НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«На правах рукопису»
УДК

«До захисту допущено»
Завідувач кафедри
Стіренко С.Г.
(підпис) (ініціали, прізвище)
“ ” 20***20*** р.

Магістерська дисертація

зі спеціальності: *123. Комп’ютерна інженерія*
(код та назва напряму підготовки або спеціальності)

Спеціалізація: *123. Програмування комп’ютерних систем та мереж*

на тему: Метод обробки медичних зображень людини на основі нейронної мережі

Виконав (-ла): студент (-ка) 2 курсу, групи ІВ-91мп
(шифр групи)

Домс Володимир Геннадійович
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник проф., д.ф.-м.н., с.н.с Гордієнко Юрій Григорович
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент к.т.н, доцент кафедри АУТС Писаренко Андрій Володимирович
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент
(підпис)

Київ – 2020 року

ЗМІСТ

ВСТУП

Швидкий розвиток сучасної медицини супроводжується тісною взаємодією з суміжними областями - математикою, фізикою, хімією. Одним з таких взаємодій є обробка та аналіз медичних зображень. Зображення анатомічного, гістологічної будови і функцій людського тіла є фундаментальними для медичної науки. Діагностика захворювань, лікування і управління терапевтичними процедурами спираються на дані, одержувані медичної візуалізацією.

Дуже широке коло завдань у аналізі медичних зображень пов'язаний з дерматології. Взагалі, досліджуються всі області, де є зображення. Кожна хвороба - окрема область досліджень. Тому з великої кількості хвороб і різних медичних апаратів складається велике розмаїття в цих дослідженнях. Для деяких аналізів, особливо для ультразвуку і ряду інших технологій (МРТ, комп'ютерної томографії), дуже важливо, на якому приладі це робиться - алгоритми пишуться для кожної конкретної моделі і її режимів.

Специфіка обробки і аналізу медичних зображень, у першу чергу, пов'язана з необхідністю працювати з медиками. Але медиків-дослідників практично немає, є тільки поодинокі винятки. Всі наші вітчизняні установи, у першу чергу, орієнтовані на лікування хворих, а не на дослідження. Тому лікарі дуже завантажені. Вони готові зустрітися з фахівцями з аналізу даних, дати якісь дані. Але якщо необхідно провести дослідження на дорогій та складній апаратурі, то це вступає в протиріччя з практикою, необхідністю оплати таких досліджень і використання цих апаратів і так далі.

Тому аналіз медичних зображень - актуальна тема, пов'язана, у першу чергу, з комп'ютерною діагностикою. Ми сподіваємося, що математичні методи і машинне навчання в майбутньому допоможуть значно спростити, прискорити та здешевити діагностику захворювань, особливо на ранніх стадіях. Діагнози будуть точніше, вони будуть встановлюватися в короткі терміни, а отже, шансів зберегти здоров'я і життя буде більше.

РОЗДІЛ 1. ОГЛЯД СФЕРИ ОБРОБКИ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

1.1. Огляд сфери машинного навчання

У машинному навчанні розробляються і вивчаються методи, які дають комп'ютерам можливість вирішувати проблеми на основі досвіду. Мета полягає в тому, щоб створити математичні моделі, які можна навчити для отримання корисних вихідних даних при подачі вхідних даних. Моделі машинного навчання надаються у вигляді навчальних даних і налаштовуються для отримання точних прогнозів навчальних даних за допомогою алгоритму оптимізації. Основна мета моделей полягає в тому, щоб мати можливість узагальнити отриманий досвід і надати правильні прогнози для нових, невидимих даних. Здатність моделі до узагальнення зазвичай оцінюється під час навчання з використанням окремого набору даних, набору перевірки, і використовується в якості зворотнього зв'язку для подальшої настройки моделі. Після декількох ітерацій навчання і настройки остаточна модель оцінюється на тестовому наборі, що використовується для моделювання того, як модель буде працювати при зіткненні з новими, невидимими даними.

Існує кілька видів машинного навчання, які можна умовно розділити на категорії в залежності від того, як моделі використовують вхідні дані під час навчання. При навчанні з підкріпленням конструюються агенти, які навчаються в своєму середовищі методом проб і помилок, оптимізуючи при цьому деякі цільові функції. При навчанні без учителя комп'ютера ставиться завдання виявляти закономірності в даних без нашого керівництва. Кластеризація - яскравий тому приклад. Більшість сьогоднішніх систем машинного навчання відносяться до класу контрольованого навчання. Тут комп'ютеру надається набір вже помічених або анотованих даних, і його просять створити правильні мітки для нових, раніше невидимих наборів даних на основі правил, виявлених в позначеному наборі даних. На основі набору прикладів введення-виведення вся модель навчається для виконання певних завдань обробки даних. Анотації до зображень з використанням

даних, позначених людьми, наприклад класифікація шкірних уражень відповідно із злоякісними новоутвореннями або виявлення факторів ризику серцево-судинних захворювань на фотографіях очного дна сітківки – два приклади безлічі проблем, пов'язаних з медичної візуалізацією, які вирішуються за допомогою контрольованого навчання.

Машинне навчання має довгу історію і розділене на безліч підрозділів, з яких глибинне навчання в даний час привертає найбільшу увагу.

Штучні нейронні мережі

Штучні нейронні мережі (ШНМ) - одна з найвідоміших моделей машинного навчання, представлена ще в 1950-х.

Грубо кажучи, нейронна мережа складається з ряду пов'язаних обчислювальних одиниць, званих нейронами, розташованих по верствам. Існує вхідний рівень, на якому дані надходять в мережу, за яким слідують один або кілька прихованих шарів, що перетворюють дані в міру їх проходження, перш ніж закінчитися вихідним шаром, який виробляє прогнози нейронної мережі. Мережа навчається виводити корисні прогнози шляхом ідентифікації шаблонів в наборі помічених навчальних даних, що подаються через мережу, в той час як вихідні дані порівнюються з фактичними мітками за допомогою цільової функції. Під час навчання параметри мережі - потужність кожного нейрона - налаштовуються до тих пір, поки патерни, ідентифіковані мережею, не приведуть до хороших прогнозів для навчальних даних. Після того, як шаблони вивчені, мережа може використовуватися для прогнозування нових, невидимих даних, тобто узагальнення для нових даних.

Давно відомо, що ШНМ дуже гнучкі, здатні моделювати і вирішувати складні завдання, але також і те, що тренувати їх складно і дуже затратно з точки зору обчислень. Це знизило їх практичну корисність і спонукало людей до недавнього часу зосередитися на інших. моделі машинного навчання. Але до теперішнього часу штучні нейронні мережі є одним з домінуючих методів

машинного навчання і найбільш інтенсивно вивчаються. Ця зміна відбулася завдяки зростанню великих даних, потужним процесорам для паралельних обчислень (зокрема, графічним процесорам), деяких важливих налаштувань алгоритмів, використовуваних для побудови і навчання мереж, а також розробка простих у використанні програмних середовищ. Сплеск інтересу до ІНС призводить до неймовірних темпах розвитку, що також стимулює інші частини машинного навчання.

Навчання з учителем (контрольоване навчання)

Навчання з учителем (supervised learning) передбачає наявність повного набору розмічених даних для тренування моделі на всіх етапах її побудови.

Наявність повністю розмічені датасета означає, що кожному наприклад в навчальному наборі відповідає відповідь, який алгоритм і повинен отримати. Таким чином, розмічений датасета з фотографій квітів навчить нейронну мережу, де зображені троянди, ромашки або нарциси. Коли мережу отримає нове фото, вона порівняє його з прикладами з навчальної датасета, щоб передбачити відповідь.

В основному навчання з учителем застосовується для вирішення двох типів задач: класифікації і регресії.

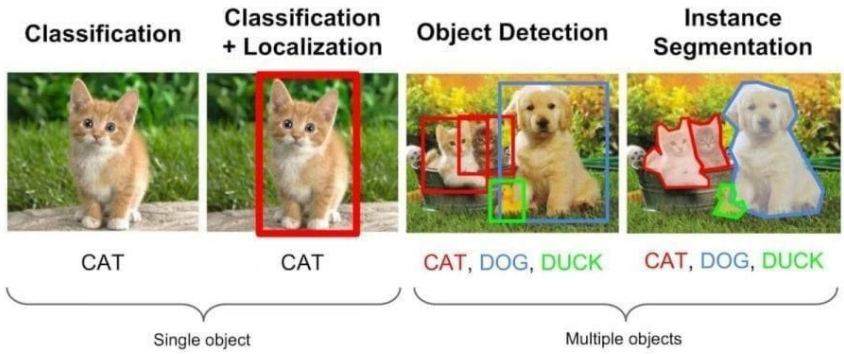


Рис. 1.1. Зображення механізму пошуку та класифікації об'єктів на зображенні

Задача класифікації.

Класифікація - це проблема ідентифікації, до якої з набору категорій (підгруп) належить нове спостереження, на основі навчального набору даних, що

містить спостереження (або випадки), приналежність яких до категорії відома. Прикладами є віднесення даного електронного листа до класу "спам" або "не спам" та встановлення діагнозу для даного пацієнта на основі спостережуваних характеристик пацієнта (стать, артеріальний тиск, наявність або відсутність певних симптомів тощо). Класифікація є прикладом розпізнавання зразків.

У термінології машинного навчання класифікація вважається екземпляром контрольованого навчання, тобто навчання, де доступний навчальний набір правильно визначених спостережень. Відповідна непідконтрольна процедура відома як кластеризація і передбачає групування даних за категоріями на основі певної міри притаманної подібності чи відстані.

Часто окремі спостереження аналізуються на набір кількісних властивостей, відомих по-різному як пояснювальні змінні або ознаки. Ці властивості можуть бути різними категоріями (наприклад, "A", "B", "AB" або "O", для групи крові), порядковими (наприклад, "великими", "середніми" або "малими"), цілочисельними (наприклад, кількість зустрічей певного слова в електронному листі) або справжнє значення (наприклад, вимірювання артеріального тиску). Інші класифікатори працюють, порівнюючи спостереження з попередніми спостереженнями за допомогою функції подібності або відстані.

Алгоритм, який реалізує класифікацію, особливо в конкретній реалізації, відомий як класифікатор. Термін "класифікатор" іноді також відноситься до математичної функції, реалізованої алгоритмом класифікації, яка відображає вхідні дані до категорії.

Термінологія між полями досить різноманітна. У статистиці, де класифікація часто проводиться за допомогою логістичної регресії або подібної процедури, властивості спостережень називаються пояснювальними змінними (або незалежними змінними, регресорами тощо), а категорії, що передбачаються, відомі як результати, які, як вважають, можливі значення залежної змінної. У машинному навчанні спостереження часто називають екземплярами, пояснювальні змінні називаються ознаками (згруповані у вектор ознак), а

можливими категоріями, які слід передбачити, є класи. В інших полях може використовуватися інша термінологія: напр. в екології громади термін "класифікація", як правило, позначає кластерний аналіз, тобто тип навчання без нагляду, а не навчання під контролем.

Типи вхідних даних:

- Ознаковий опис — найпоширеніший випадок. Кожен об'єкт описується набором своїх характеристик, що називаються ознаками. Ознаки можуть бути числовими або не числовими.
- Матриця відстаней між об'єктами. Кожен об'єкт описується відстанями до всіх інших об'єктів навчальної вибірки. З цим типом вхідних даних працюють деякі методи, зокрема, метод k найближчих сусідів, метод парзенівського вікна, метод потенційних функцій.
- Часовий ряд або сигнал є послідовністю вимірювань в часі. Кожен вимір може представлятися числом, вектором, а в загальному випадку — ознаковим описом досліджуваного об'єкта в даний момент часу.
- Зображення або відеоряд.
- Зустрічаються й складніші випадки, коли вхідні дані подаються у вигляді графів, текстів, результатів запитів до бази даних тощо. Як правило, вони приводяться до першого або другого випадку шляхом попередньої обробки даних та виділення ознак.

Типи відгуків:

- Коли множина можливих відповідей нескінченна (відповіді є дійсними числами або векторами), говорять про задачі регресії та апроксимації;
- Коли множина можливих відповідей звичайна, говорять про задачі класифікації та розпізнавання образів;
- Коли відповіді характеризують майбутню поведінку процесу або явища, кажуть про задачі прогнозування.

Задача регресії.

Лінійна регресія є привабливою моделлю, тому що її уявлення дуже просте.

Уявлення - це лінійне рівняння, що об'єднує певний набір вхідних значень (X) рішень, до якого є прогнозований висновок для цього набору вхідних значень (y). Таким чином, як значення вхідних даних (x), так і вихідне значення є числовими.

Лінійне рівняння привласнює масштабний коефіцієнт (по-англійськи "scale factor") до кожного вхідного значення X. Масштабний коефіцієнт представлений грецькою буквою Beta (B). Доданий також один додатковий коефіцієнт, що додає додатковий ступінь свободи (наприклад, рух вгору і вниз по двовимірному дільниці) і часто називають коефіцієнтом перехоплення або зсуву (по-англійськи "bias coefficient").

Найбільш просте завдання регресії коли на вхід подається одна змінна X і є одне виходить значення Y. Форма подібної моделі буде:

$$Y = B0 + B1 * X$$

У разі багатовимірних вимірів (тобто коли у нас є більше однієї вступної змінної (X)), лінія перетворюється в площиною або гіпер-площині. Таким чином, уявлення є формою рівняння і конкретні значення, використовувані для коефіцієнтів (наприклад, B0 і B1 в наведеному вище прикладі).

Коли говорять про складності регресійній моделі, такий як лінійна регресія - мають на увазі кількість коефіцієнтів, які використовуються в моделі.

Коли конкретний елемент коефіцієнт Beta стає нульовим, він ефективно видаляє вплив вхідної змінної на модель і, отже, впливу на прогноз моделі (0 * Xi = 0). Це стає актуальним, якщо ви застосовуєте методи регуляризації (про них ми розповімо окремо), які змінюють алгоритм навчання, щоб зменшити складність моделей регресії, чинячи тиск на абсолютний розмір коефіцієнтів, приводячи деякі з них до нуля.

Методи лінійної регресії

Вивчення моделі лінійної регресії означає дослідження одержуваних значень коефіцієнтів, що використовуються в поданні, на основі наявних вхідних даних.

У цій частині уроку ми коротко розглянемо чотири методи для підготовки моделі для лінійної регресії. Це мало інформації для реалізації їх з нуля, але досить, щоб отримати перші враження і компроміси при їх обчисленні.

Є ще багато методів, тому що модель лінійної регресії так добре вивчені. Важливо звернути увагу що на метод найменших квадратів, тому що це найбільш поширений метод, який використовується в цілому в індустрії для задач оптимізації. Також зверніть увагу метод градієнтного спуску (по-англійськи Gradient descent), як найбільш поширений метод застосовується в різних класах завдань машинного навчання.

Проста лінійна регресія

При простої лінійної регресії, коли у нас є один вхідний параметр, ми можемо використовувати статистику для оцінки коефіцієнтів.

Для цього необхідно обчислити статистичні властивості на таких даних, як середнє значення, стандартні відхилення, кореляції і коваріантність. Всі дані повинні бути доступні для обходу і розрахунку статистик. Це весело, як вправу корисно один раз виконати в Excel, але не дуже корисно на практиці.

Метод градієнтного спуску

При наявності однієї або декількох змінних можна використовувати процес оптимізації значень коефіцієнтів шляхом итеративной мінімізації помилки моделі на учнів даних. Ця операція називається «градієнтний спуск» і працює, починаючи з випадкових значень для кожного коефіцієнта.

За аналогією з методів найменших квадратів - ми шукаємо суму помилок в квадраті розраховується для кожної пари вхідних і вихідних значень. Як масштабного коефіцієнта в градієнтному спуску використовується частота навчання (по-англійськи "learn rate"), а коефіцієнти оновлюються в напрямку

мінімізації помилки. Процес повторюється до тих пір, поки не буде досягнута помилка в квадраті мінімальної суми або неможливо подальше поліпшення.

При використанні цього методу необхідно вибрати параметр швидкості навчання (альфа), який визначає розмір кроку поліпшення, щоб взяти на себе кожну ітерацію процедури. На практиці градієнтний спуск є корисним методом, коли у вас дуже великий датасета або в кількості рядків, або в кількості стовпців, які можуть не вміститися в пам'яті.

Прогнозування за допомогою з лінійної регресії

З огляду на, що уявлення є лінійним рівнянням, зробити прогнози так само просто, як рішення рівняння для певного набору входів.

Розглянемо конкретний приклад. Наприклад, ми прогнозуємо вага людини (y) в залежності від зросту людини (x). Наше уявлення моделі лінійної регресії для цієї проблеми буде:

$$Y = B0 + B1 * X1$$

або

*вага людини = B0 + B1 * висота людини,* де B0 є коефіцієнтом зсуву, а B1–коефіцієнтом для стовпця зросту людини. Ми використовуємо техніку навчання, щоб знайти хороший набір значень коефіцієнтів. Після того, як знайдено, ми можемо підключити різні значення висоти, щоб передбачити вага.

Наприклад, дозволяє використовувати B0 = 0,1 і B1 = 0,5. Давайте підставимо їх і розрахуємо вага (в кілограмах) для людини з ростом 182 сантиметри.

*вага людини = 0,1 + 0,5 * 182*

вага людини = 91.1

Тут можна бачити, що вищезгадане рівняння може бути відображена як лінія в двох вимірах. Коефіцієнт B0 є нашою відправною точкою незалежно від того, який зріст у людини. Ми можемо пробігти через різні висоту людини від 100 до

200 сантиметрів підставивши в рівняння і отримати значення ваги, створюючи нашу лінію.

Підготовка даних до лінійної регресії:

- Лінійні передумови. Лінійна регресія передбачає, що зв'язок між вхідними і вихідним даними є лінійної. Лінійна регресія не підтримує нічого іншого. Це може бути очевидно, але це добре, щоб пам'ятати, коли у вас є багато атрибутів. Може виникнути потреба змінити дані, щоб зробити відносини між ними лінійними (наприклад, логарифмічна перетворення для експоненційної зв'язку).
- Видаліть шум. Лінійна регресія передбачає, що змінні на виході і виведення не є гучними. Розгляньте можливість використання операцій з очищення даних, які дозволяють краще викривати і прояснювати сигнал в даних. Це найбільш важливо для змінної виводу, і, по можливості, необхідно видалити викиди в змінної виводу (у).
- Видаліть колінеарність. Лінійна регресія буде надмірно відповідати вашим даними, коли у вас є сильно корельовані вхідні змінні. Розглянемо розрахунок парних кореляцій для вхідних даних і видалення найбільш корелюється даних.
- Гауссів розподіл. Лінійна регресія зробить більш надійні прогнози, якщо вхідні і вихідні змінні мають гауссів розподіл. Ви можете отримати деяку вигоду за допомогою перетворень на змінних, щоб зробити їх розподіл більш гауссовим.

Нормалізовані вхідні дані: Лінійна регресія часто робить більш надійні прогнози, якщо масштабовані вхідні змінні за допомогою стандартизації або нормалізації.

Навчання без вчителя

Ідеально розмічені і чисті дані дістати нелегко. Тому іноді перед алгоритмом стоїть завдання знайти заздалегідь невідомі відповіді. Ось де потрібно навчання без учителя.

У навчанні без учителя (unsupervised learning) у моделі є набір даних, і немає явних вказівок, що з ним робити. Нейронна мережа намагається самостійно знайти кореляції в даних, витягуючи корисні ознаки і аналізуючи їх.

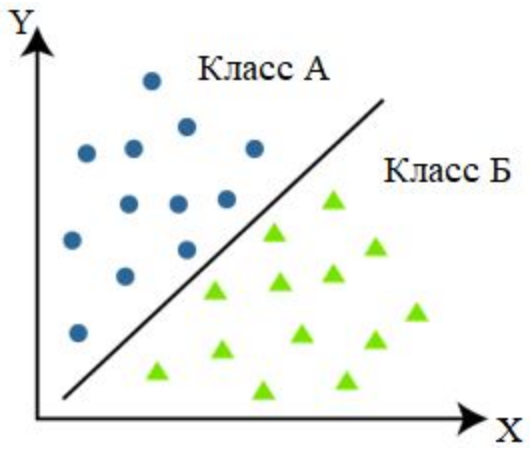


Рис. 1.2. Приклад задачі аналізу і класифікації даних

Залежно від завдання модель систематизує дані по-різному.

- Кластеризація. Навіть без спеціальних знань експерта-орнітолога можна подивитися на колекцію фотографій і розділити їх на групи за видами птахів, спираючись на колір пера, розмір або форму дзьоба. Саме в цьому полягає кластеризація - найбільш поширена завдання для навчання без учителя. Алгоритм підбирає схожі дані, знаходячи спільні ознаки, і групують їх разом.
- Виявлення аномалій. Банки можуть виявити шахрайські операції, виявляючи незвичайні дії в купівельному поводженні клієнтів. Наприклад, підозріло, якщо одна кредитна карта використовується в Каліфорнії і Данії в один і той же день. Схожим чином, навчання без вчителя використовують для знаходження викидів в даних.
- Асоціації. Виберете в онлайн-магазині підгузники, яблучне пюре і дитячу кухоль-непроливайку і сайт порекомендує вам додати нагрудник

і радионяню до замовлення. Це приклад асоціацій: деякі характеристики об'єкта корелюють з іншими ознаками. Розглядаючи пару ключових ознак об'єкта, модель може передбачити інші, з якими існує зв'язок.

- Автоенкодери. Автоенкодери приймають вхідні дані, кодують їх, а потім намагаються відтворити початкові дані з отриманого коду. Не так багато реальних ситуацій, коли використовують простий автоенкодер. Але варто додати шари і можливості розширяться: використовуючи зашумлені і вихідні версії зображень для навчання, автоенкодери можуть видаляти шум з відеоданих, зображень або медичних сканів, щоб підвищити якість даних.

У навчанні без учителя складно обчислити точність алгоритму, тому що в цих відсутні «правильні відповіді» або мітки. Але розмічені дані часто ненадійні або їх занадто дорого отримати. У таких випадках, надаючи моделі свободу дій для пошуку залежностей, можна отримати хороші результати.

Висновок до розділу 1

У даному розділі було розглянуто сферу машинного навчання, проаналізовано основні підходи і методи. Також було описано переваги та проблеми, які виникають у процесі застосування розглянутих підходів у сфері обробки медичних зображень.

На базі виконаного дослідження, описаного у розділі 1, було сформовано план використання тих чи інших методів для задачі обробки медичних зображень шкіри, складено загальний теоретичний опис підходів, які використовуються у роботі, зокрема у дослідженні класифікації зображень злоякісних уражень шкіри.

РОЗДІЛ 2. ОБРОБКА МЕДИЧНИХ ЗОБРАЖЕНЬ НА ОСНОВІ ГЛИБИННИХ НЕЙРОННИХ МЕРЕЖ

2.1. Глибинне навчання

Глибинне навчання – один з видів машинного навчання, що базується на штучних нейронних мережах. [1] Навчання може бути контрольованим, частково контрольованим або ж неконтрольованим.

Частина терміну «глибинне» пов’язана з необмеженою кількістю шарів через які перетворюються дані, що робить можливим практичне застосування та оптимізовану реалізацію, пов’язану з заміною ручної роботи алгоритмами автоматизованого навчання, зберігаючи при цьому, теоретично, універсальність підходу.

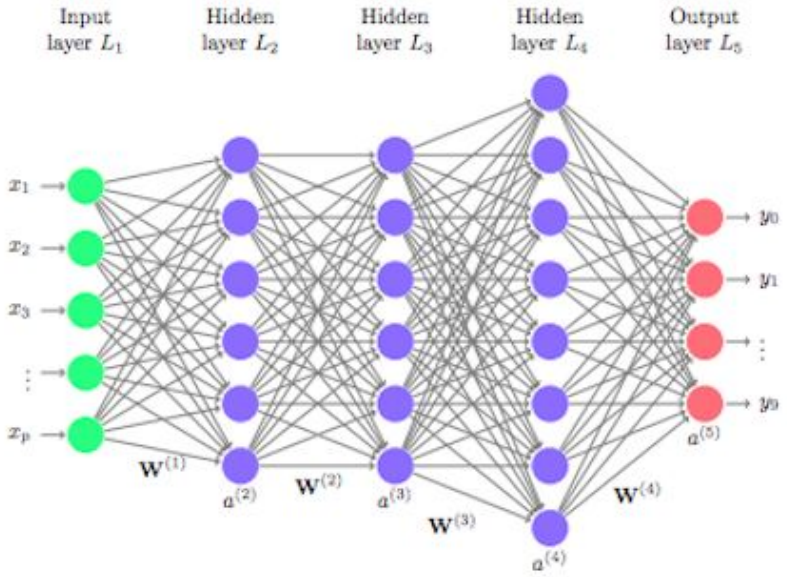


Рис. 2.1. Схематичне зображення моделі глибокого навчання

Глибинне навчання широко застосовується в наступних прикладних задачах:

- Автоматизоване розпізнавання та обробка мовлення
- Рекомендаційні системи
- Фармакологія та медицина
- Біоінформатика
- Розпізнавання та аналіз зображень, в тому числі і медичних

2.2. Згорткова нейронна мережі

CNN (англ. convolutional neural network) [2, 3] – вид архітектури глибоких штучних нейронних мереж, що націлена на ефективне розпізнавання та аналіз візуальних образів.

Виявлення об’єктів на зображенні, розпізнавання обличь і т.д. – деякі області, в яких широко застосовується архітектура CNN.

Основна відмінність від повнозв’язної архітектури полягає в тому що

на відміну від повнозв’язної архітектури де кожен нейрон зв’язаний зі всіма нейронами попереднього шару (при цьому кожний зв’язок має індивідуальний коефіцієнт ваги) в архітектурі згорткової мережі використовується лише обмежена матриця вагів невеликого розміру, яка «переміщується» по всьому оброблюваному шару. Цей процес називається операцією згортки. Її суть полягає в тому, що кожен фрагмент вхідного зображення множиться на кожний елемент матриці згортки, результат підсумовується і зберігається в аналогічному елементі вихідного зображення. Внаслідок цього після кожного зсуву формується сигнал активації нейронна у наступному шарі, який має аналогічну позицію.

Тобто для різних нейронів вихідного шару використовується одна й та сама матриця вагів, так зване «ядро згортки». При цьому такі ядра згортки не створюються програмістом заздалегідь, а генеруються самостійно класичним методом навчання.

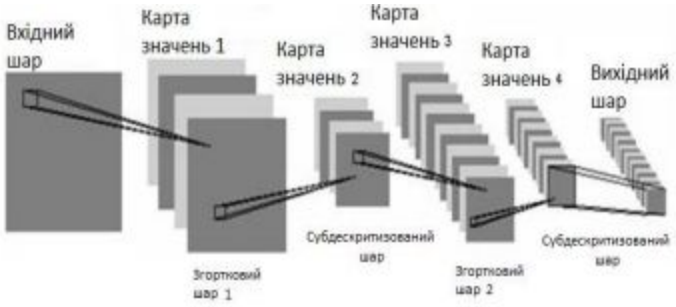


Рис. 2.2. Архітектура згорткової нейронної мережі

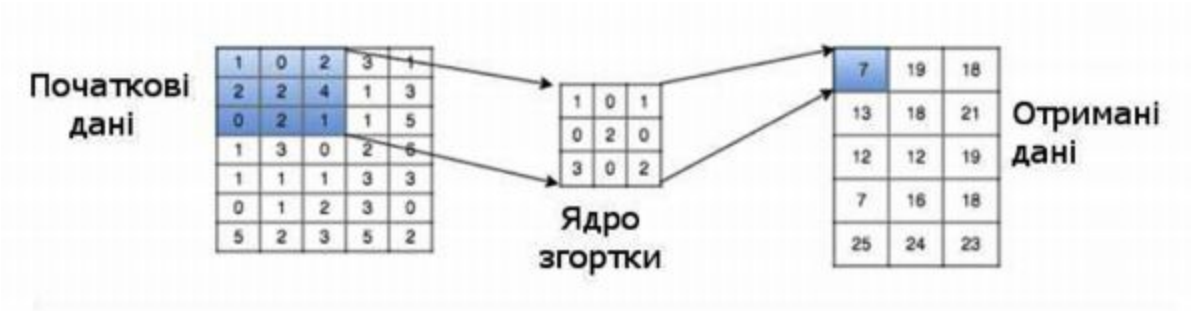


Рис. 2.3. Схема операції згортки

Основні переваги:

- Архітектура має одні з найкращих показників у сфері розпізнавання, аналізу та класифікації зображень, а також їх аналізу
- Порівняно з повнозв’язною архітектурою менш вимоглива до попереднього «ручного» налаштування
- Одна матриця вагів для всього вхідного зображення замість присвоєння кожному елементу (пікселю) вхідного зображення індивідуального значення ваги

Недоліки:

- Багато невідомих змінних мережі: незрозуміло які налаштування необхідні для конкретної задачі або ж обчислювальної потужності. К таким невідомим можна віднести кількість шарів, розмір матриці згортки для кожного з шарів, розмір зсунення матриці при обробці шару і т.д.
-

2.3. Нейронна мережа прямого розповсюдження

FNN (англ. feedforward neural network) [4, 5] – це один з видів архітектури нейронних мереж, у якій сигнали між шарами поширюються лише в одному напрямку, не утворюючи цикл. У даній архітектурі дані переміщуються від вхідних вузлів через приховані (якщо вони присутні) до вихідних вузлів.

Елементи одного й того самого шару ніколи не з’єднані між собою, а приховані шари зазвичай повністю зв’язані. Ці зв’язки не всі рівні: кожне з’єднання може мати різну вагу, яка і кодує інформацію про мережу.

Такі мережі широко використовуються и доволі успішно вирішують деякий спектр задач, такі як прогнозування, кластеризація, розпізнавання.

Нейронна мережа прямого розповсюдження зазвичай навчається методом зворотного поширення помилки, у якому мережа отримує велику кількість вхідних та вихідних даних. Такий процес називається навчанням з учителем і не передбачає створення вихідних даних самостійно, тобто без втручання дослідника.

Метод зворотного поширення помилки – це метод який використовується при оновленні вагів багатошарового перцептрону. Суть методу полягає в розповсюдженні сигналу помилки від вихідних вузлів мережі до її вхідних вузлів, тобто у протилежну від прямого напрямку розповсюдженню сигналів при нормальному режимі роботи мережі.

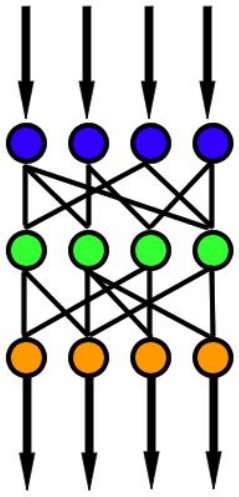


Рис. 2.3. Зображення архітектури мережі прямого розповсюдження

2.4. Рекурентна нейронна мережа

RNN (англ. recurrent neural networks) [6], або ж нейронна мережа зі зворотним зв’язком – це вид архітектури нейронних мереж, у якій вихідні сигнали нейронів використовуються в якості зворотного зв’язку для нейронів попереднього шару. По суті являється розширенням мережі прямого розповсюдження.

Дана архітектура використовується для динамічної обробки даних, тобто націлена на ефективне використання у таких областях, де кінцевий результат залежить від попередніх обчислень(обробка тексту, розпізнавання мовлення, послідовностей ДНК тощо).

Архітектура рекурентної нейронної мережі відрізняється від архітектури мережі прямого розповсюдження тим, що «дивиться» не тільки на поточні вхідні дані, а й на попередні. Тобто, архітектура працює в послідовності:

Вхідні дані + Попередній прихований шар → Поточний прихований шар →

→ Вихідні дані

замість стандартного

Вхідні дані → Прихований шар → Вихідні дані

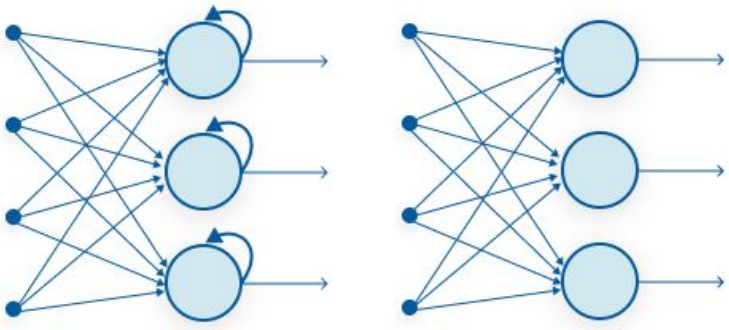


Рис. 2.4. Відмінність рекурентної архітектури від архітектури мережі прямого розповсюдження

На відміну від нейронних мереж прямого поширення, рекурентні мережі здатні використовувати внутрішню пам’ять для обробки послідовностей вхідних даних.

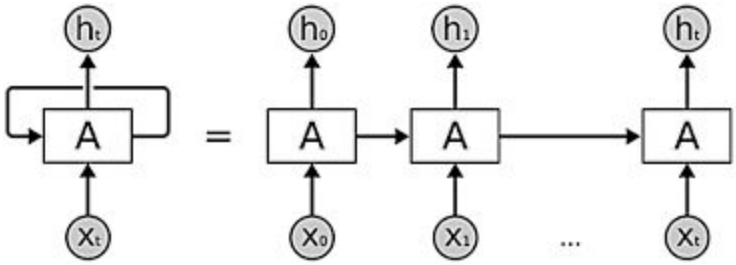


Рис. 2.5. Зображення архітектури рекурентної нейронної мережі

На рис. 2.5 схематично зображено архітектуру рекурентної нейронної мережі, де блок A – це проста нейронна мережа прямого поширення, описана вище.

Права частина рівняння на рисунку зображує мережу для кожного часового кроку, тобто при t=0 вхідний вузол x_0 переходить в мережу для утворення вихідного значення h_0, що на наступному часовому кроці являється x_1, при цьому присутній додатковий вхід від попереднього часового кроку від блоку A.

Таким чином, дана архітектура не тільки дивиться на поточний вхід, але також має контекст із попередніх входів.

Основні переваги:

- RNN здатна фіксувати послідовну інформацію, шо присутня у вхідних даних, тобто залежність між словами у тексті, роблячи пронози
- RNN розподіляють параметри на різних часових кроках. Такий механізм призводить до меншої кількості параметрів для навчання та знижує обчислювальні витрати, в основному за рахунок того що вагові матриці спільні для усіх часових кроків.

Недоліки:

- Глибокі RNN з великою кількістю часових кроків чутливі до проблеми зникаючого і зростаючого градієнта, яка є спільною для багатьох типів нейронних мереж

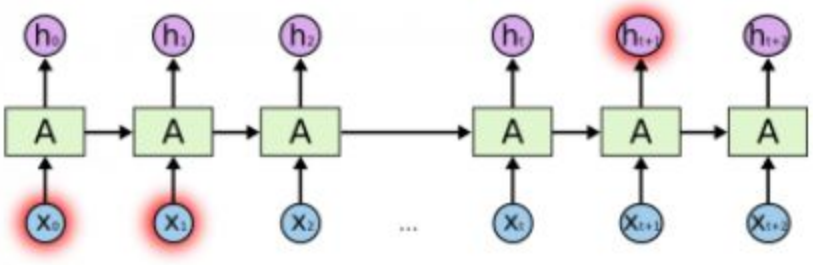


Рис. 2.6. Зображення проблеми зникаючого градієнта

Рис. 2.6. зображує як обчислений на останньому часовому кроці градієнт зникає по мірі досягнення початкового часового кроку.

2.5. Повнозв'язна нейронна мережа

Архітектура FCN (англ. Fully Convolutional Networks) [7, 8] побудована виключно із локально пов'язаних шарів, таких як згортка. У такій архітектурі кожний нейрон передає свій вихідний результат всім іншим нейронам, у тому числі і собі. Вихідними даними мережі можуть бути всі або деякі вихідні дані нейронів після декількох тактів роботи нейронної мережі.

В даній архітектурі шари містять множину нейронів з єдиними вхідними сигналами. Кількість нейронів у кожному шарі може бути будь яким и ніяк заздалегіть не пов'язане з кількістю нейронів в інших шарах. Загалом мережа складається з N шарів, пронумерованих зліва направо. Зовнішні вхідні сигнали подаються на входи нейронів першого шару, а виходами мережі є вихідні сигнали останнього шару. Вхід нейронної мережі можна розглядати як вихід вхідного шару нейронів, які служать лише в якості точок розподілення. Також сумування і перетворення сигналів в даній архітектурі не здійснюється.

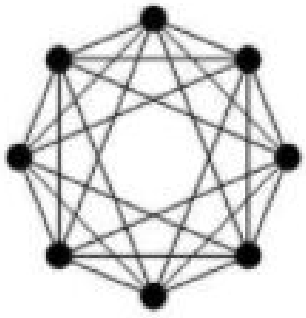


Рис. 2.7. Зображення архітектури повнозв'язної нейронної мережі

Табл. 2.1

Порівняння основних типів нейронних мереж (FNN, RNN, CNN)

	FNN	RNN	CNN

Тип даних(найбільш підходящий)	-	Послідовні дані	Зображення
Циклічні зв'язки між вузлами	Ні	Так	Ні
Обмін параметрами	Ні	Так	Так
Просторові відносини між вузлами	Ні	Ні	Так
Зникнення градієнта	Так	Так	Так

2.6. Застосування глибинного навчання для обробки та аналізу медичних зображень на базі CNN [31]

Застосування технології глибинного навчання у сфері медичного аналізу [10] та візуалізації [12, 13] у перспективі може стати найбільш успішною технологією, яку радіологія застосовувала з моменту появи цифрових зображень.

Більшість наукових дослідників вважають, що протягом наступних 10-15 років прикладні технології що базуються на основі Depp Learning не тільки візьмуть на себе більшість ручної роботи з діагностування захворювань, але й допоможуть попередити хворобу, створити оптимальний алгоритм лікування.

Такі галузі медицини як офтальмологія, рентгенологія та онкологія зазнали революційного прогресу з моменту впровадження технологій машинного навчання, в особливості саме Depp Learning.

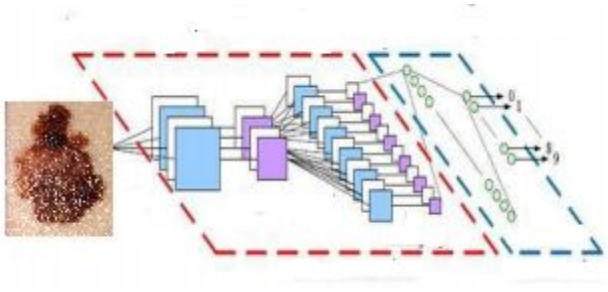


Рис.2.8. Архітектура згорткової нейронна мережа на прикладі знімка шкіри

Застосування технології глибинного навчання у сфері охорони здоров'я охоплює доволі широкий спектр проблем, такі як аналіз раку, моніторинг захворювань та інше.

Різні джерела даних, які використовуються для рішення даних проблем – рентгенологічна візуалізація (рентген, КТ, МРТ), аналіз патологій та відкриті геномні послідовності надали велику кількість даних для аналізу у розпорядження медичним працівникам різних направлень.

Як частина технології глибинного навчання згорткові нейронні мережі CNN використовуються в сфері комп'ютерного зору. Дана архітектура показала себе найефективнішою порівняно з типовими технологіями.

Основна роль згорткової нейронної мережі заключається в ідентифікації ліній та країв. Кожен прихований шар CNN складається із згорткових шарів, які містять вхідний масив даних із параметризованими за вагою ядрами згортки. Ядра генерують кілька образних зображень. Таким чином завдяки подібній архітектурі вдалося досягти успіху в багатьох завданнях комп'ютерного зору, таких як сегментація та класифікація.

Між згортковими шарами формуються карти особливостей (англ. feature map). Об'єднаний шар мережі передає максимальне або середнє значення і таким чином зменшує розмір карти ознак.

Такий алгоритм дозволяє захоплювати характерні особливості вхідного зображення враховуючи положення і форми зображення. CNN архітектура складається з багаторазового повторення таких шарів. Для завдання класифікації

повнозв’язні шари прикріплюються до згорткової мережі та видають остаточний результат.

Під час тренування втрати оцінюються шляхом порівняння промаркованого і прогнозованого значення. Але в задачі сегментації згорткові шари додаються у кінці процесу поєднання шарів з ціллю відновлення розміру вхідного зображення. Таким чином, втрати при тренування оцінюються шляхом порівняння промаркованого зображення маски і відновленого через CNN вихідного зображення.

Оскільки архітектура згорткової нейронної мережі складається з багатьох рівнів, кількість параметрів для тренування в переспективі може набувати дуже великого значення. Тобто для забезпечення необхідної точності необхідна дуже велика кількість даних, яка залежить від цілей задачі, особливостей та характеристик вхідного зображення. Проблема збору даних зазвичай актуальна, особливо в ситуації коли є необхідність в заздалегідь промаркованих даних. Рішення такої проблеми зазвичай являє собою збільшення кількості даних шляхом генерації нових зображень з наявних даних, використовуючи методи перетворення зображень, масштабування, відзеркалення та інші.

2.7. Основні моделі глибинного навчання

DenseNet121.

Densely Connected Convolutional Networks [23] – щільно пов’язані нейронні згорткові мережі – архітектура нейронних мереж, у яких зв’язки між прихованими шарами, що близькі до вхідного і вихідного шарів. Архітектура DenseNet поєднує кожний шар з кожним іншим шаром між якими є прямий зв’язок.

На відміну від традиційних архітектур CNN з N рівнями і які мають N зв’язків – по одному між кожним рівнем і його наступним рівнем – архітектура DenseNet має N(N+1)/2 з’єднань. Для кожного шару у якості вхідних даних використовуються карти характеристик усіх попередніх шарів.

Архітектура DenseNet має наступні переваги:

- Вони найменш схильні до зникнення градієнта
- Посилюють розмноження ознак
- Використовують метод повторення функцій, чим значно скорочують кількість параметрів

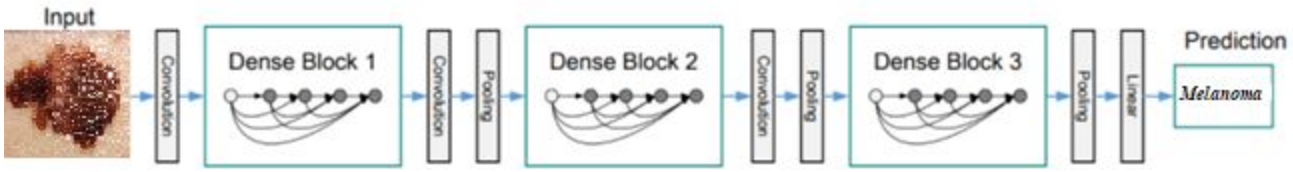


Рис 2.9. Зображення роботи архітектури DenseNet

MobileNet.

Архітектура згорткової мережі типу MobileNet [27, 28] складається з стандартного згорткового шару на початку та тринадцяти блоків з поступово зростаючою кількістю фільтрів. Дана архітектура складається з одної згортки 3x3 на початку та тринадцяти блоків з поступово зростаючою кількістю фільтрів.

Особливістью архітектури є відсутність max-pooling шарів, замість яких використовується згортка з праметром stride = 2 з ціллю зменшення розмірності. Двома гіперпараметрамі архітектури MobileNet є множник ширини і множник глибини. Множник ширини відповідає за кількість каналів в кожному шарі, а множник глибини відповідає за розміри вхідних тензорів.

Обидва параметра дозволяють варіювати розміри мережі: зменшуючи їх, зменшується точність розпізнавання, натомість зростає швидкість роботи і зменшується споживання пам'ять, що і робить дану архітектуру придатную до використання на мобільних пристроях.

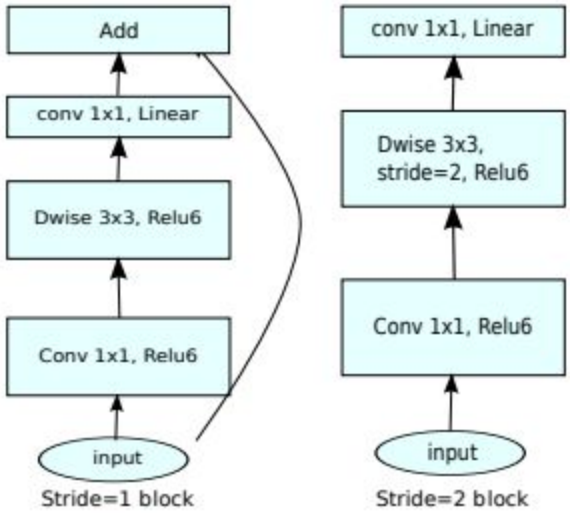


Рис. 2.10. Схема архітектури MobileNet

InceptionNet. [24, 25]

Архітектура InceptionNetV1[26] вирішує наступні проблеми:

- Основі частини зображення можуть мати велику різниці у розмірі.

Наприклад, зображення меланоми може бути різним у плані масштабу об’єкта і фона (меланома і шкіра). Через таку різницю у положенні об’єкта цільової інформації вибір коректного ядра згортки стає дещо важчим. Більші розміри ядра більш сприятливі для глобальніших даних, натомість менші розміри ядра сприятливіші для даних, які розповсюдженні більш локально.

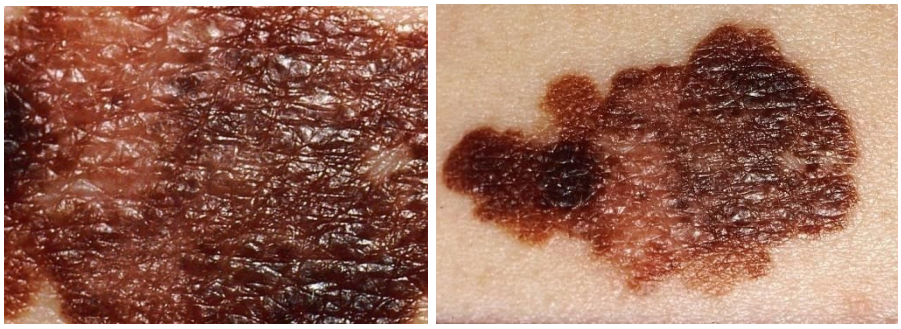


Рис.2.11. Приклади різних розмірів цільових об’єктів на зображенні

- Моделі Depp Learning схильні до перенавчання. Також проблемою є передача оновлень градієнта по всій архітектурі мережі.
- Накладення об’ємних операцій згортки потребує великих обчислювальних ресурсів.

Рішення даних проблем полягає у використанні фільтрів різних розмірів на одному й тому самому шарі. Тобто, архітектура мережі збільшується в ширину, але не в глибину.

Нижче наведена приклад початкового модулю моделі, який виконує згортку на вході з 3 різними розмірами фільтрів: 1x1, 3x3, 5x5. В додаток до цього використовується максимальне поєднання: вихідні результати об’єднуються та відправляються до наступного шару.

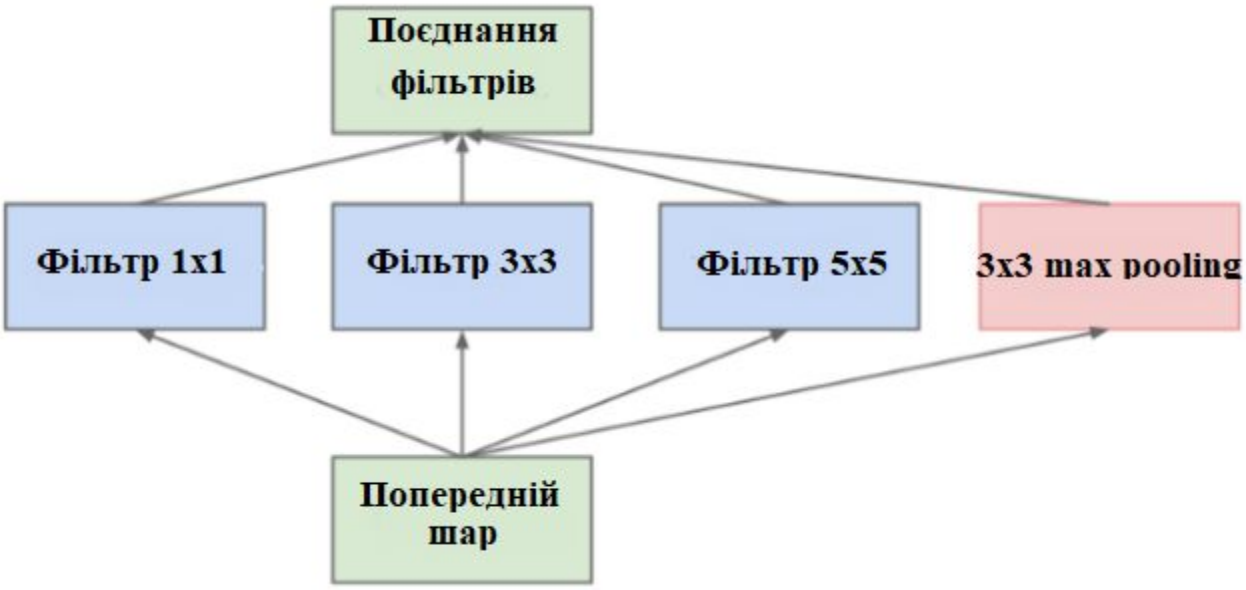


Рис.2.12. Схема застосування методу поєднання фільтрів різного розміру

InceptionV2 та InceptionV3 [26] були представлені одночасно. В порівнянні з попередньою архітектурою мають наступний список покращень:

- Зменшення репрезентативних вузьких місць у моделі. Суть полягає у тому, що нейронні моделі мають кращу ефективність при умові того що згортки не змінюють розмір вхідних даних. Занадто радикальне збільшення або ж зменшення розмірів зображень може викликати втрату важливої інформації, те саме «репрезентативно вузьке місце»
- Використання методів факторизації: згортки робляться більш ефективними з точки зору обчислювальної складності.

Рішення полягають у наступних покращеннях:

- Розбір згортки 5x5 на дві операції згортки 3x3 з ціллю підвищення швидкості обчислень. На перший погляд це здається нелогічним, адже згортка 5x5 у 2,78 різи «дорожче» згортки 3x3. Але фактично об’єднання двох згорток 3x3 призводить до підвищення ефективності.

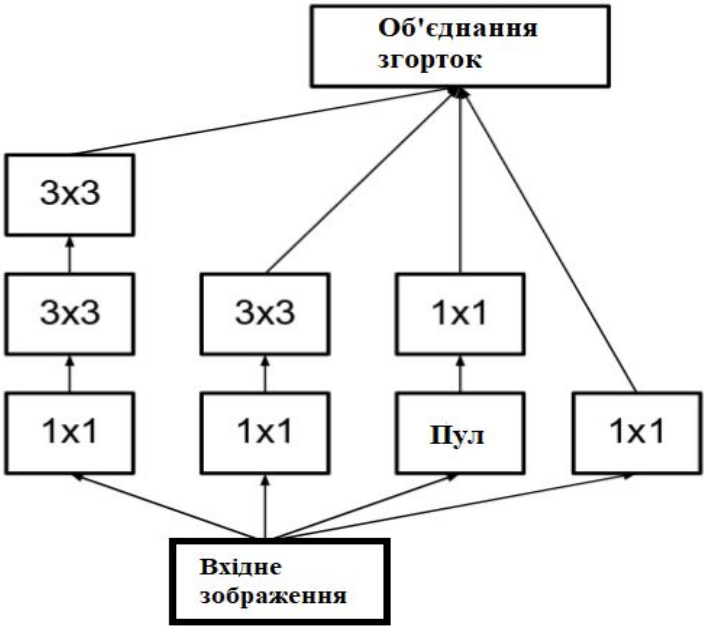


Рис.2.13. Схема об’єднання згорток: ліва згортка 5x5 представлена як об’єднання згорток 3x3

- Факторизація фільтрів. Наприклад, згортка 3x3 еквівалентна згортці 1x3 на вході, потім згортці 3x1 на виході. Таким метод на 33% дешевший ніж виконання згортки 3x3 один раз.

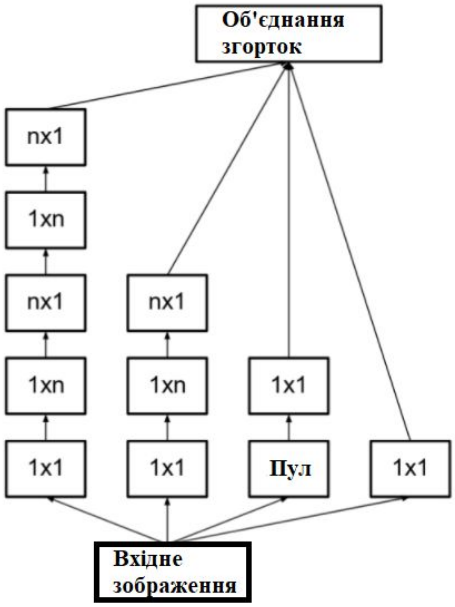


Рис.2.14. Схема факторизації фільтрів на прикладі розкладання згортки 5x5 на дві згортки 3x3, які потім представлені як згортки 1x3 та 3x1

- Блоки фільтрів було розширено з ціллю видалення репрезентативно вузького місця. Якщо замість цього виконати поглиблення це може призвести до занадто великого зменшення розмірів і, як наслідок, втрата важливої інформації.

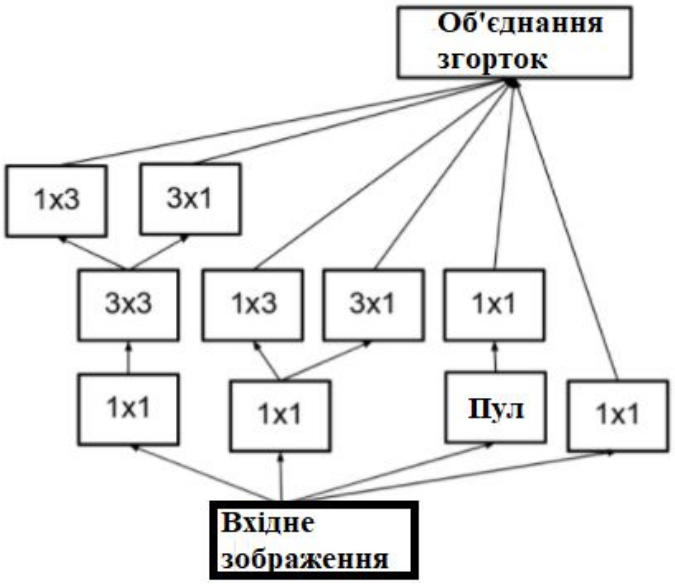


Рис.2.15. Розширення початкового модуля фільтрів

У архітектуру InceptionV3 було внесено наступні покращення:

- Оптимізатор RMSPro
- Факторизація згорток 7x7
- Допоміжні класифікатори
- Label Smoothin, так зване згладжування міток: регулюючий компонент, що додається до формули втрат і не дозволяє моделі стати занадто впевненою в вставленні до об’єктів певного класу, тобто передбачує перетренування моделі.

ResNet101, ResNet101V2

ResNet (Residual Network – залишкова мережа) [29, 30] – згорткова нейронна мережа, що містить 101 шар глибини та заснована на підході з’єднань швидкого доступу. Мережа заточена під тренування за допомогою відкритох бази даних ImageNet. [33, 34]

З’єднання моделі пропускають один або декалька шарів і виконують зіставлення ідентифікаторів, після чого вихідні дані додаються до вихідних даних складених шарів.

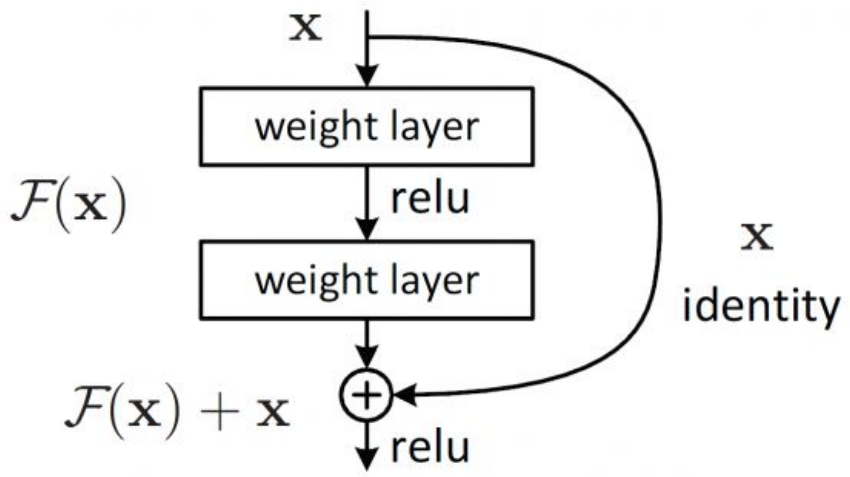


Рис.2.16. Схема архітектури ResNet

Дана архітектура вирішує наступні проблеми:

- ResNet сприятлива до оптимізації: моделі, суть роботи яких полягає у простому складанні шарів дають більше значення похибки за умови зростання глибини моделі.

- ResNet дуже сприятлива до збільшення точності завдяки збільшенню глибини моделі, що є достатньо рідким явищем серед інших архітектур.

InceptionResNetV2. [24, 25]

Дана модель архітектури CNN базується на попередній архітектурі InceptionNet і навчається на даних з загальнодоступної бази даних ImageNet.

Дана нейронна модель складається з 164 шарів та може класифікувати зображення по 1000 різним класам об’єктів. Таким чином модель натренована на розпізнавання та класифікацію широкого діапазону зображень. Модель має розміри вхідних зображень 299x299.

VGG16. [31]

VGG16 – різновид згорткових мереж, що базується на моделі AlexNet з деякими модифікаціями. Модель тренується на датасеті ImageNet.

На вхідний шар conv1 подаються зображення у форматі RGB розмірністю 224x224, після чого усі зображення проходять через набір згорткових шарів, які використовують фільтри з полем 3x3, який є мінімально можливим для визначення положення об’єкта на зображенні.

В конфігурації мережі використовується згортковий фільтр 1x1, що представлений як лінійна трансформація вхідних каналів. Згортковий крок фіксується на значенні 1 піксель. Просторове доповнення (padding) входу згорткового шару підбирається таким чином, щоб просторова роздільна здатність зберігалася після згортки, тобто доповнення дорівнює 1 для 3x3 згорткових шарів. Просторовий пулінг здійснюється за допомогою п'яти max-pooling шарів, які слідують за одним із згорткових шарів (не всі згорткові шари мають наступні max-pooling). Операція max-pooling виконується на фільтрі розміру 2x2 пікселів з кроком 2.

Після набору згорткових шарів (який має різну глибину в різних архітектурах) йдуть три повнозв'язні шари: перші два мають по 4096 каналів, третій - 1000 каналів. Останнім йде soft-max шар. Конфігурація повнозв'язних верств одна і та ж у всіх нейросетях.

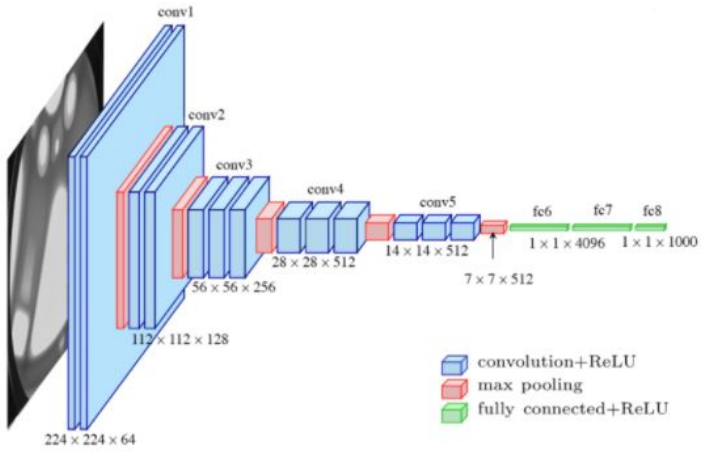


Рис.2.17. Архітектура моделі VGG16

Дана модель має декалька недоліків:

- Повільна швидкість тренування
- Завелика розмірність архітектури

Через глибину та кількість повнозв'язних вузлів модель важить больше половини гігабайта. Це створює деякі незручності у процесі розгортання моделі.

Xception [32]

Ідея моделі полягає у тому, шо замість вибору розміру ядра можна обрати декілька варіантів та використати їх усі одночасно, з слідуючою після конкатенацією результату. Такий підхід збільшує необхідну для обчислення активацій одного шару кількість операцій. Тому було прийнято перед кожним згортковим блоком робити згортку 1x1, знижуючи розмір сигналу що подається на вхід згорткам з більшими розмірами ядра.

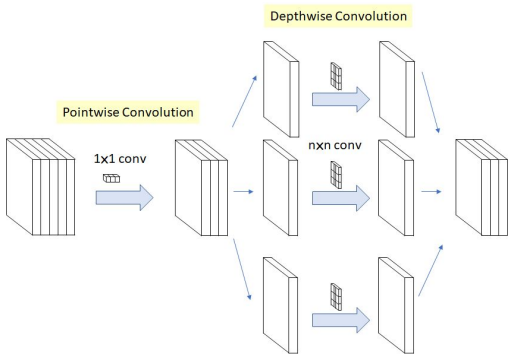


Рис.2.18. Архітектура моделі Xception

Далі виконується наступна послідовність дій:

1. Згортання базового тензора 1x1, подібно до згортки у блоці Iception. Дана операція має назву pointwise convolution.
2. Згортання кожного каналу окремо згорткою 3x3 (при цьому розмір, на відміну від стандартного згорткового шару, не зміниться, адже згортаються не усі канали одночасно). Дана операція має назву depthwise spatial convolution.

Висновок до розділу 2

У даному розділі розглянуто основні архітектури нейронних мереж, що мають перспективу отримати хороші результати при умові їх імплементування у сферу обробки та аналізу медичних зображень.

На базі характеризації моделей було сформовано план та подальші кроки дослідження обробки медичних зображень та класифікації злоякісних уражень на шкірі людини, а саме: обрано найбільш перспективні, у даній сфері, архітектури нейронних мереж для тренування та тестування, складено план їх тренування та тестування, план отримання даних з результатів дослідження.

РОЗДІЛ 3. МЕТОД ОБРОБКИ МЕДИЧНИХ ЗОБРАЖЕНЬ ШКІРИ ЛЮДИНИ

3.1. Проблема визначення стану людини за медичними зображеннями шкіри [14]

Рак шкіри являється найпоширенішим різновидом ракових захворювань. В особливості, такий вид ураження шкіри як меланома за статистикою являється причиною біля 75% летальних випадків, спричинених раком шкіри. За оцінками ACS (American Cancer Society) у 2020 році було зафіксовано біля 100 тис. випадків даної патології у пацієнтів.

Сьогодні робота дерматологів у даному направленні полягає в ручному дослідження кожної родинки пацієнтів з ціллю виявлення підозрілих уражень, які мають перспективу бути злоякісними. На жаль, сучасні підходи до впровадження технологій штучного інтелекту не враховують дані нюанси клінічного підходу. Спеціалісти у сфері онкології шкіри мали б шанс підвищити точно діагностики, як би алгоритми машинного навчання враховували «контекстні» зображення одного й того самого пацієнта в рамках одного дослідження. У разі успішної імплементації даного підходу класифікатори можуть бути точнішими, а також буде можливість до запровадження технології для підтримки роботи дерматологічних відділень.

3.2. Опис методу класифікації меланоми

У даному розділі розглядається процесс класифікації меланоми на наборі зображень ураженої шкіри. Зокрема, використовуються зображення одного й того самого пацієнта з ціллю визначити які саме з них мають перспективу виявитись злоякісними. Набір зображень представлений у формати DICOM. Даний формат набув широкого розповсюдження у сфері досліджень даних у медичній візуалізації. У роботі використовуються датасети, представлені SIIC-ISIC Melanoma Classification Challenge [16, 17] за 2018-2020 роки.

Використання контекстної інформації в рамках дослідження шкіри одного пацієнта може допомогти в розробці інструментів аналізу зображень, які мають можливість автоматизувати деякі процеси клінічних спеціалістів у сфері дерматалогії, в особливості онкології шкіри.

Представлені результати оцінюються у розділі 4 за декількома показниками, зокрема такими як час виконання, час виконання враховуючи TTA, розмір моделі з ціллю дослідження можливості імплементації у ті чи інші підходи, показник AUC, показник AUC з урахуванням TTA.

3.3. Налаштування

Для того щоб отримати правильну перехресну перевірку з, загалом, значущою оцінкою cross-validation необхідно обрати однакові розміри картинок в датасетах та архітектуру моделі EfficientNet для кожного набору даних.

- DEVICE – використовуємий для навчання нейронної моделі апаратний процесор, в даному випадку GPU або TPU
- SEED – означає стан використання генератора псевдовипадкових чисел. Якщо використовувати одне й те саме значення SEED, то алгоритм буде видавати однаковий набір результатів кожного разу. Різне значення SEED дозволяє продукувати різне потрійне kfold розшарування.
- FOLDS – кількість наборів (як правило, послідовних) записів набору даних
- IMG_SIZES – розмір зображень в кожному наборі FOLD
- INC2019 – включає у себе нову частину даних аналогічного змагання 2019 року.
- INC2018 – друга частина даних аналогічного змагання 2019 року, що складається з даних 2017 та 2018 років
- BATCH_SIZES – розмір групи даних для кожного набору даних FOLD. Для найкращої швидкості використовується максимально велике

значення BATCH_SIZE, яке здатна обробити апаратна частина GPU, або ж TPU

- EPOCHS – максимальна кількість епох. У кожному наборі записів зберігається та використовується найкраща модель епохи.
- EFF_NETS – варіант архітектури моделі сімейства EfficientNets, яку використовує кожний набір даних FOLD. У коді номер додається до змінної, тобто якщо номер дорівнює 0, то буде використовуватись архітектура EfficientNetB0.
- WGTS – це вага під час складання наборів даних для прогнозування тестового набору. Для хорошої узгодженності рекомендується використовувати однакову вагу.
- TTA – збільшення даних тестування. Кожне тестове зображення випадково доповнюється і передбачається час TTA, в результаті використовується середнє передбачення. TTA також застосовується для OOF під час перевірки.

Перехресна перевірка

Cross-validation – це процес перевірки, що використовується для оцінки моделей машинного навчання на обмеженій виборці даних. У метода є параметр k, який означає кількість груп наборів даних, на яку повинен бути розбитий вхідний датасет. Тому метод носить назву k-fold cross validation. Якщо обране чітке значення k, воно використовується замість змінної k у посиланні на модель, тобто при k=10 метод стає 10-кратною перехресною перевіркою.

Перехресна перевірка зазвичай використовується у прикладному машинному навчанні для оцінки ефективності нейронної моделі на невидимих даних.Для цього використовується обмежена вибірка з ціллю оцінки того, як модель буде працювати коли буде використовуватись для прогнозування даних, які не використовуються під час тренування моделі.

Процес поділяється на наступні етапи:

- Встановлення випадкового порядку даних у датасеті
- Розбиття датасету на k груп
- Для кожної k-ї групи:
    1. Позначення однієї групи як тестової, тобто з тестовим набором даних
    2. Всі інші групи позначаються як групи даних для навчання нейронної моделі
    3. Тренування моделі на тренувальному наборі даних та оцінка результатів на тестовому наборі
    4. Збереження результатів та оцінки моделі

Даний підхід включає випадкове розбиття набору на k групп FOLD приблизно однакового об’єму. Перше «згинання» розглядається як набір даних для перевірки та оцінки, тому метод підходить для остаточних k-1 наборів даних FOLD.

Також важливо що будь-яка підготовка даних перед тренуванням моделі відбувалась на назначеному наборі даних всередині циклу, а не на більш широкому наборі даних. Також це відноситься до будь-якого налаштування параметрів (Data augmentation). Якщо цього не притриматись це може призвести до хибної оптимістичної оцінки ефективності моделі.

Значення k необхідно уважно підбирати для вибіркового набору даних. Неправильно підібране значення k може призвести до неправильного уявлення щодо ефективності моделі.

Щоб мати можливість коректно підібрати значення k необхідно притримуватись наступних пунктів:

- Значення k повинно бути підібране таким чином, щоб кожна тренувальна і тестова група вибіркових наборів даних була достатньо великою, щоб бути статично репрезентативною для об’ємнішого набору даних.

- k = 10. Значення k, яке було підібрано шляхом експеременту і яке зазвичай призводить до оцінки ефективності моделі із низьким зміщенням та нормальним відхиленням.
- k = n, де n – розмір набору даних. Використовується для того, щоб дати кожному тестовому набору даних можливість використовуватися в утриманому наборі даних. Даний підхід має назву «перехресна перевірка з виключенням по одному»

Зазвичай рекомендовано схилятися до вибору k = 5 або ж k = 10, однак чіткого правила не існує. У міру збільшення k різниця у розмірах між тренувальними наборами даних та підмножинами повторної вибірки зменшується. Такі значення k дозволяють знайти компроміс між зміщенням і дисперсією, оскільки такі значення дають оцінку частоти помилок тесту, яким не притаманно ні занадто високе зміщення, ні висока дисперсія.

Якщо обрано таке k, що не розділяю вибірковий набір даних на еквівалентні частини, то одна з вибірок буде містити залишкові приклади. Рекомендовано розбити датасет на k групп з еквівалентною кількість вибіркових даних, для того щоб всі вибірки оцінок ефективності моделі були еквівалентними.

3.4. Апаратна конфігурація.

Graphics processing unit (GPU) – це електронне обладнання, що здатне перетворювати графічний образ, що знаходиться у пам’яті комп’ютера у таку форму, що придатне для подальшого виводу на екран, тобто відповідає за обробку та формування графічного образу.

Сучасні GPU не обмежені звичайним виводом зображення або образу на екран, вони мають вбудований графічний процесор, який здатен виробляти додаткову обробку, беручи на себе задачу CPU. Так, GPU будучи розробленими для прискорення обробки графіки здатні сильно прискорити обчислювальні процеси для глибинного навчання. Вони є невід’ємною частиною сучасної сфери

машинного навчання, а нові GPU часто розробляються спеціально із розрахунку на використання їх для сфери машинного, і в особливості глибокого, навчання.

Принцип обчислень за допомогою GPU

GPU складаються з спеціальних обчислювальних ядер, які використовуються для прискорення обчислювальних процесів. Спочатку вони були призначені виключно для обробки зображень та різного роду графічних та візуальних даних.

Однак сьогоднішня тенденція полягає у впровадженні для прискорення процесів обчислення, таких як глибинне навчання. Така тенденція пов'язана з тим, що GPU здатні ефективно використовувати паралельні обчислення для масових розподілених обчислень. Головною перевагою GPU здатність до паралельних обчислень та одночасна обробка частин одного цілого набору даних. Для реалізації паралельних обчислень використовуються чотири основні типи архітектури:

- Окрема інструкція, окремі дані (SISD - Single instruction, single data)
- Одна інструкція, декілька наборів даних (SIMD - Single instruction, multiple data)
- Багато інструкцій, окремі дані (MISD - Multiple instructions, single data)
- Багато інструкцій, декілька наборів даних (MIMD - Multiple instructions, multiple data)

Більшість процесорів мають багатоядерну архітектуру MIMD. У свою чергу GPU використовують архітектуру SIMD, що робить GPU найкращим варіантом вибору для підходів глибинного навчання, яким необхідно виконання однакових процесів для великої кількості різних даних.

Такий підхід дозволяє розподіляти процеси машинного навчання і здатний значно прискорити виконання операцій. За допомогою архітектури GPU є можливість накопичувати багато ядер, які будуть використовувати набагато

менше ресурсів не знижуючи при цьому ефективність та обчислювальні можливості машини загалом.

При використанні архітектури GPU у сфері глибинного навчання необхідно спиратися на декілька факторів, обираючи робоче рішення:

- Пропускна здатність пам'яті – використання графічних процесорів здатне забезпечити ту пропускну здатність, яка необхідна для розміщення великих наборів даних. Це пов'язано з тим, що графічні процесори включають у себе виділену відеопам'ять (VRAM), шо дає можливість зберегти пам'ять центрального процесора для інших задач.
- Розмір набору даних – графічні процесори масштабуються простіше ніж звичайні, тому вони мають можливість оброблять великі масиви даних. Відповідно чим більші набори даних, тим більшу користь можна отримати від GPU
- Оптимізація. Іншою стороною медалі є той факт, шо оптимізація довго виконуваних окремих задач буває більш складною, ніж у ситуацією зі звичайними процесорами.

GPU – це відносно дорогі технологічні ресурси, які, до того ж, необхідно оптимізувати. Однак часто складається ситуація, коли дослідники або розробники використовують лише до 30% доступних їм ресурсів GPU. Зазвичай це пов'язано з неефективною оптимізацією та розподіленням ресурсів. Ефективність коректного використання ресурсів можна оцінити за наступними показниками:

- Метрики. Метрики використання GPU вимірюються у процентах використаного часу, протягом якого ядра GPU знаходяться в робочому стані. Ці показники можна використовувати для визначення вимог до можливостей GPU. Доступ до даної метрики можна отримати за допомогою інтерфейсу управління системою NVIDIA-smi, у UNIX-like подібних системах метрики можна отримати за допомогою команди «!nvidia-smi»

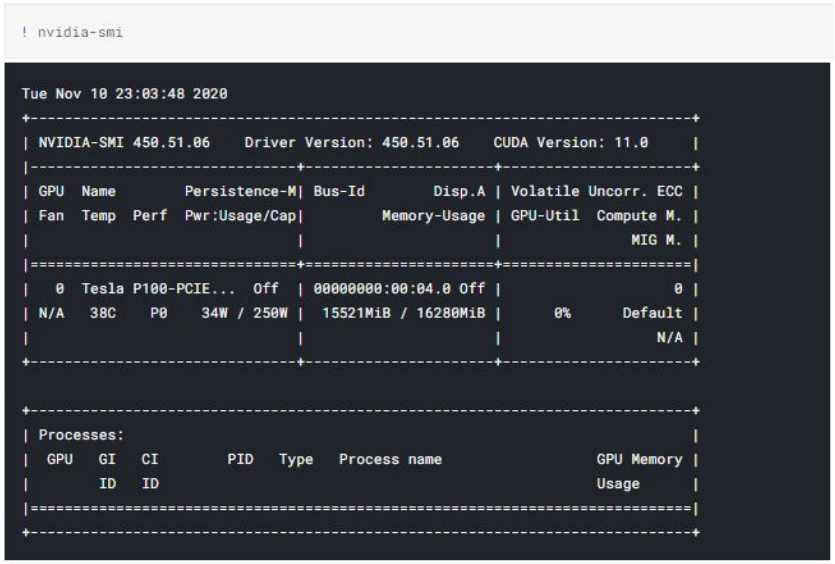


Рис. 3.1. Результат команди «!nvidia-smi»

- Доступ до пам’яті GPU та її використання. Метрики доступу та використання GPU вимірюють відсоток часу, протягом якого контролер пам’яті GPU знаходиться в робочому стані, що включає такі операції як читання та запис. Ці метрики можна використовувати щоб оптимізувати розмір пакету для машинного навчання та оцінити ефективність використовуваної моделі машинного навчання. Доступ до даної метрики можна отримати за допомогою інтерфейсу управління системою NVIDIA-smi, у UNIX-like подібних системах метрики можна отримати за допомогою команди «!nvidia-smi»
- Енергоспоживання і температура. Показники енергоспоживання і температури дозволять виміряти наскільки сильно працює система, а також можуть допомогти спрогнозувати та контролювати енергоспоживання системи. Ці метрики зазвичай вимірюються в блоку живлення і включають ресурси що використовуються обчислювальними блоками та блоками пам’яті. Ці показники важливі, тому що перевищена температура може викликати процес теплового регулювання, що сповільнює обчислювальні процеси.
- Час рішення задачі. Час до виконання задачі – це цілісний показник, який дозволяє визначити бажаний рівень точності і дізнатись скільки часу

буде необхідно для навчання моделі с необхідним рівнем точності. Цей час буде різним для різних GPU в залежності від моделі, стратегії розповсюдження та набору даних, який використовується. Обравши конфігурацію графічного процесора є можливість використовувати часові метрики для виміру рішення для налаштування розмірів пакетів або використовувати оптимізацію зі змішаною точністю для підвищення продуктивності.

NVIDIA Deep Learning SDK

NVIDIA CUDA-X AI – це набір розробки програмного забезпечення, що призначений для дослідників та розробників, які займаються розробкою моделей глибинного навчання. У наборі використовуються високоефективні GPU.

NVIDIA CUDA-X AI розроблений в основному для задач комп'ютерного зору. Його можна використовувати для прискорення роботи вже існуючих фреймворків та створення нових архітектур моделей. Бібліотека CUDA-X AI надає інструменти програмування, які дозволяють створювати моделі глибинного навчання на робочих стендах, після чого створені моделі можна розвернути у центрах обробки даних або на інших пристроях.

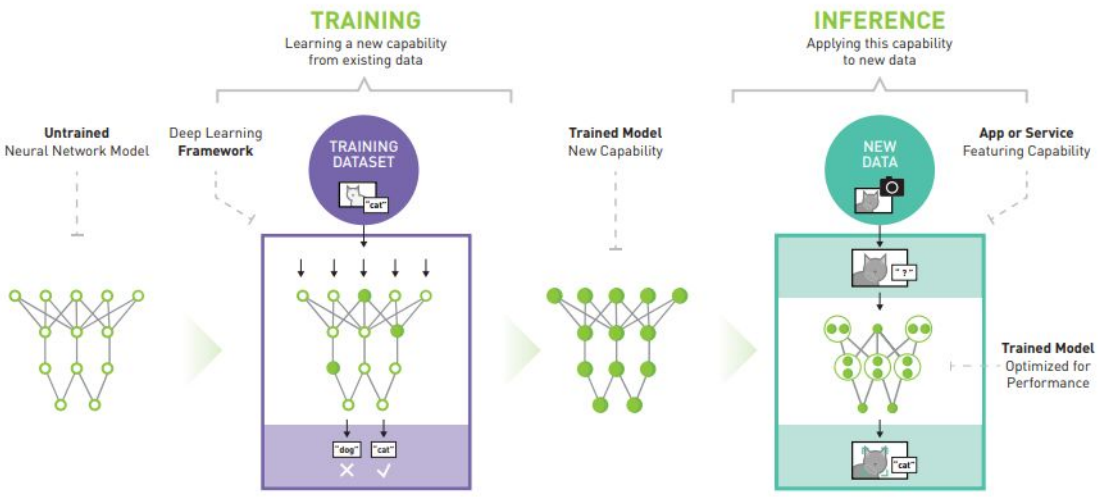


Рис. 3.2. Схема тренування моделі та її подальшої імплементації у продуктові середовища

SDK NVIDIA Deep Learning включає інструменти для наступних функцій:

- Примітивні блоки глибинного навчання - пропонують готові блоки для визначення навчальних компонентів, включаючи тензорні перетворення, функції активації та згортки.
- Механізм виводу глибинного навчання - механізм, який можна використовувати для розробки моделей у виробничому середовищі.
- Глибинне навчання для відеоаналітики - пропонує високорівневу середу виконання C ++ та API, які можна використовувати для виводу та перекодування з прискоренням на GPU.
- Лінійна алгебра - забезпечує функціональність для основних підпрограм лінійної алгебри (BLAS) з використанням прискореного графічного процесора. Це в 6–17 разів швидше, ніж при використанні лише процесора.
- Операції з розрізненими матрицями - дозволяють використовувати BLAS із прискоренням на GPU з розрізненими матрицями, такими як ті, які необхідні для обробки естетичного мови (NLP).
- Зв`язок з кількома графічними процесорами - забезпечує виконання процедур колективної зв'язку, включаючи броадкастну розсилку, скорочення та збір даних, з використанням до восьми графічних процесорів.

Принцип обчислень за допомогою TPU

Tensor Processing Unit (TPU) – тензорний процесор, розроблений компанією Google для використання з бібліотекою машинного навчання TensorFlow. Являє собою спеціалізовану інтегральну схему, що відноситься до нейронних процесорів. В порівнянні р графічними процесорами розрахований на більшу кількість обчислень з пониженою 8-розрядною точністю при менших енергозатратах.

Компанія Google застосовувала TPU для обробки фотографій Google Street View з ціллю вилучення тексту. Було заявлено, що весь об’єм даних було оброблено менш ніж за п’ять днів.

Технологія реалізована у вигляді матричного множителя для 8-розрядних чисел, що керується CISC-інструкціями CPU по шині PCI Express 3.0. Принцип у тому, що у TPU використовується менше обчислювальних бітів, ніж у GPU та CPU. Він оброблює тільки ті біти, які необхідно оброблювати у той час коли це необхідно Тобто ККД значно більший ніж у інших нейронних пристроїв.

Тензорний процесор включає в себе наступні обчислювальні ресурси:

- Матричний множитель (MXU). 65536 8-бітних модулів множення і додавання для матричних операцій.
- Уніфікований буфер (UB). 24 мегабайти SRAM, які виступають в якості регістрів.
- Блок активації (AU) – набір функцій активації.

Щоб мати можливість контролювати роботу компонентів були розроблені високо рівневі інструкції для нейронних мереж:

Таблиця 3.1

Список основного апаратного забезпечення TPU

TPU Instruction	Функція
Read_Host_Memory	Зчитування даних з пам’яті
Read_Weights	Зчитування вагів з пам’яті
MatrixMultiply / Convolve	Множення або згортка даних і вагів, накопичування результатів
Activation	Застосування функції активації
Write_Host_Memory	Запис результатів в пам’ять

Даний набір команд фокусується на основних математичних операціях, які необхідні для виводу нейронної мережі: матричного множення між вхідними даними і вагами та застосування функцій активації.

Тобто, архітектура TPU інкапсулює суть обчислень нейронної мережі і може бути запрограмована для широкого спектру моделей нейронних мереж.

Для того щоб його запрограмувати було створено компілятор і програмний стек, який переводить виклики API з графів TensorFlow в інструкції TPU.

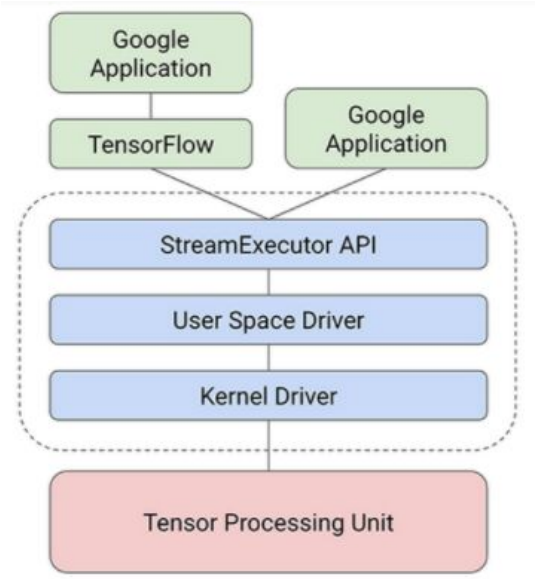


Рис. 3.3. Схема взаємодії TPU з програмним середовищем

Механізм множення матриць у TPU реалізований так, що може виконувати 65536 операцій множення і додавання 8-бітних цілих чисел за кожний цикл, що значно перевищує показники GPU і CPU. Для порівняння RISC процесор без векторної архітектури здатен виконувати декілька операцій на інструкцію, а GPU, в залежності від архітектури, біля тисячі операцій. TPU, в порівнянні з CPU та GPU, має показники у 83 і 29 разів краще і складає сотні тисяч операцій за один цикл.

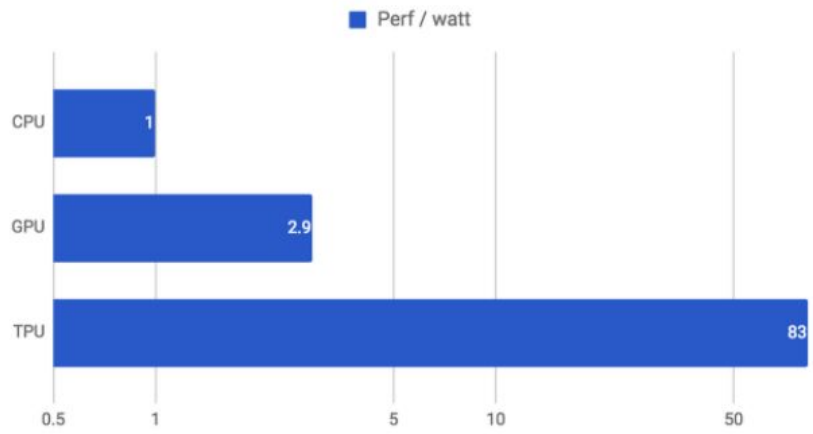


Рис. 3.4. Ефективність / ватт CPU, GPU і TPU

Також за допомогою TPU можна оцінити затрати часу для запуску нейронної мережі, що дозволяє працювати з майже піковою пропускною здатністю.

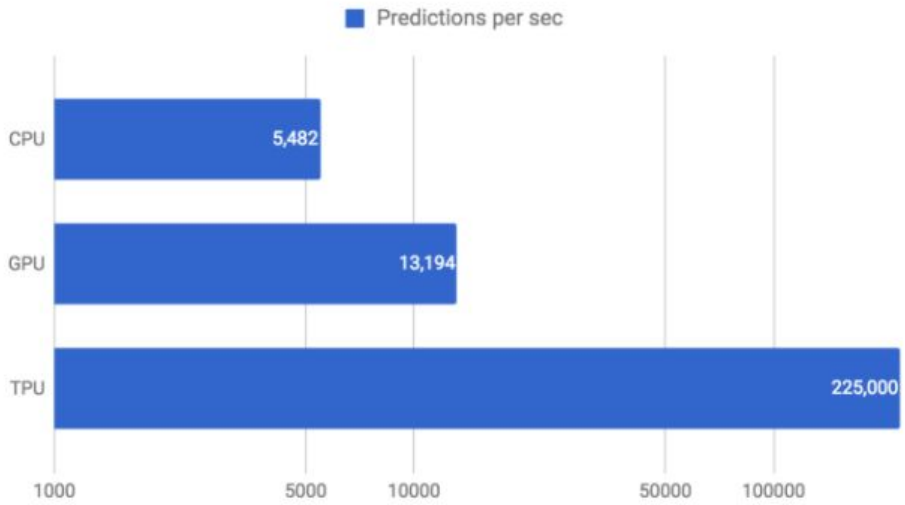


Рис. 3.5. Пропускна здатність нижче 7мс в MLP0 CPU, GPU і TPU

Нижче наведено порівняння компанії Google загальної продуктивності прогнозів за секунду меж CPU, GPU та GPU у шести додатках, що базуються на нейронних мережах з обмеженням затримки.

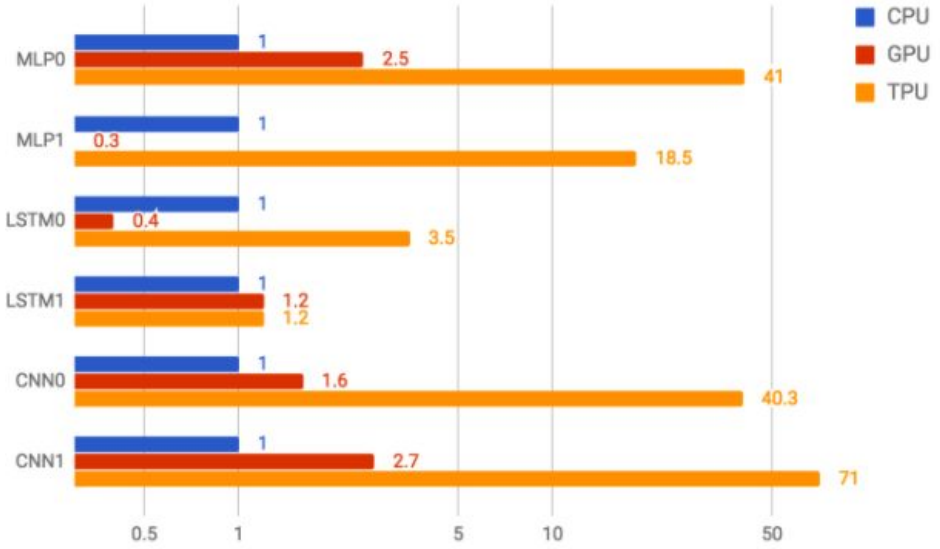


Рис. 3.6. Продуктивність CPU, GPU і TPU у шести робочих навантаженнях

3.5. Попередня обробка, опис датасетів [14, 15]

Найбільш простий тип файлу, доступний на Kaggle, - це «табличний формат даних», тобто список, елементи якого розділені спец. символами, у даному випадку комою. CSV-файли повинні мати рядок заголовку, що складається з читабельних імен полів. Наприклад, приклад списка зі строкою заголовка виглядає так:

Таблиця 3.2

Приклад формату рядка із заголовком

Id	Тип	Кількість
0	Не меланома	5
1	Меланома	7

CSV – це найпоширеніший формат файлів на платформі Kaggle, а також найкращий вибір для даних табличного формату.

Файли CSV також будуть мати зв’язані описання стовпців і метадані стовпців. Описання стовпців дозволяють назначати опис окремим стовпцям з

набору даних, що спрощує користувачам розуміння значення кожного стовпця. До того ж, показники стовпців представляють собою високо рівневі показники окремих стовпців в графічному форматі.

У використовуваному датасеті зображення представлені у форматі DICOM. Доступ до нього можна отримати за допомогою відкритих ресурсів. Даний датасет широко використовується для медичної візуалізації. Зображення представлені у форматі JPEG та TFRecord [15]. Зображення у формати JPEG мають розмір 256x256. Розмір зображень було змінено до стандартного 1024x1024. Метадані зображень також представлені у CSV файлах.

За допомогою датасету прогнозується двійкова ціль для кожного зображення. Модель повинна прогнозувати ймовірність (з плаваючою точкою) от 0.0 до 1.0, що ураження на зображенні шкіри являється злоякісним. У навчальних даних, train.csv значення 0 означає доброякісний і 1 означає злоякісний.

Файли:

- train.csv – навчальний набір даних
- test.csv – набір тестів даних
- sample_submission.csv – зразок файлу для відправки

Стовпці:

- image_name – унікальний ідентифікатор, що вказує на зображення DICOM
- patient_id – унікальний ідентифікатор пацієнта
- sex – стать пацієнта (якщо відомо, якщо ні – вільне місце)
- age_approx – приблизний вік пацієнта на момент візуалізації
- diagnosis – детальна діагностична інформація
- benign_malignant – індикатор злоякісності візуалізованого ураження шкіри
- target – бінаризована версія цільової змінної

```
feature = {
      'image': _bytes_feature,
      'image_name': _bytes_feature,
      'patient_id': _int64_feature,
      'sex': _int64_feature,
      'age_approx': _int64_feature,
      'anatom_site_general_challenge': _int64_feature,
      'diagnosis': _int64_feature,
      'target': _int64_feature,
      'width': _int64_feature,
      'height': _int64_feature
 }
```

Поля `width` та `height` – це базові ширина та висота зображення до застосування зміни розміру (кадрування) зображення.

Тестові записи TFRecord мають вищезазначене, крім diagnosis, target, `width` та `heigh`, `image_name` – це рядок, `patient_id` кодований міткою int, `sex` представлена як int за допомогою:

```
0: `male`
1: `female`
```

`anatom_site_general_challenge` кодований таким чином:

```
-1: NaN
0: 'head/neck'
1: 'upper extremity'
2: 'lower extremity'
3: 'torso',
4: 'palms/soles'
5: 'oral/genital'
```

`diagnosis` закодований наступним чином:

```
9: 'MEL'
10: 'NV'
11: 'BCC'
12: 'AK'
13: 'BKL'
14: 'DF'
15: 'VASC'
16: 'SCC'
17: 'UNK'
```

3.6. Збільшення даних

Data Augmentation – збільшення даних може бути достатньо ефективним для навчання моделей глибинного навчання. За допомогою перетворень базових зображень у датасетах можна вирішити проблему нестачі даних для навчання моделей, шляхом отримання нових даних, які є результатом перетворень базових зображень. Деякі з таких перетворень, що застосовуються до зображень для навчання нейронних моделей: геометричне перетворення, такі як перевертання, поворот, зміщення, обрізка, масштабування та зміна кольорового простору, відтінків кольору, змінення яскравості та застосування шумів.

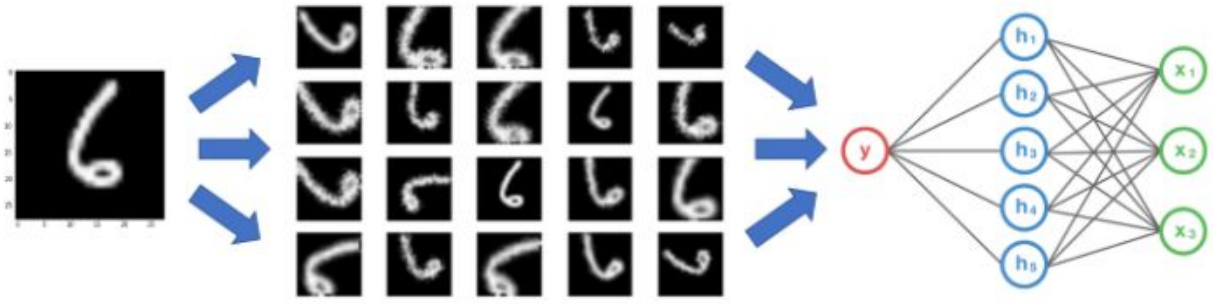


Рис. 3.7. Схема роботи процесу збільшення даних

Архітектура CNN інваріантна. Тобто, згорткові нейронні мережі можуть бути інваріантними до перетворень, точки огляду, розміру, яскравості зображення,

або ж до комбінацій таких перетворень. В реальній ситуації дані можуть знаходитись у різних умовах: бути у різному положенні у просторі, мати велику або ж малу яскравість, знаходитись у різних масштабах тощо. Щоб врахувати такий розвиток подій необхідно навчати нейронну модель з різними додатковими, штучно модифікованими даними.

У блоці “Data Augmentation” над зображеннями, шо містяться у датасетах проводяться такі геометричні перетворення: вертикальні та горизонтальні повороти зображення, зсув зображення, масштабування та зріз зображення.

Процес модифікування вхідних даних може бути реалізовано двома основними методами:

- заздалегідь обробити усі наявні дані і таким чином збільшити кількість вхідних даних у датасеті.
- Виконати перетворення перед, безпосередньо, передачею даних у нейронну модель.

Перший варіант має назву offline augmentation. Таким метод зазвичай використовується для невеликих датасетів, так як збільшивши об’єм датасета на значення, що дорівнює кількості перетворень можемо отримати завеликий об’єм. Другий варіант, навпаки, має назву online augmentation, або ж збільшення даних на льоту. Такий метод кращий для великих датасетів, оскільки може бути важко оперувати такими об’ємними даними.

- Перевертання. Можна перевертати зображення по вертикалі або горизонталі, тобто віддзеркалити його.

Рис. 3.7. Оригінальне зображення, віддзеркалення по горизонталі та вертикалі

- Обертання. При даній операції розміри зображення можуть не зберегтись після повороту. Якщо зображення квадратне, то після обертання відповідні розміри залишаться незмінними. При умові, що зображення прямокутне, то при оберті у 180 градусів розміри залишаться незмінними. Обертання на кути, що не кратні 90 градусам в будь якому разі змінять розмір зображення.

Рис. 3.8. Обертання зображення на 90 градусів та 80 градусів

- Масштабування. Зображення можна масштабувати усередину або навпаки. При масштабування назовні розміри зображення у результаті будуть більші, ніж розміри базового зображення. І навпаки, при масштабуванні усередину розміри стануть зменшуватися. Зазвичай для вирівнювання використовують інструменти, такі як вирізання із вихідного зображення розмірів базового.
- Вирізання. На відміну від масштабування випадкова частина вихідного зображення вирізається із оригінального. Після цього розмір вихідного зображення змінюється до розмірів оригінального зображення. Такий метод називається кадруванням зображення.

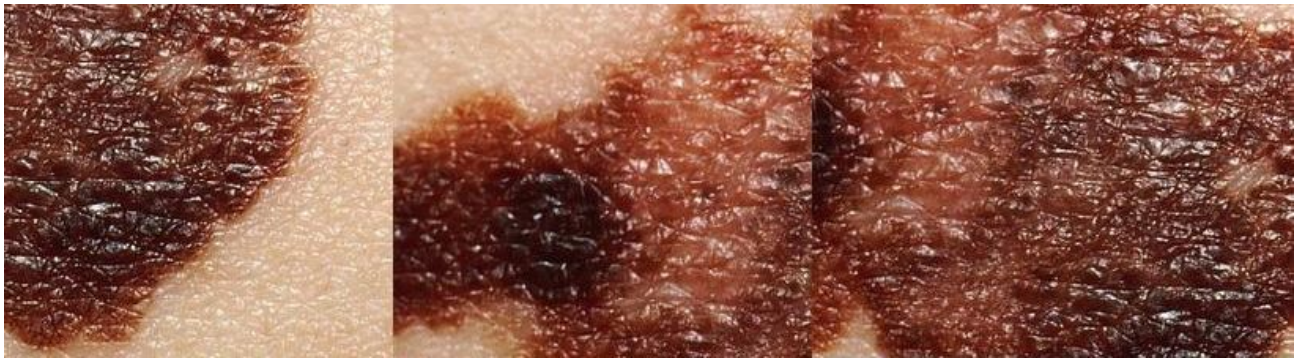


Рис. 3.9. Приклад вирізання випадкових частин вхідного зображення, так зване кадрування

- Зсув. Зсув реалізується за допомогою зміщення зображення по осям X або Y, або ж комбінований варіант. Такий метод корисний, адже більшість об’єктів на реальних зображеннях можна розташувати у будь-якому місці, тобто на різних координатах. Це тренує нейронну мережу на пошук шуканих об’єктів по всій площі вхідного зображення.

Підвищення частоти перетворень здатне призвести до перенавчання моделі, адже модель навчається на одному і тому ж прикладі декілька разів.

3.7. EfficientNet

EfficientNets [18, 19, 20] – це сімейство моделей класифікації зображень, які здатні забезпечити високу точність при тому, що їх розмір набагато менший, а також вони швидші за моделі, що широко використовувались для подібних задач до них.

В рамках ICML 2019 [21] була опублікована стаття, у якій дослідники запропонували метод для оптимізації згорткових нейронних мереж. Попередні методи, як правило, розроблялися з фіксованою кількістю ресурсів, а потім, в процесі постобробки, масштабувалися для досягнення ліпших результатів при умові доступності більшої кількості ресурсів. [11] Зазвичай практика масштабування моделей полягає в довільному збільшенні глибини, ширини згорткової мережі, або збільшенні вхідного зображення.

Запропонований метод, на відміну від загальноприйнятої практики, рівномірно масштабує усі виміри нейронної мережі з фіксованими коефіцієнтами масштабування.

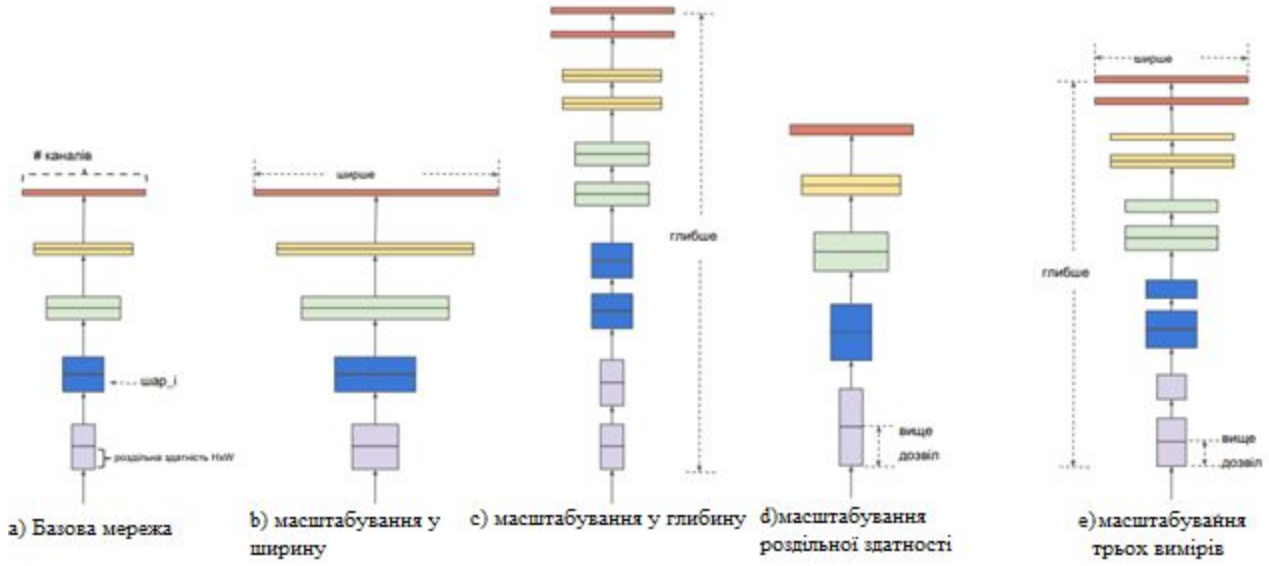


Рис. 3.11. Масштабування моделі. a) є прикладом базової мережі; b) - d) - це звичайне масштабування, яке збільшує лише один вимір ширини, глибини або роздільної здатності мережі, e) - запропонований нами метод складання масштабу, який рівномірно масштабує всі три виміри із фіксованим співвідношенням.

Ефективність масштабування нейронної мережі залежить від її початкової архітектури. Для того щоб поліпшити роботу нейронної мережі була розроблена архітектура на основі AutoML фреймворку – MNAS, який при виборі пристрою оптимізує одночасно розмір моделі (FLOPS) і її точність. На початку використовувалась модель MBConv, з якої в процесі масштабування отримали моделі з архітектурою EfficientNet.

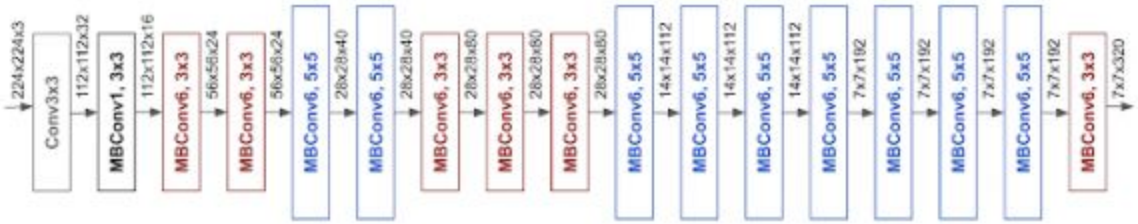


Рис. 3.12. Архітектура моделі EfficientNetB0

Робота методу починається з визначення взаємозв'язку між різними вимірами базової архітектури при одній і тій же кількості ресурсів. Це визначає підходящий коефіцієнт масштабування для кожного виміру, який потім застосовується для масштабування базової мережі до необхідних розмірів.

Наприклад якщо є необхідність використання в 2^N рази більше ресурсів, то параметри мережі збільшуються у відповідну кількість разів: глибина збільшується на α^N разів, ширина на β^N, розмір зображення на γ^N, де α, β і γ постійні коефіцієнти. Рисунок 3.10. ілюструє різницю між методом масштабування та звичайним методом.

Такий підхід обходить state-of-the-art підходи у точності при менших розмірах моделі та затратах часу. Тестування EfficientNet показало більш високу точність та кращу ефективність в порівнянні з уже існуючими архітектурами CNN, при цьому зменшивши розмір параметрів, а також FLOPS на порядок. Також модель EfficientNet досягла більше ніж 84% топ-1 та 97,1 топ-5 точності в ImageNet. Наприклад, архітектура моделі EfficientNetB7 у 8,4 рази менша і в 6,1 рази швидша за попереднього лідера, CNN модель GPipe. Також, як приклад, модель ResNet є можливість масштабувати, з ResNet-18 до ResNet-200, використовуючи більше шарів. В порівнянні з ResNet50 різновид архітектури EfficientNet – EfficientNetB4 використала стільки ж FLOPS, одночасно з цим поліпшивши точність топ-1 з 76,3% до 82,6%.

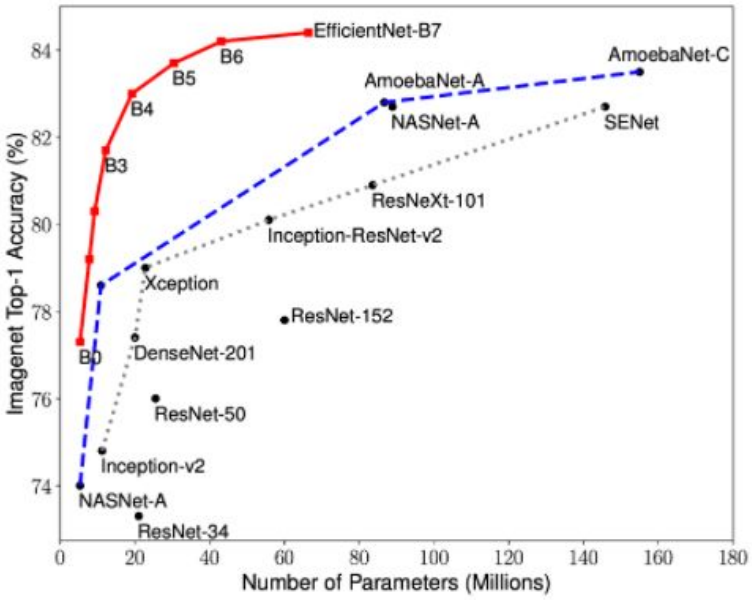


Рис. 3.13. Порівняння точності і розміру CNN архітектур і EfficientNet [23]

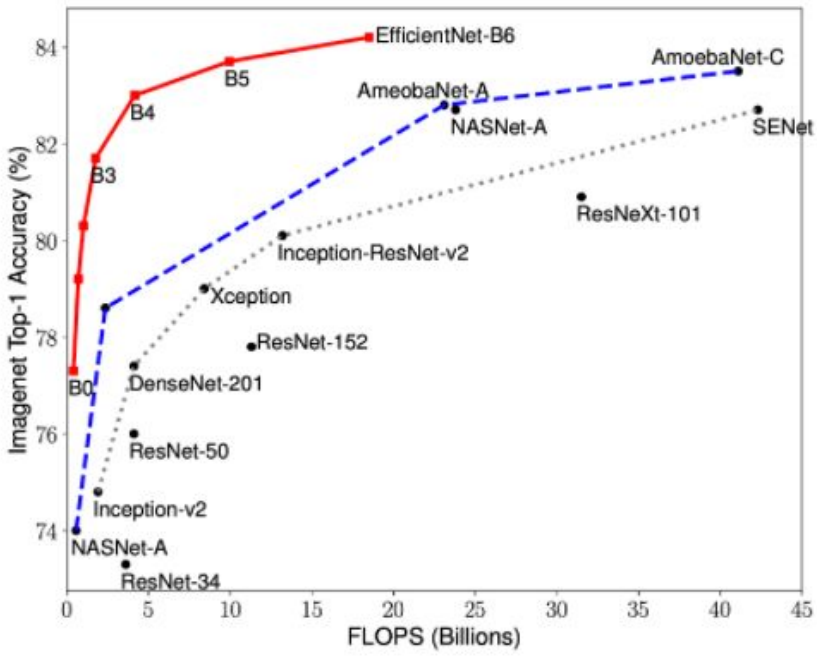


Рис.3.14. Порівняння розмірів CNN архітектур (FLOPS) і EfficientNet [23]

Збільшення даних під час тестування

Test time augmentation (TTA) – збільшення даних під час тестування. Даний підхід дозволяє покращити ефективність класифікації за рахунок усереднення передбачення для зображень та їх збільшення.

Для покращення ефективності класифікації моделей глибинного навчання використовується такий підхід, як збільшення даних, суть якої полягає у тому, шо

під час тренування нейронної мережі в якості вхідних даних окрім, безпосередньо, тренувальних даних подаються їх модифікації. Для зображень це можуть бути різного роду перетворювання: перевертання, оберти, віддзеркалення по горизонталі або вертикалі, зміна яскравості зображення, накладання шуму тощо.

Суть даного підходу полягає у тому, що подібні перетворення можно застосовувати не тільки для тренування нейронної моделі, але й при розпізнаванні. Для цього вхідні дані зазнають перетворень так само, як і під час тренування моделі. Перетворені дані подаються у якості вхідних до нейронної мережі, яка базуючись на них виконує передбачення. Результатом буде середнє значення від отриманих передбачень.

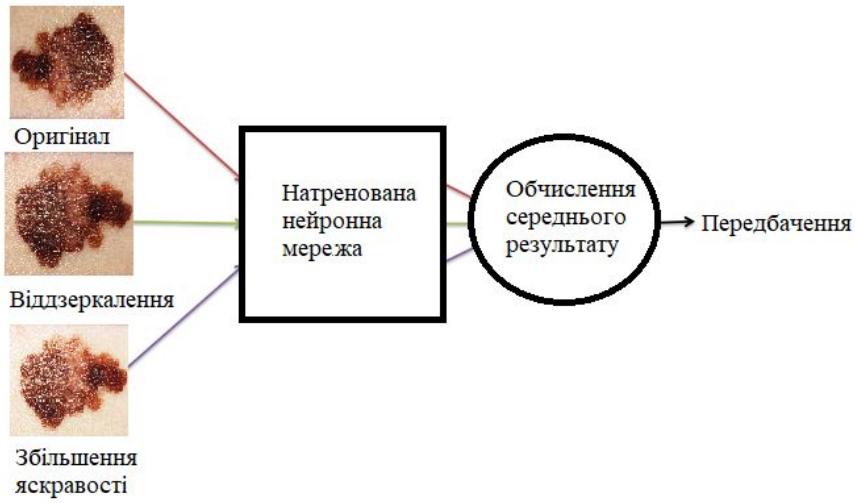


Рис.3.15. Схематичне зображення процесу збільшення даних під час тестування

3.8. Підготовка тренування моделі. K-fold перехресна перевірка [22]

Використані датасети пройшли процедуру розділення по пацієнтам з ціллю зменшити витоку даних під час тренування. У наборі даних існує 15 TFRecords, тому рекомендовано значення FOLD = 5, хоча теоретично це може бути будь-яке число з 2 до 15.

Під час завантаження датасету, у файлі CSV перераховуються зображення, які містять файли TFRecords. Зображення зі значеннями

`tfrecord = -1` є дублікатами та видаляються.

- Розділення 1.

Один і той же пацієнт може мати декілька екземплярів зображень. Усі вони складаються до єдиного TFRecords, що запобігає витоку даних під час операції cross-validation.

- Розділення 2.

Датасет містить 1.8% злоякісних уражень шкіри. Кожний TFRecord містить також 1.8% зображень з злоякісними ураженнями. Даний підхід робить процесс cross-validation надійнішим.

- Розділення 3.

Деякі пацієнти мають біля 115 зображень, деякі – лише 2. При розділенні пацієнтів в TFRecords кожний запис містить однакову кількість пацієнтів зі 115, 100, 70, 50, 20, 10, 5, 2 і тд. Зображеннями. Такий підхід робить процедуру cross-validation більш надійною.

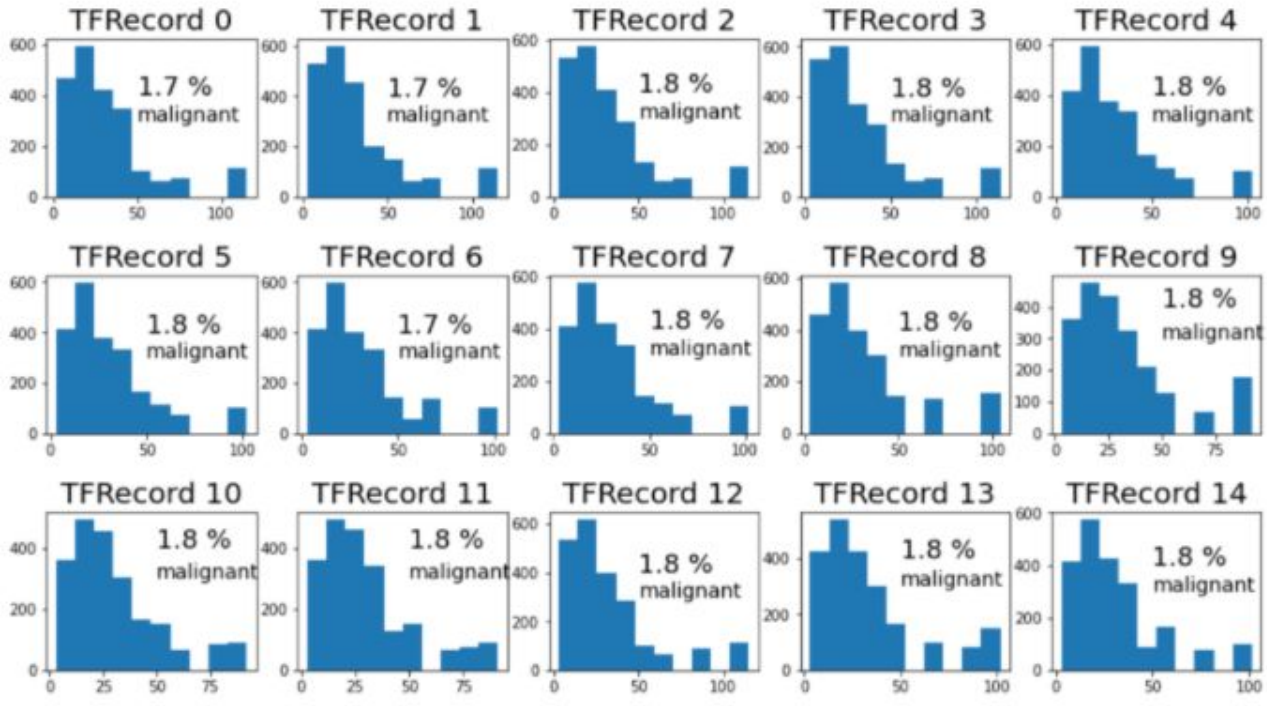


Рис.3.16. Гістрограми пацієнтів та їх кількість в кожному TFRecord файлі.

Вищезгадані розділення даних роблять процедуру cross-validation більш надійною та запобігають витоку під час даної процедури. Залишок даних у вигляді

повторюваних зображень були видалені із датасету з ціллю повністю уникнути витоку даних.

3.9. Пост обробка. Вимір часу прогнозування, AUC, розмір моделі.

Результати роботи представлені у розділі 4 декількома показниками, зокрема такими як час прогнозування, час прогнозування, враховуючи TTA, розмір моделі з ціллю дослідження можливості імплементації у ті чи інші підходи, показник AUC, показник AUC з урахуванням TTA.

Отримання даних у роботі виконується за допомогою інструментів мови програмування python. Даний модуль вбудований в основний код та виконує функцію сбору отриманих даних після виконання своєї роботи нейронною мережею. З даним модулем можна ознайомитись у прикладеному додатку А.

Висновок до розділу 3

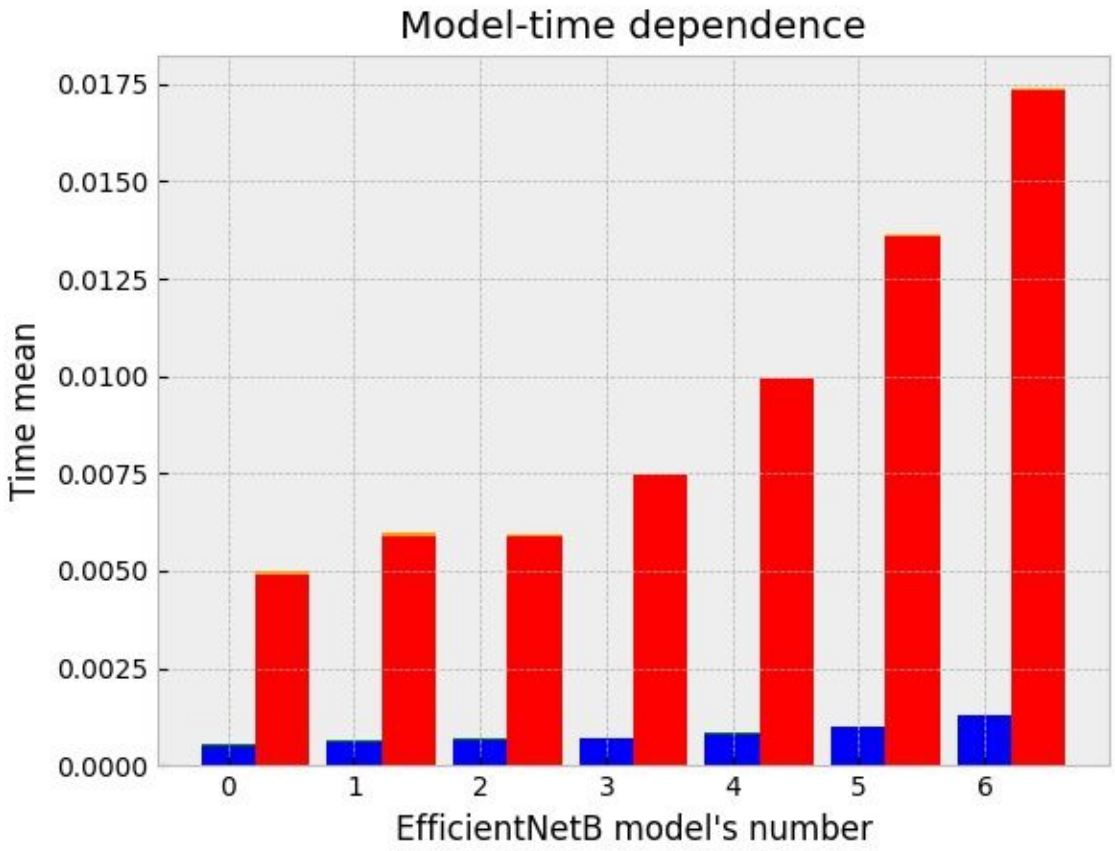
У даному розділі було описано застосований метод обробки медичних зображень, а саме класифікації на них злоякісних уражень. Було описано основні характеристики методу, його властивості та застсоовані архітектури нейронних мереж.

Також було описано і сформовано конфігурації апаратного забезпечення для виконання досліджень на базі відкритої платформи Kaggle для тренування і тестування нейронних моделей. Надано опис деталей створених конфігурацій.
Сформовано конфігурації застосованих архітектур нейронних мереж для отримання необхідних результатів та надано їх (конфігурацій) опис. Створено допоміжні модулі для отримання даних результатів роботи використаних нейронних мереж.

РОЗДІЛ 4. РЕЗУЛЬТАТИ ВИКОРИСТАННЯ МЕТОДУ ОБРОБКИ МЕДИЧНИХ ЗОБРАЖЕНЬ ШКІРИ ЛЮДИНИ

У даному розділі надано результати, отримані у налслідок виконання практичної частини дослідження методу обробки медичних зображень шкіри людини, а саме:

- Тренування стандартних і новітніх нейронних моделей поглибленого навчання із використанням методу крос-валідації (k-fold cross validation) і штучного збільшення кількості варіантів тренувальних зображень (data augmentation - DA),
- Тестування отриманих тренованих моделей із залученням методу штучного збільшення кількості варіантів тестових зображень (post-training/testing time data augmentation - TTA),
- Результати порівняльного аналізу отриманих даних на різних інфраструктурах, моделях і їх параметрах для дослідження впливу типу інфраструктури (TPU/GPU), типу і розміру моделей на час і точність прогнозування за стандартними метриками



Діаграма 1. Залежність часу тестування моделі від типу ахітектури моделі типу EfficientNet, де модель EfficientNet має відповідний числовий еквівалент (0 - EfficientNetB0, 1 – EfficientNetB1 і т.д.), результати тренування за допомогою GPU червоного кольору та standard deviation оранжевого, результати тренування за допомогою TPU синього кольору та standard deviation зеленого.

Таблиця 1

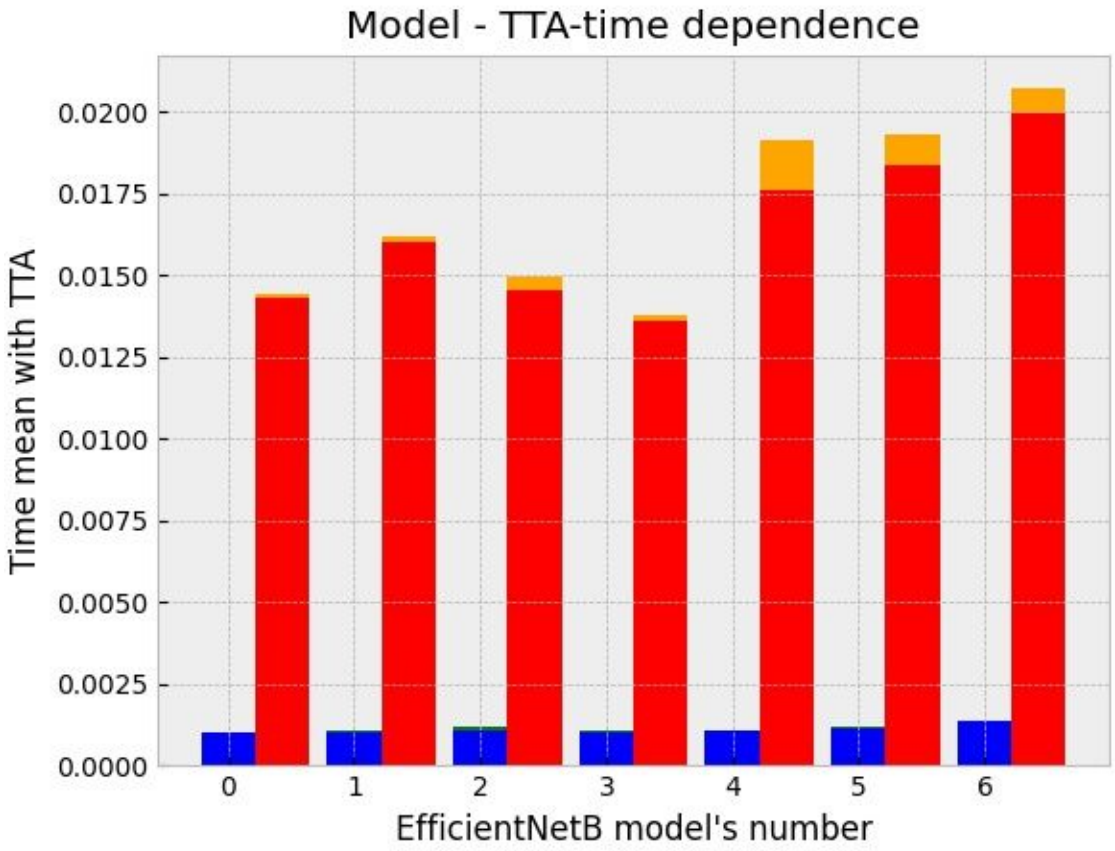
Результати часу тестування та середнього квадратичного відхилення моделей архітектури EfficintNet на GPU

model	time_mean	time_std
B0	0.0049	0.000106
B1	0.0058	0.000103
B2	0.0058	0, 000060
B3	0.00744	0,000033
B4	0.00992	0,000029
B5	0.01357	0,000079
B6	0.01732	0,000056

Таблиця 2

Результати часу тестування та середнього квадратичного відхилення моделей архітектури EfficintNet на TPU

model	time_mean	time_std
B0	0.00051	0,000015
B1	0.00061	0,000012
B2	0.00063	0,000051
B3	0.00069	0,000007
B4	0.00081	0,000014
B5	0.00097	0,000016
B6	0.00129	0,00002



Діаграма 2. Залежність часу тестування моделі з урахуванням часу test-time augmentation від типу ахітектури моделі типу EfficientNet, де модель EfficientNet має відповідний числовий еквівалент (0 - EfficientNetB0, 1 – EfficientNetB1 і т.д.), результати тренування за допомогою GPU червоного кольору та standard deviation оранжевого, результати тренування за допомогою TPU синього кольору та standard deviation зеленого

Таблиця 3

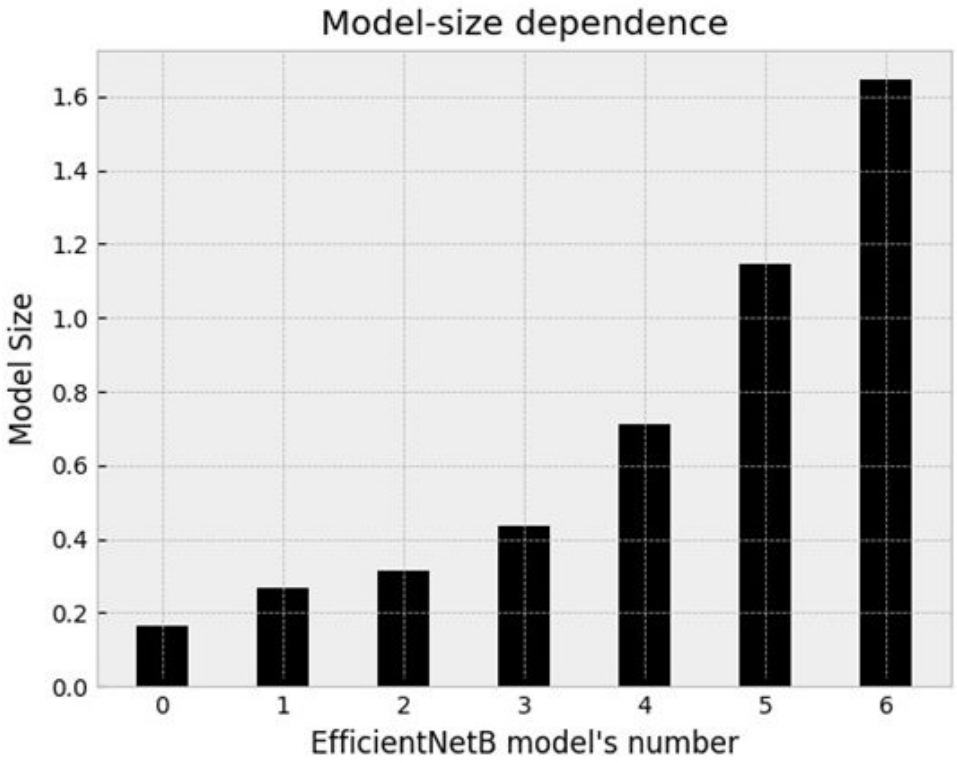
Результати часу тестування та середнього квадратичного відхилення моделей архітектури EfficintNet на GPU з урахуванням часу test-time augmentation

model	TTA_time_mean	TTA_time_std
B0	0.0143	0.00013
B1	0.0160	0.00016
B2	0.0145	0.00039
B3	0.0135	0.00021
B4	0.0176	0.00149
B5	0.0183	0.00094
B6	0.0199	0.00077

Таблиця 4

Результати часу тестування та середнього квадратичного відхилення моделей архітектури EfficintNet на TPU з урахуванням часу test-time augmentation

model	TTA_time_mean	TTA_time_std
B0	0.00102	0,0000107
B1	0.00103	0,0000092
B2	0.00109	0,0000888
B3	0.00103	0,0000115
B4	0.00105	0,0000133
B5	0.00115	0,0000167
B6	0.00137	0,0000118

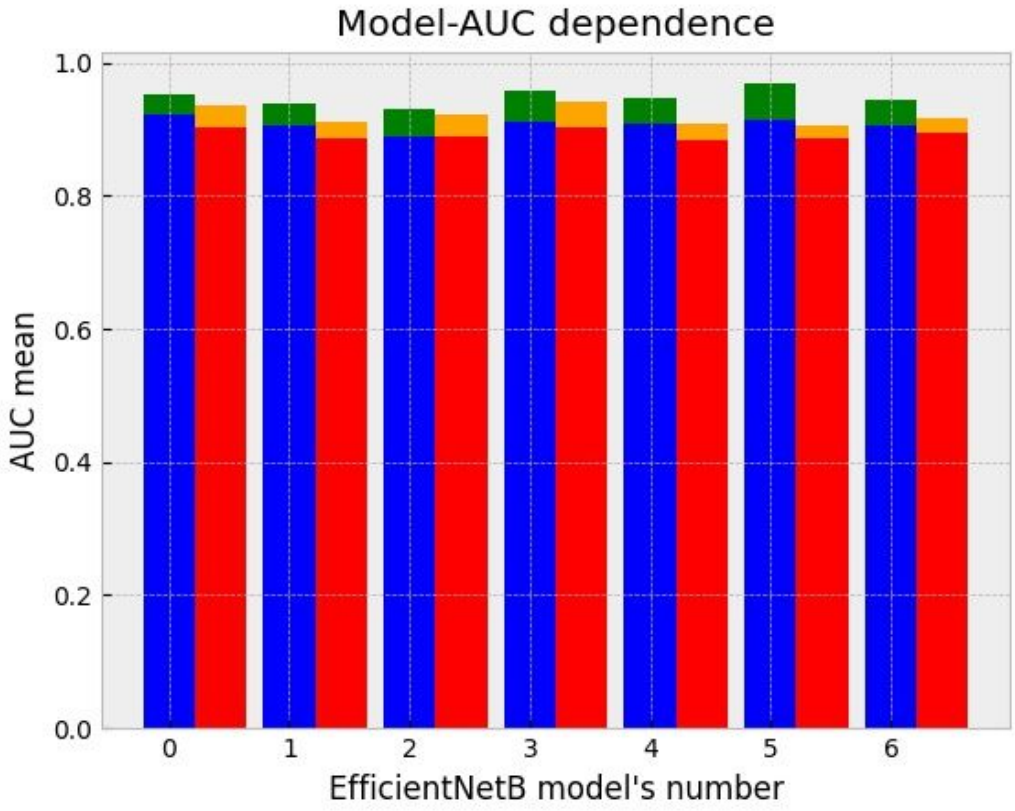


Діаграма 3. Залежність розміру нейронної моделі від типу ахітектури моделей EfficientNet, де модель EfficientNet має відповідний числовий еквівалент (0 - EfficientNetB0, 1 – EfficientNetB1 і т.д.)

Таблиця 5

Залежність розміру нейронної моделі від типу ахітектури моделей EfficientNet

model	model_size
B0	16462680
B1	26667544
B2	31441608
B3	43546888
B4	71199408
B5	114665568
B6	164544944



Діаграма 4. Залежність величини показника якості класифікації від типу ахітектури моделі типу EfficientNet, де модель EfficientNet має відповідний числовий еквівалент (0 - EfficientNetB0, 1 – EfficientNetB1 і т.д.), результати тренування за допомогою GPU червоного кольору та standard deviation оранжевого, результати тренування за допомогою TPU синього кольору та standard deviation зеленого.

Таблиця 6

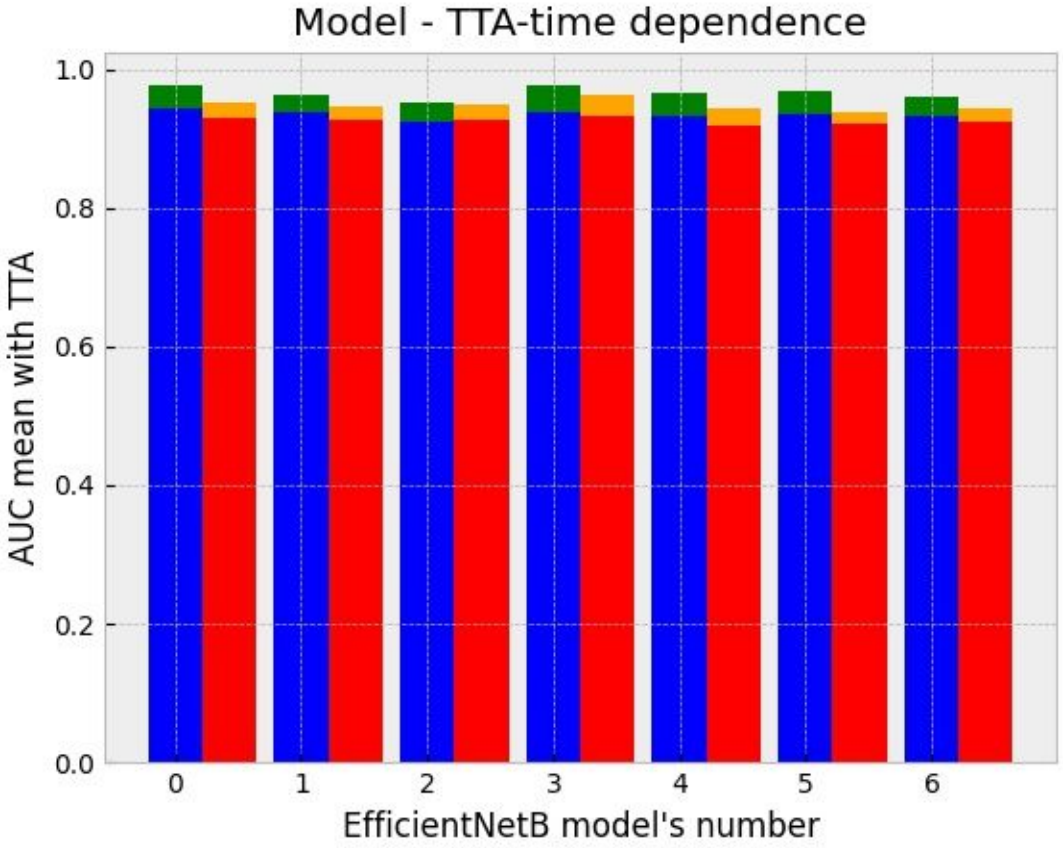
Величини показника якості класифікації та середнього квадратичного відхилення кожного типу ахітектури моделі типу EfficientNet під час тренування моделі на GPU

model	AUC _mean	AUC _std
B0	0.901	0.033
B1	0.886	0.026
B2	0.889	0.033
B3	0.902	0.039
B4	0.882	0.025
B5	0.885	0.019
B6	0.895	0.022

Таблиця 7

Величини показника якості класифікації та середнього квадратичного відхилення кожного типу ахітектури моделі типу EfficientNet під час тренування моделі на TPU

model	AUC _mean	AUC_std
B0	0.92	0.031
B1	0.905	0.032
B2	0.888	0.043
B3	0.91	0.048
B4	0.907	0.038
B5	0.914	0.054
B6	0.905	0.038



Діаграма 5. Залежність величини показника якості класифікації від типу ахітектури моделі типу EfficientNet з урахуванням test-time augmentation, де модель EfficientNet має відповідний числовий еквівалент (0 - EfficientNetB0, 1 – EfficientNetB1 і т.д.), результати тренування за допомогою GPU червоного кольору та standard deviation оранжевого, результати тренування за допомогою TPU синього кольору та standard deviation зеленого.

Таблиця 8

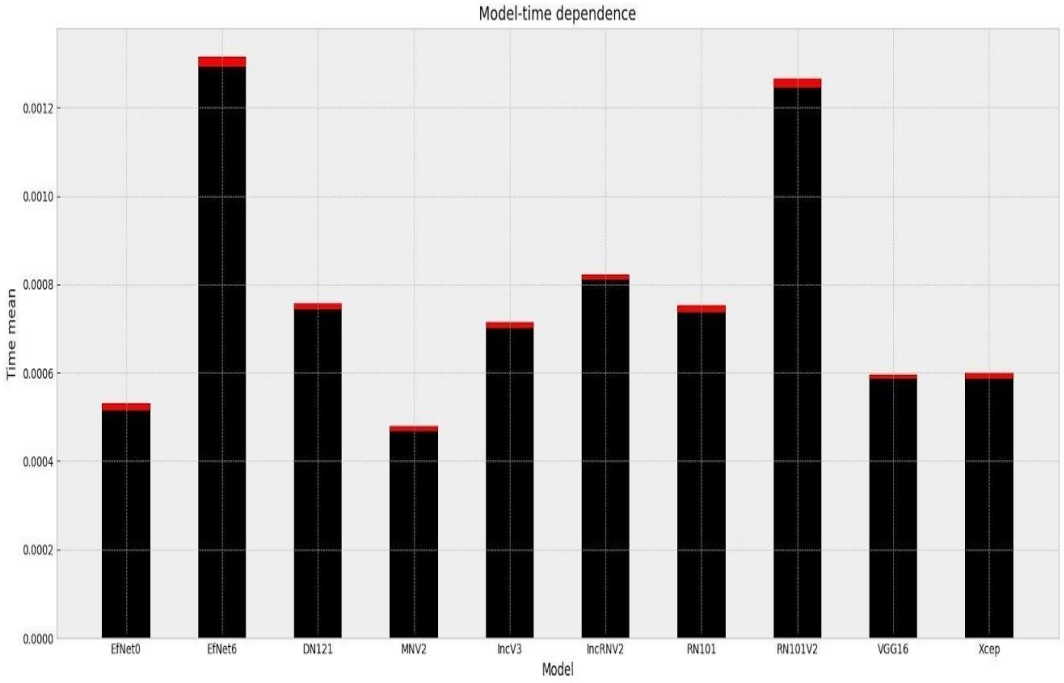
Величини показника якості класифікації кожного типу архітектури моделі типу EfficientNet під час тренування моделі на GPU з урахуванням test-time augmentation

model	TTA_AUC_mean	TTA_AUC _std
0	0.929	0.021
1	0.925	0.021
2	0.926	0.021
3	0.931	0.031
4	0.919	0.024
5	0.922	0.016
6	0.923	0.018

Таблиця 9

Величини показника якості класифікації кожного типу архітектури моделі типу EfficientNet під час тренування моделі на TPU з урахуванням test-time augmentation

model	TTA_AUC_ mean	TTA_AUC_std
0	0.942	0.033
1	0.938	0.023
2	0.923	0.027
3	0.937	0.038
4	0.931	0.034
5	0.934	0.033
6	0.931	0.027

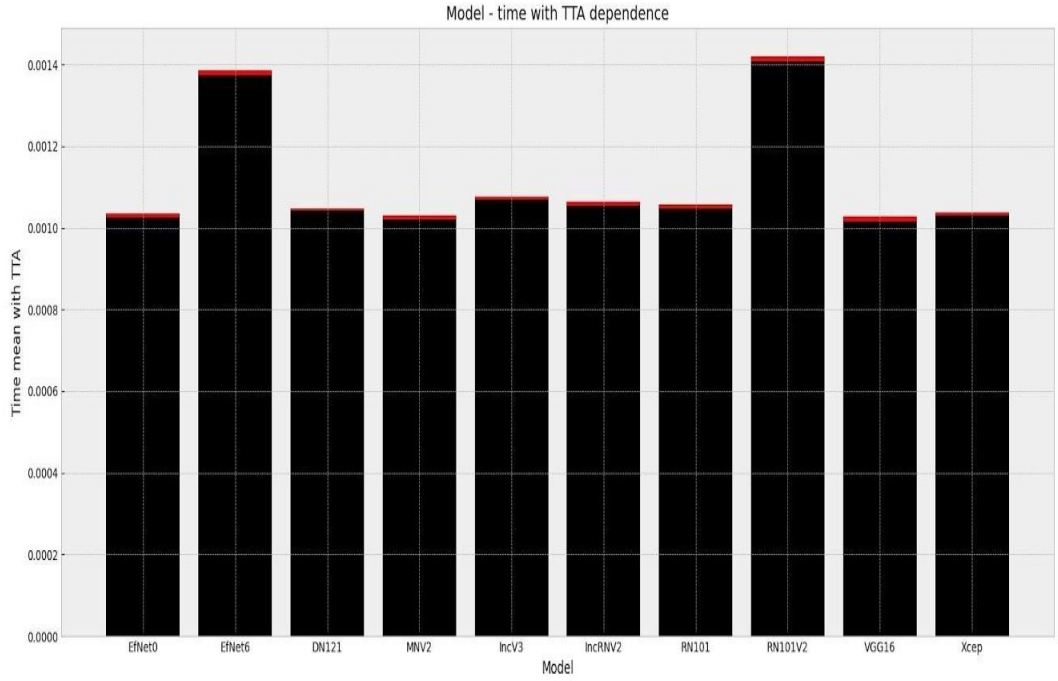


Діаграма 6. Залежність часу тестування моделі від типу ахітектури моделей

Таблиця 10

Показники часу тестування та середнього квадратичного відхилення нейронних моделей, що були використані

model	time_mean	time_std
EfficientNetB0	0.00051	0,0000158
EfficientNetB6	0.00129	0,0000208
DenseNet121	0,00074	0,0000127
MobileNetV2	0,00046	0,0000111
InceptionV3	0,00070	0,0000121
ResNet101	0,00081	0,0000121
ResNet101V2	0,00073	0,0000167
InceptionResNetV2	0,00124	0,0000205
VGG16	0,00058	0,0000086
Xception	0,00058	0,0000116

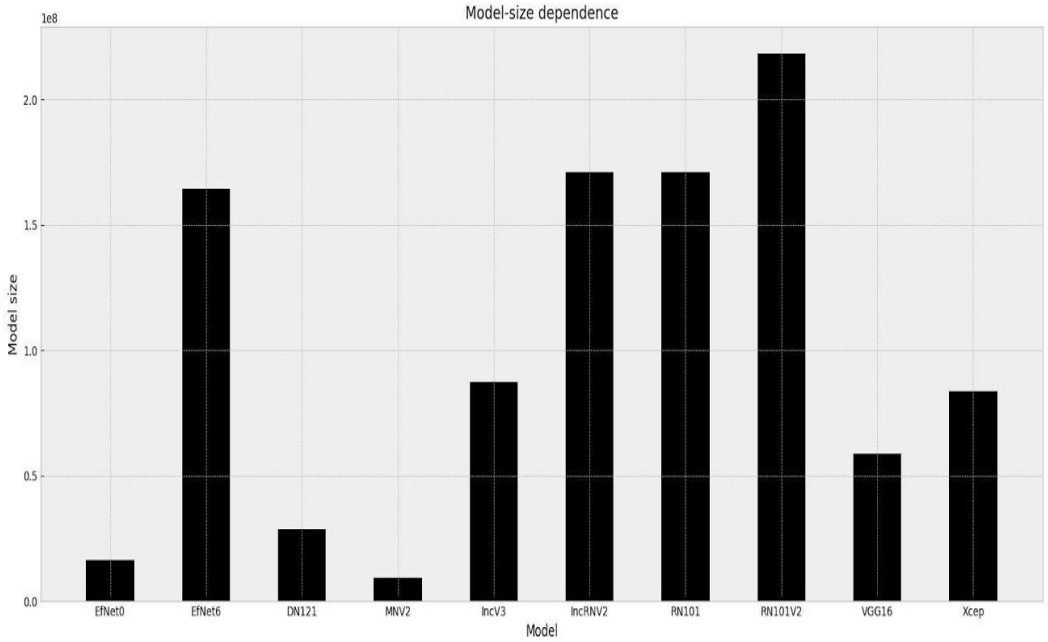


Діаграма 7. Залежність часу тестування та середнього квадратичного відхилення нейронних моделей, що були використані, з урахуванням test-time augmentation, де час тренування позначено чорним кольором, а середнє квадратичне відхилення червоним

Таблиця 11

Показники часу тестування та середнього квадратичного відхилення нейронних моделей, що були використані, з урахуванням test-time augmentation

model	TTA_time_mean	TTA_time_std
EfficientNetB0	0.00102	0, 000010
EfficientNetB6	0.00137	0,0000118
DenseNet121	0,00104	0,0000057
MobileNetV2	0,00102	0,0000084
InceptionV3	0,00106	0,0000078
ResNet101	0,00105	0,0000081
ResNet101V2	0,00104	0,0000084
InceptionResNetV2	0,00140	0,000012
VGG16	0,00101	0,000011
Xception	0,00103	0,000006

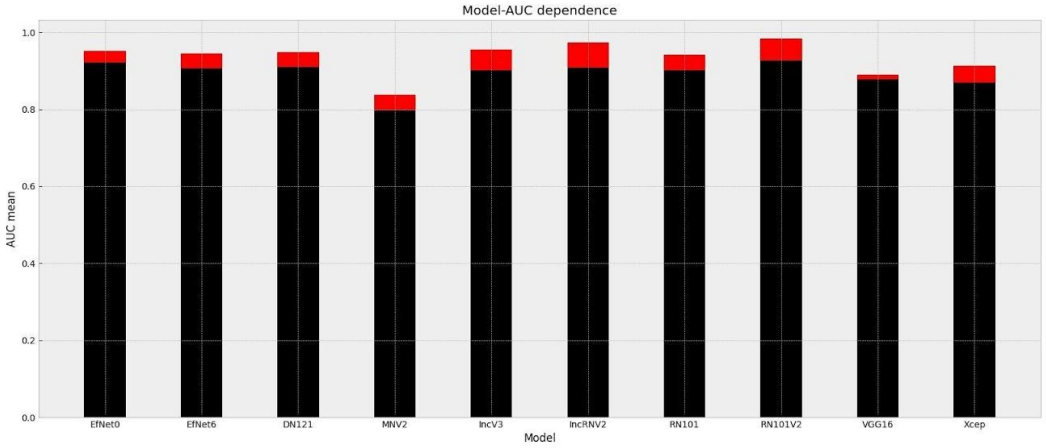


Діаграма 8. Порівняння показників розмірів досліджуваних моделей

Таблиця 12

Показники розмірів досліджуваних моделей

model	model_size
EfficientNetB0	16462680
EfficientNetB6	164544944
DenseNet121	28638208
MobileNetV2	9252416
InceptionV3	87566624
ResNet101	171090312
ResNet101V2	170935096
InceptionResNetV2	218161160
VGG16	58894832
Xception	83632392

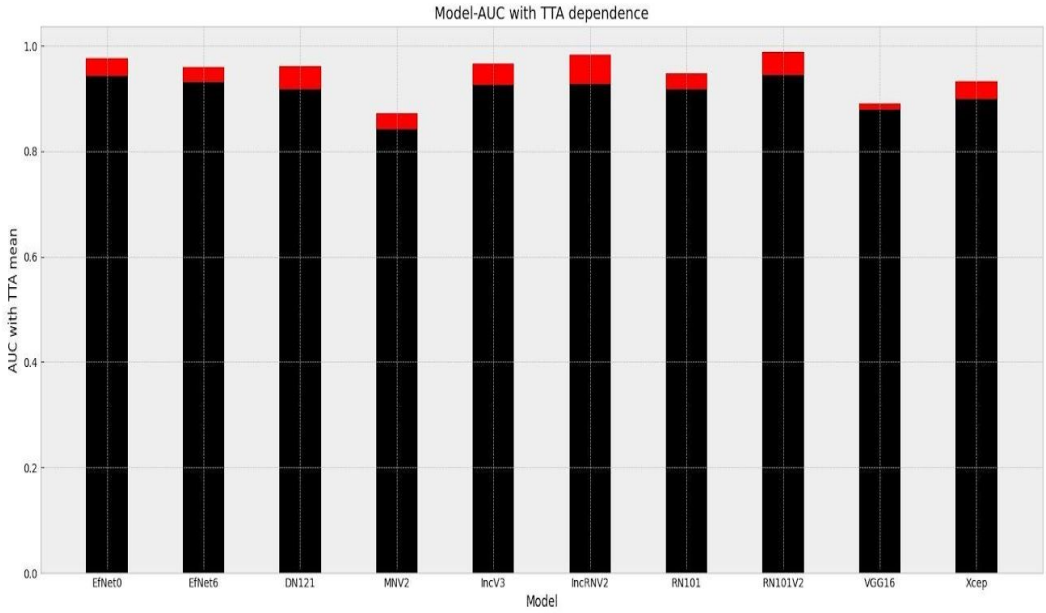


Діаграма 9. Залежність величини показника якості класифікації та середнього квадратичного відхилення від типу архітектури моделі, де показник AUC позначено чорним кольором, а середнє квадратичне відхилення червоним

Таблиця 13

Величини показника якості класифікації та середнього квадратичного відхилення кожного типу ахітектури моделі

model	AUC_mean	AUC_std
EfficientNetB0	0.920	0.031
EfficientNetB6	0.905	0.038
DenseNet121	0,91	0,037
MobileNetV2	0,797	0,04
InceptionV3	0,902	0,052
ResNet101	0,907	0,066
ResNet101V2	0,901	0,04
InceptionResNetV2	0,926	0,056
VGG16	0,877	0,011
Xception	0,869	0,043



Таблиця 13. Діаграма 10. Залежність величини показника якості класифікації та середнього квадратичного відхилення від типу архітектури моделі з урахуванням test-time augmentation, де показник AUC позначено чорним кольором, а середнє квадратичне відхилення червоним

Таблиця 14

Величини показника якості класифікації та середнього квадратичного відхилення кожного типу архітектури моделі з урахуванням test-time augmentation

model	TTA_AUC_mean	TTA_AUC_std
EfficientNetB0	0.942	0.033
EfficientNetB6	0.931	0.027
DenseNet121	0,917	0,044
MobileNetV2	0,841	0,03
InceptionV3	0,926	0,04
ResNet101	0,928	0,054
ResNet101V2	0,917	0,03
InceptionResNetV2	0,944	0,043
VGG16	0,878	0,012
Xception	0,898	0,0352

На основі здобутих результатів буде визначено, що:

Найменшими за розміром є мережі:

1. MobileNetV2
2. EfficientNetB0
3. DenseNet121

Найбільшу точність в межах похибки демонструють мережі:

1. EfficientNetB0
2. EfficientNetB6
3. InceptionResNetV

Найшвидшими без TTA є мережі:

1. MobileNetV2
2. EfficientNetB0
3. VGG16 та Xception мають майже ідентичні результати

Найшвидшими із TTA є мережі:

1. VGG16
2. MobileNetV2
3. EfficientNetB0

Отже було сформульовано наступні рекомендації для практичного виокористання обстежених мереж та конфігурацій апаратного забезпечення.

Для використання:

- для завдань із невисокими вимогами до часу прогнозу на стаціонарних пристроях із необмеженими ресурсами (хмарні ресурси на базі GPU-TPU вузлів) найбільш ефективно було б використовувати мережі із найбільшою точністю: моделі архітектури EfficientNet, віддаючи перевагу моделі EfficientNetB0.
- для завдань із високими вимогами до часу прогнозу на стаціонарних пристроях із обмеженими ресурсами (локальні ресурси на базі GPU карт) найбільш ефективно було б використовувати мережі із найбільшою точністю і найбільшою

швидкістю: EfficientNetB0 єдина серед досліджених моделей, що здатна сумістити у собі всі три фактори.

На пристроях із обмеженими ресурсами (пристроях із портативними TPU-модулями, наприклад, Google Coral або Intel Movidius):

- для завдань із високими вимогами до часу прогнозу найбільш ефективнобуло б використовувати мережі найменшого розміру та найбільшою швидкістю: EfficientNetB0 та MobileNetV2, що мають лідируючі результати у даних показниках
- для завдань із невисокими вимогами до часу прогнозу найбільш ефективно було б використовувати мережі найменшого розміру та найбільшою точністю: EfficientNetB0 має найкращу комбінацію даних показників

Висновок до розділу 4

У даному розділі описані результати роботи методу обробки медичних зображень людини на основі нейронної мережі на основі результатів дослідження процесу класифікації меланом на знімках уражень шкіри на реальних медичних зображеннях пацієнтів, із визначенням основних параметрів їх роботи для прикладного застосування у сфері класифікації шкірних захворювань загалом і зокрема меланом.

Для досягнення даної мети було виконано **наступні основні завдання:**

- підготовлено датасет із медичними даними на основі відкритих даних 2018-2020 років із https://challenge2020.isic-archive.com/
- сконфігуровано дослідницьку інфраструктуру на основі віддалених хмарних ресурсів і залученням графічних прискорювачів (graphic processing units — GPU) і прискорювачів тензорних обчислень (tensor processing units — TPU),
- виконано тренування на кількох стандартних (DenseNet121, MobileNetV2, MobileNetV2, InceptionResNetV2, ResNet101, ResNet101V2, VGG16, Xception) і новітніх (EfficientNetB0-B6) моделях поглибленого навчання із використанням методу крос-валідації (k-fold cross validation) і штучного збільшення кількості варіантів тренувальних зображень (data augmentation - DA),
- виконано тестування отриманих тренованих моделей із залученням методу штучного збільшення кількості варіантів тестових зображень (post-training/testing time data augmentation - TTA),
- виконано порівняльний аналіз отриманих даних на різних інфраструктурах, моделях і їх параметрах для дослідження впливу типу інфраструктури (TPU/GPU), типу і розміру моделей на час і точність прогнозування за стандартними метриками,
- сформулювано рекомендації щодо можливості практичного застосування отриманих результатів в якості методу обробки медичних зображень

людини на основі нейронної мережі у вигляді сервісу із локальним або віддаленим доступом.

У кінечному результаті було визначено, що модель EfficientNetB0 має найкращі перспективи, адже вона займає лідируючі позиції у більшості результатів вимірювань основних показників (розмір моделі, точність, час прогнозу) нейронних моделей

РОЗДІЛ 5. РОЗРОБКА СТАРТАП ПРОЕКТУ

Стартап в останні десятиліття набув широкого розповсюдження. Причиною тому стала поява мережі Інтернет, адже за допомогою даної технології процес пошуку потенційних клієнтів та комерціалізації того чи іншого проекту став в рази простішим.

Розробка продукту та його комерціалізація потребує спланованої стратегії, яка складається з визначення, визначення перспектив проекту, складання бізнес плану, визначення основних ризиків та методи їх нейтралізації тощо.

У даному розділі розглядається стратегія комерціалізації проекту та його вихід на ринок споживання, розглядаючи різні його сегменти, враховуючи перспективи та можливості у кожному з них.

5.1. Опис стартап-проекту

1. Назва проекту	Інтелектуальна система діагностики захворювань шкіри

2. Автори проекту	Домс Володимир

3. Коротка анотація (не більше 1/3 сторінки)	Проект ставить за мету створення більш прагматичної та доступної системи діагностики захворювань шкіри (меланом).

4. Термін реалізації проекту	12
	Тривалість проекту (в місяцях)

5. Необхідні ресурси	Фінансові: 1. Витрати на купівлю серверу для машинного навчання 65 000 грн. 2. Оренда офісного приміщення з електроенергією 300 тис. грн. 3. Витрати на купівлю та встановлення мережевого обладнання 2 тис. грн. 4. Додаткові/непередбачені витрати 60 000 грн. Матеріальні: 1. Комп'ютер для обчислювальних процесів. 2. Лептопи для роботи x4. 3. Мережеве обладнання. 4. Офісне приміщення з підключеними електроенергією і мережею. 5. Марковані дата сети. Інтелектуальні: 1. Знання з побудови інфраструктури обробки великої кількості інформації (Data engineer). 2. Знання та навички створення нейронних мереж (Machine learning engineer). 3. Знання та навички написання програмного забезпечення (Software engineer). 4. Знання та навички спілкування з клієнтом та керування проектом (Project Manager).
	Перелік усіх необхідних ресурсів (фінансових, матеріальних інтелектуальниї та ін.)
6. Опис проблеми, яку вирішує проект	На сьогоднішній день гостро стоїть проблема вирішення швидкої та своєчасної діагностики захворювань, а особливо таких, що не викликають

	підозри на перших етапах свого розвитку, зокрема і захворювань шкіри.

7. Головні цілі та завдання проекту	Створити можливість виявлення можливого захворювання шкіри, що розвивається в даний момент, без втручання представників медичної допомоги та, при можливості, відправити отриманий результат на опрацювання спеціалістами у сфері діагностики захворювань шкіри.

8. Очікувані результати (Описати позитивні зміни, які відбудуться в результаті реалізації проекту після його завершення та в довгостроковій перспективі)
Надійна нейронна мережа з високим показником розпізнавання захворювань та встановлення діагнозу. В довгостроковій перспективі впровадження програми як допоміжного інструменту діагностики захворювань шкіри в сфері медицини на офіційному рівні.

Крок 1

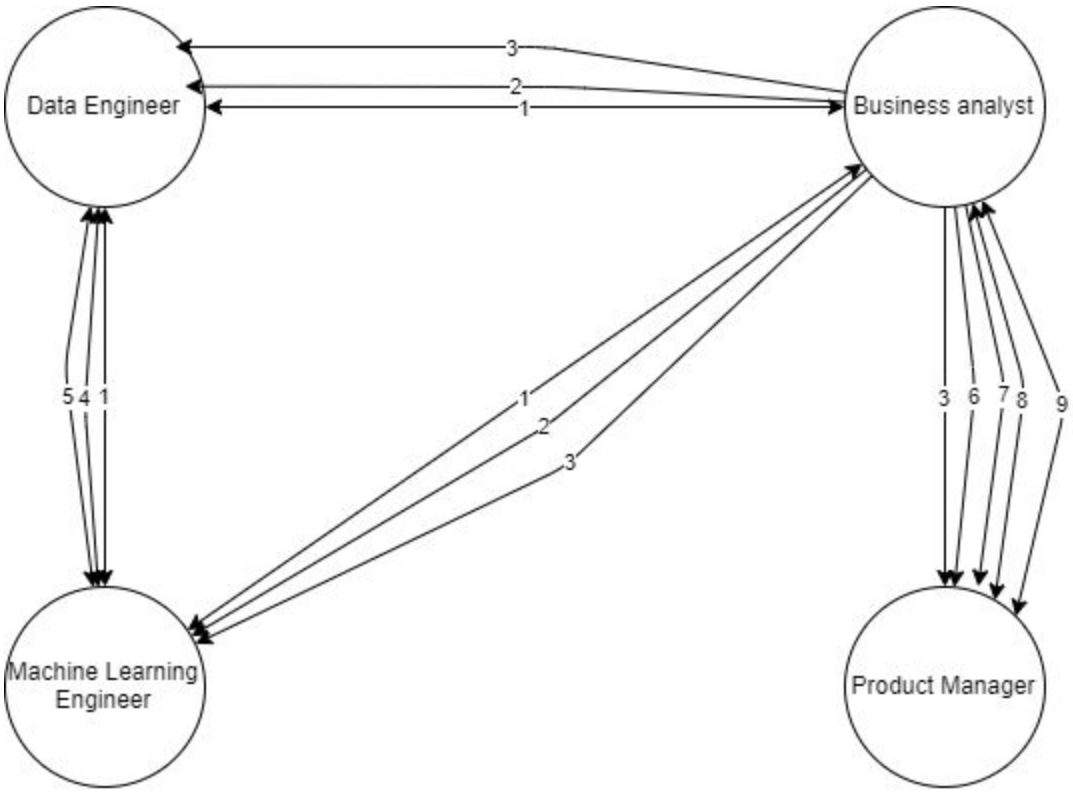
Команда стартап проекту:

1. Data engineer – Генератор ідей
2. Machine learning engineer - Виконавець
3. Software engineer - Спеціаліст
4. Project Manager - Дипломат

Крок 2

	Необхідне завдання	К-кість місяців
1	Дослідження методів реалізації	0,5
2	Видача ТЗ	0,5
3	Розподіл обов’язків	0,5
4	Розробка моделі	4
5	Тестування моделі	1,5
6	Пошук інвесторів	2
7	Юридичне забезпечення	1,5
8	Рекламна кампанія	1,5
9	Аналітика	

Крок 3



Визначення важливості факторів щодо їх вкладу у створення та реалізацію стартапу

Фактор	Вага (важливість)
Ідея	8
Підготовка бізнес плану	2
Компетентність	5
Залученість і ризики	6
Обов'язки	8

Визначення основних функцій:

- кросплатформність;
- встановлення діагнозу;
- аутентифікація користувача;
- оцінювання стану користувача;
- надання рекомендацій;

Побудова морфологічної карти:

Основні параметри	Проміжні рішення				
	1-ше	2-ше	3-ше	4-ше	5-ше
Кросплатформеність	Бот для соц. мереж/месенджерів	Мобільний додаток	Одноплатна система	Інші	
Методи самозвіту	Завантаження фото	Фото в «ріал таймі»	Завантаження з медичної картки мобільного додатку	Комбінація усіх варіантів	Інші

Аутентифікація	За допомогою електронної пошти	За допомогою соціальних мереж	Можливість під’єднувати до облікового запису з електронної пошти соціальні мережи	Інші	
Встановлення діагнозу	Визначення за допомогою штучного інтелекту	Оцінка спеціалістів	Поєдання штучного інтелекту та оцінки спеціалістів	Інші	
Надання рекомендацій	За допомогою штучного інтелекту	Оцінка спеціалістів	Поєдання штучного інтелекту та оцінки спеціалістів	Інші	

Ідея товару. Інтелектуальна система діагностики захворювань шкіри складається з комбінації методів самозвіту, а саме завантаженя фото з девайсу, виконання фото у реальному часі, можливе використання даних з медичної картки користувача, до яких надав доступ користувач. Можливе використання усіх варіантів. Система доступна для користування на мобільних платформах, у вигляді бота або як окремий електронний девайс. Для авторизації користувач може використовувати свою поштову адресу або під’єднанний обліковий запис соціальної мережі. Інтерфейс користувача є простим та інтуїтивним.

Задум товару:

1. Товар за задумом. Додаток є корисним для користувача, тому що поєднує у собі базові інструменти для відстежування власного здоров’я при наявності можливо злоякісних утворень на шкірі.
2. Товар у реальному виконанні. Додаток представлений як з інтерфейс для мобільних платформ, у вигляді бота у соц.мережах, а також як окремий електронний девайс. Сам інтерфейс користувача є простим, інтуїтивним, без складної логіки та неумісних елементів. Товар є умовно безкоштовним – за розширену функціональність (наприклад, консультація спеціаліста) можлива додаткова оплата.
3. Товар з підкріпленням. Можливе надання додаткової консультації спеціаліста розміром консультація спеціаліста + плата за надання послуг.

Головна проблема: для забезпечення здоров'я людини необхідно надавати можливість регулярної перевірки стану здоров'я.

• 1-й MVP: Лікарі і знахарі користувалися гуморальної теорією понад 1300 років. Протягом цього періоду розкриття були заборонені з релігійних міркувань, що обмежувало вивчення пухлин. Доступними методами лікування в стародавні часи і Середньовіччі були хірургія, припікання, кровопускання і обряди

• 2-й MVP: Після скасування заборони розтину в 16 столітті Андреас Везалий склав перший анатомічний довідник тіла людини, а століттям пізніше Метью Бейлі описав будову різних патологій. Гуморальну теорію довелося переглянути.

• 3-й MVP: У 1850-х німецький вчений Рудольф Вірхов виявив в пухлинах безконтрольне поділ клітин. Він назвав це явище неоплазією, а його головна праця «Клітинна патологія» став основою для розуміння причин розвитку онкологічних захворювань.

• 4-й MVP: У 1990-ті роки хірурги мінімізували втручання в здорові тканини людини. Сьогодні операції діляться на два види: відкриті і малоінвазивні. Для відкритої операції лікар робить великий розріз, щоб видалити пухлину. Для проведення малоінвазивної операції лікар робить кілька невеликих розрізів, знаходить пухлину за допомогою тонкої трубки з камерою (лапароскопа) і через інший розріз видаляє пухлину інструментами.

• 5-й MVP: Дерматоскопія - сама рання діагностика меланоми. Проводиться як за допомогою простої лупи, так і за допомогою дерматоскопи (епілюмінісцентного мікроскопа) робить прозорим роговий шар епідермісу. При цьому можна з високою ймовірністю визначити, чи є невус небезпечним чи ні на підставі системи ABCDE, запропоновану Friedman в 1985 році.

• 6-й MVP: Мобільні додатки типу SkinVision. З поширенням використання смартфонів, поширилась популярність викорстання додатків, що спрямовані на відстежування власного здоров’я.

№ з/п	**Запитання**	**Відповідь**
1	Частиною яких систем є продукт?	Продукт є частиною дерматоскопічного аналізу людини. Наприклад, як допоміжний засіб при роботі з онкологом..
2	Які функції надсистеми може виконувати продукт? Як їх з ним пов’язати?	Продукт дозволяє відстежувати власний стан здоров'я (шкіри) за допомогою методів самозвіту, що дає суб’єктивні дані стану користувача для більш глибокого розуміння спеціалістом.

3	Чи можна розділити продукт на частини?	Кожен метод самозвіту як самостійна частина: завантаження фото з девайса, фото в реальному часі та завантаження з медичної картки (особистий кабінет).
4	Чи можна об’єднати (агрегувати) кілька елементів продукту в один?	Можливе використання щоденних зборів інформації у додатку, що дозволить отримати мінімальну (урізану) інформацію про стан користувача, тобто використання невеликої анкети для нотатків про поточні візуальні зміни потенціальної меланоми.
6	Яким має бути ідеальний продукт?	Аналіз різних показників досліджуваних утворень, встановлення діагнозу, ведення медичної картки та передачі інформації спеціалісту.
7	Що відбудеться, якщо вилучити цей продукт? Чим його можна замінити?	При вилученні даного продукту, можна його функції замінити традиційним записом результатів аналізу на папері та проведенням дерматоскопічного аналізу.
8	Яким цей продукт був у минулому?	Ведення паперової медичної картки та обстеження у спеціаліста онколога за допомогою дерматоскопу та довготривалого аналізу.
9	На розвиток яких функцій було спрямоване удосконалення продукту?	Використання цифрових технологій, що поширилися за останні роки, для шівидшого встановлення можливого діагнозу та надання допомоги спеціалістом у разі термінової потреб, а також для більш швидкого та зручного запису та відстежування даних
10	Які функції залишилися	Методи самозвіту та роботи зі станом здоров'я загалом розвилися та стали оцифрованими.

	«недорозвиненими» ?	
11	Як можна натепер розвинути ці функції?	Можливий подальший розвиток функцій самозвіту шляхом додавання штучного інтелекту для обробки даних.

Відбір найцікавіших ідей, формування списку.

Ідея 1. Використання штучного інтелекту. Використання штучного інтелекту як заміна живому спеціалісту онкологу.

Ідея 2. Використання минулих показників. Можна використовувати минулі показники з історії хвороби користувача з метою покращення точності встановлення діагнозу.

Ідея 3. Часткове використання штучного інтелекту в консультаціях. Можна поєднувати діагностику зі штучним інтелектом та реальним спеціалістом-онкологом, наприклад у часи, коли спеціаліст з певних причин не зможе надати онлайн допомогу та консультацію (або у користувача немає можливості потрапити на консультацію)

Наступним кроком є об’єднання знайдених ідей, їх агрегування або комбінування.

Агрегування 1. Поєднання відстежування минулих показників здоров'я зі штучним інтелектом з метою покращення аналізу та можливості передбачення потенціальних злоякісних утворень (меланом).

Ідея 1. Використання штучного інтелекту:

- зменшення потреби у наймі спеціалістів, так як штучний інтелект зможе їх замінити;
- використання у повсякденному житті людини для регулярного трекінгу та оцінки його стану здоров'я;
- людям, які хочуть отримати детальний аналіз протягом дня, незалежно від місця їх перебування.

Ідея 2. Використання минулих медичних записів:

- такі записи як історія захворювань, поставлені діагнози, історія захворювання родичів зможуть допомогти у відстеженні та діагностиці потенціальних ракових утворень, що дозволить завчасно взяти ситуацію під контроль;
- використання у повсякденному житті людей;
- людям, які вже мають наявні діагнози, можливо знаходяться на обліку у лікаря та хочуть мати можливість постійного трекінгу їх стану, не будучи прив'заними до їх лікаря у кожний момент часу.

Ідея 3. Часткове використання штучного інтелекту в консультаціях:

- поєднання консультацій з спеціалістом та роботи штучного інтелекту, коли розмова зі спеціалістом не доступна (або поки не потрібна).
- використання у повсякденному житті людей;
- людям, які часто мають наявні утворення та хочуть мати можливість постійного трекінгу їх стану, незалежно від місця знаходження і часу.

Агрегування 1. Поєднання наявних медичних записів зі штучним інтелектом:

- оцінка передумов станів певних утворень, аналіз та передбачення їх у майбутньому;
- використання у повсякденному житті людини;
- люди, які хочуть відстежувати та аналізувати власний стан як можна краще.

Розроблення ринкової стратегії проекту

Таблиця 15

Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Люди, які мають деякі проблеми у сфері онкології, або потенційно можуть їх отримати і бажають мати можливість відстежувати свій стан.	Так як продукт є автоматизованою версією традиційних методів відстежування власного стану, а зараз велику роль у житті людей грають смартфони, то споживачі є готовими сприйняти продукт.		Наразі існує невелика кількість продуктів, що спрямовані на відстежування стану людини.	Враховуючи доволі невисоку конкуренцію, вхід у сегмент не є найскладнішим, але й простим не являється, адже існують стійкі лідери ринку. Але при правильному маркетингу можна

					полегшити вхід.
Які цільові групи обрано: люди, що бажають слідкувати за власним станом здоров’я.					

Таблиця 16

Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Стратегія спеціалізації	Концентроване	Акцент на оригінальних можливостях застосунку та клієнт-орієнтованість.	Стратегія диференціації

Таблиця 17

Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів,	Чи буде компанія копіювати основні	Стратегія конкурентної поведінки*

		або забирати існуючих у конкурентів?	характеристики товару конкурента, і які?	
	Ні, подібні проекти вже існують	В основному – шукати нових за рахунок вигідніших умов використання за стосунку.	Так, копіювання буде пристунє, а саме: використання методів штучного інтелекту для діагностики потенційних утворень.	Стратегія наслідування лідеру (діагностика), створення найбільш сприятливих умов використання («інфраструктура » проекту)

Таблиця 18

Визначення стратегії позиціонування

№ п / п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	- зручність у користуванні; - точність діагностики;	Стратегія диференціації	- Використання провідних технологій - Орієнтованість на споживача	Розумний, зручний, мобільний моніторинг

	- персоналізація використання			

Розроблення маркетингової програми стартап-проекту

Таблиця 19

Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Стеження за станом утворень на шкірі	Можливість легко та зручно стежити за даним про свій стан та його аналізувати	- впровадження штучного інтелекту; - прийняття до уваги вже існуючих показників при аналізі; - можливість аналізу зв'язку уже наявних діагнозів та отриманих результатів діагностики у реальному часі.

Таблиця 20

Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Стеження за та аналіз стану людини, на основі даних, що вона надає застосунку та даних з медичної картки.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Збір даних за допомогою методів самоаналізу 2. Збір даних з медичної картки.		

	3. Аналіз зібраних даних 4. Можливе передбачення стану утворення на шкірі або діагноз/аналіз його поточного стану.		
	Якість: швидкість та коректність у використанні, логічність та простота інтерфейсу користувача.		
III. Товар із підкріпленням	До продажу: відсутнє		
	Можливе надання додаткової консультації спеціаліста розміром консультація спеціаліста + плата за надання послуг.		

Таблиця 21

Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	~30$ / місяць	~0-30$ / місяць	Середній (~10000-30000 грн/міс)	- 0 грн (для трекінгу стану) - консультація спеціаліста + плата за надання послуг (15$ / місяць)

Таблиця 22

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Збут через Інтернет-платформи	- розміщення застосунку на власному маркеті; - реклама застосунку на головних сторінках;	1	Залучена система збуту через сторонніх посередників

Таблиця 23

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонуванн я	Завдання рекламного повідомлення	Концепція рекламного звернення
	Часте використання соціальних мереж, що призводить до деяких	Соціальні мережі	За сферою застосування; показники якості; позитивні особливості технології	Інформування споживачів; Презентація на конференціях; Інформування в тематичних	Даний продукт має гнучку архітектуру, що може підлаштовуватис я під бажання клієнтів, має

	психологічних розладів			засобах масової інформації; Стимулювання продажу;	високі характеристики та надійність.

Концепція маркетингових комунікацій

Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 24

Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	5
2	Загальний обсяг продаж, грн/ум.од	5 мільярдів доларів
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Обов’язкова підтримка регламенту GDPR
6	Середня норма рентабельності в галузі (або по ринку), %	

Таблиця 25

Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Трекінг стану здоров’я за допомогою мобільних та веб-застосунку.	Люди, які хочуть стежити за своїм поточним станом здоров’я.	- фінансове положення споживача, можливість його купляти продукт(додаткові консультації); - технології та пристрої які він використовує;	**До продукції:** - легкий у використанні; - швидка робота; - візуальне відображення даних. **До компанії-постачальника:** - оперативна технічна підтримка; - регулярне оновлення(навчання) застосунку.

Таблиця 26

Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Наявність лідера (монополіста) на ринку діагностики онкології шкіри,а також багатьох спроб молодих стартапів увійти до ринку, що підвищує рівень конкуренції	Створення оригінальних характеристик застосунку у порівнняні з вже усноючими конкурентами.
2	Використання особистих даних	Так як для аналізу та відстеження стану використовуются такі дані, що розцінюються чутливими та потрапляють під регламент GDPR, будь-яке злиття інформації може призвести до втрати клієнтів та негативногої репутації	Забезпечення високого рівня безпеки та шифрування даних.

3	Правові чинники у роботі з даними користувача	Можливе накладання певних обмежень у використанні та збереженні даних користувачів з різних країн	Консультування зі спеціалістами правової сфери.

Таблиця 27

Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Популяризація здорового образу життя	З підвищенням популярності стеження за своїм здоровям, потенційні користувачі звертаються по допомогу до застосунків, що дозволяють підстежувати його стан, що підвищує попит таких застосунків у даній сфері.	Стеження за популярними концептами та орієнтування на споживача.

2	Збільшення поширенності онкологічних захворювань	При підвищенні поширенності окологічних захворювань, що спостерігається все більше і більше з кожним роком, зростає усвідомленість людей у стеженнні за власним станом здоров’я.	Збільшення можливостей для аналізу та моніторингу стану здоров’я.

Таблиця 28

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції **- досконала конкуренція**	Ціна подібних систем не є фіксованою, тож кожен сам обирає ціну	Встановлення ціни на використання додатку за власним бажанням.
2. За рівнем конкурентної боротьби **- міжнародний**	Платіжні компанії розповсюджені о всьому світу.	Необхідність правильного таргетингу аудиторії через різні

		культурні та соціальні особливості.
3. За галузевою ознакою **- внутрішньогалузева**	Конкуренція не виходить за рамки онлайн платежів.	Аналіз ринку даної галузі з метою покращення власного продукту.
4. Конкуренція за видами товарів: **- товарно-родова**	Схожі системи присутні, але кожна з них займає різну тематику.	Розвиток функціоналу для збільшення кількості тематик.
5. За характером конкурентних переваг **- нецінова**	Застосунки модернізуються з використанням спеціально розроблених методик, поширюється використання різних технологій, що покращують їх використання.	Необхідність креативного та наукового підходу до розробки можливостей застосунку.
6. За інтенсивністю **- не марочна**	Більш цінніше концентруватися на якості продукту, а не на рекламу.	Пріоритет на якість та безпеку розроблюваного продукту.

Таблиця 29

Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)

1	Використання провідних технологій	Використання таких технологій як штучний інтелект, впровадження розширеного збору даних, що допоможе покращити аналіз та діагностику стану людини.
2	Орієнтованість на споживача	Прислухання до споживачів, аналіз їх потреб та впровадження потрібних функцій у застосунок.
3	Freemium політика	Така політика дозволить отримати основну частину функцій безкоштовно, а за додаткову консультацію з спеціалістом треба буде доплатити. Таким чином користувач може поширювати функціональність застосунку за потреби.

Таблиця 30

Порівняльний аналіз сильних та слабких сторін «Diagnostic»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з empatr						
			–3	–2	–1	0	+1	+2	+3
1	Використання провідних технологій	18							+
2	Орієнтованість на споживача	15						+	
3	Безкоштовна діагностика	11				+			

Таблиця 31

SWOT- аналіз стартап-проекту

Сильні сторони: - використання провідних технологій;	**Слабкі сторони:** - ціна за консультативні послуги;

- креативний підхід до розробки функціональності; - клієнт-орієнтованість; - технічна підтримка.	- навантаженість даними;
Можливості: - розширення функціональності для аналізу та діагностики стану; - повна автоматизація процесу аналізу та дігностики стану утворення; - збільшення поширеності онкологічних захворювань шкіри.	**Загрози:** - впровадження законів, що обмежують використання персональних даних користувачів; - труднощі впровадження функціональності, що базується на штучному інтелекті; - труднощі в пошуку кваліфікованих спеціалістів для співпраці

Таблиця 32

Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Презентація на тематичних заходах для того, щоб про продукт дізналися	Дуже висока	2 місяці
2	Пропонувати продукт компаніям	Висока	6 місяців
3	Бізнес-ангел	Дуже низька	1-24 місяця

Бізнес-план

Мета: довести можливості ефективного організаційного та ресурсного забезпечення проекту; продемонструвати, що запропоновані організаційні рішення дозволять ефективно реалізувати стартап-проект. Скласти календарний план-графік реалізації проекту за формою, наведеною в табл. 33.

Таблиця 33

Календарний план-графік реалізації стартап-проекту

№ з/п	Етапи реалізації	Період реалізації проекту				
		0-й рік[1]				1-й рік
		1-й кв.	2-й кв.	3-й кв.	4-й кв.	
1.	Проведення НДДКР					
2.	Створення компанії					
3.	Придбання нематеріальних активів, отримання дозвільних документів тощо					
4.	Оренда приміщення					
5.	Придбання обладнання, програмного забезпечення					
6.	Передвиробничі маркетингові дослідження					
7.	Найм потрібних кадрів					

1

8.	Створення та видача технічного завдання					
9.	Розподіл обов`язків					
10.	Розробка програмного забезпечення					
11.	Тестування програмного забезпечення					
13.	Пошук інвесторів					
13.	Рекламна кампанія					
14.	Аналітика					
15.	Запуск продукту					

Організаційний план.

Інвестиційне забезпечення:

1) Продукт буде презентований на різних тематичних конференціях (як локальних, так і міжнародних). Це дасть змогу не тільки залучити інвесторів, а і показати продукт на весь світ, що дасть змогу отримати фідбек від людей, які працюють в цій сфері вже багато років.
2) Якщо не вдасться знайти інвесторів на конференціях, то можна самостійно надсилати інформацію про продукт в ті компанії, які пов’язані з медичної діагностикою.
3) Спробувати знайти бізнес-ангела. Бізнес-ангел – це фіз. особа, яка готова вкладати власні кошти в стартап на нульовому або початковому етапі, в

обмін на частку в майбутньому підприємстві. Працювати з однією фіз. особою набагато легше ніж з компанію чи групою інвесторів.

Вибір договору інвестування:

1) Якщо інвестор знайдеться лише один або хтось захоче підписати ексклюзивний контракт, то буде обраний інвестиційний договір. Він дозволяє детально прописати ті цілі, на які здійснюються капіталовкладення, закріпити основні параметри і орієнтири стартаперів. У ньому можна встановити і закріпити якусь схему звітності перед інвестором, позначити кінцевий результат і терміни його досягнення, прописати обов’язки сторін тощо.
2) Якщо інвесторів буде кілька то буде укладено договір товариства. Він розроблений для того, щоб у всіх інвесторів були рівні права. Подібним способом фіксуються порядки користування майном загального підприємства, способи комунікації серед інвесторів і учасників товариства, порядок розірвання договору, виплат компенсацій тощо.

Одним з варіантів залучення інвестицій був краудфандинг.

Краудфандинг (громадське фінансування) — це співпраця людей, які добровільно об'єднують свої гроші чи інші ресурси разом, як правило через інтернет, аби підтримати зусилля інших людей або організацій.

Але такий варіант не є перспективним для проекту з двох причин:

1) Тематика продукту є вузько направленою, тому вона не буде цікава звичайним людям, що за хочуть кудись вкласти гроші;
2) Нестабільність інвестувань, через відсутність договору, тобто в якийсь час можуть знизитися або взагалі пропасти інвестиції.

Робота з інвестором:

Основними вимогами з нашого боку буде наступне:

1) Фінансування буде фіксоване у часі та терміні, не зважаючи на різні обставини;
2) Трафік (кількість платежів) будуть переносити на наш продукт поступово. Наприклад, після завершення етапу розробки інвестор не одразу переведе всі свої платежі на нас, а лише 1%. Після тижня роботи, якщо не було виявлено помилок, то трафік підвищується до 10% и тд.
3) Інвестор не повинен впливати на шлях розвитку продукту, тобто він буде отримувати лише кошти з прибутків.

Обов’язки та права продукту:

1) Продукт бере не себе усі збитки, у разі підтвердження факту їх провини.
2) Якщо у продукта не буде достатньо коштів, то може бути запропонований варіант на дешевшу діагностику протягом визначеного терміну.
3) Критична підтримка надається 24/7, всі інші питання у робочий час.

Компанії, яким би був цікавий данний стартап:

1) MIT Media Lab — міждисциплінарна дослідницька лабораторія Массачусетського технологічного інституту (англ. MIT), що займається проектами, спрямованими на зближення технологій, мультимедіа, науки, мистецтва і дизайну.
2) "Лабораторії DataRoot (DRL) створюють та впроваджують системи, що працюють на AI, в різних вертикалях, щоб допомогти нашим клієнтам ефективно працювати"

Висновок до розділу 5

Проаналізувавши всі результати описані в даному розділі можна зробити висновок що проект має перспективи у сфері стартапів та має високі шанси на успішну комерціалізацію. Наявність конкуренції у вигляді існуючих на ринку продуктів вказує на те, що наявний попит на продукти даної тематики.

Динаміка ринку не зростає вкрай швидко, а являється більше стабільною, ніж зростаючою. Це дає можливість на швидкий розвиток проекту на ринку та захоплення високих позицій при умові успішного впровадження нового функціоналу, що відсутній у основних конкурентів.

З огляду на потенційні групи клієнтів є можливість розвивати проект у кількох напрямках: у якості веб технології зі стандартним алгоритмом застосування платної підписки, або ж запуск виробництва з ціллю продажу продукту компаніям із сегменту медичних, в особливості онкологічних, досліджень. Дана ситуація дозволяє мати запасний план для виходу на ринок у разі невдалої спроби зайняти місце в одному з його сегментів.

Подальша імплементація проекту є доцільною, спираючись на рівень попиту у тому чи іншому сегменті ринку, враховуючи рівень попиту відповідно.

ВИСНОВКИ

У даній роботі було розглянуто сферу машинного навчання, проаналізовано основні підходи і методи. Також було описано переваги та проблеми, які виникають у процесі застосування розглянутих підходів у сфері обробки медичних зображень.

На базі виконаного дослідження, описаного у розділі 1, було сформовано план використання тих чи інших методів для задачі обробки медичних зображень шкіри, складено загальний теоретичний опис підходів, які використовуються у роботі, зокрема у дослідженні класифікації зображень злоякісних уражень шкіри.

У розділі 2 розглянуто основні архітектури нейронних мереж, що мають перспективу отримати хороші результати при умові їх імплементування у сферу обробки та аналізу медичних зображень.

На базі характеризації моделей було сформовано план та подальші кроки дослідження обробки медичних зображень та класифікації злоякісних уражень на шкірі людини, а саме: обрано найбільш перспективні, у даній сфері, архітектури нейронних мереж для тренування та тестування, складено план їх тренування та тестування, план отримання даних з результатів дослідження.

У розділі 3 було описано застосований метод обробки медичних зображень, а саме класифікації на них злоякісних уражень. Було описано основні характеристики методу, його властивості та застовані архітектури нейронних мереж.

Також було описано і сформовано конфігурації апаратного забезпечення для виконання досліджень на базі відкритої платформи Kaggle для тренування і тестування нейронних моделей. Надано опис деталей створених конфігурацій.
Сформовано конфігурації застосованих архітектур нейронних мереж для отримання необхідних результатів та надано їх (конфігурацій) опис. Створено допоміжні модулі для отримання даних результатів роботи використаних нейронних мереж.

У розділі 4 описані результати роботи методу обробки медичних зображень людини на основі нейронної мережі на основі результатів дослідження процесу класифікації меланом на знімках уражень шкіри на реальних медичних зображеннях пацієнтів, із визначенням основних параметрів їх роботи для прикладного застосування у сфері класифікації шкірних захворювань загалом і зокрема меланом.

Для досягнення даної мети було виконано **наступні основні завдання:**

- підготовлено датасет із медичними даними на основі відкритих даних 2018-2020 років із https://challenge2020.isic-archive.com/
- сконфігурувано дослідницьку інфраструктуру на основі віддалених хмарних ресурсів і залученням графічних прискорювачів (graphic processing units — GPU) і прискорювачів тензорних обчислень (tensor processing units — TPU),
- виконано тренування на кількох стандартних (DenseNet121, MobileNetV2, MobileNetV2, InceptionResNetV2, ResNet101, ResNet101V2, VGG16, Xception) і новітніх (EfficientNetB0-B6) моделях поглибленого навчання із використанням методу крос-валідації (k-fold cross validation) і штучного збільшення кількості варіантів тренувальних зображень (data augmentation - DA),
- виконано тестування отриманих тренованих моделей із залученням методу штучного збільшення кількості варіантів тестових зображень (post-training/testing time data augmentation - TTA),
- виконано порівняльний аналіз отриманих даних на різних інфраструктурах, моделях і їх параметрах для дослідження впливу типу інфраструктури (TPU/GPU), типу і розміру моделей на час і точність прогнозування за стандартними метриками,
- сформулювано рекомендації щодо можливості практичного застосування отриманих результатів в якості методу обробки медичних зображень

людини на основі нейронної мережі у вигляді сервісу із локальним або віддаленим доступом.

У розділі 5 було проведено аналіз ринку та зроблено висновок що проект має перспективи у сфері стартапів та має високі шанси на успішну комерціалізацію. Наявність конкуренції у вигляді існуючих на ринку продуктів вказує на те, що наявний попит на продукти даної тематики.

Динаміка ринку не зростає вкрай швидко, а являється більше стабільною, ніж зростаючою. Це дає можливість на швидкий розвиток проекту на ринку та захоплення високих позицій при умові успішного впровадження нового функціоналу, що відсутній у основних конкурентів.

З огляду на потенційні групи клієнтів є можливість розвивати проект у кількох напрямках: у якості веб технології зі стандартним алгоритмом застосування платної підписки, або ж запуск виробництва з ціллю продажу продукту компаніям із сегменту медичних, в особливості онкологічних, досліджень. Дана ситуація дозволяє мати запасний план для виходу на ринок у разі невдалої спроби зайняти місце в одному з його сегментів.

Подальша імплементація проекту є доцільною, спираючись на рівень попиту у тому чи іншому сегменті ринку, враховуючи рівень попиту відповідно.

СПИСОК ЛІТЕРАТУРИ

1. Schmidhuber J. Deep learning in neural networks: An overview / J. Schmidhuber // Neural Networks. – 2015. – V. 61. – P. 85–117. – doi: 10.1016/j.neunet.2014.09.003

2. Understanding of Convolutional Neural Network (CNN) [Електронний ресурс]. – 2018. – Режим доступу: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

3. Z. J. Wang et al. CNN EXPLAINER: Learning Convolutional Neural Networks with Interactive Visualization / IEEE Transactions on Visualization and Computer Graphics. – 2020. – doi: 10.1109/TVCG.2020.3030418.

4. Baldi P. The capacity of feedforward neural networks / P. Baldi, R. Vershynin // Neural Networks. – 2019. – V. 116. – P. 288–311. – doi: 10.1016/j.neunet.2019.04.009

5. Dai. D. et al. Understanding the Feedforward Artificial Neural Network Model From the Perspective of Network Flow [Електронний ресурс] // ArXiv˚ abs/1704.08068.˚ –˚ 2017.˚ –˚ Режим˚ доступу: https://arxiv.org/ftp/arxiv/papers/1704/1704.08068.pdf

6. Sherstinsky A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network / A. Sherstinsky // Physica D: Nonlinear Phenomena. – 2020. – V. 404. – doi: 10.1016/j.physd.2019.132306

7. Yi L. et al. Graph-FCN for Image Semantic Segmentation / Advances in Neural Networks. – 2019. – P. 97–105. – doi: 10.1007/978-3-030-22796-8_11

8. Shelhamer E. et al. Fully Convolutional Networks for Semantic Segmentation / *IEEE* Transactions on Pattern Analysis and Machine Intelligence. – 2017. – V. 39. – № 4. – P. 640–651. – doi: 10.1109/TPAMI.2016.2572683

9. Triple Stratified KFold with TFRecords [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/cdeotte/triple-stratified-kfold-with-tfrecords

10. Detecting Melanoma Cancer using Deep Learning [Електронний

ресурс]. – Режим доступу: https://github.com/heytanay/melanoma-cancer-detection

11. EfficientNet: как масштабировать нейросеть с использованием AutoML [Електронний ресурс]. – Режим доступу: https://neurohive.io/ru/novosti/efficientnet-kak-masshtabirovat-nejroseti-s-pomoshhju-automl/

12. Kim M., Yun J., Cho Y., et al. Deep Learning in Medical Imaging / Neurospine. – 2019. – V. 16. – № 4. – P. 657–668. – doi:10.14245/ns.1938396.198

13. Razzak M.I. Deep Learning for Medical Image Processing: Overview, Challenges and the Future / M.I. Razzak, S. Naz, A. Zaib // Springer. – 2018. – V. 26. – P. 323-350 – doi: 10.1007/978-3-319-65981-7_12

14. SIIM-ISIC Melanoma Classification Identify melanoma in lesion images [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/c/siim-isic-melanoma-classification

15. Melanoma TFRecord 256x256 [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/cdeotte/melanoma-256x256

16. ISIC 2019 [Електронний ресурс]. – Режим доступу: https://challenge2019.isic-archive.com/

17. The ISIC 2020 Challenge Dataset [Електронний ресурс]. – Режим доступу: https://challenge2020.isic-archive.com/

18. EfficientNet Keras (and TensorFlow Keras) [Електронний ресурс]. – Режим доступу: https://pypi.org/project/efficientnet/

19. Tan M. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [Електронний ресурс] / M. Tan, Q.V Le // *ICML*. – 2019. – Режим доступу: https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet

20. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [Електронний ресурс]. – Режим доступу:

https://arxiv.org/abs/1905.11946

21. ICML° 2019° [Електронний° ресурс].° –° Режим° доступу: https://icml.cc/Conferences/2019

22. Triple Stratified Leak-Free KFold CV [Електронний ресурс]. – Режим доступу:°

https://www.kaggle.com/c/siim-isic-melanoma-classification/discussion/165526

23. Huang G. Densely Connected Convolutional Networks [Електронний ресурс] / G. Huang, Liu Z., Maarten L. // IEEE Conference on Computer Vision and Pattern° Recognition.° –° 2017.° –° P.° 4700–4708.° –° Режим° доступу: https://arxiv.org/abs/1608.06993

24. Szegedy C. et al. Rethinking the Inception Architecture for Computer Vision / *IEEE* Conference on Computer Vision and Pattern Recognition (CVPR). – 2016 – P. 2818–2826. – doi: 10.1109/CVPR.2016.308

25. Szegedy C. et al. 2017. Inception-v4, inception-ResNet and the impact of residual connections on learning [Електронний ресурс] / In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI'17). – AAAI Press. – 2017. – P. 4278–4284. – Режим доступу: https://arxiv.org/abs/1602.07261

26. A Simple Guide to the Versions of the Inception Network [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

27. Howard, A. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Електронний ресурс] / *ArXiv* abs/1704.04861. – 2017. – Режим доступу: https://arxiv.org/abs/1704.04861

28. Sandler M. et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks / IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 2018. – P. 4510– 4520. – doi: 10.1109/CVPR.2018.00474

29. He K. et al. Deep Residual Learning for Image Recognition / IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016. – P. 770-778. – doi: 10.1109/CVPR.2016.90.

30. He K. et al. Identity Mappings in Deep Residual Networks / Springer/ Lecture Notes in Computer Science. – 2016. – V. 9908. – P. 630-645. – doi: 10.1007/978-3-319-46493-0_38

31. -Simonyan, K. Very Deep Convolutional Networks for Large-Scale Image Recognition [Електронний ресурс] / K. Simonyan, A. Zisserman. – 2015. – Режим доступу: https://arxiv.org/abs/1409.1556

32. Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions / IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – P. 1800–1807. – doi: 10.1109/CVPR.2017.195.

33. A Gentle Introduction to the ImageNet Challenge (ILSVRC) [Електронний° ресурс].° –° Режим° доступу:

https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc/

34. ImageNet Overview [Електронний ресурс]. – Режим доступу: http://image-net.org/about-overview

Метод обробки медичних зображень людини на основі нейронної мережі

Основна частину коду програми

Д1

Аркушів 15

Київ 2020

KFold перехресна валідація з використанням **TFRecords**

Це блокнот для **SIIM-ISIC Melanoma Classification** на платформі **Kaggle,** що демонструє **KFold** з **TFRecords.** Нижче є багато змінних конфігурації, що дозволяють експериментувати. Хмарна платформа дозволяє використовувати **GPU** або **TPU.** Є можливість обирати які розміри зображень завантажуються, які ефективні мережі використовуються та чи використовуються зовнішні дані. Також можна експериментувати з різним збільшенням даних, архітектурою моделі, втратами, оптимізаторами та графіками навчання. **TFRecords** містять метадані, тому їх можна ввести це у свою нейронну мережу.

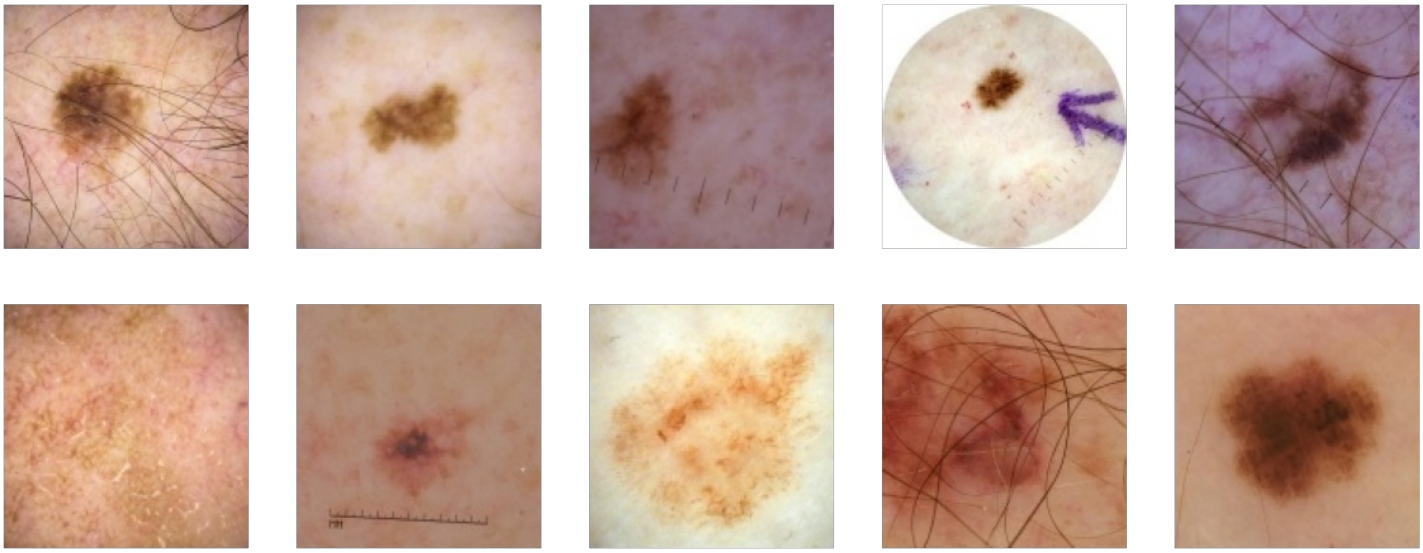
Класифікація меланоми **SIIM-ISIC** від **Kaggle**

У цьому блокноті ставиться за мету виявити меланому на знімках уражень шкіри. Це доволі складне завдання класифікації зображень, як видно з перегляду зразків зображень нижче. Нижче наведено приклади зображень шкіри з меланомою та без неї.

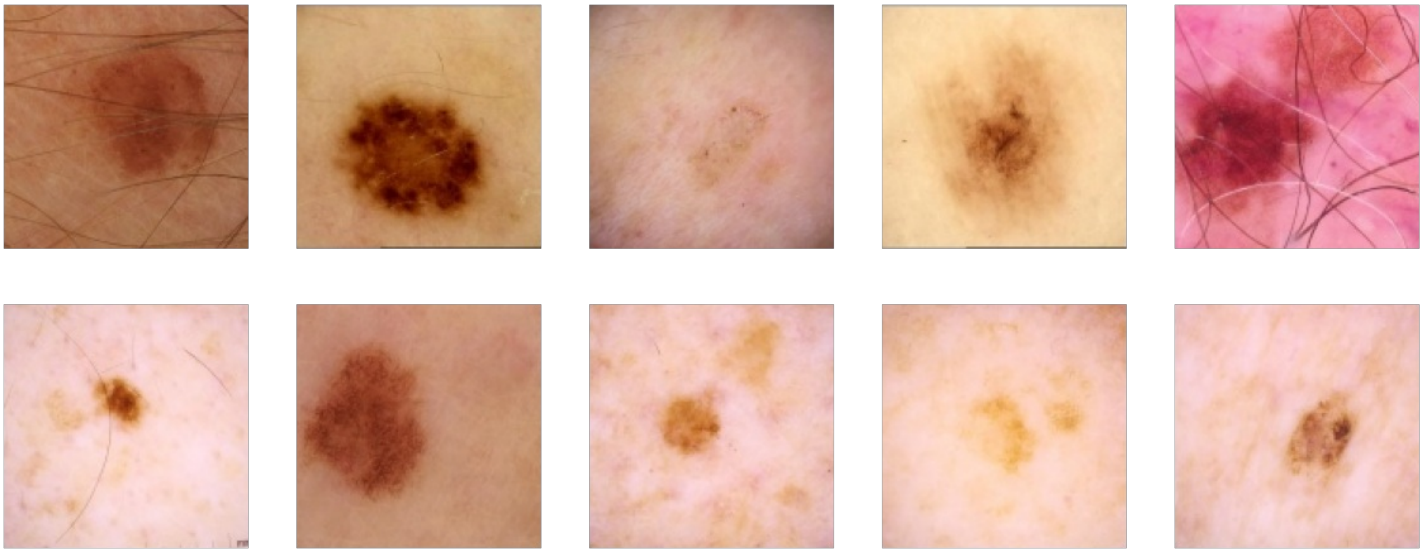
In [2]:

```
import cv2, pandas as pd, matplotlib.pyplot as plt
train = pd.read_csv('../input/siim-isic-melanoma-classification/train.csv')
print('Examples WITH Melanoma')
imgs = train.loc[train.target==1].sample(10).image_name.values
plt.figure(figsize=(20,8))
for i,k in enumerate(imgs):
    img = cv2.imread('../input/jpeg-melanoma-128x128/train/%s.jpg'%k)
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    plt.subplot(2,5,i+1); plt.axis('off')
    plt.imshow(img)
plt.show()
print('Examples WITHOUT Melanoma')
imgs = train.loc[train.target==0].sample(10).image_name.values
plt.figure(figsize=(20,8))
for i,k in enumerate(imgs):
    img = cv2.imread('../input/jpeg-melanoma-128x128/train/%s.jpg'%k)
    img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    plt.subplot(2,5,i+1); plt.axis('off')
    plt.imshow(img)
plt.show()
```

```
Examples WITH Melanoma
```



```
Examples WITHOUT Melanoma
```



In [3]:

```
import pandas as pd, numpy as np
from kaggle_datasets import KaggleDatasets
import tensorflow as tf, re, math
import tensorflow.keras.backend as K
#import efficientnet.tfkeras as efn
from sklearn.model_selection import KFold
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
```

Для того щоб отримати правильну перехресну перевірку з, загалом, значущою оцінкою **cross-validation** необхідно обрати однакові розміри картинок в датасетах та архітектуру моделі для кожного набору даних.

- **DEVICE –** використовуємий для навчання нейронної моделі апаратний процесор, в даному випадку **GPU** або **TPU**
- **SEED –** означає стан використання генератора псевдовипадкових чисел. Якщо використовувати одне й те саме значення **SEED**, то алгоритм буде видавати однаковий набір результатів кожного разу. Різне значення **SEED** дозволяє продукувати різне потрійне **kfold** розшарування.
- **FOLDS –** кількість наборів (як правило, послідовних) записів набору даних
- **IMG_SIZES –** розмір зображень в кожному наборі **FOLD**
- **INC2019 –** включає у себе нову частину даних аналогічного змагання **2019** року.
- **INC2018 –** друга частина даних аналогічного змагання **2019** року, що складається з даних **2017** та **2018** років
- **BATCH_SIZES –** розмір групи даних для кожного набору даних **FOLD**. Для найкращої швидкості використовується максимально велике значення **BATCH_SIZE**, яке здатна обробити апаратна частина **GPU**, або ж **TPU**
- **EPOCHS –** максимальна кількість епох. У кожному наборі записів зберігається та використовується найкраща модель епохи.
- **EFF_NETS –** варіант архітектури моделі сімейства **EfficientNets**, яку використовує кожний набір даних **FOLD**. У коді номер додається до змінної, тобто якщо номер дорівнює **0**, то буде використовуватись архітектура **EfficientNetB0**.
- **WGTS –** це вага під час складання наборів даних для прогнозування тестового набору. Для хорошої узгодженності рекомендується використовувати однакову вагу.
- **TTA –** збільшення даних тестування. Кожне тестове зображення випадково доповнюється і передбачається час **TTA**, в результаті використовується середнє передбачення. **TTA** також застосовується для **OOF** під час перевірки.

In [4]:

```
DEVICE = "TPU" or "GPU"

# USE DIFFERENT SEED FOR DIFFERENT STRATIFIED KFOLD
SEED = 42

# NUMBER OF FOLDS. USE 3, 5, OR 15
```

```
FOLDS = 5

# WHICH IMAGE SIZES TO LOAD EACH FOLD
# CHOOSE 128, 192, 256, 384, 512, 768
#IMG_SIZES = [384,384,384,384,384]

#IMG_SIZES = [128,128,128,128,128]
SIZE = 128
IMG_SIZES = [SIZE,SIZE,SIZE,SIZE,SIZE]

# INCLUDE OLD COMP DATA? YES=1 NO=0
INC2019 = [1,1,1,1,1]
INC2018 = [0,0,0,0,0]

# BATCH SIZE AND EPOCHS
BATCH_SIZES = [32]*FOLDS
EPOCHS = [12]*FOLDS

# WHICH EFFICIENTNET B? TO USE
MODEL = 1
EFF_NETS = [MODEL,MODEL,MODEL,MODEL,MODEL]

# WEIGHTS FOR FOLD MODELS WHEN PREDICTING TEST
WGTS = [1/FOLDS]*FOLDS

# TEST TIME AUGMENTATION STEPS
TTA = 11
```

In [5]:

```
if DEVICE == "TPU":
    print("connecting to TPU...")
    try:
        tpu = tf.distribute.cluster_resolver.TPUClusterResolver()
        print('Running on TPU ', tpu.master())
    except ValueError:
        print("Could not connect to TPU")
        tpu = None

    if tpu:
        try:
            print("initializing  TPU ...")
            tf.config.experimental_connect_to_cluster(tpu)
            tf.tpu.experimental.initialize_tpu_system(tpu)
            strategy = tf.distribute.experimental.TPUStrategy(tpu)
            print("TPU initialized")
        except _:
            print("failed to initialize TPU")
    else:
        DEVICE = "GPU"

if DEVICE != "TPU":
    print("Using default strategy for CPU and single GPU")
    strategy = tf.distribute.get_strategy()

if DEVICE == "GPU":
    print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU'
)))


AUTO     = tf.data.experimental.AUTOTUNE
REPLICAS = strategy.num_replicas_in_sync
print(f'REPLICAS: {REPLICAS}')
```

```
connecting to TPU...
Running on TPU  grpc://10.0.0.2:8470
initializing  TPU ...
TPU initialized
REPLICAS: 8
```

Крок 1: Попередня обробка

Крок 1: Попередня обробка

Попередня обробка вже виконана і збережена в **TFRecords.** Тут необхідно обрати, який розмір завантажувати. Ми можемо використовувати **128x128, 192x192, 256x256, 384x384, 512x512, 768x768,** змінивши змінну `IMG_SIZES` у попередньому розділі коду.

In [6]:

```
%%time
GCS_PATH = [None]*FOLDS; GCS_PATH2 = [None]*FOLDS
for i,k in enumerate(IMG_SIZES):
    GCS_PATH[i] = KaggleDatasets().get_gcs_path('melanoma-%ix%i'%(k,k))
    GCS_PATH2[i] = KaggleDatasets().get_gcs_path('isic2019-%ix%i'%(k,k))
files_train = np.sort(np.array(tf.io.gfile.glob(GCS_PATH[0] + '/train*.tfrec')))
files_test  = np.sort(np.array(tf.io.gfile.glob(GCS_PATH[0] + '/test*.tfrec')))
```

```
CPU times: user 131 ms, sys: 10.4 ms, total: 142 ms
Wall time: 3.37 s
```

Крок 2: Збільшення даних

Цей блокнот використовує збільшення даних: обертання, масштабування, масштабування, зміщення. Цей ноутбук також використовує горизонтальне перевертання, відтінок, насиченість, контраст, збільшення яскравості.

Крім того, ми можемо вирішити використовувати зовнішні дані, змінивши змінні `INC2019` та `INC2018` у попередньому розділі коду. Ці змінні відповідно вказують, чи потрібно завантажувати дані за минулий рік **2019** та / або дані **2018 + 2017** року.

In [7]:

```
ROT_ = 180.0
SHR_ = 2.0
HZOOM_ = 8.0
WZOOM_ = 8.0
HSHIFT_ = 8.0
WSHIFT_ = 8.0
```

In [8]:

```
def get_mat(rotation, shear, height_zoom, width_zoom, height_shift, width_shift):
    # returns 3x3 transformmatrix which transforms indicies

    # CONVERT DEGREES TO RADIANS
    rotation = math.pi * rotation / 180.
    shear    = math.pi * shear    / 180.

    def get_3x3_mat(lst):
        return tf.reshape(tf.concat([lst],axis=0), [3,3])

    # ROTATION MATRIX
    c1   = tf.math.cos(rotation)
    s1   = tf.math.sin(rotation)
    one  = tf.constant([1],dtype='float32')
    zero = tf.constant([0],dtype='float32')

    rotation_matrix = get_3x3_mat([c1,   s1,   zero,
                                   -s1,  c1,   zero,
                                   zero, zero, one])
    # SHEAR MATRIX
    c2 = tf.math.cos(shear)
    s2 = tf.math.sin(shear)

    shear_matrix = get_3x3_mat([one,  s2,   zero,
                                zero, c2,   zero,
                                zero, zero, one])
    # ZOOM MATRIX
    zoom_matrix = get_3x3_mat([one/height_zoom, zero,           zero,
```

```
                                zero,            one/width_zoom, zero,
                                zero,            zero,           one])
    # SHIFT MATRIX
    shift_matrix = get_3x3_mat([one,  zero, height_shift,
                                zero, one,  width_shift,
                                zero, zero, one])

    return K.dot(K.dot(rotation_matrix, shear_matrix),
                 K.dot(zoom_matrix,     shift_matrix))


def transform(image, DIM=256):
    # input image - is one image of size [dim,dim,3] not a batch of [b,dim,dim,3]
    # output - image randomly rotated, sheared, zoomed, and shifted
    XDIM = DIM%2 #fix for size 331

    rot = ROT_ * tf.random.normal([1], dtype='float32')
    shr = SHR_ * tf.random.normal([1], dtype='float32')
    h_zoom = 1.0 + tf.random.normal([1], dtype='float32') / HZOOM_
    w_zoom = 1.0 + tf.random.normal([1], dtype='float32') / WZOOM_
    h_shift = HSHIFT_ * tf.random.normal([1], dtype='float32')
    w_shift = WSHIFT_ * tf.random.normal([1], dtype='float32')

    # GET TRANSFORMATION MATRIX
    m = get_mat(rot,shr,h_zoom,w_zoom,h_shift,w_shift)

    # LIST DESTINATION PIXEL INDICES
    x   = tf.repeat(tf.range(DIM//2, -DIM//2,-1), DIM)
    y   = tf.tile(tf.range(-DIM//2, DIM//2), [DIM])
    z   = tf.ones([DIM*DIM], dtype='int32')
    idx = tf.stack( [x,y,z] )

    # ROTATE DESTINATION PIXELS ONTO ORIGIN PIXELS
    idx2 = K.dot(m, tf.cast(idx, dtype='float32'))
    idx2 = K.cast(idx2, dtype='int32')
    idx2 = K.clip(idx2, -DIM//2+XDIM+1, DIM//2)

    # FIND ORIGIN PIXEL VALUES
    idx3 = tf.stack([DIM//2-idx2[0,], DIM//2-1+idx2[1,]])
    d    = tf.gather_nd(image, tf.transpose(idx3))

    return tf.reshape(d,[DIM, DIM,3])
```

In [9]:

```
def read_labeled_tfrecord(example):
    tfrec_format = {
        'image'                        : tf.io.FixedLenFeature([], tf.string),
        'image_name'                   : tf.io.FixedLenFeature([], tf.string),
        'patient_id'                   : tf.io.FixedLenFeature([], tf.int64),
        'sex'                          : tf.io.FixedLenFeature([], tf.int64),
        'age_approx'                   : tf.io.FixedLenFeature([], tf.int64),
        'anatom_site_general_challenge': tf.io.FixedLenFeature([], tf.int64),
        'diagnosis'                    : tf.io.FixedLenFeature([], tf.int64),
        'target'                       : tf.io.FixedLenFeature([], tf.int64)
    }
    example = tf.io.parse_single_example(example, tfrec_format)
    return example['image'], example['target']


def read_unlabeled_tfrecord(example, return_image_name):
    tfrec_format = {
        'image'                        : tf.io.FixedLenFeature([], tf.string),
        'image_name'                   : tf.io.FixedLenFeature([], tf.string),
    }
    example = tf.io.parse_single_example(example, tfrec_format)
    return example['image'], example['image_name'] if return_image_name else 0


def prepare_image(img, augment=True, dim=256):
    img = tf.image.decode_jpeg(img, channels=3)
```

```
    img = tf.cast(img, tf.float32) / 255.0

    if augment:
        img = transform(img,DIM=dim)
        img = tf.image.random_flip_left_right(img)
        #img = tf.image.random_hue(img, 0.01)
        img = tf.image.random_saturation(img, 0.7, 1.3)
        img = tf.image.random_contrast(img, 0.8, 1.2)
        img = tf.image.random_brightness(img, 0.1)

    img = tf.reshape(img, [dim,dim, 3])

    return img

def count_data_items(filenames):
    n = [int(re.compile(r"-([0-9]*)\.").search(filename).group(1))
         for filename in filenames]
    return np.sum(n)
```

In [10]:

```
def get_dataset(files, augment = False, shuffle = False, repeat = False,
                labeled=True, return_image_names=True, batch_size=16, dim=256):

    ds = tf.data.TFRecordDataset(files, num_parallel_reads=AUTO)
    ds = ds.cache()

    if repeat:
        ds = ds.repeat()

    if shuffle:
        ds = ds.shuffle(1024*8)
        opt = tf.data.Options()
        opt.experimental_deterministic = False
        ds = ds.with_options(opt)

    if labeled:
        ds = ds.map(read_labeled_tfrecord, num_parallel_calls=AUTO)
    else:
        ds = ds.map(lambda example: read_unlabeled_tfrecord(example, return_image_names)
,
                    num_parallel_calls=AUTO)

    ds = ds.map(lambda img, imgname_or_label: (prepare_image(img, augment=augment, dim=d
im),
                                               imgname_or_label),
                num_parallel_calls=AUTO)

    ds = ds.batch(batch_size * REPLICAS)
    ds = ds.prefetch(AUTO)
    return ds
```

Крок 3: Створення моделі

Тут можна обрати архітектуру моделі для експеременті**в.**

In [ ]:

```
#EFNS = [efn.EfficientNetB0, efn.EfficientNetB1, efn.EfficientNetB2, efn.EfficientNetB3,
#        efn.EfficientNetB4, efn.EfficientNetB5, efn.EfficientNetB6]

# YG
# These models HAVE PASSED the initial test!
#from tensorflow.keras.applications import MobileNetV2
#model_name = 'MobileNetV2'
from tensorflow.keras.applications import DenseNet121
model_name = 'DenseNet121'

# These should be checked:
```

```
#from tensorflow.keras.applications import InceptionV3
#model_name = 'InceptionV3'
#from tensorflow.keras.applications import InceptionResNetV2
#model_name = 'InceptionResNetV2'
#from tensorflow.keras.applications import MobileNetV2
#model_name = 'MobileNetV2'
#from tensorflow.keras.applications import ResNet101
#model_name = 'ResNet101'
#from tensorflow.keras.applications import ResNet101V2
#model_name = 'ResNet101V2'
#from tensorflow.keras.applications import VGG16
#model_name = 'VGG16'
#from tensorflow.keras.applications import Xception
#model_name = 'Xception'

#def build_model(dim=128, ef=0):
def build_model(dim=128):
    inp = tf.keras.layers.Input(shape=(dim,dim,3))
# YG
#    base = EFNS[ef](input_shape=(dim,dim,3),weights='imagenet',include_top=False)
#    base = MobileNetV2(input_shape=(dim,dim,3),weights='imagenet',include_top=False)
    base = DenseNet121(input_shape=(dim,dim,3),weights='imagenet',include_top=False)
    x = base(inp)
    x = tf.keras.layers.GlobalAveragePooling2D()(x)
    x = tf.keras.layers.Dense(1,activation='sigmoid')(x)
    model = tf.keras.Model(inputs=inp,outputs=x)
    opt = tf.keras.optimizers.Adam(learning_rate=0.001)
    loss = tf.keras.losses.BinaryCrossentropy(label_smoothing=0.05)
    model.compile(optimizer=opt,loss=loss,metrics=['AUC'])
    return model
```

Крок 4: План тренування

Це загальний план тренування моделі для трансферного навчання. Швидкість навчання починається близько нуля, потім збільшується до максимуму, а потім з часом занепадає. Швидкість навчання **max** більша при більших розмірах **BATCH_SIZE**. Це хороша практика, якої слід дотримуватися.

In [32]:

```
def get_lr_callback(batch_size=8):
    lr_start   = 0.000005
    lr_max     = 0.00000125 * REPLICAS * batch_size
    lr_min     = 0.000001
    lr_ramp_ep = 5
    lr_sus_ep  = 0
    lr_decay   = 0.8

    def lrfn(epoch):
        if epoch < lr_ramp_ep:
            lr = (lr_max - lr_start) / lr_ramp_ep * epoch + lr_start

        elif epoch < lr_ramp_ep + lr_sus_ep:
            lr = lr_max

        else:
            lr = (lr_max - lr_min) * lr_decay**(epoch - lr_ramp_ep - lr_sus_ep) + lr_min

        return lr

    lr_callback = tf.keras.callbacks.LearningRateScheduler(lrfn, verbose=False)
    return lr_callback
```

Тренування моделі

Наша модель буде навчена за кількістю **FOLDS** та **EPOCHS**, яку була обрана у конфігурації вище. Кожен згин моделі з найменшими втратами при валідації буде збережений і використаний для прогнозування **OOF** та тестування. Відрегулюйте змінні `VERBOSE` і `DISPLOY_PLOT` нижче, щоб визначити, який результат ви хочете

відображати. Змінна `VERBOSE = 1 або 2` відображатиме втрати навчання та перевірки та **auc** для кожної епохи у вигляді тексту. Змінна `DISPLAY_PLOT` відображає цю інформацію як графік.

In [33]:

```
from tqdm import tqdm
```

In [34]:

```
! nvidia-smi
```

```
/bin/sh: 1: nvidia-smi: not found
```

In [ ]:

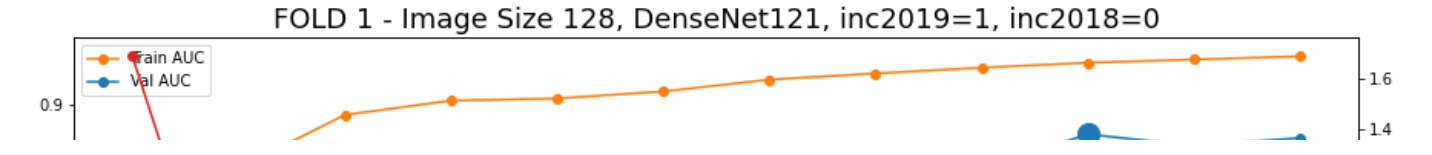
```
%%time
# USE VERBOSE=0 for silent, VERBOSE=1 for interactive, VERBOSE=2 for commit
VERBOSE = 1
DISPLAY_PLOT = True

skf = KFold(n_splits=FOLDS,shuffle=True,random_state=SEED)
oof_pred = []; oof_tar = []; oof_val = []; oof_names = []; oof_folds = []
preds = np.zeros((count_data_items(files_test),1))

for fold,(idxT,idxV) in tqdm(enumerate(skf.split(np.arange(15)))):

    # DISPLAY FOLD INFO
    if DEVICE=='TPU':
        if tpu: tf.tpu.experimental.initialize_tpu_system(tpu)
    print('#'*25); print('#### FOLD',fold+1)
# YG
#    print('#### Image Size %i with EfficientNet B%i and batch_size %i'%
#          (IMG_SIZES[fold],EFF_NETS[fold],BATCH_SIZES[fold]*REPLICAS))
    print('#### Image Size %i with %s and batch_size %i'%
          (IMG_SIZES[fold],model_name,BATCH_SIZES[fold]*REPLICAS))

    # CREATE TRAIN AND VALIDATION SUBSETS
    files_train = tf.io.gfile.glob([GCS_PATH[fold] + '/train%.2i*.tfrec'%x for x in idxT
])
    if INC2019[fold]:
        files_train += tf.io.gfile.glob([GCS_PATH2[fold] + '/train%.2i*.tfrec'%x for x i
n idxT*2+1])
        print('#### Using 2019 external data')
    if INC2018[fold]:
        files_train += tf.io.gfile.glob([GCS_PATH2[fold] + '/train%.2i*.tfrec'%x for x i
n idxT*2])
        print('#### Using 2018+2017 external data')
    np.random.shuffle(files_train); print('#'*25)
    files_valid = tf.io.gfile.glob([GCS_PATH[fold] + '/train%.2i*.tfrec'%x for x in idxV
])
    files_test = np.sort(np.array(tf.io.gfile.glob(GCS_PATH[fold] + '/test*.tfrec')))

    # BUILD MODEL
    K.clear_session()
    with strategy.scope():
# YG
#        model = build_model(dim=IMG_SIZES[fold],ef=EFF_NETS[fold])
        model = build_model(dim=IMG_SIZES[fold])

    # SAVE BEST MODEL EACH FOLD
    sv = tf.keras.callbacks.ModelCheckpoint(
        DEVICE + '_' + str(MODEL) + '_' + str(SIZE) + '_fold-%i.h5'%fold, monitor='val_l
oss', verbose=0, save_best_only=True,
        save_weights_only=True, mode='min', save_freq='epoch')

    # TRAIN
    print('Training...')
    history = model.fit(
        get_dataset(files_train, augment=True, shuffle=True, repeat=True,
                dim=IMG_SIZES[fold], batch_size = BATCH_SIZES[fold]),
        epochs=EPOCHS[fold], callbacks = [sv,get_lr_callback(BATCH_SIZES[fold])],
```

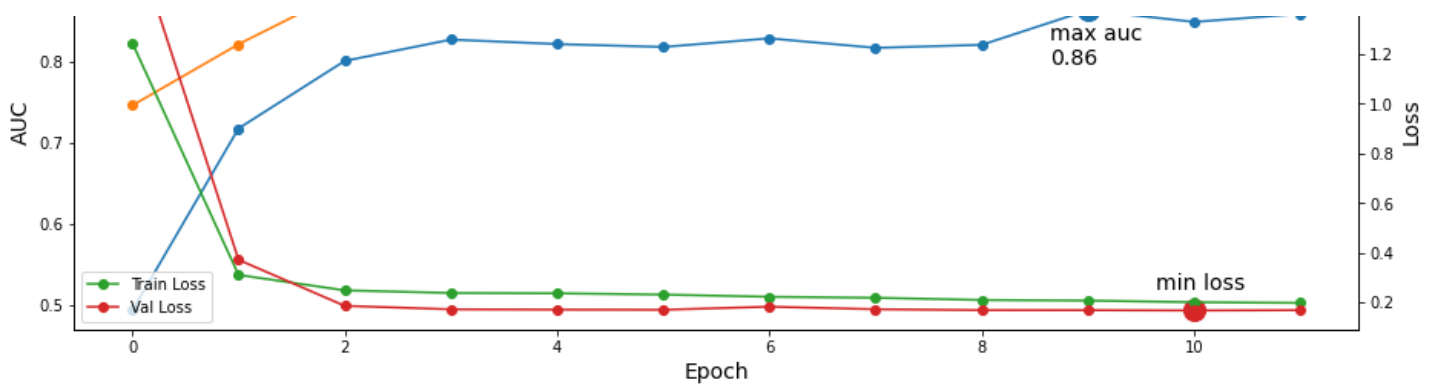
```
        steps_per_epoch=count_data_items(files_train)/BATCH_SIZES[fold]//REPLICAS,
        validation_data=get_dataset(files_valid,augment=False,shuffle=False,
                repeat=False,dim=IMG_SIZES[fold]), #class_weight = {0:1,1:2},
        verbose=VERBOSE
    )

    print('Loading best model...')
    model.load_weights(DEVICE + '_' + str(MODEL) + '_' + str(SIZE) + '_fold-%i.h5'%fold)

    # PREDICT OOF USING TTA
    print('Predicting OOF with TTA...')
    ds_valid = get_dataset(files_valid,labeled=False,return_image_names=False,augment=Tr
ue,
            repeat=True,shuffle=False,dim=IMG_SIZES[fold],batch_size=BATCH_SIZES[fold]*4
)
    ct_valid = count_data_items(files_valid); STEPS = TTA * ct_valid/BATCH_SIZES[fold]/4
/REPLICAS
    pred = model.predict(ds_valid,steps=STEPS,verbose=VERBOSE)[:TTA*ct_valid,]
    oof_pred.append( np.mean(pred.reshape((ct_valid,TTA),order='F'),axis=1) )

    #oof_pred.append(model.predict(get_dataset(files_valid,dim=IMG_SIZES[fold]),verbose=1
))

    # GET OOF TARGETS AND NAMES
    ds_valid = get_dataset(files_valid, augment=False, repeat=False, dim=IMG_SIZES[fold]
,
            labeled=True, return_image_names=True)
    oof_tar.append( np.array([target.numpy() for img, target in iter(ds_valid.unbatch())
]) )
    oof_folds.append( np.ones_like(oof_tar[-1],dtype='int8')*fold )
    ds = get_dataset(files_valid, augment=False, repeat=False, dim=IMG_SIZES[fold],
                labeled=False, return_image_names=True)
    oof_names.append( np.array([img_name.numpy().decode("utf-8") for img, img_name in it
er(ds.unbatch())]))

    # PREDICT TEST USING TTA
    print('Predicting Test with TTA...')
    ds_test = get_dataset(files_test,labeled=False,return_image_names=False,augment=True
,
            repeat=True,shuffle=False,dim=IMG_SIZES[fold],batch_size=BATCH_SIZES[fold]*4
)
    ct_test = count_data_items(files_test); STEPS = TTA * ct_test/BATCH_SIZES[fold]/4/RE
PLICAS
    pred = model.predict(ds_test,steps=STEPS,verbose=VERBOSE)[:TTA*ct_test,]
    preds[:,0] += np.mean(pred.reshape((ct_test,TTA),order='F'),axis=1) * WGTS[fold]

    # REPORT RESULTS
    auc = roc_auc_score(oof_tar[-1],oof_pred[-1])
    oof_val.append(np.max( history.history['val_auc'] ))
    print('#### FOLD %i OOF AUC without TTA = %.3f, with TTA = %.3f'%(fold+1,oof_val[-1]
,auc))

    # PLOT TRAINING
    if DISPLAY_PLOT:
        plt.figure(figsize=(15,5))
        plt.plot(np.arange(EPOCHS[fold]),history.history['auc'],'-o',label='Train AUC',c
olor='#ff7f0e')
        plt.plot(np.arange(EPOCHS[fold]),history.history['val_auc'],'-o',label='Val AUC'
,color='#1f77b4')
        x = np.argmax( history.history['val_auc'] ); y = np.max( history.history['val_au
c'] )
        xdist = plt.xlim()[1] - plt.xlim()[0]; ydist = plt.ylim()[1] - plt.ylim()[0]
        plt.scatter(x,y,s=200,color='#1f77b4'); plt.text(x-0.03*xdist,y-0.13*ydist,'max
auc\n%.2f'%y,size=14)
        plt.ylabel('AUC',size=14); plt.xlabel('Epoch',size=14)
        plt.legend(loc=2)
        plt2 = plt.gca().twinx()
        plt2.plot(np.arange(EPOCHS[fold]),history.history['loss'],'-o',label='Train Loss
',color='#2ca02c')
        plt2.plot(np.arange(EPOCHS[fold]),history.history['val_loss'],'-o',label='Val Lo
ss',color='#d62728')
        x = np.argmin( history.history['val_loss'] ); y = np.min( history.history['val_l
```

```
oss'] )
        ydist = plt.ylim()[1] - plt.ylim()[0]
        plt.scatter(x,y,s=200,color='#d62728'); plt.text(x-0.03*xdist,y+0.05*ydist,'min
loss',size=14)
        plt.ylabel('Loss',size=14)
# YG
#         plt.title('FOLD %i - Image Size %i, EfficientNet B%i, inc2019=%i, inc2018=%i'%
#                 (fold+1,IMG_SIZES[fold],EFF_NETS[fold],INC2019[fold],INC2018[fold]),siz
e=18)
        plt.title('FOLD %i - Image Size %i, %s, inc2019=%i, inc2018=%i'%
                (fold+1,IMG_SIZES[fold],model_name,INC2019[fold],INC2018[fold]),size=18)

        plt.legend(loc=3)
        plt.savefig('AUC_' + DEVICE + '_model' + str(MODEL) + '_' + str(SIZE) + '_fold'
+ str(fold) + '.png' ,bbox_inches='tight', dpi=300)
        plt.show()
```

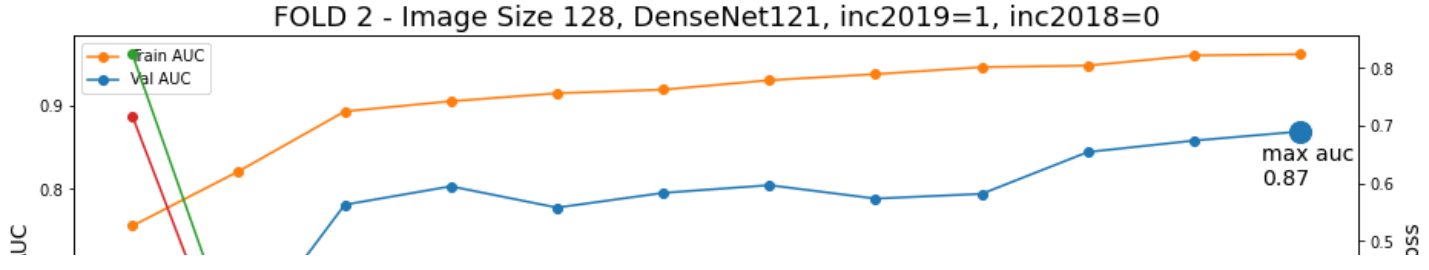
```
0it [00:00, ?it/s]
```

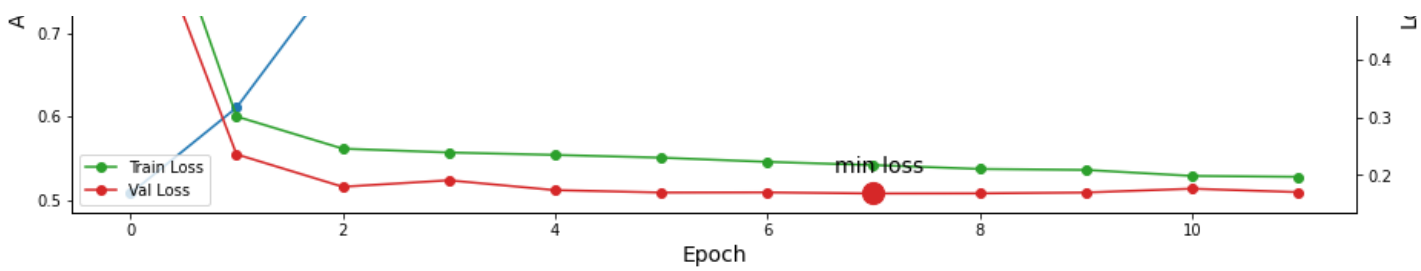
```
#########################
#### FOLD 1
#### Image Size 128 with DenseNet121 and batch_size 256
#### Using 2019 external data
#########################
Training...
Epoch 1/12
140/140 [==============================] - 45s 321ms/step - auc: 0.7466 - loss: 1.2448 -
val_auc: 0.4940 - val_loss: 1.6906 - lr: 5.0000e-06
Epoch 2/12
140/140 [==============================] - 20s 145ms/step - auc: 0.8219 - loss: 0.3109 -
val_auc: 0.7182 - val_loss: 0.3718 - lr: 6.8000e-05
Epoch 3/12
140/140 [==============================] - 20s 143ms/step - auc: 0.8876 - loss: 0.2488 -
val_auc: 0.8014 - val_loss: 0.1865 - lr: 1.3100e-04
Epoch 4/12
140/140 [==============================] - 20s 142ms/step - auc: 0.9052 - loss: 0.2383 -
val_auc: 0.8277 - val_loss: 0.1725 - lr: 1.9400e-04
Epoch 5/12
140/140 [==============================] - 20s 142ms/step - auc: 0.9079 - loss: 0.2369 -
val_auc: 0.8221 - val_loss: 0.1713 - lr: 2.5700e-04
Epoch 6/12
140/140 [==============================] - 20s 143ms/step - auc: 0.9165 - loss: 0.2317 -
val_auc: 0.8186 - val_loss: 0.1706 - lr: 3.2000e-04
Epoch 7/12
140/140 [==============================] - 17s 123ms/step - auc: 0.9311 - loss: 0.2229 -
val_auc: 0.8292 - val_loss: 0.1830 - lr: 2.5620e-04
Epoch 8/12
140/140 [==============================] - 18s 125ms/step - auc: 0.9387 - loss: 0.2189 -
val_auc: 0.8175 - val_loss: 0.1728 - lr: 2.0516e-04
Epoch 9/12
140/140 [==============================] - 20s 142ms/step - auc: 0.9458 - loss: 0.2100 -
val_auc: 0.8212 - val_loss: 0.1692 - lr: 1.6433e-04
Epoch 10/12
140/140 [==============================] - 20s 143ms/step - auc: 0.9519 - loss: 0.2077 -
val_auc: 0.8639 - val_loss: 0.1691 - lr: 1.3166e-04
Epoch 11/12
140/140 [==============================] - 20s 143ms/step - auc: 0.9559 - loss: 0.2010 -
val_auc: 0.8494 - val_loss: 0.1678 - lr: 1.0553e-04
Epoch 12/12
140/140 [==============================] - 18s 125ms/step - auc: 0.9599 - loss: 0.1983 -
val_auc: 0.8593 - val_loss: 0.1691 - lr: 8.4624e-05
Loading best model...
Predicting OOF with TTA...
71/70 [==============================] - 6s 87ms/step
Predicting Test with TTA...
118/117 [==============================] - 10s 83ms/step
#### FOLD 1 OOF AUC without TTA = 0.864, with TTA = 0.871
```






```
1it [08:12, 492.36s/it]

##########################
#### FOLD 2
#### Image Size 128 with DenseNet121 and batch_size 256
#### Using 2019 external data
##########################
Training...
Epoch 1/12
140/140 [==============================] - 41s 296ms/step - auc: 0.7558 - loss: 0.8240 -
val_auc: 0.5075 - val_loss: 0.7168 - lr: 5.0000e-06
Epoch 2/12
140/140 [==============================] - 21s 147ms/step - auc: 0.8214 - loss: 0.3017 -
val_auc: 0.6108 - val_loss: 0.2361 - lr: 6.8000e-05
Epoch 3/12
140/140 [==============================] - 20s 143ms/step - auc: 0.8934 - loss: 0.2458 -
val_auc: 0.7813 - val_loss: 0.1797 - lr: 1.3100e-04
Epoch 4/12
140/140 [==============================] - 18s 125ms/step - auc: 0.9057 - loss: 0.2392 -
val_auc: 0.8034 - val_loss: 0.1909 - lr: 1.9400e-04
Epoch 5/12
140/140 [==============================] - 20s 144ms/step - auc: 0.9152 - loss: 0.2350 -
val_auc: 0.7778 - val_loss: 0.1739 - lr: 2.5700e-04
Epoch 6/12
140/140 [==============================] - 21s 149ms/step - auc: 0.9196 - loss: 0.2302 -
val_auc: 0.7955 - val_loss: 0.1698 - lr: 3.2000e-04
Epoch 7/12
140/140 [==============================] - 18s 126ms/step - auc: 0.9309 - loss: 0.2230 -
val_auc: 0.8047 - val_loss: 0.1700 - lr: 2.5620e-04
Epoch 8/12
140/140 [==============================] - 21s 148ms/step - auc: 0.9382 - loss: 0.2173 -
val_auc: 0.7886 - val_loss: 0.1681 - lr: 2.0516e-04
Epoch 9/12
140/140 [==============================] - 18s 127ms/step - auc: 0.9466 - loss: 0.2106 -
val_auc: 0.7943 - val_loss: 0.1683 - lr: 1.6433e-04
Epoch 10/12
140/140 [==============================] - 18s 125ms/step - auc: 0.9485 - loss: 0.2090 -
val_auc: 0.8446 - val_loss: 0.1698 - lr: 1.3166e-04
Epoch 11/12
140/140 [==============================] - 18s 126ms/step - auc: 0.9607 - loss: 0.1986 -
val_auc: 0.8583 - val_loss: 0.1764 - lr: 1.0553e-04
Epoch 12/12
140/140 [==============================] - 18s 126ms/step - auc: 0.9621 - loss: 0.1970 -
val_auc: 0.8693 - val_loss: 0.1704 - lr: 8.4624e-05
Loading best model...
Predicting OOF with TTA...
71/70 [==============================] - 6s 83ms/step
Predicting Test with TTA...
118/117 [==============================] - 10s 89ms/step
#### FOLD 2 OOF AUC without TTA = 0.869, with TTA = 0.793
```






```
2it [16:05, 486.67s/it]
```

```
#########################
#### FOLD 3
#### Image Size 128 with DenseNet121 and batch_size 256
#### Using 2019 external data
#########################
Training...
Epoch 1/12
140/140 [==============================] - ETA: 0s - auc: 0.7526 - loss: 1.3085
```

Обчислення **AUC OOF**

Прогнози **OOF** зберігаються на диску. Якщо ви хочете зібрати декілька моделей, використовуйте **OOF**, щоб визначити, які найкращі ваги поєднувати з моделями. Вибирайте ваги, які максимізують коефіцієнт перехресної валідації **OOF**, коли використовуються для змішування **OOF**. Потім використовуйте ті самі ваги, щоб поєднати ваші прогнозні тести.

In [ ]:

```
# COMPUTE OVERALL OOF AUC
oof = np.concatenate(oof_pred); true = np.concatenate(oof_tar);
names = np.concatenate(oof_names); folds = np.concatenate(oof_folds)
auc = roc_auc_score(true,oof)
print('Overall OOF AUC with TTA = %.3f'%auc)

# SAVE OOF TO DISK
df_oof = pd.DataFrame(dict(
    image_name = names, target=true, pred = oof, fold=folds))
df_oof.to_csv(DEVICE + '_' + str(MODEL) + '_oof.csv',index=False)
df_oof.head()
```

Крок **5:** Пост процес

Вимірювання часу прогнозування, **AUC,** розміу файлу моделі

З урахуванням збільшення даних під час тестування

In [ ]:

```
import time
import statistics

VERBOSE = 1
```

In [ ]:

```
predict_withTTA_time_list = []
AUC_withTTA_list = []

oof_pred = []; oof_tar = []

for fold,(idxT,idxV) in enumerate(skf.split(np.arange(15))):
    print('Fold %i. Loading the current fold model...'%fold)
    model.load_weights(DEVICE + '_' + str(MODEL) + '_' + str(SIZE) + '_fold-%i.h5'%fold)
    #model.load_weights(DEVICE + '_fold-%i.h5'%fold)
```

```
    # PREDICT with TTA # augment=True
    ds_valid = get_dataset(files_valid,labeled=False,return_image_names=False,augment=Tr
ue,
            repeat=True,shuffle=False,dim=IMG_SIZES[fold],batch_size=BATCH_SIZES[fold]*4
)
    ct_valid = count_data_items(files_valid); STEPS = TTA * ct_valid/BATCH_SIZES[fold]/4
/REPLICAS
    print('Predicting Test with TTA for %i files...'%(ct_valid))

    # Start timer
    ts_eval = time.time()
    pred = model.predict(ds_valid,steps=STEPS,verbose=VERBOSE)[:TTA*ct_valid,]
    # End timer
    te_eval = time.time()

    test_time = (te_eval-ts_eval)/ct_valid
    predict_withTTA_time_list.append(test_time)
    print('Fold %i, test_time=%.6f seconds.'%(fold,test_time))

    # Add predictions to list
    oof_pred.append( np.mean(pred.reshape((ct_valid,TTA),order='F'),axis=1) )
    # GET OOF TARGETS AND NAMES
    ds_valid = get_dataset(files_valid, augment=False, repeat=False, dim=IMG_SIZES[fold]
,
            labeled=True, return_image_names=True)
    # Add targets to list
    oof_tar.append( np.array([target.numpy() for img, target in iter(ds_valid.unbatch())
]) )
    # Calculate AUC
    auc = roc_auc_score(oof_tar[-1],oof_pred[-1])
    # Add AUC to list
    AUC_withTTA_list.append(auc)
    print('Fold %i, AUC=%.6f'%(fold,auc))

# Calculate AUC mean value to list
AUC_withTTA_mean = statistics.mean(AUC_withTTA_list) # mean
# Calculate AUC standard deviation value to list
AUC_withTTA_std = statistics.stdev(AUC_withTTA_list) # standard devition
print('#### OOF AUC with TTA: mean = %.6f, stdev = %.6f.'%(AUC_withTTA_mean,AUC_withTTA_s
td))

# Add AUC mean value to list
predict_withTTA_time_mean = statistics.mean(predict_withTTA_time_list) # mean
# Add AUC standard deviation value to list
predict_withTTA_time_std = statistics.stdev(predict_withTTA_time_list) # standard devitio
n
print('#### Time without TTA: mean = %.6f, stdev = %.6f seconds.'%(predict_withTTA_time_m
ean,predict_withTTA_time_std))
```

Без урахування збільшення даних під час тестування

In [ ]:

```
predict_woTTA_time_list = []
AUC_woTTA_list = []

oof_pred = []; oof_tar = []

for fold,(idxT,idxV) in enumerate(skf.split(np.arange(15))):
    print('Fold %i. Loading the current fold model...'%fold)
    model.load_weights(DEVICE + '_' + str(MODEL) + '_' + str(SIZE) + '_fold-%i.h5'%fold)
    #model.load_weights(DEVICE + '_fold-%i.h5'%fold)

    # PREDICT without TTA # augment=False
    ds_valid = get_dataset(files_valid,labeled=False,return_image_names=False,augment=Fa
lse,
            repeat=True,shuffle=False,dim=IMG_SIZES[fold],batch_size=BATCH_SIZES[fold]*4
)
    ct_valid = count_data_items(files_valid); STEPS = TTA * ct_valid/BATCH_SIZES[fold]/4
```

```
/REPLICAS
    print('Predicting Test without TTA for %i files...'%(ct_valid))

    # Start timer
    ts_eval = time.time()
    pred = model.predict(ds_valid,steps=STEPS,verbose=VERBOSE)[:TTA*ct_valid,]
    # End timer
    te_eval = time.time()

    test_time = (te_eval-ts_eval)/ct_valid
    predict_woTTA_time_list.append(test_time)
    print('Fold %i, test_time=%.6f seconds.'%(fold,test_time))

    # Add predictions to list
    oof_pred.append( np.mean(pred.reshape((ct_valid,TTA),order='F'),axis=1) )
    # GET OOF TARGETS AND NAMES
    ds_valid = get_dataset(files_valid, augment=False, repeat=False, dim=IMG_SIZES[fold]
,
            labeled=True, return_image_names=True)
    # Add targets to list
    oof_tar.append( np.array([target.numpy() for img, target in iter(ds_valid.unbatch())
]) )
    # Calculate AUC
    auc = roc_auc_score(oof_tar[-1],oof_pred[-1])
    # Add AUC to list
    AUC_woTTA_list.append(auc)
    print('Fold %i, AUC=%.6f'%(fold,auc))

# Calculate AUC mean value to list
AUC_woTTA_mean = statistics.mean(AUC_woTTA_list) # mean
# Calculate AUC standard deviation value to list
AUC_woTTA_std = statistics.stdev(AUC_woTTA_list) # standard devition
print('#### OOF AUC without TTA: mean = %.6f, stdev = %.6f.'%(AUC_woTTA_mean,AUC_woTTA_st
d))

# Add AUC mean value to list
predict_woTTA_time_mean = statistics.mean(predict_woTTA_time_list) # mean
# Add AUC standard deviation value to list
predict_woTTA_time_std = statistics.stdev(predict_woTTA_time_list) # standard devition
print('#### Time without TTA: mean = %.6f, stdev = %.6f seconds.'%(predict_woTTA_time_mea
n,predict_woTTA_time_std))
```

Розмір файлу моделі

In [ ]:

```
import os
model_size = os.path.getsize(DEVICE + '_' + str(MODEL) + '_' + str(SIZE) + '_fold-0.h5')
# >> 20
print(str(model_size) + ' Bytes')
```

Збереження моделі у файл

In [ ]:

```
results = pd.DataFrame(data=[[MODEL, model_size,
                              predict_woTTA_time_mean, predict_woTTA_time_std, AUC_woTTA
_mean, AUC_woTTA_std,
                              predict_withTTA_time_mean, predict_withTTA_time_std, AUC_w
ithTTA_mean, AUC_withTTA_std]],
                       columns=['model', 'model_size',
                                'time_mean','time_std', 'AUC_mean', 'AUC_std',
                                'TTA_time_mean','TTA_time_std','TTA_AUC_mean','TTA_AUC_s
td']])
results.to_csv(DEVICE + '_' + str(MODEL) + '_' + str(SIZE) + '_time_AUC.csv', index=Fals
e)
results.head()
```

In [ ]:

In [ ]:

```
! nvidia-smi
```

In [ ]:

```
!ls -all
```

In [ ]:

```

```