, визначається через функцію щільності ймовірності:

$$f(y_j; \theta, \pi) = \sum_{k=1}^K \pi_k f_k(y_j; \theta_k), \quad (1.11)$$

де y_j – інтенсивність вокселя,

k, f_k – складна функція щільності ймовірності, як параметризована через θ_k , де $\theta = [\theta_1 \dots \theta_K]$

Змінні є змішуваними коефіцієнтами, які враховують внесок від кожної функції щільності. Щоб зібрати навчальні дані, отримують типові вибірки від кожної компоненти змішаної моделі. Кожна θ_k оцінюється з неї. Для гаусових розподілів це k-середні, коваріації та коефіцієнти розподілу. Класифікація нових даних проводиться шляхом віднесення кожного вокселя до класу з найбільшою ймовірністю апостеріорної. У випадку, коли дані дійсно відповідають кінцевому

розподілу гауса, метод максимальної ймовірності дає непогані результати і здатний забезпечити м'яку сегментацію, складену з апостеріорних ймовірностей.

Стандартним класифікаторам необхідно, щоб сегментована структура мала певні кількісні характеристики. Так як навчальні дані можуть бути марковані, класифікатори можуть переносити маркери на новий набір даних доти, поки простір параметрів досить чітко розрізняє кожну мітку. Будучи інтерактивними, класифікатори є відносно обчислювально ефективними і, на відміну від порогової класифікації можуть застосовуватися до багатоканальних даних.

Недоліком класифікаторів і те, що вони виконують будь-якого просторового моделювання. Інший недолік – необхідність у навчальному наборі даних. Такий набір даних може бути отриманий з кожного сегментованого обсягу, але це довго і трудомістко. З іншого боку, використання одного й того ж навчального ряду для великої кількості зображень може призвести до неправильних результатів, які не враховуватимуть анатомічну та фізіологічну мінливість через суб'єкти.

1.2.3 Кластерний аналіз (clustering algorithms)

Сегментація, заснована на методах кластерного аналізу [35], аналогічна методам класифікації, проте, на відміну від них, не вимагає будь-якого навчального набору даних. Методи кластерного аналізу дозволяють групувати об'єкти в окремі кластери, члени яких мають схожі характеристики. Як “об'єкти” виступають вокселі, а як “кластери” – сегментовані області. Подібними характеристиками можуть бути будь-які властивості вокселя, наприклад його інтенсивність, градієнт, колір (у разі кольорового зображення). Для кількісної оцінки подібності використовується поняття метрики (міра відстані між об'єктами [18]).

1.2.3.1 Метод одиночного зв'язку (single-link clustering)

Алгоритм організації кластерів ґрунтується на використанні матриці подібності [36]. Визначаються два найбільш подібні об'єкти. Вони утворюють перший кластер. На наступному кроці вибирається об'єкт, який має найбільшу схожість з одним із об'єктів, що вже включені в кластер. При збігу даних на підставі однакових подібних заходів йтиме освіта відразу декількох кластерів.

1.2.3.2 Метод повних зв'язків (complete-link clustering)

Включення нового об'єкта в кластер відбувається тільки в тому випадку, якщо відстань між об'єктами не менша за певний заданий рівень [7].

1.2.3.3 Метод середнього зв'язку (average-link clustering)

У цьому алгоритмі для вирішення питання про включення нового об'єкта вже існуючий кластер обчислюється середнє значення міри подібності, яке потім порівнюється із заданим пороговим рівнем [37]. Якщо йдеться про об'єднання двох кластерів, то обчислюють відстань між їхніми центрами і порівнюють його із заданим граничним значенням.

1.2.3.4 Метод Уорда (Ward's clustering)

Цей метод [38] передбачає, що у першому кроці кожен кластер складається з одного об'єкта. Спочатку об'єднуються два найближчих кластери. Для них визначаються середні значення кожної ознаки та розраховується сума квадратів відхилень від середнього S . Далі на кожному кроці роботи алгоритму поєднуються ті об'єкти або кластери, які дають найменше збільшення величини S . Метод Уорда призводить до утворення кластерів приблизно рівних розмірів з мінімальною внутрішньокластерною варіацією.

Описані вище методи відносяться до ієрархічних (агломераційних) методів кластерного аналізу, в яких процес об'єднання кластерів відбувається послідовно, на підставі матриці подібності або відстаней. До переваг методів слід віднести їх нечутливість до перетворень вихідних змінних, а також відносну простоту. Основним недоліком є необхідність постійного обчислення та зберігання матриці подібності чи відстаней.

Існує також група про ітеративних методів. Їх сутність у тому, що класифікації починається із завдання деяких початкових умов (кількість утворених кластерів, поріг завершення класифікації тощо.). На відміну від ієрархічних методів, вони більш чутливі до початкової ініціалізації. Тому доцільно спочатку проводити класифікацію по одному з ієрархічних методів, а потім підбирати початкове розбиття для роботи ітераційного алгоритму.

1.2.3.5 Класифікація k-середніх (k-means clustering)

Метод k-середніх [39] належить до групи ітеративних методів. Його алгоритм передбачає використання лише вихідних значень змінних. Для початку класифікації мають бути задані k випадково обраних (із загальної кількості n)

об'єктів, які будуть центрами кластерів. Кожному кластеру надається порядковий номер. На першому кроці з об'єктів, що залишилися $(n-k)$ витягується $k+1$ об'єкт, і на основі обчислення однієї з метрик перевіряється, до якого з центрів він знаходиться найближче. Об'єкт, що перевіряється, приєднується до того центру, якому відповідає найменша метрика. Центр замінюється на новий, перерахований з урахуванням приєднаного об'єкта, і вага його збільшується на одиницю. Якщо зустрічаються два або більше мінімальні відстані, то об'єкт приєднують до центру з найменшим порядковим номером.

Подібна процедура повторюється всім $(n-k)$ точок. Для того щоб досягти стійкості розбиття, використовуючи те ж правило, n точок знову приєднуються до отриманих кластерів. При цьому ваги центрів продовжують накопичуватись. Нове розбиття порівнюється з попереднім.

Якщо вони збігаються, робота алгоритму завершується. Інакше цикл повторюється.

1.2.3.6 Нечітка класифікація (fuzzy clustering)

Як вступні дані алгоритм нечіткої кластеризації [40] використовує набір векторів характеристик: $X = (x_1, x_2, \dots, x_n)$. Кожен вектор показників $x_i \in X$, $i = 1, \dots, n$, має розмірність s : $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$, де x_{ij} , $j = 1, \dots, s$, є j -а характеристика підмножини x_i .

Розглянемо функцію U , визначену на множині X і область значень $E(U) = [0; 1]$. Нехай функція U надає кожному $x_i \in X$ певне значення – ступінь вкладу в розмите множини u . Функція u називається розмитим підмножиною X . Метою алгоритму є розподіл множини X за допомогою розмитих підмножин. Розмите s -розподіл визначається як матриця $s \times n$ така, що:

- 1) Кожен рядок U_i матриці U є розмитим підмножиною множини X .
- 2) Кожен стовпець U_j матриці U показує ступінь вкладу j -ого елемента кожне розмитої підмножини.
- 3) Сума вкладів всіх елементів дорівнює 1.
- 4) Жодна з розмитих підмножин не є порожньою.
- 5) Жодна з розмитих підмножин не містить всі елементи множини X .

Позначимо через M_{fc} розмиті s -поділу множини X . Тоді $U \in M_{fc}$. Розмитий алгоритм “ s ”- середніх використовує ітеративну оптимізацію знаходження мінімуму цільової функції J_m :

$$J_m(U, v) = \sum_{k=1}^n \sum_{i=1}^c (v_{ik})^m (d_{ik})^2, \quad (1.12)$$

де $v = (v_1, v_2, \dots, v_c)$,

v_i - є центр кластера класу $i, 1 \leq i \leq c, d_{ik}^2 = \|x_k - v_i\|^2$.

1.2.3.7 Методи, що використовують теорію графів (clustering using graph theory).

Було запропоновано кілька різних підходів, які використовують теорію графів для кластеризації даних. Розглянемо один із них, запропонований Wu та Leahy у 1993 році. У цьому алгоритмі [41] дані, які необхідно класифікувати, надаються за допомогою неорієнтованого суміжного графа G . Кожна вершина графа G відповідає точці з набору даних. Ребро з'єднує дві вершини графа G , якщо відповідні їм точки з набору даних є сусідніми, згідно з заданим критерієм сусідства. Далі кожному ребру надається деяке значення ємності (flow capacity). Це робиться для того, щоб відобразити міру подібності між парою пов'язаних вершин. Кластеризація досягається шляхом видалення ребер графа G для створення взаємовиключних підграфів. При оптимальному розподілі графа G на підграф K для видалення вибираються такі ребра, розрив яких дасть максимальну втрату ємності. Для цього перебирається низка всіх можливих $K-1$ розривів, що розділяють пари вершин.

Метод мінімізує максимальну втрату ємності при утворенні K -підграф з графа G , а, отже, мінімізує подібність між підграф, які є в цьому випадку кластерами.

Таким чином, завдання алгоритмів кластеризації полягає в групуванні елементів у мінімально можливу кількість кластерів. Це можна сформулювати у термінах суміжного графа G , що складається з елементів (вершин). Розподіл графа G на ряд незв'язаних підграфів здійснюється шляхом видалення ребер, що зв'язують підграфи. При цьому набір вершин у кожному підграфі буде одиничний кластер, а об'єднання вершин – просторову зв'язкову область обсягу. Оскільки ємність ребра є мірою подібності між сусідніми вершинами, то розподіл графа G на два підграфа включатиме мінімізацію максимальної ємності між двома підграфами.

Хоча алгоритми кластеризації не вимагають набору навчальних даних, вони потребують початкової сегментації (або, що еквівалентно, початкових параметрах). Алгоритм мінімізації очікування (The Expectation-Minimization

Algorithm) показав більшу чутливість до ініціалізації, ніж алгоритм k-середніх або нечіткий алгоритм c-середніх. Як і класифікатори, алгоритми кластеризації не включають просторове моделювання і тому можуть бути чутливі до шуму та неоднорідності в інтенсивності. Відсутність просторового моделювання дає, однак, істотну перевагу в плані швидкості обчислень. Були представлені роботи з поліпшення стійкості алгоритмів до неоднорідностей в інтенсивності МРТ-даних. Стійкість до шуму може бути включена з допомогою випадкового поля Маркова.

1.2.4 Випадкове поле Маркова (Markov random fields)

Випадкове поле Маркова [3] є ймовірнісна модель, задану на решітці. Це набір випадкових змінних, де кожна змінна відповідає певному вузлу у цій решітці. За такого завдання можливі статистичні кореляції між вузлом решітки та його сусідами. Ці локальні кореляції забезпечують механізм моделювання властивостей зображення. Так, наприклад, враховується факт, що більшість пікселів/вокселів на зображенні належать до того ж класу, що й їхні сусіди. Ймовірнісна модель, побудована для набору випадкових змінних, визначає їх відповідні можливості для кожної події. Така модель має на увазі статистичну незалежність, при якій загальна ймовірність події, що включає k змінних, обчислюється як сумарний результат індивідуальних ймовірностей.

Решітка може бути представлена як прямокутна матриця, що містить m рядків і n стовпців. Кожен елемент такої матриці відповідає пікселю на зображенні. Грати $m \times n$ записуються за допомогою подвійного або одинарного індексів:

$$S = \{(i, j), 1 \leq i \leq m, 1 \leq j \leq n\}, \text{ або } S = \{i | 1 \leq i \leq m \times n\}. \quad (1.13)$$

Випадкове поле F – це набір випадкових змінних F_i , заданих на гратах: $F = \{F_i | i \in S\}$. Конфігурація f випадкового поля F задається певними значеннями випадкових змінних F_i . Конфігураційний простір F довільного поля F – це набір всіх можливих конфігурацій.

Система N сусідів визначає вузли на гратах, найближчих до цього вузла. Система сусідів першого порядку містить 4 сусіда: $N_{ij} = \{(i - 1, j), (i, j - 1), (i, j + 1), (i + 1, j)\}$ – для внутрішнього пікселя (i, j). Система сусідів другого порядку включає сусідів першого порядку плюс діагональні щодо заданого вузла елементи (8 сусідів).

Випадкове поле F називається випадковим полем Маркова, визначеним на ґратах S щодо системи N сусідів, тоді і лише тоді, коли виконано дві умови:

- 1) $P(f) > 0, \forall f \in \{F\}$.
- 2) $P(f_i | f_S \setminus \{i\}) = P(f_i | f_{N_i})$.

Перша умова виключає нульову ймовірність кожної конфігурації f випадкового поля F , тоді як друга умова передбачає, що у розподіл випадкової змінної, відповідної i -ому вузлу ґрати, можуть проводити лише найближчі N сусідів.

Насправді основною метою застосування випадкового поля Маркова є виявлення найімовірнішої конфігурації f^* випадкового поля $F: f' = \arg \max_f P(f)$.

Моделювання випадкового поля Маркова перестав бути самостійним алгоритмом сегментації. Його, як правило, включають в інші методи (наприклад, алгоритм кластеризації до середніх).

Основні складнощі, пов'язані з використанням даного методу, полягають у правильному виборі параметрів, що контролюють силу зростання взаємодії. Завдання надто сильного зв'язку може призвести до надмірно згладженої сегментації у разі втрати важливих деталей у структурі. Крім того, випадкове поле Маркова потребує обчислювально ємних алгоритмів.

Незважаючи на це, метод широко використовується для моделювання не тільки класів сегментації, а й неоднорідностей в інтенсивності, які часто зустрічаються на МРТ-зображеннях [17].

1.3 Змішані методи

Змішані (або гібридні) методи включають характеристики структурних і стохастичних методів сегментації.

1.3.1 Нарощування області (region growing)

Нарощування області [42] – найпростіший алгоритм серед методів змішаного типу. Він сегментує зображення, виходячи з певного критерію зв'язності. Нарощування області проводиться від обраної всередині об'єкта точки і продовжується до тих пір, поки критерій зв'язності не перевищить встановлене значення. Простою формою критерію зв'язності, чи критерію зупинки, є межі об'єкта. Більш складні форми створюються з використанням складних математичних викладок, що включають інформацію про просторові та структурні характеристики об'єкта.

Метод дуже ефективний при оконтурюванні дрібних і найпростіших структур, коли є точно певний критерій зупинки. Таким критерієм може бути чітка межа або область з однорідною інтенсивністю. Головним недоліком методу є необхідність у виборі початкової точки для кожної області, що сегментується, що виробляється вручну.

Отже, метод може бути повністю автоматизований. Крім того, нарощування області може бути чутливе до шуму та ефекту парціального об'єму, що призводить до небажаних порожнеч або розривів.

Нарощування області не є самостійним алгоритмом сегментації. Метод зазвичай використовується як попередній етап процесу обробки зображення розуміння тривимірних даних. Після чого застосовується вже складніша сегментація.

1.3.2 Метод "розділити та об'єднати" (split and merge)

Метод “розділити та об'єднати” [43] вимагає, щоб усі дані були організовані в пірамідальну ґраткову структуру з областей, де кожна область організована до груп по вісім (для тривимірного випадку). Будь-яка область може бути розбита на вісім частин, і, відповідно, вісім частин можуть бути об'єднані в одну велику область. Як і у випадку алгоритму нарощування області, критерієм для об'єднання може бути все, що завгодно: інтенсивність вокселя (як найпростіший варіант), або деяка умова, що базується на результатах попередньої сегментації. Припустимо, що такою умовою є C . Тоді алгоритм можна записати так:

1) вибрати область R у пірамідальній ґратковій структурі; якщо $C(R) = \text{хибно}$, тоді розбити область на вісім частин; якщо для восьми областей R_1, R_2, \dots, R_8 $C(R_1, R_2, \dots, R_8) = \text{вірно}$, об'єднати їх в одну область; коли залишаться області, які можна об'єднати, зупинитися;

2) якщо всі сусідні області R_i та R_j такі, що $C(R_i \cup R_j) = \text{вірно}$, то об'єднати ці області.

Великою перевагою даного методу перед алгоритмом нарощування області є відсутність необхідності втручання оператора вибору початкової точки. З іншого боку, метод вимагає організації вихідних даних у пірамідальну ґратову структуру, що може бути небажано для великих обсягів даних.

1.3.3 Методи, засновані на атласах (atlas-guided approaches)

Для виконання сегментації ці методи [44] використовують атлас чи модель об'єкта. За допомогою стандартних лінійних перетворень зображення атласу переноситься на об'єкт, що сегментується.

Застосовуються такі методи, головним чином, МРТ-зображень мозку. Їх перевага полягає в тому, що разом з областями, що сегментуються, переносяться і маркери областей.

Основним недоліком методу є можлива анатомічна мінливість структурованих структур. Цю проблему намагалися подолати, використовуючи послідовність лінійних та і не лінійних перетворень. Проте сегментація складних структур досі утруднена. У 1997 році Thompson та Toga [45] запропонували ймовірнісний атлас для моделювання анатомічної мінливості. Але такий підхід потребує додаткового часу та втручання оператора для накопичення даних.

Отже, засновані на атласах методи найбільше підходять для сегментації стійких через популяцію структур.

1.3.4 Штучні нейронні мережі

Алгоритми сегментації, які використовують інформацію про структуру об'єкта, часто потребують експертизи спеціаліста [13]. Методи, засновані на штучних нейронних мережах (ІНП), частково долають цю проблему.

Нейронну мережу в спрощеному вигляді можна розглядати як спосіб моделювання в технічні системи принципів організації та механізмів функціонування головного мозку людини. Висока продуктивність обробки інформації в мозку людини пояснюється паралельною роботою множини щодо повільних нейронів та великою кількістю взаємних зв'язків з-поміж них. За аналогією, ІНС є масштабними паралельними мережами елементів. Кожен елемент може виконувати елементарне обчислення.

Незалежно від способу реалізації, нейронну мережу можна як зважений орієнтований граф. Вузли графа відповідають нейронам, а ребрам – зв'язкам між нейронами. Кожному зв'язку зіставлено вагу, який відображає оцінку збуджуючого або гальмівного сигналу, що передається у зв'язку з цим на вхід нейрона-реципієнта. При конструюванні мережі ваги зв'язків можуть бути призначені апріорно або з часом змінюватися.

Вага можна інтерпретувати як відображення знань, а процес їх налаштування – як процес навчання системи.

Серед основних характеристик ІНС виділяють навчання на прикладах та узагальнення отриманих знань, придушення шумів, стійкість до відмов, оптимальна поведінка пошуку.

Було показано, що нейронні мережі успішно поєднує різні типи апріорної інформації, що міститься у зображеннях біологічних об'єктів. Ці підходи мають підвищену стійкість і чутливість. ІНС можуть застосовуватися в сегментації як методи класифікації та кластеризації, а також використовуватися для деформованих [14, 21, 27].

1.3.5 Модель LEGION

Ця модель [4] заснована на принципі роботи локально збуджуваної глобально пригніченої осциляційної нейронної мережі (locally excitatory globally inhibitory oscillator network (LEGION)).

Теорія коливальних кореляцій має два основні аспекти: синхронізацію всередині групи осциляторів, що представляють той самий об'єкт, і десинхронізацію між групами осциляторів, що представляють різні об'єкти. Відповідно до теорії, модель LEGION включає три елементи:

- 1) модель одиничного осцилятора;
- 2) локальні збудливі зв'язки щодо фазової синхронізації всередині групи осциляторів;
- 3) загальний (глобальний) інгібітор, який отримує на вхід сигнал від усієї мережі і у відповідь генерує зворотний сигнал для проведення десинхронізації між групами осциляторів.

Така напрочуд проста структура нейронної мережі здатна забезпечити елементарний підхід до проблеми сегментації двійкових та чорно-білих зображень.

Функціональною одиницею моделі LEGION є одиничний осцилятор, що складається з збуджуючого елемента x і переважного елемента y . Елемент x посиляє сигнал збудження на y , що y у відповідь генерує зворотний сигнал гальмування. При безперервній подачі зовнішніх стимулів на вхід x подібна "петля відгуку" виробляє осциляції.

Поведінка кожного осцилятора, індексованого в мережі через i визначається наступними рівняннями:

$$\left\{ \begin{array}{l} x_i = 3x_i - x_i^3 + 2 - y_i + \rho + I_i H(p_i + \exp(-\alpha t) - \theta) + S_i \\ y_i = \varepsilon \left(\gamma \left(1 + \operatorname{tg} \left(\frac{x_i}{\beta} \right) \right) - i \right) \\ p_i = \lambda(1 - p_i) H \left[\sum_{k \in N_1(i)} T_{ik} H(x_k - \theta_x) - \theta_p \right] - \mu p_i \\ S_i = \sum_{k \in 2(i)} W_{ik} H(x_k - \theta_x) - W_z H(z - \theta_z) \\ z_i = \varphi(\sigma_\infty - z). \end{array} \right. \quad (1.14)$$

Таким чином, після того, як зроблено ряд сциляцій, блок осциляторів, відповідних одній з основних областей зображення, буде працювати синхронно, в той час як будь-які два блоки осциляторів, що відповідають двом основним областям зображення, будуть десинхронізовані один від одного. Для тих вокселів, що не належать до жодної з основних областей зображення, відповідний осцилятор становить коливання невдовзі після початку роботи мережі.

Симуляція моделі LEGION є дорогою щодо обчислень, оскільки вимагає чисельної інтеграції величезного числа диференціальних рівнянь, що робить цю модель неприйнятною для сегментації великих наборів даних. З іншого боку, порівняно з більшістю структурних методів, ця модель вимагає меншого втручання з боку оператора, а встановлення початкових параметрів може бути повністю автоматизовано. Крім того, модель має стійкість до шуму [5, 7, 9, 20, 21].

Структурні методи використовують інформацію про структуру об'єкта, що сегментується. Основна їх перевага полягає в здатності сегментувати окремих орган або область, що цікавить нас. Точність сегментації залежить від якості даних, що вводяться. Будь-який шум може вплинути на результат сегментації. Структурні методи вимагають високої роздільної здатності вихідних зображень. Низький контраст може викликати труднощі виявлення кордонів. Іншим недоліком і те, що структурні методи неможливо знайти повністю автоматизовані. Необхідне втручання оператора для вибору початкових параметрів.

Стохастичні методи не розглядають інформацію про структуру. Вони засновані на математичному аналізі даних та здатні сегментувати повний набір даних із зображень органів людини. Однак, незважаючи на повну автоматизацію

стохастичних методів, вибір області, що цікавить, може виявитися стомлюючою процедурою і вимагати втручання оператора. Приміром, при класифікації кісток людини всі кістки сегментуються, як одне область. Якщо нас цікавить, скажімо, череп, то доведеться вручну виділяти його, видаляючи інші кістки. Точність і якість сегментації цих алгоритмах залежить від вибору початкових параметрів. З одного боку, правильне завдання початкових параметрів дасть алгоритм стійкий до шуму. З іншого боку, завдання параметрів може бути критичним і призвести до небажаної сегментації.

Перераховані вище властивості роблять стохастичні методи зручнішими при обробці МРТ-даних, ніж структурні методи

2 НЕЙРОННА МЕРЕЖА FAST FCN. АРХІТЕКТУРА ТА ПРИНЦИПИ ФУНЦІОНУВАННЯ

Сучасні підходи до семантичної сегментації зазвичай використовують розширені згортання для вилучення карт функцій з високою роздільною здатністю, що приносить велику складність обчислень та обсяг пам'яті. Щоб замінити розширені згортання, що забирають багато часу та пам'яті, метод FCN пропонує новий модуль передискретизації під назвою Joint Pyramid Upsampling (JPU), формулюючи завдання вилучення карт функцій високої роздільної здатності у спільну проблему підвищення вибірки.

За допомогою запропонованого JPU метод зменшує складність обчислень більш ніж у три рази без втрати продуктивності. Експерименти показують, що JPU перевершує інші модулі підвищення вибірки, які можна включити до багатьох існуючих підходів для зменшення складності обчислень та підвищення продуктивності. Замінюючи розширені згортання запропонованим модулем JPU, метод досягає найсучасніших показників у наборі даних Pascal Context (mIoU 53.13%) та наборі даних ADE20K (кінцевий бал 0.5584) в 3 рази швидше.

2.1 Огляд методу

Семантична сегментація є одним із фундаментальних завдань комп'ютерного зору, з метою присвоєння семантичної мітки кожному пікселю зображення. Сучасні підходи зазвичай використовують повністю згорнуту мережу (FCN) для вирішення цього завдання, досягаючи надзвичайного успіху серед кількох критеріїв сегментації.

Оригінальний FCN запропоновано Long et al, який перетворений із згорткової нейронної мережі (CNN), призначеної для класифікації зображень. Успадкований від дизайну для класифікації зображень, вихідний FCN поступово зменшує дискретизацію вхідного зображення за допомогою згорток кроків та/або шарів просторового об'єднання, що призводить до остаточної карти об'єктів із низькою роздільною здатністю. Хоча остаточною картою ознак кодує багату семантичну інформацію, точна інформація про структуру зображення втрачається, що призводить до неточних прогнозів навколо меж об'єкта. Як показано на рис. 2.1, вихідний FCN зазвичай зменшує дискретизацію вхідного зображення в 5 разів, зменшуючи просторову роздільну здатність остаточної карти об'єктів у 32 рази.

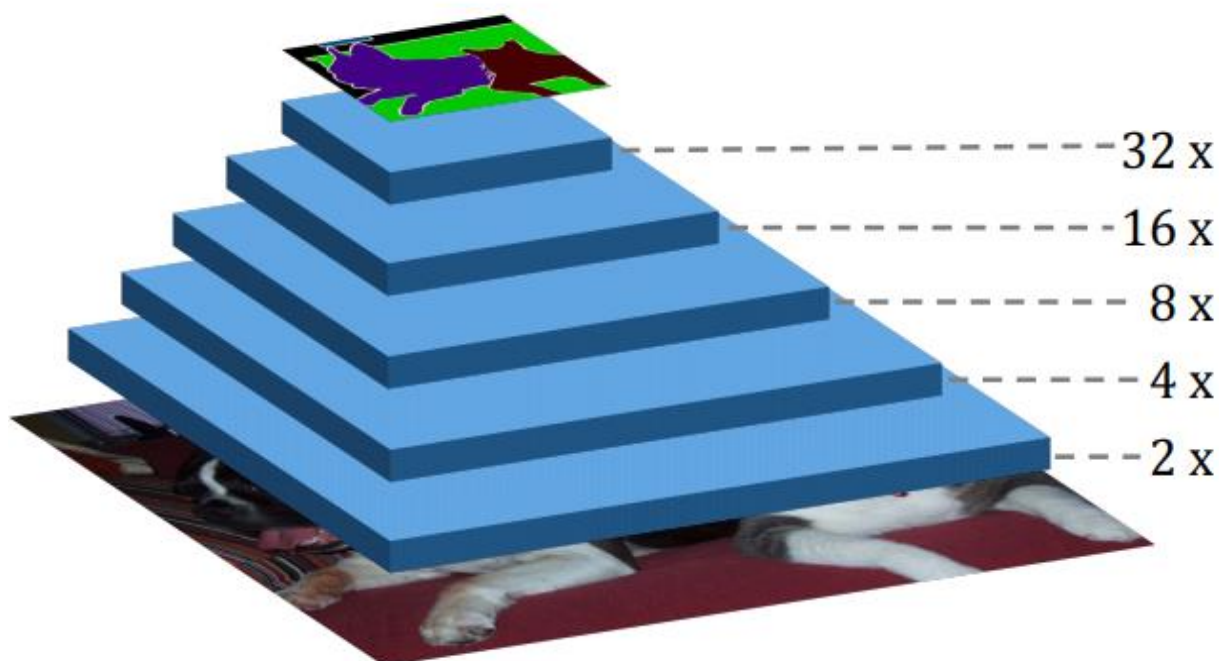


Рис. 2.1. Тип мережі для семантичної сегментації FCN

Щоб отримати остаточну карту об'єктів високої роздільної здатності, використовують оригінальний FCN як кодер для захоплення семантичної інформації високого рівня, а декодер призначений для поступового відновлення просторової інформації шляхом поєднання багаторівневих карт функцій із кодера. Як показано на рис. 2.2 методи EncoderDecoder мають високу роздільну здатність.

Альтернативно, DeepLab видаляє дві останні операції зниження дискретизації з вихідного FCN і вводить розширені згортки, щоб зберегти сприйнятливий поле зору незмінним. Після, DeepLab використовують багатомасштабний контекстний модуль на вершині остаточної карти характеристик, значно перевершуючи більшість методів EncoderDecoder у кількох тестах сегментації. Як показано на рис. 2.3, просторова роздільна здатність останньої карти об'єктів у DilatedFCN в 4 рази більша, ніж у вихідної FCN, таким чином зберігається більше інформації про структуру та розташування.

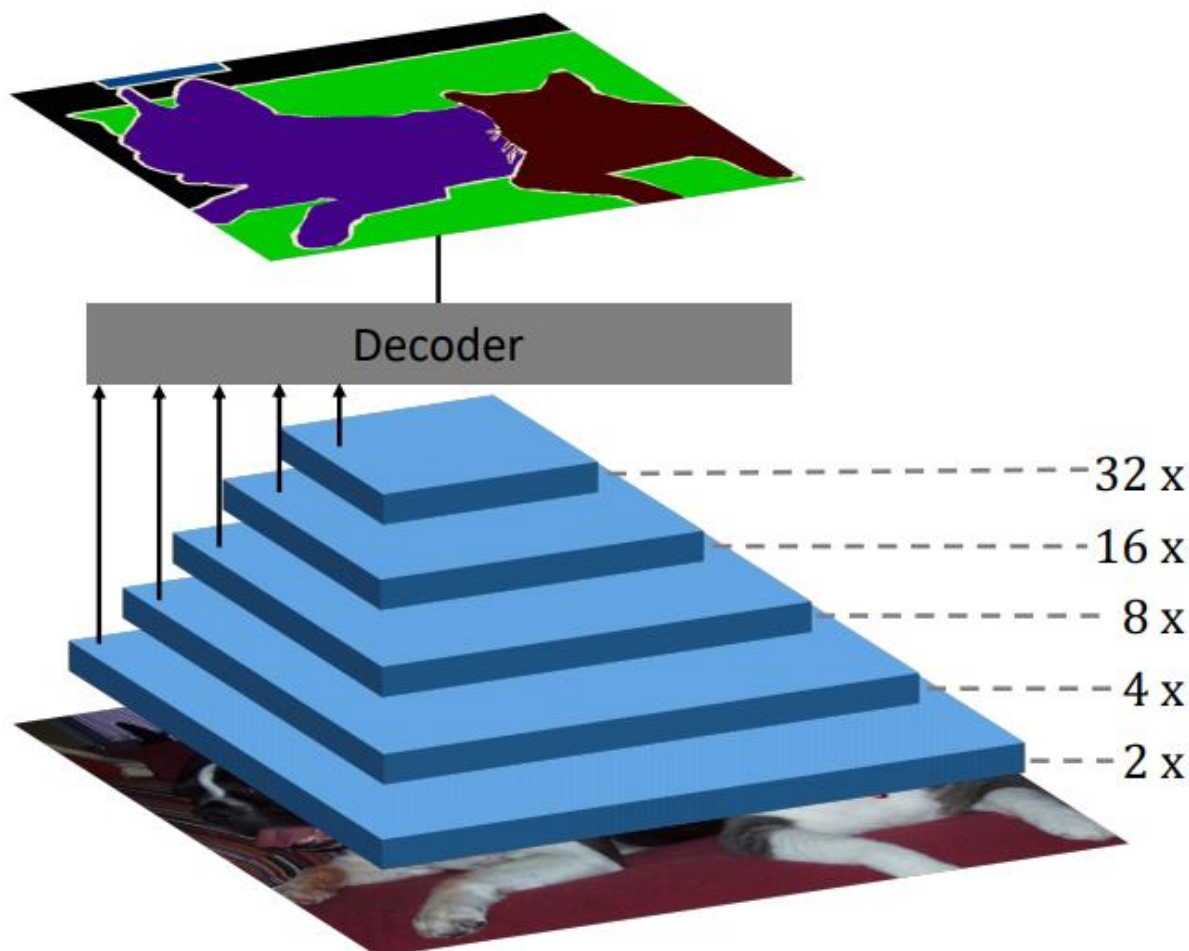


Рис. 2.2. Тип мережі для семантичної сегментації, що відповідає стилю кодер-декодер

Розширені згортки відіграють важливу роль у підтримці просторової роздільної здатності остаточної карти об'єктів, що призводить до кращої продуктивності порівняно з більшістю методів у EncoderDecoder. Проте введені розширені згортки створюють велику складність обчислень і потребує великого об'єм пам'яті, що обмежує використання в багатьох програмах реального часу. Наприклад, у порівнянні з оригінальним FCN ResNet-101, 23 залишкові блоки (69 шарів згортки) у DilatedFCN [47] вимагають у 4 рази більше обчислювальних ресурсів і використання пам'яті, а 3 залишкових блоків (9 шарів згортки) потрібно забирають у 16 разів більше ресурсів [46].

При розробленні даного методу вирішувалася вищезгадану проблему, спричинену розширеними згортками. Щоб досягти цього, запропоновано новий модуль спільного підвищення дискретизації, щоб замінити розширені згортки, що споживають час і пам'ять, а саме — спільне підвищення дискретизації (JPU). В результаті наш метод використовує оригінальний FCN як основу під час

застосування JPU для підвищення вибірки кінцевої карти об'єктів з низькою роздільною здатністю з вихідним кроком (OS) 32, що призводить до карти об'єктів високої роздільної здатності (OS=8). Відповідно, час обчислень і обсяг пам'яті всієї структури сегментації різко скорочуються. Тим часом, при заміні розширених згорток запропонованим JPU не відбувається втрати продуктивності. Це пов'язано зі здатністю JPU використовувати багатомасштабний контекст на багаторівневих картах об'єктів.

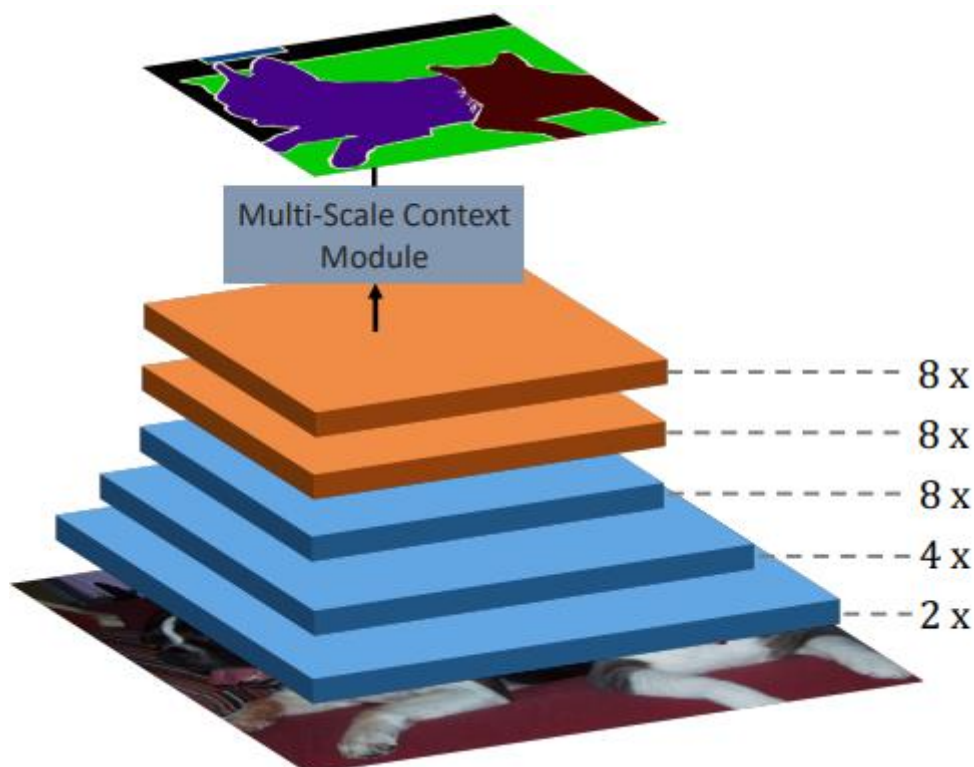


Рис 2.3. Тип мережі для семантичної сегментації з використанням розширених згорток для отримання кінцевих мап функцій з високою роздільною здатністю (DilatedFCN)

Для підтвердження ефективності методу був проведений експеримент, який показав, що запропонований JPU може замінити розширені згортки в кількох популярних підходах без втрати продуктивності. Потім запропонований метод був протестований на кількох тестах сегментації. Результати показують, що метод досягає найсучаснішої продуктивності, працюючи в 3 рази швидше. Конкретно, перевершуючи всі базові показники для набору даних Pascal Context з великим відривом, що досягає найсучаснішої продуктивності з mIoU 53,13%. На наборі даних ADE20K метод отримує mIoU 42,75% з ResNet50, що встановлює новий рекорд у наборі val. Крім того, метод із ResNet-101 досягає найсучаснішої продуктивності в тестовому наборі даних ADE20K.

Таким чином:

- запропоновано обчислювально ефективний модуль спільної передискретизації під назвою JPU, щоб замінити розширені звивини в магістралі, що забирають час і пам'ять;
- виходячи з розподіленого JPU, час обчислення та об'єм пам'яті всієї структури сегментації можна зменшити в 3 рази і таким чином досягти кращих результатів продуктивності.

2.2 Пов'язані роботи

У цьому розділі спочатку приведено огляд методів семантичної сегментації, які можна розділити на два напрямки. Далі буде представлено деякі пов'язані роботи з підвищення дискретизації.

2.2.1 Семантична сегментація

FCN досягли величезних успіхів у семантичній сегментації. Після FCN є два відомі напрямки, а саме DilatedFCN та EncoderDecoder. FCN використовують розширені згортки, щоб зберегти сприйнятливий поле зору та використовують багатомасштабний контекстний модуль для обробки високоякісних карт функцій, EncoderDecoders пропонують використовувати кодер для вилучення багаторівневих карт функцій, які потім об'єднуються в остаточне передбачення декодером.

2.2.2 DilatedFCN

Для того, щоб зафіксувати багатомасштабну контекстну інформацію на кінцевій карті з високою роздільною здатністю, PSPNet виконує операції об'єднання в декількох масштабах сітки, тоді як DeepLabV3 використовує паралельні згінні згорток з різними швидкостями з назвою ASPP. Крім того, EncNet використовує модуль кодування контексту для збору глобальної контекстної інформації. По-іншому, досліджувальний метод пропонує спільний модуль передискретизації під назвою JPU для заміни розширених згорток на магістралі DilatedFCN, що може значно зменшити складність обчислень без втрати продуктивності.

2.2.3 EncoderDecoder.

Для поступового відновлення просторової інформації вводить пропуск з'єднань для побудови UNet, який поєднує функції кодера і відповідні активації

декодера. пропонує мережу уточнення багатопроменевих шляхів, яка явно використовує всю інформацію, доступну в процесі подання вибірки.

DeepLabV3+ поєднує в собі переваги DilatedFCN та EncoderDecoder, де DeepLabV3 використовується як кодер. FastFCN доповнює DeepLabV3+, що може зменшити обчислювальне перевантаження DeepLabV3 без втрати продуктивності.

2.2.4 Підвищення вибірки

У методі FastFCN пропонується модуль для вибірки карти функцій з низькою роздільною здатністю, наданої в якості орієнтації карт функцій з високою роздільною здатністю, який тісно пов'язаний із спільною вибіркою, а також залежністю від даних.

2.2.5 Joint Upsampling

У літературі з обробки зображень спільна вибіркова діагностика спрямована на використання керівного зображення як попереднього та перенесення структурних деталей із керівного зображення на цільове зображення, створює спільний фільтр на основі CNN, який навчається відновлювати деталі структури на настановному зображенні. пропонує наскрізний навчальний керований фільтруючий модуль, який умовно збільшує вибірку зображення з низькою роздільною здатністю. Метод пов'язаний із вищезазначеними підходами. Однак пропонований JPU призначений для обробки карт функцій з великою кількістю каналів, тоді як спеціально розроблені для обробки 3-канальних зображень, які не вловлюють складних зв'язків на високовимірних картах об'єктів. Крім того, мотивація та ціль методу абсолютно різні. Залежне від даних вибіркове випробування DUpsampling також пов'язаний з методом, який використовує переваги надмірності в просторі міток сегментації та здатний відновити піксельне передбачення з виходів CNN з низькою роздільною здатністю. Порівняно з методом FastFCN, DUpsampling має сильну залежність від простору міток, що погано узагальнює більший або складніший простір міток.

2.3 Функціонування FastFCN

У цьому розділі наведено методи семантичної сегментації, названі DilatedFCT, описано реформовану архітектуру DilatedFCN за допомогою нового спільного модуля підвищення дискретизації, Joint Pyramid Upsampling (JPU), та

описано запропонований JPU, перед яким коротко вводиться спільне підвищення дискретизації, розширена згортка та згортка кроків.

2.3.1 DilatedFCN

Щоб використовувати глибокі CNN у семантичній сегментації пропонується [46] перетворюються CNN, призначену для класифікації зображень, у FCN. Взявши ResNet-101 як приклад, оригінальна CNN містить 5 етапів згортки, глобальний середній рівень об'єднання та лінійний шар. Для побудови FCN глобальний середній шар об'єднання та лінійний шар замінюється шаром згортки, який використовується для створення остаточної карти міток, як показано на рис 2.1. Між кожними двома послідовними етапами згортки використовуються згортки кроків та/або шари просторового об'єднання, що призводить до 5 карт об'єктів із поступово зменшеною просторовою роздільною здатністю.

Просторова роздільна здатність останньої карти об'єктів у FCN зменшується в 32 рази, що призводить до неточних прогнозів щодо розташування та деталей. Щоб отримати остаточно карту ознак з високою роздільною здатністю, ДеерЛаб видаляє операції зниження дискретизації перед останніми двома картами ознак, як показано на рис. 2.3. Крім того, шари згортки всередині двох останніх стадій згортки замінюються розширеними згортками для підтримки сприйнятливою поля зору, які називаються DilatedFCN. В результаті роздільна здатність останньої карти об'єктів зменшується в 8 разів, що зберігає більше інформації про місцезнаходження та детальну інформацію. Слідом за ДеерЛаб пропонується багатомасштабний контекстний модуль для захоплення контекстної інформації з останньої карти об'єктів, досягаючи надзвичайного успіху в кількох тестах сегментації.

2.3.2 Основа методу

Щоб отримати остаточно карту об'єктів з високою роздільною здатністю, методи в DilatedFCN видаляють дві останні операції зниження дискретизації з вихідної FCN, що спричиняє значну складність обчислень і об'єм пам'яті через збільшені карти об'єктів. Автори методу прагнули знайти альтернативний спосіб наближення остаточної карти характеристик DilatedFCN без обчислень і перевантаження пам'яті. Між тим, вони очікували, що продуктивність методу буде такою ж високою, як і вихідних DilatedFCN.

Щоб досягти цього, спочатку було повернуто всі згортки кроку, видалені в DilatedFCN, замінивши всі розширені згортки шарами звичайних згорток. Як показано на рис. 2.4, основа методу така ж, як і в оригінальному FCN, де просторові роздільні здатності п'яти карт об'єктів (Conv1–Conv5) поступово зменшуються в 2 рази. Щоб отримати схожу карту об'єктів до остаточної карти характеристик DilatedFCN ми запропоновано новий модуль під назвою Joint Pyramid Upsample (JPU), який приймає останні три карти характеристик (Conv3–Conv5) як вхідні дані.

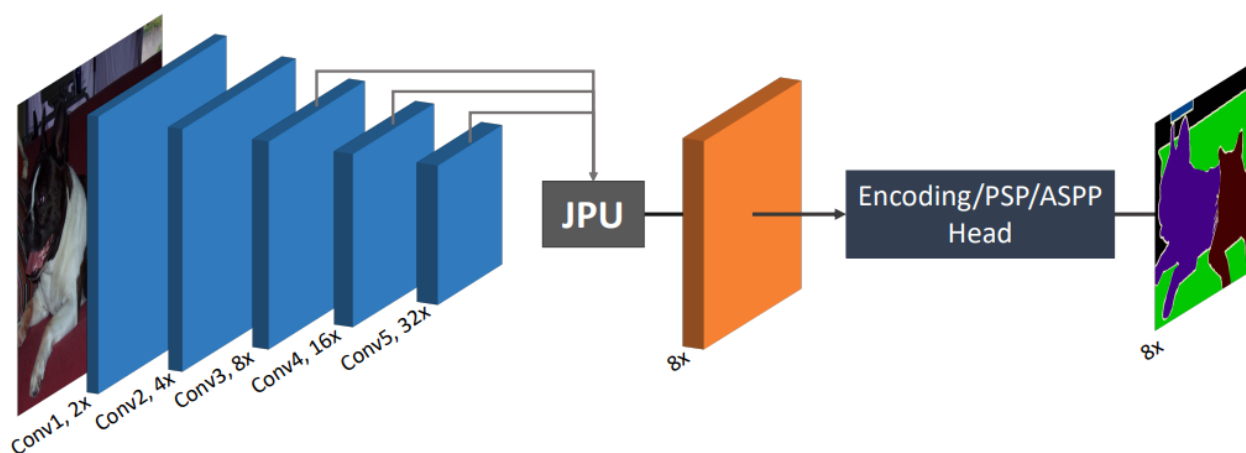


Рис. 2.4. Схематичне зображення роботи досліджувального методу

Потім для створення остаточних прогнозів використовується багатомасштабний контекстний модуль (PSP/ASPP) або глобальний контекстний модуль (coder). У порівнянні з DilatedFCN метод займає в 4 рази менше обчислювальних і пам'ятних ресурсів у 23 залишкових блоках (69 шарів) і в 16 разів менше у 3 блоках (9 шарів) з ResNet-101. Таким чином, метод працює набагато швидше, ніж DilatedFCN, споживаючи менше пам'яті.

2.3.3 Joint Pyramid Upsampling

Запропонований JPU призначений для створення карти об'єктів, яка наближує активації остаточної карти об'єктів з основи DilatedFCN. Подібну проблему можна переформулювати у спільне збільшення вибірки, яке потім вирішується CNN, призначеним для цього завдання.

Враховуючи цільове зображення з низькою роздільною здатністю та орієнтирне зображення з високою роздільною здатністю, спільне підвищення вибірки має на меті створення цільового зображення з високою роздільною здатністю, передаючи деталі та структури з орієнтовного зображення. Як

правило, цільове зображення з низькою роздільною здатністю u_1 генерується шляхом використання перетворення $f(\cdot)$ на орієнтовному зображенні x_1 з низькою роздільною здатністю, тобто $u_1 = f(x_1)$. Враховуючи x_1 та u_1 , нам потрібно отримати перетворення \hat{f} наближеного до $f(\cdot)$, де складність обчислення \hat{f} набагато нижча, ніж $f(\cdot)$. Наприклад, якщо $f(\cdot)$ - багатошаровий перцептрон (MLP), то \hat{f} можна спростити як лінійне перетворення. Потім отримують цільове зображення з високою роздільною здатністю u_h , застосовуючи \hat{f} на керівном у зображенні з високою роздільною здатністю x_h , тобто $u_h = \hat{f}(x_h)$. Формально, враховуючи x_1 , u_1 та x_h , спільна вибірка визначається наступним чином:

$$u_h = \hat{f}(x_h), \text{ де } \hat{f}(\cdot) = \underset{h(\cdot) \in H}{\operatorname{argmin}} \|u_1 - h(x_1)\|, \quad (2.1)$$

де H - сукупність усіх можливих функцій перетворення, а $\|\cdot\|$ є заздалегідь визначеною метрикою відстані. Розширена згортка представлена в DeerLab для отримання карт функцій з високою роздільною здатністю, зберігаючи сприйнятливим поле зору. На рис. 2.5 наведено ілюстрацію розширеної згортки в 1D (швидкість розширення = 2), яку можна розділити на такі три етапи:

- розділити вхідний елемент плавника на дві групи f_{in}^0 і f_{in}^1 відповідно до парності індексу
- обробити кожен характеристику одним і тим же шаром згортки, що призводить до f_{out}^0 і f_{out}^1 ;
- об'єднати дві згенеровані функції, що переплітаються, для отримання вихідної функції f_{out} .

Крок згортання пропонується для перетворення вхідної ознаки у вихідну характеристику зі зменшеною просторовою роздільною здатністю, що еквівалентно наступним двом крокам, як показано на рис. 2.6:

- обробка вхідної ознаки f_{in} звичайною звивиною, щоб отримати проміжну функцію f_m ;
- видалити елементи з непарним індексом, що призведе до f_{out} .

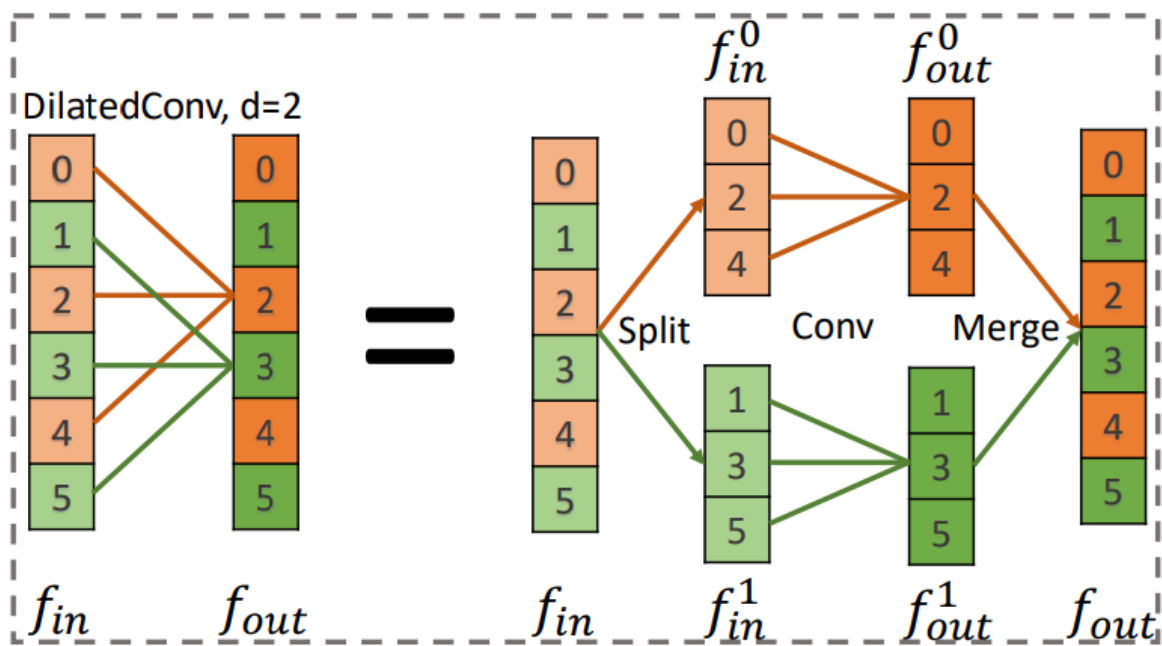


Рис 2.5. Розширена згортка (dilation rate = 2, stride = 2)

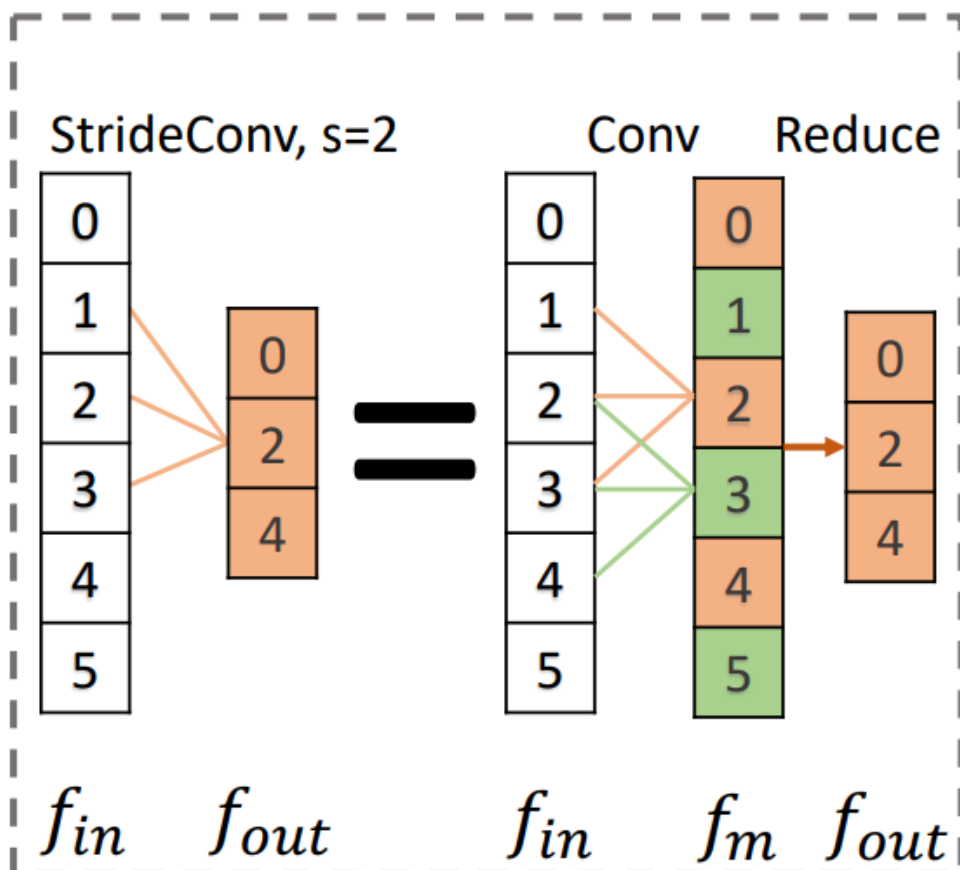


Рис 2.6. Розширена згортка (stride = 2)

2.3.4 Зміна до Joint Upsampling

Відмінності між основою методу FastFcn та DilatedFCN лежать на останніх двох стадіях згортки. На прикладі 4-ї стадії згортки (Conv4) у DilatedFCN вхідна карта функцій спочатку обробляється звичайним шаром згортки, а потім низкою розширених звивин ($d=2$). Інакше, метод FastFCN обробляє карту вхідних ознак з покроковою згорткою ($s=2$), а потім використовує кілька регулярних згорток для формування вихідних даних.

Формально, з огляду на вхідну карту функцій x , вихідну карту функцій y_d у DilatedFCN отримуємо таким чином:

$$\begin{aligned}
 y_d &= x \rightarrow C_r \rightarrow \underbrace{C_d \rightarrow \dots \rightarrow C_d}_n = x \rightarrow C_r \rightarrow \underbrace{SC_rM \rightarrow \dots \rightarrow SC_rM}_n = \\
 &= x \rightarrow C_r \rightarrow S \rightarrow \underbrace{C_r \rightarrow \dots \rightarrow C_r}_n \rightarrow M = \\
 &= y_m \rightarrow S - C_r^n \rightarrow M = \{y_m^0, y_m^1\} \rightarrow C_r^n \rightarrow M.
 \end{aligned} \tag{2.2}$$

В той час як у методі FastFCN вихідна функція y_s генерується наступним чином (рис 2.8).

$$\begin{aligned}
 y_s &= x \rightarrow C_s \rightarrow \underbrace{C_r \rightarrow \dots \rightarrow C_r}_n = \\
 &= x \rightarrow C_r \rightarrow \underbrace{C_r \rightarrow \dots \rightarrow C_r}_n = \\
 &= y_m \rightarrow R \rightarrow C_r^n = y_m^0 \rightarrow C_r^n.
 \end{aligned} \tag{2.3}$$

C_r , C_d та C_s представляють регулярну/розширену/крокову згортку відповідно, а C_{nr} n шарів регулярних звивин.

S і R поділяються, об'єднуються та зменшуються (рис. 3.2, 3.3), де сусідні операції S і M можуть бути скасовані. Примітно, що згортки у рівняннях 2 та 3 знаходяться в 1D, що для простоти. Подібні результати можна отримати для двовимірних згорток. Вищезазначені рівняння показують, що y_s і y_d можна отримати з однією і тією ж функцією C_{nr} з різними входами: y_m^0 і y_m , де перше відбирається від останнього. Таким чином, з огляду на x та y_s , карту об'єктів y , яка наближається до y_d , можна отримати таким чином:

$$y = \{y_m^0, y_m^1\} \rightarrow \hat{h} \rightarrow M, \tag{2.4}$$

$$\text{де } \hat{h} = \underset{h \in H}{\operatorname{argmin}} \|y_s - h(y_m^0)\|,$$

$$y_m = x \rightarrow C_r.$$

Рівняння 2.4 – це задача оптимізації, яка займає багато часу, щоб сходиться через ітеративний градієнтний спуск. В якості альтернативи пропонується наблизити процес оптимізації за допомогою модуля CNN. Для досягнення цього спочатку потрібно сформулювати y_m із заданим x , як показано у форм. 4. Потім, функції з y_m^0 та y_s потрібно зібрати для вивчення відображення \hat{h} . Нарешті, необхідний блок згортки, щоб перетворити зібрані ознаки в остаточне передбачення u . Після вищезазначеного аналізу був розроблений модуль JPU, як на рис. 2.4. По суті, кожна карта вхідних характеристик спочатку обробляється звичайним блоком згортки (рис. 2.7.a), який призначений для генерації y_m із заданим x , та перетворення f_m у вбудовувальний простір зі зменшеними розмірами. В результаті всі вхідні функції відображаються в одному просторі, що забезпечує кращий синтез і зменшує складність обчислень.

Потім згенеровані карти об'єктів підбираються за допомогою вибірки та об'єднуються, що призводить до u_c (рис. 2.7.a). Чотири роздільні згортки з різною швидкістю розширення (1, 2, 4 та 8) використовуються паралельно для вилучення ознак з u_c , де різні швидкості розширення виконують різні функції. Конкретно згортка зі швидкістю розширення 1 використовується для фіксації співвідношення між y_m^0 та іншою частиною y_m , як показано синім полем на рис. 2.8. Альтернативно, згортки зі швидкістю розширення 2, 4 та 8 призначені для вивчення відображення \hat{h} для перетворення y_m^0 в y_s , як показано зеленим кольором. Таким чином, JPU може витягувати багатомасштабну контекстну інформацію з багаторівневих карт функцій, що призводить до кращої продуктивності. Це суттєво відрізняється від ASPP, який використовує лише інформацію на останній карті об'єктів.

Видобуті функції кодують відображення між y_m^0 та y_s , а також відношення між y_m^0 та рештою частиною y_m . Таким чином, використовується інший регулярний блок згортки, який перетворює ознаки на остаточні прогнози (рис. 2.7.c). Примітно, що запропонований модуль JPU спільно вирішує дві тісно пов'язані, які підвищують вибірку Conv4 на основі Conv3 (4-та стадія згортки) та підвищують Conv5 під керівництвом розширеної Conv4 (5-й етап згортання).

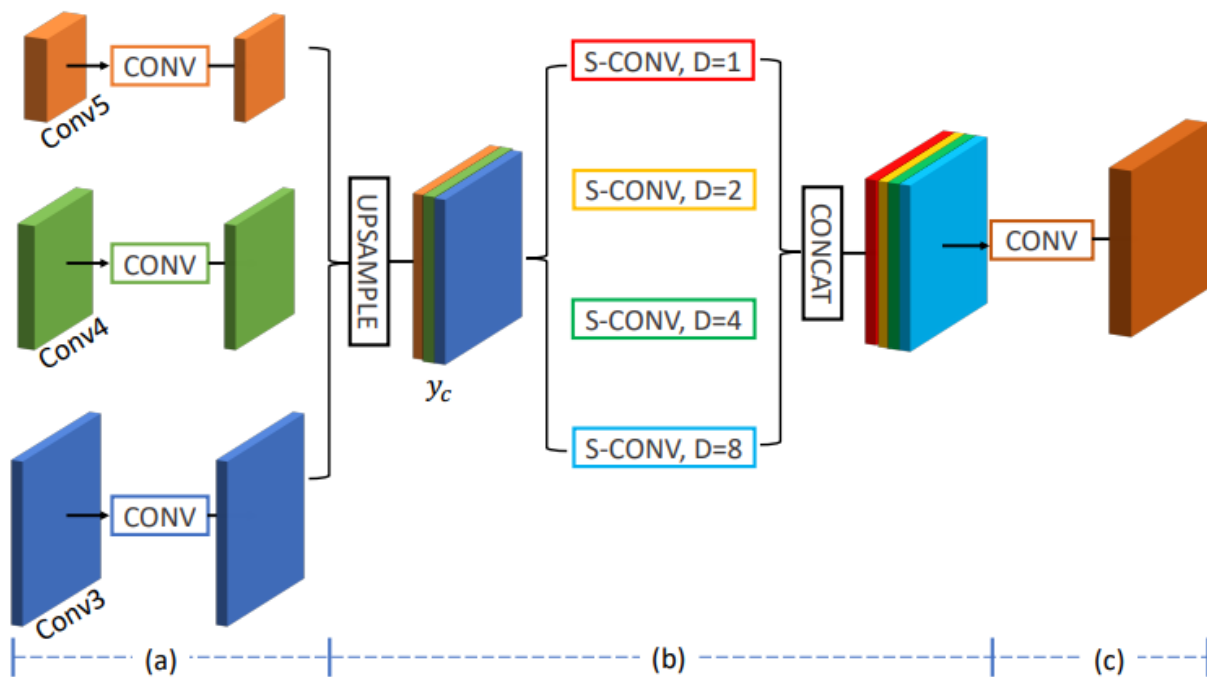


Рис. 2.7. Запропонований Joint Pyramid Upsampling (JPU)

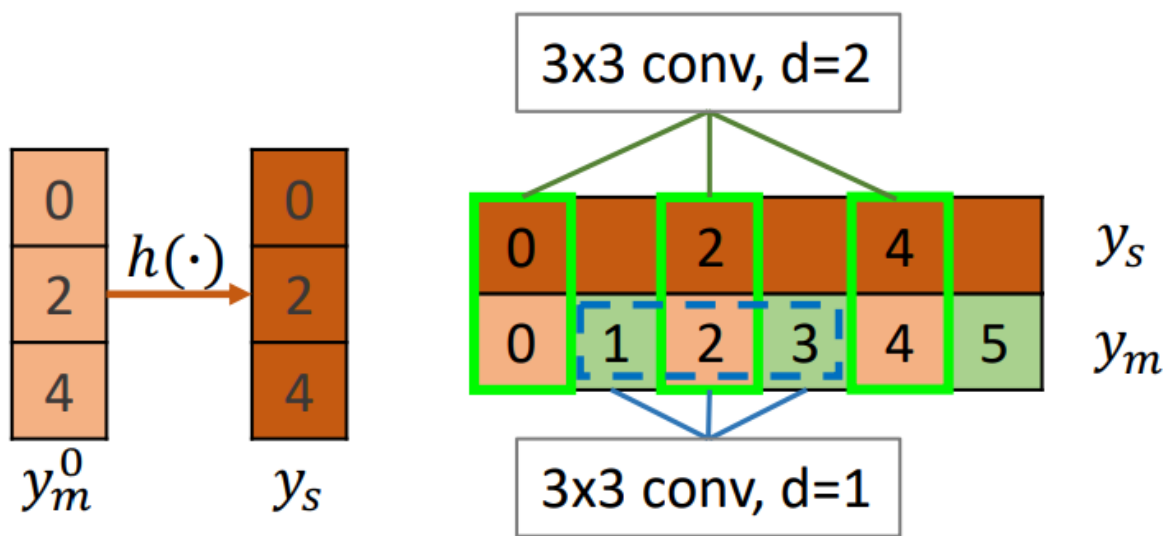


Рис 2.8. Розширена згортка зі швидкістю розширення 1 яка націлена на y_m^0 та решті частини y_m , а згортка зі швидкістю розширення 2 націлена на y_m^0 та y_s

3. ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

Для проведення експериментального випробовування було обрано фреймворк для машинного навчання PyTorch, що в своїй реалізації базується на TensorFlow.

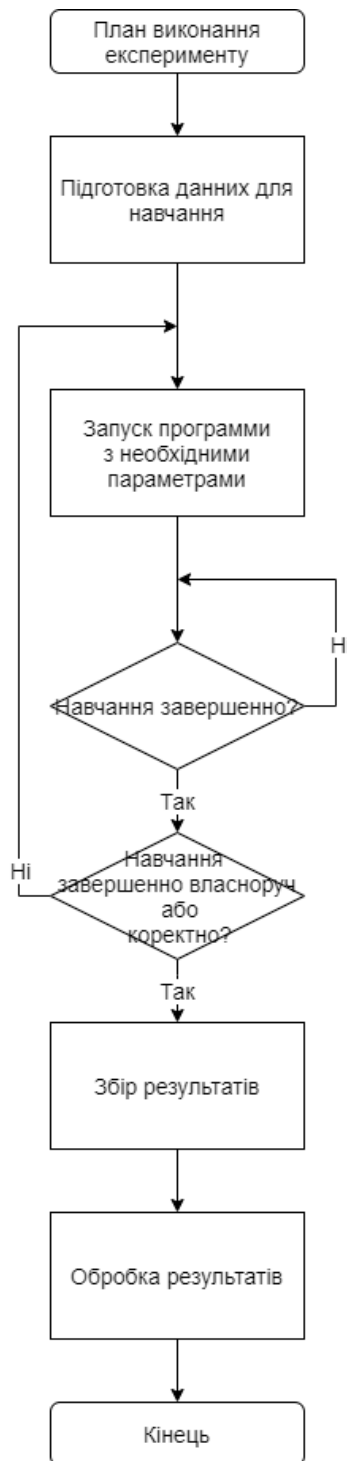


Рис. 3.1. Схема алгоритму проведення експериментів

3.1 Налаштування середовища експерименту.

Експеримент проводився на трьох комп'ютерах з різними конфігураціями. Конфігурації кожного комп'ютера приведені нижче:

Перший комп'ютер:

- процесор: Intel Core i5-7500 CPU @ 3.40GHz 3.41GHz. Кількість ядер – 4, без HyperThreading;
- встановлена пам'ять: 8GB DDR4;
- накопичувач: 1TB;
- операційна система: Windows 10.

Другий комп'ютер:

- процесор: Intel Core i5-6200 CPU @ 2.30GHz 2.4GHz. Кількість ядер – 4, без HyperThreading;
- встановлена пам'ять: 32GB DDR3 @ 1600 MHz;
- накопичувач: 1TB;
- операційна система: Windows 10 Pro.

Третій комп'ютер:

- процесор: Intel Core i5-7400 CPU @ 3.00GHz 2.300Hz. Кількість ядер – 4, без HyperThreading;
- встановлена пам'ять: 32GB DDR3 @ 2400MHz;
- накопичувач: 1TB;
- операційна система: Windows 10 Pro

На кожен із комп'ютерів було встановлено програмне забезпечення у вигляді інтерпретатора мови Python. Програма для проведення експериментів потребує ряд залежностей, які були встановлені завдяки пакетному менеджеру Pip. Виконання програми виконувалося одразу у встановленій операційній системі без використання віртуальних машин, засобів контейнеризації (Docker, docker-compose).

3.2 Підготовка даних до навчання.

Для проведення експерименту в якості даних для навчання та перевірки було вирішено обрати набір даних BraTS (Brain Tumor Segmentation) [29, 30, 31]. Вибір саме цього набору можна обґрунтувати тим що BraTS є одним із відомих датасетів у світі для проведення задач сегментації зображень в галузі магнітно-резонансної томографії головного мозку. Також причиною вибору саме цього набору, а не якогось іншого, полягає в тому що у відкритому доступ інших наборів даних не так багато, а ті що існують мають невелику кількість зображень,

або потребують достатньої кількості перетворень для їх подальшого використання у розробленій програмі.

Для отримання доступу до набору BraTS необхідно було отримати дозвіл від Центру біомедичних обчислень та аналітики оброки зображень, що базується у Медичній школі Перельмана у Пенсільванському університеті. Для початку необхідно було пройти реєстрацію на інформаційному порталі центру, яка теж потребувала підтвердження зі сторони адміністрації порталу. Після підтвердження реєстрації необхідно було подати запит на отримання доступу до конкретного набору даних. На час подання запитів були доступні датасети з 2018 по 2020 роки. Для проведення експерименту було обрано набір даних за 2020 рік, на який було подано запит у адміністрації. Через деякий час на електронну пошту надійшов лист з посиланням до архіву з набором даних.

Наданий набір даних BraTS 2020 містить 369 файлів у форматі NIfTI.

Інформаційна технологія нейровізуалізації (NIfTI) – це відкритий формат файлу, який зазвичай використовується для зберігання даних візуалізації мозку, отриманих за допомогою методів магнітно-резонансної томографії.

У форматі NIfTI перші три виміри зарезервовані для визначення трьох просторових вимірів — x , y та z , а четвертий вимір зарезервовано для визначення часових точок — t . Решта розмірів, від п'ятого до сьомого, призначені для інших цілей. П'ятий вимір, однак, все ще може мати деякі заздалегідь визначені види використання, наприклад, для зберігання специфічних для вокселів параметрів розподілу або для зберігання даних на основі векторів.

Кожен із наборів даних BraTS містить файли MPT декількох типів.

MPT головного мозку з контрастом (режим Flair) – це технологія магнітно-резонансної томографії, яка центрально візуалізує всю структуру і тканину нервової системи. Порядком сканування вдається виявити різні структурні патології ЦНС: доброякісні та злоякісні пухлини мозку, осередки зміни при розсіяному склерозі. У найкращих клініках Турції, Германії, Ізраїлю проводиться MPT-сканування головного мозку з контрастом у режимі Flair. Цей режим діагностики використовується переважно для виявлення осередків розсіяного склерозу. На рис. 3.2 зображена візуалізація MPT в режимі Flair.

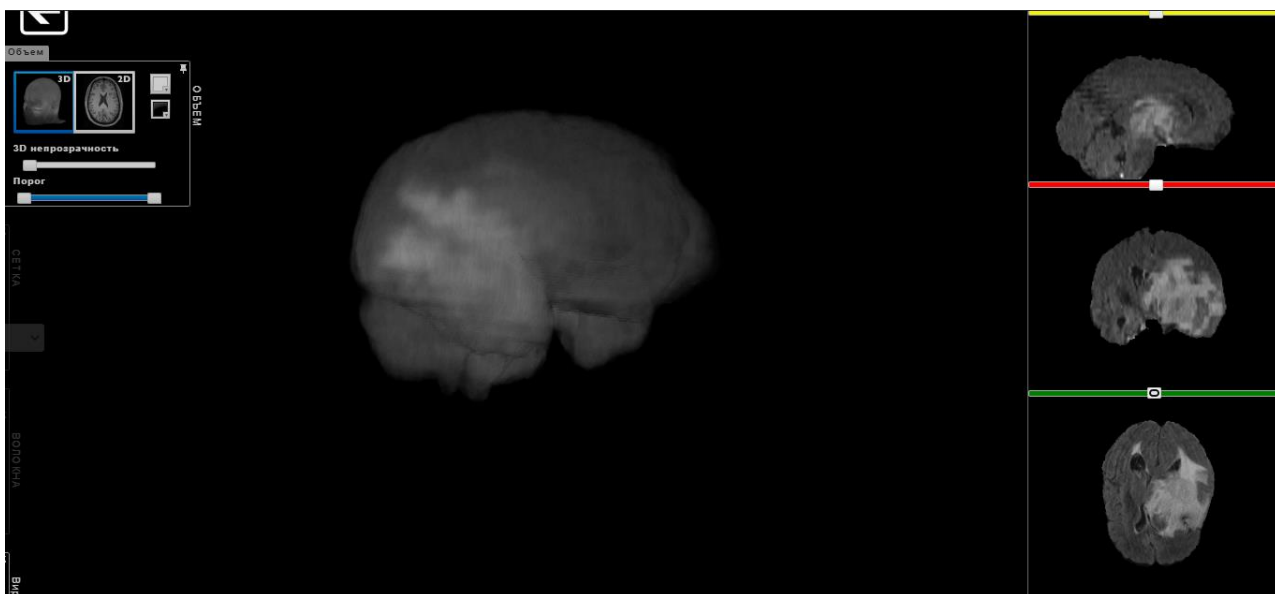


Рис. 3.2. BraTS'20: набір Flair

T1-зважене зображення є основною послідовністю імпульсів у магнітно-резонансній томографії і відображає відмінності в сигналі на основі внутрішнього часу релаксації T1 різних тканин. Клінічно T1-зважені зображення, як правило, краще відображають нормальну анатомію і можуть підкреслити патологію, якщо надано контраст гадолінієм. Наприклад, жир демонструє високу інтенсивність сигналу на T1-зважених зображеннях, тоді як рідина демонструє низьку інтенсивність сигналу. МРТ головного мозку зображено на рис. 3.3.

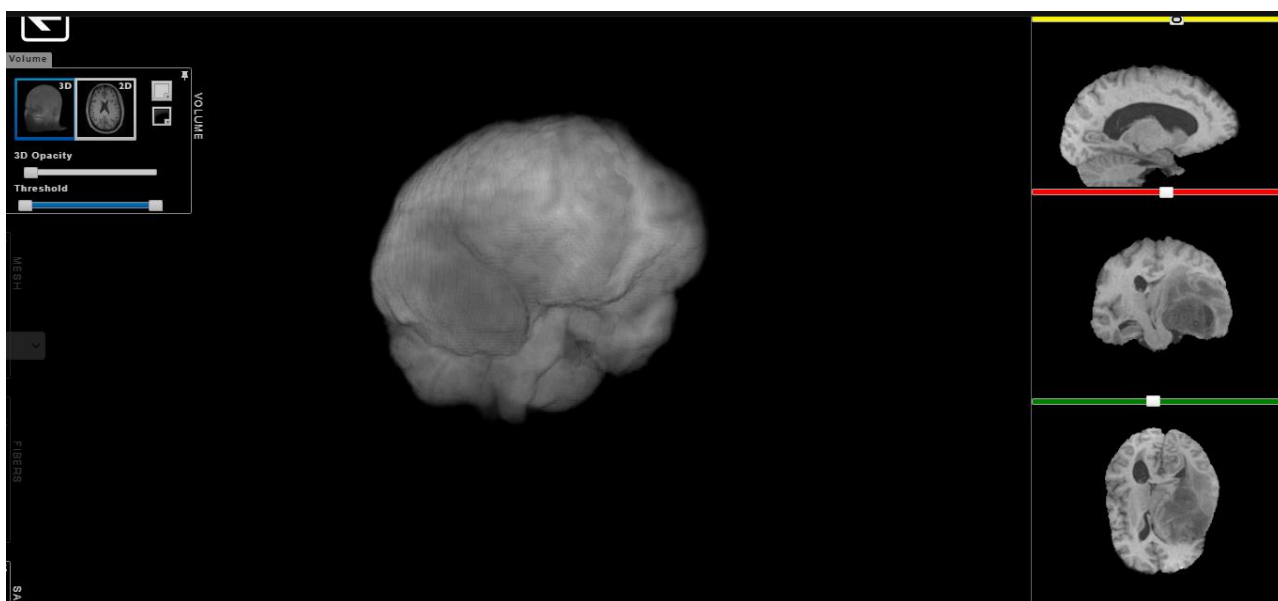


Рис. 3.3. BraTS'20: набір T1

T2-зважене зображення — це базова імпульсна послідовність при магнітно-резонансній томографії, яка відображає відмінності в часі релаксації T2 різних тканин. Клінічно T2-зважені зображення дають найкраще зображення захворювання, оскільки більшість тканин, які беруть участь у патологічному процесі, мають вищий вміст води, ніж зазвичай, а рідина (наприклад, спинномозкова рідина, скловидне тіло) призводить до того, що уражені ділянки виглядають яскравими на T2-зваженому зображенні. На рис. 3.4 зображене МРТ в даному режимі.

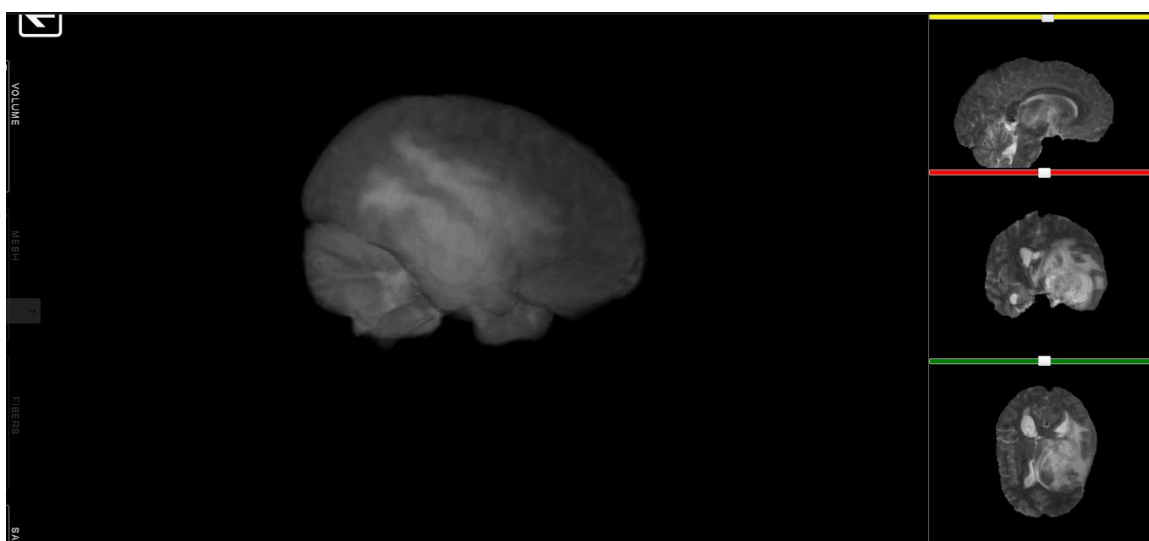


Рис. 3.4. BraTS'20: набір T2

В наборі представлені данні у режимі T1CE. Візуалізація цього режиму представлена на рис. 3.5.

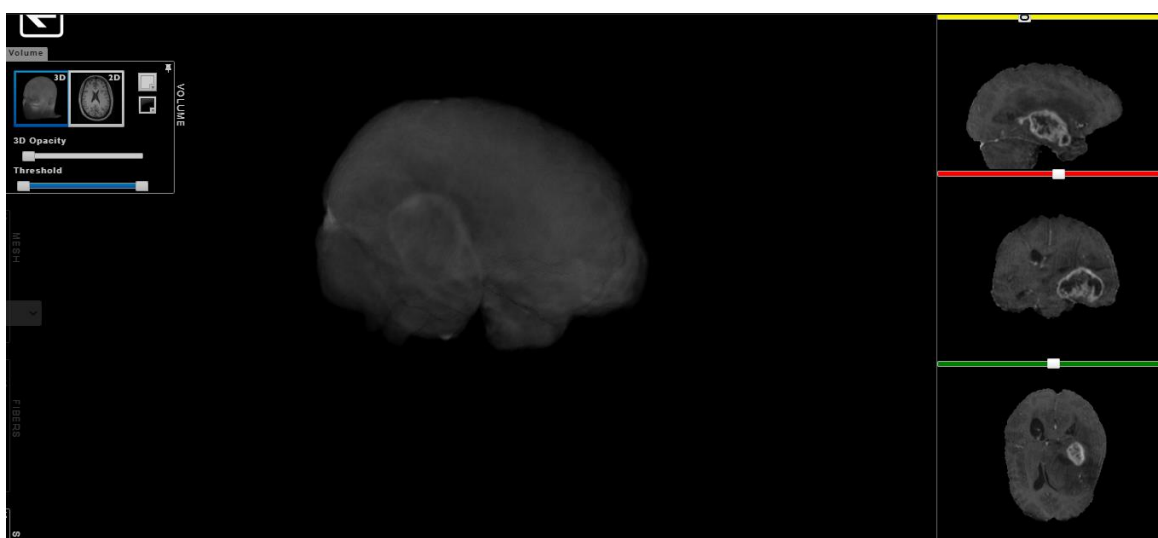


Рис. 3.5. BraTS'20: набір T1CE

Усі набори даних візуалізації були сегментовані вручну (тип Seg) від одного до чотирьох оцінювачів, дотримуючись одного й того самого протоколу анотації, і їхні анотації були схвалені досвідченими нейрорадіологами. Анотації включають пухлину, перитуморальний набряк та некротичне та не збільшу ядро пухлини.

Надані дані розподілялися після їх попередньої обробки, тобто спільно реєструвалися в одному анатомічному шаблоні, інтерполювалися з однаковою роздільною здатністю (1 мм³) і очищені від черепа.

Сегментоване МРТ зображено на рис. 3.6.

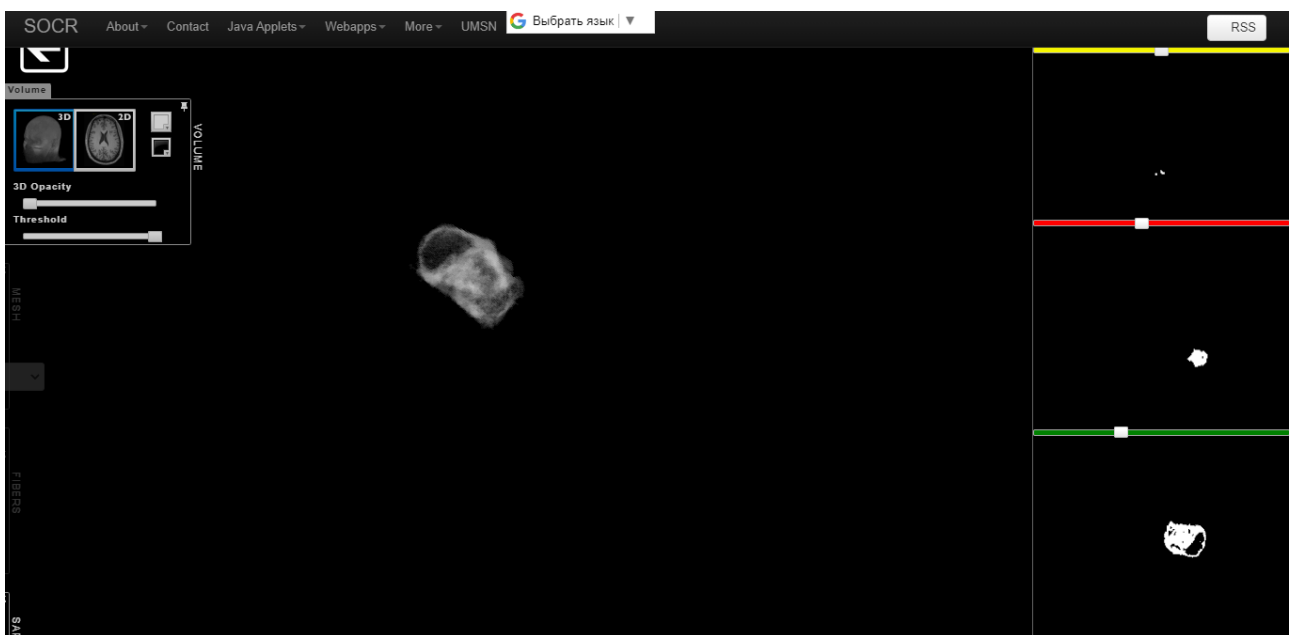


Рис. 3.6. BraTS'20: набір Seg – сегментовані дані

Так як розроблена програма для проведення експериментів розрахована для обробки зображень у форматі JPG та PNG, то набори даних NIfTi необхідно було конвертувати до цих форматів. Вивчивши структуру формату NIfTu, візуалізація якого приведена на рис. 3.7 у спрощеному вигляді. Для цього було розроблено скрипт (див. додаток А), який розкладає формат NIfTu на файли PNG.

Кожен файл у форматі NIfTu було конвертовано у 155 файл у форматі PNG. Приклад візуалізації NIfTu файлу у форматі PNG представлено на рис. 3.8. Таким чином весь набір даних BraTS 2020 має 57195 файлів на кожен формат (Flair, Seg, T1, t2, T1ce) розміром 240x240 пікселів.

Додатково файли для навчання було конвертовано до формату JPG. Файли з набору Seg залишено без змін у форматі PNG.



	Slice code: 1 2 3 4 5 6						
	File slice #12	(Not MRI acquired/Padded slices)					
	File slice #11	10th	1st	10th	1st	5th	6th
	File slice #10	9th	2nd	5th	6th	10th	1st
	File slice #09	8th	3rd	9th	2nd	4th	7th
	File slice #08	7th	4th	4th	7th	9th	2nd
	File slice #07	6th	5th	8th	3rd	3rd	8th
	File slice #06	5th	6th	3rd	8th	8th	3rd
	File slice #05	4th	7th	7th	4th	2nd	9th
	File slice #04	3rd	8th	2nd	9th	7th	4th
	File slice #03	2nd	9th	6th	5th	1st	10th
	File slice #02	1st	10th	1st	10th	6th	5th
	File slice #01	(Not MRI acquired/Padded slices)					

Рис. 3.7. Спрощенный пример формата NIfTy

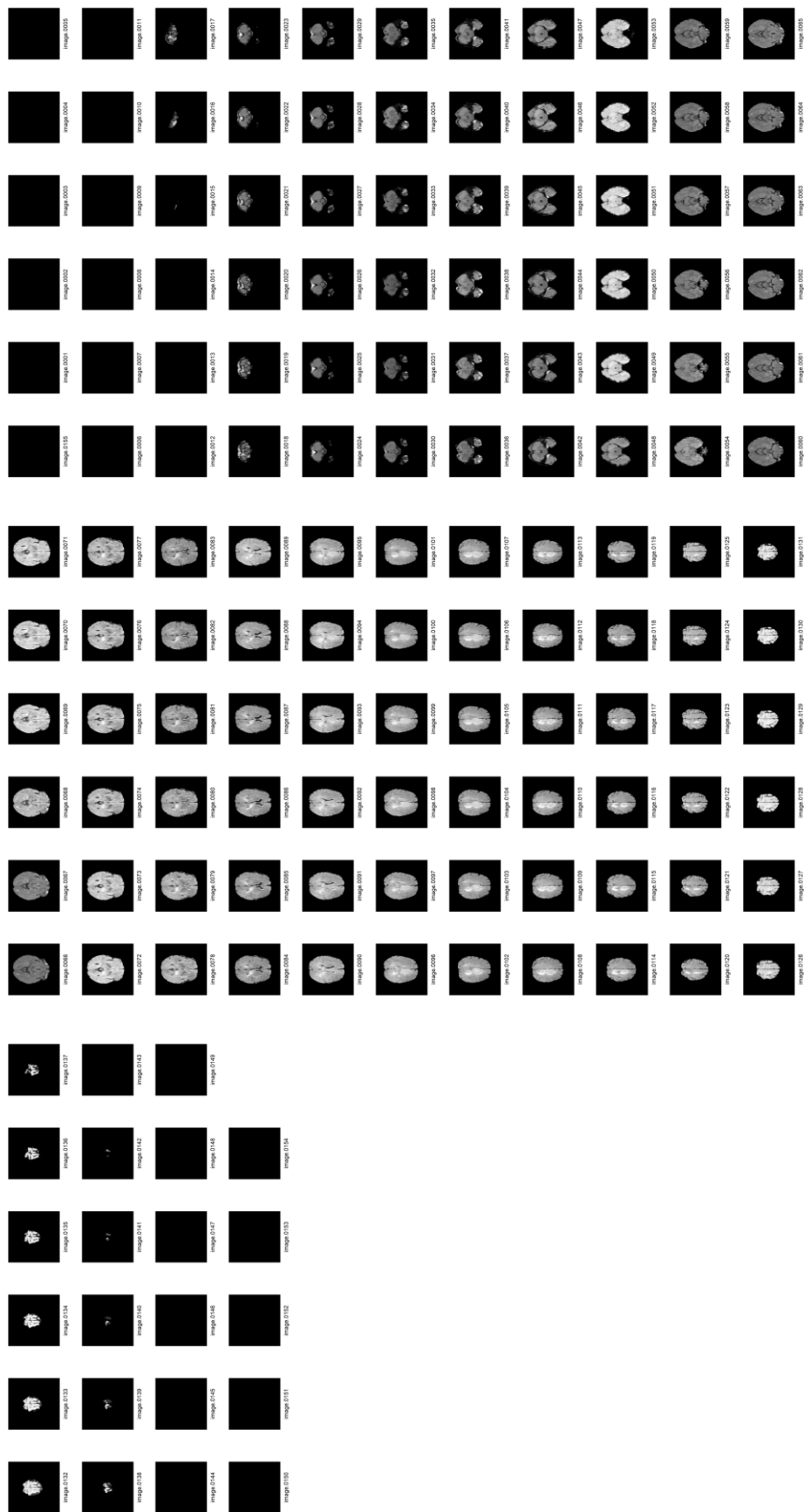


Рис. 3.8 Приклад конвертації NIfTy файлу до файлів у форматі PNG

3.3 Організація та проведення експериментів

Експерименти проводилися на CPU через те відсутність GPU на двох із робочих станцій та не сумісність встановленої GPU на третій наявній робочій станції з технологією CUDA, за допомогою якої можна було б прискорити час навчання.

Експерименти можна проводити з деякими параметрами за замовчуванням, або задати їх значення якщо це необхідно. Програма (див. додаток Б) реалізована таким чином що при необхідності можна реалізувати будь-яку нейронну мережу і на її основі виконати експеримент на будь-якому наборі даних. В програмі для поставленої задачі передбачена реалізація нейронної мережі Fast FCN та робота с датасетом BraTS. Тому відповідна програма для проведення експериментів виконувалася із заданням параметрів моделі та датасету. Також для отримання як найбільшої кількості епох параметр їх кількості був становлений на 99999. В якості кістяка для реалізованої моделі була обрана Res-Net50 – згорткову нейронну мережу, що має глибину в 50 шарів. Кістяк теж можна брати інший, заздалегідь завантаживши її до проекту.

Так як під час проведення експерименту можливі непередбачувані ситуації, через які можливе переривання навчання, то на випадок їх виникнення передбачено збереження останньої навченої моделі та механізм продовження навчання із вказанням файлу з якого треба продовжити навчання та навіть із вказанням початку навчання по номеру епохи.

Для проведення експериментів із раніше описаного набору даних BraTS було обрано зображення із виду MPT Flair. Так як кожен із видів MPT в BraTS представлений зображеннями в кількості 57195 зображень то передбачуваний час навчання декілька годин на одну епоху, то вирішено було проводити навчання лише на частині обраного набору даних, яка складає 10000 зображень для навчання та 5000 зображень для перевірки.

Для розглянутого експерименту, спочатку встановлено швидкість навчання 0.001, яка поступово зменшується до 0, дотримуючись стратегії "poly" (потужність = 0.9). Для збільшення даних випадковим чином масштабується (від 0.5 до 2.0) і перевертаються вхідні зображення вліво. Потім зображення обрізаються до розміру 120×120 і групуються за розміром партії 16.

Мережа тренується протягом з SGD (стахостичний градієнтний спуск), яких імпульс встановлений на 0.9, а спад ваги встановлений на 1e-4. Використовувалася піксельна перехресна ентропія як функція втрат.

3.4 Збір результатів.

Після звершення або зупинення навчання для кожного із проведених запусків сформовані файли із зібраними параметрами на основі яких була проведена оцінка ефективності сегментації обраної нейронної мережі. Також протягом усього часу навчання формувалася модель із найкращим рівнем ефективності.

3.5 Обробка результатів.

Набір даних під час навчання нейронної мережі Fast FCN являє собою набір значень, вказані в табл. 3.1.

Таблиця 3.1

Параметр	Опис
Функція втрат (train loss)	Показник коректності навчання. У поточній реалізації показником є двомірна перехресна втрата ентропії з допоміжними втратами (модифікація стандартної функції перехресної ентропії із фрейморку PyTorch). Приклад розрахунку показника під час навчання представлено на рис. 3.10
Pixel accuracy	Це відсоток пікселів у зображенні, які були правильно класифіковані. точність пікселів зазвичай повідомляється для кожного класу окремо, а також глобально для всіх класів). Формула розрахунку приведена в формулі (3.1)

Закінчення Табл. 3.1

Параметр	Опис
Intersection-Over-Union (IoU)	Це область перекриття між прогнозованою сегментацією та основною істиною, поділена на площу об'єднання між передбачуваною сегментацією та основною істиною. Формула розрахунку приведена в формулі (3.2). Візуалізація показника представлена на рис.3.9.

$$pixAcc = \frac{TP + TN}{TP + TN + FP + FN'} \quad (3.1)$$

де TP – істинно позитивний піксель правильно класифікований як X,
 FP – хибнопозитивний піксель неправильно класифікований як X,
 TN – істинно негативний піксель правильно класифікований як не X,
 FN – хибнонегативний піксель неправильно класифікований як не X.

$$IoU = \frac{\text{Intersection}}{\text{Union}} = \frac{TP}{TP + TN + FN} \quad (3.2)$$

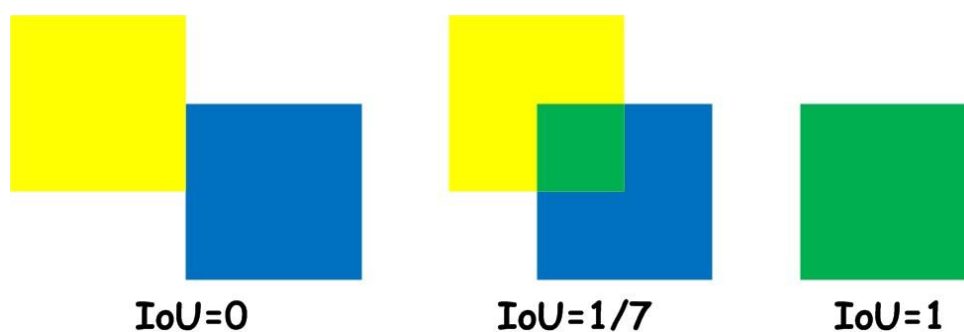


Рис. 3.9. Візуалізація показника IoU

```

Train loss: 1.577: 77%|-----? | 119/155 [17:28<05:15, 8.76s/it]torch.Size([100, 100])
Train loss: 1.580: 77%|-----? | 120/155 [17:37<05:06, 8.75s/it]torch.Size([100, 100])
Train loss: 1.581: 78%|-----? | 121/155 [17:46<05:01, 8.86s/it]torch.Size([100, 100])
Train loss: 1.577: 79%|-----? | 122/155 [17:55<04:50, 8.80s/it]torch.Size([100, 100])
Train loss: 1.581: 79%|-----? | 123/155 [18:04<04:41, 8.80s/it]torch.Size([100, 100])
Train loss: 1.585: 80%|-----? | 124/155 [18:12<04:31, 8.75s/it]torch.Size([100, 100])
Train loss: 1.582: 81%|-----? | 125/155 [18:21<04:22, 8.76s/it]torch.Size([100, 100])
Train loss: 1.584: 81%|-----? | 126/155 [18:31<04:19, 8.95s/it]torch.Size([100, 100])
Train loss: 1.583: 82%|-----? | 127/155 [18:39<04:07, 8.84s/it]torch.Size([100, 100])
Train loss: 1.581: 83%|-----? | 128/155 [18:48<03:58, 8.83s/it]torch.Size([100, 100])
Train loss: 1.579: 83%|-----? | 129/155 [18:57<03:50, 8.86s/it]torch.Size([100, 100])
Train loss: 1.580: 84%|-----? | 130/155 [19:06<03:41, 8.86s/it]torch.Size([100, 100])
Train loss: 1.582: 85%|-----? | 131/155 [19:14<03:30, 8.78s/it]torch.Size([100, 100])
Train loss: 1.583: 85%|-----? | 132/155 [19:23<03:23, 8.85s/it]torch.Size([100, 100])
Train loss: 1.585: 86%|-----? | 133/155 [19:32<03:14, 8.82s/it]torch.Size([100, 100])
Train loss: 1.585: 86%|-----? | 134/155 [19:41<03:04, 8.79s/it]torch.Size([100, 100])
Train loss: 1.583: 87%|-----? | 135/155 [19:50<02:57, 8.86s/it]torch.Size([100, 100])
Train loss: 1.584: 88%|-----? | 136/155 [19:59<02:47, 8.80s/it]torch.Size([100, 100])
Train loss: 1.585: 88%|-----? | 137/155 [20:08<02:39, 8.88s/it]torch.Size([100, 100])
Train loss: 1.586: 89%|-----? | 138/155 [20:16<02:30, 8.83s/it]torch.Size([100, 100])
Train loss: 1.587: 90%|-----? | 139/155 [20:25<02:21, 8.83s/it]torch.Size([100, 100])
Train loss: 1.586: 90%|-----? | 140/155 [20:34<02:12, 8.83s/it]torch.Size([100, 100])
Train loss: 1.585: 91%|-----? | 141/155 [20:43<02:03, 8.80s/it]torch.Size([100, 100])
Train loss: 1.586: 92%|-----? | 142/155 [20:51<01:54, 8.79s/it]torch.Size([100, 100])
Train loss: 1.585: 92%|-----? | 143/155 [21:00<01:46, 8.85s/it]torch.Size([100, 100])
Train loss: 1.584: 93%|-----? | 144/155 [21:09<01:37, 8.83s/it]torch.Size([100, 100])
Train loss: 1.584: 94%|-----? | 145/155 [21:18<01:27, 8.78s/it]torch.Size([100, 100])
Train loss: 1.583: 94%|-----? | 146/155 [21:27<01:19, 8.82s/it]torch.Size([100, 100])
Train loss: 1.584: 95%|-----? | 147/155 [21:35<01:09, 8.71s/it]torch.Size([100, 100])
Train loss: 1.584: 95%|-----? | 148/155 [21:44<01:01, 8.77s/it]torch.Size([100, 100])
Train loss: 1.585: 96%|-----? | 149/155 [21:53<00:52, 8.76s/it]torch.Size([100, 100])
Train loss: 1.584: 97%|-----? | 150/155 [22:02<00:44, 8.87s/it]torch.Size([100, 100])
Train loss: 1.584: 97%|-----? | 151/155 [22:11<00:35, 8.88s/it]torch.Size([100, 100])
Train loss: 1.587: 98%|-----? | 152/155 [22:20<00:26, 8.94s/it]torch.Size([100, 100])
Train loss: 1.587: 99%|-----? | 153/155 [22:29<00:17, 8.82s/it]torch.Size([100, 100])
Train loss: 1.590: 99%|-----? | 154/155 [22:37<00:08, 8.72s/it]torch.Size([100, 100])

```

Рис 3.10. Приклад зміни показника «Train loss» під час навчання

Обробивши отриманні данні було сформовано графіки зміни показників «Train loss», «PixAcc» та «IoU» для періоду навчання, який представлено на рис. 3.11-3.13.

Після проведення навчання на будь-якій кількості епох формується найкраща модель по середньому значенню показників «PixAcc» а «IoU». Цю модель можна використати для тестування на зображеннях. Вихідний код скрипту приведений в додатку Б. Декілька прикладів порівняння сегментованих зображень з набору BraTS 2020, які були сегментовані вручну, з зображеннями які були сегментовані нейронною мережею, яка піддавалася дослідженню, приведено на рис. 3.14.

Важливим показником ефективності будь-якої нейронної мережі є час навчання. Під час проведення експерименту було встановлено що на обраному об'ємі зображень для тренування та перевірки результатів тренування після кожної із епох, час навчання на обраних робочих комп'ютерів складав приблизно 30 хвилин.

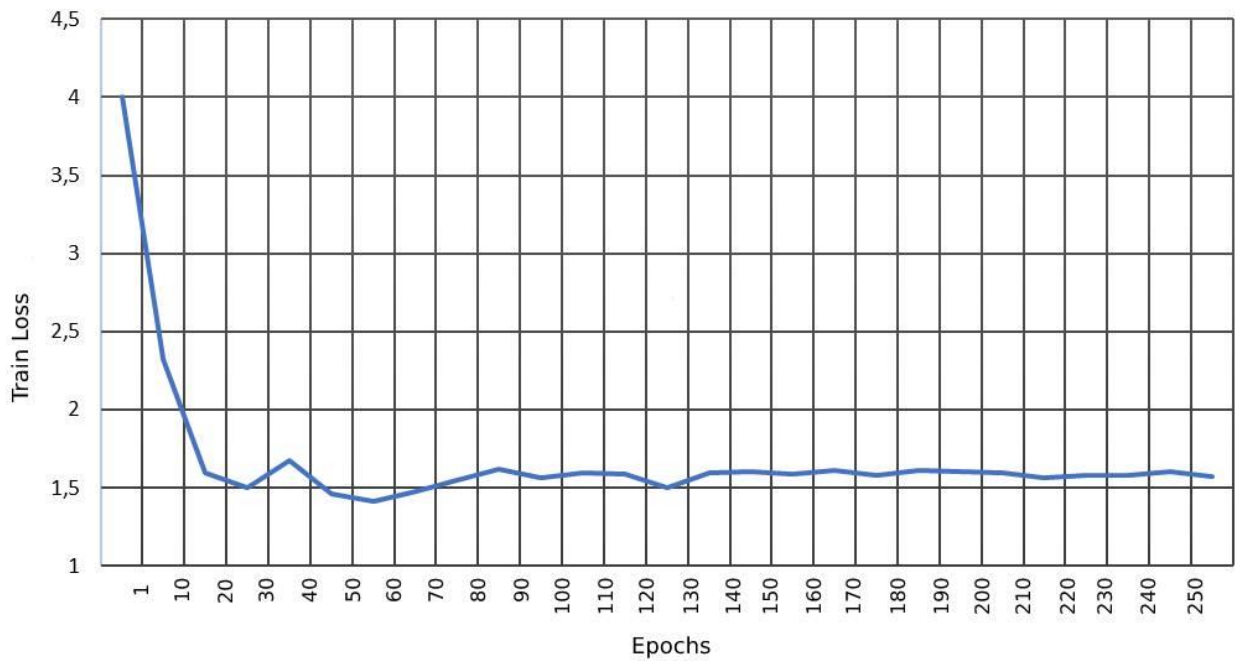


Рис 3.11. Графік залежності коефіцієнту «Train Loss» відносно часу навчання (кількості епох)

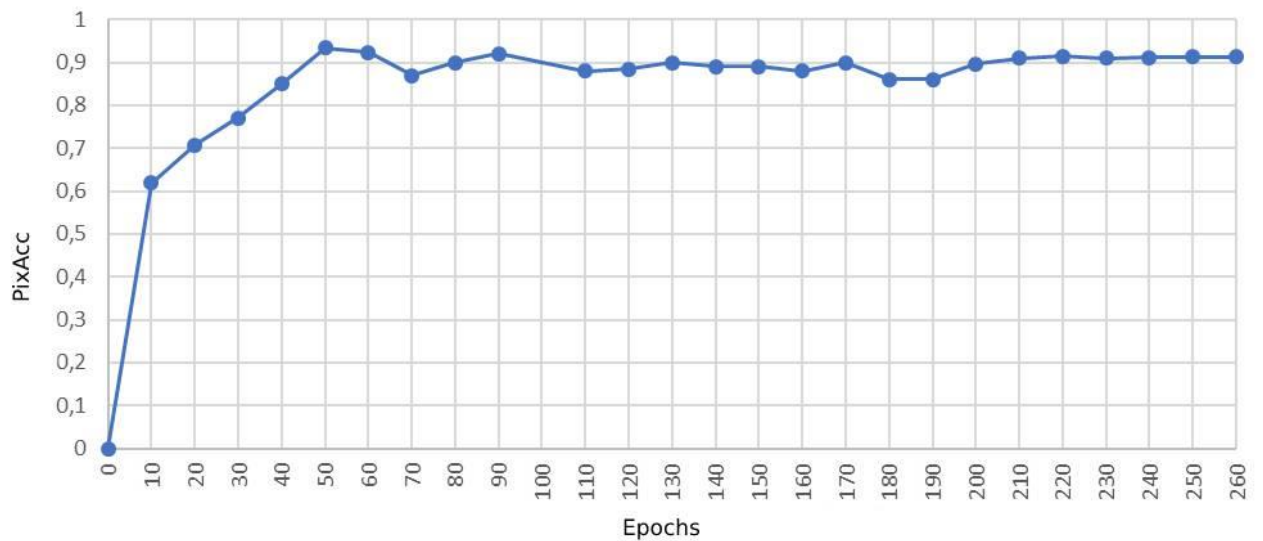


Рис 3.12. Графік залежності коефіцієнту «PixAcc» відносно часу навчання (кількості епох)

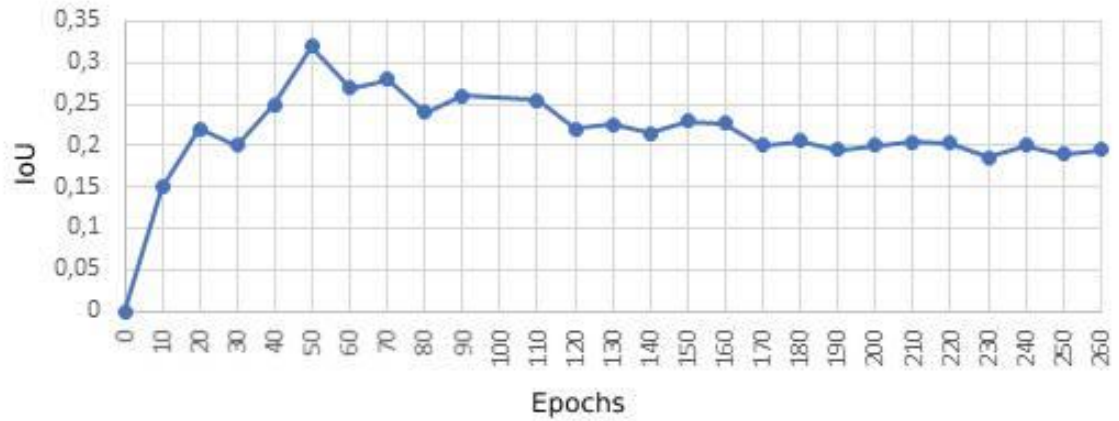


Рис 3.13. Графік залежності коефіцієнту «IoU» відносно часу навчання (кількості епох)

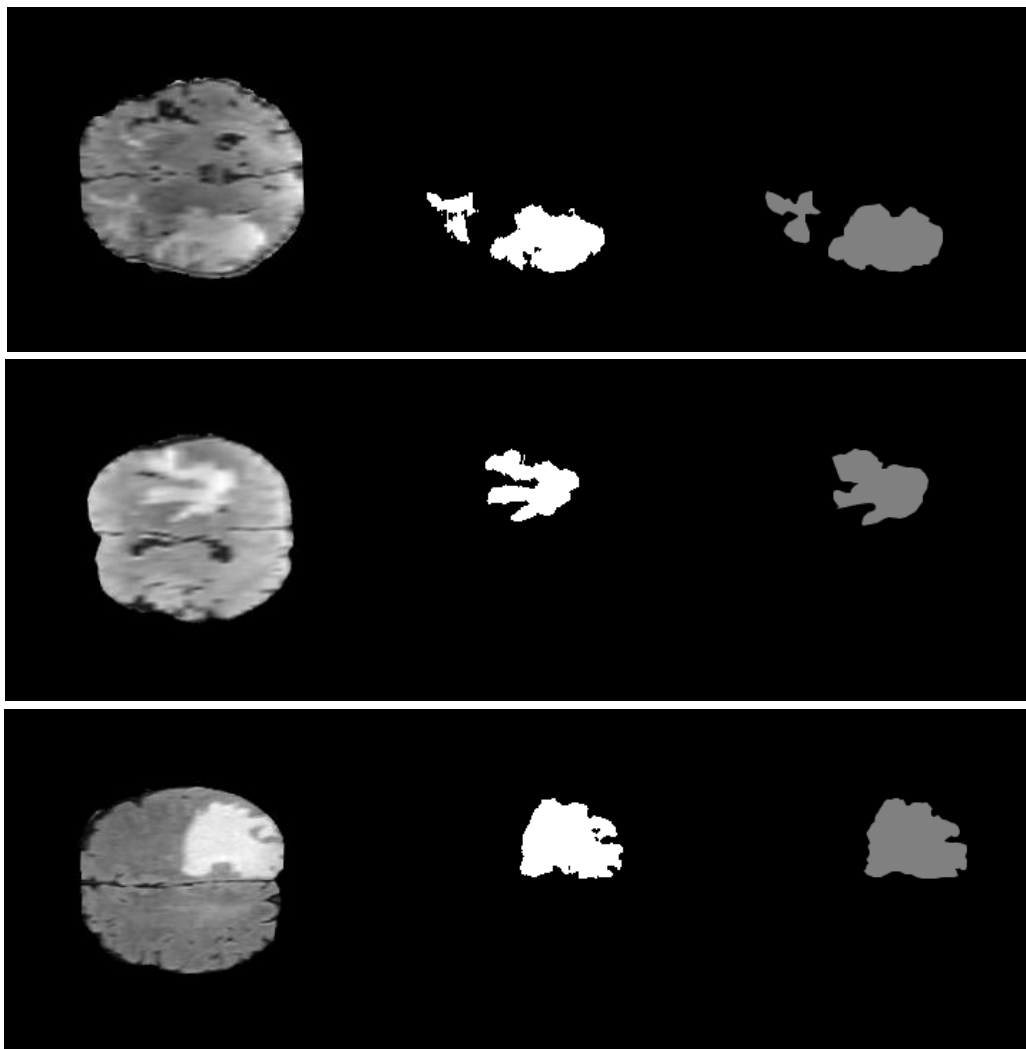


Рис 3.14. Приклади тестування навченої моделі на зображеннях (зліва – зображення із набору даних, посередині – вручну сегментоване зображення, справа – сегментоване зображення моделлю нейронної мережі)

Для порівняння складності обчислень, використовується кадр в секунду (Frames Per seconds) як метрику оцінки із вхідним зображенням 512×512 .

Для цього порівняння параметру з існуючими методами сегментації зображень було проведено експеримент (див. додаток Г) із двома іншими нейронними мережами: PSP та DeepLab. Як показано на рис. 3.15, показник частоти кадрів у середньому становить 100 прогонів. Для ResNet-50 досліджувальний метод (Encoding-JPU) працює приблизно в два рази швидше, ніж EncNet (Encoding-None). Швидкість методу також порівнянна з FPN, але метод досягає набагато кращих показників. Що стосується DeepLabV3 (ASPP) та PSP, метод може їх прискорити до певної міри, маючи кращу продуктивність.

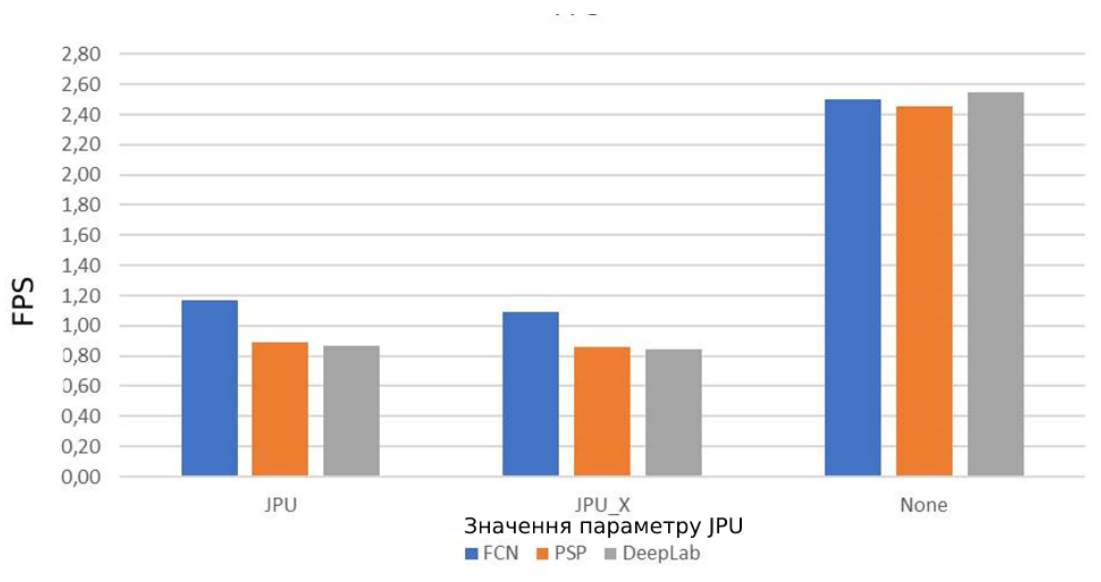


Рис. 3.15. Графік залежності показника FPS при вхідних параметрів JPU для різних нейронних мереж

3.6 Порівняння результатів експериментів на ПК з різними архітектурами

Зазвичай високонавантажені програми та програми пов'язані з обробкою великого обсягу даних проводять під Unix-подібними системами. Вище описані експерименти проводилися на операційній системі Windows, що зазвичай не досить добре підходить до виконання даного класу задач.

Для перевірки підвищення швидкодії було проведено описаний вище експеримент під операційною системою на базі Linux.

Експеримент проводився на комп'ютері наступної конфігурації:

- процесор: Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz. Кількість ядер – 6, кількість потоків – 12. Кеш – 15 MB Intel(R) Smart Cache

- встановлена пам'ять: 32GB DDR4;
- накопичувач: 1TB;

Для надійного запуску програми на іншому комп'ютері та під іншою операційною системою було вирішено створити Docker-образ наявної програми та виконати його запуск з використанням інструменту контейнеризації Docker. Для цього було створено Dockerfile та docker-compose файл (див. додаток Д.) для автоматичної конфігурації середовища для виконання програми. Після успішного налагодження програми вона була комп'ютері з описаними апаратними та програмним забезпеченням (рис. 3.16).

```

mslyu@tinuklin-srv: E - FastFCM - sudo docker logs 5e0ba29f2a0
Sun 29 Nov 2021 09:37:02 PM EET
Namespace(model='fcn', backbone='resnet50', gpu='GPU', dilated=False, lateral=False, dataset='ade20k', workers=16, base_size=520, crop_size=100, train_split='train', aux=False, aux_weight=0.2, se_loss=False, se_weight=0.2, epochs=99999, start_epoch=0, batch_size=16, test_batch_size=16, lr=0.01, lr_scheduler='poly', momentum=0.9, weight_decay=0.0001, no_cuda=False, seed=1, resume=None, checkpoint='test', model_zoo=None, ft=False, split_val, mode='testval', ns=False, no_val=True, save_folder='experiments/segmentation/results', cuda=False)
/etc/datasets
baseDataset: base_size 520, crop_size 100
len(img_paths): 7500
<enc.datasets.brats.BratsSegmentation object at 0x7f6b1c6e7130>
/etc/datasets
Using poly LR Scheduler!
Starting Epoch: 0
Total Epochs: 99999
Train Loss: 50.4219 / 240 | 113/46d
mslyu@tinuklin-srv: E - FastFCM -
Sun 29 Nov 2021 09:39:51 PM EET

```

Рис. 3.16. Виконання програми експерименту Unix-подібній системі в Docker

Слід зазначити, що завчасно було відомо що комп'ютер на якому проводився даний експеримент має краще апаратне забезпечення та покладалися надії підвищення ефективності виконання програми під Linux. Через це обсяг зображень було збільшено в 2 рази порівняно з попередніми експериментами.

Через деякий час було зібрано метрики, отримані під час навчання та тестування нейронної мережі. Отримані данні представлено на рис. 3.17-3.19.

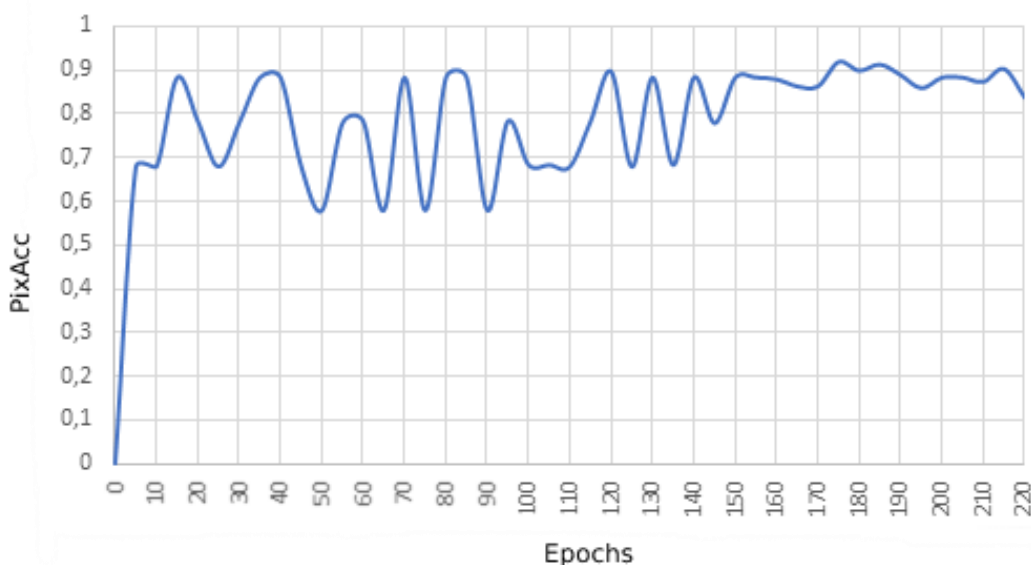


Рис. 3.17. Графік залежності коефіцієнту «PixAcc» відносно часу навчання (кількості епох) на Unix-подібній системі

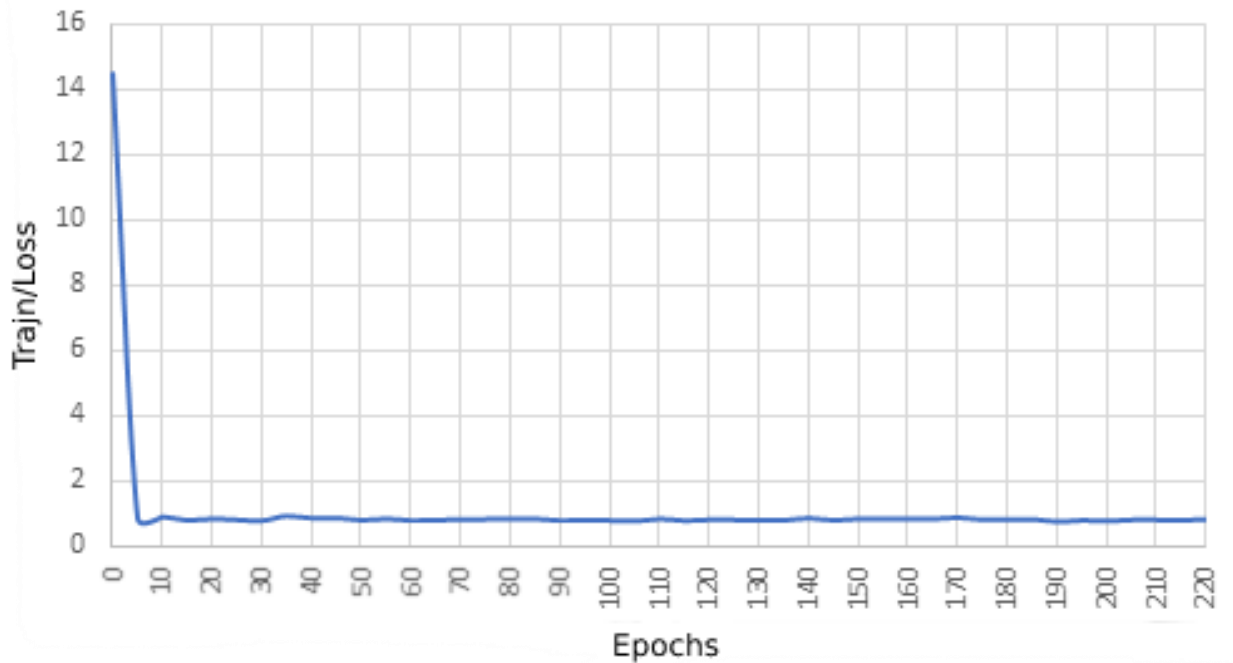


Рис. 3.18. Графік залежності коефіцієнту «Train Loss» відносно часу навчання (кількості епох) на Unix-подібній системі

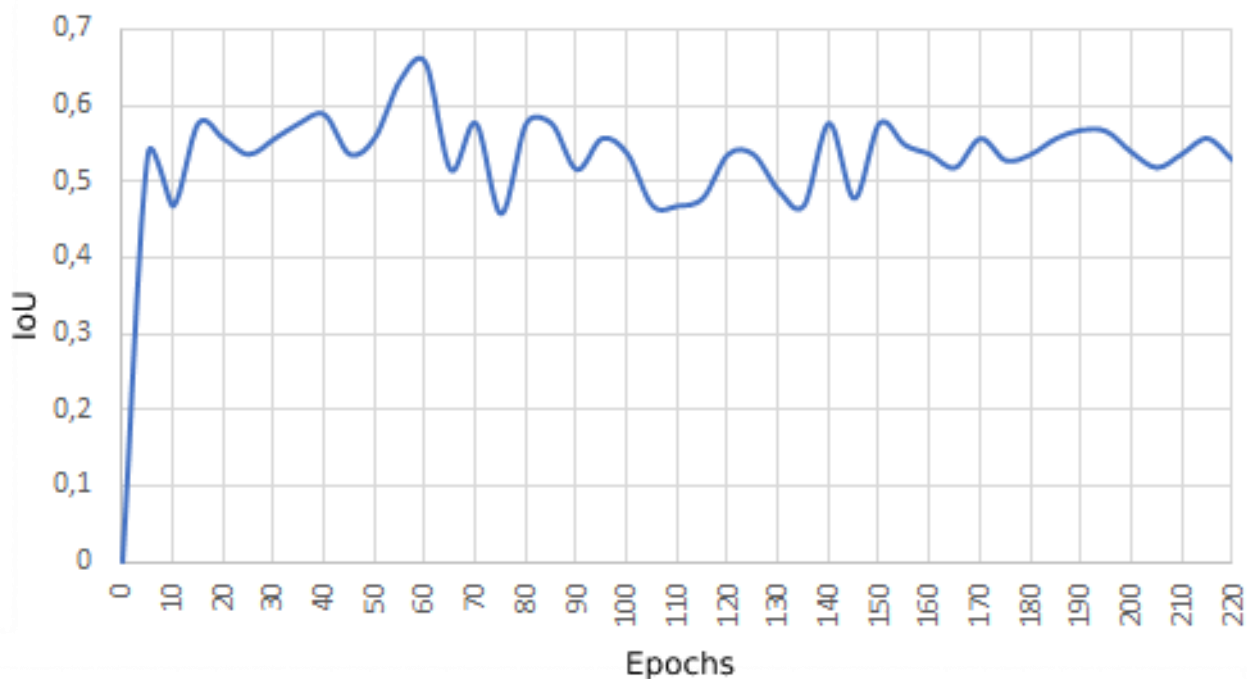


Рис. 3.19. Графік залежності коефіцієнту «IoU» відносно часу навчання (кількості епох) на Unix-подібній системі

Проаналізувавши та порівнявши отримані дані експериментів зроблено наступні висновки. Для двох наборів експериментів на різних архітектурах

рівень точності навчання нейронної мережі не знизився, тобто нейронна мережа стійка до зміни апаратного забезпечення в середовищі де вона навчається.

Час навчання обох експериментів відрізняється. На перших наборах експериментів час навчання складає приблизно 0,5 год. на епоху, в той час коли набір експериментів на Linux 0,2 год. на одну епоху. Навіть при більшому обсязі зображень нейронна мережа під операційною системою Linux виконує навчання в 2.5 рази швидше аніж Windows.

Щодо параметрів, відносно яких можна суди про ефективність нейронної мережі, то вони на обох наборах дещо відрізняються. Але загальна динаміка в цілому майже однакова.

На графіках експерименту з більшим обсягом зображень (рис. 3.17-3.19) виразно видно що рівень похибки в навчанні значно знизився. Але в той же час досягнення прийнятних рівнів параметрів P_{ix}Acc та IoU потребує більшої кількості епох.

Практична значимість проведених експериментів. За отриманими графіками, представленими на рис. 3.11-3.13 маємо такі результати:

- показник Train/Loss перестав змінюватися з 130 епохи;
- показник P_{ix}Acc не змінюється із 110 епохи;
- значення параметра IoU не змінюється при досягненні 170 епох.

Ці дані свідчать що для досягнення отриманих показників, при повторних проведеннях експериментів, можна проводити їх не 260 епох, а 170 епох.

Відповідні висновки за рис. 3.17-3.19:

- показник Train/Loss перестав змінюватися з 5 епох;
- показник P_{ix}Acc майже не змінюється із 150 епохи;
- значення параметра IoU не досягає майже стійкого рівня із 150-160 епохи.

Для отримання ефективних показників можна досягнути на навчанні в 160 епох.

ВИСНОВКИ

Під час проведення дослідження методу семантичної сегментації МРТ-зображень із використанням нейронної мережі було проаналізовано її відмінність від інших нейронних мереж і основі отриманих результатів аналізу функціонування обраної мережі було розроблено програмне забезпечення для проведення експериментального дослідження її ефективності.

Проведено експерименти з використанням розробленого програмного забезпечення на датасеті BraTS, призначеного для сегментації зображень магнітно-резонансної томографії головного мозку з обсягом в 5 000 і 10 000 зображень на різних операційних системах і комп'ютерах.

Розглянута нейронна мережа дозволяє отримати прийнятні результати за декількома коефіцієнтами (IoU, PixAcc, Train/Loss), які використовуються для визначення ефективності методів машинного навчання.

Точність навчання параметру PixAcc досягає я 92% при відносно не великій кількості епох.

Час обробки кадра в секунду в декілька разів кращий при використанні в нейронній мережі методу JPU та його модифікації JPUX.

Виконаний порівняльний аналіз досліджуваного методу при його виконання на комп'ютерах із різною архітектурою: з різними апаратним та програмним забезпеченням та при різній кількості зображень для навчання. Результатом є висновок що досліджувальна нейронна мереже може досягти високого рівня точності на різних комп'ютерах з різним рівнем продуктивності. Тому найголовніший параметр ефективності нейронної мережі такий як час отримання результату залежить від об'єму даних для навчання та продуктивності робочого комп'ютера де вона навчається.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Prince, J.L.: Snakes, Shapes, and Gradient Vector Flow. IEE Transactions on Image Processing: веб-сайт. URL: https://www.researchgate.net/publication/3326757_Prince_JL_Snakes_Shapes_and_Gradient_Vector_Flow_IEE_Transactions_on_Image_Processing_73_359-369 (дата звернення: 10.11.2021).
2. Gradient Vector Flow Deformable Models: веб-сайт. URL: <http://iacl.ece.jhu.edu/pubs/p120.pdf> (дата звернення: 10.11.2021).
3. An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation. веб-сайт. URL: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.214.8532&rep=rep1&type=pdf> (дата звернення: 23.09.2021).
4. Locally excitatory globally inhibitory oscillator networks. Веб-сайт. URL: <https://web.cse.ohio-state.edu/~wang.77/papers/Wang-Terman.tnn95.pdf> (дата звернення: 10.08.2021).
5. Image Segmentation Based on Oscillatory Correlation/ Веб-сайт. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.30.4931&rep=rep1&type=pdf> (дата звернення: 10.08.2021).
6. Deformable Models. The Visual Computer. Веб-сайт. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.221.6258&rep=rep1&type=pdf> (дата звернення: 10.08.2021).
7. Synchronization in Relaxation Oscillator Networks with Conduction Delays. Веб-сайт. URL: <http://web.cse.ohio-state.edu/~wang.77/papers/FJWC.nc01.pdf> (дата звернення: 03.10.2021).
8. Bayesian modeling of uncertainty in low-level vision. Веб-сайт. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.529.6143&rep=rep1&type=pdf> (дата звернення: 03.10.2021).
9. Segmentation of Medical Images Using LEGION. Веб-сайт. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.2406&rep=rep1&type=pdf> (дата звернення: 01.09.2021).
10. Survey over Image Thresholding techniques and quantitative performance evaluation. Веб-сайт. URL: http://www.cs.tau.ac.il/~turkel/imagepapers/segmentation_rev.pdf (дата звернення: 03.10.2021).
11. Image Analysis and Mathematical Morphology. Веб-сайт. URL: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.705.2541&rep=rep1&type=pdf> (дата звернення: 19.10.2021).
12. A Survey of Thresholding Techniques. Веб-сайт. URL: <https://pequan.lip6.fr/~bereziat/pima/2012/seuillage/sahoo88.pdf>. (дата звернення: 21.10.2021).
13. A System for Brain Image Segmentation and Classification Based on Three-Dimensional Convolutional Neural Network. Веб-сайт. URL:

- http://www.scielo.org.mx/scielo.php?pid=S1405-55462020000401617&script=sci_arttext&tlng=en (дата звернення: 17.11.2021).
14. Current Methods in Medical Image Segmentation. Веб-сайт. URL: <http://ee.sharif.edu/~miap/Files/MedicalImagesSegmentationReview.pdf> (дата звернення: 17.11.2021).
15. A Review on Image Segmentation Techniques. Веб-сайт. URL: https://www.isical.ac.in/~sankar/paper/PR_93_NRP_SKP.PDF. (дата звернення: 20.11.2021).
16. Веб-сайт. URL: https://www.researchgate.net/publication/224272048_Physics-based_deformable_organisms_for_medical_image_analysis (дата звернення: 01.11.2021).
17. Graph Theoretic Techniques for Cluster Analysis Algorithms. Веб-сайт. URL: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.7328&rep=rep1&type=pdf> (дата звернення: 01.11.2021).
18. Data Clustering: A Review. Веб-сайт. URL: http://users.eecs.northwestern.edu/~yingliu/datamining_papers/survey.pdf (дата звернення: 01.11.2021).
19. Gradient Vector Flow Deformable Models. Веб-сайт. URL: <http://iacl.ece.jhu.edu/pubs/p120.pdf> (дата звернення: 20.10.2021).
20. Markov Random Field Segmentation of Brain MR Images. Веб-сайт. URL: <https://arxiv.org/ftp/arxiv/papers/0903/0903.3114.pdf>. (дата звернення: 20.10.2021).
21. Neural Networks: A Comprehensive Foundation. Веб-сайт. URL: https://cdn.preterhuman.net/texts/science_and_technology/artificial_intelligence/Neural%20Networks%20-%20A%20Comprehensive%20Foundation%20-%20Simon%20Haykin.pdf (дата звернення: 20.11.2021).
22. A Neural Network Approach to Medical Image Segmentation and Three-Dimensional Reconstruction. Веб-сайт. URL: https://www.researchgate.net/publication/220776479_A_Neural_Network_Approach_to_Medical_Image_Segmentation_and_Three-Dimensional_Reconstruction (дата звернення: 20.11.2021).
23. Segmentation of Skull in 3D Human MR Images Using Mathematical Morphology. Веб-сайт. URL: <https://neuroimage.usc.edu/paperspdf/spie02dogdas.pdf/> (дата звернення: 17.11.2021).
24. An Active Contour Model for Mapping the Cortex. Веб-сайт. URL: <https://www.cbica.upenn.edu/sbia/papers/13.pdf> (дата звернення: 19.10.2021).
25. The Use of Active Shape Models For Locating Structures in Medical Images Веб-сайт. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.1047&rep=rep1&type=pdf> (дата звернення: 17.11.2021).

26. On Active Contour Models and Balloons. Веб-сайт. URL: <https://www.ceremade.dauphine.fr/~cohen/mypapers/cohenCVGIP91.pdf>. (дата звернення: 17.11.2021).

27. Neural network modelling. Веб-сайт. URL: https://www.researchgate.net/publication/21409449_Neural_network_modelling (дата звернення: 01.11.2021).

28. A Computational Approach to Edge Detection. Веб-сайт. URL: <http://pzs.dstu.dp.ua/ComputerGraphics/bibl/Canny.pdf> (дата звернення: 16.10.2021).

29. B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)", IEEE Transactions on Medical Imaging 34(10), 1993-2024 (2015) DOI: 10.1109/TMI.2014.2377694 (дата звернення: 16.10.2021).

30. S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J.S. Kirby, et al., "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features", Nature Scientific Data, 4:170117 (2017) DOI: 10.1038/sdata.2017.117.

31. S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, et al., "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge", arXiv preprint arXiv:1811.02629 (2018).

32. Завдання. Математична морфологія. Веб-сайт. URL: <https://kerchtt.ru/uk/zadanie-matematicheskaya-morfologiya/> (дата звернення: 16.10.2021)

33. Застосування методів штучного інтелекту до сегментації графічного образу. Веб-сайт:

http://ena.lp.edu.ua:8080/bitstream/ntb/12066/1/18_%D0%97%D0%90%D0%A1%D0%A2%D0%9E%D0%A1%D0%A3%D0%92%D0%90%D0%9D%D0%9D%D0%AF%20%D0%9C%D0%95%D0%A2%D0%9E%D0%94%D0%86%D0%92%20%D0%A8%D0%A2%D0%A3%D0%A7%D0%9D%D0%9E%D0%93%D0%9E%20%D0%86%D0%9D%D0%A2%D0%95%D0%9B%D0%95%D0%9A%D0%A2%D0%A3.pdf (дата звернення: 16.10.2021)

34. Метод автоматизированной сегментации изображений цитологических препаратов на основе модели активного контура. Веб-сайт: <https://cyberleninka.ru/article/n/metod-avtomatizirovannoy-segmentatsii-izobrazheniy-tsitologicheskikh-preparatov-na-osnove-modeli-aktivnogo-kontura/viewer> (дата звернення: 06.08.2021)

35. Image Segmentation with Clustering. Веб-сайт: <https://towardsdatascience.com/image-segmentation-with-clustering-b4bbc98f2ee6>. (дата звернення: 01.08.2021).

36. A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets. Веб-сайт: <https://link.springer.com/article/10.1007/s11042-021-10594-9/>. (дата звернення: 01.09.2021).

37. Introduction to Hierarchical Clustering. Веб-сайт: <https://towardsdatascience.com/introduction-hierarchical-clustering-d3066c6b560e>. (дата звернення: 19.09.2021).

38. Applicability and interpretability of Ward's hierarchical agglomerative clustering with or without contiguity constraints. Веб-сайт: <https://hal.archives-ouvertes.fr/hal-02294847v2/document>. (дата звернення: 28.07.2021).

39. Introduction to Image Segmentation with K-Means clustering. Веб-сайт: <https://www.kdnuggets.com/2019/08/introduction-image-segmentation-k-means-clustering.html> (дата звернення: 18.10.2021).

40. Image Segmentation Method Using Fuzzy C Mean Clustering Based on Multi-Objective Optimization: <https://iopscience.iop.org/article/10.1088/1742-6596/1004/1/012035/pdf> (дата звернення: 08.10.2021).

41. An optimal graph theoretic approach to data clustering. Веб-сайт: <https://cse.sc.edu/~songwang/csce790/Suppl/WuLeahy.pdf> (дата звернення: 15.10.2021).

42. Region Growing and Clustering Segmentation. Веб-сайт: https://sbme-tutorials.github.io/2019/cv/notes/6_week6.html (дата звернення: 08.10.2021).

43. Integral split-and-merge methodology for real-time image segmentation. Веб-сайт: <https://www.spiedigitallibrary.org/journals/journal-of-electronic-imaging/volume-24/issue-01/013007/Integral-split-and-merge-methodology-for-real-time-image-segmentation/10.1117/1.JEI.24.1.013007.full?SSO=1> (дата звернення: 08.10.2021).

44. Multi-Atlas Segmentation of Biomedical Images: A Survey. Веб-сайт: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4532640/> (дата звернення: 08.10.2021).

45. Genetics of brain structure and intelligence. Веб-сайт: https://www.researchgate.net/publication/8079309_Toga_AW_Thompson_PM_Genetics_of_brain_structure_and_intelligence_Ann_Rev_Neurosci_28_1-23 (дата звернення: 01.10.2021).

46. Fully Convolutional Networks for Semantic Segmentation. Веб-сайт: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Long_Fully_Convolutional_Networks_2015_CVPR_paper.pdf (дата звернення: 08.10.2021).

47. Review: DilatedNet — Dilated Convolution (Semantic Segmentation). Веб-сайт: <https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5> (дата звернення: 04.07.2021).

ДОДАТКИ

ДОДАТОК А

Вихідний код програми перетворення NifTy файлу до PNG-файлу

```
#!/usr/bin/python

import gzip
import shutil
import os

# traverse root directory, and list directories as dirs and files as files
if __name__ == '__main__':
    for dirpath, subdirs, files in os.walk('.'):
        for x in files:
            if x.find('flair') > 0 and x.find('gz')>0:
                with gzip.open(os.path.join(dirpath, x), 'rb') as f_in:
                    with open(os.path.join(os.path.join('.',f),x.replace('.gz','')), 'wb') as f_out:
                        shutil.copyfileobj(f_in, f_out)

!/usr/bin/env python

import scipy, numpy, shutil, os, nibabel
import sys, getop
import imageio
import scipy, numpy, shutil, os, nibabel
import sys, getopt

import imageio
from PIL import Image

def main(inputfile, outputfile):
    image_array = nibabel.load(inputfile).get_data()

    nx, ny, nz = image_array.shape

    if not os.path.exists(outputfile):
        os.makedirs(outputfile)

    total_slices = image_array.shape[2]
```

```

slice_counter = 0
for current_slice in range(0, total_slices):
    if (slice_counter % 1) == 0:

        data = image_array[:, :, current_slice]

        if (slice_counter % 1) == 0:
            image_name = inputfile[:-4] + "_z" + "{:0>3}".format(str(current_slice + 1)) + ".png"
            imageio.imwrite(image_name, data)

            src = image_name
            shutil.move(src, outputfile)
            slice_counter += 1

#call the function to start the program
if __name__ == "__main__":
    for dirpath, subdirs, files in os.walk(os.path.join('.', 's')):
        for x in files:
            main(os.path.join(dirpath, x), os.path.join(".", 's_png'))

if __name__ == "__main__":
    for dirpath, subdirs, files in os.walk(os.path.join('.', 'f_png')):
        for x in files:
            if x.find('seg'):
                im = Image.open(os.path.join(dirpath, x))
                rgb_im = im.convert('RGB')
                rgb_im.save(os.path.join(os.path.join('.', 'f_jpg'), x.replace('png','jpg')))

```


ДОДАТОК Б

Вихідний код навчання нейронної мережі

```

import os
import numpy as np
from tqdm import tqdm
import torch
from torch.utils import data
import torchvision.transforms as transform
from torch.nn.parallel.scatter_gather import gather
import encoding.utils as utils
from encoding.nn import SegmentationLosses, SyncBatchNorm
from encoding.parallel import DataParallelModel, DataParallelCriterion
from encoding.datasets import get_segmentation_dataset
from encoding.models import get_segmentation_model
from option import Options

torch_ver = torch.__version__[:3]
if torch_ver == '0.3':
    from torch.autograd import Variable

class Trainer():
    def __init__(self, args):
        self.args = args
        # data transforms
        input_transform = transform.Compose([
            transform.ToTensor(),
            transform.Normalize([.485, .456, .406], [.229, .224, .225])])
        # dataset
        data_kwargs = {'transform': input_transform, 'base_size': args.base_size,
            'crop_size': args.crop_size}
        trainset = get_segmentation_dataset(args.dataset, split=args.train_split, mode='train',
            **data_kwargs)

        print(trainset)
        testset = get_segmentation_dataset(args.dataset, split='val', mode='val',
            **data_kwargs)

        # dataloader
        kwargs = {'num_workers': args.workers, 'pin_memory': True} \
            if args.cuda else {}
        self.trainloader = data.DataLoader(trainset, batch_size=args.batch_size,
            drop_last=True, shuffle=True, **kwargs)
        self.valloader = data.DataLoader(testset, batch_size=args.batch_size,
            drop_last=False, shuffle=False, **kwargs)
        self.nclass = trainset.num_class
        # model

```

```

model = get_segmentation_model(args.model, dataset = args.dataset,
                               backbone = args.backbone, dilated = args.dilated,
                               lateral = args.lateral, jpu = args.jpu, aux = args.aux,
                               se_loss = args.se_loss, norm_layer = torch.nn.BatchNorm2d,
                               base_size = args.base_size, crop_size = args.crop_size)
# optimizer using different LR
params_list = [{'params': model.pretrained.parameters(), 'lr': args.lr},]
if hasattr(model, 'jpu'):
    params_list.append({'params': model.jpu.parameters(), 'lr': args.lr*10})
if hasattr(model, 'head'):
    params_list.append({'params': model.head.parameters(), 'lr': args.lr*10})
if hasattr(model, 'auxlayer'):
    params_list.append({'params': model.auxlayer.parameters(), 'lr': args.lr*10})
optimizer = torch.optim.SGD(params_list, lr=args.lr,
                             momentum=args.momentum, weight_decay=args.weight_decay)
# criterions
self.criterion = SegmentationLosses(se_loss=args.se_loss, aux=args.aux,
                                    nclass=self.nclass,
                                    se_weight=args.se_weight,
                                    aux_weight=args.aux_weight)
self.model, self.optimizer = model, optimizer
# resuming checkpoint
self.best_pred = 0.0
if args.resume is not None:
    if not os.path.isfile(args.resume):
        raise RuntimeError("=> no checkpoint found at '{}'".format(args.resume))
    checkpoint = torch.load(args.resume)
    args.start_epoch = checkpoint['epoch']
    self.model.load_state_dict(checkpoint['state_dict'])
    if not args.ft:
        self.optimizer.load_state_dict(checkpoint['optimizer'])
    self.best_pred = checkpoint['best_pred']
    print("=> loaded checkpoint '{}' (epoch {})"
          .format(args.resume, checkpoint['epoch']))
# clear start epoch if fine-tuning
if args.ft:
    args.start_epoch = 0
# lr scheduler
self.scheduler = utils.LR_Scheduler(args.lr_scheduler, args.lr,
                                    args.epochs, len(self.trainloader))

def training(self, epoch):
    train_loss = 0.0
    self.model.train()
    tbar = tqdm(self.trainloader)
    for i, (image, target) in enumerate(tbar):
        self.scheduler(self.optimizer, i, epoch, self.best_pred)

```

```

self.optimizer.zero_grad()

if torch_ver == "0.3":
    image = Variable(image)
    target = Variable(target)

outputs = self.model(image)
loss = self.criterion(*outputs, target)
loss.backward()
self.optimizer.step()
train_loss += loss.item()
tbar.set_description('Train loss: %.3f % (train_loss / (i + 1)))

def validation(self, epoch):
    # Fast test during the training
    def eval_batch(model, image, target):
        outputs = model(image)
        pred = outputs[0]
        correct, labeled = utils.batch_pix_accuracy(pred.data, target)
        inter, union = utils.batch_intersection_union(pred.data, target, self.nclass)
        return correct, labeled, inter, union

is_best = False
self.model.eval()
total_inter, total_union, total_correct, total_label = 0, 0, 0, 0
tbar = tqdm(self.valloader, desc='\r')
for i, (image, target) in enumerate(tbar):
    if torch_ver == "0.3":
        image = Variable(image, volatile=True)
        correct, labeled, inter, union = eval_batch(self.model, image, target)
    else:
        with torch.no_grad():
            correct, labeled, inter, union = eval_batch(self.model, image, target)

    total_correct += correct
    total_label += labeled
    total_inter += inter
    total_union += union
    pixAcc = 1.0 * total_correct / (np.spacing(1) + total_label)
    IoU = 1.0 * total_inter / (np.spacing(1) + total_union)
    mIoU = IoU.mean()
    tbar.set_description(
        'pixAcc: %.3f, mIoU: %.3f % (pixAcc, mIoU))

new_pred = (pixAcc + mIoU)/2
if new_pred > self.best_pred:
    is_best = True

```

```
        self.best_pred = new_pred
    utils.save_checkpoint({
        'epoch': epoch + 1,
        'state_dict': self.model.state_dict(),
        'optimizer': self.optimizer.state_dict(),
        'best_pred': new_pred,
    }, self.args, is_best)

if __name__ == "__main__":
    args = Options().parse()
    torch.manual_seed(args.seed)
    trainer = Trainer(args)
    print('Starting Epoch:', trainer.args.start_epoch)
    print('Total Epoches:', trainer.args.epochs)
    for epoch in range(trainer.args.start_epoch, trainer.args.epochs):
        trainer.training(epoch)
        trainer.validation(epoch)
```

ДОДАТОК В

Вихідний код скрипту тестування нейронної мережі

```

import os
import torch
import torchvision.transforms as transform
import encoding.utils as utils
from tqdm import tqdm
from torch.utils import data
from encoding.nn import BatchNorm
from encoding.datasets import get_segmentation_dataset, test_batchify_fn
from encoding.models import get_model, get_segmentation_model, MultiEvalModule
from .option import Options

def test(args):
    # output folder
    outdir = args.save_folder
    if not os.path.exists(outdir):
        os.makedirs(outdir)
    # data transforms
    input_transform = transform.Compose([
        transform.ToTensor(),
        transform.Normalize([.485, .456, .406], [.229, .224, .225])])
    # dataset
    testset = get_segmentation_dataset(args.dataset, split=args.split, mode=args.mode,
    transform=input_transform)

    # dataloader
    loader_kwargs = {'num_workers': args.workers, 'pin_memory': True}
    test_data = data.DataLoader(testset, batch_size=args.test_batch_size, drop_last=False,
    shuffle=False, collate_fn=test_batchify_fn, **loader_kwargs)
    # model
    if args.model_zoo is not None:
        model = get_model(args.model_zoo, pretrained=True)
    else:
        model = get_segmentation_model(args.model, dataset = args.dataset, backbone =
    args.backbone, dilated = args.dilated, lateral = args.lateral, jpu = args.jpu,
    aux = args.aux, se_loss = args.se_loss, norm_layer = BatchNorm, base_size
    = args.base_size, crop_size = args.crop_size)
    # resuming checkpoint
    if args.resume is None or not os.path.isfile(args.resume):
        raise RuntimeError("=> no checkpoint found at '{}'".format(args.resume))
    checkpoint = torch.load(args.resume)
    # strict=False, so that it is compatible with old pytorch saved models
    model.load_state_dict(checkpoint['state_dict'])
    print("=> loaded cdddcheckpoint '{}' (epoch {})".format(args.resume, checkpoint['epoch']))

```

```

scales = [0.5, 0.75, 1.0, 1.25, 1.5, 1.75]
if not args.ms:
    scales = [1.0]
evaluator = MultiEvalModule(model, testset.num_class, scales=scales, flip=args.ms).cuda()
evaluator.eval()
metric = utils.SegmentationMetric(testset.num_class)

tbar = tqdm(test_data)
for i, (image, dst) in enumerate(tbar):
    if 'val' in args.mode:
        with torch.no_grad():
            predicts = evaluator.parallel_forward(image)
            metric.update(dst, predicts)
            pixAcc, mIoU = metric.get()
            tbar.set_description('pixAcc: %.4f, mIoU: %.4f' % (pixAcc, mIoU))
    else:
        with torch.no_grad():
            outputs = evaluator.parallel_forward(image)
            predicts = [testset.make_pred(torch.max(output, 1)[1].cpu().numpy()) for output in
outputs]
            for predict, impath in zip(predicts, dst):
                mask = utils.get_mask_pallette(predict, args.dataset)
                outname = os.path.splitext(impath)[0] + '.png'
                mask.save(os.path.join(outdir, outname))

if __name__ == "__main__":
    args = Options().parse()
    torch.manual_seed(args.seed)
    args.test_batch_size = 1
    test(args)

```

ДОДАТОК Г

Вихідний код тестування параметру FPS

```
import time
import torch
import encoding
from encoding.nn import BatchNorm
from .option import Options

if __name__ == "__main__":
    args = Options().parse()
    model = encoding.models.get_segmentation_model(args.model, dataset = args.dataset, backbone
= args.backbone, dilated = args.dilated, lateral = args.lateral, jpu = args.jpu, aux = args.aux, se_loss
= args.se_loss, norm_layer = BatchNorm)

    num_parameters = sum([l.nelement() for l in model.pretrained.parameters()])
    print(num_parameters)
    num_parameters = sum([l.nelement() for l in model.head.parameters()])
    print(num_parameters)

    model.eval()
    x = torch.Tensor(1, 3, 512, 512).cuda()

    N = 10
    with torch.no_grad():
        for _ in range(N):
            out = model(x)

    result = []
    for _ in range(10):
        st = time.time()
        for _ in range(N):
            out = model(x)
        result.append(N/(time.time()-st))

    import numpy as np
    print(np.mean(result), np.std(result))
```

ДОДАТОК Д

Вихідний код файлів Dockerfile та docker-compose для запуску програми на Linux

Вміст файлу Dockerfile:

```
FROM python:3.9
ARG epochs
ARG checkname
WORKDIR .
COPY ./ ./
RUN pip install --upgrade pip
RUN pip install -r requirements.txt

ENTRYPOINT python ./train.py --dataset brats --model fcn --jpu JPU --backbone resnet50 --
checkname $checkname --no-val --epochs $epochs --crop-size 100
```

Вміст файлу docker-compose.yml:

```
version: '3'

services:
  fastfcn:
    build: .
    restart: always
    volumes:
      - ./runs:/experiments/segmentation/runs
      - ./results:/resFiles
    environment:
      - epochs=99999
      - checkname=test
```