НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

На правах рукопису
УДК 004.852

До захисту допущено
В. о. зав. кафедри
_____О.І. Чумаченко
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра
зі спеціальності 122 «Комп'ютерні науки»
на тему: «Зоровий трансформер для задачі класифікації раку шкіри»

Виконав:
студент ІІ курсу, групи КІ-11мп
Нікітін Владислав Олегович _____

Керівник:
к.т.н., ст. викладач Шаповал Н. В. _____

Рецензент:
доцент кафедри ІКТС ІТС НТУУ "КПІ"
к.т.н. Астраханцев А. А. _____

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

Студент _____

Київ
2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Рівень вищої освіти – другий (магістерський)

Спеціальність – 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

В. о. зав. кафедри

_____ О.І. Чумаченко

«___» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Нікітіну Владиславу Олеговичу

1. Тема дисертації: «Зоровий трансформер для задачі класифікації раку шкіри», науковий керівник роботи Шаповал Наталія Віталіївна, ст. викладач.
2. Термін подання студентом дисертації 15.12.2022
3. Об'єкт дослідження: Зоровий трансформер в задачах класифікації.
4. Предмет дослідження: Моделі та архітектура зорового трансформеру.
5. Перелік завдань, які потрібно зробити:
    1) здійснити огляд технічної літератури за темою роботи;
    2) дослідити актуальність обраної теми;
    3) ознайомитись із існуючими підходами до класифікації раку шкіри та застосування зорових трансформерів;

4) провести навчання зорового трансформеру на наборі даних, що репрезентує ракові/неракові утворення на шкірі;

5) створити веб додаток на основі моделі, що міг би класифікувати запропоноване користувачем зображення;

6) провести аналіз ринкових можливостей запуску стартап проекту;

7) розробити концептуальні висновки;

8) підготувати ілюстративний матеріал;

9) оформити пояснювальну записку.

6. Перелік ілюстративного матеріалу.

7. Дата видачі завдання: 1 вересня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	01.09.2022 30.09.2022	Виконано
2.	Підготовка першого розділу.	01.10.2022 16.10.2022	Виконано
3.	Підготовка другого розділу.	16.10.2022 05.11.2022	Виконано
4.	Розробка програмного продукту.	05.11.2022 10.11.2022	Виконано
5.	Підготовка третього розділу	10.11.2022 20.11.2022	Виконано
6.	Підготовка частини стартап-проєкту	20.11.2022 01.12.2022	Виконано
9.	Концептуальні висновки. Перспективи розвитку отриманих рішень	01.12.2022 03.12.2022	Виконано
10.	Оформлення пояснювальної записки	03.12.2022 05.12.2022	Виконано

Студент Владислав НІКІТІН

Керівник Наталія ШАПОВАЛ

РЕФЕРАТ

Магістерська дисертація: 122 с., 24 табл., 20 рис., 64 джерела, 1 додаток.

КОМП’ЮТЕРНИЙ ЗІР, КЛАСИФІКАТОР НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ, ТРАНСФОРМЕРИ, ЗОРОВІ ТРАНСФОРМЕРИ, ФУНКЦІЯ УВАГИ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ

Об’єктом дослідження є набір даних із зображеннями раку шкіри та здорових шкірних утворень.

Предмет дослідження – архітектури моделі зорового трансформера для класифікації раку шкіри.

Мета дослідження полягає у аналізі архітектури зорового трансформера для задачі класифікації раку шкіри, порівняння отриманої моделі зорового трансформера з аналогічними моделями, що мають згорткову архітектуру, розробка користувацького додатку, що міг би класифікувати запропоноване користувачем зображення утворення на шкірі як ракове чи доброякісне.

Розроблена модель підтверджує ефективність зорового трансформера в задачі класифікації раку шкіри. Після порівняння моделі зі згортковою аналогічною моделлю потенціал зорового трансформера, що може розкритися за умови наявності більших обчислювальних потужностей та більшого датасету є очевидним.

Запропонована архітектура зорового трансформера здатна ефективно класифікувати зображення та додаток, створений на її основі може бути дієвим інструменти ідентифікації того чи потрібно звернутись до лікаря із шкірним утворенням.

ABSTRACT

The theme: ‘Vision transformer for skin cancer classification’

Diploma work: 122 p., 20 fig., 24 tabl., 1 appendix, 64 references.

COMPUTER VISION, NEURAL NETWORK CLASSIFIER, TRANSFORMER, VISION TRANSFORMER, ATTENTION, CONVOLUTIONAL NEURAL NETWORKS, MACHINE LEARNING

The object of the research – dataset with skin cancer normal skin lesions.

The subject of the research – architecture of vision transformers for skin cancer classification.

The purpose of the work is analysis of vision transformers for skin cancer classification, comparison ViT models with same purpose models that have convolutional architecture, development of a user app that can classify skin lesions proposed by the user as cancerous or a healthy one.

Developed model proves effectiveness of vision transformer in skin cancer classification task. After comparison with the convolutional model the potential of a ViT based model that can be unleashed with greater computational powers and larger dataset is obvious.

Proposed architecture of the vision transformer can effectively classify skin cancer images and an app developed on its base can be a resultative instrument of identifying if the doctor appointment should be scheduled for more detailed diagnostics.

.

ЗМІСТ

ВСТУП

Рак шкіри – це неконтрольований ріст патологічних клітин в епідермісі викликаний виправленими пошкодженнями ДНК, що запускає мутації. Мутації змушують клітини шкіри швидко розмножуватись та створювати шкідливі пухлини. Головні типи раку шкіри це базаліома, плоскоклітинний рак та меланома. Всі типи раку шкіри можуть бути викликані двома головними причинами. По-перше, це шкідливі сонячні ультрафіолетові промені та ультрафіолетові солярії [1]. А по-друге, це спадкова генетична схильність до цієї хвороби.

Тим не менш, візит до лікаря з проблемою, що наразі не завдає значних незручностей не завжди являється пріоритетом людини, через що велика кількість випадків раку, та раку шкірі в тому числі, виявляються не на перших етапах через що хвороба має значний відсоток смертності.

Наприклад, для меланоми смертність за 5 років смертність може сягати від 30 до 68% залежно від місця виявлення пухлини [8]. В Україні ми маємо дещо іншу, але тим не менш сумну статистику. Станом на 2018 рік було виявлено 2835 випадків меланоми. 10.9 відсотків цих пацієнтів не прожили одного року з моменту виявлення захворювання [2].

В той же час при виявленні меланоми на ранніх стадіях з високою ймовірністю можливе повне видалення пухлини з мінімальними рубцями або безслідно, не згадуючи вже про мінімальний ризик для здоров’я. Особливо це стосується випадків, доктор виявляє пухлину ще до її перетворення на злоякісну та проникнення під шар шкіри [1].

Автоматизація процесу виявлення раку шкіри могла б значно збільшити відсоток хворих, що були виявлені на ранніх стадіях і відповідно збільшити шанси успішного лікування хвороби. Отже, ця робота має за мету дослідження

можливості раннього виявлення раку шкіри (зокрема меланоми) за допомогою зорових трансформерів.

Зоровий трансформер це модель машинного навчання похідна від моделі трансформера, що застосовується в обробці природної мови. Ця нова модель в багатьох задачі переважає стандартні підходи, як наприклад, різні види згорткових нейронних мереж.

Задачі цієї роботи полягають у автоматизації задачі класифікації ракових пухлин за допомогою зображень за допомогою зорового трансформера та аналіз можливості використання моделі на основі зорового трансформера як ефективного класифікатора патологічних утворень на шкірі.

РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ, ПРЕДМЕТНА ОБЛАСТЬ

1.1 Актуальність роботи.

Найбільш поверхнево рак шкіри поділяють на 2 типи — меланома та не-меланома. Найчастішими видами раку шкіри що не є меланомою є базаліома так плоскоклітинний рак.

Рак шкіри представляє особливу проблему для оцінки захворюваності з кількох причин. Існує кілька підтипів раку шкіри, які можуть створювати проблеми при зіставленні даних. Наприклад, немеланомний рак шкіри часто не відстежується в реєстрах раку, або реєстрація цього раку часто неповна, оскільки більшість випадків успішно лікуються хірургічним шляхом або абляцією. У зв’язку з цими факторами ймовірно, що зареєстрована глобальна захворюваність на рак шкіри є заниженою.

Меланома шкіри є 17-м за поширеністю раком у світі. Це 13-те за поширеністю рак у чоловіків і 15-те за поширеністю у жінок. У 2020 році було зареєстровано понад 150 000 нових випадків меланоми шкіри.

Немеланомний рак шкіри часто виключається зі звітності про рак. У глобальних загальних випадках раку не повідомляється. Це пояснюється тим, що воно дуже поширене, та зазвичай лікується в рамках первинної медичної допомоги, а отже, ймовірно, про нього буде недостатньо повідомлено в даних національного ракового реєстру.

10 країн з найвищим рівнем обох типів раку шкіри та найбільшою кількістю смертей від обох типів раку шкіри у 2020 році наведено в таблиці нижче.

СВП = стандартизовані за віком показники. Це сумарний показник рівня захворювань, який мало би населення, якби воно мало стандартну вікову структуру. Стандартизація необхідна при порівнянні популяцій, які

відрізняються за віком, оскільки вік має потужний вплив на ризик смерті від раку [10].

Таблиця 1.1 – СВП країн з найвищим рівнем раку шкіри [10].

Номер	Країна	Число	СВП/100,000
	Світ	324,635	3.4
1	Австралія	16,171	36.6
2	Нова Зеландія	2,801	31.6
3	Данія	2,886	29.7
4	Нідерланди	8,310	27.0
5	Норвегія	2,567	26.4
6	Швеція	4,266	23.3
7	Швейцарія	3,357	21.6
8	Німеччина	31,468	20.5
9	Словенія	735	19.7
10	Фінляндія	2,090	19.5

Тим не менш, на відміну від інших типів раку, цей має великі шанси бути діагностованим на ранніх стадіях у випадку, якщо хворий передчасно звернеться до спеціаліста. На жаль, пріоритет людини часто зміщений і вона не в змозі пройти огляд спеціаліста.

Автоматизація задачі класифікації пухлин на шкірі, могла б дати людині можливість провести первинний медичний огляд раніше, ніж вона потрапить до лікаря. І, у випадку отримання позитивного результату тесту на рак шкіри, мотивувати її звернутись до лікаря раніше. Відповідно, повна автоматизація цієї задачі могла б, в перспективі, значно збільшити показники шанси середнього хворого на своєчасне і успішне лікування.

В цій роботі пропонується розглянути модель зорового трансформера як класифікатор шкірних пухлин. Вибір цієї моделі зумовлений її новизною та високими показниками в деяких типах задач класифікації порівняно зі згортковими моделями, що давно стали стандартом у задачах обробки та аналізу зображень.

1.2 Рак шкіри.

Рак це захворювання при якому клітини тіла починають неконтрольовано рости і розповсюджуватись до інших частин тіла. Рак може початися фактично будь де в людському тілі, так як кожна людина складається з трильйонів клітин. При нормальному життєвому циклі, людська клітина росте і ділиться, щоб створити нові клітини коли тіло цього потребує (тобто, клітина отримує певні тригери для стимуляції бажаної поведінки). Коли клітини старіють, або травмуються, то вони помирають (що також зумовлено тригерами), та нові клітини займають їх місце [3].

Іноді це послідовний процес перестає працювати належним чином. Клітини починають рости та ділитись тоді, коли немає відповідних умов для цього. В деяких особливих випадках ці клітини здатні сформувати пухлини, — скупчення тканини з патологічних клітин. Ці пухлини можуть бути двох типів: злоякісними (раковими) чи доброякісними.

Ракові пухлини агресивно розповсюджуються в сусідні тканини, та навіть можуть переміщуватися на значні відстані в тілі, утворюючи нові пухлини, що називаються метастазами.

На відміну від злоякісних, доброякісні пухлини не розповсюджуються в сусідні тканини. При усуненні таких пухлин, вони зазвичай не утворюються знову в той час як ракові іноді мають тенденцію до рецидиву. Тим не менш, доброякісні пухлини можуть бути досить великими. Іноді, вони спричиняють серйозні симптоми, що можуть загрожувати життю.

Основні відмінності ракових клітин від звичайних:

- Ростуть без отримання відповідних сигналів. Ріст звичайних клітин може бути спровокований тільки визначеними тригерами.
- Ігнорують сигнали припинити процес ділення чи припинити життєдіяльність.

- Агресивно розповсюджуються в сусідні та інші тканини/органи тіла. Звичайні клітини припиняють ділення, коли стикаються з клітинами сусідніх тканин і не розповсюджуються в інші частини тіла.
- Не розпізнаються імунною системою. Імунна система зазвичай знаходить і знищує клітини, що функціонують відмінно від норми.
- Накопичують зміни в хромосомах, як дублікація чи видалення деяких частин хромосоми.
- Споживають інші види нутрієнтів, можуть добувати енергію іншими, більш ефективними шляхами і, відповідно, рости швидше.
- Зазвичай тіло знищує клітини з пошкодженим ДНК до того як вони стануть раковими. Але ця можливість імунної системи слабшає з віком людини. Це одна з причин, чому ризик раку вищий для старших людей.

Рак кожної людини має унікальну комбінацію генетичних змін. З ростом ракових пухлин, додаткові виникають додаткові зміни. Навіть в одній пухлині можуть знаходитись клітини з різними генетичними змінами [3].

Рак шкіри — патологічний ріст клітин шкіри. Найчастіше, утворюється на ділянках шкіри, відкритих до сонця. Але є випадки, коли ця типова форма раку також утворюється на ділянках, що на зазнають впливу сонячних променів .

Є 3 основні типи раку шкіри — це базаліома, плоскоклітинний рак та меланома. Основні способи запобігти розвитку пухлин — уникнення та обмеження ультрафіолетового випромінювання, та регулярна перевірка підозрілих утворень на шкірі. Раннє виявлення всіх цих типів раку значно збільшує шанси успішного лікування.

Рак шкіри розвивається на ділянках найбільш уражених сонячним опроміненням включаючи шкіру голови, обличчя, губи, вуха, шию, руки,

долоні, та ноги в жінок. Але він також іноді виникає і на місцях, що зазвичай не зазнають впливу сонячного світла як шкіра під нігтями чи інтимні зони.

Ймовірність виникнення раку шкіри не залежить від кольору чи тону шкіри, крім того, що в людей з темною шкірою меланома зазвичай виникає на місцях, що не зазнають прямого впливу сонячних променів [9].

Меланома може виникнути будь де на тілі: на звичайній шкірі чи на родимці, що стає раковою. Частіше всього вона розвивається на обличчі. Меланома, як і інші типу раку шкіри, може виникати на ділянках закритих від сонячних променів.

Ознаки меланоми:

- Велика коричнева пляма з темними вкрапленнями.
- Пляма, що змінює колір, розмір чи кровоточить.
- Невелике ураження з неправильною межею та частинами, які виглядають червоними, рожевими, білими, синіми або синьо-чорн ими.
- Болісне ураження, що свербить чи пече.
- Темні плями на долонях, підошвах, кінчиках пальців або ніг, або на слизових оболонках, що вистилають рот, ніс.
- Базаліома частіше всього виникає на ділянках, що часто опромінюються сонячними променями, таких як обличчя та руки людини.

Базаліома виглядає як:

- Перламутрова або воскова шишка.
- Плоске ураження тілесного або коричневого кольору, схоже на рубець.
- Кровоточива виразка, яка заживає та повертається.

Найчастіше плоскоклітинний рак виникає на ділянках тіла, які піддаються впливу сонця, наприклад на обличчі, вухах і руках. У людей із

темнішою шкірою частіше розвивається плоскоклітинний рак на ділянках, які рідко потрапляють на сонце.

Плоскоклітинний рак може виглядати як:

- Твердий червоний вузлик.
- Плоске ураження з лускатою, покритою кіркою поверхнею.

1.3 Автоматизація задачі класифікації раку шкіри.

В останні кілька років багато вчених працювали над розробкою комп'ютерних систем діагностики (КСД) для класифікації раку шкіри. До появи глибокого навчання системи в основному проектувалися за допомогою алгоритмів машинного навчання [23]. Однак через складність розробки функцій і обмеження функцій ручної роботи ці методи на основі машинного навчання можуть діагностувати лише підмножину шкірних захворювань. Алгоритми глибокого навчання, з іншого боку, можуть автоматично вивчати семантичні характеристики з великомасштабних наборів даних з більшою точністю та ефективністю. У результаті такі методи глибокого навчання, як згорткова нейронна мережа, використовувалися для вирішення переважної більшості проблем класифікації раку шкіри в останні роки із задовільними результатами.

Однак, коли ми глибше вивчаємо проблеми класифікації раку шкіри, виявляється, що вони не такі прості, як проблеми в немедичній сфері. По-перше, багато наборів даних зображень шкіри є незбалансованими через диспропорції між різними класами раку шкіри, що збільшує ризик помилкового діагнозу діагностичною системою. Крім того, оскільки правильна анотація потребує великої кількості експертних знань і займає багато часу та трудомісткості, багато наборів даних надають лише обмежену кількість зображень (наприклад, набір даних ISIC є найбільшим загальнодоступним

набором даних про захворювання шкіри на сьогоднішній день, який містить близько 13 000 зображень шкіри). Як наслідок, для розробки точнішої системи потрібно більше позначених даних. Крім того, коли кількість навчальних даних недостатня, продуктивність узагальнення моделі погіршується. Крім того, різні шуми, створювані різними пристроями або різними умовами зйомки, також вносять зміщення в модель, що призводить до зниження діагностики. Крім того, операційна ефективність і споживання ресурсів моделі також обмежують її клінічне застосування на різних медичних пристроях.

Незважаючи на ефективність дермаскопічної діагностики раку шкіри, для експертів дерматологів досить складно зробити точну класифікацію злоякісної меланоми чи доброякісної пухлини для великої кількості дермоскопічних зображень. Тому, часто постає необхідність побудови комп'ютерну систему діагностики (КСД) для класифікації шкірних пухлин. КСД зазвичай складається з 4 основних кроків: попередньої обробки зображення, сегментації, виділення ознак та класифікації. Кожен крок дуже сильно впливає на продуктивність класифікації всієї КСД. Тому ефективні алгоритми мають бути застосовані на кожному з цих крокі[12].

Багато досліджень було проведено для визначення унікальних методів машинного навчання для класифікації різних типів раку. Більшість з них використовували набір даних з ознак, що були вручну вилучені із зображень. Більшість з методів машинного навчання потребують високопотужних систем та багато часу для визначення точних діагнозів та їх ефективність залежить від ознак, що характеризують потенційну ділянку виявлення пухлини. Згорткові нейронні мережі є важливими для автоматичної діагностики багатьох типів раку. Глибоке навчання досягло досить вражаючих результатів в задачах класифікації зображень.

Є приклади [13], коли для класифікації уражень шкіри за допомогою дермоскопічних зображень було використано підхід глибокого навчання, заснований на ансамблях нейронних мереж. Система класифікації меланоми

була розроблена для виділення асиметрії, нерівності меж, варіації кольору, діаметра та текстури ураження шкіри та класифікації злоякіснох та доброякісних уражень шкіри за допомогою методу опорних векторів [14]. Існує приклад, де була розроблена модель згорткової нейронної мережі на основі вейвлетів Габора для діагностики уражень шкіри шляхом використання фільтрів Габора на зображеннях шкіри для отримання детальних характеристик та подальшого моделювання цих деталей за допомогою глибоких згорткових моделей. Результуюча точність склала точністю 83% [12].

Більшість існуючих систем діагностики уражень шкіри повідомляють обґрунтовані результати класифікації для розрізнення злоякісної меланоми від доброякісних уражень. Проте все ще дуже важко розробити ефективну автоматизовану структуру, яка може надати дуже точні результати діагностики, працюючи в режимі реального часу з низькими специфікаціями обладнання.

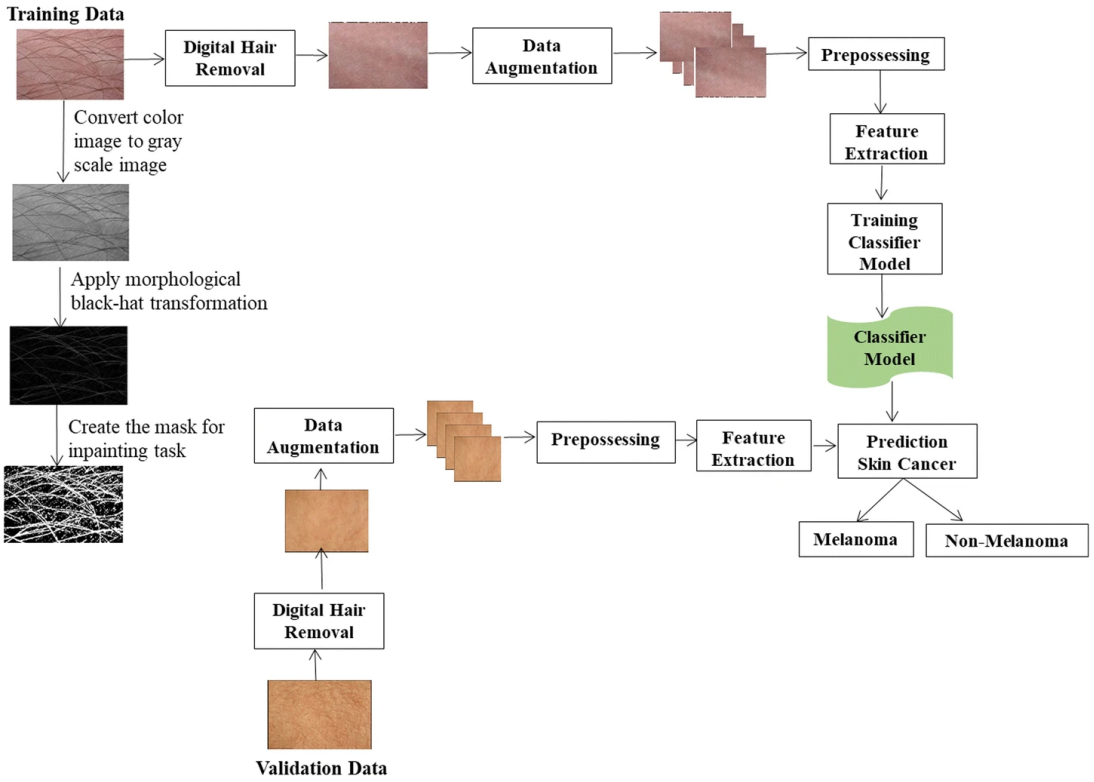

Рис. 1.1 — Діаграма системи автоматичної діагностики раку шкіри застосованої в роботі “Автоматизований підхід глибокого навчання для класифікації злоякісної меланоми та доброякісних уражень шкіри” [12].

В роботі [15] стверджується, що вихідні дані правильно налаштованої КСД разом з діагностикою за допомогою радіології збільшують точність прогнозу для пухлини. Процес створення моделі, що працює в КСД в цій роботі має п’ять ключових стадій: збір даних, попередня обробка, вилучення та вибір ознак та, власне, класифікація.

На етапі попередньої обробки даних були складнощі з усуненням непотрібних елементів із зображення до того як воно буде оброблене. В цій роботі для цього розглядалися наступні методи: CLAHE[16], Адаптивний дифузійний фільтр Unsharp Masking[17], Трансформація циліндра [18].

Одним з найважливіших та критичних етапів для класифікації і діагностики є правильне визначення границі ураження шкіри. В роботі [19] для цього використовувалися алгоритми k-середніх та модифікований алгоритм k-середніх.

За допомогою вилучення ознак за допомогою метода гістограм можна отримати тільки локальні ознаки, тож щоб отримати і просторові ознаки в роботі використовується вилучення ознак на основі матриці спільного входження. Матриця показує значення частоти рівня сірого в заданому просторі зі значенням рівня сірого області інтересу. Кількість рядків і стовпців у матриці показує рівень сірого або колір пікселя значення поверхні зображення [20].

Обчислення ознак текстур використовує матрицю спільного входження, щоб порахувати інтенсивність варіацій у пікселі інтересу. Їх обчислення, загалом, залежить від 2 параметрів таких як відносна відстань між парами пікселів та їх відносна орієнтація. Матриця складається з чотирнадцяти ознак. З них найбільш цінними є ентропія, контраст, енергія, зворотній момент різниці, кореляція, та інформаційні границі кореляції [15].

Етап вибору ознак потрібен так як не всі ознаки можуть бути корисними для класифікації. Вибір ознак проводиться з метою визначення найбільш

підходящого набору ознак для вирішення задачі. Для цього використовується алгоритм HHO [21].

Після цього групи зображень були класифіковані за вибраними ознаками. В якості моделі використовувалася гібридна згорткова мережа IHHO [15].

В роботі “Класифікація зображень меланоми, невуса та себорейного кератозу” [13] була поставлена задача класифікації зображень уражень шкіри на три класи — меланома, невус та себорейний кератоз через бінарні класифікатори: меланома проти не-меланоми та себорейний кератоз проти не себорейного кератозу. Учасники змагання під час якого була створена модель з цієї роботи оцінювались за середнім значення під ROC кривою. Було дозволено використовувати додаткові набори даних для тренування та бінарну ознаку статевої приналежності.

На рисунку 1.3.2. зображена запропонована в роботі загальна модель класифікації. Перш за все, яскравість та баланс кольорів вхідних зображень нормалізуються з використанням сталості кольору. Нормалізовані зображення є вхідними даними для класифікаторів “меланома проти не-меланоми” та “себорейний кероз проти не себорейного керозу”. Базова структура класифікатора представлена на рисунку 1.2.

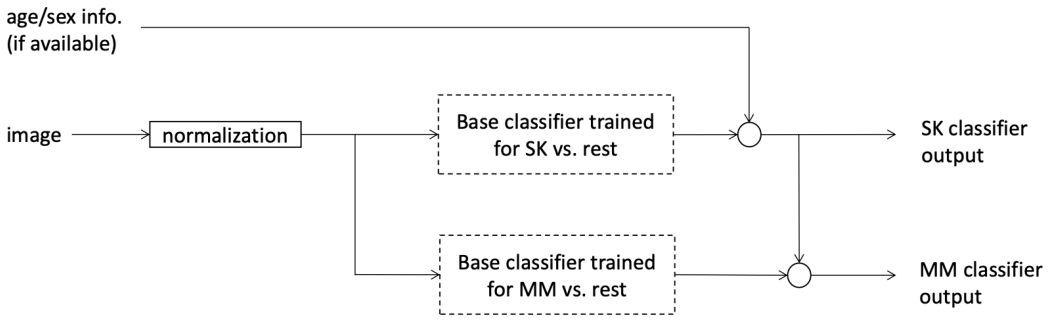

Рисунок 1.2 – Запропонована загальна модель класифікації в роботі “Класифікація зображень меланоми, невуса та себорейного кератозу” [13]

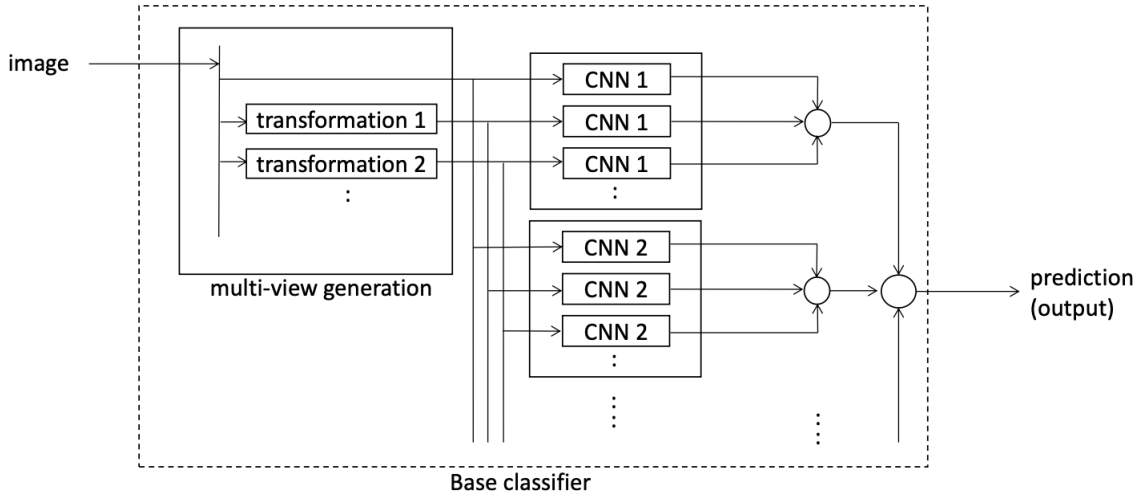

Рисунок 1.3. – Базова структура класифікатора в роботі “Класифікація зображень меланоми, невуса та себорейного кератозу” [13]

Аугментовані зображення в той же час є вхідними даними для класифікатора на основі ансамбля згорткових нейронних мереж, вихідними даними для якого є бінарний вектор, що відображає належність екземпляру до певного класу . В роботі, що описується була адаптована мережа ResNet [23] з 50 шарами імплементована за допомогою фреймворку keras з незначними модифікаціями.

До того як робити прогноз, може бути перевірена інформація про стать та вік пацієнта. На тренувальному наборі даних було помічено, що себорейний кератоз та меланома достатньо рідкі явища для пацієнтів молодшого віку. Також було виявлено досить прямо порогове розділення по віку та статі для класифікації на себорейний кератоз. Для меланоми не було помічено ніяких змін результатів при кросс-валідації

Додатково, було помічено, що класифікатор себорейного кератозу був більш надійним ніж класифікатор меланоми. Границя класу себорейного кератозу має бути краще представлена класифікатором себорейного кератозу ніж границя меланоми, хоча вони були навчені на одному наборі даних, тож було прийнято рішення про інтеграцію інформації до класифікатору меланоми.

Тобто якщо певний об’єкт скоріше за все відноситься до класу себорейного кератозу, він ймовірніше за все не відноситься до класу меланоми.

На ранніх етапах розвитку CNN люди зазвичай використовували мережі, що самостійно будували, для виконання конкретного завдання. Наприклад [24], представив самоконтрольовану модель для виявлення меланоми. По-перше, для навчання мічених і не мічених зображень використовували метод опорних векторів. Після цього був використаний підхід початкового завантаження для випадкового вибору навчальних зображень для мережі, щоб покращити здатність до узагальнення та зменшити надмірність моделі. Експерименти показали, що запропонований метод перевершив інші методи. Потім в роботі [25] було розроблено просту згорткову мережу для виявлення меланоми. По-перше, усі вхідні зображення були попередньо оброблені для усунення ефектів шуму та завад. Потім оброблені зображення були подані в попередньо підготовлену згорткову мережу, щоб визначити, чи є вони меланомою чи доброякісністю пухлиною. Нарешті, результати експерименту показали, що підхід з використанням згорткової нейронної мережі перевершив інші методи класифікації.

З розвитком глибокого навчання різні відомі мережі, такі як VGGNet [26], GoogleNet [27] і ResNet [28], були застосовані для класифікації раку шкіри з позитивними результатами. Найбільш знаковою була робота [29]. Це був перший раз, коли згорткову нейронну мережу було використано для підготовки великомасштабних клінічних зображень для класифікації раку шкіри. Вони розробили наскрізну мережу для автоматизованої класифікації раку шкіри за допомогою Inception v3. Загалом для навчання моделі було використано 129 450 клінічних зображень із 2032 різними захворюваннями шкіри. Тим часом, щоб використовувати дрібнозернисту інформацію в таксономічній структурі, вони запропонували алгоритм розподілу захворювань, щоб розділити рак шкіри на дрібнозернисті класи (наприклад, меланома була підрозділена на амеланотичну меланому та акролентигінозну

меланому). Зрештою, результати експериментів показали, що система класифікації раку шкіри може досягти рівня діагностики, еквівалентного дерматологам. У тому ж році [30] успішно впроваджено VGGNet для класифікації уражень шкіри (меланома чи доброякісна пухлина) і порівняння для наборів даних ISIC за допомогою дермоскопічних зображень. У цьому дослідженні вони розробили три різні модулі на основі VGG-16 для порівняння. Перший модуль навчав мережу з початкових ваг. Другий модуль використовував попередньо підготовлений VGG-16 для навчання, а потім використовував поточний набір даних для навчання повністю підключеного класифікатора. Третій модуль також використовував навчання передачі для навчання мережі, але ваги в частині високого рівня згорткових шарів були ініціалізовані з першого модуля. Зрештою, результати показали, що третій модуль отримав відмінні результати в класифікації раку шкіри. На відміну від попередніх класифікаційних завдань [31], для оцінки товщини меланоми вперше використано структуру CNN (VGG-19). Вони почали з визначення місця ураження та виділення областей інтересу (ROI). Вирішити проблему обмеження даних і дисбалансу даних; потім вони застосували техніку надмірної вибірки синтетичної меншості для створення синтетичних зразків. Після цього попередньо навчений VGG-19 використовувався для прогнозування товщини. Нарешті, результати продемонстрували, що алгоритм може оцінити товщину меланоми з точністю 87,5%. Вперше багатозадачна мережа була запропонована [32] на основі Inception v3 шляхом використання трьох різних модальностей даних для прогнозування семиточкового критерію. Крім того, вони розробили мультимодально-багатозадачну функцію втрат для роботи з комбінаціями вхідних модальностей, яка також могла робити прогнози з неповною інформацією. Нарешті, результати показали найкращу ефективність у класифікації уражень шкіри. Крім того, запропонований метод мав здатність ідентифікувати нерелевантну інформацію та генерувати вектори ознак для пошуку зображення. В роботі [33] було створено дві системи для

класифікації шкірних захворювань на основі нових алгоритмів глибокого навчання. Крім того, вони додали новий шар для підвищення чутливості моделі. У першій системі була запропонована згорткова архітектура на основі Inception v2 для ідентифікації пухлин шкіри (доброякісних або злоякісних) за допомогою дермоскопічних зображень. Друга система перетворила представлення ознак, згенероване в попередній системі, на звукові дані. Потім цю звукову інформацію поміщали в класифікатор машинного навчання або переводили на спектрограми для подальшого аналізу. Зрештою, обидві системи продемонстрували виняткові результати з точки зору класифікації та озвучення. Після того, як методи глибокого навчання досягли достатніх результатів у завданні класифікації раку шкіри d в роботі [34] було запропоновано, як покращити класифікацію дермоскопії на основі глибокого навчання та створення набору даних. Вони проаналізували чотири архітектури ResNet у дермоскопічній класифікації, а саме ResNet-34, ResNet-50, ResNet-101 та ResNet-152, щоб зрозуміти механізми та певні причини помилок. По-перше, чотири мережі ResNet були навчені на їхній найкращий підхід, щоб побачити, чи призведуть структурні відмінності між моделями до різних результатів класифікації. Після тестування з кількома епохами вони виявили, що точність різних моделей, як правило, була узгодженою та змінювалась залежно від різних налаштувань гіперпараметрів. При цьому вони мали високий рівень стабільності під час тренувань. Таким чином, помилки навчання моделей класифікації були пов’язані з неправильними анотаціями та складністю медичних зображень.

Було виявлено, що застосування однієї згорткової нейронної мережі до КСД зазвичай не дає бажаних результатів через великі розбіжності в глибоких нейронних мережах. Після цього було запропоновано ансамблеве навчання як спосіб обмежити помилку, створювану однією моделлю, шляхом навчання кількох моделей і подальшого об’єднання їхніх результатів для отримання кінцевих результатів класифікації. В роботі [35] порівняли продуктивність між

моделями ансамблю та окремою моделлю, використовуючи дев’ять різних архітектур CNN у класифікації раку шкіри. Після різних порівняльних експериментів вони виявили значення ансамблевого навчання для отримання оптимальних моделей класифікації. Крім того, вони досліджували ефективність між двома різними стратегіями вибору в ансамблевому навчанні: випадковим вибором і використанням набору перевірки. Для менших моделей ансамблю вони виявили, що другий метод мав більше переваг, але перший також був ефективним. Для більших моделей ансамблю можна було просто вибрати моделі довільно. На основі того самого методу [36] запропоновано два різні методи класифікації раку шкіри при зменшенні складності моделі за допомогою стратегії ОПВ: 1) окрема модель згорткової мережі і 2) включення семи згорткових моделей. У першому методі зображення з набору даних безпосередньо поміщалися в єдину згорткову модель для остаточного прогнозу. У другому методі стратегія «один проти всіх» (ОПВ) використовувалася для поєднання семи окремих моделей із двома класами для отримання остаточного прогнозу. Кожен клас у цьому методі був класифікований відповідно до істинних і хибних міток, таким чином підвищуючи ефективність моделі. Результати показали, що другий метод перевершує перший з точки зору точності класифікації. В роботі [37] прийняли стратегію пошуку в сітці, щоб знайти найкращі методи вивчення ансамблю для класифікації семи уражень шкіри. Під час навчання п’ять мереж CNN: ResNeXt, SeResNeXt, ResNet, Xception і DenseNet були використані як базова лінія. Після цього було проведено дві стратегії навчання ансамблю, а саме, середній ансамбль і зважений ансамбль, щоб знайти оптимальну модель. Зрештою, результати показали, що модель зваженого ансамблю має більше переваг, ніж модель середнього ансамблю.

1.4 Висновки

В цьому розділі було проведено ознайомлення з предметною областю. Було наведено опис раку як явища, особливості раку шкіри. Було сформульовано особливості раку шкіри та актуальність створення системи комп’ютерної діагностики раку шкіри. Рак шкіри є складним, часто летальним захворюванням рання діагностика якого може значно підвищити шанси пацієнта на повне одужання

Було проведено аналіз робіт в яких вже створювалась комп’ютерна система діагностики раку шкіри. Для задачі було застосовано багато стандартних підходів, зокрема на основі згорткових нейронних мереж як, наприклад VGG, ResNet та ін.

Ці нейронні мережі показали досить непогані результати, тим не менш загальновідомим є факт, що працюють вони досить повільно і, часто, потребують відносно великих потужностей для навчання і використання.

Тож можна зробити висновок, що в ситуації, коли класичні моделі хоч і показують задовільні результати є повільними і вимогливими до обчислювальних потужностей є місце для більш швидких і менш вимогливих до техніки моделей

РОЗДІЛ 2 ЗОРОВИЙ ТРАНСФОРМЕР

2.1 Архітектура моделі

Сьогодні в задачах обробки природної мови (NLP), трансформери стали стандартними моделями (наприклад, BERT, GPT-3 і т. д.). Тим не менш, можливості трансформерів у задачах комп'ютерного зору все ще є досить обмеженими. Більшість дослідників використовують згорткові мережі напряму, чи так чи інакше використовують згорткові шари у мережах (такі моделі як Xception, ResNet, EfficientNet, DenseNet, Inception).

Стаття про зоровий трансформер "Картинка це як 16x16 слів" [4] демонструє реалізацію чистої моделі трансформера без потреби в згорткових шарах. Стаття показує, як використовуючи зоровий трансформер можна отримати кращі результати ніж використовуючи будь яку згорткову модель в задачі розпізнавання картинок, при цьому використовуючи відносно менші ресурси [4].

Трансформери це нейронні мережі, що оперують послідовностями даних (наприклад, набором слів). Ці набори слів токенізуються і потім подаються на вхід трансформеру. Трансформери використовують функцію Уваги – квадратичну операцію, що обчислює попарний добуток між парами токенізованих слів. Зі зростанням кількості слів, росте і кількість операцій.

З картинками трансформер використовувати дещо важче. Картинка складається з тисяч чи мільйонів пікселів. Таким чином, якщо трансформер виконує попарну операцію між кожною парою пікселів на зображенні, складність виростає до складно обчислювального рівня. Тому для картинок використовують дещо іншу форму локальної функції уваги замість глобальної [5].

Автори зорового трансформера використовують функцію уваги, не для всієї картинки, а для багатьох вхідних частин картинки. Тож, на першому етапі

картинка розбивається на патчі (тобто менші частини зображення) розміром 16 на 16.

Рис. 2.1 – Розбиття картинки на патчі [5].

Потім створюється послідовність з патчів (рисунок 2.2).

Рисунок 2.2 – Послідовність з патчів[5].

Через нумерацію патчів моделі надається інформація про їх позицію. Це відбувається за допомогою пошукової таблиці, що зберігає вектор для кожного номеру позиції патчу. Тож, для першої підкртинки вектор з таблиці подається у модель з першим патчем. Аналогічно з іншими патчами.

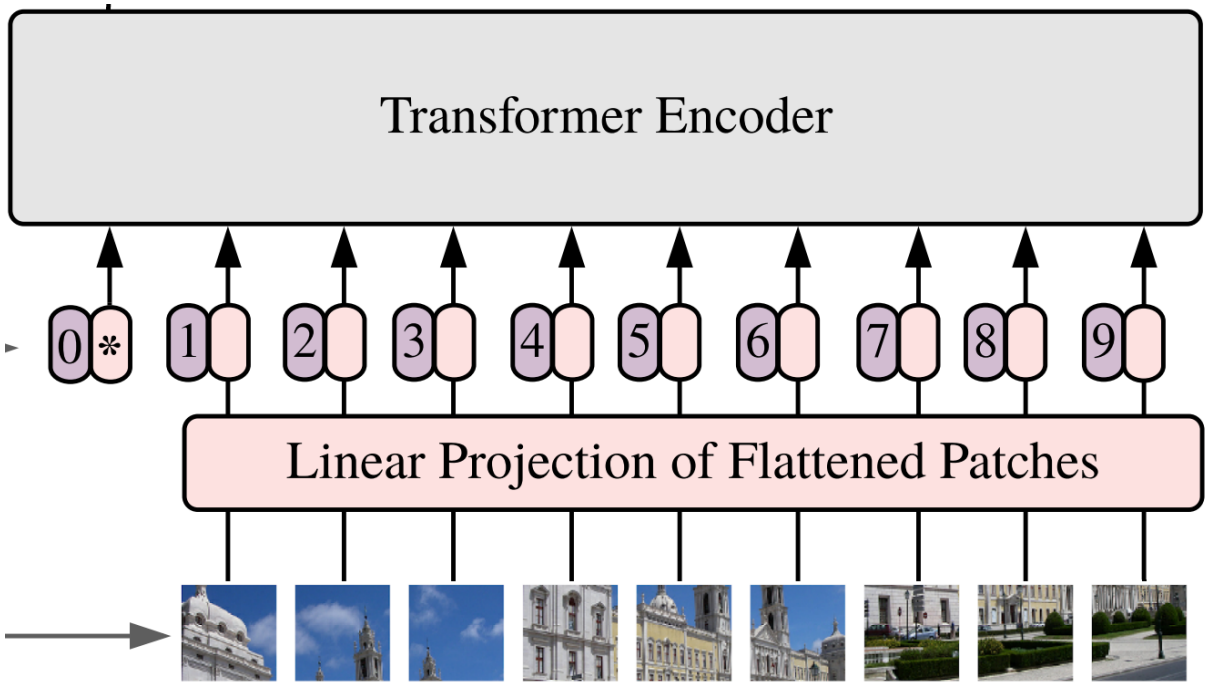

Рис. 2.3 – Патчі та вектори подаються парами на вхід у модель [5].

Патч це зображення розміром 16 на 16. Вона має бути подана на вхід трансформеру у форматі прийнятному для обробки. Один зі способів це зробити це перетворити матрицю 16 на 16 на вектор розміром 256. Тим не менш, автори статті використали Лінійну проекцію. Це означає, що підкартинки зазнають однакового лінійного перетворення. Після цього векторні представлення підкартинок подаються на вхід до трансформеру.

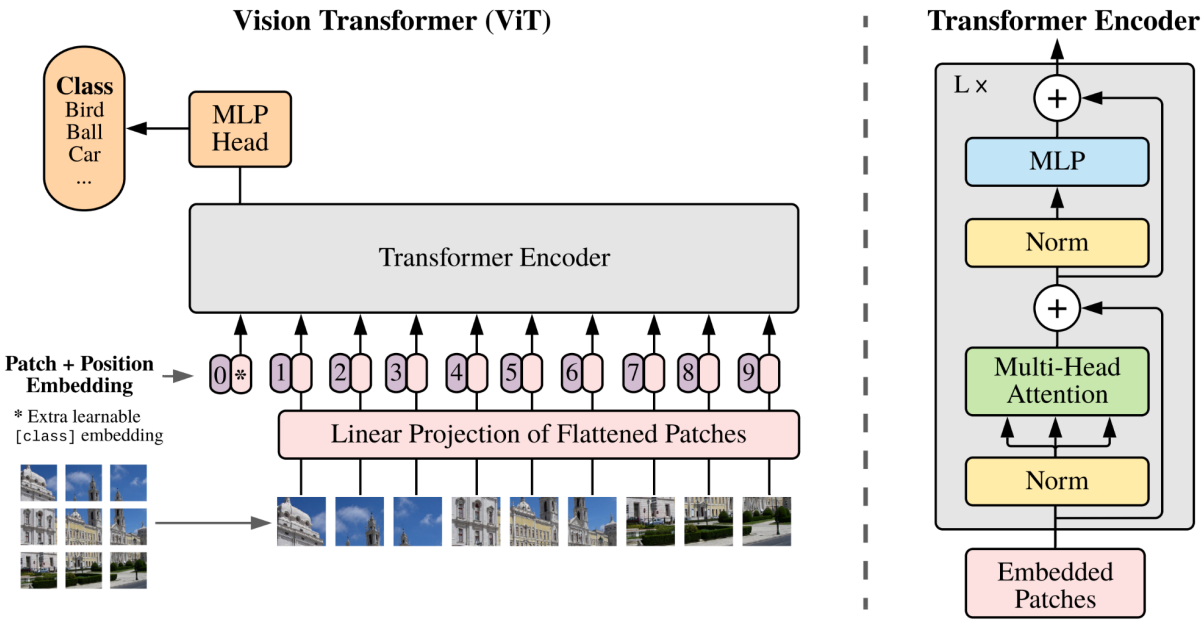

Рис. 2.4 – Загальна схема зорового трансформера [5].

2.2 Функція уваги

Механізм функції був представлений у 2014 як вирішення проблеми яка виникає при використанні вектора кодування сталої довжини в моделях типу кодувальник-декодувальник. Вона полягає в тому, що декодувальник може мати обмежений доступ до інформації представленої моделі на вході.

В загальному при обчисленні функції уваги використовуються 3 ключові компоненти: Запит (Query), Ключ (Key) та Значення (Value). Це можуть бути 3 однакові матриці.

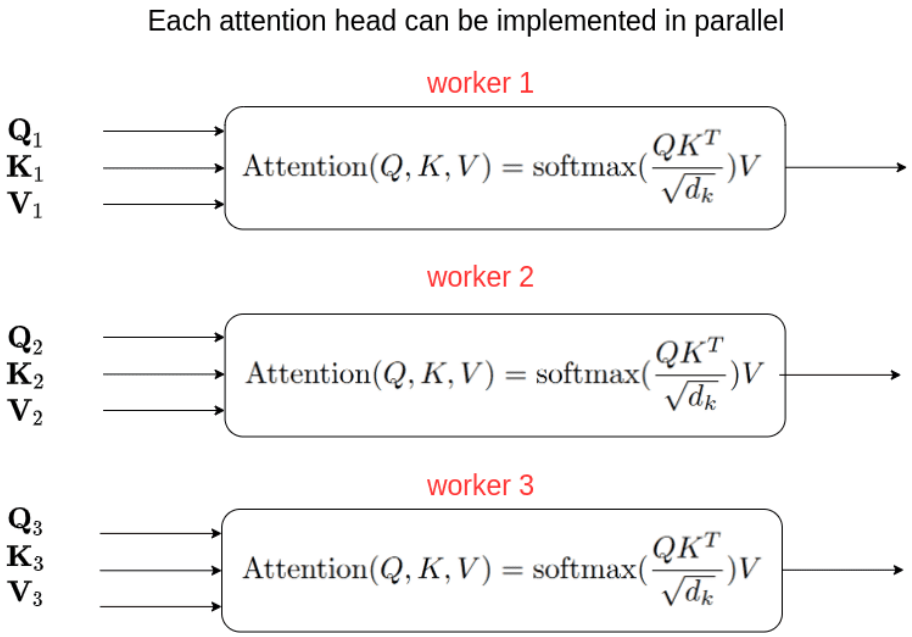

Рис. 2.5 – Схема функції мультиуваги

Значення уваги обчислюється за допомогою матричного множення Запиту на Ключ, ділення на корінь з розмірності Ключа, застосування функції softmax та матричного множення на Значення. Формулу можна побачити на рисунку 2.5.

В цій роботі була застосована функція мультиуваги, що застосовує кілька функцій уваги до патчів одночасно для виділення різних фіч.

2.3 Набори даних

Щоб створити надійну та надійну систему класифікації раку шкіри, потрібні різноманітні набори даних із усіма видами дерматологічних зображень. Оскільки потреба в ресурсах медичної візуалізації в академічних колах зростає, все більше і більше наборів даних стають загальнодоступними. Далі представлені теоретичні відомості про популярні та найбільш змістовні набори даних, що використовуються чи можуть бути потенційно використані для вирішення поставленої задачі.

2.3.1. PH2

Цей набір даних [38] побудовано за [39] для підтримки досліджень методів класифікації та сегментації. Він містить 200 кольорових дермоскопічних зображень (768 × 560) трьох типів шкірних захворювань, включаючи звичайні невуси, атипові невуси та меланоми. Крім того, він містить повні медичні анотації, такі як результати сегментації ураження та патологічний діагноз.

PH2 часто використовується як набір даних для тестування діагностичних алгоритмів захворювань шкіри. Наприклад в роботі [40], була використана структуру SegNet для автоматичної діагностики та сегментації дермоскопічних зображень у PH2. В роботі [41] була запропонована багатофокусна мережа сегментації для завдань сегментації раку шкіри на основі набору даних PH2 за допомогою карт ознак різних масштабів. Два модулі граничної уваги та два модулі зворотної уваги були використані для створення масок пошкоджень шкіри. На додаток до вищезазначених робіт, набір даних PH2 використовується все більшою кількістю алгоритмів для перевірки їх ефективності та точності.

2.3.2. MED-NODE

Набір даних MED-NODE [42] містить в собі дані зібрані відділом дерматології Університетського медичного центру Гронінгена (UMCG), який містить 170 цифрових зображень меланоми [43] і випадків невусів [44]. Він використовується для створення та оцінки системи MED-NODE для виявлення раку шкіри за допомогою макроскопічних зображень [45]. У наборі даних MED-NODE різноманітні підходи дали значні результати класифікації. був випадок застосування AlexNet для завдання класифікації раку шкіри за

допомогою трьох різних методів навчання передачі, включаючи точне налаштування вагових параметрів моделі, заміну функції шару класифікації та виконання доповнення даних до вихідного набору даних. Згорткові нейронні мережі отримали гідні результати класифікації на цьому наборі даних; однак кількість включених зображень захворювань шкіри відносно обмежена.

2.3.3. HAM10000

Набір даних HAM100004 [46] було зібрано організацією International Skin Imaging Collaboration (ISIC) для вирішення проблеми дисбалансу та обмеження даних у наборах даних про захворювання шкіри. Він містить 10015 дермоскопічних зображень із сімома репрезентативними захворюваннями пігментних уражень шкіри: нематодозом і внутрішньоепітеліальною карциномою, базально-клітинною карциномою, доброякісними кератоїдними ураженнями, фібромою шкіри, меланомою, меланоцитарними невусами та судинними ураженнями (включаючи гемангіоми, гнійні гранульоми та підшкірні крововиливи).

Набір даних HAM10000 широко використовується багатьма вченими через його різноманітність уражень шкіри. Наприклад [47], використовували чотири нові глибокі моделі CNN, DenseNet-201, ResNet-152, Inception-v3 і InceptionResNet-v2, щоб класифікувати вісім різних типів раку шкіри на наборах даних HAM10000 і PH2.

Нарешті, експериментальні результати показали, що діагностичний рівень цих моделей CNN перевищує рівень дерматологів з точки зору ROC AUC [48]. навчив 30 різних моделей на наборі даних HAM10000, щоб дослідити ефективність класифікації різних моделей. У той же час вони також використовували два локально інтерпретовані методи GradCAM і техніку Kernel SHAP для спостереження за механізмом моделі класифікації. Завдяки поглибленому вивченню завдань класифікації раку шкіри вченими все більше

нових методів класифікації випробовуються на наборі даних HAM10000 для кращого порівняння, де застосування модуля Soft-Attention дає найкращі результати класифікації.

2.3.4. Derm7pt

Набір даних Derm7pt містить приблизно 2000 клінічних і дермоскопічних кольорових зображень шкірних захворювань, а також структуровану інформацію для навчання та оцінки систем CAD. Він служить базою даних для аналізу результатів прогнозування семиточкового контрольного списку злоякісних утворень шкіри [49].

Завдяки мультимодальній інформації, що міститься в наборі даних Derm7pt, його поступово почали широко використовувати для тестування різних багатозадачних мереж. Під час випуску набору даних [50] також запропоновано багатозадачну мережу для прогнозування меланоми з критеріями контрольного списку із семи пунктів і результатами діагностики. Модель використовувала різні функції втрат для обробки різних модальностей введення, водночас маючи можливість робити прогнози щодо відсутніх даних на виході. Нарешті, модель досягла точності класифікації 73,7% на наборі даних Derm7pt, що також порівнює підхід. Щоб підвищити його інтерпретативність [50], створено багатозадачну модель на основі набору даних Derm7pt, щоб проілюструвати механізм між різними завданнями. Шлюзи, які можна вивчати, використовувалися в моделі, щоб показати, як метод використовує або поєднує функції з різних завдань. Ця стратегія може бути використана для дослідження того, як поводяться моделі CNN, потенційно підвищуючи їхню клінічну корисність [51]. запропонував глибоку згорточну мережу для класифікації уражень шкіри на наборі даних Derm7pt. Тим часом вони реалізували регуляризовані DropOut і DropBlock, щоб збільшити можливості узагальнення моделі та зменшити переобладнання.

Крім того, щоб усунути дисбаланс і обмеження набору даних, вони розробили нову функцію втрат, яка призначає різні ваги різним зразкам, а також метод наскрізного накопичувального навчання. Нарешті, метод досяг відмінної продуктивності класифікації на наборі даних Derm7pt і наборі даних ISIC при низьких обчислювальних ресурсах. Випуск набору даних Derm7pt має великий поштовх у просуванні використання мультимодальних даних у задачах класифікації раку шкіри, а також нових ідей і рішень.

2.3.5. BCN20000

Набір даних BCN200005 [52] містить 5 583 пошкодження шкіри та 19 424 дермоскопічних зображення, зроблених за допомогою дермоскопії з високою роздільною здатністю. Усі вони були зібрані між 2010 і 2016 роками. У той же час колекціонер застосував різноманітні методи комп'ютерного зору, щоб видалити шум, фон та інші перешкоди із зображень. Нарешті, вони були ретельно розглянуті численними експертами, щоб переконатися в достовірності діагнозу [53].

BCN20000 зазвичай використовується в класифікації та сегментації раку шкіри як частина набору даних для конкурсу ISIC-2019. Наприклад, щоб захистити конфіденційність даних і уникнути зловживання даними [54], запропоновано метод розподіленої аналітики для розподіленого навчання зображень шкірних захворювань, який гарантує, що навчальні дані залишаються в початковій установі. Нарешті, після навчання на наборі даних BCN20000 модель досягає точності класифікації, порівнянної з централізованим розподілом. Щоб оцінити стійкість різних моделей CNN [55], створив серію зображень поза розповсюдженням (OOD) за допомогою різних методів доповнення даних на основі BCN20000, HAM10000 та інших наборів даних про шкірні захворювання. Цей метод встановлює еталон для тестування OOD і значно полегшує клінічне використання методів класифікації раку

шкіри. Зокрема, використовуючи різні методи доповнення даних із стратегією ансамблевого навчання (включно з EfficientNets, SENet і ResNeXt101_wsl) [56], було досягнуто першого результату класифікації зі збалансованою точністю 74,2% на наборі даних BCN20000.

2.3.6. ISIC

Щоб зменшити смертність від раку шкіри, одночасно сприяючи розробці та використанню цифрових зображень шкіри [57], Міжнародна співпраця з візуалізації шкіри (ISIC) створила загальнодоступний набір даних про захворювання шкіри [52] для спільноти інформатики в усьому світі. Наразі архів ISIC містить понад 13 000 репрезентативних дермоскопічних зображень із клінічних закладів по всьому світу, усі з яких були перевірені та анотовані експертами для забезпечення якості зображень [58].

Більшість досліджень, у яких використовувався набір даних ISIC, були зосереджені на задачах класифікації та сегментації раку шкіри, причому найпопулярнішою була задача бінарної класифікації. Наприклад [59], розробив різні модулі на основі VGGNet для класифікації шкірних захворювань (меланоми або доброякісних) і порівняв їх з набором даних ISIC-2016. В роботі [60] використовували дві методики усунення упереджень: «Навчання, щоб не вчитися» (LNTL) і «Закривання очей» (TABE), щоб зменшити нерівності в прогнозах моделі та помилкові зміни в зображеннях меланоми. Серед них метод LNTL поєднав нову втрату регуляризації з шаром градієнтної інверсії, щоб дозволити моделі відхиляти функції CNN у зворотному поширенні. Метод TABE зменшив зміщення, використовуючи різні допоміжні класифікатори для виявлення зміщень у функціях. Нарешті, експериментальні результати показали, що TABE має більш ефективний ефект шумопоглинання з покращенням на 11,6% показника AUC за набором даних ISIC. Оскільки набір даних ISIC широко використовується в конкурсах і дослідженнях, читачі

можуть знайти більше методів для порівняння в таблиці лідерів конкурсу або в Інтернеті.

2.4 Висновки

В цьому розділі було наведено опис зорового трансформера як моделі машинного навчання. Була пояснена архітектура моделі, її відмінності від згорткових мереж та потенціал в задачах класифікації. Було наведено опис наборів даних, що були застосовані у роботі для навчання зорового трансформера.

Можна зробити висновок, що зоровий трансформер є моделлю з великим потенціалом до ефективного вирішення задачі класифікації раку шкіри. Зорові трансформери потребують менше ресурсів для навчання ніж більшість згорткових мереж, в той же час перевершуючи їх по якості на великих наборах даних. Крім того зорові трансформери працюють, зазвичай, швидше.

Тобто дослідження зорового трансформера в задачі класифікації раку шкіри є доцільним

РОЗДІЛ 3 ЕКСПЕРИМЕНТИ І РЕЗУЛЬТАТИ

3.1 Інструменти

Для навчання більшості моделей машинного навчання потрібно мати доступ до графічного процесора. Зорові трансформери також не є виключенням. Через апаратні обмеження, експерименти неможливо було проводити локально, тому за середовище для розробки моделі був обраний Google Colab. Він забезпечує доступ до малопотужного графічного процесора за допомогою якого була навчені всі згадані моделі. В якості фреймворку розробки був обраний keras через доступність інформації, щодо реалізації зорових трансформерів з його допомогою.

Іноді, все ж, Google Colab не дозволяв продуктивно використовувати параметри моделі. Наприклад, при розбитті на патчі, графічної потужності не хватало для підтримки розбиття зображення 100x100 на патчі розміром 5x5, тобто 400 патчів з зображення. Тому іноді, доводилось підлаштовувати архітектуру моделі до таких апаратних обмежень.

Для розробки додатка використовувся фреймворк Flask. Він був вибраний через швидкість розробки невеликого додатку на ньому та мову Python, тобто моделі машинного навчання можуть бути використані так само, як і при їх створенні без пошуку додаткових фреймворків.

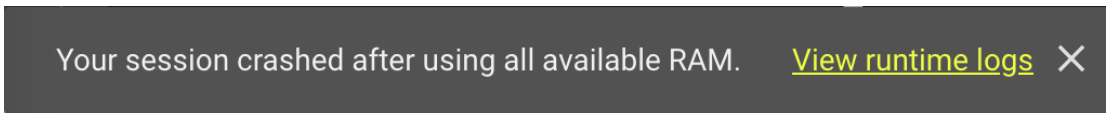

Рис. 3.1. - Помилка в Google Colab при тренуванні моделі

3.2 Набір даних

Для навчання був використаний набір синтезований з вищезгаданих наборів даних (PH2, MED-NODE, HAM10000, Derm7pt, BCN20000, ISIC). Так

як всі вони мають зображення в різних форматах, всі вони були приведені до трьоканальних зображень розміром 100x100.

Для попередньої обробки зображення з набору даних маштабувалися до розміру 100x100. Також розглядалися варіанти з підвищенням контрастності зображень та переводу в один канал (чорно-білий колір). Та все ж, навчання моделей на зображеннях з оригінальними кольорами виявилося більш результативним.

Для збільшення розмірів датасету та, як наслідок, покращення якості моделі в результаті, застосовувалась аугментація – горизонтальне відображення та зміна масштабу зображення.

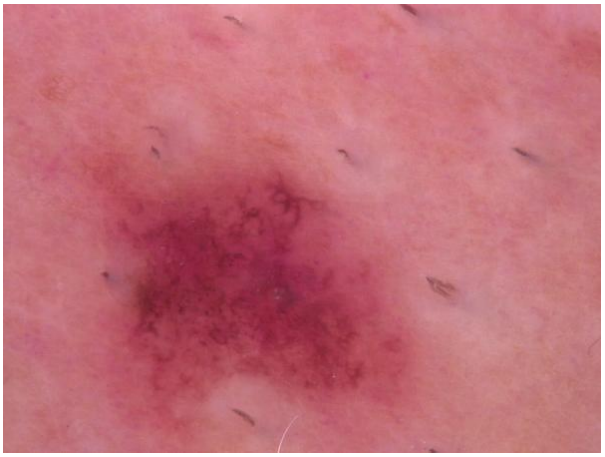

Рис. 3.2 – Приклад зображення з тренувального набору даних

Рис. 3.3 – Зображення розміром 100x100

3.3 Моделі

В експериментах були застосовані різні архітектури зорового трансформера. Вони включали в себе різну кількість шарів-трансформерів, різні функції уваги та різні масштаби зображень, що обробляються. Ціль дослідження архітектур та параметрів це визначення моделі, що буде давати найкращий результат на тестовому наборі даних за метриками класифікації ключовими з яких будуть accuracy та кількість помилок першого та другого роду.

3.3.1. Базова архітектура

Базова архітектура моделі представлена на малюнку 3.3. Вона включає в себе вхідний шар, розбиття на патчі, кодування патчів, 1-10 шарів трансформерів та класифікацію.

Модель приймає на вхід набір з трьохканальних зобржень 100x100 на яких відбувається аугментація. До складу аугментації входять зміна розміру зображення, випадкове горизонтальне відображення, випадкові повороти та збільшення.

Далі йде розбиття на патчі. Кількість та розмір патчів є параметрами моделі. Для прикладу, нехай розмір зображення залишиться 100x100 та матимемо патчі розміром 10. Тоді кожен патч розгортається в послідовність пікселів (всі три канала разом), тому в результаті зображення міняє розмір на 300x100, тобто 100 патчів і 300 пікселів в одному патчі (три канали). Під час кодування патчів, до них додається позиція їх на зображенні

В таблицях така архітектура буде називатися base_vit.

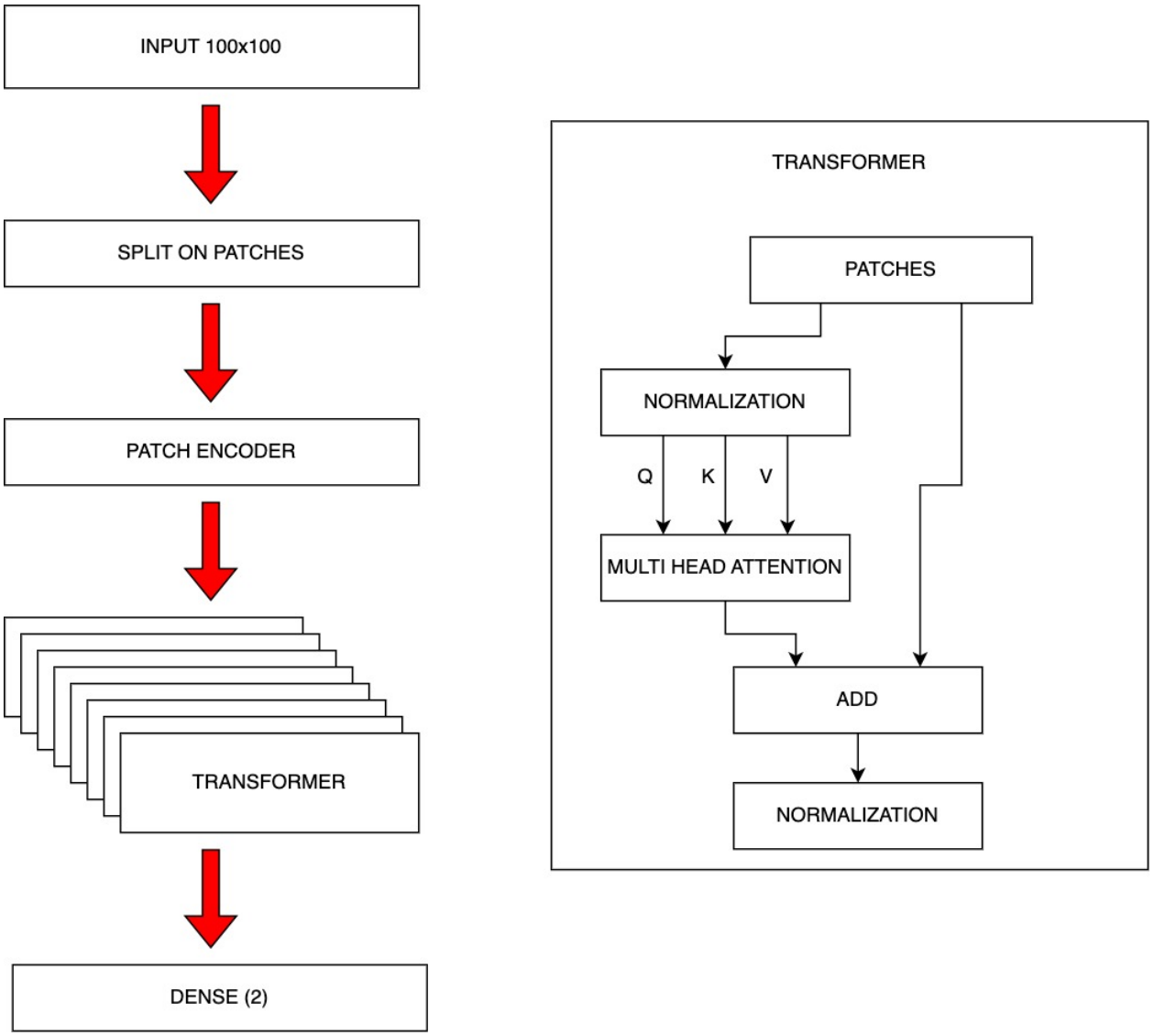

Рис. 3.3 – Архітектура моделі зорового трансформера

3.3.2. CrossViT

Основна ідея перехресної функції уваги полягає в використанні cls токену та токенів патчів з різних гілок. Для більш ефективного злиття ознак різного масштабу cls токен однієї гілки використовується як агент обміну інформацією з патч токенами іншої гілки та зворотної її проекції. Так як cls токен вже має абстрактну інформацію з патч токенів своєї гілки, взаємодія з патч токенами іншої гілки допомагає додати в них інформацію іншого масштабу. Після злиття з токенами іншої гілки, cls токен взаємодіє знову з патч токенами своєї гілки, де він може передати їм отриману інформацію з іншої

гілки на наступному етапі кодування трансформера, щоб досягти репрезентації кожного патч токена [61].

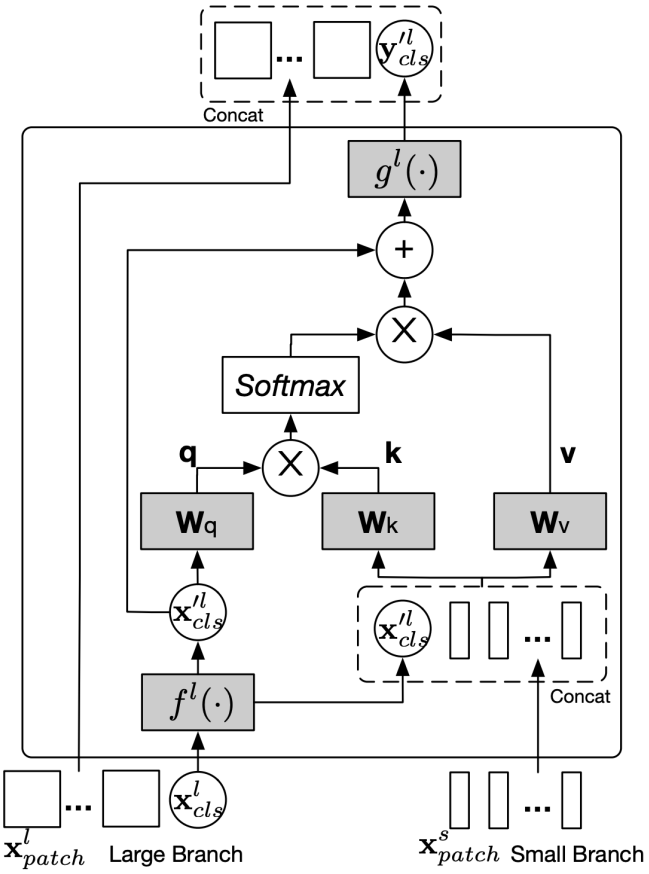

Рис. 3.4 – CrossViT, схема функції уваги [61]

Використання CrossViT може допомогти з виділення областей уваги на зображенні, особливо, враховуючи нерівні краї шкірних утворень, що не завжди можуть поміщатися в патчі.

Архітектура буде позначатися в таблицях як cross_base_vit.

3.3.3. Модель, що використовує додаткові більші патчі

Вище вже було згадано, що утворення на шкірі не завжди рівномірно розподіляються по патчам. Як наслідок, околиці не завжди потрапляють в області уваги. Приклад можна побачити на малюнку.

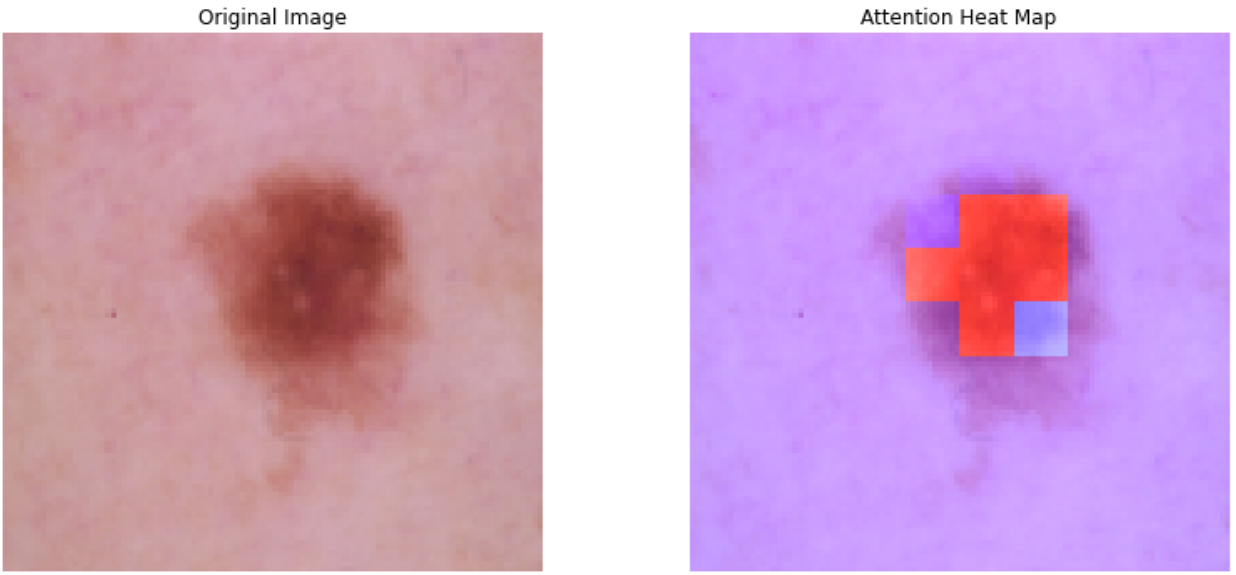

Рис. 3.5 – Теплова карта функції уваги

Для вирішення цієї проблеми в роботі був запропоновано використання додаткових більших патчів для збільшення значення уваги патчів-околиць.

Функція уваги однаково паралельно відпрацьовує для патчів обох розмірів на кожному застосуванні функції уваги. Перед наступним шаром функції уваги, значення уваги більших патчів проектуються на значення менших патчів, додаються та нормалізуються. Як результат, значення уваги для околиць збільшується відносно центру.

Така архітектура буде позначатися в роботі як merge_vit

На рисунку 3.6 значення жовтих патчів будуть додані до менших патчів, що вони зачіпають. Як результат, патчі на околиці можуть мати більше значення уваги.

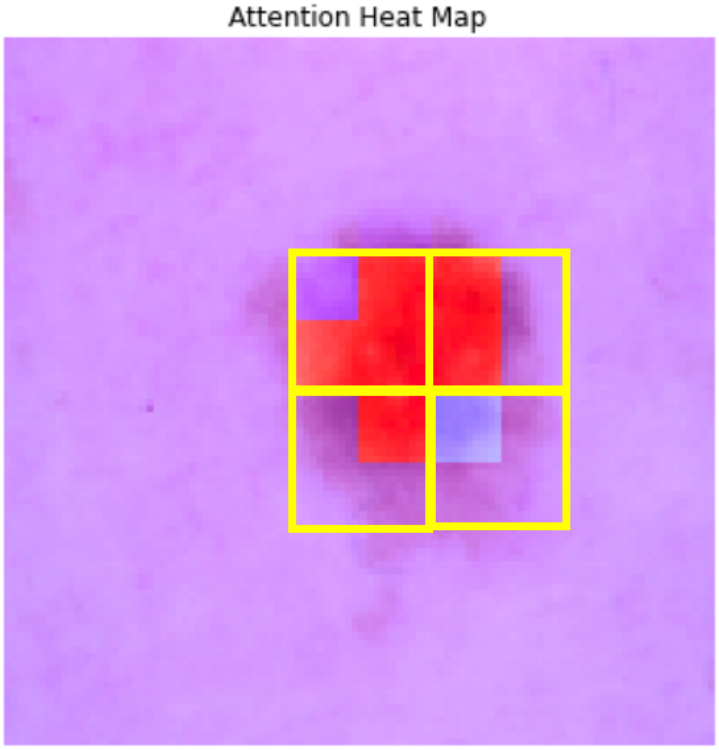

Рис. 3.6 – На рисунку показані більші патчі, що зачіпають також околиці шкірного утворення

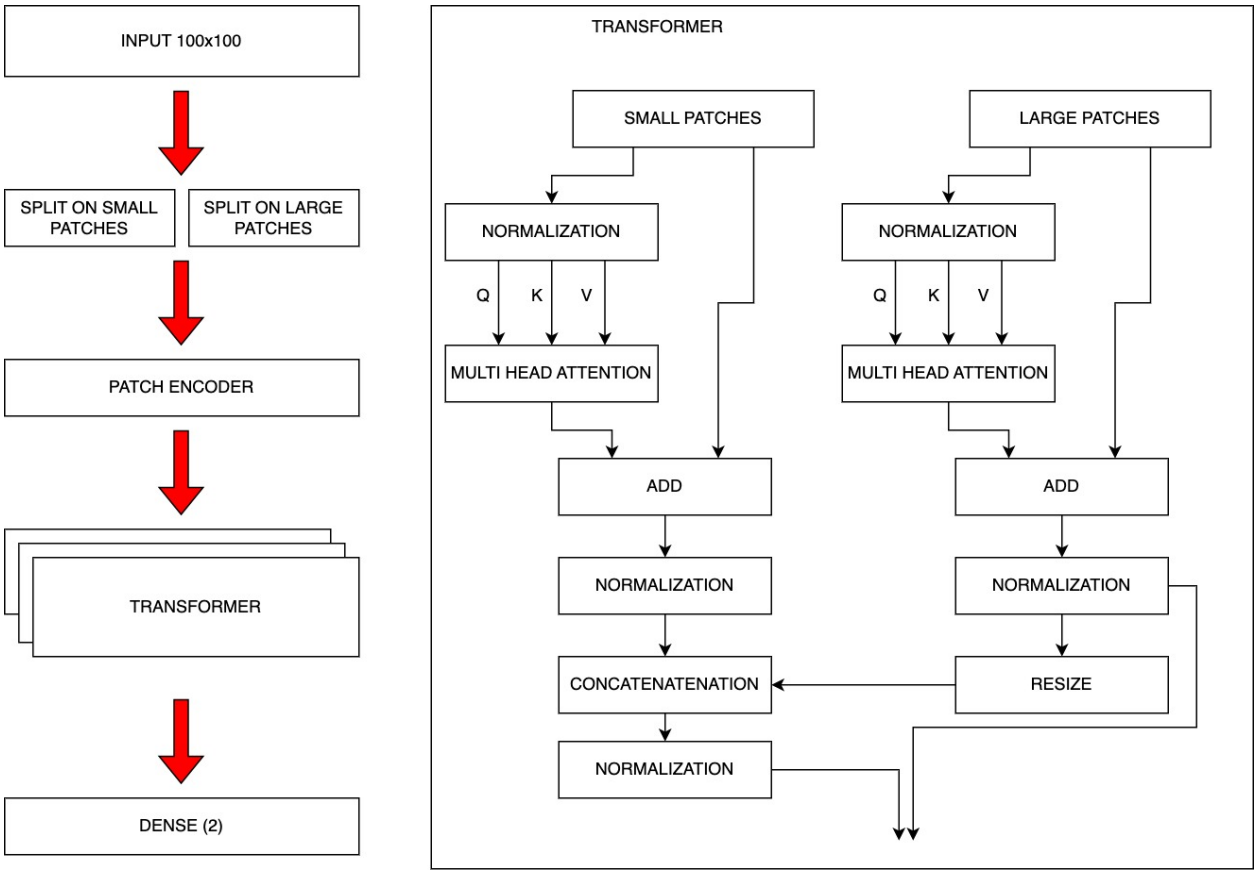

Рис. 3.8 – На рисунку зображена схема зорового трансформера із застосуванням більших патчів. Після кожної ітерації уваги, значення більших патчів додаються до менших.

3.4 Параметри

Базовими параметрами моделі є img_size, patch_size, tfrm_layers, proj_dim та att_heads. img_size відповідає розміру до якого буде зменшена картинка під час навчання. patch_size відповідає розміру однієї сторони квадратного патча на які буде розбита картинка. tfrm_layers це скільки разів буде до картинки застосована функція мультиуваги. proj_dim - це розмірність до якої буде зменшено патч. att_heads відповідає за те скільки одиничних функцій уваги функцій буде застосовано до зображення.

Для моделей, що мають два типи патчів параметр patch_size будуть зазначенний у форматі (x, y), де x та y розміри меншого та більшого типів патчів відповідно.

Інші параметри моделі менше впливали на точність моделі при їх зміні. Кількість епох визначалася динамічно при навчанні кожної моделі – зупинка проводилась, коли значення accuracy на валідаційному наборі даних не покращилося більше 15 епох.

У якості оптимізатора був використаний AdamW з додатку до фреймворку tensorflow – tensorflow-addon. Це стохастичний оптимізатор моделі машинного навчання, що модифікує стандартне затухання вагів в алгоритмі Adam роблячи його менш залежним від оновлення значення градієнта. Він був використаний головним чином через швидкість роботи (що було досить важливо так як обране середовище дозволяє використовувати графічні процесори обмежену кількість часу).

3.5 Метрики

За головну метрику за якою йде спостереження під час навчання моделі та за якою визначають найкращу модель є accuracy, тобто відсоток зображень з тестової вибірки, що класифіковані правильно.

Також для порівняння було враховано помилки першого та другого роду. Так як задача є медичною, прийнятим стандартом є те, що помилка “першого” роду є кращою за помилку другого роду. На наш випадок це також розповсюджується через те, що невус класифікований як меланома може мати найгірші наслідки лише як додатковий візит до лікаря. Помилка же другого роду, тобто меланома класифікована як невус в даному випадку може призвести до затримки початку лікування, що є найгіршим можливим наслідком помилки.

Варто зазначити, що порівняння абсолютних значень помилок першого та другого роду в даній роботі має сенс через те, що розбиття вибірки на тренувальну і тестову проводилось один раз незалежно від навчання будь якої з моделей, тобто всі моделі мають однакові тренувальну та тестову вибірки.

Інші традиційні метрики класифікація такі як precision, recall, f1-score, support та графіки навчання будуть наведені лише для найбільш ефективних моделей.

3.6 Результати навчання зорових трансформерів

В таблиці позначені результати навчання зорових трансформерів з різними архітектурами та параметрами. Ідеально було б вибирати ті ж параметри для різних архітектур відповідно, але апаратні обмеження у вигляді малопотужного графічного процесора не завжди дозволяли виставити необхідні параметри, тому параметри моделі підбирались максимально наближеними одне до одного.

3.6.1. Порівняльна таблиця моделей

В лівій стороні таблиці зображені параметри з якими створювалась модель та її архітектура. В правій три ключові метрики, що використовувались для вибору найбільш ефективних моделей

В подальшому при згадуванні якоїсь з моделей з таблиці, буде використовуватись позначення *архітектура_розмір_патча*

Таблиця 3.1 – порівняльна таблиця моделей на основі зорового трансформера

Параметри						**Метрики**		
Архітектура	*img_size*	*patch_size*	*tfrm_layers*	*proj_dim*	*att_heads*	*accuracy, %*	*помилки*	
							1 роду	*2 роду*
base_vit	100	10	8	64	4	76.05	35	45
base_vit	72	6	8	64	4	74.55	52	33
base_vit	36	2	6	36	4	72.75	50	41
base_vit	12	1	6	12	3	78.14	45	28
cross_vit	100	(10, 20)	8	64	3	67.22	70	39
cross_vit	72	(6, 8)	6	64	3	68.34	98	7
cross_vit	36	(2, 6)	6	36	2	74.12	81	5
cross_vit	12	(1, 4)	6	12	2	72.11	26	67
merge_vit	100	(10, 20)	8	64	3	60.48	132	0
merge_vit	72	(6, 12)	6	48	3	70.66	74	24
merge_vit	24	(2, 6)	6	24	2	73.65	61	27
merge_vit	12	(1, 3)	8	12	3	81.14	33	30

Як видно з таблиці, найбільш ефективними моделями моделями виявились base_vit_12 та merge_vit_12.

3.6.2. Метрики найкращих моделей

Нижче наведені розширені метрики моделей, що показали найкращі результати при навчанні

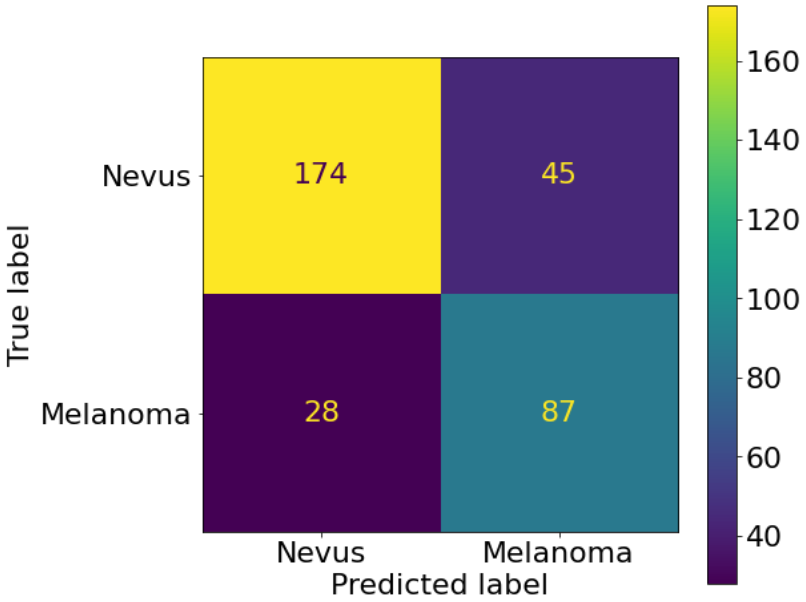

Рис. 3.9 – Матриця невідповідностей моделі base_vit_12

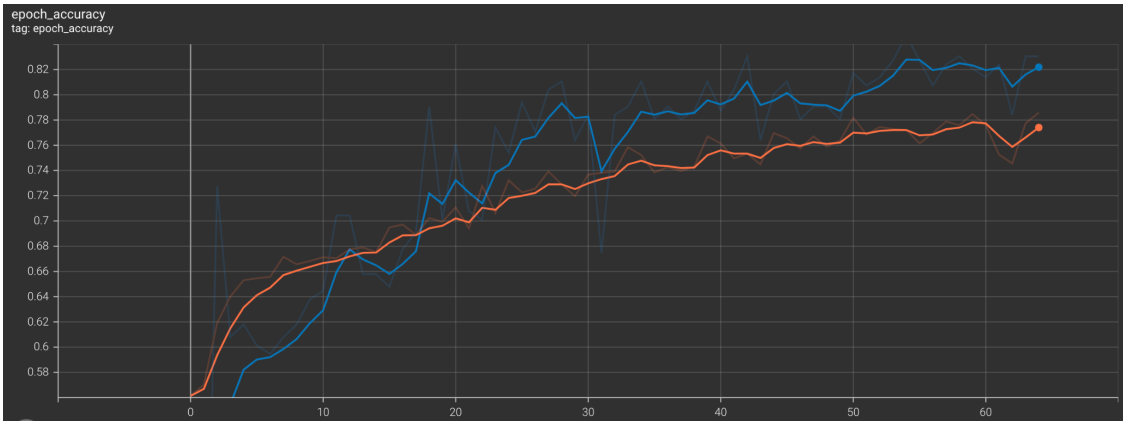

Рис. 3.10 – accuracy під час навчання моделі base_vit_12

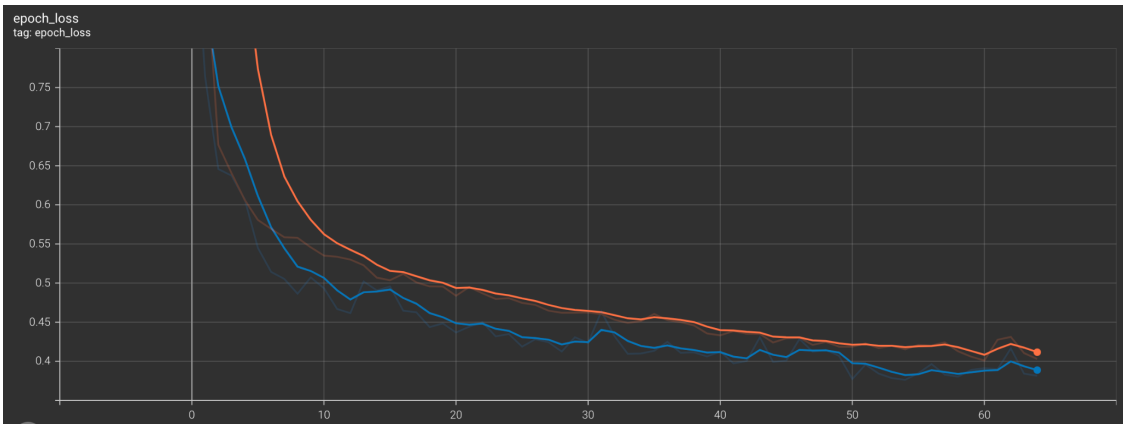

Рис 3.11 – loss під час навчання моделі base_vit_12

Таблиця 3.2 – Метрики класифікації моделі loss base_vit_12.

	precision	**recall**	**f1-score**	**Екземпляри**
Невус	0.79	0.86	0.83	202
Меланома	0.76	0.66	0.70	132

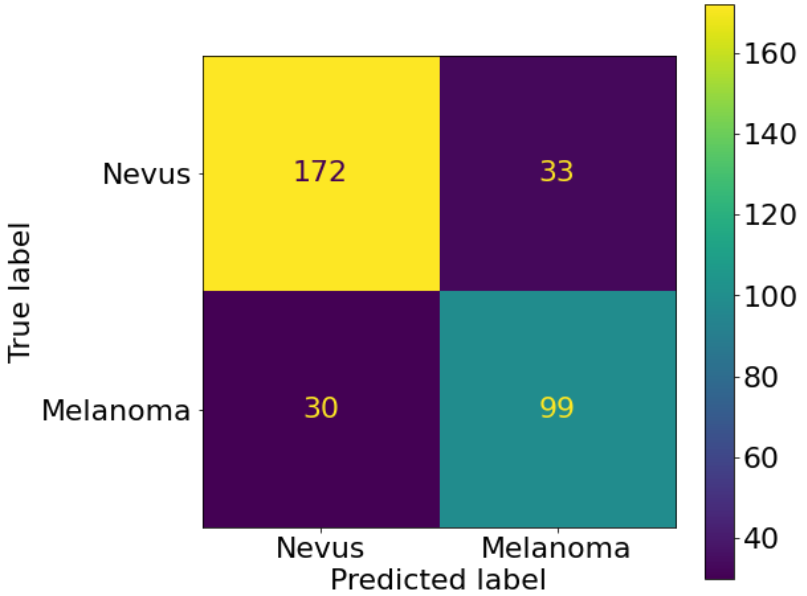

Рис. 3.12 – Матриця невідповідностей моделі merge_vit_12

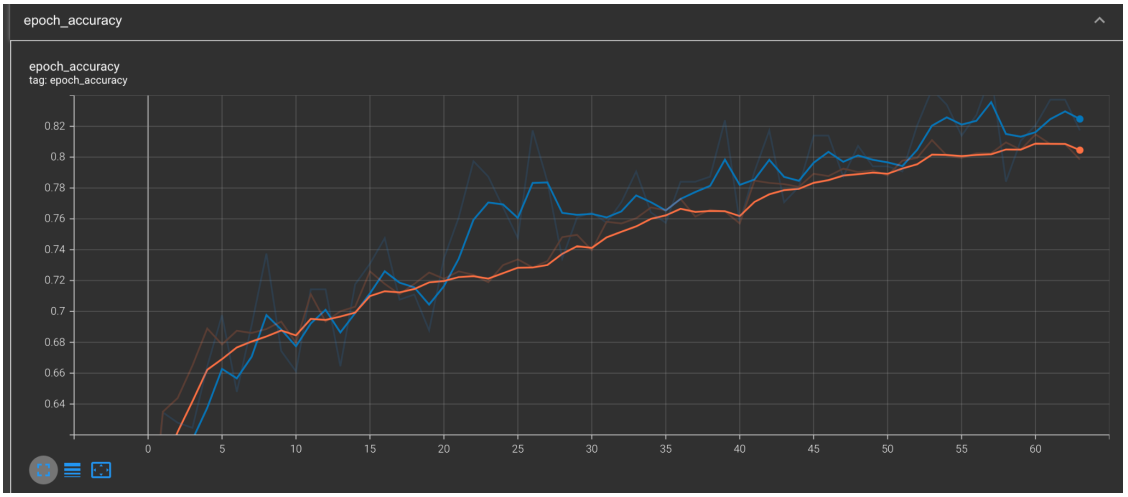

Рис. 3.13 – accuracy під час навчання моделі merge_vit_12

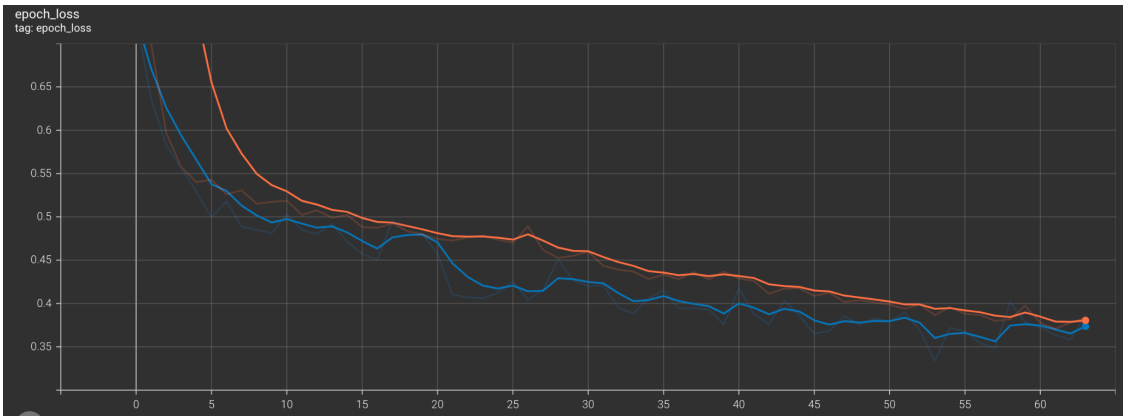

Рис. 3.14 – loss під час навчання моделі merge_vit_12

Таблиця 3.3 – Метрик класифікації моделі loss merge_vit_12.

	precision	**recall**	**f1-score**	**Екземпляри**
Невус	0.84	.85	0.85	202
Меланома	0.77	0.75	0.76	132

3.7 Порівняння зі згортковою мережею

Для порівняння була взята архітектура з існуючого дослідження по темі класифікації раку шкіри [62]. Автори зорового трансформера стверджують, що на великих наборах даних він перевершує згорткові моделі. У порівнянні пропонується застосовувати навчання для підмножинах загального тренувального набору даних розміром 100, 200 500, 1500 (весь набір - 3005), щоб оцінити залежність ключових метрик кожної з моделей від розміру набору даних

В згорткову нейронну мережу входили 4 2d згорткових шари розмірами 32, 64, 128, 256, нормалізація, шари dense з функцією активації relu, загалом 19 прихованих шарів.

Таблиця 3.4 – Порівняння згорткової моделі та зорових трансформерів для наборі даних різних розмірів

Параметри		**Метрики**		
Модель	*розмір набору даних*	*accuracy, %*	*помилки*	
			1 роду	*2 роду*
cnn	100	60.48	132	0
cnn	200	60.48	132	0
cnn	500	45.51	80	102
cnn	1500	64.37	88	31
cnn	3005	68.26	90	16
base_vit_12	100	60.48	132	0
base_vit_12	200	60.48	132	0
base_vit_12	500	60.48	132	0
base_vit_12	1500	68.34	73	33
base_vit_12	3005	78.14	45	28
merge_vit_12	100	60.48	132	0
merge_vit_12	200	60.48	132	0
merge_vit_12	500	68.34	93	13
merge_vit_12	1500	76.35	32	47
merge_vit_12	3005	81.14	33	30

Як видно з таблиці 3.4 модель cnn перевершила модель base_vit_12 для наборів невеликих розмірів, але merge_vit_12 показала найкращі результати, особливо зі збільшенням набору даних.

Загалом, можна зробити висновок, що якість зорових трансформерів зростає швидше з розміром набору даних, отже потенційно для великих даних згорткові та трансформерні моделі будуть мати значну відмінність у ефективності на користь останніх.

3.8 Розроблений додаток

Був розроблений веб додаток на основі моделі merge_vit_12. Він включає в себе бек-енд API на яке надсилається зображення шкірного утворення та фронт-енд частини, що дозволяє користувачу відправити форму із зображеннм на API та отримати відповідь від моделі base_vit_12. У випадку, якщо модель вважає утворення раковим, з'являєтсья напис "See doctor asap" тобто "Слід якомога скоріше відвідати лікаря".

Додатково для збільшення практичності застосунку можливо додати просте відстеження країв утворення в цілях користувацького моніторингу стану невусу.

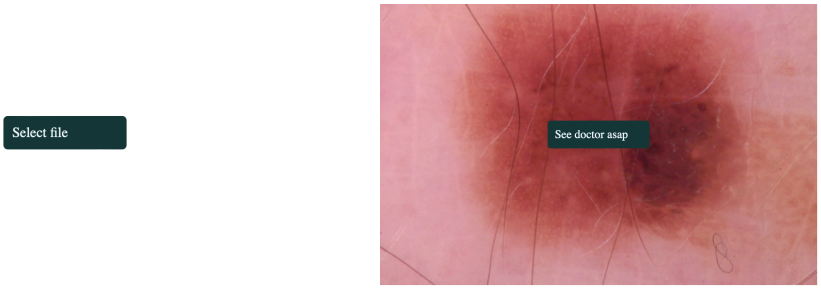

Рис. 3.15. – Приклад роботи додатка з класифікації раку шкіри на основі зорового трансформера

3.9 Висновки

В розділі було описано три можливі архітектури зорового трансформеру. Також, було наведено опис метрик, визначення які з них є більш важливими для вибору більш ефективної моделі. результати експериментів показали, що

моделі, що мають найменшу розмірність та відносно невелику кількість патчів досить добре справляються із задачею.

Було обрано найбільш ефективні моделі. Ними виявились найменші моделі двох запропонованих в цій роботі архітектур зорового трансформера, одна з яких використовує злиття патчів різних розмірів.

Після проведення порівняння із згортковою мережею було виявлено, що моделі трансформери показують більшу залежність від розміру набору даних, через що можна припустити значну їх перевагу над згортковими моделями при подальшому збільшенні набору даних.

Також, на основі однієї з моделей було розроблено веб додаток, що дозволяє користувачу перевірити свої шкірні утворення на злоякісність. Такий додаток дозволяє провести первинну діагностику будь кому з будь якої точки світу не виходячи з дому, що, теоретично, могло б збільшити відсоток ранньо діагностованих випадків раку шкіри.

Було показано що зоровий трансформер є ефективною моделлю в задачах класифікації зображень за ознакою раку шкіри. При збільшенні набору даних та меншої залежності від апаратних обмежень, щоб збільшити кількість патчів та шарів уваги, результати могли би бути навіть кращі ніж наведені в цій роботі.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Рання діагностика це чи не найголовніша складова успішного лікування раку шкіри. З даних наведених в першому розділі цієї роботи випливає, що абсолютна більшість пацієнтів, що були діагностовані при перших проявах цієї хвороби, були вилікувані. Більш того, видалення пухлини навіть не залишилося рубців, тобто рак в житті людини минувся цілком безболісно і безнаслідково, що є великою вдачею.

Додаток, що допоміг би людям раніше виявити можливі проблеми пов’язані з потенційно раковими шкірними утвореннями, міг би значно збільшити відсоток ранньо діагностованих випадків раку шкіри і, як наслідок, значно покращити шанси на успішне лікування для цих пацієнтів.

Крім того, такий додаток міг би спростити життя людям, що страждають від іпохондричних розладів. Як відомо, такі люди часто витрачають значні суми грошей для діагностики захворювань, яких вони не мають. Додаток, що дозволив би їм проводити таку діагностику не виходячи з дому, міг би дещо заспокоїти таких людей, зменшити кількість їх відвідувань до лікарні та залишити для них більше часу для особистого життя.

Застосунок для діагностики раку шкіри за фотографією може допомогти багатьом людям зекономити значні суми грошей та час на лікування хвороби в пізніх стадіях, в деяких випадках, можливо, навіть врятувавши життя.

4.1 Опис ідеї стартапу

Ідея стартапу полягає у створення додатку, що дозволяв би класифікувати зображенням пухлин на шкірі як ракових чи доброякісних. Також, додаток має мати функціонал з відстеження зміни країв родимки в часі, бо це також може свідчити про злоякісність утворення.

Таблиця 4.1 – Інформаційна карта проекту.

Назва проекту	Skin Cancer Detector
Автор проекту	Нікітін Владислав
Анотація	Застосунок для класифікації шкірної пухлини за ознакою злоякісності
Термін реалізації	8 місяців
Ресурси	Обчислювальна техніка з потужними графічними процесорами, персональний комп’ютер для розробки, середа розробки
Мета	Зайняти нішу hands-on класифікаторів раку шкіри
Очікування	Збільшення кількості ранньо діагностованих випадків раку шкіри

4.2 Технологічний аудит ідеї проекту

На цьому етапі було проведено технічний аналіз запланованого застосунку. Такий підрозділ потрібен для розуміння перспектив реалізації

проекту, можливості його розробки у визначений термін та відповідності необхідним вимогам поставленим у минулому підрозділі.

Також в цьому підрозділі буде проведений порівняльний аналіз конкурентів проекту

Таблиця 4.2. – Технологічна здійсненність продукту

Ідея	Технологія реалізації	Простота реалізації
Застосунок для класифікації раку шкіри	Мова програмування Python	Відносно проста
	Мова програмування Java	Відносно складна
	Мова програмування Prolog	Неможлива

Після адутиту мов програмування є очевидним, що найкращим можливим вибором мови програмування для реалізації проекту є мова Python.

Далі проведемо порівняльний аналіз конкурентів на теперішньому ринку.

Таблиця 4.3 – Порівняльний аналіз потенційних конкурентів на ринку.

Проект	Ціновий діапазон	Мета проекту	Цільова аудиторія
Skin Cancer Detector (цей проект)	Наразі безкоштовний	Первинна діагностика раку шкіри	Потенційні пацієнти
Computer Skin Cancer Diagnosis System	від 2500$ доларів на рік	Додатковий інструмент діагностики для професіоналів	Професійні лікарі
Cancer Detection	1300$ на рік	Дослідження доброякісних та злоякісних пухлин різних типів та тканин	Наукова

Як видно з порівняльного аналізу ніша первинної діагностики ще не є зайнятою на ринку, що дозволяє стартапу досить швидку захопити потенційну цільову аудиторію та набрати користувачів.

4.3 Аналіз ринкових можливостей продукту

В цьому підрозділі проведений аналіз потенційного ринку продукту. Це необхідно для підготовки продукту до максимально ефективного виходу на користувацький ринок, щоб якомога скоріше зайняти свою нішу та набрати максимальну кількість користувачів.

Вдалий старий старт на ринку значно підвищить шанси продукту на успішну монетизацію за допомогою реклами чи переводу додатку в категорію платних.

А таблиці 4.4 (нижче) наведений аналіз ситуації на ринку

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн/ум.од	приблизно 976000
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Враховуючи, що діагнози додатку носять рекомендаційний характер – немає
6	Середня норма рентабельності в галузі, %	58%

З таблиці 4.4 видно, що запропонований застосунок має великий потенціал на ринку. В наступній таблиці (4.5) сформовано портрети потенційної цільової аудиторії проекту.

При правильній подачі продукту до потенційної цільової аудиторії шанси швидко привабити користувачів значно збільшується. Саме тому на цьому етапі важливо правильно визначити цільову аудиторію.

Таблиця 4.5. – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Первинна діагностика раку шкіри	Люди, що мають підозри до родинок на шкірі	Хочуть знати чи варто звернутися до лікаря	Доступність діагностики
Можливість проводити більше сеансів діагностики раку шкіри	Люди, що не мають часу для відвідин лікаря	Хочуть дізнатися чи родинка може буде раковою, що не відвідувати лікаря зайвий раз	Швидкість діагностики

За таблиці 4.5 ми можемо побачити, що додаток, що розробляється цілком задовольняє потреби обох категорій потенційної цільової аудиторії та легко займе свою нішу на ринку.

В наступних таблицях визначено фактори загроз та можливостей для проекту. Це етап важливий для вдалого старту проекту, щоб розуміти модель поведінки на ринку в перший час після виходу.

Таблиця 4.6. – Фактори загроз для продукту.

/ п	Фактор	Зміст загрози	Можлива реакція компанії
	Конкуренція	Вихід на ринок більш привабливих продуктів для цільової аудиторії	Попередній аналіз продуктів, що скоро мають з’явитися на ринку, розуміння проблем та переваг свого продукту
	Якість діагностики	Користувачам необхідний сервіс з більшим/новим функціоналом.	Постійна реакція на відгуки користувачів та покращення якості моделі, робота з лікарями та експертами в предметній області

Таблиця 4.7. – Фактори можливостей для продукту.

п/п	Фактор	Зміст можливості	Можлива реакція компанії
	Популяризація додатку серед професійної спільноти	Додаток популяризується через лекторів, мед блогерів, тощо.	Реклама продукту серед професійної спільноти
	Збільшення функціонал у додатку	Додаток має достатньо широкий функціонал, щоб бути популярним незалежно від своїх первинних функцій	Уважне стеження за потребами аудиторії додатку та відповідна реакція, інтеграція з професійними медичними додатками (Apple Health тощо)

Наступним в цьому підрозділі було проведено ступеневий аналіз конкуренції.

Таблиця 4.8. – Ступеневий аналіз конкуренції на ринку.

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції- чиста	Існують інші засоби діагностики	Провести рекламу з акцентом на продукті як на засобі саме первинної діагностики та простоті діагностики
2. За рівнем конкурентної боротьби - міжнародний	Потенційні іноземні конкуренти	Розробити мультинаціональний інтерфейс
3. Конкуренція за видами товарів: - товарно-видова	Конкуренція з іншими подібними продуктами	Наголосити на унікальності даного продукту
4. За характером конкурентних переваг - нецінова	Більш якісна діагностика інших ресурсів	Зробити акцент на первинності та швидкості діагностики, можливо, порівняти з експрес-текстом
5. За інтенсивністю - не марочна	Бренд відсутній	Участь у заходах медичної спільноти

Далі був проведений аналіз конкуренції за моделлю Портера

Таблиця 4.9. – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товарозамінники
	Інші гравці ринку	Можливі конкуренти з інших систем діагностики	Продукт не має постачальників	Контроль якості моделі	Якість моделі
Висновки	Наразі на ринку немає аналогічного продукту	Інші системи діагностики не підходять для цільової аудиторії продукту	Це є загрозою	Необхідно регулярно проводити консультації з експертами в предметній області	Необхідно регулярно проводити оцінку якості по відгуках користувачів

Тож з наведених вище таблиць можливість продукту для виходу на ринок є очевидною.

Надалі проведено обгрунтування факторів конкурентоспроможності продукту.

Табл. 4.10. – Обґрунтування факторів конкурентоспроможності.

/п	Фактор конкурентоспроможності	Обґрунтування
	Ціна	Продукт є наразі безкоштовним
	Якість	Для первинної діагностики якість є достатньою

Далі було проведено сильних та слабких сторін продукту.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні						
			–3	–2	–1	0	+1	+2	+3
1	Ціна	20					+		+
2	Якість	10						+	

Ще однією необхідною складовою первинного аналізу продукту є SWOT аналіз наведений у таблиці нижче.

Таблиця 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони простота якість швидкість доступність	Слабкі сторони складна предметна область слабка геймифікація процесу використання
Можливості покращення якості моделі швидке захоплення цільової аудиторії	Загрози невдалий вихід на ринок недостатня поширеність додатку серед професійної спільноти

Після SWOT аналізу необхідно було провести альтернативну ринкову поведінку для інтеграції стартап проекту в сьогоднішній ринок.

Таблиця 4.13. – Альтернативи ринкового впровадження стартап-проекту

/п	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Просування тільки серед професійної спільноти	43%	4 місяців
2	Збільшення часу на розробку	55%	12 місяців
3	Замовлення рекламу у професійних виданнях	70%	3 місяці

На основі аналізу альтернативних шляхів ринкового впровадження було обрано альтернативу 3.

4.4 Розробка ринкової стратегії продукту

Проведемо попередній аналіз цільової аудиторії

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Індивідуальні користувачі	Висока	Високий	Сильна	Просто
2	Лікарі	Низька	Низький	Сильна	Складно
3	Юридичні особи	Низька	Низький	Слабка	Складно
Які цільові групи обрано:1					

Таблиця 4.15 – Визначення базової стратегії розвитку

п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
	3	Диференційованого маркетингу	Швидкодія, якість продукту	Концентрований маркетинг

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку.

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Так.	Компанія буде шукати нових споживачів	Буде вдосконалюватися	Бути кращим

Таблиця 4.17 – Концентрований маркетинг

/п	Вимоги до товару цільової аудиторії	Базова стратегія розвит	Ключові конкурентоспр ом ожні позиц власного стартаппроект	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
	Точна та швидка діагностика	Концентрований маркетинг	Продукт є єдиним на ринку наразі	Економія часу; Зручність застосування;

2.1 Розроблення маркетингової програми стартап-проекту

Після проведення дуже детального аналізу всіх сторін проекту, ринку, цільової аудиторії тощо, було розроблено маркетингову програму продукту-стартапу.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Швидкість отримання результату	Модель працює лічені секунди	Покращувати швидкість
2	Доступність	Додаток може застосований з будь якого пристрою	Збільшувати зручність інтерфейсу
3	Точність результату	Результати роботи моделі дуже значні для моделі первинної діагностики	Результат відповідає необхідним показникам якості

Таблиця 4.19 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламно о повідомлення	Ключові позиції, обрані для позиціонування
1	Пошук первинної діагностики для отримання рекомендаційного результату	Телеграм канали, ТБ	Точність і швидкість	Донести, що система є якісною	Таргетова на реклама
2	Пошук первинної діагностики для уникнення зайвого візиту до лікаря	Спеціалізовані канали комунікації	Швидкість і доступність	Донести, що система працює з будь якого пристрою	Реклама через експертів в області

4.5 Висновки до розділу

В розділі був проведений детальний технічний, ринковий, маркетинговий аналіз продукту.

З технічної точки зору стало зрозуміло, що продукт можливо якісно реалізувати в запропоновані строки. З ринкової, що застосунок може досить просто зайняти нішу на ринку та набрати цільову аудиторію. Маркетинговий

аналіз визначив ключові техніки і стратегії для просування додатку на цільову аудиторію та наявність цільової аудиторії як такої.

Застосунок має всі шанси стати достатньо популярним та успішно монетизуватися на сучасному ринку України та світу.

ВИСНОВКИ

В першому розділі цієї роботи був проведений огляд предметної області з якою довелося працювати, тобто раку, його природи, його видів, зокрема раку шкіри. Рак – хвороба, що відзначається нетиповістю та неоднорідністю утворень. Рак шкіри може бути як вилікуваним безнаслідково і безболісно, так і виявитись летальним. Ключовою компонентою, що відрізняє ці два можливі випадки є рання діагностика. Саме тому дослідження і створення засобів первинної діагностики цієї хвороби є актуальним.

Було проведено огляд існуючих досліджень з комп'ютерної діагностики раку шкіри та зроблено висновок, що недоліком запропонованих систем є ресурсоємність та відносно низька швидкість.

В другому розділі був проведений огляд механізму роботи моделі зорового трансформеру та його особливостей. Було згадано основні компоненти моделі та їх функції. Також були наведені описи наборів даних, які використовувались для навчання моделі в цій роботі. З цього було зроблено висновок про ефективність зорового трансформера для вирішення задач класифікації та сенс його використання для задачі класифікації раку шкіри.

Практичний розділ має в собі опис інструментів, що були використані при розробці моделі, формування набору даних, попередній обробці даних. Було наведено три архітектури зорових трансформерів (одна з яких є унікальною так як запропонована вперше в цій роботі), що використовувалися в цій роботі та результати їх навчання з різними параметрами, загалом 12 моделей. Для порівняння зі згортковою мережею було обрано 2 з 12 моделей, обидві з яких показали кращу динаміку залежності результатів навчання від розміру набору даних. Тобто зі збільшенням набору даних метрики зорових трансформерів покращувались більше ніж ті ж самі метрики згорткової мережі — при поступовому збільшенні розміру датасету із 100 до 3000, моделі

починали з одного показника accuracy при 100 зображеннях, а закінчили з приблизно 10% різниці на користь зорових трансформерів. Це також підтверджує аналогічні висновки розробників моделі зорового трансформера. Таким чином був зроблений висновок про ефективність застосування згорткової мережі для класифікації раку шкіри.

Крім того, на основі кращої з моделей був розроблений користувацький додаток, що дозволяє класифікувати шкірне утворення по фото як ракове чи здорове.

В четвертому розділі був проведений детальний аналіз додатку як стартапу, описана ідея стартапу, проаналізовані ринок, цільова аудиторія, конкуренти, обрана маркетингова стратегія. Був зроблений висновок про великий потенціал застосунку на сучасному ринку.

В роботі були досягнуті всі цілі та задачі, а саме, проведено огляд літератури за темою, проаналізовано зоровий трансформер як ефективний класифікатор раку шкіри, створено застосунок на основі авторської моделі зорового трансформера, що дозволяє результативно класифікувати користувацькі зображення.

ДЖЕРЕЛА

1. Skin Cancer Information. The Skin Cancer Foundation. [Електронний ресурс] – Режим доступу до ресурсу: https://www.skincancer.org/skin-cancer-information/ Дата доступу – 10.12.2022

2. РАК В УКРАЇНІ, Національний канцер-реєстр України (НКРУ),

Захворюваність, смертність, показники діяльності онкологічної служби [Електронний ресурс] Режим доступу до ресурсу: http://www.ncru.inf.ua/publications/BULL_21/PDF/mel.pdf Дата доступу – 10.12.2022

3. What Is Cancer?. National Cancer Institute. [Електронний ресурс] Режим доступу: https://www.cancer.gov/about-cancer/understanding/what-is-cancer (дата звернення: 10.12.2022).

4. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin. Attention is all you need. [Електронний ресурс] – Режим доступу до ресурсу:https://arxiv.org/pdf/1706.03762.pdf Дата доступу – 10.12.2022

5. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby. AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE [Електронний ресурс] – Режим доступу до ресурсу:https://arxiv.org/pdf/2010.11929.pdf Дата доступу – 10.12.2022

6. [Електронний ресурс] – Режим доступу до ресурсу:https://challenge2020.isic-archive.com/ Дата доступу - 10.12.2022

7. [Електронний ресурс] – Режим доступу до ресурсу:https://keras.io/examples/vision/image_classification_with_vision_transformer/ Дата доступу – 10.12.2022

8. [Електронний ресурс] – Режим доступу до ресурсу:https://www.aad.org/media/stats-skin-cancer#:~:text=The%20five%2Dyear%20survival%20rate,the%20lymph%20nodes%20is%2099%25.&text=The%20five%2Dyear%20survival%20rate%20for%20melanoma%20that%20spreads%20to,and%20other%20organs%20is%2030%25 Дата доступу – 10.12.2022

9. [Електронний ресурс] – Режим доступу до ресурсу:https://www.mayoclinic.org/diseases-conditions/skin-cancer/symptoms-causes/syc-20377605#:~:text=Skin%20cancer%20%E2%80%94%20the%20abnormal%20growth,squamous%20cell%20carcinoma%20and%20melanoma Дата доступу - 10.12.2022

10. [Електронний ресурс] – Режим доступу до ресурсу:https://www.wcrf.org/cancer-trends/skin-cancer-statistics/ Дата доступу – 10.12.2022

11. [Електронний ресурс] – Режим доступу до ресурсу:https://www.skincancer.org/early-detection/#:~:text=Early%20detection%20saves%20lives.,become%20dangerous%2C%20disfiguring%20or%20deadly Дата доступу – 10.12.2022

12. [Електронний ресурс] – Режим доступу до ресурсу:https://link.springer.com/article/10.1007/s11042-022-13081-x Дата доступу – 10.12.2022

13. Kazuhisa Matsunaga, Akira Hamada, Akane Minagawa, Hiroshi Koga. Image Classification of Melanoma, Nevus and Seborrheic Keratosis by Deep Neural Network Ensemble [Електронний ресурс] – Режим доступу до ресурсу:https://arxiv.org/pdf/1703.03108.pdf Дата доступу - 10.12.2022

14. [Електронний ресурс] – Режим доступу до ресурсу:http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf Дата доступу – 10.12.2022

15. [Електронний ресурс] – Режим доступу до ресурсу:https://www.hindawi.com/journals/jhe/2022/6952304/ Дата доступу – 10.12.2022

16. [Електронний ресурс] – Режим доступу до ресурсу:https://patentimages.storage.googleapis.com/98/90/ef/7bd575ca689fe2/US10516865.pdf Дата доступу – 10.12.2022

17. [Електронний ресурс] – Режим доступу до ресурсу:https://www.researchgate.net/publication/356670829_Performance_Analysis_of_Adaptive_Unsharp_Masking_Filter_Techniques_for_Image_Contrast_Enhancement Дата доступу – 10.12.2022

18. Serra J. Image Analysis and Mathematical Morphology. 3rd ed. Academic Pr, 1982. 610 p.

19. Pelleg, Dan; Moore, Andrew (1999). "Accelerating exact k -means algorithms with geometric reasoning". Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '99. San Diego, California, United States: ACM Press: 277–281.

20. J. F. Alcon, C. Ciuhu, W. Ten Kate et al., “Automatic imaging system with decision support for inspection of pigmented skin lesions and melanoma diagnosis,” IEEE Journal of Selected Topics in Signal Processing, vol. 3, no. 1, pp. 14–25, 2009.

21. A. A Heidari, S. Mirjalili, H. Faris et al., “Harris hawks optimization: algorithm and applications,” Future Generation Computer Systems, vol. 97, pp. 849–872, 2019.

22. e, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE. pp. 770–778. arXiv:1512.03385. doi:10.1109/CVPR.2016.90. ISBN 978-1-4673-8851-1.

23. Ali AR, Li J, Yang G, O’Shea SJ. A Machine Learning Approach to Automatic Detection of Irregularity in Skin Lesion Border Using Dermoscopic Images. PeerJ Comput Sci (2020) 6:e268. doi: 10.7717/peerj-cs.268

24. Masood A, Al-Jumaily A, Anam K. Self-Supervised Learning Model for Skin Cancer Diagnosis, in: 7th International IEEE/EMBS Conference on Neural Engineering (NER), Manhattan, New York, U.S.:Institute of Electrical and Electronics Engineers (IEEE) (2015). 1012–5 pp. doi: 10.1109/NER.2015.7146798

25. Nasr-Esfahani E, Samavi S, Karimi N, Soroushmehr S, Jafari M, Ward K, et al. Melanoma Detection by Analysis of Clinical Images Using Convolutional Neural Network, in: 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Manhattan, New York, U.S: Institute of Electrical and Electronics Engineers (IEEE) (2016). 1373–6 pp. doi: 10.1109/EMBC.2016.7590963

26. Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR (2014) abs/1409.1556:arXiv:1409.1556. doi: 10.48550/arXiv.1409.1556

27. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going Deeper With Convolutions. Proc IEEE Conf Comput Vision Pattern Recognit (2015), 1–9. doi: 10.1109/CVPR.2015.7298594

28. He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. Proc IEEE Conf Comput Vision Pattern Recognit (2016), 770–8. doi: 10.1109/CVPR.2016.90

29. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, et al. Dermatologist-Level Classification of Skin Cancer With Deep Neural Networks. Nature (2017) 542:115–8. doi: 10.1038/nature21056

30. Lopez AR, Giro-i Nieto X, Burdick J, Marques O. Skin Lesion Classification From Dermoscopic Images Using Deep Learning Techniques, in: 13th IASTED International Conference on Biomedical Engineering (BioMed), Manhattan, New York, U.S: Institute of Electrical and Electronics Engineers (IEEE) (2017). pp. 49–54.

31. Jaworek-Korjakowska J, Kleczek P, Gorgon M. Melanoma Thickness Prediction Based on Convolutional Neural Network With Vgg-19 Model Transfer

Learning. Proc IEEE/CVF Conf Comput Vision Pattern Recognit Workshops (2019), 0–0. doi: 10.1109/CVPRW.2019.00333

32. Kawahara J, Daneshvar S, Argenziano G, Hamarneh G. Seven-Point Checklist and Skin Lesion Classification Using Multitask Multimodal Neural Nets. IEEE J Biomed Health Inf (2019) 23:538–46. doi: 10.1109/JBHI.2018.2824327

33. Walker B, Rehg J, Kalra A, Winters R, Drews P, Dascalu J, et al. Dermoscopy Diagnosis of Cancerous Lesions Utilizing Dual Deep Learning Algorithms via Visual and Audio (Sonification) Outputs: Laboratory and Prospective Observational Studies. EBioMedicine (2019) 40:176–83. doi: 10.1016/j.ebiom.2019.01.028

34. Mishra S, Imaizumi H, Yamasaki T. Interpreting Fine-Grained Dermatological Classification by Deep Learning. Proc IEEE/CVF Conf Comput Vision Pattern Recognit Workshops (2019), 0–0. doi: 10.1109/CVPRW.2019.00331

35. Perez F, Avila S, Valle E. Solo or Ensemble? Choosing a Cnn Architecture for Melanoma Classification. Proc IEEE/CVF Conf Comput Vision Pattern Recognit Workshops (2019), 0–0. doi: 10.1109/CVPRW.2019.00336

36. Polat K, Koc KO. Detection of Skin Diseases From Dermoscopy Image Using the Combination of Convolutional Neural Network and One-Versus-All. J Artif Intell Syst (2020) 2:80–97. doi: 10.33969/AIS.2020.21006

37. Rahman Z, Hossain MS, Islam MR, Hasan MM, Hridhee RA. An Approach for Multiclass Skin Lesion Classification Based on Ensemble Learning. Inf Med Unlocked (2021) 25:100659. doi: 10.1016/j.imu.2021.100659

38. [Електронний ресурс] – Режим доступу до ресурсу:https://www.fc.up.pt/addi/ph2%20database.html Дата доступу – 10.12.2022

39. Mendonça T, Ferreira PM, Marques JS, Marcal AR, Rozeira J. Ph 2-a Dermoscopic Image Database for Research and Benchmarking, in: 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society

(EMBC). Manhattan, New York, U.S: Institute of Electrical and Electronics Engineers (IEEE) (2013). pp. 5437–40.

40. Brahmbhatt P, Rajan SN. Skin Lesion Segmentation Using Segnet With Binary Cross-Entropy. In: International Conference on Artificial Intelligence and Speech Technology (Aist2019), 15th, vol. 14. Delhi, India: Excel India Publishers (2019).

41. Basak H, Kundu R, Sarkar R. Mfsnet: A Multi Focus Segmentation Network for Skin Lesion Segmentation. Pattern Recognit (2022) 128:108673. doi: 10.1016/j.patcog.2022.108673.

42. [Електронний ресурс] – Режим доступу до ресурсу:https://www.cs.rug.nl/~imaging/databases/melanoma_naevi/ Дата доступу – 10.12.2022

43. Matsunaga K, Hamada A, Minagawa A, Koga H. Image Classification of Melanoma, Nevus and Seborrheic Keratosis by Deep Neural Network Ensemble. ArXiv (2017) abs/1703.03108:arXiv:1703.03108. doi: 10.48550/arXiv.1703.03108.

44. West J, Ventura D, Warnick S. Spring Research Presentation: A Theoretical Foundation for Inductive Transfer. Brigham Young University: College of Physical and Mathematical Sciences (2007).

45. Giotis I, Molders N, Land S, Biehl M, Jonkman MF, Petkov N. Med-Node: A Computer-Assisted Melanoma Diagnosis System Using non-Dermoscopic Images. Expert Syst Appl (2015) 42:6578–85. doi: 10.1016/j.eswa.2015.04.034

46. Електронний ресурс] – Режим доступу до ресурсу:https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T Дата доступу – 10.12.2022

47. Rezvantalab A, Safigholi H, Karimijeshni S. Dermatologist Level Dermoscopy Skin Cancer Classification Using Different Deep Learning Convolutional Neural Networks Algorithms. ArXiv (2018) abs/1810.10348:arXiv:1810.10348. doi: 10.48550/arXiv.1810.10348.

48. Young K, Booth G, Simpson B, Dutton R, Shrapnel S. Deep Neural Network or Dermatologist? In: Interpretability of Machine Intelligence in Medical Image Computing and Multimodal Learning for Clinical Decision Support. New York, NY, United States:Springer (2019). p. 48–55.

49. Kawahara J, Daneshvar S, Argenziano G, Hamarneh G. Seven-Point Checklist and Skin Lesion Classification Using Multitask Multimodal Neural Nets. IEEE J Biomed Health Inf (2018) 23:538–46. doi: 10.1109/JBHI.2018.2824327.

50. Coppola D, Lee HK, Guan C. Interpreting Mechanisms of Prediction for Skin Cancer Diagnosis Using Multi-Task Learning. Proc IEEE/CVF Conf Comput Vision Pattern Recognit Workshops (2020), 734–5. doi: 10.1109/CVPRW50498.2020.00375.

51. Yao P, Shen S, Xu M, Liu P, Zhang F, Xing J, et al. Single Model Deep Learning on Imbalanced Small Datasets for Skin Lesion Classification. IEEE Trans Med Imaging (2021) 41:1242–54. doi: 10.1109/TMI.2021.3136682.

52. [Електронний ресурс] – Режим доступу до ресурсу:https://www.isic-archive.com/ Дата доступу – 10.12.2022

53. Combalia M, Codella NC, Rotemberg V, Helba B, Vilaplana V, Reiter O, et al. Bcn20000: Dermoscopic Lesions in the Wild. ArXiv (2019) abs/1908.02288:arXiv:1908.02288. doi: 10.48550/arXiv.1908.02288

54. Mou Y, Welten S, Yediel YU, Kirsten T, Beyan OD. Distributed Learning for Melanoma Classification Using Personal Health Train. ArXiv (2021) abs/2103.13226:arXiv:2103.13226. doi: 10.48550/arXiv.2103.13226

55. Maron RC, Schlager JG, Haggenmüller S, von Kalle C, Utikal JS, Meier F, et al. A Benchmark for Neural Network Robustness in Skin Cancer Classification. Eur J Cancer (2021) 155:191–9. doi: 10.1016/j.ejca.2021.06.047

56. Gessert N, Nielsen M, Shaikh M, Werner R, Schlaefer A. Skin Lesion Classification Using Ensembles of Multi-Resolution Efficientnets With Meta Data. MethodsX (2020) 7:100864. doi: 10.1016/j.mex.2020.100864.

57. Gutman D, Codella NC, Celebi E, Helba B, Marchetti M, Mishra N, et al. Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the International Symposium on Biomedical Imaging (Isbi) 2016, Hosted by the International Skin Imaging Collaboration (Isic). IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018) (2016) abs/1605.01397:arXiv:1605.01397. doi: 10.1109/ISBI.2018.8363547

58. Cassidy B, Kendrick C, Brodzicki A, Jaworek-Korjakowska J, Yap MH. Analysis of the Isic Image Datasets: Usage, Benchmarks and Recommendations. Med Image Anal (2022) 75:102305. doi: 10.1016/j.media.2021.102305.

59. Lopez AR, Giro-i Nieto X, Burdick J, Marques O. Skin Lesion Classification From Dermoscopic Images Using Deep Learning Techniques, in: 13th IASTED International Conference on Biomedical Engineering (BioMed), Manhattan, New York, U.S: Institute of Electrical and Electronics Engineers (IEEE) (2017). pp. 49–54.

60. Bevan PJ, Atapour-Abarghouei A. Skin Deep Unlearning: Artefact and Instrument Debiasing in the Context of Melanoma Classification. ArXiv (2021) abs/2109.09818:arXiv:2109.09818. doi: 10.48550/arXiv.2109.09818.

61. Chun-Fu (Richard) Chen, Quanfu Fan, Rameswar Panda MIT-IBM Watson AI Lab. CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification [Електронний ресурс] – Режим доступу до ресурсу:https://arxiv.org/pdf/2103.14899.pdf Дата доступу – 10.12.2022

62. [Електронний ресурс] – Режим доступу до ресурсу:https://github.com/Tirth27/Skin-Cancer-Classification-using-Deep-Learning Дата доступу – 10.12.2022

63. [Електронний ресурс] – Режим доступу до ресурсу:https://theaisummer.com/self-attention/ Дата доступу – 10.12.2022

64. Нікітін В. О., Шаповал Н. В. ЗОРОВИЙ ТРАНСФОРМЕР ДЛЯ ЗАДАЧІ КЛАСИФІКАЦІЇ РАКУ ШКІРИ. м. Київ, 25 листоп. 2022 р.

ДОДАТОК А. ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
from google.colab import drive
drive.mount('/content/drive')
import matplotlib.pylab as plt
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_addons as tfa
import pickle
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D
from tensorflow.keras.models import Sequential
from sklearn.metrics import plot_confusion_matrix, ConfusionMatrixDisplay, confusion_matrix
import datetime
class Patches(layers.Layer):
  def __init__(self, patch_size):
    super(Patches, self).__init__()
    self.patch_size = patch_size

  def call(self, images):
    batch_size = tf.shape(images)[0]
    patches = tf.image.extract_patches(
        images=images,
        sizes=[1, self.patch_size, self.patch_size, 1],
        strides=[1, self.patch_size, self.patch_size, 1],
        rates=[1, 1, 1, 1],
        padding="VALID",
    )
```

```
    patch_dims = patches.shape[-1]
    patches = tf.reshape(patches, [batch_size, -1, patch_dims])
    return patches

class PatchEncoder(layers.Layer):
  def __init__(self, num_patches, projection_dim):
    super(PatchEncoder, self).__init__()
    self.num_patches = num_patches
    self.projection = layers.Dense(units=projection_dim)
    self.position_embedding = layers.Embedding(
        input_dim=num_patches, output_dim=projection_dim
    )

  def call(self, patch):
    positions = tf.range(start=0, limit=self.num_patches, delta=1)
    encoded = self.projection(patch) + self.position_embedding(positions)
    return encoded

def mlp(x, hidden_units, dropout_rate):
  for units in hidden_units:
      x = layers.Dense(units, activation=tf.nn.gelu)(x)
      x = layers.Dropout(dropout_rate)(x)
  return x

image_size = 12  # We'll resize input images to this size
patch_size = 1  # Size of the patches to be extract from the input images
transformer_layers = 6
projection_dim = 12
num_heads = 3
```

```
        learning_rate = 0.001
        weight_decay = 0.0001
        batch_size = 256
        num_epochs = 100

        num_patches = (image_size // patch_size) ** 2

        transformer_units = [
            projection_dim * 2,
            projection_dim,
        ]  # Size of the transformer layers

        mlp_head_units = [2048, 1024]
        TARGET_IMAGE_SHAPE = (100, 100)
        num_classes = 2
        input_shape = (*TARGET_IMAGE_SHAPE, 3)
        PATH = "drive/MyDrive/diploma"
        MODEL_NAME =
f"model_base_vit_{image_size}_{patch_size}_{transformer_layers}_{num_heads}
_{projection_dim}"
        def create_model():
          data_augmentation = keras.Sequential(
            [
                layers.Normalization(),
                layers.Resizing(image_size, image_size),
                layers.RandomFlip("horizontal"),
                layers.RandomRotation(factor=0.02),
                layers.RandomZoom(
```

```
            height_factor=0.2, width_factor=0.2
        ),
    ],
    name="data_augmentation",
)
inputs = layers.Input(shape=input_shape)
augmented = data_augmentation(inputs)
patches = Patches(patch_size)(augmented)
encoded_patches = PatchEncoder(num_patches, projection_dim)(patches)
for _ in range(transformer_layers):
    # Layer normalization 1.
    x1 = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
    # Create a multi-head attention layer.
    attention_output = layers.MultiHeadAttention(
        num_heads=num_heads, key_dim=projection_dim, dropout=0.1
    )(x1, x1)
    # Skip connection 1.
    x2 = layers.Add()([attention_output, encoded_patches])
    # Layer normalization 2.
    x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
    # MLP.
    x3 = mlp(x3, hidden_units=transformer_units, dropout_rate=0.1)
    # Skip connection 2.
    encoded_patches = layers.Add()([x3, x2])
representation = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
representation = layers.Flatten()(representation)
representation = layers.Dropout(0.5)(representation)
# Add MLP.
```

```
        features = mlp(representation, hidden_units=mlp_head_units,
dropout_rate=0.5)
        # Classify outputs.
        logits = layers.Dense(num_classes)(features)
        # Create the Keras model.
        return keras.Model(inputs=inputs, outputs=logits)
    create_model().summary()
    def run(model, name, x_train, x_test, y_train, y_test):
        optimizer = tfa.optimizers.AdamW(
            learning_rate=learning_rate, weight_decay=weight_decay
        )

        model.compile(
            optimizer=optimizer,
            loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
            metrics=[
                keras.metrics.SparseCategoricalAccuracy(name="accuracy"),
                keras.metrics.SparseTopKCategoricalAccuracy(5,
name="top-5-accuracy"),
            ],
        )

        checkpoint_filepath = "/tmp/checkpoint"
        checkpoint_callback = keras.callbacks.ModelCheckpoint(
            checkpoint_filepath,
            monitor="val_accuracy",
            save_best_only=True,
            save_weights_only=True,
        )
```

```
        log_dir = f"logs/fit/{name}/" +
datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
        tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,
histogram_freq=1)

        es_callback = keras.callbacks.EarlyStopping(
            monitor="val_loss",
            restore_best_weights=True,
            patience=10
        )
        history = model.fit(
            x=x_train,
            y=y_train,
            batch_size=batch_size,
            epochs=num_epochs,
            validation_split=0.1,
            callbacks=[checkpoint_callback, tensorboard_callback, es_callback],
        )

        model.load_weights(checkpoint_filepath)
        _, accuracy, top_5_accuracy = model.evaluate(x_test, y_test)
        print(f"Test accuracy: {round(accuracy * 100, 2)}%")
        print(f"Test top 5 accuracy: {round(top_5_accuracy * 100, 2)}%")

    with open(f"{PATH}/data/split_data.pkl", "rb") as f:
        x_train, x_test, y_train, y_test = pickle.loads(f.read())
    def get_part_of_data(n):
        x_train_n, y_train_n = x_train[:n], y_train[:n]
```

```
    print(f"x_{n} has {y_train_n[y_train_n==1].shape[0]} cancer imgs")
    return x_train_n, y_train_n

x_train_100, y_train_100 = get_part_of_data(100)
x_train_200, y_train_200 = get_part_of_data(200)
x_train_500, y_train_500 = get_part_of_data(500)
x_train_1500, y_train_1500 = get_part_of_data(1500)
modell00 = create_model()
model200 = create_model()
model500 = create_model()
model1500 = create_model()
modelall = create_model()

def teach_model(model, name, x_train, y_train):
  run(model, name, x_train, x_test, y_train, y_test)
  with open(f"{PATH}/models/{name}.pkl", "wb") as f:
    f.write(pickle.dumps(model))
  y_probs = model.predict(x_test)
  plt.rcParams.update({'font.size': 22})
  #Convert prediction probabilities into integers
  y_preds = y_probs.argmax(axis=1)
  cm=confusion_matrix(y_preds,y_test)
  #Plot

disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=["Nevus",
"Melanoma"])
    fig, ax = plt.subplots(figsize=(8,8))
    disp.plot(ax=ax)
    print("-"*50,name,"-"*50)
```

```
        teach_model(modell00, "base_vit_modell00", x_train_100, y_train_100)
teach_model(model200, "base_vit_model200", x_train_200, y_train_200)
teach_model(model500, "base_vit_model500", x_train_500, y_train_500)
teach_model(model1500, "vit_model1500", x_train_1500, y_train_1500)
teach_model(modelall, MODEL_NAME, x_train, y_train)
# -*- coding: utf-8 -*-
"""CNN.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1BqQGvdw6LV4GOfKwv3vom-f-06c3amtO
"""

from google.colab import drive
drive.mount('/content/drive')

!pip install -U tensorflow-addons

import matplotlib.pylab as plt
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_addons as tfa
import pickle
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D
```

```
from tensorflow.keras.models import Sequential
from sklearn.metrics import plot_confusion_matrix, ConfusionMatrixDisplay,
confusion_matrix
import datetime

learning_rate = 0.001
weight_decay = 0.0001
batch_size = 256
num_epochs = 100
image_size = 72  # We'll resize input images to this size
patch_size = 6  # Size of the patches to be extract from the input images
num_patches = (image_size // patch_size) ** 2
projection_dim = 64
num_heads = 4
transformer_units = [
    projection_dim * 2,
    projection_dim,
]  # Size of the transformer layers
transformer_layers = 8
mlp_head_units = [2048, 1024]
TARGET_IMAGE_SHAPE = (100, 100)
num_classes = 2
input_shape = (*TARGET_IMAGE_SHAPE, 3)
PATH = "drive/MyDrive/diploma"

def create_model():
  data_augmentation = keras.Sequential(
    [
        layers.Normalization(),
```

```
      layers.Resizing(image_size, image_size),
      layers.RandomFlip("horizontal"),
      layers.RandomRotation(factor=0.02),
      layers.RandomZoom(
          height_factor=0.2, width_factor=0.2
      ),
  ],
  name="data_augmentation",
)
model = Sequential()
for layer in data_augmentation.layers:
  model.add(layer)
model.add(
  Conv2D(
      16,
      kernel_size=(3, 3),
      input_shape=(100, 100, 3),
      activation="relu",
      padding="same",
  )
)
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(tf.keras.layers.BatchNormalization())
model.add(Conv2D(32, kernel_size=(3, 3), activation="relu"))
model.add(Conv2D(64, kernel_size=(3, 3), activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(tf.keras.layers.BatchNormalization())
model.add(Conv2D(128, kernel_size=(3, 3), activation="relu"))
model.add(Conv2D(256, kernel_size=(3, 3), activation="relu"))
```

```
  model.add(Flatten())
  model.add(tf.keras.layers.Dropout(0.2))
  model.add(Dense(256, activation="relu"))
  model.add(tf.keras.layers.BatchNormalization())
  model.add(tf.keras.layers.Dropout(0.2))
  model.add(Dense(128, activation="relu"))
  model.add(tf.keras.layers.BatchNormalization())
  model.add(Dense(64, activation="relu"))
  model.add(tf.keras.layers.BatchNormalization())
  model.add(tf.keras.layers.Dropout(0.2))
  model.add(Dense(32, activation="relu"))
  model.add(tf.keras.layers.BatchNormalization())
  model.add(Dense(2, activation="softmax"))
  return model

def run(model, name, x_train, x_test, y_train, y_test):
  optimizer = tfa.optimizers.AdamW(
      learning_rate=learning_rate, weight_decay=weight_decay
  )

  model.compile(
      optimizer=optimizer,
      loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
      metrics=[
          keras.metrics.SparseCategoricalAccuracy(name="accuracy"),
          keras.metrics.SparseTopKCategoricalAccuracy(5, name="top-5-accuracy"),
      ],
  )
```

```
  checkpoint_filepath = "/tmp/checkpoint"
  checkpoint_callback = keras.callbacks.ModelCheckpoint(
      checkpoint_filepath,
      monitor="val_accuracy",
      save_best_only=True,
      save_weights_only=True,
  )

  log_dir = f"logs/fit/{name}/" +
datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
  tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,
histogram_freq=1)

  es_callback = keras.callbacks.EarlyStopping(
      monitor="val_loss",
      restore_best_weights=True,
      patience=10
  )
  history = model.fit(
      x=x_train,
      y=y_train,
      batch_size=batch_size,
      epochs=num_epochs,
      validation_split=0.1,
      callbacks=[checkpoint_callback, tensorboard_callback, es_callback],
  )

  model.load_weights(checkpoint_filepath)
  _, accuracy, top_5_accuracy = model.evaluate(x_test, y_test)
```

```
  print(f"Test accuracy: {round(accuracy * 100, 2)}%")
  print(f"Test top 5 accuracy: {round(top_5_accuracy * 100, 2)}%")



with open(f"{PATH}/data/split_data.pkl", "rb") as f:
  x_train, x_test, y_train, y_test = pickle.loads(f.read())

x_train.shape

def get_part_of_data(n):
  x_train_n, y_train_n = x_train[:n], y_train[:n]
  print(f"x_{n} has {y_train_n[y_train_n==1].shape[0]} cancer imgs")
  return x_train_n, y_train_n

x_train_100, y_train_100 = get_part_of_data(100)
x_train_200, y_train_200 = get_part_of_data(200)
x_train_500, y_train_500 = get_part_of_data(500)
x_train_1500, y_train_1500 = get_part_of_data(1500)

modell00 = create_model()
model200 = create_model()
model500 = create_model()
model1500 = create_model()
modelall = create_model()

def teach_model(model, name, x_train, y_train):
  run(model, name, x_train, x_test, y_train, y_test)
  with open(f"{PATH}/models/{name}.pkl", "wb") as f:
```

```
    f.write(pickle.dumps(model))
  y_probs = model.predict(x_test)
  plt.rcParams.update({'font.size': 22})
  #Convert prediction probabilities into integers
  y_preds = y_probs.argmax(axis=1)
  cm=confusion_matrix(y_preds,y_test)
  #Plot
  disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=["Nevus",
"Melanoma"])
  fig, ax = plt.subplots(figsize=(8,8))
  disp.plot(ax=ax)
  print("-"*50,name,"-"*50)

teach_model(modell00, "cnn_modell00", x_train_100, y_train_100)

teach_model(model200, "cnn_model200", x_train_200, y_train_200)

teach_model(model500, "cnn_model500", x_train_500, y_train_500)

teach_model(model1500, "cnn_model1500", x_train_1500, y_train_1500)

teach_model(modelall, "cnn_modelall", x_train, y_train)

# Commented out IPython magic to ensure Python compatibility.
# %load_ext tensorboard

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_modell00
```

```
# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_model200

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_model500

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_model1500

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_modelall
# -*- coding: utf-8 -*-
"""CNN.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1BqQGvdw6LV4GOfKwv3vom-f-06c3amtO
"""

from google.colab import drive
drive.mount('/content/drive')

!pip install -U tensorflow-addons

import matplotlib.pylab as plt
import numpy as np
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_addons as tfa
import pickle
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D
from tensorflow.keras.models import Sequential
from sklearn.metrics import plot_confusion_matrix, ConfusionMatrixDisplay,
confusion_matrix
import datetime

learning_rate = 0.001
weight_decay = 0.0001
batch_size = 256
num_epochs = 100
image_size = 72  # We'll resize input images to this size
patch_size = 6  # Size of the patches to be extract from the input images
num_patches = (image_size // patch_size) ** 2
projection_dim = 64
num_heads = 4
transformer_units = [
    projection_dim * 2,
    projection_dim,
]  # Size of the transformer layers
transformer_layers = 8
mlp_head_units = [2048, 1024]
TARGET_IMAGE_SHAPE = (100, 100)
num_classes = 2
input_shape = (*TARGET_IMAGE_SHAPE, 3)
```

```
PATH = "drive/MyDrive/diploma"


def create_model():
  data_augmentation = keras.Sequential(
    [
        layers.Normalization(),
        layers.Resizing(image_size, image_size),
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(factor=0.02),
        layers.RandomZoom(
            height_factor=0.2, width_factor=0.2
        ),
    ],
    name="data_augmentation",
  )
  model = Sequential()
  for layer in data_augmentation.layers:
    model.add(layer)
  model.add(
    Conv2D(
        16,
        kernel_size=(3, 3),
        input_shape=(100, 100, 3),
        activation="relu",
        padding="same",
    )
  )
  model.add(MaxPool2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.BatchNormalization())
```

```
  model.add(Conv2D(32, kernel_size=(3, 3), activation="relu"))
  model.add(Conv2D(64, kernel_size=(3, 3), activation="relu"))
  model.add(MaxPool2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.BatchNormalization())
  model.add(Conv2D(128, kernel_size=(3, 3), activation="relu"))
  model.add(Conv2D(256, kernel_size=(3, 3), activation="relu"))
  model.add(Flatten())
  model.add(tf.keras.layers.Dropout(0.2))
  model.add(Dense(256, activation="relu"))
  model.add(tf.keras.layers.BatchNormalization())
  model.add(tf.keras.layers.Dropout(0.2))
  model.add(Dense(128, activation="relu"))
  model.add(tf.keras.layers.BatchNormalization())
  model.add(Dense(64, activation="relu"))
  model.add(tf.keras.layers.BatchNormalization())
  model.add(tf.keras.layers.Dropout(0.2))
  model.add(Dense(32, activation="relu"))
  model.add(tf.keras.layers.BatchNormalization())
  model.add(Dense(2, activation="softmax"))
  return model

def run(model, name, x_train, x_test, y_train, y_test):
  optimizer = tfa.optimizers.AdamW(
      learning_rate=learning_rate, weight_decay=weight_decay
  )

  model.compile(
      optimizer=optimizer,
      loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
```

```
    metrics=[
        keras.metrics.SparseCategoricalAccuracy(name="accuracy"),
        keras.metrics.SparseTopKCategoricalAccuracy(5, name="top-5-accuracy"),
    ],
)

checkpoint_filepath = "/tmp/checkpoint"
checkpoint_callback = keras.callbacks.ModelCheckpoint(
    checkpoint_filepath,
    monitor="val_accuracy",
    save_best_only=True,
    save_weights_only=True,
)

log_dir = f"logs/fit/{name}/" +
datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,
histogram_freq=1)

es_callback = keras.callbacks.EarlyStopping(
    monitor="val_loss",
    restore_best_weights=True,
    patience=10
)
history = model.fit(
    x=x_train,
    y=y_train,
    batch_size=batch_size,
    epochs=num_epochs,
```

```
    validation_split=0.1,
    callbacks=[checkpoint_callback, tensorboard_callback, es_callback],
  )

  model.load_weights(checkpoint_filepath)
  _, accuracy, top_5_accuracy = model.evaluate(x_test, y_test)
  print(f"Test accuracy: {round(accuracy * 100, 2)}%")
  print(f"Test top 5 accuracy: {round(top_5_accuracy * 100, 2)}%")



with open(f"{PATH}/data/split_data.pkl", "rb") as f:
  x_train, x_test, y_train, y_test = pickle.loads(f.read())

x_train.shape

def get_part_of_data(n):
  x_train_n, y_train_n = x_train[:n], y_train[:n]
  print(f"x_{n} has {y_train_n[y_train_n==1].shape[0]} cancer imgs")
  return x_train_n, y_train_n

x_train_100, y_train_100 = get_part_of_data(100)
x_train_200, y_train_200 = get_part_of_data(200)
x_train_500, y_train_500 = get_part_of_data(500)
x_train_1500, y_train_1500 = get_part_of_data(1500)

modell00 = create_model()
model200 = create_model()
model500 = create_model()
```

```
model1500 = create_model()
modelall = create_model()

def teach_model(model, name, x_train, y_train):
  run(model, name, x_train, x_test, y_train, y_test)
  with open(f"{PATH}/models/{name}.pkl", "wb") as f:
    f.write(pickle.dumps(model))
  y_probs = model.predict(x_test)
  plt.rcParams.update({'font.size': 22})
  #Convert prediction probabilities into integers
  y_preds = y_probs.argmax(axis=1)
  cm=confusion_matrix(y_preds,y_test)
  #Plot
  disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=["Nevus",
"Melanoma"])
  fig, ax = plt.subplots(figsize=(8,8))
  disp.plot(ax=ax)
  print("-"*50,name,"-"*50)

teach_model(modell00, "cnn_modell00", x_train_100, y_train_100)

teach_model(model200, "cnn_model200", x_train_200, y_train_200)

teach_model(model500, "cnn_model500", x_train_500, y_train_500)

teach_model(model1500, "cnn_model1500", x_train_1500, y_train_1500)

teach_model(modelall, "cnn_modelall", x_train, y_train)
```

```
# Commented out IPython magic to ensure Python compatibility.
# %load_ext tensorboard

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_modell00

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_model200

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_model500

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_model1500

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/cnn_modelall

# -*- coding: utf-8 -*-
"""CustomVit.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1JPneuXLLV-0L8xqQMY3dXKL_HiGr0sz3
"""

from google.colab import drive
```

```
drive.mount('/content/drive')

!pip install -U tensorflow-addons

import matplotlib.pylab as plt
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_addons as tfa
import pickle
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D
from tensorflow.keras.models import Sequential
from sklearn.metrics import plot_confusion_matrix, ConfusionMatrixDisplay, confusion_matrix
import datetime

class Patches(layers.Layer):
  def __init__(self, patch_size):
    super(Patches, self).__init__()
    self.patch_size = patch_size

  def call(self, images):
    batch_size = tf.shape(images)[0]
    patches = tf.image.extract_patches(
        images=images,
        sizes=[1, self.patch_size, self.patch_size, 1],
        strides=[1, self.patch_size, self.patch_size, 1],
        rates=[1, 1, 1, 1],
```

```
        padding="VALID",
    )
    patch_dims = patches.shape[-1]
    patches = tf.reshape(patches, [batch_size, -1, patch_dims])
    return patches

class PatchEncoder(layers.Layer):
  def __init__(self, num_patches, projection_dim):
    super(PatchEncoder, self).__init__()
    self.num_patches = num_patches
    self.projection = layers.Dense(units=projection_dim)
    self.position_embedding = layers.Embedding(
        input_dim=num_patches, output_dim=projection_dim
    )

  def call(self, patch):
    positions = tf.range(start=0, limit=self.num_patches, delta=1)
    encoded = self.projection(patch) + self.position_embedding(positions)
    return encoded

def mlp(x, hidden_units, dropout_rate):
  for units in hidden_units:
      x = layers.Dense(units, activation=tf.nn.gelu)(x)
      x = layers.Dropout(dropout_rate)(x)
  return x

TARGET_IMAGE_SHAPE = (100, 100)
num_classes = 2
input_shape = (*TARGET_IMAGE_SHAPE, 3)
```

```
PATH = "drive/MyDrive/diploma"

class CustomAttention(layers.MultiHeadAttention):
  def _compute_attention(
    self, query, key, value, attention_mask=None, training=None
):
    # Note: Applying scalar multiply at the smaller end of einsum improves
    # XLA performance, but may introduce slight numeric differences in
    # the Transformer attention head.
    query = tf.multiply(query, 1.0 / (self._key_dim**0.5))

    # Take the dot product between "query" and "key" to get the raw
    # attention scores.
    attention_scores = tf.einsum(self._dot_product_equation, key, query)

    attention_scores = self._masked_softmax(
        attention_scores, attention_mask
    )

    # This is actually dropping out entire tokens to attend to, which might
    # seem a bit unusual, but is taken from the original Transformer paper.
    attention_scores_dropout = self._dropout_layer(
        attention_scores, training=training
    )

    # `context_layer` = [B, T, N, H]
    attention_output = tf.einsum(
        self._combine_equation, attention_scores_dropout, value
    )
```

```
    return attention_output, attention_scores

def call(
    self,
    query,
    value,
    key=None,
    attention_mask=None,
    return_attention_scores=False,
    training=None,
    use_causal_mask=False,
):
    attention_mask = self._compute_attention_mask(
        query,
        value,
        key=key,
        attention_mask=attention_mask,
        use_causal_mask=use_causal_mask,
    )

    if not self._built_from_signature:
        self._build_from_signature(query=query, value=value, key=key)
    if key is None:
        key = value

    query_is_ragged = isinstance(query, tf.RaggedTensor)
    if query_is_ragged:
        query_lengths = query.nested_row_lengths()
        query = query.to_tensor()
```

```
key_is_ragged = isinstance(key, tf.RaggedTensor)
value_is_ragged = isinstance(value, tf.RaggedTensor)
if key_is_ragged and value_is_ragged:
    # Ensure they have the same shape.
    bounding_shape = tf.math.maximum(
        key.bounding_shape(), value.bounding_shape()
    )
    key = key.to_tensor(shape=bounding_shape)
    value = value.to_tensor(shape=bounding_shape)
elif key_is_ragged:
    key = key.to_tensor(shape=tf.shape(value))
elif value_is_ragged:
    value = value.to_tensor(shape=tf.shape(key))

#   N = `num_attention_heads`
#   H = `size_per_head`
# `query` = [B, T, N ,H]
query = self._query_dense(query)

# `key` = [B, S, N, H]
key = self._key_dense(key)

# `value` = [B, S, N, H]
value = self._value_dense(value)

attention_output, attention_scores = self._compute_attention(
    query, key, value, attention_mask, training
)
```

```
    attention_output = self._output_dense(attention_output)

    if query_is_ragged:
        attention_output = tf.RaggedTensor.from_tensor(
            attention_output, lengths=query_lengths
        )

    if return_attention_scores:
        return attention_output, attention_scores
    return attention_output

def create_model(
image_size = 100, # We'll resize input images to this size
patch_size = 5, # Size of the patches to be extract from the input images
projection_dim = 64,
num_heads = 4,  # Size of the transformer layers
transformer_layers = 8,
mlp_head_units = [2048, 1024]):
  num_patches = (image_size // patch_size) ** 2
  transformer_units = [
      projection_dim * 2,
      projection_dim,
  ]
  data_augmentation = keras.Sequential(
    [
        layers.Normalization(),
        layers.Resizing(image_size, image_size),
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(factor=0.02),
```

```
        layers.RandomZoom(
            height_factor=0.2, width_factor=0.2
        ),
    ],
    name="data_augmentation",
)
inputs = layers.Input(shape=input_shape)
augmented = data_augmentation(inputs)
def get_patches(augmented, patch_size):
    patches = Patches(patch_size)(augmented)
    encoded_patches = PatchEncoder(num_patches, projection_dim)(patches)

small_encoded_patches = get_patches(augmented, patch_size)
large_encoded_patches = get_patches(augmented, patch_size*4)

for _ in range(transformer_layers):
    # Layer normalization 1.
    x1 = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
    # Create a multi-head attention layer.
    attention_output = CustomAttention(
        num_heads=num_heads, key_dim=projection_dim, dropout=0.1
    )(x1, x1)
    # Skip connection 1.
    x2 = layers.Add()([attention_output, encoded_patches])
    # Layer normalization 2.
    x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
    # MLP.
    x3 = mlp(x3, hidden_units=transformer_units, dropout_rate=0.1)
    # Skip connection 2.
```

```
    encoded_patches = layers.Add()([x3, x2])
  return encoded_patches



  representation = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
  representation = layers.Flatten()(representation)
  representation = layers.Dropout(0.5)(representation)
  # Add MLP.
  features = mlp(representation, hidden_units=mlp_head_units, dropout_rate=0.5)
  # Classify outputs.
  logits = layers.Dense(num_classes)(features)
  # Create the Keras model.
  return keras.Model(inputs=inputs, outputs=logits)

def run(model, name, x_train, x_test, y_train, y_test,learning_rate = 0.001,
weight_decay = 0.0001,
batch_size = 256,
num_epochs = 100):
  optimizer = tfa.optimizers.AdamW(
      learning_rate=learning_rate, weight_decay=weight_decay
  )

  model.compile(
      optimizer=optimizer,
      loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
      metrics=[
          keras.metrics.SparseCategoricalAccuracy(name="accuracy"),
          # keras.metrics.SparseTopKCategoricalAccuracy(5, name="top-5-accuracy"),
```

```
    ],
)

checkpoint_filepath = "/tmp/checkpoint"
checkpoint_callback = keras.callbacks.ModelCheckpoint(
    checkpoint_filepath,
    monitor="val_accuracy",
    save_best_only=True,
    save_weights_only=True,
)

log_dir = f"logs/fit/{name}/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

es_callback = keras.callbacks.EarlyStopping(
    monitor="val_loss",
    restore_best_weights=True,
    patience=10
)
history = model.fit(
    x=x_train,
    y=y_train,
    batch_size=batch_size,
    epochs=num_epochs,
    validation_split=0.1,
    callbacks=[checkpoint_callback, tensorboard_callback,
        #  es_callback
```

```
            ],
  )

  model.load_weights(checkpoint_filepath)
  _, accuracy = model.evaluate(x_test, y_test)
  print(f"Test accuracy: {round(accuracy * 100, 2)}%")
  # print(f"Test top 5 accuracy: {round(top_5_accuracy * 100, 2)}%")

from google.colab.patches import cv2_imshow

with open(f"{PATH}/data/split_data.pkl", "rb") as f:
  x_train, x_test, y_train, y_test = pickle.loads(f.read())

# x_train, x_test = x_train, x_test
# import cv2

# def increase_contrast(img):
#   lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
#   l_channel, a, b = cv2.split(lab)

#   # Applying CLAHE to L-channel
#   # feel free to try different values for the limit and grid size:
#   clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
#   cl = clahe.apply(l_channel)

#   # merge the CLAHE enhanced L-channel with the a and b channel
#   limg = cv2.merge((cl,a,b))

#   # Converting image from LAB Color model to BGR color spcae
```

```
#   enhanced_img = cv2.cvtColor(limg, cv2.COLOR_LAB2BGR)

#   # Stacking the original image with the enhanced image
#   result = np.hstack((img, enhanced_img))
#   return img
# for i in range(x_train.shape[0]):
#   x_train[i,:] = increase_contrast(x_train[i,:])

# for i in range(x_test.shape[0]):
#   x_test[i,:] = increase_contrast(x_test[i,:])

def get_part_of_data(n):
  x_train_n, y_train_n = x_train[:n], y_train[:n]
  print(f"x_{n} has {y_train_n[y_train_n==1].shape[0]} cancer imgs")
  return x_train_n, y_train_n

# x_train_100, y_train_100 = get_part_of_data(100)
# x_train_200, y_train_200 = get_part_of_data(200)
# x_train_500, y_train_500 = get_part_of_data(500)
# x_train_1500, y_train_1500 = get_part_of_data(1500)

# modell00 = create_model()
# model200 = create_model()
# model500 = create_model()
# model1500 = create_model()
modelall = create_model()

def teach_model(model, name, x_train, y_train):
  run(model, name, x_train, x_test, y_train, y_test)
```

```
  with open(f"{PATH}/models/{name}.pkl", "wb") as f:
    f.write(pickle.dumps(model))
  y_probs = model.predict(x_test)
  plt.rcParams.update({'font.size': 22})
  #Convert prediction probabilities into integers
  y_preds = y_probs.argmax(axis=1)
  cm=confusion_matrix(y_preds,y_test)
  #Plot
        disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=["Nevus",
"Melanoma"])
  fig, ax = plt.subplots(figsize=(8,8))
  disp.plot(ax=ax)
  print("-"*50,name,"-"*50)

list(i.output for i in modelall.layers)

from keras.models import Model
a = Model(modelall.input, modelall.layers[1].output)(x_test[:1])

cv2_imshow(a.numpy().reshape((256, 108))[:, 0].reshape((16, 16)))

for i in range(5):
  cv2_imshow(tf.image.extract_patches(
        images=x_test[:1],
        sizes=[1, 20, 20, 1],
        strides=[1, 20, 20, 1],
        rates=[1, 1, 1, 1],
        padding="VALID",
      )[0,i,i,:].numpy().reshape((20, 20, 3)))
```

```
x_test[:1].shape

teach_model(modelall, "vit_model_custom", x_train, y_train)



_, accuracy = modelall.evaluate(x_test, y_test)
print(accuracy)

modelall.layers

from keras.models import Model

model2 = Model(modelall.input,modelall.layers[-8].output)
model2.summary()

import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

import torch
import torch.nn as nn
import torchvision.models as models

import cv2
from PIL import Image
from torchvision import transforms
```

```
transform = transforms.Compose([
    transforms.Resize((100, 100)),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225],
    ),
])

model = model2
def get_attention_map(img, get_mask=False):
    x = transform(img)
    x.size()

    logits, att_mat = model(x.unsqueeze(0))

    att_mat = torch.stack(att_mat).squeeze(1)

    # Average the attention weights across all heads.
    att_mat = torch.mean(att_mat, dim=1)

    # To account for residual connections, we add an identity matrix to the
    # attention matrix and re-normalize the weights.
    residual_att = torch.eye(att_mat.size(1))
    aug_att_mat = att_mat + residual_att
    aug_att_mat = aug_att_mat / aug_att_mat.sum(dim=-1).unsqueeze(-1)

    # Recursively multiply the weight matrices
```

```
    joint_attentions = torch.zeros(aug_att_mat.size())
    joint_attentions[0] = aug_att_mat[0]

    for n in range(1, aug_att_mat.size(0)):
        joint_attentions[n] = torch.matmul(aug_att_mat[n], joint_attentions[n-1])

    v = joint_attentions[-1]
    grid_size = int(np.sqrt(aug_att_mat.size(-1)))
    mask = v[0, 1:].reshape(grid_size, grid_size).detach().numpy()
    if get_mask:
        result = cv2.resize(mask / mask.max(), img.size)
    else:
        mask = cv2.resize(mask / mask.max(), img.size)[..., np.newaxis]
        result = (mask * img).astype("uint8")

    return result
get_attention_map(x_test[0])

model2(x_test[:1])

# Commented out IPython magic to ensure Python compatibility.
# %load_ext tensorboard

# Commented out IPython magic to ensure Python compatibility.
# %tensorboard --logdir logs/fit/vit_model_custom
# -*- coding: utf-8 -*-
"""Preprocess.ipynb

Automatically generated by Colaboratory.
```

Original file is located at

https://colab.research.google.com/drive/1ygfVcctdPNHk2IrmUa99fCOC_C3W59e7
"""

```
from google.colab import drive
drive.mount('/content/drive')

import pickle
from sklearn.model_selection import train_test_split
import numpy as np

PATH = "drive/MyDrive/diploma"
TEST_SPLIT = 0.1
non_cancer_to_cancer = 2

with open(f"{PATH}/data/ham10000.pkl", "rb") as f:
  X, y = pickle.loads(f.read())

X_cancer = X[y != 0]
X_non_cancer = X[y != 1]
non_cancer_num = int(X_cancer.shape[0]*non_cancer_to_cancer)
X = np.vstack((X_cancer, X_non_cancer[:non_cancer_num]))
y = np.array([1]*X_cancer.shape[0] + [0]*non_cancer_num)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=TEST_SPLIT,
random_state=42)

with open(f"{PATH}/data/split_data.pkl", "wb") as file:
```

```
file.write(pickle.dumps((x_train, x_test, y_train, y_test)))
```