КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «магістр»

НА ТЕМУ:

Нейромережна система розпізнавання пухлин головного мозку на зображеннях МРТ з використанням глибинного навчання

Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітньо-наукова програма «Технології штучного інтелекту»

Виконала:
студентка 2 курсу магістратури, групи ТШІ-21
Лимар Д.С.
(ПІБ)
Науковий керівник: Снитюк В.Є.
(ПІБ)
доктор технічних наук, професор
(науковий ступінь, вчене звання)

Засвідчую, що в цій кваліфікаційній роботі немає запозичень з праць інших авторів без відповідних посилань

Студентка _____
підпис

Кваліфікаційна робота допущена до захисту
рішенням кафедри *інтелектуальних технологій*

Протокол № 12 від «11» травня 2023 р.

Зав. кафедри _____ доц. Іларіонов О.Є.
підпис

Київ 2023

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, 3 розділів, висновків, списку використаної літератури із 61 джерела та 9 додатків. Загальний обсяг роботи 135 сторінок. Робота містить 8 таблиць та 73 рисунки.

Актуальність теми. Пухлина головного мозку є одним із найпоширеніших та найагресивніших злоякісних пухлинних захворювань. Від швидкості та точності діагностики пухлини головного мозку залежать шанси на те, що лікування допоможе пацієнту одужати. Точність діагностики має велике значення для життя пацієнта, тому розробка та дослідження методів діагностики є актуальним напрямом. На сьогодні існує значна кількість методів діагностики, але не всі вони є однаково ефективними. Тож пошук рішень, що допоможуть підвищити ефективність існуючих методів є актуальною темою.

Мета кваліфікаційної роботи: розробка та дослідження технологій розпізнавання пухлин головного мозку на основі глибинного навчання нейронних мереж.

Об'єкт дослідження – розпізнавання аномальних утворень на знімках МРТ.

Предмет дослідження – нейромережні методи та засоби розпізнавання пухлин головного мозку на знімках МРТ на основі глибинного навчання.

Результати роботи. У випускній кваліфікаційній роботі проведено аналіз сучасних методів розпізнавання пухлин на зображеннях МРТ головного мозку, досліджено ефективність різних нейромережних методів розпізнавання пухлин головного мозку на зображеннях МРТ, виконано ансамблеве поєднання кількох моделей нейронних мереж для підвищення ефективності розпізнавання, розроблено нейромережну систему розпізнавання пухлин головного мозку на зображеннях МРТ.

Апробація роботи. Основні положення кваліфікаційної роботи були представлені на CXXII Міжнародній науково-практичній інтернет-конференції «Весняні наукові читання – 2023», 17 квітня 2023 року. Тези доповіді за

матеріалами кваліфікаційної роботи було опубліковано в збірнику праць конференції.

Ключові слова: пухлина головного мозку, розпізнавання образів, МРТ, згорткові нейронні мережі, глибинне навчання, ансамблеві методи.

ABSTRACT

The qualification work consists of an introduction, 3 chapters, conclusions, a list of references (61 sources) and 9 appendices. The total volume of work is 135 pages. The work contains 8 tables and 73 figures.

Actuality of theme. Brain tumor is one of the most common and aggressive malignant tumors. The chances that the treatment will help the patient recover depend on the speed and accuracy of the brain tumor diagnosis. Diagnostic accuracy is of great importance for a patient's life, so the development and research of diagnostic methods is relevant. There are a significant number of diagnostic methods to date, but not all of them are equally effective. Therefore, the search for solutions that will help to improve the effectiveness of existing methods is an actual issue.

The purpose of the qualification work: development and research of brain tumor detection technologies based on deep learning of neural networks.

The object of the research is the detection of abnormal formations on MRI images.

The subject of the research is neural network methods and tools for brain tumor detection on MRI images using deep learning.

Results of the work. In this graduation thesis the modern methods of brain tumor detection on MRI images are analyzed, the effectiveness of various neural network methods for brain tumor detection on MRI images is investigated, an ensemble combination of several neural network models was performed to improve detection efficiency, a neural network system for brain tumor detection on MRI images is developed.

Approbation of work. The main points of the qualification work were presented as thesis at the CXXII International Scientific and Practical Internet Conference «Spring Scientific Readings – 2023», April 17, 2023. Abstracts of the report based on the materials of the qualification work were published in the conference proceedings.

Keywords: brain tumor, pattern recognition, MRI, convolutional neural networks, deep learning, ensemble methods.

ЗМІСТ

ВСТУП

Метою роботи є розробка та дослідження технологій розпізнавання пухлин головного мозку на основі глибинного навчання нейронних мереж. Об’єктом дослідження є розпізнавання аномальних утворень на знімках МРТ. Предметом дослідження є нейромережні методи та засоби розпізнавання пухлин головного мозку на знімках МРТ на основі глибинного навчання. Сферою застосування технологій, що розглядаються, є медицина.

Пухлина головного мозку є одним із найпоширеніших та найагресивніших злоякісних пухлинних захворювань. Через низьку виживаність і агресивну природу, пухлина головного мозку вважається найбільш смертоносною і руйнівною хворобою. Швидка діагностика пухлини головного мозку здатна врятувати людині життя, піднявши шанси на те, що призначене лікарем лікування допоможе пацієнту.

Поширеність пухлини головного мозку викликає потребу у швидких і точних методах діагностики даного захворювання. Існує значна кількість досліджень у даній сфері, але незважаючи на це, досі існують певні обмеження у виявленні пухлин головного мозку. А зважаючи на руйнівний характер пухлини головного мозку, велику швидкість розвитку захворювання та високий рівень смертності, можна зробити висновок про необхідність проведення досліджень, спрямованих на підвищення якості діагностики даного захворювання.

Дослідження націлене на розпізнавання пухлин на зображеннях МРТ, що є найпоширенішим методом отримання візуалізованих медичних даних. Ручна діагностика є недоцільною при виконанні цієї задачі, оскільки трапляються випадки, коли на МРТ зображеннях немає видимих ознак, які б дозволили спеціалісту прийняти обґрунтоване рішення. Сучасні технології звісно не здатні повністю замінити лікарів, проте їм під силу допомогти спеціалістам у їх роботі та покращити її якість.

Згорткові нейронні мережі є найчастіше використовуваною технологією для діагностики захворювань на основі медичних зображень, завдяки своїй ефективності у подібних задачах. Тому саме моделі згорткових нейронних мереж були вибрані для дослідження у даній роботі, а врахування результатів прогнозу кількох моделей має підвищити ефективність розпізнавання. Використання згорткових нейронних мереж для задач розпізнавання досягло високого рівня розвитку і окремі моделі згорткових нейронних мереж уже показують досить високу ефективність. Окрема модель може показувати високу ефективність на одному наборі даних та бути неефективною на іншому. Поєднання кількох моделей може дозволити мінімізувати відхилення, характерні для окремих моделей та створити більш стабільну модель. На даний момент активно досліджуються можливості поєднання кількох моделей для класифікації, адже це може дозволити підвищити показники ефективності. Важко знайти сферу, у якій навіть мінімальне підвищення ефективності може бути настільки корисним, як у медицині. Тому було вирішено проводити дослідження саме у цій сфері.

Поширеною проблемою при проведенні досліджень у сфері медицини є обмеженість доступних наборів даних. З цього випливає, по-перше, необхідність підвищення ефективності моделей розпізнавання, для забезпечення можливості показувати хороші результати при навчанні на невеликому наборі даних. По-друге, це викликає потребу забезпечення механізму поповнення наборів даних для подальшого їх використання для повторного навчання моделей з метою покращення їх ефективності.

РОЗДІЛ 1. ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ РОЗПІЗНАВАННЯ ПУХЛИН ГОЛОВНОГО МОЗКУ НА ЗОБРАЖЕННЯХ МРТ

1.1 Актуальність проблеми виявлення пухлин на зображеннях МРТ головного мозку

Застосування інформаційних технологій в медицині набуло важливості в сучасному світі. Пухлинні клітини демонструють дуже невизначену поведінку, що занадто складно, для того, щоб контролюватись методами традиційної медицини. Для вирішення цієї проблеми на допомогу приходить штучний інтелект – наукова галузь, що займається розробкою машини, здатної навчатися самостійно без потреби у втручанні людини, для підготовки до самостійної роботи з потенційними випадками [1].

Пухлина головного мозку — це фіброзна сітка небажаного росту тканини всередині нашого мозку, яка безперешкодно розмножується. Пухлина головного мозку є одним із найпоширеніших та найагресивніших злоякісних пухлинних захворювань. Цей тип пухлин становить від 85% до 90% усіх первинних пухлин центральної нервової системи [2]. Через низьку виживаність і агресивну природу, пухлина головного мозку вважається найбільш смертоносною і руйнівною хворобою. Чим швидше буде діагностовано пухлину головного мозку, тим більші шанси на те, що лікування допоможе пацієнту одужати і його життя буде врятовано. Адже згідно з джерелом [3] типова пухлина головного мозку може подвоїтися протягом лише двадцяти п’яти днів. А якщо людина з пухлиною головного мозку не отримує вчасно необхідного лікування, то зазвичай при пізній діагностиці вона проживе з таким захворюванням не більше року. Рак головного мозку та інших органів нервової системи займає 10 місце серед причин смерті чоловіків і жінок [2]. У 2020 році у всьому світі приблизно у 308 102 людей діагностували первинну пухлину головного або спинного мозку [2], що говорить про значну

поширеність цього небезпечного захворювання, а отже і необхідність у швидких і точних методів діагностики.

Оцінка великої кількості зображень пухлин, отриманих у клініці, вручну є складним процесом, а крім цього недостатнім для розуміння поведінки різних пухлин. Це викликає необхідність у більш точних комп'ютерних технологіях виявлення пухлин, щоб забезпечити повну автоматизацію даного процесу.

Існують різні методи отримання медичних даних візуалізації, включаючи рентгенографію, магнітно-резонансну томографію (МРТ), томографію та ехокардіографію. Серед них МРТ є найвидатнішим, оскільки забезпечує зображення вищої роздільної здатності без будь-якого випромінювання [4]. Виявлення та відстеження пухлин за допомогою МРТ дає детальну інформацію про аномальні тканини, необхідні для терапевтичного втручання [1], проте це завдання є складним, через різноманітність форм і розмірів пухлин. Навіть за використання передових технологій не вдається уникнути проблем з часом та точністю. Незважаючи на численні дослідження у даній сфері, все ще існують певні обмеження у виявленні пухлин головного мозку через варіації розташування пухлини, типу, розміру та форми [5]. Знайти ділянку з невеликою кількістю уражень може бути важко, оскільки невеликі ділянки, як правило, виглядають здоровими. Отже, визначення типу пухлини за допомогою МРТ є складним, схильним до помилок і займає багато часу, тому потрібні радіологи з великим досвідом. Іноді на МРТ зображеннях немає видимих ознак, які б дозволили прийняти обґрунтоване рішення, що свідчить про недоцільність використання ручної діагностики.

Глибоке навчання є потужним та перевіреним інструментом машинного навчання і вже використовувалося в кількох програмах для вирішення різноманітних складних проблем, які вимагають надзвичайно високої точності та чутливості, зокрема в галузі медицини [6].

Останнім часом для розпізнавання пухлин головного мозку було застосовано багато методів штучного інтелекту, таких як штучна нейронна мережа (ANN), метод опорних векторів (SVM) і згорткова нейронна мережа

(CNN) [6]. CNN представляє найновіші досягнення у галузі машинного навчання і використовується в галузі діагностики захворювань на основі медичних зображень. Важливою перевагою цього методу є те, що він не потребує попередньої обробки або виділення ознак перед процесом навчання.

Беручи до уваги руйнівний характер пухлини головного мозку, велику швидкість розвитку захворювання та високий рівень смертності можна зробити висновок про значну необхідність проведення досліджень націлених на підвищення якості та покращення швидкості діагностики хвороби. Незважаючи на те, що вже наявні певні методи автоматизованої діагностики даного захворювання, досягнуті на даний момент результати у цій області недостатньо хороші для припинення активних досліджень і пошуків нових рішень.

1.2 Представлення нейрона

З точки зору біології у нервовій системі один основний елемент – нейрон, у якого виділяють тіло клітини, що називають сома, з нього виходять 2 відростки: дендрит і аксон. Будова нейрона представлена на рисунку 1.1.

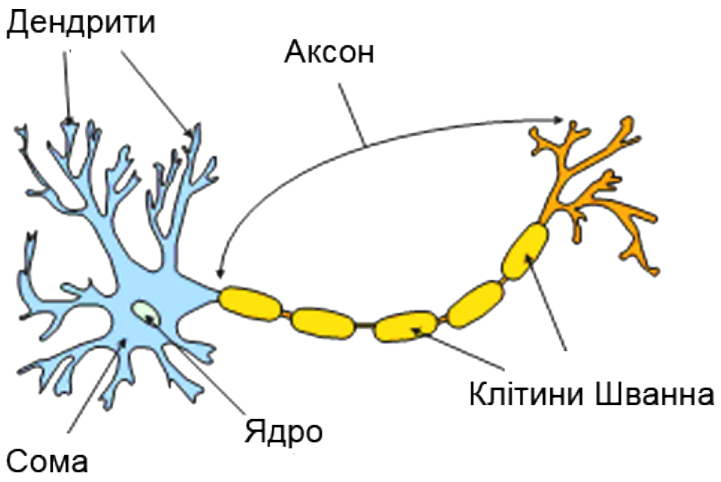


Рисунок 1.1 – Будова нейрона [7]

Дендрит забезпечує отримання інформації тілом, а аксон – передачу інформації. Нейрон передає збудження до інших нейронів завдяки синапсам, тобто нервовим з'єднанням. Синапси функціонують як підсилювачі, до нейрона надходять сигнали, один з них сумує збудження, інший – гальмівні імпульси.

Нейрон сумує збудження та гальмівні імпульси, і якщо ця сума перетворює визначене порогове значення, то сигнал пересилається до інших нейронів.

Штучний нейрон працює аналогічним чином, маючи з'єднання та ваги. Кожен нейрон має свій поріг. Для розрахунку суми активації нейрона потрібно скласти суму всіх ваг і відняти граничне значення. Вихідний сигнал отримується шляхом перетворення сигналу активації за допомогою передавальної функції. Штучний нейрон має три основні елементи: синапс, суматор та функція активації. Будова штучного нейрону представлена на рисунку 1.2.

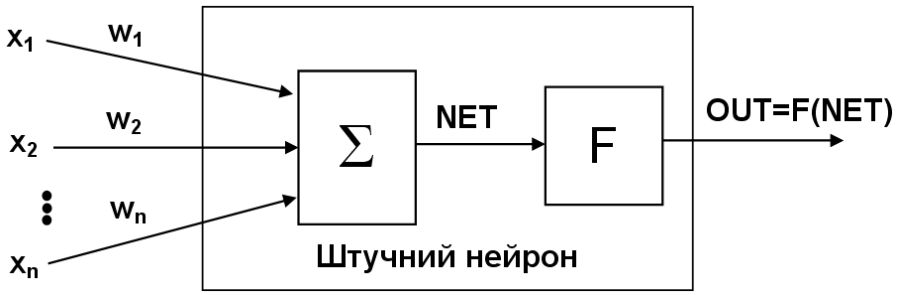


Рисунок 1.2 – Будова штучного нейрону [8]

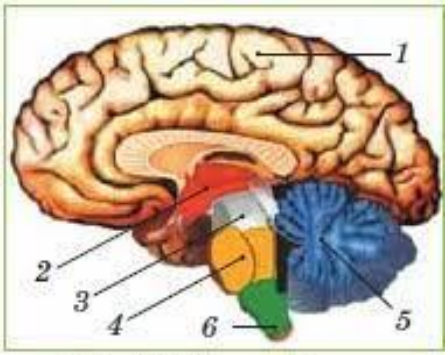
Через синапс нейрони зв’язуються між собою. Кожен синапс має вагу, тобто важливість, що надається значенню сигналу, що проходить через цей синапс. Суматор сумує вхідні сигнали інших нейронів. Функція активації нейрона визначає вихідний сигнал, що визначається вхідним сигналом або набором вхідних сигналів.

1.3 Будова головного мозку

У головному мозку розрізняють три великі частини: великий мозок, малий мозок або мозочок і мозковий стовбур [9]. Найбільший об’єм головного мозку займають півкулі.

Головний мозок ділиться на п’ять відділів [9]: довгастий мозок, задній мозок, середній мозок, проміжний мозок, кінцевий мозок.

Схема будови головного мозку представлена на рисунку 1.3.



Мал. 88. Схема будови головного мозку: *1* – кінцевий мозок; *2* – проміжний мозок; *3* – середній мозок; *4* – міст; *5* – мозочок; *6* – довгастий мозок

Рисунок 1.3 – Схема будови головного мозку [10]

У головному мозку виділяють мозковий стовбур і півкулі головного мозку, які у людини є найбільш розвиненими. Мозковий стовбур складається з довгастого, заднього (за виключенням мозочка), середнього і проміжного відділів.

– Довгастий мозок.

Довгастий мозок є продовженням спинного в стовбур головного мозку і він поєднує у своїй будові риси будови спинного мозку й початкового відділу головного мозку [9]. За своєю формою він нагадує конус і на його зовнішній поверхні є дві центральні борозни: передня і задня, котрі є продовженням спинномозкових. Сіра речовина усередині довгастого мозку розташовується у вигляді окремих ядер, а біла речовина складається з довгих і коротких волокон. У довгастому мозку присутня сітчаста речовина, що називається ретикулярною формацією, інакше кажучи це нейронні структури, де різні за своїм розміром нейрони тісно переплетені між собою за допомогою своїх відростків. ЇЇ основними функціями є активація нейронів кори великих півкуль. У цій сітчастій речовині розташовуються нервові центри, що представляють собою сукупність нервових клітин, які виконують певні функції. Ретикулярна формація грає важливу роль у підтриманні стану неспання мозку.

У довгастому мозку розміщуються ядра чотирьох пар черепно-мозкових нервів. Довгастий мозок відповідає за такі рефлекси [9]: гемодинамічні, що здійснюють регуляцію діяльності судин і серця; дихальні; харчові; захисні рефлекси (кашлю, чихання, миготіння, сльозовиділення).

– Задній мозок.

Задній мозок складається з мосту й мозочка. Міст розташовується на схилі потиличної кістки і представляє собою майже чотирибічний великий білий вал. За характером розміщення волокон розділяють такі частини, як основа – розміщення внизу, трапецієподібне тіло – розміщення посередині, покрив – розміщення зверху.

– Мозочок.

Мозочок є похідним заднього мозку, який розвинувся у зв'язку з рецепторами статики. Він є органом пристосування організму до подолання основних властивостей маси тіла – ваги й інерції, крім цього він має пряме відношення до координації рухів. Мозочок розміщується під потиличними частками півкуль кінцевого мозку і складається з непарної середньої частини – черв'яка й парних частин – півкуль. Він має верхню, задню й нижню півкулі, утворені групами звивин мозочка. Складається мозочок із сірої речовини, яка утворює його кору, та білої речовин. Мозочок є центром координації роботи м'язів і несвідомим центром болю. Однією з основних виконуваних ним в здійсненні рухових актів функцій є полегшення спрацьовування антагоністичної мускулатури на початку й наприкінці руху.

– Середній мозок.

Середній мозок знаходиться поруч з тілом основної кістки черепа. У людини він є найменшим і має найпростішу будову. Він складається із покрівлі, покриву й ніжок мозку.

– Проміжний мозок.

Проміжний мозок розміщується на рівні турецького сідла основної кістки черепа, під мозолистим тілом і склепінням, він складається з двох ділянок і порожнини, що називається третім шлуночком.

– Кінцевий мозок.

Кінцевий мозок є найбільшим відділом головного мозку. По зовнішній поверхні мозку проходить поздовжня щілина, що розділяє великий мозок на дві півкулі: праву й ліву, які зв’язані одна з одною спайками. Порожнини кінцевого мозку утворюють правий і лівий бічні шлуночки. Бічний шлуночок складається з таких частин: передній ріг, центральна частина, задній ріг і нижній ріг. Півкулю розділяють на такі частини: лобова, тім’яна, потилична, скронева, острівець.

Лобова частка у людини бере участь у формуванні складних програм поведінки, а також у деяких видах пам’яті, головним чином на недавні події і забезпеченні координації рухів.

У тім’яній частці формуються відчуття локалізації, ваги, шорсткості, напрямку руху, просторове чуття, у ній також міститься центр письма, читання, мовної пам’яті.

Потилична частка зв’язана із зоровою функцією.

Скронева частка має широкі зв’язки з різними відділами мозку, бере участь у механізмах мови.

Кінцевий мозок складається із сірої й білої речовин. В ньому виділяють у порядку історичного розвитку три частини: стародавня – представлена нюховим мозком; давня – представлена базальними ядрами, які відповідають за несвідомі рухи; нова – представлена плащем, і покриває інші частини.

1.4 Розпізнавання образів

Розпізнавання образів є розділом теорії штучного інтелекту, що вивчає методи класифікації об’єктів. Образом є об’єкт, який розпізнається за ознаками, що збираються та обробляються індивідуально і у сукупності [11]. Ознаки ж є описом образу.

Система розпізнавання образів представляє собою електронно-обчислювальний комплекс, що може моделювати розумові процеси, властиві

людині під час прийняття рішень із метою виявлення аналогій серед досліджуваних об’єктів [12].

Для розпізнавання образів необхідно виконати розбиття простору ознак розпізнавання на області, що відповідають певному класу об’єктів, та визначити приналежність вибраного для розпізнавання образу до відповідного класу.

Загальний алгоритм, за яким працюють системи розпізнавання образів можна представити так [11]:

– сприйняття об’єкта за допомогою датчиків у вигляді деякого вектора оброблених кодованих ознак, які також можуть піддаватись додатковій обробці;
– розпізнання (класифікація) об’єкта у певному логічному пристрої на основі значень ознак; еталонні вектори кодованих ознак зберігаються у пам’яті системи розпізнавання образів і конкурують між собою за правилом «переможець отримує все» за «привласнення» вхідного кодованого вектора об’єкта шляхом порівняння відстані, або близькості вектора з даними еталонів.
– прийняття рішення про виконання відповідних дій у логічному пристрої, що може бути спрямована або на зміну параметрів об’єкта за допомогою зворотного зв’язку, або на передачу інформації в іншу інтелектуальну систему для запам’ятовування отриманих результатів чи подальшої їх обробки.

Типовими задачами розпізнавання є:

– ідентифікація, тобто визначення певного об’єкта серед йому подібних;
– класифікація, тобто віднесення об’єкта до певного класу;
– кластерний аналіз, тобто поділ заданого набору об’єктів на класи за їх подібністю за певним критерієм.

У даній роботі виконується задача класифікації.

Задача класифікації належить до задач машинного навчання з учителем [13]. Метою класифікації є поділ деякої множини об’єктів на заздалегідь

визначену кількість класів. При цьому для підмножини об’єктів, яку називають навчальною вибіркою, клас об’єкта заздалегідь відомий.

Задача класифікації – це задача розбиття множини об'єктів або спостережень на апріорно задані групи, називані класами, всередині кожної з яких вони вважаються схожими один на одного, та мають приблизно однакові властивості й ознаки [14]. При цьому рішення здійснюється на основі аналізу значень атрибутів (ознак).

Випадок, коли всього визначено два класи, називають бінарною класифікацією.

Введемо позначення і сформулюємо математичну постановку задачі класифікації [15].

Нехай Ω – простір образів; $\omega \in \Omega$ – образ; $M = \{1,2,\dots,m\}$ – номери класів $\Omega_1, \Omega_2, \dots, \Omega_m$, таких що $\Omega_i \cap \Omega_j$, якщо $i \neq j$ і $\bigcup_{i=1}^{m} \Omega_i = \Omega$; $g: \Omega \to \mathrm{M}$ – індикаторна функція, що є невідомою; X – простір ознак, тобто векторний простір, точками якого є вектори ознак образів; $x: \Omega \to \mathrm{X}$ – функція, що ставить у відповідність образу ω його вектор ознак $x(\omega)$; $K_1, K_2, \dots, K_m$ – підмножини простору X, такі що $K_i \cap K_j$, якщо $i \neq j$ і $\bigcup_{i=1}^{m} K_i = \mathrm{X}$; $\hat{g}: X \to M$ – вирішальне правило, яке ставить у відповідність вектору ознак образа номер класу, якому він належить.

Задача класифікації з учителем полягає у тому, щоб на підставі множини прецедентів (g_j, x_j), $j = 1, \dots, N$, яка називається навчальною вибіркою, побудувати вирішальне правило $\hat{g}$, що мінімізує кількість помилок [15]. Вчителем вважається або сама навчальна вибірка, або той, хто указав значення.

Задача класифікації без учителя часто називається кластеризацією. В цій задачі вибірка образів x_j, $j = 1,2,\dots,N$ розбивається на підмножини, що не перетинаються (кластери), які складаються із схожих один на одного об’єктів, до того ж вимагається, щоб об’єкти із різних кластерів істотно відрізнялися один від одного [15].

Задача кластеризації [16] полягає у визначенні груп об'єктів (процесів), які є найближчими один до одного за деяким критерієм. Якщо критерієм або метричним свідченням є відстань, то кластером називають групу точок Ω, таку, що середній квадрат внутрішньогрупової відстані до центру групи менший за середню відстань до загального центру в початковому наборі об'єктів, тобто $\bar{d}_\Omega^2 < d^2$, де $\bar{d}_\Omega^2 = \frac{1}{N}\sum_{X_i \in \Omega}(X_i - \bar{X}_\Omega)^2$, $\bar{X}_\Omega = \frac{1}{N}\sum_{X_i \in \Omega} X_i$, N – кількість точок у кластері Ω. Початковими даними завдання кластеризації є значення параметрів об'єктів дослідження. Найчастіше визначення оптимальної кількості кластерів є прерогативою дослідника. Припустимо, що число кластерів K задане і $K \ll m$, де m кількість об'єктів. Отримаємо задачу $\sum_{i=1}^{K}\sum_{j=1}^{m_i}\left\|X_j - \bar{X}_i\right\| \to min$, де $m_i, i = \overline{1,K}$, – кількість об'єктів у i-му кластері, $\bar{X}_i, i = \overline{1,K}$, – середнє значення в кластері, $\left\|X_j - \bar{X}_i\right\|$ – відстань між об'єктами.

Для проведення класифікації за допомогою математичних методів необхідно мати формальний опис об'єкта, яким можна оперувати, використовуючи математичний апарат класифікації [14]. Зазвичай таким описом виступає база даних, у якій кожен запис несе інформацію про деяку властивість об'єкта.

Набір вхідних даних розбивають на навчальну та тестову вибірки.

Навчальна вибірка включає об'єкти, для яких відомі значення як незалежних, так і залежних змінних. На підставі навчальної вибірки будується модель визначення значення залежної змінної [14]. Її зазвичай називають функцією класифікації. Для одержання максимально точної функції до навчальної вибірки пред'являються такі основні вимоги:

– кількість об'єктів у вибірці має бути достатньо великою;

– вибірка має включати об'єкти, що представляють усі можливі класи;

– для кожного класу у вибірці має бути достатня кількість об'єктів.

Тестова вибірка також містить вхідні та вихідні значення параметрів, але вихідні значення тестової вибірки використовуються лише для перевірки працездатності моделі.

Процес класифікації складається із двох етапів [14]:

1. Конструювання моделі здійснюється на основі навчальної вибірки. В результаті одержуємо модель, яка представляється або класифікаційними правилами або деревом розв'язків або математичною формулою або комп'ютерним об'єктом (як нейронні мережі).
2. Використання моделі: класифікація нових або невідомих значень. На цьому етапі виконується оцінка точності моделі. Відомі значення з тестового набору порівнюються з результатами використання отриманої моделі. За рівень точності ухвалюється відсоток правильно класифікованих прикладів у тестовій вибірці. Якщо точність моделі прийнятна, модель можна використовувати для класифікації нових прикладів, клас яких невідомий.

Проблеми, що зустрічаються при розв'язку задач класифікації [14]:

– Перенавчання. Суть перенавчання полягає в тому, що класифікаційна функція при побудові «занадто добре» адаптується до даних, і помилки, що зустрічаються в них, і аномальні значення намагається інтерпретувати як частину внутрішньої структури даних. Така модель буде некоректно працювати надалі з іншими даними, де характер помилок буде дещо іншим. Ця ситуація проявляється в тому, що помилка на тестовому наборі значно більше за помилку на навчальному;
– Недонавчання. Цим терміном позначають ситуацію, коли спостерігається занадто велика кількість помилок при перевірці класифікатора на навчальній множині. Це означає, що особливих закономірностей у даних не було виявлено. У такому випадку їх або немає взагалі, або необхідно вибрати інший метод їх виявлення.

1.5 Обробка зображень МРТ

Магнітно-резонансна томографія (МРТ) – це метод відображення, що використовується, для отримання високоякісних зображень органів людського тіла. Цей метод заснований на поглинанні та випромінюванні енергії в радіочастотному діапазоні електромагнітного спектра. На рисунку 1.4 представлено приклад МРТ зображень головного мозку людини.

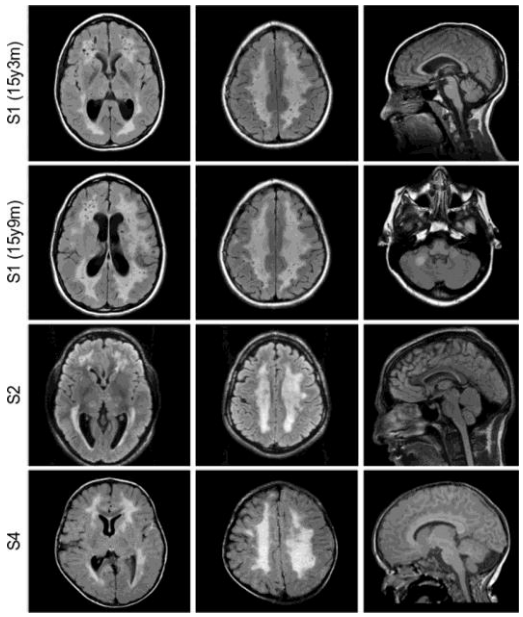


Рисунок 1.4 – Зображення МРТ головного мозку людини [17]

Використання саме зображень МРТ для діагностики є найбільш широко застосовуваним методом візуалізації в медичній галузі, головним чином завдяки високій роздільній здатності таких знімків і їх здатності чітко відображати структуру, розмір і розташування пухлин. Іншими перевагами МРТ є можливість зображення будь-якої площини, що дозволяє отримувати тривимірні зображення, здатність детального дослідження всіх анатомічних областей завдяки невидимості кістки на зображенні і, звісно, відсутність іонізуючого випромінювання.

Існують звісно і недоліки МРТ, такі як висока вартість необхідного обладнання та необхідність відповідності приміщення його розміщення спеціальним вимогам, та не моментальність отримання зображення, що може негативно вплинути на його якість через високу чутливість до рухових артефактів.

Знімки МРТ можуть мати такі спотворення, як шуми, невідповідна освітленість об’єкта дослідження, низька контрастність, артефакти візуалізації. Для покращення якості зображення МРТ до нього застосовують методи попередньої обробки. Попередня обробка зазвичай є першим кроком під час обробки зображень. Для зміни контрасту, покращення чіткості та вирівнювання тону, наприклад, можуть використовуватись градаційні перетворення.

Обробка зображень МРТ для виявлення та класифікації пухлин головного мозку є важливою проблемою в галузі медицини. Зображення МРТ можуть бути занадто темними або яскравими тощо, що значно ускладнює визначення медиками уражень мозку. Одним із методів обробки медичних зображень, що сприяє полегшенню його аналізу, є сегментація [18]. Сегментація передбачає розбиття зображення на рівні частини, що дозволяє зосередити увагу на відповідному об’єкті. Крім цього до зображення може застосовуватись виділення ознак, що передбачає пошук особливостей зображення МРТ. Використовуються також такі методи попередньої обробки, як вейвлет-перетворення, фільтрація, текстурний аналіз з використанням матриці суміщення рівнів сірого, маркування об’єктів. Обробку зображень МРТ також можна виконувати за допомогою методів інтелектуального аналізу даних, які складаються з чотирьох етапів, включаючи попередню обробку для першого кроку, сегментацію зображення, виділення ознак за кольором, формою чи текстурою та класифікація для ідентифікації пухлини мозку [18].

Попередня обробка передбачає зміну завеликих зображень МРТ на певні пікселі відповідно до потреб обробки зображення, а потім якість зображення покращуватиметься шляхом налаштування яскравості, контрастності тощо. Для попередньої обробки можуть виконуватись такі етапи: підвищення яскравості; порогування, що змінює зображення у відтінках сірого на двійкове; фільтрація; контроль меж об’єкта шляхом застосування маски; вилучення ознак, метою якого є збір даних зображення у вигляді форми, текстури, кольору та контрасту [18].

Фільтрація, мабуть, є найпростішим методом, він передбачає видалення аспектів, що відповідають або не відповідають вказаному порогу. Однак, при фільтрації існує висока ймовірність видалення дійсних сигналів та створення нових артефактів.

Сегментація необхідна для полегшення отримання точної інформації, адже завдяки її використанню можна отримати пікселі або об’єкти відповідно до їх текстури чи кольору, і розділити об’єкти та фон, або ж розділити зображення МРТ на кілька сегментів, для полегшення його аналізу. Сегментація розділяється на сегментацію на основі пікселів, на основі країв і на основі регіонів, де тип обирається в залежності від поставленої мети.

Класифікація зображень МРТ за допомогою методу опорних векторів є одним із найефективніших алгоритмів машинного навчання. Класифікація повинна визначати шаблон до отримання зображення, яке буде оброблено [18]. Метод виконує навчання великих наборів даних і досліджує ці моделі даних.

1.6 Огляд існуючих систем розпізнавання пухлин головного мозку на зображеннях МРТ

У літературі стверджується, що машинне навчання, зокрема глибоке навчання, має потенціал для подолання проблем, пов’язаних із виявленням та втручанням у пухлини мозку [1]. Глибоке навчання є однією з операцій штучного інтелекту і нагадує завдання людського мозку. Згорткові нейронні мережі — це глибокі нейронні мережі, які часто використовуються в глибокому аналізі візуального зображення [19]. У цьому розділі представлено дослідження існуючої літератури щодо застосування глибокого навчання в медичній діагностиці.

МРТ є, мабуть, найбільш часто використовуваним методом візуалізації ділянок мозку через свою властивість згладжувати відмінності в тканинах, що робить цей метод візуалізації найбільш універсальним для моделювання цікавих ділянок мозку, таких як пухлини.

Ефективні механізми, залучені до класифікації пухлинних і непухлинних одиниць у зображенні МРТ головного мозку, розглядаються в [20]. Запропонована у роботі [1] стратегія використовує CNN і VGG 16 для виявлення пухлин мозку за допомогою даних МРТ мозку. У даній роботі [1] також представлена порівняльна таблиця, що коротко описує суттєві характеристики представлених раніше робіт з напряму діагностики пухлин головного мозку. Вище згадана порівняльна таблиця включає деякі обмеження та недоліки попередніх робіт, а також отримані результати.

У роботі [21] запропоновано систему, яка поєднує функції дискретного вейвлет-перетворення і методи глибокого навчання, точність отриманих у дослідженні результатів досягла рівня 96,97%. Дослідження [22] демонструє глибоку систему на основі CNN для автоматизованого виявлення та класифікації пухлин головного мозку. Система базується на методі нечітких с-середніх для сегментації мозку, і на основі цих сегментованих областей було витягнуто текстуру та особливості форми, а потім ці характеристики введено в класифікатори SVM і DNN, результат точності системи досяг 97,5%.

У статті [6] запропоновано нову архітектуру CNN, що складається з 18 рівнів, завдяки чому може дозволити класифікатору ефективно класифікувати пухлину мозку. Дана архітектура є зміненим варіантом архітектури, представленої у документі [23], у роботі модифікована архітектура застосувалась до трьох різних наборів даних зображень: обрізаних, не обрізаних та сегментованих. Отримані результати перевищили 95% для всіх випадків.

У статті [24] нейронні мережі використовуються для класифікації нормального та пухлинного мозку. У роботі використано згортковий клас нейронних мереж MobileNet, розроблений дослідниками Google, а саме MobileNetV2. У результаті експерименту було отримано точність 89%.

Реалізована у дослідницькій роботі [25] згорткова нейронна мережа забезпечує загальну точність 91,3%. У документі [26] представлено метод, що базується на голосуванні Хафа, стратегії, яка дозволяє повністю автоматично

локалізувати та сегментувати цікаві анатомії. Отримані результати експерименту показали середній успіх 95,62%.

У роботі [27] самовизначена штучна нейронна мережа і згорткова нейронна мережа застосовуються для виявлення пухлини головного мозку та аналізується їх ефективність. Представлена модель штучної нейронної мережі забезпечує 65,21% точності тестування.

У дослідженні [5] запропоновано гібридну модель глибокого навчання CNN-LSTM на основі CNN для класифікації пухлин головного мозку. Запропонована модель досягла 99,1% точності.

У роботі [4] представлені дві окремі моделі для діагностики бінарних (нормальних і аномальних) і мультикласових (менінгіома, гліома та гіпофіз) пухлин головного мозку. У документі також пропонується порівняння запропонованих моделей з вже існуючими сучасними моделями, знайденими в літературі. У роботі пропонуються такі архітектури: 23-шарова CNN та тонко налаштована CNN із додатком трансферного навчання на основі VGG16. У результаті було отримано високу точність – 100% та 97,8%. Але представлені моделі стикаються з певними труднощами, однією з ключових є вимога до значної кількості анотованих зображень, зібраних кваліфікованим лікарем або радіологом, що не завжди є доступним.

У роботі [28] представлено генетичний алгоритм (GA) з CNN для прогнозування пухлин мозку. Однак GA не завжди демонструє хорошу точність при роботі з CNN і крім цього є обчислювально дорогою моделлю. У дослідженнях [29] та [30] застосована попередньо підготовлена модель VGG19.

У роботах [31] та [32] використовувалась модель CNN та отримано точність понад 90%. Однак обидві моделі є дорогими з точки зору обчислень і не пропонують метод перевірки системи, що спричиняє можливість виникнення ситуації, коли конкретна модель може добре працювати на одному наборі даних, і невдало на іншому.

У роботі [33] пропонується метод класифікації пухлин головного мозку з використанням ансамблю глибинних ознак і класифікаторів машинного

навчання. У цьому дослідженні запропоновано гібридне рішення, яке використовує різні попередньо навчені глибокі згорткові нейронні мережі як екстрактори ознак і різні класифікатори машинного навчання для ідентифікації нормальних і аномальних зображень МРТ головного мозку. Три найкращі глибинні ознаки, які добре працюють на кількох класифікаторах машинного навчання, обираються та об'єднуються в ансамбль глибоких ознак, який потім подається в декілька класифікаторів машинного навчання для прогнозування кінцевого результату.

Дослідження літературних джерел, у яких представлені технології для виконання того ж завдання, що реалізовується у даній роботі, чи подібних до нього показало, що на даний момент існує значна кількість реалізацій вирішення поставленої задачі. Проте незважаючи на це, дослідження у даному напрямі продовжуються, оскільки універсальне рішення з максимальною точністю результатів, високою швидкістю та здатністю до якісного навчання на невеликій вибірці даних ще не було знайдено. Наприклад, для більшості існуючих рішень потрібні ручні методи виділення ознак, що може бути не дуже ефективними при роботі з великою кількістю зображень.

1.7 Класифікація нейронних мереж

Нейронні мережі різняться своєю архітектурою, яка обирається в залежності від поставленої задачі. За архітектурою ШНМ можна розділити на два класи (рисунок 1.5). Перший клас представляє мережі прямого поширення, у яких графове представлення не має циклів, другий – рекурентні мережі або мережі зі зворотними зв'язками [34].

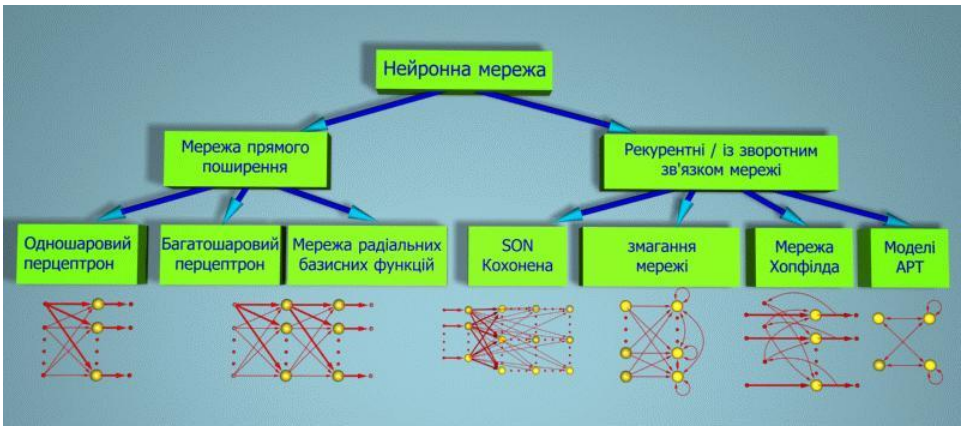


Рисунок 1.5 – Класифікація нейронних мереж за архітектурою [34]

Крім поділу за архітектурою, штучні нейронні мережі також розділяють за їх структурою, особливістю моделі нейрона, та особливостями навчання самої мережі.

За своєю структурою нейронні мережі можуть бути повнозв’язними чи неповнозв’язними, з випадковими чи регулярними зв'язками, з симетричними чи несиметричними зв'язками [34].

За особливостями моделі нейрону розрізняють нейрони з різними нелінійними функціями.

За організацією навчання розділяють навчання з вчителем та без вчителя.

З точки зору топології виділяють три основні типи нейронних мереж: повнозв’язні, багатошарові та слабозв’язні.

Кожен нейрон повнозв’язної мережі передає свій вихідний сигнал іншим нейронам, в тому числі і самому собі. Всі вхідні сигнали подаються на всі нейрони. Вихідними сигналами мережі можуть бути всі або деякі вихідні сигнали нейронів після кількох циклів роботи мережі.

В багатошарових нейронних мережах нейрони об'єднані у шари. Кожен шар включає в себе сукупність нейронів з унікальними вхідними сигналами. Кількість нейронів у шарі незалежна від їх кількості у інших шарах і може приймати будь-яке значення. Загалом мережа складається з певної кількості шарів, що нумеруються зліва направо. На входи нейронів вхідного шару надходять зовнішні сигнали. Виходи нейронної мережі є вихідними сигналами останнього шару. Крім цих двох шарів багатошарова нейронна мережа включає

ще один або більше прихованих шарів. Зв’язки між виходами нейронів попереднього шару та входами наступного називають послідовними. Серед багатошарових нейронних мереж можна виділити монотонні, мережі без зворотних зв’язків та з ними.

У монотонних нейронних мережах всі шари крім останнього діляться на два блоки: збудливий і гальмуючий, як і зв’язки між блоками.

У мережах без зворотних зв'язків нейрони вхідного шару отримавши на вхід сигнали, виконують їх перетворення і пересилають до нейронів першого прихованого шару, ця процедура продовжується, доки сигнали не будуть передані на вихід. Приклад багатошарової нейронної мережі прямого поширення представлено на рисунку 1.6.

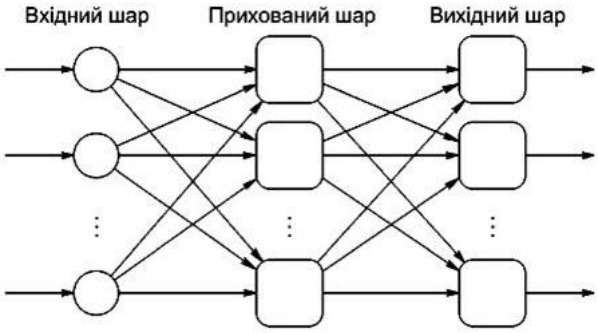


Рисунок 1.6 – Багатошарова (2 шари) нейронна мережа прямого поширення

В мережах зі зворотними зв'язками інформація з наступних шарів надсилається до попередніх. На рисунку 1.7 представлено приклад мережі зі зворотними зв'язками.

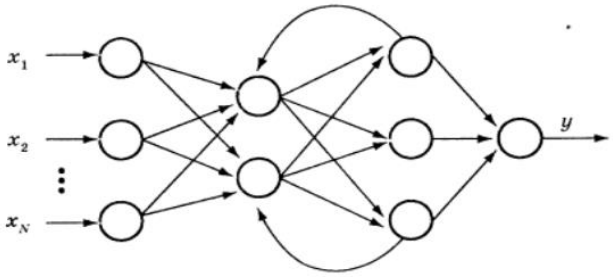


Рисунок 1.7 – Штучна нейронна мережа із непрямими зворотними зв’язками

У слабозв'язаних нейронних мережах нейрони розміщуються у вузлах прямокутної або гексагональної решітки, і кожен з них зв’язаний з чотирма, шістьма або вісьмома найближче розташованими сусідами.

Найбільш відомими та поширеними моделями ШНМ є такі, як:

– мережа Хопфілда;
– мережа Кохонена;
– рекурентна нейронна мережа;
– згорткова нейронна мережа;
– персептрон (одно- та багатошаровий).

1.8 Навчання нейронної мережі

Навчанням нейронної мережі називають процес налаштування її параметрів шляхом моделювання середовища, в яке вбудована модель. Навчання полягає у виробленні правильної реакції на подані нейронній мережі різні вхідні сигнали [35]. Типи навчання поділяються на навчання з та без вчителя.

Навчання з учителем полягає у поданні мережі набору навчальних прикладів. Зразок надсилається на входи мережі, після чого виконується його обробка в структурі мережі та обчислення виходу. Далі відбувається порівняння з відповідним значенням цільового вектора. Після чого за певним правилом обчислюється похибка і виконується зміна вагових коефіцієнтів зв'язків мережі, відповідно до вибраного алгоритму. Коригування проводиться до тих пір, поки похибка не досягне необхідного рівня.

При навчанні без учителя навчальний набір включає лише вхідні вектори, а алгоритм навчання коригує ваги мережі з метою отримання такого результату, при якому подання досить близьких вхідних векторів дає однакові результати, тобто відбувається об’єднання подібних векторів у класи. Реакція мережі на вхідний вектор з певного класу до її навчання є непередбаченою, тому у ході навчання виходи мережі мають перетворюватись в деяку зрозумілу форму. Це не несе значного обмежувального впливу, оскільки зазвичай ідентифікація

зв’язку між вхідними векторами й відповідною реакцією мережі не є складною задачею.

Крім двох попередньо зазначених видів навчання, існує ще один, який називають навчанням з підкріпленням. Даний вид передбачає наявність учителя, але на відміну від класичного навчання з учителем, у цьому випадку учитель не підказує мережі правильної відповіді, а лише повідомляє, правильно чи неправильно мережа опрацювала поданий на вхід образ [35]. Базуючись на цьому мережа корегує свої параметри, надаючи зв’язкам, що правильно зреагували на вхідний сигнал більшу вагу, а всім іншим – меншу.

На рисунку 1.8 схематично зображено процес навчання нейронної мережі.

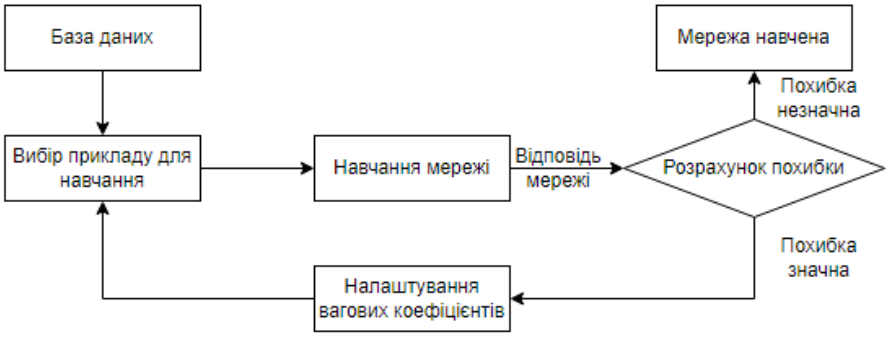


Рисунок 1.8 – Процес навчання нейронної мережі [36]

1.9 Використання нейронних мереж у задачах розпізнавання

Розпізнавання та класифікація об’єктів зображень є поширеною задачею, що має великий обсяг можливих сфер застосування, таких як медицина, економіка, обробка інформації, комп’ютерний зір тощо [37]. Ця задача отримує кращі рішення завдяки використанню штучних нейронних мереж (ШНМ). Цей напрям на сьогодні активно досліджується та відкриває нові можливості.

Штучні нейронні мережі – це обчислювальні математичні моделі, які дозволяють із сукупності вхідних даних отримати необхідні вихідні дані [37]. Вони представляють собою імітацію біологічних процесів у нервовій системі людини. Штучні нейронні мережі були натхнені біологічними нейронними мережами. У нейронній мережі нейрон є математичною функцією, що збирає та

класифікує інформацію відповідно до певної архітектури. Кожен нейрон визначається через його вхід, функцію активації та вихід.

Розрізняють ієрархічні, рекурентні та конкурентні структури штучних нейронних мереж. Проте загалом будова усіх моделей ШНМ подібна. Нейрони розподіляються в мережі по шарах. Нейрони вхідного шару отримують інформацію, потім передають її до нейронів прихованого шару, де виконується основна обробка даних, після чого інформація передається в останній шар – вихідний. Вибір кількості прихованих шарів та нейронів залежить від окремої задачі, на яку націлена конкретна нейронна мережа, обсягу даних та доступних обчислюваних ресурсів [37].

Нейронні мережі що у своєму складі містять більше одного прихованого шару називають глибокими нейронними мережами. Така мережа здатна виконувати більш складні обчислення. При її навчанні у всіх прихованих шарах має використовуватись нелінійна активаційна функція.

Згорткові нейронні мережі (ЗНМ) – це ШНМ такої архітектури, яка побудована на основі багатошарового перцептрона, модифікованого так, аби підготовка вхідних даних була мінімальною [37]. Найчастіше вони застосовуються для розпізнавання та класифікації об’єктів зображень. Згорткова нейронна мережа, як і класичний багатошаровий персептрон, складається з вхідного, вихідного та декількох прихованих шарів, але в ЗНМ приховані шари містять згорткові, агрегувальні, повнозв’язні та шари нормалізації.

Згорткова нейронна мережа є спеціальною архітектурою штучних нейронних мереж, що була запропонована Яном Лекуном. Робота цієї мережі націлена на ефективне розпізнавання образів. На відміну від багатошарового персептрона у згортковій нейронній мережі враховується двомірна топологія зображення, що дозволяє набагато точніше розпізнавати об’єкти на зображеннях. Багатошарові персептрони працюють з векторами, через що для них немає значення розташування точок стосовно одна одної, згорткові мережі ж на противагу працюють саме із зображеннями, що робить їх здатними

виділяти особливості, властиві саме зображенням. Згорткові нейронні мережі є стійкими до незначних зсувів, змін масштабу та поворотів об’єктів на поданих на вхід зображеннях. Порівнюючи згорткові нейронні мережі з мережами прямого поширення у задачах розпізнавання образів, на користь ЗНМ грає те, що вони працюють із зображеннями у вигляді тензорів, а не з даними у вигляді векторів. Використання тензорів для представлення зображень є логічним: кожна матриця тензора відповідає за інтенсивність свого каналу, а сукупність всіх матриць описує все зображення.

Нейрони згорткового шару обробляють лише дані, що розташовуються в їх рецептивному полі. Головною задачею агрегувальних шарів є формування взаємного зв’язку між кількома нейронами попереднього шару і єдиним нейроном поточного шару [37]. Цей шар зменшує розмірність даних та зберігає найбільш важливі характеристики нейронів, виявлені у результаті роботи згорткового шару, тому зазвичай у архітектурі ЗНМ ці два типи шарів чергуються. Повнозв’язні шари, це ті, в яких всі нейрони попереднього шару пов’язані з нейронами наступного шару. Такі шари ускладнюють модель ЗНМ і крім цього можуть призводити до ігнорування виявлених на попередніх шарах ознак, тому зазвичай не використовуються.

Кожен шар мережі відповідає за окрему властивість зображення. Перші шари містять основні ознаки: контури, межі, після обробки яких стає можливим розпізнавання текстур та окремих частин об’єктів. Вихідні ж шари призначені для вміщення результатів класифікації зображення.

Вибір моделі нейронної мережі залежить від задачі, яку необхідно виконати. У даному випадку для розпізнавання пухлин на зображеннях ЗНМ є найкращим вибором завдяки згортковому шару, що дозволяє зменшити обсяг інформації, що зберігається у пам’яті, і крім цього ієрархічно виділити та агрегувати ознаки вхідних даних.

Згорткові нейронні мережі мали значний вплив на удосконалення комп’ютерного зору. CNN є процесом глибокого навчання, який визначає

пріоритетність кількох елементів у зображенні, що дозволяє відрізнити їх один від одного [1].

Глибоке навчання є областю штучного інтелекту, що пов’язана з пошуком набору правил і процедур, які дозволяють машинам діяти та приймати рішення на основі даних, а не бути явно запрограмованими для виконання певного завдання [25,26].

Згорткові нейронні мережі відрізняються кількістю реалізованих шарів, розміром і типом використовуваних методів активації. Змінні CNN емпірично визначаються та емпірично підтверджуються методом проб і помилок [1]. Вони здатні, аналізуючи великий обсяг інформації, виявляти закономірності та приймати рішення за допомогою моделей.

1.10 Набори МРТ зображень головного мозку

Для роботи було вибрано два набори даних, що вміщують МРТ зображення головного мозку.

Перший набір даних був взятий з джерела з відкритим доступом за посиланням: https://www.kaggle.com/datasets/abhranta/brain-tumor-detection-mri. Цей набір даних створений з метою допомогти людям створювати моделі машинного навчання для виявлення пухлин головного мозку, він містить всього 3060 МРТ знімків. Зображення у наборі даних розбиті на три папки: перша містить 1500 МРТ знімків, на яких є пухлина, друга – 1500 знімків мозку без пухлин, а третя містить 60 немаркованих зображень мозку для тестування. На рисунку 1.9 представлені приклади зображень МРТ головного мозку з першого набору даних.

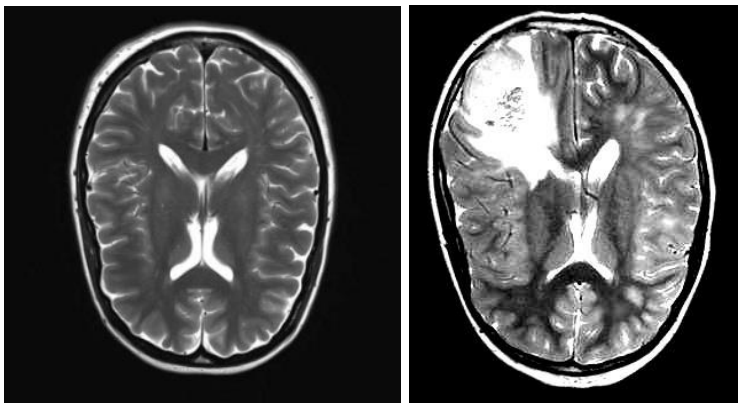


Рисунок 1.9 – Зображення з першого набору даних, зліва зображення мозку без пухлини, справа – мозку з пухлиною

Другий набір даних був взятий з джерела з відкритим доступом за посиланням: https://www.kaggle.com/datasets/mhantor/mri-based-brain-tumor-images. Набір даних містить всього 400 МРТ знімків. Зображення у наборі даних розбиті на дві папки: перша містить 170 МРТ знімків мозку без пухлин, друга – 230 знімків на яких є пухлина. На рисунку 1.10 представлені приклади зображень МРТ головного мозку з другого набору даних.

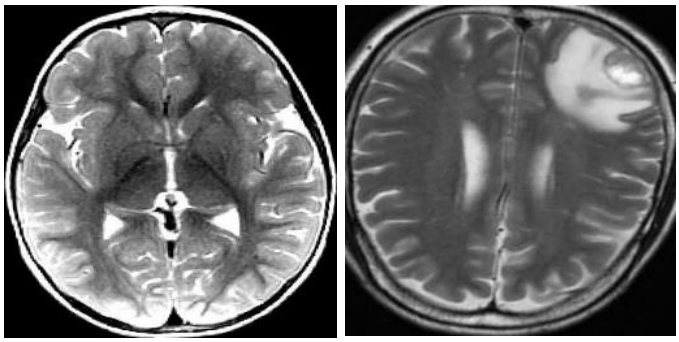


Рисунок 1.10 – Зображення з другого набору даних, зліва зображення мозку без пухлини, справа – мозку з пухлиною

1.11 Постановка задачі розпізнавання пухлин головного мозку на зображеннях МРТ

Тема: нейромережна система розпізнавання пухлин головного мозку на зображеннях МРТ з використанням глибинного навчання.

Мета: розробка та дослідження технологій розпізнавання пухлин головного мозку на основі глибинного навчання нейронних мереж.

Об’єктом дослідження є розпізнавання аномальних утворень на знімках МРТ.

Предмет – нейромережні методи та засоби розпізнавання пухлин головного мозку на знімках МРТ на основі глибинного навчання.

Задача, яка розв’язується у випускній кваліфікаційній роботі – це дослідження методів розпізнавання пухлин на зображеннях МРТ та оптимізацію процесу розпізнавання пухлин шляхом створення моделі, на базі кількох існуючих, з урахуванням якості роботи кожної моделі. Всі створені моделі мають бути оцінені за критеріями, ці оцінки будуть використані для аналізу їх ефективності та порівняння цих показників для різних моделей. Має бути створено нейромережну систему, у результаті роботи якої користувач зможе, надавши на вхід зображення МРТ головного мозку, отримати результат стосовно наявності чи відсутності пухлини у мозку на поданому зображенні.

Висновок

Отже, у даному розділі було проаналізовано проблему розпізнавання пухлин на зображеннях МРТ головного мозку та важливість проведення досліджень, націлених на її вирішення. Сформовано представлення нейрона та розглянуто основні елементи будови головного мозку людини. Визначено поняття розпізнавання образів та задачу класифікації. Визначено поняття МРТ зображення, переваги даного методу візуалізації та його проблеми, зазначено методи обробки МРТ зображень. Виділено особливості розв’язання поставленої задачі, розглянуто її предметну область та існуючі рішення представленої у випускній кваліфікаційній роботі проблеми. Представлено класифікацію нейронних мереж та розглянуто механізм їх навчання. Визначено поняття штучних та згорткових нейронних мереж, виділено переваги використання згорткових нейронних мереж для вирішення задачі розпізнавання. Представлено набори даних, вибрані для роботи та виділено основні завдання створюваної системи.

РОЗДІЛ 2. ПРОЕКТНІ РІШЕННЯ ДЛЯ НЕЙРОМЕРЕЖНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ПУХЛИН ГОЛОВНОГО МОЗКУ НА ЗОБРАЖЕННЯХ МРТ

2.1 Функції нейромережної системи розпізнавання пухлин головного мозку

Для розпізнавання пухлин головного мозку лікар як правило користується власним досвідом, інтуїцією та результатами аналізів. Інколи трапляються такі випадки, коли діагностувати пухлину головного мозку, використовуючи такі фактори, складно. Допомогти в цьому могла б автоматизована система розпізнавання пухлин головного мозку. Звичайно така система не є керівництвом до дій лікаря, а є лише консультативним дорадчим інструментом, який допоможе правильно поставити діагноз. Потрібно виділити основні функції, які має виконувати така система і на яких базуватиметься її структура.

Попередньо було визначено, що створювана нейромережна система має, отримавши на вхід зображення МРТ головного мозку, видати результат про наявність чи відсутність на ньому пухлини.

Далі буде оцінюватись ефективність роботи різних архітектур нейронних мереж для розпізнавання пухлин головного мозку на зображеннях МРТ, після чого буде запропоновано модель, засновану на вже існуючих та розглянутих. У результаті буде отримано можливість оцінити, чи покращить поєднання кількох моделей якість розпізнавання МРТ зображень.

Приведемо функції, що має виконувати створювана нейромережна система:

- забезпечити попередню обробку зображень з набору даних;
- побудувати моделі нейронних мереж з використанням вибраного набору даних;
- забезпечити збереження побудованих моделей;
- виводити матрицю плутанини у результаті виконання прогнозу;

– оцінити якість роботи побудованих моделей за такими метриками: accuracy, precision, recall та значенням функції втрат;
– точність результатів на тестовій вибірці має перевищувати 95%;
– реалізувати прогнозування за використанням кількох моделей, поєднуючи їх прогнози у рівному співвідношенні та із урахуванням оцінки точності;
– отримавши на вхід зображення МРТ головного мозку, видати результат про наявність чи відсутність на ньому пухлини;
– реалізовувати збереження зображень до набору даних.

На рисунку 2.1 зображена загальна схема процесу навчання та оцінювання моделей нейронних мереж.

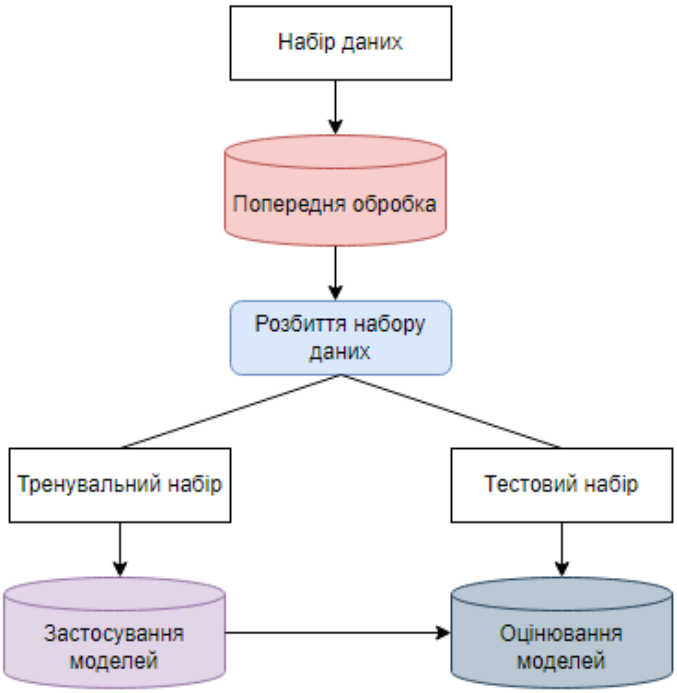


Рисунок 2.1 – Загальна схема процесу навчання та оцінювання моделей нейронних мереж

2.2 Аналіз моделей згорткових нейронних мереж, використаних у дослідженні

Згорткові нейронні мережі стали домінуючим підходом машинного навчання для візуального розпізнавання об’єктів, тож для роботи було вибрано попередньо навчені моделі саме згорткових нейронних мереж.

2.2.1 Модель MobileNet

Виконаємо аналіз однієї з моделей згорткових нейронних мереж, а саме моделі MobileNet [38].

MobileNet – це клас моделей згорткових нейронних мереж для програм комп’ютерного зору. MobileNet є моделлю комп’ютерного зору з відкритим вихідним кодом від Google, і призначена для навчання класифікаторів. MobileNet є першою мобільною моделлю комп’ютерного зору Tensorflow. Класифікація зображень є найбільш поширеним застосуванням для цих моделей.

MobileNets базуються на спрощеній архітектурі, яка використовує згортки, що розділяються по глибині, щоб значно зменшити кількість параметрів порівняно з іншими мережами з регулярними згортками та такою ж глибиною мереж. Це призводить до створення легких глибоких нейронних мереж.

Модель MobileNet особлива тим, що має меншу обчислювальну потужність для запуску або застосування трансферного навчання. Це робить модель ідеальною для мобільних пристроїв, вбудованих систем і комп’ютерів без графічного процесора або низької обчислювальної ефективності зі значним компромісом із точністю результатів. Крім цього MobileNet найкраще підходить для веб-браузерів, оскільки браузери мають обмеження щодо обчислень, обробки графіки та зберігання.

– Архітектура MobileNet.

Модель MobileNet базується на згортках, які розділяються по глибині, що є формою факторизованих згорток, які розкладають стандартну згортку на

згортку по глибині та згортку 1×1, яка називається поточковою згорткою. Для MobileNets згортка по глибині застосовує один фільтр до кожного вхідного каналу. Потім поточкова згортка застосовує згортку 1×1, щоб об'єднати виходи згортки по глибині. Стандартна згортка фільтрує та поєднує вхідні дані в новий набір виходів за один крок. Згортка, що розділяється по глибині, розділяє це на два шари: окремий шар для фільтрації та окремий шар для об'єднання. Ця факторизація має ефект різкого зменшення обчислень і розміру моделі.

Стандартний згортковий шар приймає як вхідні дані $D_F \times D_F \times M$ карту ознак F і створює $D_F \times D_F \times N$ карту ознак G, де D_F — просторова ширина та висота квадратної вхідної карти ознак, M — кількість вхідних каналів (вхідна глибина), D_G — просторова ширина та висота квадратної вихідної карти ознак, а N — номер вихідного каналу (вихідна глибина).

Стандартний згортковий шар параметризовано ядром згортки K розміром $D_K \times D_K \times M \times N$, де D_K – просторовий розмір ядра, що вважається квадратним, M — кількість вхідних каналів, а N — кількість вихідних каналів, як визначено раніше.

Вихідна карта ознак для стандартної згортки за умови першого кроку та відступу обчислюється як:

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m} \cdot$$

Обчислювальна вартість стандартних згорток становить:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F, \qquad (2.2)$$

де обчислювальна вартість мультиплікативно залежить від кількості вхідних каналів M, кількості вихідних каналів N, розміру ядра $D_k \times D_k$ і розміру карти ознак $D_F \times D_F$.

MobileNet використовує згортки, що розділяються по глибині, щоб розірвати взаємодію між кількістю вихідних каналів і розміром ядра.

Стандартна операція згортки має ефект фільтрації ознак на основі згорткових ядер і їх комбінування для створення нового представлення. Етапи фільтрації та комбінування можуть бути розділені на два етапи за допомогою

факторизованих згорток (згортка, що розділяється по глибині), для істотного зниження витрат на обчислення.

Згортка, що розділяється по глибині, складається з двох шарів: згортки по глибині та поточкової згортки. Згортка по глибині застосовується, щоб застосувати один фільтр для кожного вхідного каналу. Поточкова згортка використовується для створення лінійної комбінації результату шару по глибині. MobileNets використовують як batchnorm (пакетна нормалізація), так і ReLU нелінійності для обох шарів. Пакетна нормалізація — це шар мережі, який вставляється між прихованим шаром і наступним прихованим шаром. Його робота полягає в тому, щоб отримати вихідні дані з першого прихованого шару та нормалізувати їх перед тим, як передати їх як вхідні дані наступного прихованого шару.

Згортка по глибині з одним фільтром на вхідний канал може бути записана так [38]:

$$G_{k,l,m} = \sum_{i,j} \widehat{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m},$$

де $\widehat{K}$ є глибинним згортковим ядром розміру $D_K \times D_K \times M$, де m-й фільтр у $\widehat{K}$ застосовується до m-го каналу у F для отримання m-го каналу відфільтрованої вихідної карти ознак $\widehat{G}$.

Згортка по глибині має таку обчислювальну вартість:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F. \quad (2.4)$$

Поглиблена згортка надзвичайно ефективна порівняно зі стандартною згорткою, але вона лише фільтрує вхідні канали, а не поєднує їх для створення нових ознак. Що викликає необхідність у додатковому шарі для генерації нових, який обчислює лінійну комбінацію результату згортки по глибині через згортку 1 × 1. Комбінація згортки по глибині та поточкової згортки називається роздільною згорткою по глибині.

Вартість згорток, що розділяються по глибині є сумою глибинної та поточкової згорток:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F. \quad (2.5)$$

Виражаючи згортку як двоетапний процес фільтрації та об’єднання, отримується скорочення обчислень:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}.$$

MobileNet використовує згортки 3 × 3 із розділенням по глибині, які використовують у 8-9 разів менше обчислень, ніж стандартні згортки, лише з невеликим зниженням точності.

– Структура мережі MobileNet та навчання.

Структура MobileNet побудована на згортках, що розділяються по глибині, за винятком першого рівня, який є повною згорткою. Визначаючи мережу такими простими термінами, можна легко досліджувати мережеві топології, щоб знайти хорошу мережу. Архітектура MobileNet зображена на рисунку 2.2, а її шари перелічені на рисунку 2.3.

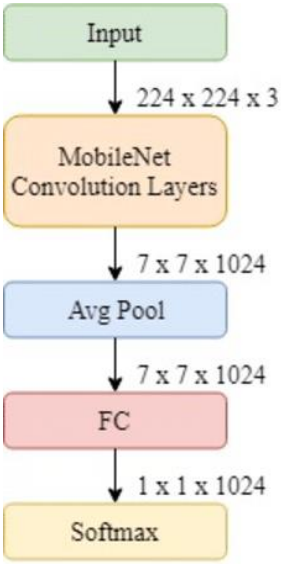


Рисунок 2.2 – Архітектура оригінальної MobileNet [39]

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Рисунок 2.3 – Шари архітектури MobileNet [40]

Усі шари супроводжуються пакетною нормалізацією і нелінійністю ReLU, за винятком остаточного повністю зв’язаного шару, який не має нелінійності та подається на рівень softmax для класифікації.

Моделі MobileNet були навчені в TensorFlow за допомогою RMSprop з асинхронним градієнтним спуском. На відміну від навчання великих моделей, у MobileNet використовується менше методів регуляризації та розширення даних, оскільки маленькі моделі мають менше проблем із перенавчанням.

Архітектура MobileNet V1 починається зі звичайної згортки 3×3, а потім – 13 згорткових блоків, які розділяються по глибині. У MobileNet V2 кожен блок містить шар розширення 1 x 1 на додаток до згорткових шарів по глибині та поточкових. На відміну від V1, поточковий згортковий шар V2 проектує дані з великою кількістю каналів у тензор із значно меншою кількістю каналів. Згортковий шар розширення 1 x 1 розширить кількість каналів залежно від коефіцієнта розширення в даних, перш ніж він перейде до згортки по глибині. Друга нова річ у будівельному блоці MobileNet V2 — це залишкове з’єднання. Залишкове з’єднання існує, щоб допомогти потоку градієнтів через мережу. MobileNet V2 все ще використовує глибоку роздільну згортку, але тепер вона

має залишковий блок вузького місця замість просто глибокого роздільного блоку згортки. Кожен рівень MobileNet V2 має пакетну нормалізацію та ReLU6 як функцію активації. Однак вихід проекційного шару не має функції активації [41]. Повна архітектура MobileNet V2 складається з 17 вузьких залишкових блоків у рядку, за якими слідує регулярна згортка 1×1, шар глобального середнього об'єднання та шар класифікації.

2.2.2 Модель DenseNet

Виконаємо аналіз однієї з моделей згорткових нейронних мереж, а саме моделі DenseNet [42].

DenseNet пропонує новий режим підключення, з'єднуючи кожен поточний шар мережі з попередніми шарами, так що поточний шар може приймати вихідні карти ознак усіх попередніх шарів як вхідні ознаки. Оскільки кожен шар пов'язаний з усіма попередніми шарами, попередні ознаки можна неодноразово використовувати для створення більшої кількості карт ознак із меншим ядром згортки.

DenseNet використовує щільні блоки як базові одиничні модулі, як показано на рисунку 2.4.

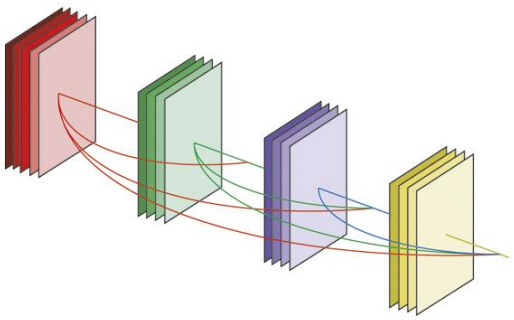


Рисунок 2.4 – Структура DenseNet [43]

На рисунку 2.4 щільна блокова структура складається з 4 щільно з'єднаних шарів зі швидкістю зростання 4. Кожен шар у цій структурі приймає вихідні карти ознак попередніх шарів як вхідні карти ознак [43]. Щільний блок передає карти ознак на всі наступні шари, додаючи розмірність карт ознак. DenseNet використовує швидкість зростання гіперпараметрів для контролю кількості каналів карти ознак у мережі.

Шари DenseNet є дуже вузькими і додають лише невеликий набір карт ознак до «колективних знань» мережі, а інші карти залишаються незмінними [42]. Остаточний класифікатор приймає рішення на основі всіх карт ознак в мережі.

Окрім кращої ефективності параметрів, великою перевагою DenseNets є покращений потік інформації та градієнти по всій мережі, що полегшує їх навчання. Кожен шар має прямий доступ до градієнтів від функції втрат і оригінального вхідного сигналу, що веде до неявного глибокого контролю. Це допомагає навчанню глибших мережевих архітектур. Крім цього, щільні з'єднання мають регуляризуючий ефект, який зменшує перенавчання завдань із меншим розміром навчального набору.

Замість того, щоб черпати репрезентативну потужність із надзвичайно глибоких або широких архітектур, DenseNets використовують потенціал мережі шляхом повторного використання ознак, створюючи стислі моделі, які легко навчати та котрі мають високу ефективність параметрів [42]. Об'єднання карт ознак, отриманих різними шарами, збільшує варіації вхідних даних наступних шарів і покращує ефективність. Це головна відмінність між DenseNets і ResNets. Мережі Inception також об'єднують ознаки з різних шарів, але DenseNets простіші та ефективніші.

Наведемо приклад [42], де одне зображення x_0, проходить через згорткову мережу. Мережа складається з L шарів, кожен з яких реалізує нелінійне перетворення $H_\ell(\cdot)$, де ℓ індексує шар. $H_\ell(\cdot)$, може бути складовою функцією таких операцій, як пакетна нормалізація, ReLU, об'єднання або згортка. Вихід ℓ-го шару позначають як x_ℓ.

– ResNets.

Традиційні згорткові мережі прямого зв'язку з'єднують вихід 1-го шару як вхідний сигнал із $(\ell + 1)$-м рівнем, що призводить до наступного переходу шару: $x_\ell = H_\ell(x_{\ell-1})$. ResNets додають skip-connection, що обходить нелінійні перетворення за допомогою функції ідентифікації:

$$x_\ell = H_\ell(x_{\ell-1}) + x_{\ell-1}.$$

Перевагою ResNets є те, що градієнт може проходити безпосередньо через функцію ідентичності з наступних шарів на попередні. Однак функція ідентифікації та вихід H_ℓ об'єднуються шляхом підсумовування, що може перешкоджати потоку інформації в мережі.

– Щільне підключення.

Щоб ще більше покращити потік інформації між шарами, запропоновано інший шаблон з'єднання: прямі з'єднання від будь-якого шару до всіх наступних шарів [42]. Отже, ℓ-й шар отримує карти ознак усіх попередніх шарів, $x_0, \ldots, x_{\ell-1}$, як вхідні дані:

$$x_\ell = H_\ell([x_0, \ldots, x_{\ell-1}]),$$

де $[x_0, \ldots, x_{\ell-1}]$ відноситься до конкатенації карт ознак, створених у шарах $0, \ldots, \ell - 1$. Через щільне підключення таку мережеву архітектуру називають щільною згортковою мережею (DenseNet). Для простоти реалізації численні входи $H_\ell(\cdot)$ об'єднують в один тензор.

$H_\ell(\cdot)$ визначають як композиційну функцію трьох послідовних операцій [42]: пакетна нормалізація, за якою слідує ReLU і згортка 3 × 3.

– Шари об'єднання.

Важливою частиною згорткових мереж є шари зменшення дискретизації, які змінюють розмір карт ознак. Щоб полегшити зменшення дискретизації в архітектурі, мережу ділять на кілька щільно з'єднаних щільних блоків. Шари між блоками називають перехідними шарами, які виконують згортку та об'єднання.

– Швидкість зростання.

Якщо кожна функція H_ℓ виробляє k карт ознак, з цього випливає, що ℓ-й шар має $k_0 + k \times (\ell - 1)$ вхідних карт ознак, де k_0 є кількістю каналів у вхідному шарі. Важливою відмінністю між DenseNet та існуючими мережевими архітектурами є те, що DenseNet може мати дуже вузькі шари, наприклад, k = 12. Гіперпараметр k називають швидкістю зростання мережі. Карти ознак можна розглядати як глобальний стан мережі. Кожен шар додає до цього стану

k власних карт ознак. Швидкість зростання регулює, скільки нової інформації кожен шар вносить у глобальний стан. Глобальний стан, коли він записаний, може бути доступний з будь-якого місця в мережі, і, на відміну від традиційних мережевих архітектур, немає необхідності копіювати його з шару на шар.

– Вузькі шари.

Хоча кожен шар виробляє лише k вихідних карт ознак, зазвичай він має набагато більше вхідних даних. Згортка 1×1 може бути введена як вузький шар перед кожною згорткою 3×3, щоб зменшити кількість вхідних карт ознак і, таким чином, підвищити ефективність обчислень.

– Компресія.

Щоб ще більше покращити компактність моделі, можна зменшити кількість карт ознак на перехідних шарах. Якщо щільний блок містить m карт ознак, наступному перехідному шару дозволяється генерувати [θm] вихідні карти ознак, де $0 < \theta \leq 1$ називають коефіцієнтом стиснення. Коли $\theta = 1$, кількість карт ознак на перехідних шарах залишається незмінною.

DenseNet вводить прямі зв’язки між будь-якими двома шарами з однаковим розміром карти ознак. DenseNets природним чином масштабуються до сотень шарів, не виявляючи труднощів з оптимізацією. DenseNets вимагають значно менше параметрів і менше обчислень для досягнення найсучаснішої продуктивності. Завдяки своїм компактним внутрішнім представленням і зменшеній надлишковості ознак, DenseNets можуть бути хорошими екстракторами ознак для різних завдань комп’ютерного зору.

2.2.3 Ансамблеве моделювання

Один алгоритм може не зробити ідеального прогнозу для певного набору даних. Алгоритми машинного навчання мають свої обмеження, тому досягнення високої точності моделлю є складним завданням. Створення кількох моделей та їх поєднання дає шанс підвищити загальну точність. Результати різних моделей агрегуються з метою зменшення похибки та підтримання узагальненості моделі.

Ансамбль означає групу елементів, які розглядаються як ціле, а не окремо [44].

Ансамблеве моделювання — це процес, у якому створюється кілька різноманітних моделей для прогнозування результату за допомогою багатьох різних алгоритмів моделювання або різних навчальних наборів даних [45]. Після чого модель ансамблю агрегує прогнози кожної базової моделі та призводить до остаточного прогнозу. Метою використання ансамблевих моделей є зменшення помилки узагальнення прогнозу. Хоча модель ансамблю містить кілька базових моделей усередині моделі, вона діє та працює як єдина модель. Методи ансамблю допомагають покращити надійність/узагальненість моделі.

Основна концепція [46], що лежить в основі ансамблевого навчання, полягає в тому, щоб пом’якшити помилки або упередження, які можуть існувати в окремих моделях, шляхом використання колективного розуму багатьох моделей, що зрештою призводить до більш точного прогнозування.

Базові учні — це перший рівень архітектури ансамблевого навчання, і кожен з них навчений робити індивідуальні прогнози [47].

Методи ансамблю поділяються на дві широкі категорії: техніки послідовного ансамблю та техніки паралельного ансамблю. Методи послідовного ансамблю генерують базових учнів у послідовності, що сприяє зміцненню залежності між ними. У техніках паралельного ансамблю базові учні генеруються в паралельному форматі. Ці методи використовуються для заохочення незалежності між базовими учнями.

Більшість методів ансамблю застосовують єдиний алгоритм базового навчання, що призводить до однорідності всіх базових учнів. Однорідні базові учні – це базові учні одного типу з подібними якостями [48].

Основні методи ансамблю [46]:

– Максимальне голосування. Метод полягає в створенні кількох моделей незалежно та отриманні їх індивідуального результату під назвою

«голосування». Клас із максимальною кількістю голосів повертається як результат;

– Метод усереднення. Цей метод складається з незалежної побудови кількох моделей і повернення середнього прогнозу всіх моделей. Загалом сукупний вихід кращий, ніж окремий, оскільки дисперсія зменшується;
– Середнє зважене – це розширення методу усереднення. Усім моделям призначаються різні ваги, що визначає важливість кожної моделі для прогнозування.

Ансамблеві методи ідеально підходять для зменшення дисперсії в моделях, тим самим підвищуючи точність прогнозів. Ансамбль моделей поєднує різні моделі, з метою отримання у результаті прогнозу, що є найкращим із можливих на основі розгляду всіх прогнозів.

2.3 Програмні технології створення нейромережних систем

Програма, представлена у даній роботі, написана мовою програмування Python. Python — інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Python підтримує модулі та пакети, що сприяє модульності програми та повторному використанню коду. Інтерпретатор Python і велика стандартна бібліотека доступні у вихідному або двійковому вигляді безкоштовно для всіх основних платформ і можуть вільно поширюватися. Для роботи використовувалась версія Python 3.7.12.

TensorFlow — це бібліотека з відкритим вихідним кодом, розроблена Google Brain Team. Це популярний фреймворк машинного та глибокого навчання.

Програмне забезпечення TensorFlow обробляє набори даних, які згруповані як обчислювальні вузли у формі графа. Ребра, які з’єднують вузли в графі, можуть представляти багатовимірні вектори або матриці, створюючи так звані тензори.

Архітектура Tensorflow працює в три важливі етапи:

- попередня обробка даних – структурування даних та їх приведення до одного граничного значення;
- побудова моделі;
- навчання та оцінка моделі – використання даних для навчання моделі та її перевірки на невідомих даних.

Програми TensorFlow можуть працювати як на звичайних процесорах, так і на високопродуктивних графічних процесорах (GPU), а також на власних тензорних процесорах (TPU) Google.

Keras — це прикладний програмний інтерфейс глибокого навчання, написаний на Python, який працює на платформі машинного навчання TensorFlow.

Шари є основними будівельними блоками нейронних мереж у Keras. Кожен шар отримує вхідну інформацію, виконує деякі обчислення і, нарешті, виводить перетворену інформацію. Вихід одного шару перетікає в наступний шар як його вхід. Для шару Keras потрібна форма входу, щоб зрозуміти структуру вхідних даних, ініціалізатор, щоб встановити вагу для кожного входу, і активатори, щоб перетворити вихід, щоб зробити його нелінійним. Обмеження вказують діапазон, у якому вага вхідних даних, що будуть згенеровані, і регуляризатор намагатиметься оптимізувати шар, динамічно застосовуючи штрафи до ваг під час процесу оптимізації.

Для створення повного шару Keras необхідні:

- форма вхідних даних;
- кількість нейронів у шарі;
- ініціалізатори;
- регулятори;
- обмеження;
- активації.

Запуск програми виконується засобами Kaggle Notebooks, хмарного обчислювального середовища, яке забезпечує відтворюваний та спільний аналіз. Kaggle Notebooks працюють у віддаленому обчислювальному середовищі. Програма запускалась на графічному процесорі (GPU). Kaggle Notebooks забезпечує 20 гігабайт автоматично збереженого місця на диску. Технічні характеристики GPU P100:

– 1 графічний процесор Nvidia Tesla P100;

– 2 ядра ЦП;

– 13 гігабайт оперативної пам'яті.

Для розробки веб-додатка, що дозволить користувачу проводити розпізнавання зображень МРТ головного мозку, використовуючи навчену модель, було використано наступні технології.

Flask – це невеликий і легкий фреймворк, написаний мовою Python, що пропонує корисні інструменти та функції для полегшення процесу створення веб-застосунків з використанням Python.

Bootstrap – це відкритий та безкоштовний HTML, CSS та JavaScript фреймворк, який використовується веб-розробниками для створення дизайнів сайтів та веб-додатків.

HTML (HyperText Markup Language) – мова гіпертекстової розмітки, що дозволяє повідомляти браузеру, як і який текст та інші елементи розміщувати на веб-сторінці [49]. Також ця мова дозволяє будувати взаємозв'язки між сторінками за допомогою посилань.

CSS (Cascading Style Sheets) – це формальна мова опису зовнішнього вигляду документа (веб-сторінки), написаного з використанням мови розмітки (найчастіше HTML, як і у нашому випадку) [49].

SQL – мова програмування, що фактично є мовою запитів, що використовується для управління або маніпуляції даними в базах даних реляційного типу [50]. Завдяки використанню цієї універсальної мови запитів можливо швидко і легко працювати з великими обсягами інформації.

MySQL – реляційна система управління базами даних.

XAMPP — це безкоштовний міжплатформний веб-сервер із відкритим кодом. На локальному комп'ютері серверне програмне забезпечення XAMPP забезпечує відповідне середовище для тестування проектів MySQL, PHP, Apache та Perl.

2.4 Метрики оцінки моделей

Модель оцінюється на тестовому наборі даних.

Матриця плутанини — це таблиця, що використовується для опису ефективності моделі класифікації. Нижче представлені пояснення складових матриці плутанини [14].

TP (True Positives) – кількість вірно класифікованих позитивних прикладів (істинно позитивні випадки).

TN (True Negatives) – кількість вірно класифікованих негативних прикладів (істинно негативні випадки).

FP (False Positives) – кількість негативних прикладів, що класифіковані як позитивні. Це «помилкове виявлення», тому що при відсутності події помилково виноситься рішення про її присутність (хибно позитивні випадки).

FN (False Negatives) – кількість позитивних прикладів, класифікованих як негативні. Це так званий «помилковий пропуск», коли подія, що нас цікавить, помилково не виявляється (хибно негативні приклади).

Що є позитивною подією, а що – негативною, залежить від конкретної задачі [14]. На головній діагоналі матриці плутанини показана кількість правильно класифікованих прикладів, на побічній діагоналі – кількість неправильно класифікованих прикладів.

Ефективність моделей оцінюється за допомогою наступних оціночних метрик:

Accuracy (точність): відношення випадків правильних результатів до всіх результатів.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

Precision (влучність): відношення кількості істинно позитивних результатів до кількості всіх позитивних результатів (як хибних, так і істинних). Тобто, це відношення кількості випадків, які були віднесені до певного класу правильно, до загальної кількості випадків, що були віднесені до цього класу.

$$Precision = \frac{TP}{TP + FP}.$$

Recall (повнота): співвідношення кількості істинно позитивних результатів до суми істинно позитивних і хибно негативних. Тобто, це відношення кількості випадків, які були віднесені до певного класу правильно, до загальної кількості випадків, що були віднесені до іншого класу.

$$Recall = \frac{TP}{TN + FN}.$$

Крім зазначених вище показників, розраховується також значення функції втрат. Ці функції повідомляють, наскільки прогнозований результат моделі відрізняється від фактичного результату. У даній роботі як функція втрат використана перехресна ентропія (крос-ентропія). Втрата перехресної ентропії використовується як мета оптимізації під час навчання для коригування параметрів моделі. У випадку, якщо прогнозована ймовірність класу значно відрізняється від фактичної мітки класу (0 або 1), значення втрати крос-ентропії є високим. У випадку, якщо прогнозована ймовірність класу близька до мітки класу (0 або 1), втрата крос-ентропії буде меншою.

$$L_{log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p)),$$

де $y \in \{0,1\}$ – фактичні мітки, p – прогнозовані ймовірності.

2.5 Архітектура створюваної нейромережної системи

Ми передбачаємо наступні етапи застосування нейронних мереж до набору даних пухлин головного мозку:

1. Імпорт необхідних бібліотек.
2. Зчитування зображень, надання їм міток і збереження у фреймі даних.
3. Зміна розміру зображень на 128x128.

4. Нормалізація зображень.

5. Розбиття набору даних на набори для навчання та тестування.

6. Створення моделі.

7. Компіляція моделі.

8. Застосування моделі до навчального набору.

9. Оцінка моделі на тестовому наборі.

На рисунку 2.5 зображено алгоритм роботи створеної системи розпізнавання пухлин на зображеннях МРТ головного мозку із застосуванням нейронних мереж.

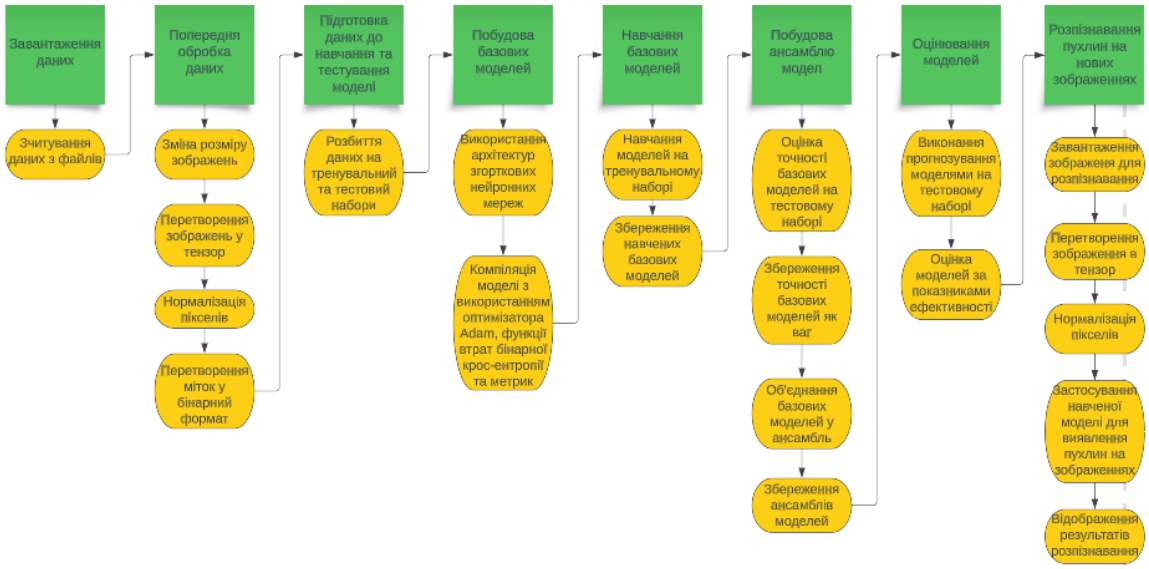


Рисунок 2.5 – Структура створеної системи розпізнавання пухлин на зображеннях МРТ головного мозку з використанням нейронних мереж

У вигляді чорної скриньки модуль навчання нейронних мереж представлено на рисунку 2.6.

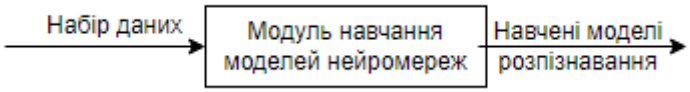


Рисунок 2.6 – Модуль навчання нейронних мереж у вигляді чорної скриньки

Визначимо поняття, що фігурують у навчанні нейронних мереж:

– епоха – прямий і зворотний прохід по вибраним для навчання мережі тренувальним даним;

– розмір серії (batch) – фіксована кількість визначених для тренування мережі наборів даних для однієї ітерації;
– кількість ітерацій – це кількість проходів, що використовують тренувальні дані. Один прохід дорівнює прямому та зворотному проходу.

2.5.1 MobileNetV2

Попередньо навчена модель MobileNetV2 імпортується без підключення повнозв'язного шару у верхній частині мережі. Використовуються ваги попередньо навчені на базі даних зображень ImageNet. До імпортованої моделі додаються нові шари:

– шар GlobalAveragePooling2D застосовує глобальне середнє об'єднання до просторових розмірів, поки кожен просторовий розмір не стане одним, а інші розміри залишає незмінними. Він генерує одну карту ознак для кожної відповідної категорії завдання класифікації в останньому згортковому шарі. Замість того, щоб додавати повнозв'язані шари поверх карт ознак, береться середнє значення кожної карти ознак;
– шар Dense є звичайним щільно зв'язаним шаром нейронної мережі. Це найпоширеніший і часто використовуваний шар. Щільний шар виконує операцію: output = activation(dot(input, kernel) + bias), на вхідних даних і повертає вихідні дані, де input представляє вхідні дані, kernel представляє дані ваги, dot представляє numpy скалярний добуток усіх вхідних даних і відповідних ваг, bias представляє зміщене значення, яке використовується в машинному навчанні для оптимізації моделі, activation представляє функцію активації. Розмірність вихідного простору рівна 64. Застосовано функцію активації ReLU. Перевагою використання ReLU як функції активації є те, що за рахунок активації лише певної кількості нейронів, функція ReLU є набагато більш ефективною в обчислювальному відношенні порівняно з сигмовидною функцією і функцією тангенсу [14];

– шар Dropout випадковим чином встановлює вхідні одиниці на 0 із частотою rate на кожному кроці під час навчання, тобто випадково вибрані нейрони ігноруються, що допомагає запобігти перенавчанню. Значення rate встановлене рівним 0,2;
– шар Dense із розмірністю вихідного простору 1 та використанням сигмовидної функції активації.

При компіляції моделі використовується оптимізатор Адам. Оптимізатор – це алгоритм, який використовується для незначної зміни параметрів, таких як ваги та швидкість навчання, щоб модель працювала правильно та швидко. Алгоритм оптимізації Адам є розширенням стохастичного градієнтного спуску. Останнім часом він набув широкого поширення для додатків глибокого навчання в галузі комп'ютерного зору. Алгоритм поєднує переваги двох інших розширень стохастичного градієнтного спуску: адаптивного градієнтного алгоритму (AdaGrad) і середньоквадратичне поширення (RMSProp). Замість адаптації швидкості навчання параметрів на основі середнього першого моменту, як у RMSProp, Адам також використовує середнє значення других моментів градієнтів. Правило оновлення ваг для Адам визначається на основі використання оцінок двох різних моментів, у першому з яких використовуються обчислені раніше значення часткових похідних (як у методі моментів), а у другому їх квадрати (як у RMSProp) [51]. Метод дійсно ефективний при роботі з великою проблемою, що включає велику кількість даних або параметрів. Він потребує менше пам'яті та є ефективним.

Як функція втрат використовувалась бінарна крос-ентропія, а як метрики задані метрики, зазначені у підрозділі 2.4.

На рисунку 2.7 відображені параметри, використані для навчання моделі.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_128 (Funct  (None, 4, 4, 1280)       2257984
 ional)

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 64)                81984

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 2,340,033
Trainable params: 82,049
Non-trainable params: 2,257,984
```

Рисунок 2.7 – Опис використаної для навчання моделі MobileNetV2

З рисунку можна побачити назву та тип усіх шарів у моделі, вихідну форму для кожного шару, кількість вагових параметрів кожного шару та загальну кількість тренованих і нетренованих параметрів моделі.

2.5.2 DenseNet121

Попередньо навчена модель DenseNet121 імпортується без підключення повнозв'язного шару у верхній частині мережі. Використовуються ваги попередньо навчені на базі даних зображень ImageNet, як і для попередньо описаної моделі. Як функція активації для використання на «верхньому» шарі застосовано softmax. Softmax — це математична функція, яка перетворює вектор чисел у вектор ймовірностей, де ймовірності кожного значення пропорційні відносному масштабу кожного значення у векторі. Функція softmax використовується для нормалізації виходів, перетворюючи їх із зважених сумарних значень у ймовірності, сума яких дорівнює одиниці. До імпортованої моделі додаються нові шари, які додавались у попередній моделі. Параметри компіляції було використано ті ж, що і для нашої моделі MobileNetV2 (пункт 2.5.1).

На рисунку 2.8 відображені параметри, використані для навчання моделі.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 4, 4, 1024)        7037504

 global_average_pooling2d_1   (None, 1024)            0
 (GlobalAveragePooling2D)

 dense_2 (Dense)             (None, 64)                65600

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 7,103,169
Trainable params: 65,665
Non-trainable params: 7,037,504
```

Рисунок 2.8 – Опис використаної для навчання моделі DenseNet121

2.5.3 Власна модель згорткової нейронної мережі

У моделі ми використали алгоритм згорткової нейронної мережі Depthwise Separable (роздільна згортка по глибині), який є кращим, ніж звичайна згорткова нейронна мережа, оскільки він має більше фільтрів для виявлення шаблонів у магнітно-резонансному зображенні. Алгоритм базується на спрощеній архітектурі для створення легкої глибокої нейронної мережі.

Власна глибинна модель згорткової нейронної мережі включає такі шари:

– вхідний шар;
– два шари Conv2D – шари 2D згортки. Цей шар створює ядро згортки, яке згортається із входом шару для отримання тензора виходів. Кількість вихідних фільтрів у згортці – 64, висота та ширина вікна двовимірної згортки – 3, використана функція активації ReLU, аргумент padding приймає значення «same» та призводить до рівномірного доповнення нулями;
– шар MaxPooling2D – операція максимального об’єднання для двовимірних просторових даних. Слугує для зменшення дискретизації вхідного сигналу вздовж його просторових розмірів, беручи максимальне значення у вікні введення для кожного каналу вхідного сигналу. Вікно зсувається кроками вздовж кожного виміру.

Використовується розмір вікна 2x2. Шари об’єднання використовуються для зменшення розмірів карт ознак. Шар об’єднання підсумовує ознаки, присутні в діапазоні карти ознак, згенерованої згортковим шаром, тому подальші операції виконуються над підсумковими ознаками. Це робить модель більш потужною, щоб розрізняти положення ознак на вхідному зображенні;

- два шари SeparableConv2D – двовимірна згортка, що розділяється по глибині. Роздільні згортки складаються з виконання спочатку просторової згортки по глибині (яка діє на кожен вхідний канал окремо), а потім поточкової згортки, яка змішує отримані вихідні канали. Кількість вихідних фільтрів у згортці – 128, висота та ширина вікна двовимірної згортки – 3, використана функція активації ReLU, аргумент padding приймає значення «same»;
- шар MaxPooling2D з розміром вікна 2x2.
- шар SeparableConv2D з кількістю вихідних фільтрів у згортці – 256, розміром вікна двовимірної згортки 3x3, з використанням функції активації ReLU тиа значенням «same» аргумента padding;
- шар BatchNormalization – шар, який нормалізує свої вхідні дані. Пакетна нормалізація застосовує перетворення, яке підтримує середнє значення виходу близько до 0 і стандартне відхилення виходу близького до 1;
- шар SeparableConv2D з кількістю вихідних фільтрів у згортці – 256, розміром вікна двовимірної згортки 3x3, з використанням функції активації ReLU та значенням «same» аргумента padding;
- шар BatchNormalization;
- шар MaxPooling2D з розміром вікна 2x2;
- шар GlobalAveragePooling2D – середнє глобальне об'єднання просторових даних;
- шар Dropout із значенням rate рівним 0,2;

– шар Dense з розимірністю вихідного простору 64 та застосуванням функції активації ReLU;
– шар Dropout із значенням rate рівним 0,2;
– шар Dense із розмірністю вихідного простору 1 та використанням сигмовидної функції активації.

Параметри компіляції було використано ті ж, що і для двох попередньо описаних моделей.

На рисунку 2.9 відображені параметри, використані для навчання власної моделі.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv1_1 (Conv2D)            (None, 128, 128, 64)      1792

 Conv1_2 (Conv2D)            (None, 128, 128, 64)      36928

 MaxPooling1 (MaxPooling2D)  (None, 64, 64, 64)        0

 Conv2_1 (SeparableConv2D)   (None, 64, 64, 128)       8896

 Conv2_2 (SeparableConv2D)   (None, 64, 64, 128)       17664

 MaxPooling2 (MaxPooling2D)  (None, 32, 32, 128)       0

 Conv3_1 (SeparableConv2D)   (None, 32, 32, 256)       34176

 Batch_normalization3_1 (Bat  (None, 32, 32, 256)      1024
 chNormalization)

 Conv3_2 (SeparableConv2D)   (None, 32, 32, 256)       68896

 Batch_normalization3_2 (Bat  (None, 32, 32, 256)      1024
 chNormalization)

 MaxPooling3 (MaxPooling2D)  (None, 16, 16, 256)       0

 global_average_pooling2d_2   (None, 256)              0
 (GlobalAveragePooling2D)

 dropout_2 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 64)                16448

 dropout_3 (Dropout)         (None, 64)                0

 dense_5 (Dense)             (None, 1)                 65

=================================================================
Total params: 186,113
Trainable params: 185,089
Non-trainable params: 1,024
```

Рисунок 2.9 – Опис використаної для навчання власної моделі згорткових нейронних мереж

2.5.4 Параметри навчання

Для тренування власної моделі задано 100 епох, для попередньо навчених – 25 епох. Також для всіх моделей задано попередню зупинку, для власної моделі тренування зупиняється після 10-ти епох без покращення результатів, для попередньо навчених – після 2-х епох без покращення. Значення аргумента batch_size, що представляє кількість вибірок на одне оновлення градієнта, встановлено 64. Набір даних було розбито у відношенні 80% для тренування і 20% для тестування.

2.5.5 Виконання прогнозу

Отримані при виконанні прогнозування на тестовому наборі значення для всіх методів поділялись на два класи: якщо прогнозоване значення було більшим за 0.5, то зображення отримувало мітку 1, якщо меншим – відповідно мітку 0. Мітка «0» відповідає класу зображень без пухлини, мітка «1» – класу зображень з пухлиною.

Оцінка моделей виконується за використанням таких метрик: BinaryAccuracy, Precision, Recall та значення Loss.

2.5.6 Об’єднання прогнозів моделей

Описані моделі поєднуються у модель ансамблю за використання шару Average, який усереднює список входів по елементах.

Прогноз ансамблю моделей з використанням усереднення можна представити таким чином:

$$Pred = \frac{1}{n}\sum_{i=1}^{n} p_i,$$

де n – кількість базових моделей, p_i – прогнози базових моделей.

На рисунку 2.10 представлено опис моделі ансамблю з застосуванням шару усереднення, де depthwise_model – власна модель, model_densenet – модель DenseNet, model_mobilenet – модель MobileNet.

```
Model: "ensamble_avg"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_3 (InputLayer)        [(None, 128, 128, 3)]     0         []

 depthwise_model (Sequential)  (None, 1)               186113    ['input_3[0][0]']

 model_densenet (Sequential)  (None, 1)                7103169   ['input_3[0][0]']

 model_mobilenet (Sequential)  (None, 1)               2340033   ['input_3[0][0]']

 average (Average)           (None, 1)                 0         ['depthwise_model[0][0]',
                                                                  'model_densenet[0][0]',
                                                                  'model_mobilenet[0][0]']

==================================================================================================
Total params: 9,629,315
Trainable params: 332,803
Non-trainable params: 9,296,512
__________________________________________________________________________________________________
```

Рисунок 2.10 – Опис використаної моделі ансамблю з усередненням

Не всі базові моделі є однаково ефективними, тому при об’єднанні прогнозів окремих моделей корисним може бути використання показників їх точності як вагових коефіцієнтів.

У роботі також виконується прогноз, що поєднує моделі в співвідношенні, залежному від результатів показаної ними точності (формула 2.9). Прогноз ансамблю моделей з використанням точності базових моделей, як вагових коефіцієнтів можна представити таким чином:

$$Pred = \frac{\sum_{i=1}^{n} w_i \cdot p_i}{\sum_{i=1}^{n} w_i},$$

де – кількість базових моделей, w_i – точність базових моделей, p_i – прогнози базових моделей.

Графіки обох ансамблів моделей представлені у додатку А.

2.6.7 Отримання ансамблю моделей з найкращими результатами

Для дослідження ефективності окремих моделей та об’єднання їх прогнозів не було накладено умов на точність базових моделей.

Для практичного застосування ансамблю моделей для розпізнавання пухлин на МРТ зображеннях головного мозку необхідно обирати базові моделі з найкращими результатами. Для цього була накладена умова, за якою

здійснюватиметься перенавчання кожної базової моделі доти, доки її точність (розділ 2.4) не перевищить 95% і лише після цього прогноз базової моделі буде використаний для виконання прогнозу ансамблем моделей. Процес отримання прогнозу ансамблю моделей описаним способом представлено на рисунку 2.11.

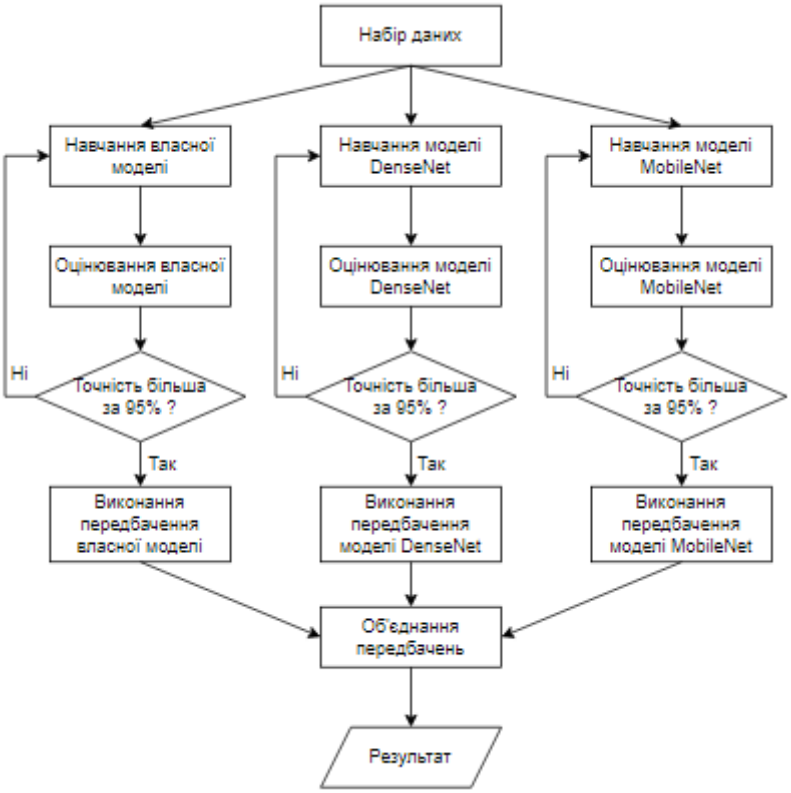


Рисунок 2.11 – Схема процесу отримання прогнозу ансамблю моделей з урахуванням базових моделей з високою точністю

2.6 Веб-додаток розпізнавання пухлин на зображеннях МРТ головного мозку

Для зручного застосування навчених моделей для розпізнавання пухлин на зображеннях МРТ головного мозку необхідно побудувати інтерфейс.

Розробка програмного веб-додатку умовно поділяється на дві частини: фронтенд і бекенд.

Фронтенд – це публічна (тобто клієнтська) частина веб-додатків, з якою користувач може взаємодіяти і контактувати напряму [52]. Тобто фронтенд включає все те, що клієнт бачитиме при відкритті веб-сторінки.

Бекенд – це програмно-апаратна частина проекту, це все те, що відбувається на стороні сервера і що залишається невидимим користувачеві [52].

Процес взаємодії фронтенд і бекенд [52]:

- фронтенд відправляє призначену для користувача інформацію в бекенд;
- інформація обробляється;
- інформація повертається назад, прийнявши цілісну форму і виконавши оброблений запит.

Для реалізації розробки клієнтської частини, тобто інтерфейсу, у даній роботі використані HTML, CSS і Python.

Мова гіпертекстової розмітки HTML передає інформацію про те, як побудовано текст і як він має виводитися на екран [49]. Мова каскадних таблиць CSS дозволяє оформити веб-сторінку, задавши необхідні кольори, шрифти та інші елементи стилю. Тобто за допомогою HTML ми наповнюємо сторінку змістом, а за допомогою CSS – оформлюємо її зовнішній вигляд.

Для того, щоб зробити створені сторінки функціональними, було використано Python.

Для виконання звернень до бази даних необхідно використовувати елементи мови програмування SQL.

Загалом в розробці веб-додатку можна виділити такі етапи:

1. Створити інтерфейс.
2. Додати функціонал.
3. Створити та під’єднати базу даних.

Значною проблемою у сфері розпізнавання медичних зображень є обмеженість доступних наборів даних. Тож можливість поповнення існуючого набору даних у процесі використання вже навченої моделі для розпізнавання може бути корисним. Створений додаток слугуватиме не лише для отримання результатів про відсутність або наявність пухлини на МРТ зображенні, але і для поповнення набору даних.

Алгоритм дій користувача у додатку можна представити наступним чином:

- користувач обирає зображення для введення;
- користувач обирає одну з трьох опцій: зображення містить пухлину, зображення не містить пухлину або ж дана інформація невідома;
- після цього користувач підтверджує свій вибір і відбувається процес обробки зображення та виконання прогнозу про наявність чи відсутність на ньому пухлини;
- користувач опиняється на сторінці з результатом прогнозу;
- у випадку, якщо попередньо користувач обрав пункт, що йому невідомо про наявність чи відсутність пухлини на зображенні, йому не пропонується зберігати зображення;
- у випадку, якщо попередньо користувач обрав один з пунктів, що позначають наявність чи відсутність пухлини, йому пропонується обрати, чи хоче він зберегти зображення у набір даних;
- якщо користувач обирає зберігати зображення, воно поповнює набір даних, потрапляючи в одну з двох папок: зображення з та без пухлини, залежно від обраного користувачем значення, також створюється відповідний запис у базі даних;
- якщо користувач обирає не зберігати зображення, воно потрапляє у папку з невизначеними зображеннями, також створюється відповідний запис у базі даних;
- після цього користувач може повернутись на початкову сторінку і почати працювати з іншим зображенням.

Ідея полягає в тому, щоб набір даних поповнювався правильними зображеннями, для яких відомий результат. Далі такий поповнений набір даних можна буде використати для проведення повторного навчання моделей нейронних мереж.

Процес роботи у веб-додатку розпізнавання пухлин головного мозку на зображеннях МРТ представлено у вигляді діаграми BPMN на рисунку 2.12.

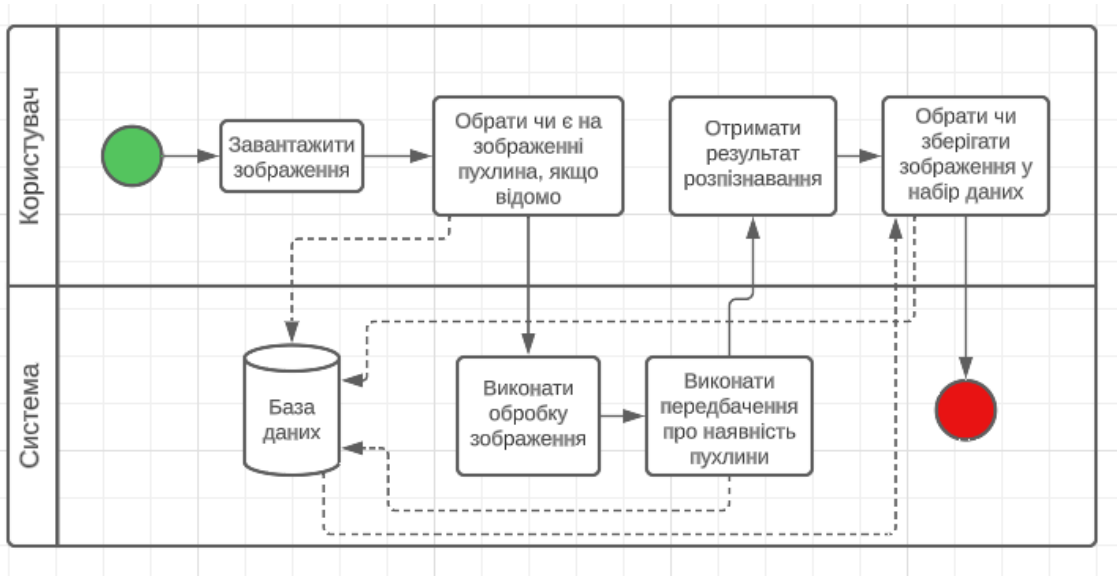


Рисунок 2.12 – BPMN діаграма процесу роботи у веб-додатку розпізнавання пухлин головного мозку на зображеннях МРТ

У базі даних міститься одна таблиця, що містить такі поля:

– img_id – ідентифікатор запису у базі даних;
– img_path – шлях файлу, що показує розташування файлу після збереження;
– saved – зберігає інформацію про те, чи було обрано зберегти зображення до набору даних;
– is_tumor – зберігає інформацію про те, яке значення (є пухлина / немає пухлини) було обране користувачем;
– is_tumor_predicted – зберігає інформацію про результати розпізнавання зображення;
– created_at – зберігає дату та час опрацювання зображення.

Висновок

Отже, у даному розділі був описаних функціонал створюваної нейромережної системи розпізнавання пухлин на зображеннях МРТ головного мозку. Був проведений аналіз використаних у дослідженні попередньо навчених моделей нейронних мереж, розглянуто їх архітектуру, визначено їх переваги та недоліки, та розглянуто поняття створення ансамблів моделей. Визначено програмні технологій, використані для створення нейромережної системи. Представлено структуру системи та будову використаних моделей нейронних мереж та логіку об’єднання прогнозів базових моделей у ансамблі. Представлено метрики, використані для оцінки ефективності моделей розпізнавання. Представлено процес роботи веб-додатку розпізнавання пухлин головного мозку на зображеннях МРТ.

РОЗДІЛ 3. АНАЛІЗ РЕАЛІЗОВАНОЇ НЕЙРОМЕРЕЖНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ПУХЛИН ГОЛОВНОГО МОЗКУ НА ЗОБРАЖЕННЯХ МРТ ТА РЕЗУЛЬТАТІВ ЇЇ ТЕСТУВАННЯ

3.1 Аналіз результатів роботи розробленої системи

З метою дослідження ефективності роботи різних моделей нейронних мереж було вирішено провести ряд експериментів.

Для верифікації одержаних результатів були проведені чисельні експерименти. Програмний код використаний для проведення експериментів розташований у додатку И.

Було зазначено, що для тренування власної моделі задано 100 епох, для попередньо навчених – 25 епох, при цьому для всіх моделей задано попередню зупинку, для власної моделі тренування зупиняється після 10-ти епох без покращення результатів, для попередньо навчених – після 2-х епох без покращення.

3.1.1 Результати для першого набору даних

У ході першого експерименту проводилось тестування власної моделі згорткової нейронної мережі, попередньо навчених моделей MobileNet та DenseNet, а також ансамблю з перелічених моделей на першому наборі даних, що включає 3000 зображень.

При виконанні першого запуску програми на першому наборі даних навчання власної моделі зупинилось на 56 епосі. У додатку Б представлені графіки змін метрик та функції втрат для власної моделі (синя лінія демонструє результати на навчальних даних, помаранчева – на тестових). На графіках можна спостерігати різкі перепади показників для тестової вибірки, що свідчить про нестабільність роботи власної моделі.

Навчання моделі DenseNet зупинилось на 15 епосі. У додатку В представлені графіки змін метрик та функції втрат для моделі DenseNet (синя лінія демонструє результати на навчальних даних, помаранчева – на тестових). На графіках можна спостерігати, що показники, як на навчальній так і на

тренувальній вибірках, змінюються досить рівномірно, при чому значення recall та accuracy на тестовому наборі із самого початку є досить хорошими.

Навчання моделі MobileNet зупинилось на 12 епосі. У додатку Г представлені графіки змін метрик та функції втрат для моделі MobileNet (синя лінія демонструє результати на навчальних даних, помаранчева – на тестових). На графіках можна спостерігати схожий результат з результатом, отриманим у експерименті з моделлю DenseNet. Показники, як на навчальній так і на тренувальній вибірках, змінюються досить рівномірно, а значення recall та accuracy на тестовому наборі із самого початку є досить хорошими. Тож обидві попередньо навчені моделі продемонстрували позитивну динаміку зміни показників ефективності у даному експерименті.

На рисунках 3.1-3.5 представлені матриці плутанини для кожного прогнозу.

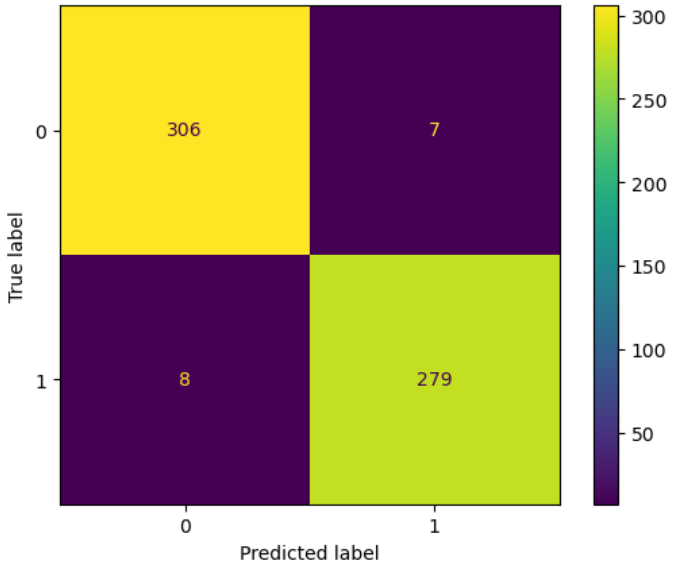


Рисунок 3.1 – Матриця плутанини прогнозу власної моделі

З рисунку можна зробити висновок, що 306 зображень без пухлини було віднесено до правильного класу, а 7 – ні, 279 зображень з пухлиною було віднесено до правильного класу, а 8 помилково визначені як такі, на яких відсутня пухлина.

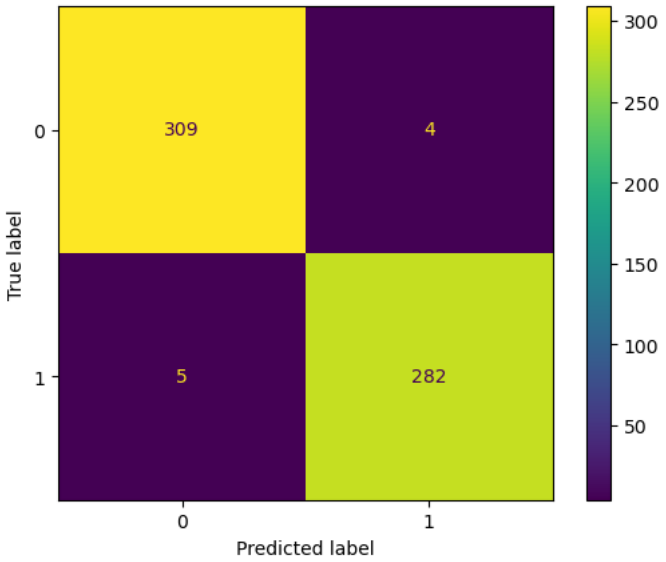


Рисунок 3.2 – Матриця плутанини прогнозу моделі DenseNet

З рисунку можна зробити висновок, що 309 зображень без пухлини було віднесено до правильного класу, а 4 – ні, 282 зображення з пухлиною були віднесені до правильного класу, а 5 помилково визначені як такі, на яких відсутня пухлина.

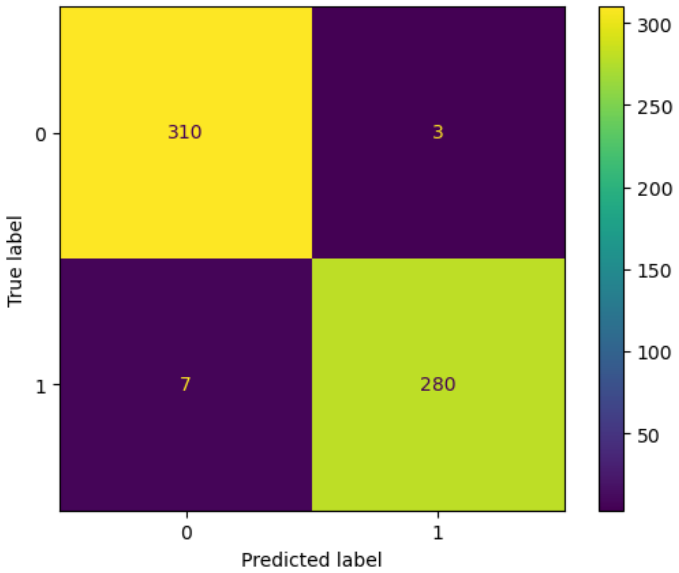


Рисунок 3.3 – Матриця плутанини прогнозу моделі MobileNet

З рисунку можна зробити висновок, що 310 зображень без пухлини було віднесено до правильного класу, а 3 – ні, 280 зображень з пухлиною було віднесено до правильного класу, а 7 помилково визначені як такі, на яких відсутня пухлина.

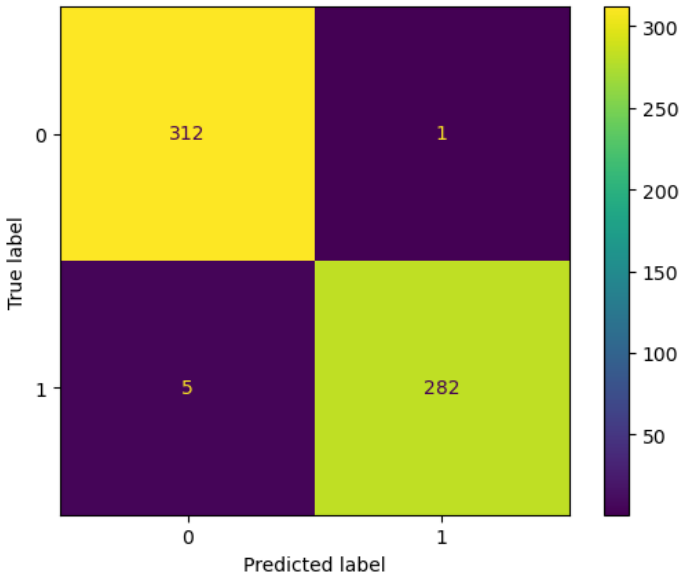


Рисунок 3.4 – Матриця плутанини прогнозу ансамблю моделей з використанням усереднення

З рисунку можна зробити висновок, що 312 зображення без пухлини були віднесені до правильного класу, а 1 – ні, 282 зображення з пухлиною були віднесені до правильного класу, а 5 помилково визначені як такі, на яких відсутня пухлина.

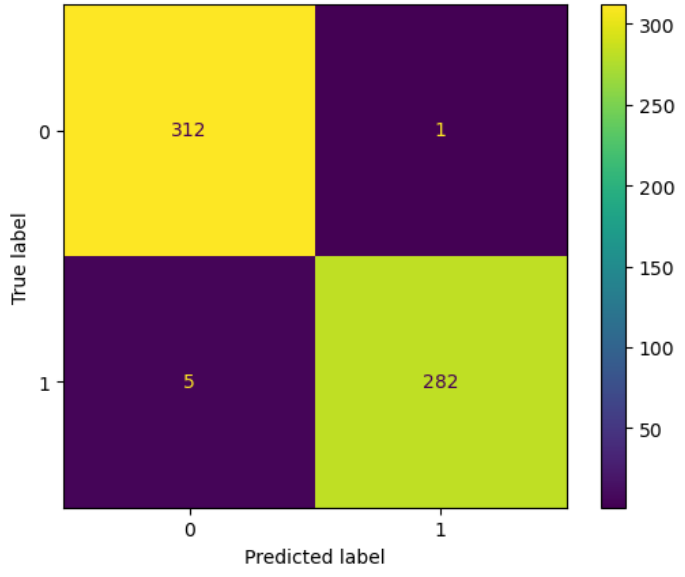


Рисунок 3.5 – Матриця плутанини прогнозу ансамблю моделей з використанням вагових коефіцієнтів

З рисунку можна зробити висновок, що 312 зображення без пухлини були віднесені до правильного класу, а 1 – ні, 282 зображення з пухлиною були

віднесені до правильного класу, а 5 помилково визначені як такі, на яких відсутня пухлина.

Була проведена оцінка показників ефективності для побудованих моделей при виконанні ними прогнозів на тестовій вибірці.

Результати роботи всіх розглянутих моделей для першого запуску на першому наборі даних представлені у таблиці 3.1.

Таблиця 3.1 – Показники ефективності розглянутих моделей для першого запуску на першому наборі даних

	Precision	Recall	Accuracy	Loss
CNN	0.9755244755	0.9721254355	0.975	0.0691939157
DenseNet	0.9860139860	0.9825783972	0.985	0.0581679653
MobileNet	0.9893992933	0.9756097561	0.9833333333	0.0707583255
Усереднення	0.9964664311	0.9825783972	0.99	0.0413697210
З різними вагами	0.9964664311	0.9825783972	0.99	0.0413630694

Результати отримані обома методами поєднання моделей для першого запуску на першому наборі даних рівні і зовсім незначно відрізняються лише показником loss, котрий кращий для методу, що враховує точність кожної базової моделі при поєднанні.

На рисунку 3.6 синіми крапками позначені цільові значення тестового набору, зеленою лінією – прогноз ансамблю моделей з використанням усереднення значень, а помаранчевими крапками – прогноз ансамблю моделей з використанням ваг, рівних значенню точності базових моделей (на графіку представлена 1/10 частина тестового набору).

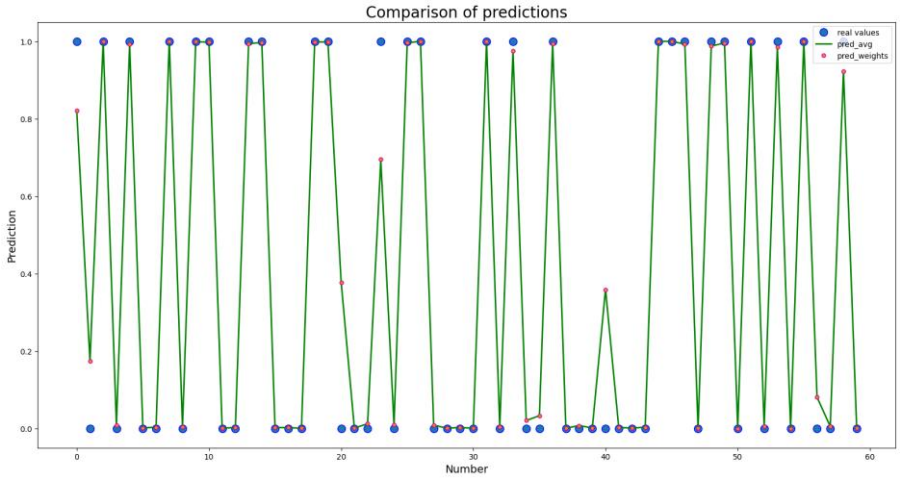


Рисунок 3.6 – Порівняння результатів прогнозів ансамблів моделей для першого запуску на першому наборі даних

З рисунку можна побачити, наскільки отримане при виконанні прогнозування значення близьке до реального, та як відрізняються прогнози двох ансамблів моделей. У даному експерименті значення отримані двома різними ансамблями моделей майже збігаються.

Також проведено ручне тестування на окремих зображеннях. Було вибрано зображення з другого набору даних, щоб оцінити ефективність роботи на зображеннях, що не використовувались для навчання. Прогноз виконувався за використання ансамблю моделей з ваговими коефіцієнтами. На рисунку 3.7 представлено результат виявлення пухлини на зображенні.

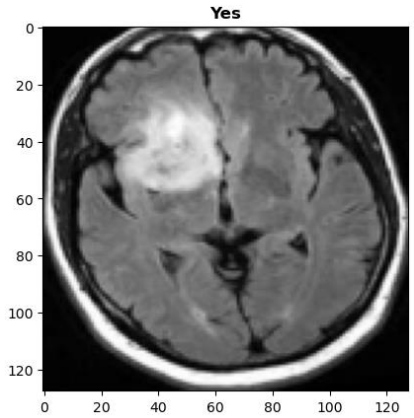


Рисунок 3.7 – Результат виявлення пухлини на зображенні МРТ головного мозку

Усі взяті для тестування зображення були віднесені до правильного класу.

На першому наборі даних було проведено ще 9 запусків для спостереження різних результатів і отримання середніх значень. Середні значення показників ефективності для 10 запусків програми на першому наборі даних представлені у таблиці 3.2.

Таблиця 3.2 – Середні значення показників ефективності розглянутих моделей для 10 запусків програми на першому наборі даних

	Precision	Recall	Accuracy	Loss
CNN	0.623119694	0.702339472	0.775	0.34157785
DenseNet	0.983160176	0.986560478	0.98547619	0.052063234
MobileNet	0.978246425	0.982080637	0.980952381	0.072367216
Усереднення	0.989027954	0.985564958	0.987857143	0,116651489
З різними вагами	0.989047307	0.987555998	0.988809524	0.087118497

За отриманими для першого набору даних результатами, що наведені у таблиці, можна зробити висновок, що серед базових моделей найкращі результати показала модель DenseNet, а поєднання моделей у ансамбль дало кращі результати ніж кожна модель окремо. Отримані результати показали, що попередньо навчені моделі показують досить стабільні результати і відрізняються для різних запусків програми в межах 2-3%, а власна модель працює досить нестабільно і може не впоратись із завданням розпізнавання. Власна модель значно простіша за своєю структурою і складається з невеликої кількості шарів, тож значно поступається своєю ефективністю попередньо навченим моделям. При цьому обидва ансамблі моделей стабільно демонструють хороші результати, а результати для ансамблю з вагами трохи кращі, ніж для ансамблю з використанням усереднення.

Для використання у веб-додатку було вирішено вибрати ансамбль моделей з вагами, навчений на першому наборі даних, що поєднує прогнози базових моделей, що показали точність більше 95%. Для цього здійснювалось перенавчання базових моделей, що для яких показник accuracy був меншим за 0.95. Програмний код для поєднання прогнозів базових моделей з точністю

більшою за 95% для використання у виконанні прогнозування веб-додатком розташований у додатку К. Вибраний для використання у виконанні прогнозування веб-додатком ансамбль моделей показав точність 0.998.

3.1.2 Результати для другого набору даних

У ході другого експерименту проводилось тестування власної моделі згорткової нейронної мережі, попередньо навчених моделей MobileNet та DenseNet, а також ансамблю з перелічених моделей на другому наборі даних, що включає 400 зображень.

При виконанні першого запуску програми на другому наборі даних навчання власної моделі зупинилось на 11 епосі. У додатку Д представлені графіки змін метрик та функції втрат для власної моделі (синя лінія демонструє результати на навчальних даних, помаранчева – на тестових). На графіках можна спостерігати, що всі показники дещо покращуються для навчального набору і залишаються незмінними та демонструють поганий результат на тестовому наборі. Можна зробити висновок, що модель не здатна впоратись із поставленою задачею.

Навчання моделі DenseNet досягло 25 епохи. У додатку Е представлені графіки змін метрик та функції втрат для моделі DenseNet (синя лінія демонструє результати на навчальних даних, помаранчева – на тестових). На графіках можна спостерігати, що загалом показники, як на навчальній так і на тренувальній вибірках, змінюються досить рівномірно, окрім значення recall, яке дещо коливається для тестової вибірки, але при цьому зберігає досить хороший результат. Порівнюючи з результатами, отриманими у попередньому експерименті на великому наборі даних, можна відзначити, що на меншому наборі даних показники змінюються менш рівномірно.

Навчання моделі MobileNet зупинилось на 16 епосі. У додатку Ж представлені графіки змін метрик та функції втрат для моделі MobileNet (синя лінія демонструє результати на навчальних даних, помаранчева – на тестових). На графіках можна спостерігати, що загалом показники, як на навчальній так і на тренувальній вибірках, змінюються досить рівномірно і досягають хороших

результатів, як і для моделі DenseNet, а значення recall, хоч і має деякі коливання на початку тренування для тестової вибірки, але швидко приймає хороше значення і зберігає цей рівень далі.

На рисунках 3.8-3.12 представлені матриці плутанини для кожного прогнозу.

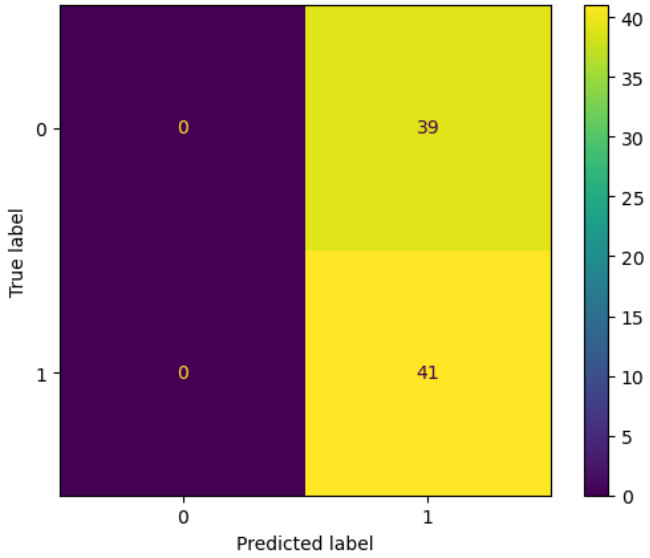


Рисунок 3.8 – Матриця плутанини прогнозу власної моделі

З рисунку можна зробити висновок, що всі зображення без пухлини, а саме 39, були помилково визначені як такі, на яких є пухлина, а всі зображення з пухлиною, а саме 41, були визначені правильно. Тобто власна модель віднесла всі зображення до одного класу – класу зображень з пухлиною.

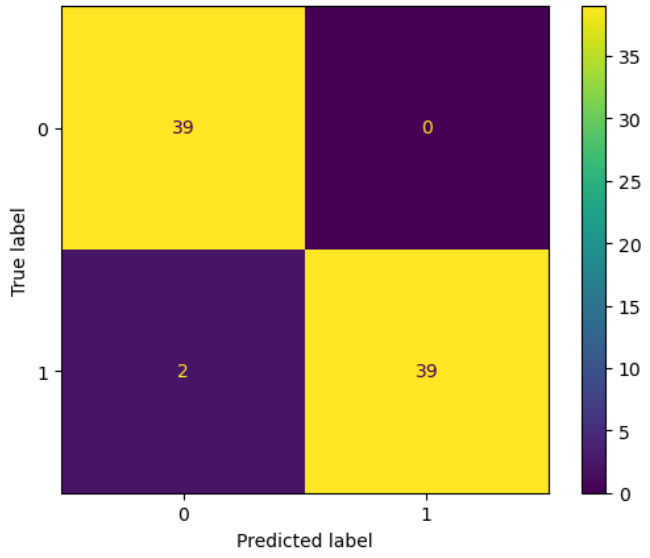


Рисунок 3.9 – Матриця плутанини прогнозу моделі DenseNet

З рисунку можна зробити висновок, що всі зображення без пухлини, а саме 39, було віднесено до правильного класу, 39 зображень з пухлиною були віднесені до правильного класу, а 2 помилково визначені як такі, на яких відсутня пухлина.

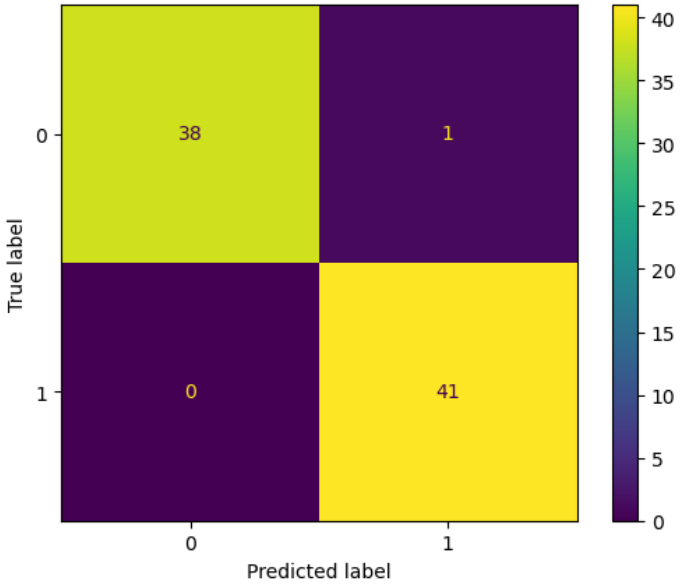


Рисунок 3.10 – Матриця плутанини прогнозу моделі MobileNet

З рисунку можна зробити висновок, що 38 зображень без пухлини було віднесено до правильного класу, а 1 – ні, всі зображення з пухлиною, а саме 41, було віднесено до правильного класу.

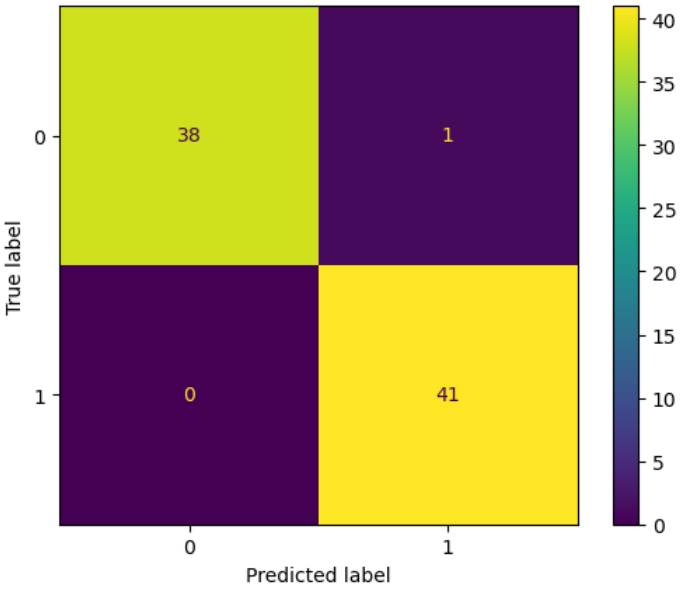


Рисунок 3.11 – Матриця плутанини прогнозу ансамблю моделей з використанням усереднення

З рисунку можна зробити висновок, що 38 зображень без пухлини були віднесені до правильного класу, а 1 – ні, всі зображення з пухлиною, а саме 41, були віднесені до правильного класу.

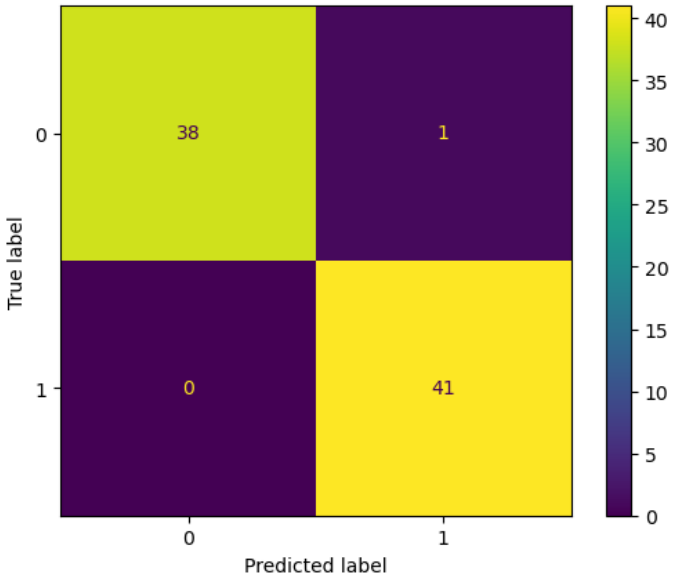


Рисунок 3.12 – Матриця плутанини прогнозу ансамблю моделей з використанням вагових коефіцієнтів

З рисунку можна зробити висновок, що 38 зображень без пухлини були віднесені до правильного класу, а 1 – ні, всі зображення з пухлиною, а саме 41, були віднесені до правильного класу.

Результати роботи всіх розглянутих моделей для першого запуску на другому наборі даних представлені у таблиці 3.3.

Таблиця 3.3 – Показники ефективності розглянутих моделей для першого запуску на другому наборі даних

	Precision	Recall	Accuracy	Loss
CNN	0.5125	1.0	0.5125	0.6928344982
DenseNet	1.0	0.9512195122	0.975	0.0523965833
MobileNet	0.9761904762	1.0	0.9875	0.0392137257
Усереднення	0.9761904762	1.0	0.9875	0.2131858015
З різними вагами	0.9761904762	1.0	0.9875	0.1440551598

Результати отримані обома методами поєднання моделей для першого запуску на другому наборі даних рівні за всіма показниками, крім loss, її значення краще для методу, що враховує точність кожної базової моделі при поєднанні.

На рисунку 3.13 синіми крапками позначені цільові значення тестового набору, зеленою лінією – прогноз ансамблю моделей з використанням усереднення значень, а помаранчевими крапками – прогноз ансамблю моделей з використанням ваг, рівних значенню точності базових моделей.

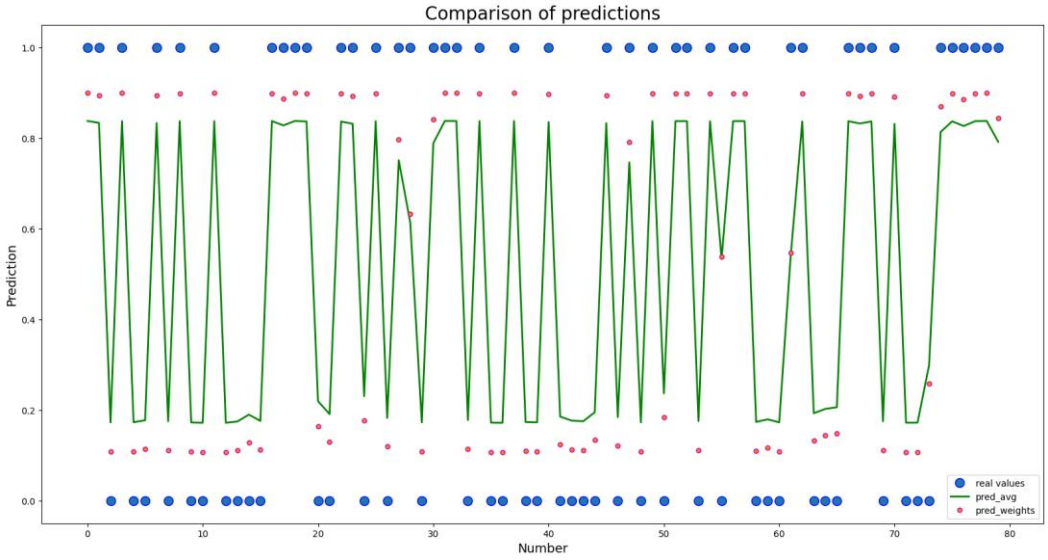


Рисунок 3.13 – Порівняння результатів прогнозів ансамблів моделей для першого запуску на другому наборі даних

З рисунку можна побачити, наскільки отримане при виконанні прогнозування значення близьке до реального, та як відрізняються прогнозів двох ансамблів моделей. У даному експерименті наочно можна побачити, що початкові результати прогнозування, що поєднує базові моделі з вагами, рівними отриманій ними точності, значно ближчі до реального значення, ніж для прогнозу, що поєднує базові моделі у рівному співвідношенні. З цього можна зробити висновок про кращу ефективність прогнозу, що враховує ефективність базових моделей. У такому випадку, якщо окрема модель виявиться неефективною на певному наборі даних, використання ваг дозволяє уникнути погіршення результатів ансамблю моделей.

Як і для попереднього експерименту, було проведено ручне тестування на окремих зображеннях. Зображення було вибрано з першого набору даних, щоб оцінити ефективність роботи на зображеннях, що не використовувались для навчання. На рисунку 3.14 представлено результат виявлення пухлини на зображенні.

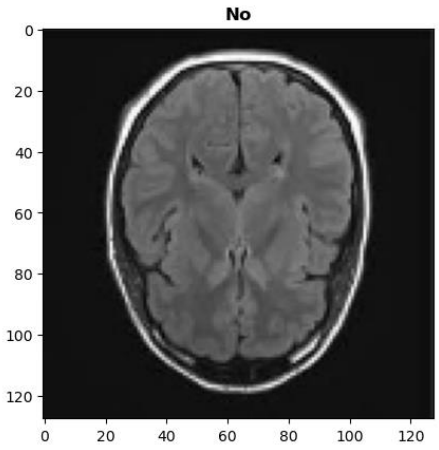


Рисунок 3.14 – Результат виявлення пухлини на зображенні МРТ головного мозку

Усі взяті для тестування зображення були віднесені до правильного класу.

Як і у попередньому експерименті, на другому наборі даних було проведено ще 9 запусків для спостереження різних результатів. Середні значення показників ефективності для 10 запусків програми на другому наборі даних представлені у таблиці 3.4.

Таблиця 3.4 – Середні значення показників ефективності розглянутих моделей для 10 запусків програми на другому наборі даних

	Precision	Recall	Accuracy	Loss
CNN	0.5125	1.0	0.5125	0.69306431
DenseNet	0.971931862	0.959349593	0.964583333	0.100351961
MobileNet	0.971733449	0.971544715	0.970833333	0.075256236
Усереднення	0.987998451	0.995934959	0.991666667	0.236578899
З різними вагами	0.992063492	0.991869919	0.991666667	0.172045516

За отриманими для другого набору даних результатами можна зробити висновок, що власна модель згорткової нейронної мережі не впоралась з поставленою задачею жодного разу, а найкращі результати серед базових моделей показала модель DenseNet, хоча її результати не надто відрізнялись від результатів MobileNet. Попередньо навчені моделі, як і для великого набору даних, показують досить стабільні та високі результати. Хоча низькі показники ефективності власної моделі не завадили ансамблю моделей отримати хороші результати, проте можна прослідкувати негативний вплив на значення loss. Ансамблі моделей стабільно демонструють хороші результати. Ансамбль моделей може показати результати, незначно гірші за показані однією з базових моделей, проте проявили себе більш стабільними, зберігаючи високу ефективність під час всіх запусків і для обох розглянутих наборів даних. Середні значення показників precision, recall, accuracy для даного набору даних кращі, ніж для першого, проте отримане значення loss гірше для меншого набору даних.

3.1.3 Результати для об’єданого набору даних

Два використані набори даних було об’єднано в один, всього у новому наборі даних отримано 3400 зображень. До новоутвореного набору даних було застосовано збережену модель DenseNet, створену у першому експерименті та навчену на даних першого набору даних, та збережену модель MobileNet, створену у другому експерименті та навчену на даних другого набору даних. Проводилась оцінка моделей на об’єднаному наборі даних та відповідно створення ансамблів цих моделей, як і у двох попередніх експериментах. Результати представлені нижче.

На рисунках 3.15-3.18 представлені матриці плутанини для кожного прогнозу.

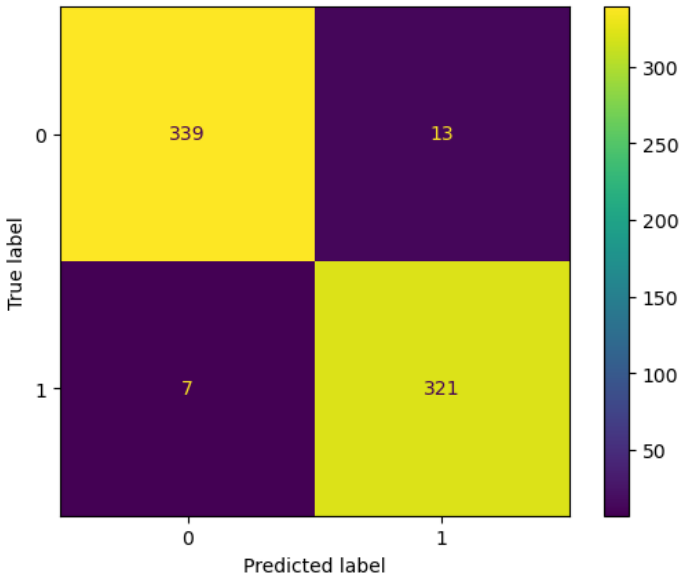


Рисунок 3.15 – Матриця плутанини прогнозу моделі DenseNet

З рисунку можна зробити висновок, що 339 зображень без пухлини було віднесено до правильного класу, а 13 – ні, 321 зображення з пухлиною було віднесені до правильного класу, а 7 помилково визначені як такі, на яких відсутня пухлина.

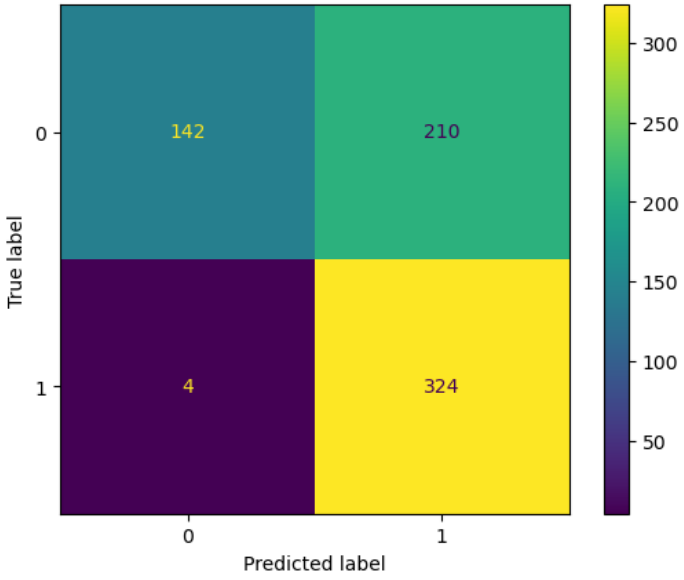


Рисунок 3.16 – Матриця плутанини прогнозу моделі MobileNet

З рисунку можна зробити висновок, що 142 зображення без пухлини було віднесено до правильного класу, а 210 – ні, 324 зображення з пухлиною було віднесено до правильного класу, а 4 помилково визначені як такі, на яких відсутня пухлина.

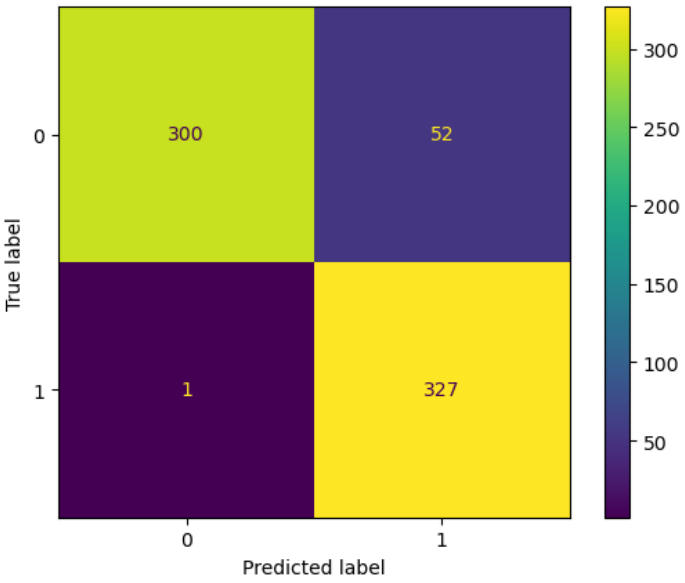


Рисунок 3.17 – Матриця плутанини прогнозу ансамблю моделей з використанням усереднення

З рисунку можна зробити висновок, що 300 зображень без пухлини були віднесені до правильного класу, а 52 – ні, 327 зображень з пухлиною були віднесені до правильного класу, а 1 помилково визначене як таке, на якому відсутня пухлина.

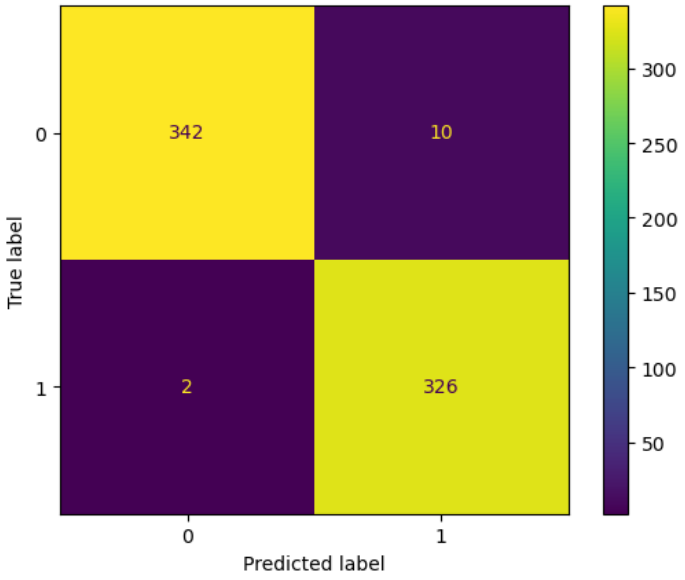


Рисунок 3.18 – Матриця плутанини прогнозу ансамблю моделей з використанням вагових коефіцієнтів

З рисунку можна зробити висновок, що 342 зображення без пухлини були віднесені до правильного класу, а 10 – ні, 326 зображень з пухлиною були

віднесені до правильного класу, а 2 помилково визначені як такі, на яких відсутня пухлина.

На рисунку 3.19 синіми крапками позначені цільові значення тестового набору, зеленою лінією – прогноз ансамблю моделей з використанням усереднення значень, а помаранчевими крапками – прогноз ансамблю моделей з використанням ваг, рівними значенню точності базових моделей.

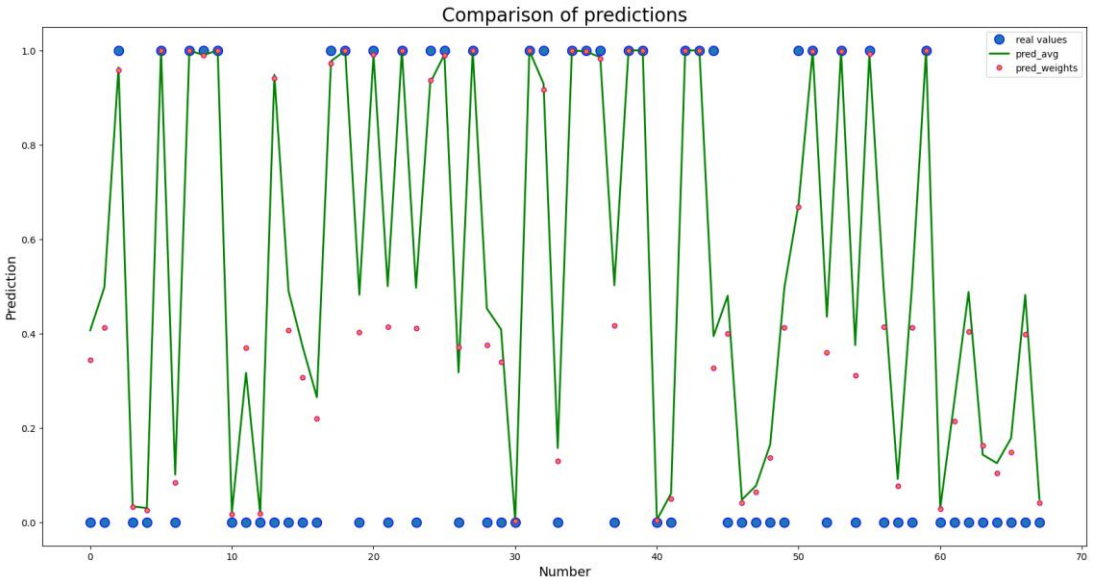


Рисунок 3.19 – Порівняння результатів прогнозів для першого запуску на об’єднаному наборі даних

З рисунку можна побачити, наскільки отримане при виконанні прогнозування значення близьке до реального, та як відрізняються прогнози двох ансамблів моделей. У даному експерименті наочно можна побачити, що початкові результати прогнозу, що поєднує базові моделі з вагами, рівними отриманій ними точності, значно ближчі до реального значення, ніж для прогнозу, що поєднує базові моделі у рівному співвідношенні. Враховуючи це, можна зробити висновок про кращу ефективність прогнозу, що враховує ефективність базових моделей.

На об’єднаному наборі даних, як і у попередніх експериментах було проведено ще 9 запусків для спостереження різних результатів. Середні значення показників ефективності для 10 запусків програми на об’єднаному наборі даних представлені у таблиці 3.5.

Таблиця 3.5 – Середні значення показників ефективності розглянутих моделей для 10 запусків програми на об’єднаному наборі даних

	Precision	Recall	Accuracy	Loss
DenseNet	0.965809819	0.990091463	0.978308824	0.073191563
MobileNet	0.607259399	0.988567073	0.686029412	1.190340221
Усереднення	0.864926439	0.995426829	0.922794118	0.235550619
З різними вагами	0.974719761	0.995426829	0.985294118	0.192883506

За отриманими для об’єднаного набору даних результатами можна зробити висновок, що модель MobileNet, що була навчена на другому наборі даних, показала досить низькі показники ефективності. Попередньо навчені моделі показують досить стабільні результати: результати застосування моделі DenseNet кожного разу були хорошими, а результати застосування моделі MobileNet завжди були поганими, що дозволяє стверджувати про гіршу ефективність моделі, навченої на малому наборі даних. Ансамблі моделей також демонструють стабільні результати. Ансамбль моделей з використанням усереднення показують результати, гірші за показані кращою з базових моделей. А ансамбль моделей, що враховував точність базових моделей показав найкращий результат. Поєднання двох моделей, замість трьох, як було у попередніх експериментах, дозволяє краще продемонструвати ефективність врахування оцінок базових моделей, адже ансамбль моделей з використанням ваг завжди покращував результат.

3.1.4 Порівняння оптимізаторів

У процесі роботи проводився підбір найбільш оптимального для поставленої задачі оптимізатора. Тестування виконувалось на двох попередньо навчених моделях для першого набору даних.

У процесі навчання ми намагаємося мінімізувати значення функції втрат та оновлювати параметри підвищення точності. Параметри нейронної мережі – це зазвичай ваги зв'язків. Оптимізатор – це метод досягнення найкращих результатів, допомога у прискоренні навчання, тобто це алгоритм, що

використовується для незначної зміни параметрів, таких як ваги та швидкість навчання, щоб модель працювала правильно та швидко.

Тестувались такі оптимізатори:

- Adam (adaptive moment estimation). Оптимізація Адам — це метод стохастичного градієнтного спуску, який базується на адаптивній оцінці моментів першого та другого порядку. Він поєднує принципи інерції MomentumSGD та адаптивного оновлення параметрів AdaGrad та його модифікацій.
- SGD (stochastic gradient descent). Оптимізатор градієнтного спуску (з імпульсом), при використанні SGD з накопиченням імпульсу враховуються значення минулих градієнтів, загальний напрямок зберігається, і траєкторія залишається правильною.
- RMSPrиор (root mean square propagation). Суть RMSprop полягає в наступному: підтримувати ковзне середнє квадрата градієнтів, розділити градієнт на корінь із цього середнього. Цей алгоритм оптимізації розроблений паралельно з AdaDelta і його складовою. Обидва алгоритми були створені для вирішення основної проблеми AdaGrad: безконтрольного накопичення квадратів градієнтів, що зрештою призводило до паралічу процесу навчання.
- Adadelta. Оптимізація Adadelta — це метод стохастичного градієнтного спуску, який базується на адаптивній швидкості навчання для кожного виміру для усунення двох недоліків: постійне зниження темпів навчання протягом усього навчання, необхідність вибраної вручну глобальиної швидкості навчання. Adadelta — це більш надійне розширення Adagrad, яке адаптує швидкість навчання на основі рухомого вікна оновлень градієнтів замість накопичення всіх попередніх градієнтів. Таким чином Adadelta продовжує навчатися навіть після багатьох оновлень.
- Adagrad (adaptive gradient algorithm). Adagrad — це оптимізатор зі швидкістю навчання залежно від параметра, яка адаптована до частоти

оновлення параметра під час навчання. Цей метод враховує історію всіх миниулих градієнтів кожного окремо взятого параметра. Це дає можливість зменшувати розмір кроку навчання для параметрів, які мають великий градієнт.

Результати оцінювання оптимізаторів приведені у таблицях 3.6 і 3.7.

Таблиця 3.6 – Показники ефективності моделі DenseNet з використанням різних оптимізаторів для першого набору даних

Оптимізатор	Precision	Recall	Accuracy	Loss
Adam	0.9827586209	0.9930313589	0.9883333333	0.0550092915
SGD	0.9578947368	0.9512195122	0.9566666667	0.1230148917
RMSprop	0.8636363636	0.9930313589	0.9216666667	0.2116242118
Adadelta	0.4943181818	0.606271777	0.515	0.8366111557
Adagrad	0.9277978339	0.8954703833	0.9166666667	0.234461487

Таблиця 3.7 – Показники ефективності моделі MobileNet з використанням різних оптимізатоирів для першого набору даних

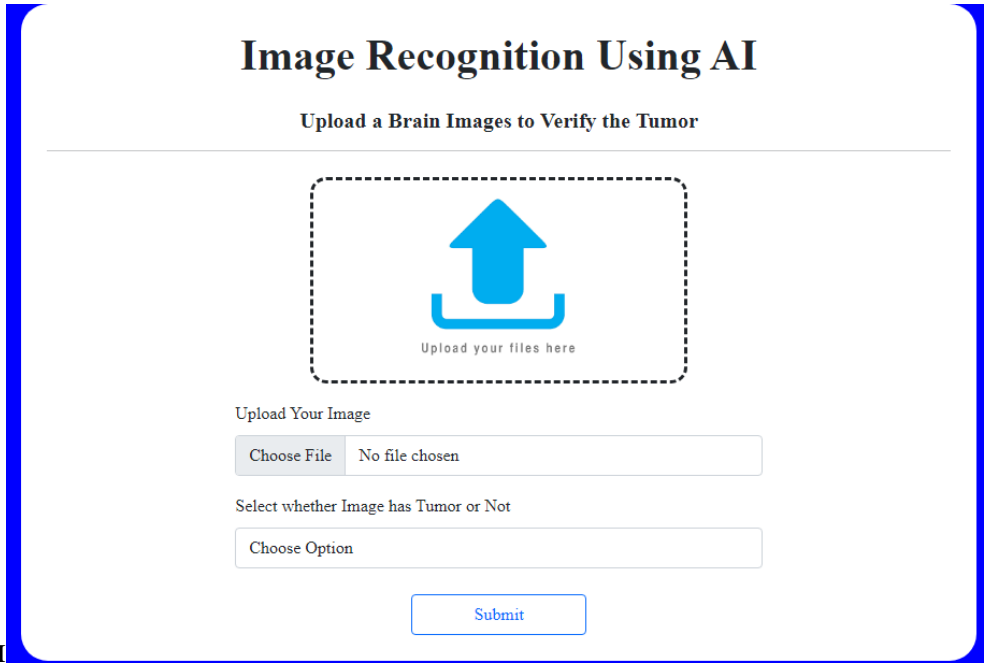
Оптимізатор	Precision	Recall	Accuracy	Loss
Adam	0.969072165	0.9825783972	0.9766666667	0.0849088968
SGD	0.9611307421	0.9477351916	0.9566666667	0.1446276354
RMSprop	0.9240924092	0.9756097561	0.95	0.1386175942
Adadelta	0.6605504587	0.7526132404	0.6966666667	0.594848810
Adagrad	0.9217081851	0.9024390244	0.9166666667	0.2364766582

Обидві досліджувані моделі показали найкращі результати за використання оптимізатора Адам. Отже, за отриманими результатами найоптимальнішим для виконання поставленої у роботі задачі виявився оптимізатор Адам, що і був застосований для всіх моделей.

и

3.2 Демонстрація роботи розробленого веб-додатку

Продемонструємо, як користувач може використовувати розроблений веб-додаток для розпізнавання пухлин на зображенні МРТ. Початкова сторінка додатку представлена на рисунку 3.20.



и

Рисунок 3.20 – Початкова сторінка веб-додатку розпізнавання пухлин на зображенні МРТ головного мозку

Натиснувши на кнопку «Choose File» користувач може обрати зображення з комп’ютера, після чого обрати чи на зображенні є пухлина, чи ні, або залишити поле без вибору (рис. 3.21).

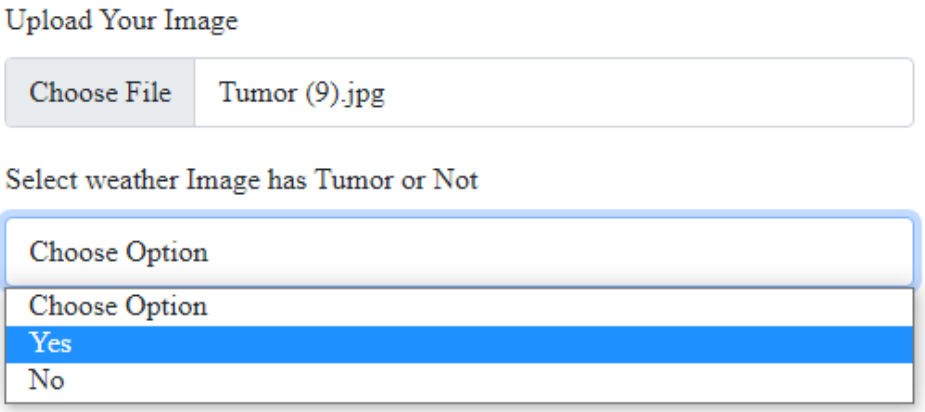


Рисунок 3.21 – Вибір користувачем опції про наявність чи відсутність пухлини на зображенні

Після того, як вибір зроблено, користувач натискає «Submit» і зображення відправляється на обробку.

Далі система виконує розпізнавання пухлини на зображенні з використанням вже навченого на першому наборі даних ансамблю моделей з вагами.

На наступній сторінці користувач може побачити результат про наявність чи відсутність на зображенні пухлини отриманий за допомогою нейромережних моделей (рис. 3.22).

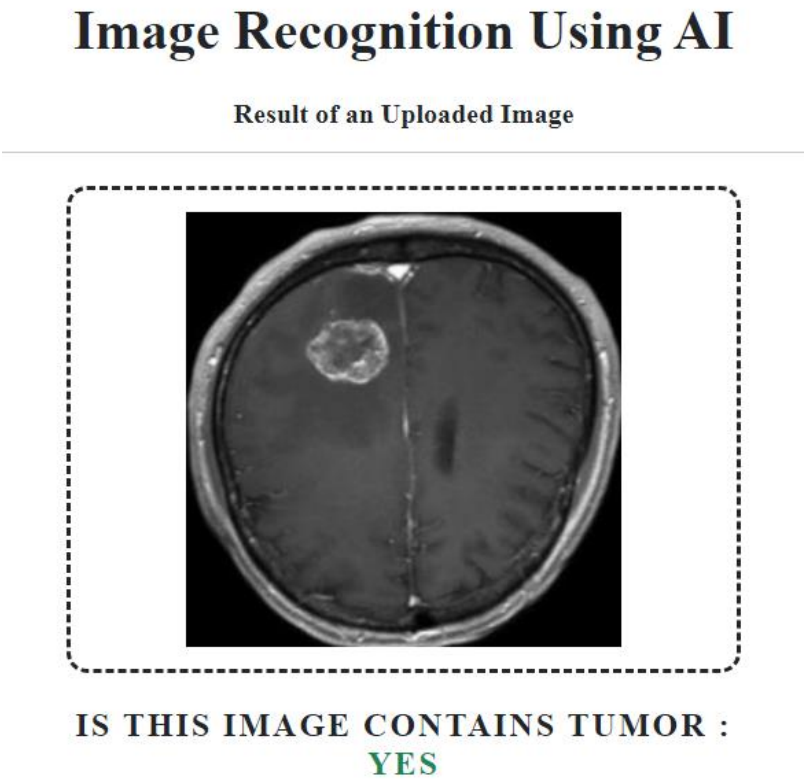


Рисунок 3.22 – Відображення результату розпізнавання пухлини головного мозку

Після перегляду результатів розпізнавання користувач може зробити вибір, чи бажає він зберегти зображення у набір даних.

Зробивши вибір користувач натискає кнопку «Clear & Back to Home», його дії зберігаються, а він повертається на початкову сторінку. Якщо було обрано зберігати зображення, з’явиться спливаюче вікно, що сповіщає про успішне збереження.

Якщо користувач на першій сторінці не обрав опції про наявність чи відсутність пухлини, він може отримати результат прогнозу, проте збереження зображення не пропонується (рис 3.23).

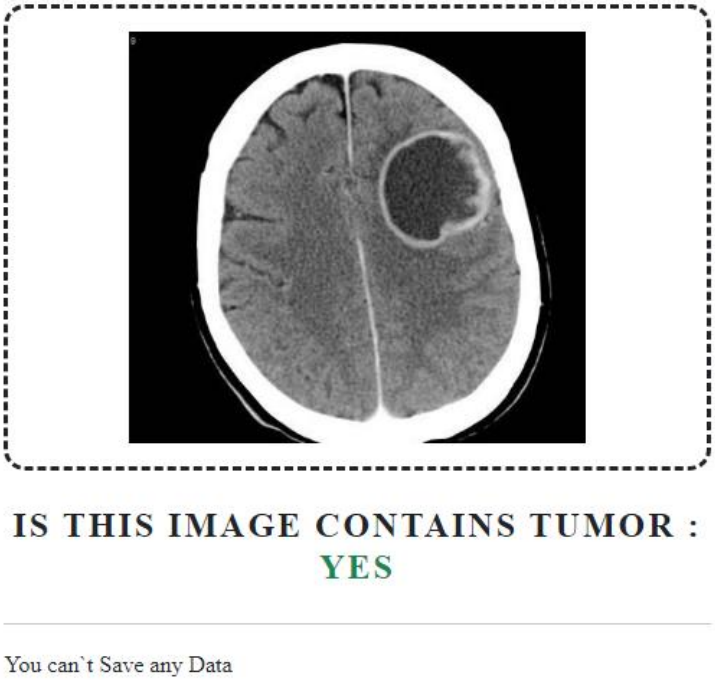


Рисунок 3.23 – Вікно з прогнозом для користувача, що не обрав опцію про наявність чи відсутність пухлини на зображенні

Переглядаючи базу даних, можна спостерігати шлях збереження зображення, вибір користувача та результати прогнозу стосовно наявності пухлини (рис. 3.24).

img_id	img_path	saved	is_tumor	is_tumor_predicted	created_at
2	uploads/yes/3549d393-e792-443e-bf92-6c0df212fdb8Tu...	Yes	Yes	Yes	2023-05-01 03:46:10

Рисунок 3.24 – Запис у базі даних для опрацьованого зображення

У базі даних ми можемо переглянути результати для інших опрацьованих зображень (рис. 3.25).

img_id	img_path	saved	is_tumor	is_tumor_predicted
1	uploads/no/2f4c7446-8c3a-4606-b682-1cd4ae47469eno8...	Yes	No	No
2	uploads/yes/3549d393-e792-443e-bf92-6c0df212fdb8Tu...	Yes	Yes	Yes
3	uploads/no/00383d63-a782-4391-8fb1-ef6ada80cd3bNor...	Yes	No	No
4	uploads/not_defined/908725b9-4007-441e-9ebe-081958...	No	No	No

Рисунок 3.25 – Записи у базі даних для інших опрацьованих зображень

Таким чином за допомогою створеного додатку поповнюється набір даних який в подальшому зможе бути використаний для повторного навчання моделей.

3.3 Порівняння отриманих результатів з результатами використання інших методів

Порівняємо отримані у даній роботі результати з деякими існуючими технологіями, націленими на виконання тієї ж задачі. У порівнянні бере участь ансамбль моделей, що використовує ваги.

У роботі [24] виконується класифікація нормального та пухлинного мозку з використанням нейронних мереж. У роботі використовувалась попередньо навчена модель MobileNet. Набір даних взято з веб-сайту Kaggle. Цей набір містить дві папки, одна з яких представляє зображення нормального мозку, а інша – зображення мозку з пухлиною. Всього у наборі даних 3762 МРТ зображення.

У роботі [27] порівнюються моделі штучної та згорткової нейронних мереж. Набір даних взято з сайту Github. Цей набір даних містить дві папки, одна з яких представляє зображення нормального мозку, а інша – зображення мозку з пухлиною. Всього у наборі даних 2065 зображень.

У роботі [5] пропонується гібридна модель глибокого навчання згорткова нейронна мережа – довгострокова пам’ять (CNN-LSTM) для прогнозування пухлин мозку за допомогою МРТ зображень. Набір даних зібрано з Kaggle.

Набір даних складається із 253 зображень, що розділені на дві папки, що містять зображення мозку з та без пухлини.

У роботі [53] запропоновано модель глибокого вейвлет-автокодувальника під назвою «модель DWAE», яка використовується для поділу вхідних даних зображення мозку з та без пухлини. У цій роботі використовувались п’ять типів баз даних МРТ мозку, включаючи BRATS2012, BRATS2013, BRATS2014, 2015 challenge, Brats 2015 та ISLES. Набір даних містить 2500 МРТ-зображень мозку з та без пухлини.

У роботі [54] запропоновано гібридний алгоритм, що поєднує модель середнього Гауса SVM (MG-SVM), модель Fine KNN та модель косинусної KNN. Набір даних, проаналізований під час дослідження, є доступний в Інтернеті через Kaggle. Набір даних містить всього 150 зображень і розділений на дві підпапки: зображення мозку з пухлинами, і зображення здорового мозку.

У дослідженні [55] була запропонована багатоканальна модель на основі Faster R-CNN з новою формулою вибору каналу для виявлення пухлин мозку. У дослідженні використовували три різні набори даних. Перший набір даних був отриманий із джерела з відкритим доступом у відкритому доступі figshare, другий набір даних було взято зі сховища Kaggle, а третім набором даних було вибрано BRATS 2018. У дослідженні було використано 3064 зображення з першого набору даних, 253 зображення з другого набору даних і 285 зображень з третього набору даних із шістьма різними класифікаторами та підходами доповнення даних.

У роботі [56] використовується нейронна мережа Depthwise Separable Convolution на основі глибокого навчання для виявлення пухлини на основі зображень МРТ. Експерименти проводяться на загальнодоступному наборі даних, доступному веб-сайті Kaggle. Цей набір даних містить 253 зображення МРТ, що розбиті на зображення не ракового та ракового типу.

У статті [57] представлено модель багатозадачного навчання (MTL), яка може приймати зображення 2D-магнітно-резонансної томографії як вхідні дані та дає прогнози для кількох виходів, таких як виявлення та сегментація. Для

оцінки запропонованої архітектури було використано набір даних Brain Tumor Segmentation (BRaTs) 2019 та BRaTs 2020.

В дослідженні [58] запропоновано метод виявлення пухлини на МРТ-зображеннях головного мозку на основі згорткової нейронної мережі. розробленої на основі Inception V3.

У статті [59] робота спрямована на розробку та оцінку згорткової нейронної мережі найсучаснішої продуктивності Transfer Learning, запропонованої для класифікації зображень протягом останніх років. У експерименті використані такі попередньо навчені мережі: AlexNet, Vgg16, GoogLeNet, Resnet50, Inceptionv3. В оцінці використовувались два набори даних: перший, стандартний набір даних із бази даних RIDER Neuro MRI, що включає 349 зображень МРТ головного мозку, другий набір даних складається із 120 зображень МРТ головного мозку, обидва набори розділені на нормальні та аномальні зображення.

У статті [60] запропоновано новий метод діагностики пухлин головного мозку шляхом поєднання інтелектуального аналізу даних і методів машинного навчання. Для класифікації виділених ознак і виявлення пухлин головного мозку використовується комбінація алгоритму Наївного Байєса (NB), методу опорних векторів (SVM) та K-найближчих сусідів (KNN). Кожен із трьох алгоритмів виконує класифікацію ознак окремо, а кінцевий результат запропонованої моделі створюється шляхом інтеграції трьох незалежних результатів і голосування за результати. Для експерименту були використані набори даних BRATS 2014, що містить 120 МРТ зображень, та BTD20, що містить 3000 МРТ, обидва набори даних розділені на дві категорії: норма і пухлина.

У роботі [61] запропоновано нову двофазну структуру, засновану на глибокому навчанні, для ідентифікації та класифікації пухлин мозку на зображеннях МРТ. Пропонується нова схема глибоко підсилених ознак і класифікаторів ансамблю (DBF-EC) для ефективного виявлення пухлин на зображеннях МРТ. Також пропонується новий підхід до класифікації пухлин

головного мозку на основі злиття ознак, що складається як зі статичних, так і динамічних ознак і класифікатора машинного навчання для класифікації різних типів пухлин. Ефективність запропонованої схеми двофазного аналізу пухлин головного мозку перевірялась на двох стандартних наборах даних, зібраних з Kaggle і Figshare, всього 5058 зображень.

Порівняння оцінок ефективності зазначених методів із запропонованим у даній роботі рішенням наведено у таблиці 3.8.

Таблиця 3.8 – Порівняння оцінок ефективності існуючих рішень із запропонованим

Технологія	Precision	Recall	Accuracy
MobileNet [24]	-	-	89%
ШНМ [27]	65%	-	65.21%
ЗНМ [27]	89%	-	89%
CNN-LSTM [5]	98.8%	98.9%	99.1%
DWAE [53]	97.4%	-	99.3%
Hybrid algorithm [54]	-	-	96,6%
Faster R-CNN based [55]	98.51%, 99.75%, 100%	-	98.31%, 99.6%, 99.82%
Depthwise Separable CNN [56]	-	-	92%
MTL [57]	-	-	98%
Inception V3 based [58]	91%	76%	84%
Transfer Learning [59]	-	-	100%
Ensemble Model (NB + SVM + KNN) [60]	-	-	98.61%, 99.13%
DL-based DBFS-EC [61]	99.91%	98.99%	99.56%
Proposed technology	98.91%, 99.2%	98.76%, 99.19%	98.88%, 99.17%

Аналізуючи результати, представлені у таблиці, можна зробити висновок, що створений у даній роботі ансамбль моделей показав досить хорошу

ефективність, але не є найкращим серед існуючих технологій. Також за результатами огляду існуючих технологій можна стверджувати, що поєднання різних технологій позитивно впливає на ефективність прогнозу і такі методи показують найкращі результати.

Висновок

Отже, у даному розділі були представлені результати роботи створеної технології, проведено експерименти на двох наборах даних (3000 та 400 зображень) та оцінено ефективність використання попередньо навчених моделей DenseNet, MobileNet і власної згорткової нейронної мережі із використанням depthwise separable convolution, а також поєднання їх прогнозів у рівному співвідношенні та з використанням ваг, рівних точності, отриманій кожною базовою моделлю. Власна модель показала нестабільні результати на великому наборі даних і не впоралась з розпізнаванням на малому наборі даних, попередньо навчені моделі показали хороші результати на обох наборах даних та в усіх експериментах. Об'єднання прогнозів моделей із використанням ваг виявилось найбільш ефективним методом і показало хороші результати в усіх експериментах. Об'єднання прогнозів моделей з використанням усереднення в більшості експериментів не поступалось ефективністю об'єднанню з використанням ваг. У ході експериментів вдалось досягти таких середніх результатів на першому наборі даних: precision – 98.91%, recall – 98.76%, accuracy – 98.88%, loss – 0.087. На другому наборі даних середні результати були наступними: precision – 99.2%, recall – 99.19, accuracy – 99.17, loss – 0.172. Створена у процесі технологія була застосована для виявлення пухлини на окремих зображеннях з набору даних, що не використовувались при навчанні, із застосуванням розробленого веб-додатку і отримані результати були успішними. Реалізовано механізм збереження зображень, що подаються на вхід системи для виконання прогнозування. Отримані у роботі результати було порівняно з існуючими технологіями.

ВИСНОВКИ

Отже, у ході виконання дипломної роботи виконано дослідження предметної області поставленої задачі, розглянуто існуючі рішення, сформовано мету та завдання роботи і вимоги до створюваної нейромережної системи.

Визначено набір даних та технології, що необхідні для роботи. Проаналізовано вибрані для дослідження методи та представлено структуру нейромережної системи. У дослідженні розглядались попередньо навчені моделі DenseNet, MobileNet і згорткова нейронна мережа із використанням depthwise separable convolution. Розглянуті моделі виступили базовими для створення ансамблю моделей. Представлено архітектуру застосованих моделей.

Розроблено нейромережну систему, що відповідає поставленим до неї вимогам. Продемонстровано роботу створеної системи та оцінено її результати. Проведено порівняння ефективності роботи вибраних для дослідження методів для розпізнавання пухлин головного мозку на зображеннях МРТ. Проаналізовано ефективність врахування оцінок точності базових моделей при побудові ансамблю моделей. Отримано систему, здатну розпізнавати пухлини на МРТ зображеннях мозку з високою точністю. У результаті створена нейромережна система дозволила отримати точність вищу за 98%, використання більшого набору даних для навчання мережі сприяє отриманню кращих результатів базовими моделями. Застосування ансамблю моделей із врахуванням точності базових моделей продемонструвало найкращі результати в усіх експериментах.

Користь від використання ансамблю моделей полягає не лише у підвищенні точності прогнозу, але і в кращій надійності такої технології, оскільки демонструє стабільно хороші результати для великого і малого наборів даних. Базові моделі відрізняються механізмом роботи, а поєднання прогнозів кількох моделей може дозволити отримати їх переваги. Використання точності базових моделей як ваг при об’єднанні їх прогнозів

дозволяє віддавати перевагу моделі, що ефективніше працює для конкретного завдання.

Значний вплив при навчанні моделей розпізнавання має набір даних, збільшення набору даних та різноманітності зображень може дозволити підвищити точність прогнозу. Адже якщо модель навчена на занадто малому наборі даних, навіть якщо показані нею результати на тестовій вибірці є хорошими, вона може не впоратись, стикнувшись із зображеннями з іншого набору. Тож поповнення набору даних для навчання моделей є важливим завданням, особливо у медичній сфері. У ході роботи створено веб-додаток, що реалізує процес розпізнавання пухлин на зображеннях МРТ головного мозку, та передбачає механізм поповнення набору даних поданими на вхід зображеннями.

Також підвищити ефективність створеної технології можливо шляхом використання більшої кількості базових моделей, а також задіяння більш різноманітних технологій. Технологія може бути розвинута шляхом проведення тестування різних моделей та об’єднання прогнозів лише найкращих із них.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Younis A. Brain Tumor Analysis Using Deep Learning and VGG-16 Ensembling Learning Approaches / A. Younis, L. Qiang, C.O. Nyatega, M.J. Adamu, H.B. Kawuwa // Applied Sciences. – 2022. – Vol. 12, No. 14/7282. https://doi.org/10.3390/app12147282
2. Brain Tumor: Statistics. [Електронний ресурс]. – Режим доступу: https://www.cancer.net/cancer-types/brain-tumor/statistics
3. Amin J. A distinctive approach in brain tumor detection and classification using MRI / J. Amin, M. Sharif, M. Yasmin, S.L. Fernandes // Pattern Recognition Letters. – 2020. – Vol. 139. – p. 118-127. https://doi.org/10.1016/j.patrec.2017.10.036
4. Khan M.S.I. Accurate brain tumor detection using deep convolutional neural network / M.S.I. Khan, A. Rahman, T. Debnath, M.R. Karim, M.K. Nasir, S.S. Band, A. Mosavi, I. Dehzangi // Computational and Structural Biotechnology Journal. – 2022. – Vol. 20. – p. 4733-4745. https://doi.org/10.1016/j.csbj.2022.08.039
5. Alsubai S. Ensemble deep learning for brain tumor detection / S. Alsubai, H.U. Khan, A. Alqahtani, M. Sha, S. Abbas, U.G. Mohammad // Frontiers in Computational Neuroscience. – 2022. – Vol. 16/1005617. https://doi.org/10.3389/fncom.2022.1005617
6. Alqudah A.M. Brain Tumor Classification Using Deep Learning Technique - A Comparison between Cropped, Uncropped, and Segmented Lesion Images with Different Sizes / A.M. Alqudah, H. Alquraan, I. Abu-Qasmieh, A. Alqudah, W. Al-Sharu // International Journal of Advanced Trends in Computer Science and Engineering. – 2019. – Vol. 8, No. 6. – p. 3684-3691. https://doi.org/10.30534/ijatcse/2019/155862019
7. Що таке нейрон? [Електронний ресурс]. – Режим доступу: https://dovidka.biz.ua/shho-take-neyron/

8. Навчання нейронних мереж. [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/5461803/
9. Боярчук О.Д. Анатомія та еволюція нервової системи : підручник / О.Д. Боярчук — Луганськ: Видавництво ДЗ «ЛНУ імені Тараса Шевченка», 2014. – 395 с. http://anatomy.luguniv.edu.ua/ukr_studies/anatomy_NS_tutorial.pdf
10. Підручник Біологія 8 клас [Електронний ресурс]: Урок 38 / О.В. Костильов, С.П. Яценко. – Режим доступу: http://kvs2110.at.ua/index/urok_38_budova_viddiliv_golovnogo_mozku/0-1588
11. Кутковецький В.Я. Розпізнавання образів : навчальний посібник / В.Я. Кутковецький. – Миколаїв: Видавництво ЧНУ ім. Петра Могили, 2017. – 420 с. – ISBN 978-966-336-384-4. https://dspace.chmnu.edu.ua/jspui/handle/123456789/60
12. Довбиш А.С. Основи теорії розпізнавання образів : навчальний посібник / А.С. Довбиш, І.В. Шелехов. – Суми: Сумський державний університет, 2015. – Ч. 1. – 109 с. ISBN 987-966-657-596-1. http://kist.ntu.edu.ua/textPhD/tro2.pdf
13. Моторна Я.С. Реалізація та дослідження алгоритму random forest для розв'язування задач класифікації / Я.С. Моторна, Н.О. Красношлик, О.В. Піскун // Вісник Черкаського університету: прикладна математика. Інформатика. – 2020. – No. 1. https://ami-ejournal.cdu.edu.ua/issue/view/308/291
14. Біла Н.І. Лабораторні роботи з інформаційних систем та технологій в управлінні / Н.І. Біла. – Запоріжжя: ЗНТУ, 2014. – Ч. 3. – 50 с. http://eir.zntu.edu.ua/handle/123456789/342
15. Лекція 1. Основні поняття розпізнавання образів. [Електронний ресурс]. – Режим доступу: http://om.univ.kiev.ua/users_upload/15/upload/file/pr_lecture_01.pdf

16. Снитюк В.Е. Прогнозирование. Модели, методы, алгоритмы: учебное пособие / В.Е. Снитюк. – К.: «Маклаут», 2008. – 364 с. https://www.researchgate.net/publication/281110176_PROGNOZIROVANIE_Modeli_Metody_Algoritmy

17. Vabres P. Postzygotic inactivating mutations of RHOA cause a mosaic neuroectodermal syndrome / P. Vabres, A. Sorlin, S.S. Kholmanskikh, B. Demeer, J. St-Onge, Y. Duffourd, P. Kuentz, J.B. Courcet, V. Carmignac, P. Garret, D. Bessis, D. Boute, A. Bron, G. Captier, G. Carmi, B. Devauchelle, D. Geneviève, C. Gondry-Jouet, L. Guibaud, J.B. Rivière // Nature Genetics. – 2019. – Vol. 51. – p. 1438-1441. doi:10.1038/s41588-019-0498-4. https://www.researchgate.net/publication/336148130_Postzygotic_inactivating_mutations_of_RHOA_cause_a_mosaic_neuroectodermal_syndrome

18. Tomasila G. MRI image processing method on brain tumors: A review / G. Tomasila, A.W. Rahardjo Emanuel // AIP Conference Proceedings. – 2020. – Vol. 2296. – p. 20023. https://doi.org/10.1063/5.0030978

19. Jia H. Atlas registration and ensemble deep convolutional neural network-based prostate segmentation using magnetic resonance imaging / H. Jia, Y. Xia, Y. Song, W. Cai, M. Fulham, D.D. Feng // Neurocomputing. – 2018. – Vol. 275. – p. 1358-1369. https://doi.org/10.1016/j.neucom.2017.09.084

20. Joshi S.R. Classification of Brainwaves Using Convolutional Neural Network / S.R. Joshi, D.B. Headley, K.C. Ho, D. Paré, S.S. Nair // 27th European Signal Processing Conference (EUSIPCO). – 2019. doi:10.23919/eusipco.2019.8902952. https://www.eurasip.org/Proceedings/Eusipco/eusipco2019/Proceedings/papers/1570529424.pdf

21. Mohsen H. Classification using deep learning neural networks for brain tumors / H. Mohsen, E.-S.A. El-Dahshan, El.-S.M. El-Horbaty, A.-B.M. Salem // Future Computing and Informatics Journal. – 2018. – Vol. 3, No. 1. – p. 68-71. https://doi.org/10.1016/j.fcij.2017.12.001

22. Seetha J. Brain Tumor Classification Using Convolutional Neural Networks / J. Seetha, R. Selvakumar // Biomedical and Pharmacology Journal. – 2018. – Vol. 11. – p. 1457-1461. https://dx.doi.org/10.13005/bpj/1511

23. Alqudah A.M. AOCT-NET: a convolutional network automated classification of multiclass retinal diseases using spectral-domain optical coherence tomography images / A.M. Alqudah // Medical & biological engineering & computing. – 2020. – Vol. 58. – p. 41-53. https://doi.org/10.1007/s11517-019-02066-y

24. Santos D. Brain Tumor Detection Using Deep Learning / D.Santos, E. Santos. – 2022. https://doi.org/10.1101/2022.01.19.22269457

25. Milletari F. Hough-CNN: Deep Learning for Segmentation of Deep Brain Regions in MRI and Ultrasound / F. Milletari, S.A. Ahmadi, C. Kroll, A. Plate, V. Rozanski, J. Maiostre, J. Levin, O. Dietrich, B. Ertl-Wagner, K. Bötzel, N. Navab // Computer Vision and Image Understanding. – 2017. – Vol. 164. – p. 92-102. https://doi.org/10.48550/arXiv.1601.07014

26. Özyurt F. Brain tumor detection based on Convolutional Neural Network with neutrosophic expert maximum fuzzy sure entropy / F. Özyurt, E. Sert, E. Avci, E. Dogantekin. – 2019. – Vol. 147. https://doi.org/10.1016/j.measurement.2019.07.058

27. Gokila Brindha P. Brain tumor detection from MRI images using deep learning techniques / P. Gokila Brindha, M. Kavinraj, P. Manivasakam, P. Prasanth // IOP Conference Series: Materials Science and Engineering. — 2021. – Vol. 1055/012115. https://doi.org/10.1088/1757-899X/1055/1/012115

28. Anaraki A.K. Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms / A.K. Anaraki, M. Ayati, F. Kazemi // Biocybernetics and Biomedical Engineering. – 2019. – Vol. 39, No. 1. – p. 63-74. https://doi.org/10.1016/j.bbe.2018.10.004

29. Swati Z.N.K. Brain tumor classification for mr images using transfer learning and fine-tuning / Z.N.K. Swati, Q. Zhao, M. Kabir, F. Ali, Z. Ali, S. Ahmed, J. Lu // Computerized Medical Imaging and Graphics. – 2019. – Vol. 75. – p. 34-46. https://doi.org/10.1016/j.compmedimag.2019.05.001

30. Sajjad M. Multi-grade brain tumor classification using deep cnn with extensive data augmentation / M. Sajjad, S. Khan, K. Muhammad, W. Wu, A. Ullah, S.W. Baik // Journal of Computational Science. – 2019. – Vol 30. – p. 174-182. https://doi.org/10.1016/j.jocs.2018.12.003

31. Díaz-Pernas F.J. A Deep Learning Approach for Brain Tumor Classification and Segmentation Using a Multiscale Convolutional Neural Network / F.J. Díaz-Pernas, M. Martínez-Zarzuela, M. Antón-Rodríguez, M. González-Ortega // Healthcare. – 2021. Vol. 9, No. 2/153. https://doi.org/10.3390%2Fhealthcare9020153

32. Irmak E. Multi-Classification of Brain Tumor MRI Images Using Deep Convolutional Neural Network with Fully Optimized Framework / E. Irmak // Iranian Journal of Science and Technology, Transactions of Electrical Engineering. – 2021. – Vol. 45. – p. 1015-1036. https://doi.org/10.1007/s40998-021-00426-9

33. Kang J. MRI-Based Brain Tumor Classification Using Ensemble of Deep Features and Machine Learning Classifiers / J. Kang, Z. Ullah, J. Gwak // Sensors. – 2021. – Vol. 21, No. 6, Article 2222. https://doi.org/10.3390/s21062222

34. Рєпка В.Б. Основні положення теорії штучних нейронних мереж / В.Б. Рєпка. – Розділ 1. – 2010. – Режим доступу: https://dl.nure.ua/pluginfile.php/634/mod_resource/content/2/001.pdf

35. Нейронні системи [Електронний ресурс] : Конспект лекцій / В.М. Коцовський // ДВНЗ "Ужгородський національний університет". – Ужгород. – 2013. Режим доступу: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/16450/1/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%96%20%D1%81%D

0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8.%20%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%97.pdf

36. Навчання нейронних мереж. [Електронний ресурс]. – Режим доступу: https://helpiks.org/5-72196.html
37. Кушнір Н.О. Використання згорткових нейронних мереж у задачах розпізнавання та класифікації об’єктів зображень / Н.О. Кушнір, Т.М. Локтікова, Т.М. Морозов, Т.М. Юрченко // Державний університет «Житомирська політехніка». – 2022. http://eztuir.ztu.edu.ua/123456789/8007
38. Howard A. G. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications / A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam. – 2017. https://arxiv.org/abs/1704.04861
39. Phiphiphatphaisit S. Food Image Classification with Improved MobileNet Architecture and Data Augmentation / S. Phiphiphatphaisit, S. Surinta // 3rd International Conference on Information Science and Systems (ICISS '20). – 2020. – p. 51-56. https://doi.org/10.1145/3388176.3388179
40. Niu Q. Design of gesture recognition system based on Deep Learning / Q. Niu, Y. Teng, L. Chen // Journal of Physics: Conference Series. – 2019. – Vol. 1168, No. 3. https://iopscience.iop.org/article/10.1088/1742-6596/1168/3/032082
41. Michele A. MobileNet Convolutional Neural Networks and Support Vector Machines for Palmprint Recognition / A. Michele, V. Colin, D. D. Santika. – 2019. – Vol. 157. – p. 110-117. https://doi.org/10.1016/j.procs.2019.08.147
42. Huang G. Densely Connected Convolutional Networks / G. Huang, Z. Liu, K.Q. Weinberger // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – p. 2261-2269. https://doi.org/10.48550/arXiv.1608.06993

43. Wang W. A Novel Image Classification Approach via Dense-MobileNet Models / W. Wang, Y. Li, T. Zou, X. Wang, J. You, Y. Luo // Mobile Information Systems. – 2020. https://doi.org/10.1155/2020/7602384
44. Ensemble Methods in Python. [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/ensemble-methods-in-python/
45. Kotu V. Predictive Analytics and Data Mining / V. Kotu, B. Deshpande. – Chapter 2. – 2015. https://doi.org/10.1016/B978-0-12-801460-8.00002-1
46. A Comprehensive Guide to Ensemble Learning. [Електронний ресурс]. – Режим доступу: https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/
47. Ensemble Modeling Tutorial: Explore Ensemble Learning Techniques. [Електронний ресурс]. – Режим доступу: https://www.datacamp.com/tutorial/ensemble-learning-python
48. Ensemble Methods [Електронний ресурс]. – Режим доступу: https://corporatefinanceinstitute.com/resources/data-science/ensemble-methods/
49. Создание веб-сайтов (HTML, CSS, PHP, MySQL, JavaScript, JQuery). 2017 [Електронний ресурс]. – Режим доступу: http://www.omis.ee/wp-content/uploads/2017/04/Создание-веб-сайтов-HTML-CSS-PHP-MySQL-JavaScript-JQuery.pdf
50. Старушенкова Е. Е. Язык SQL как средство создания баз данных: текст научной статьи по специальности «Компьютерные и информационные науки» / Е.Е. Старушенкова, Е.Н. Шиманова, К.Д. Радаев. https://cyberleninka.ru/article/n/yazyk-sql-kak-sredstvo-sozdaniya-baz-dannyh
51. Каширина И.Л. Исследование и сравнительный анализ методов оптимизации, используемых при обучении нейронных сетей / И.Л. Каширина, М.В. Демченко // Вестник ВГУ. Серия: Системный анализ и информационные технологии. – 2018. – №4. – с. 123-132.

https://doi.org/10.17308/sait.2018.4/1262

52. Front end(Фронтенд) - что это? Все что нужно знать [Електронний ресурс]. – Режим доступу: https://dan-it.com.ua/blog/razrabotka-so-storony-front-end-chto-jeto-takoe-i-chem-otlichaetsja-ot-back-end/

53. El Kader I.A. Brain Tumor Detection and Classification on MR Images by a Deep Wavelet Auto-Encoder Model / I.A. El Kader, G. Xu, Z. Shuai, S. Saminu, I. Javaid, I.S. Ahmad, S. Kamhi // Diagnostics. – 2021. – Vol. 11, No. 9/1589. https://doi.org/10.3390/diagnostics11091589

54. Saad G. Developing a hybrid algorithm to detect brain tumors from MRI images / G. Saad, A. Suliman, L. Bitar, S. Bshara // Egyptian Journal of Radiology and Nuclear Medicine. – 2023. – No. 54, Article 14. https://doi.org/10.1186/s43055-023-00962-w

55. Yilmaz A. Brain tumor detection from MRI images with using proposed deep learningmodel: the partial correlation-based channel selection // Turkish Journal of Electrical Engineering and Computer Sciences. – 2021. – Vol. 29, No. 8, Article 3. https://doi.org/10.3906/elk-2103-37

56. Kavita B. Brain Tumor Detection Using Deep Learning Techniques / B. Kavita, R. Varun, S. Sanjay, S. Vijay // 4th International Conference on Advances in Science & Technology (ICAST2021). – 2021. http://dx.doi.org/10.2139/ssrn.3867216

57. Nazir M. Multi-task learning architecture for brain tumor detection and segmentation in MRI images / M. Nazir, M.J. Ali, H.Z. Tufail, A.R. Shahid, B. Raza, S. Shakil, K. Khurshid // Journal of Electronic Imaging. – 2022. – Vol. 31, No. 5/051606. https://doi.org/10.1117/1.JEI.31.5.051606

58. Indraswari R. Brain Tumor Detection on Magnetic Resonance Imaging (MRI) Images Using Convolutional Neural Network (CNN) / R. Indraswari, I.S. Ardan, A.Z. Arifin, A. Tjahyanto, N.A. Rakhmawati, R. Kusumawardani // 9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI). – 2022. – p. 367-373, https://doi.org/10.23919/EECSI56542.2022.9946622

59. Bayoumi E. S. Brain Tumor Automatic Detection from MRI Images Using Transfer Learning Model with Deep Convolutional Neural Network / E. S. Bayoumi, M.K. Abd-Ellah, A.A.M. Khalaf, R.R. Gharieb. – 2022. – Vol. 41, No.2. https://doi.org/10.21608/jaet.2020.42896.1051

60. Ghafourian E. An Ensemble Model for the Diagnosis of Brain Tumors through MRIs / E. Ghafourian, F. Samadifam, H. Fadavian, P. Jerfi Canatalay, A. Tajally, S. Channumsin // Diagnostics. – 2023. – Vol. 13, No. 3, Article 561. https://doi.org/10.3390/diagnostics13030561

61. Zahoor M.M. A New Deep Hybrid Boosted and Ensemble Learning-Based Brain Tumor Analysis Using MRI / M.M. Zahoor, S.A. Qureshi, S. Bibi, S.H. Khan, A. Khan, U. Ghafoor, M.R. Bhutta // Sensors. – 2022. – Vol. 22, No. 7, Article 2726. https://doi.org/10.3390/s22072726

ДОДАТОК А

Графіки ансамблів моделей, використаних у роботі

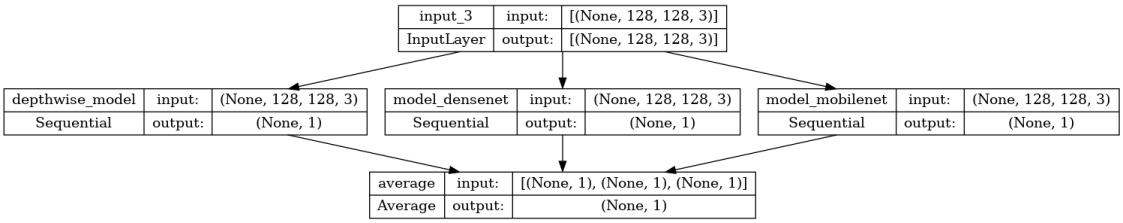


Рисунок А.1 – Графік ансамблю моделей з використанням шару усереднення

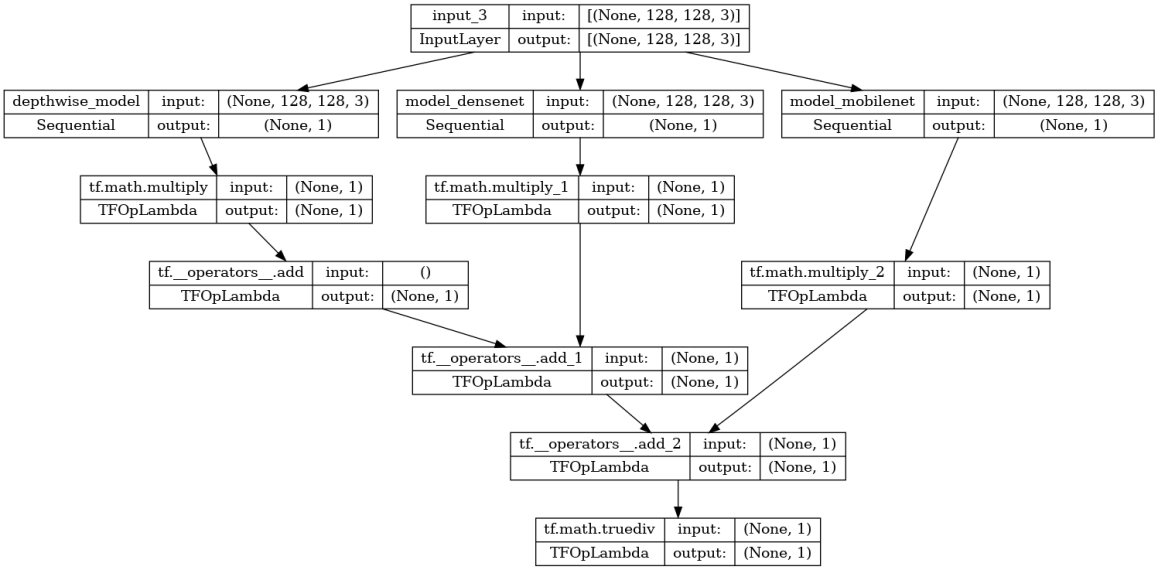


Рисунок А.2 – Графік ансамблю моделей з використання ваг

ДОДАТОК Б

Графіки змін метрик та функції втрат для власної моделі у першому експерименті

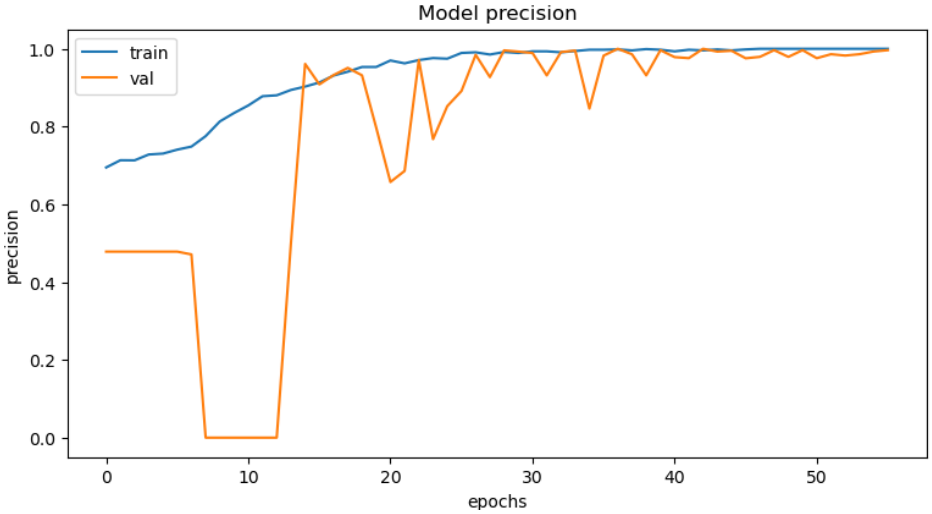


Рисунок Б.1 – Графік зміни показника prescision для власної моделі

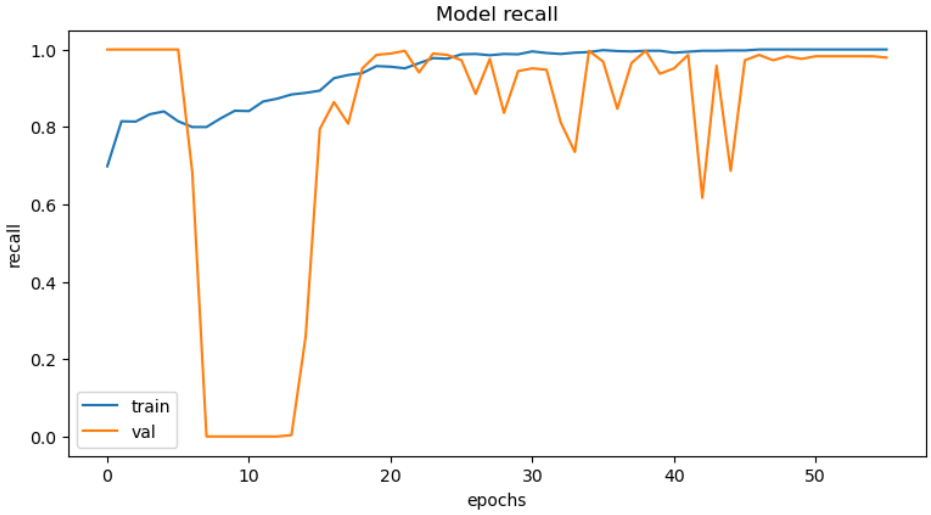


Рисунок Б.2 – Графік зміни показника recall для власної моделі

ДОДАТОК Б (продовження)

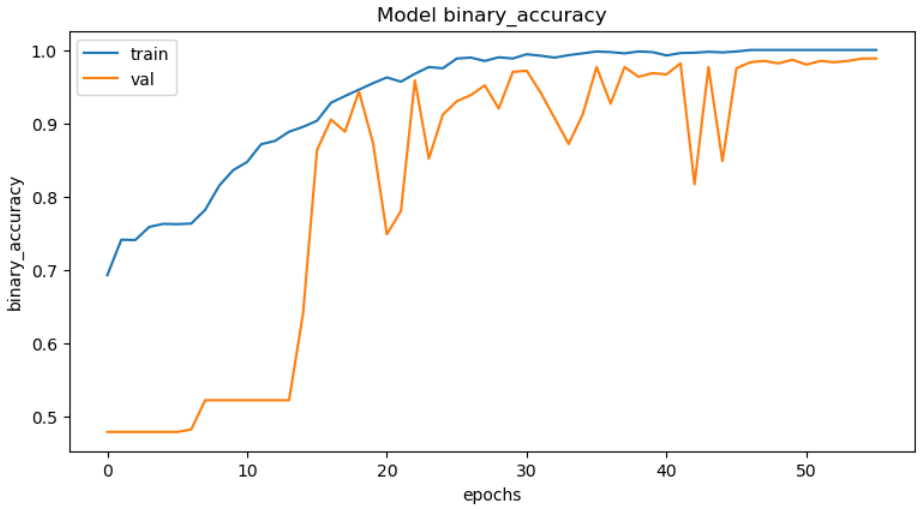


Рисунок Б.3 – Графік зміни показника accuracy для власної моделі

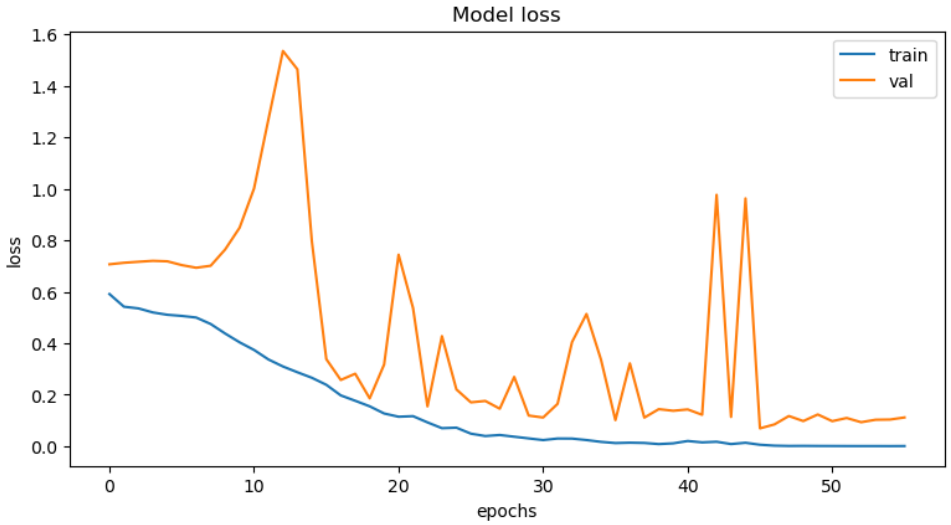


Рисунок Б.4 – Графік зміни показника loss для власної моделі

ДОДАТОК В

Графіки змін метрик та функції втрат для моделі DenseNet у першому експерименті

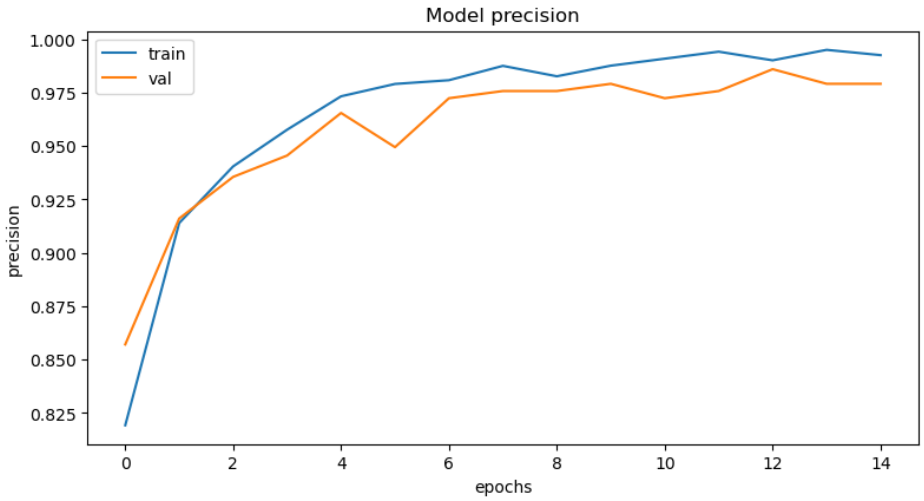


Рисунок В.1 – Графік зміни показника prescision для моделі DenseNet

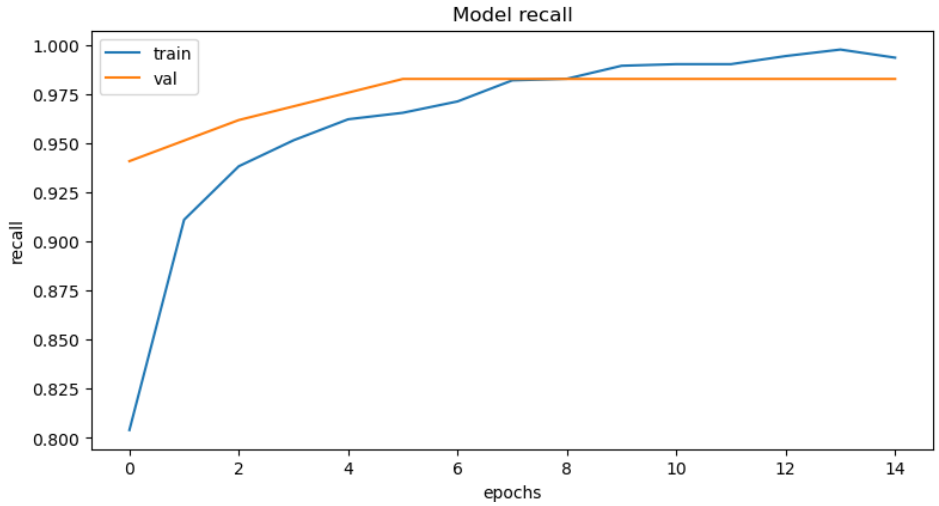


Рисунок В.2 – Графік зміни показника recall для моделі DenseNet

ДОДАТОК В (продовження)

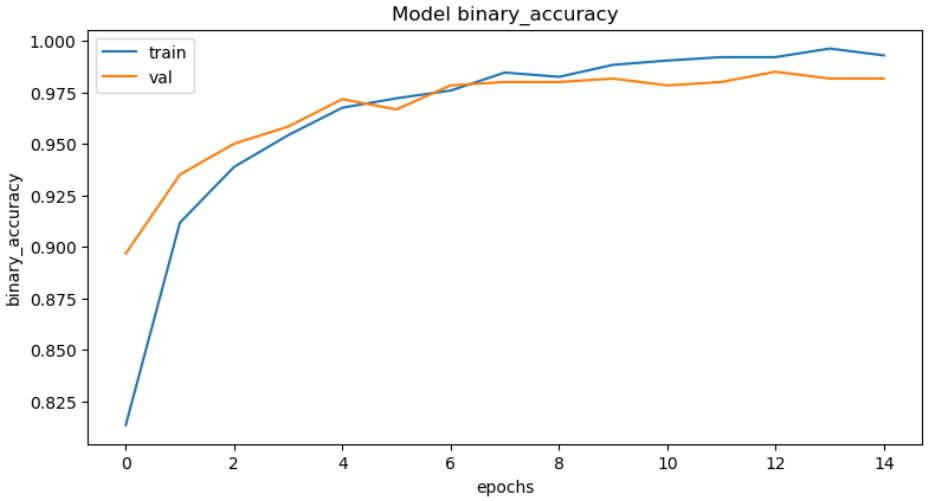


Рисунок В.3 – Графік зміни показника accuracy для моделі DenseNet

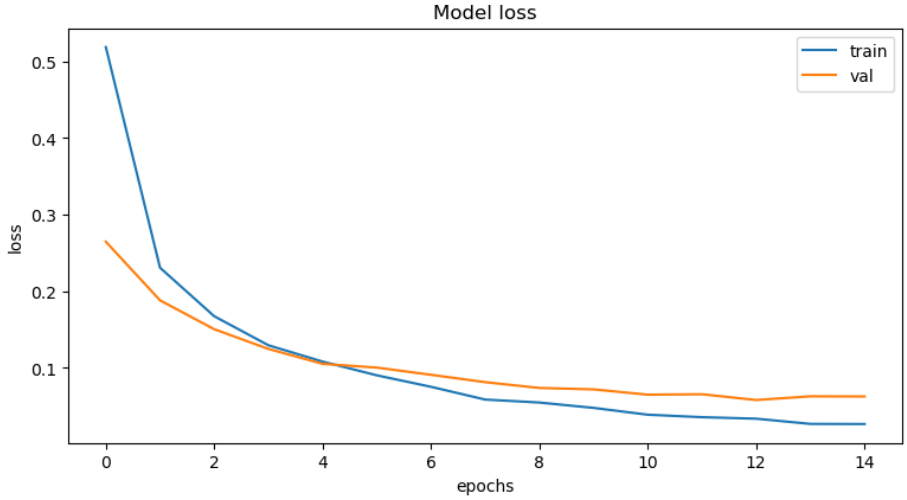


Рисунок В.4 – Графік зміни показника loss для моделі DenseNet

ДОДАТОК Г

Графіки змін метрик та функції втрат для моделі MobileNet у першому експерименті

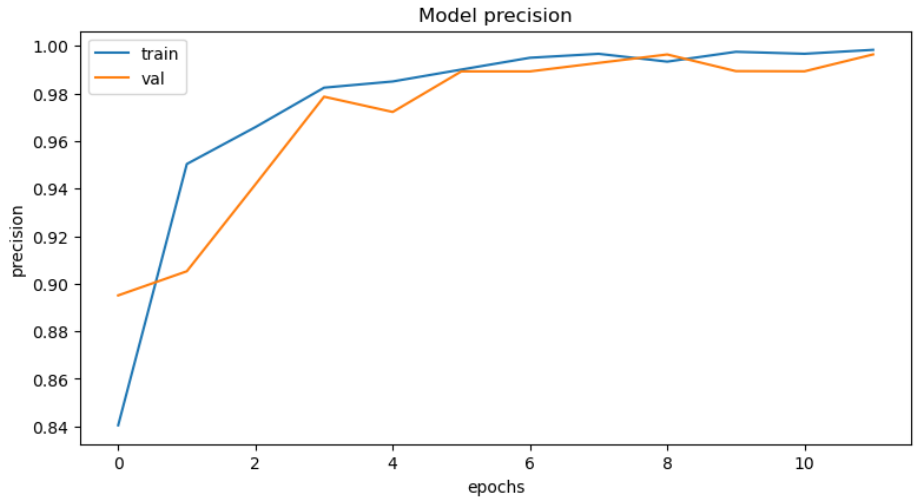


Рисунок Г.1 – Графік зміни показника prescision для моделі MobileNet

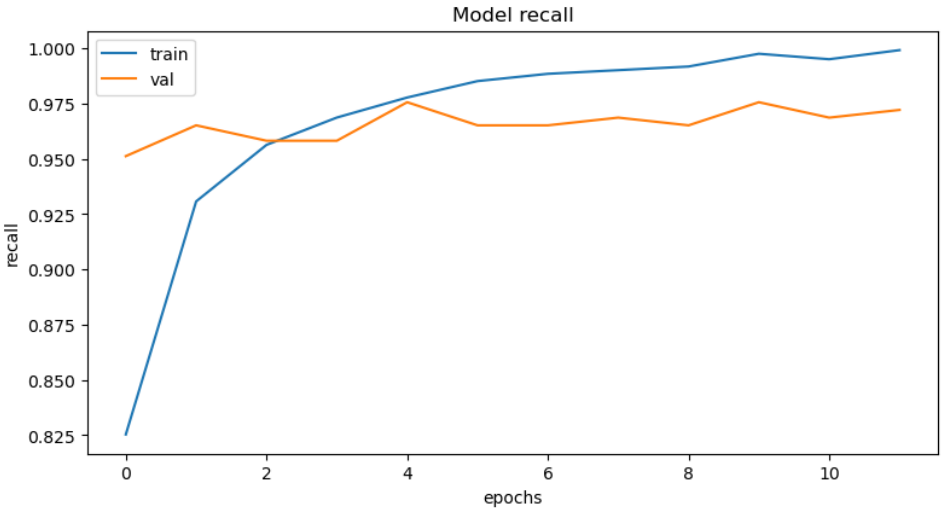


Рисунок Г.2 – Графік зміни показника recall для моделі MobileNet

ДОДАТОК Г (продовження)

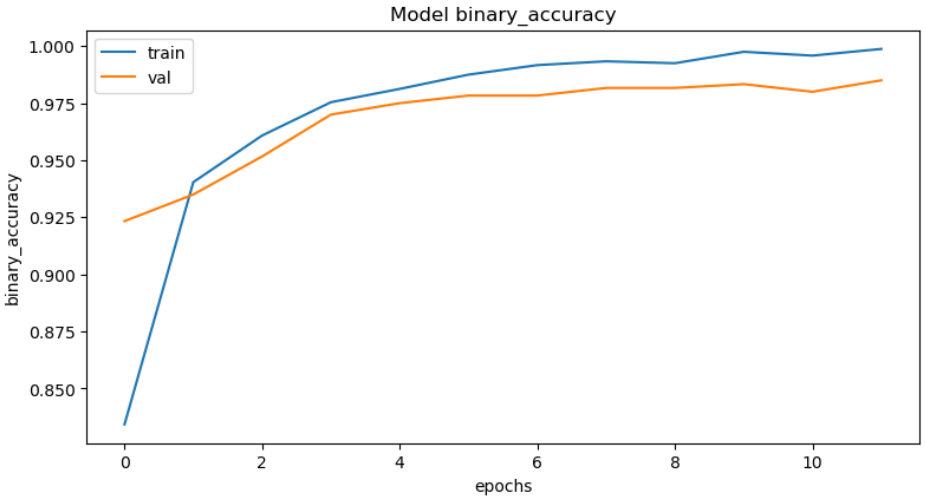


Рисунок Г.3 – Графік зміни показника accuracy для моделі MobileNet

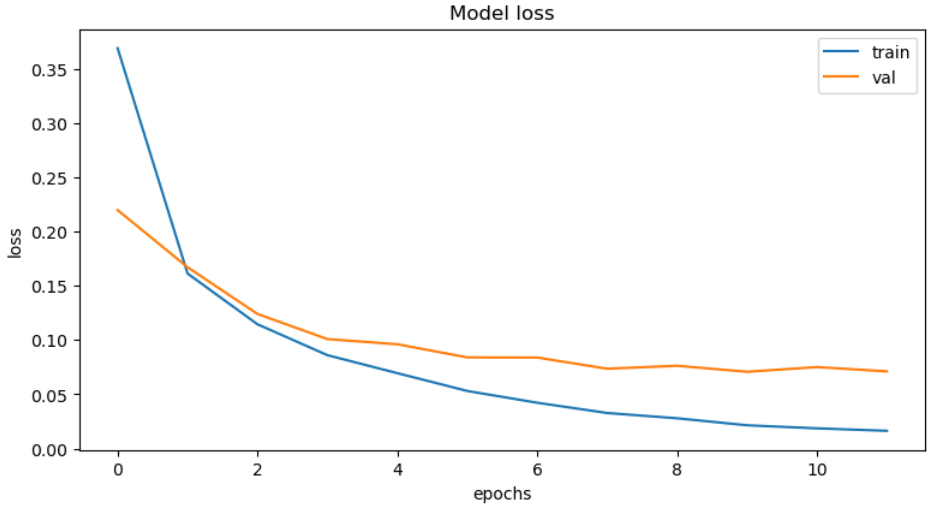


Рисунок Г.4 – Графік зміни показника loss для моделі MobileNet

ДОДАТОК Д

Графіки змін метрик та функції втрат для власної моделі у другому експерименті

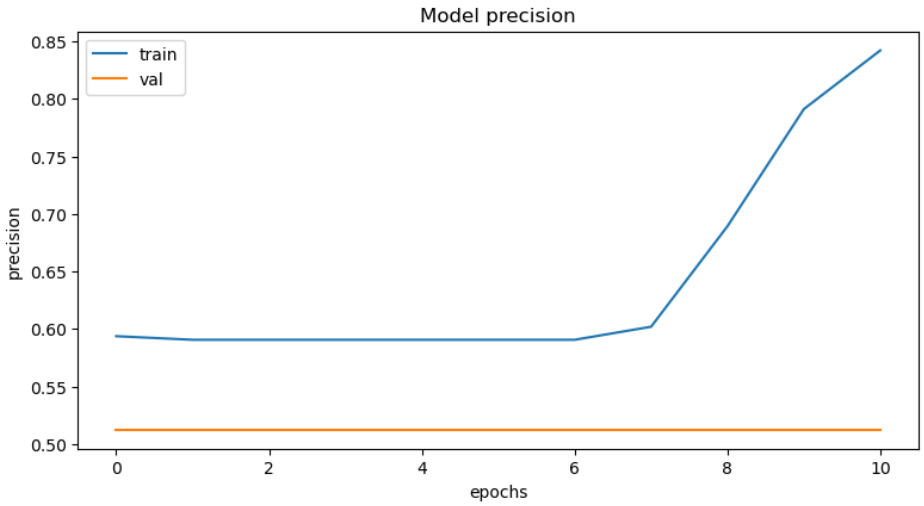


Рисунок Д.1 – Графік зміни показника prescision для власної моделі

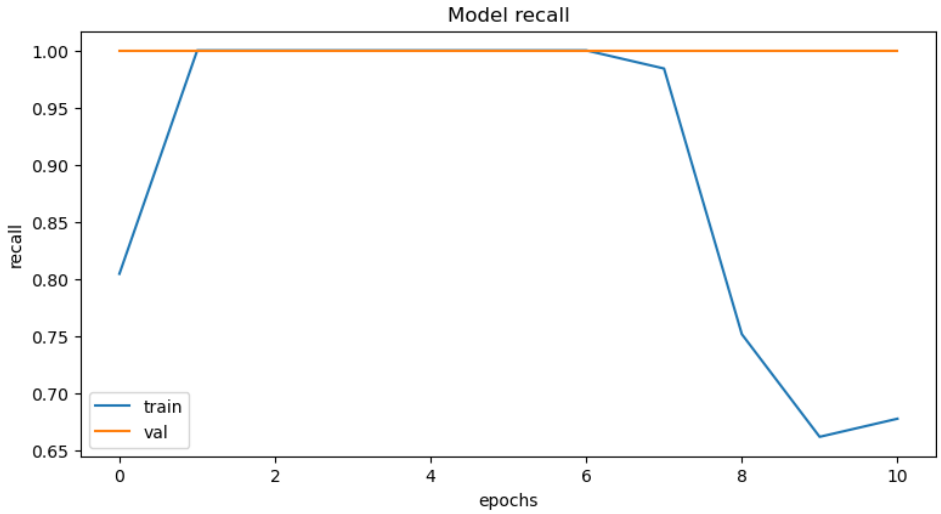


Рисунок Д.2 – Графік зміни показника recall для власної моделі

ДОДАТОК Д (продовження)

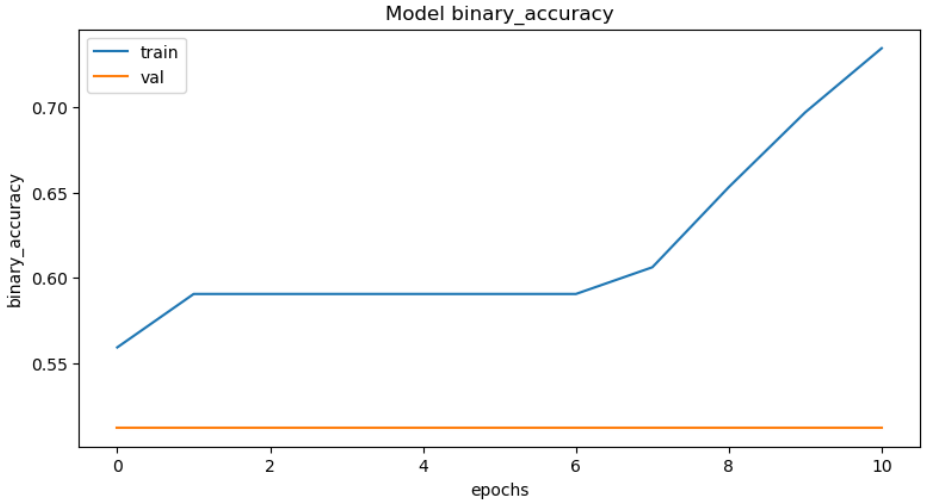


Рисунок Д.3 – Графік зміни показника accuracy для власної моделі

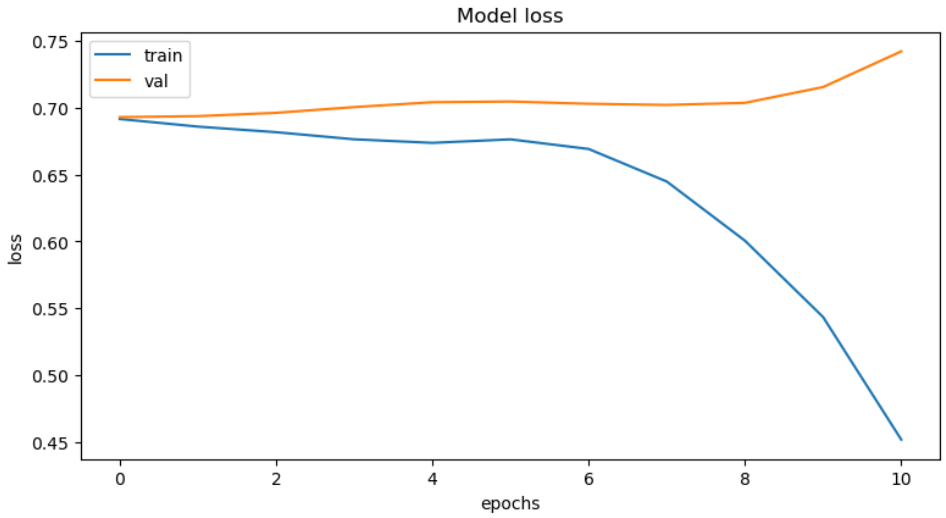


Рисунок Д.4 – Графік зміни показника loss для власної моделі

ДОДАТОК Е

Графіки змін метрик та функції втрат для моделі DenseNet у другому експерименті

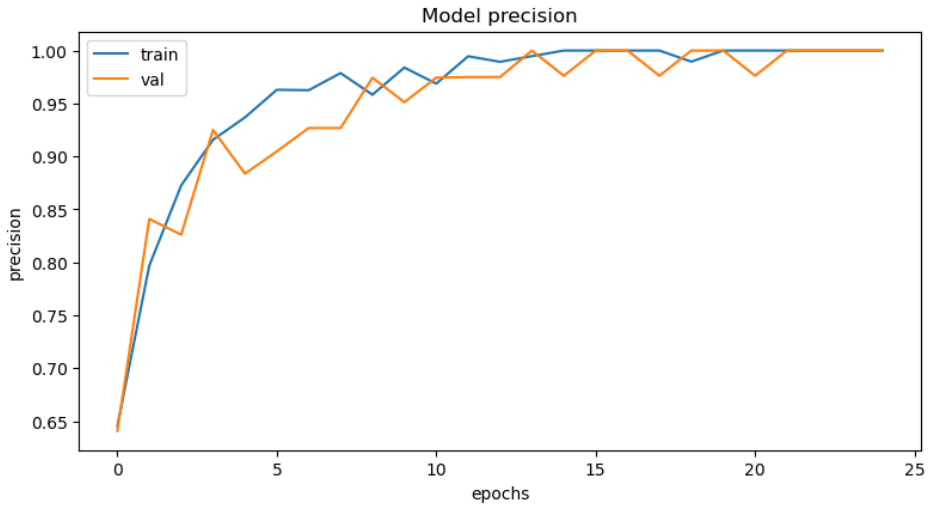


Рисунок Е.1 – Графік зміни показника prescision для моделі DenseNet

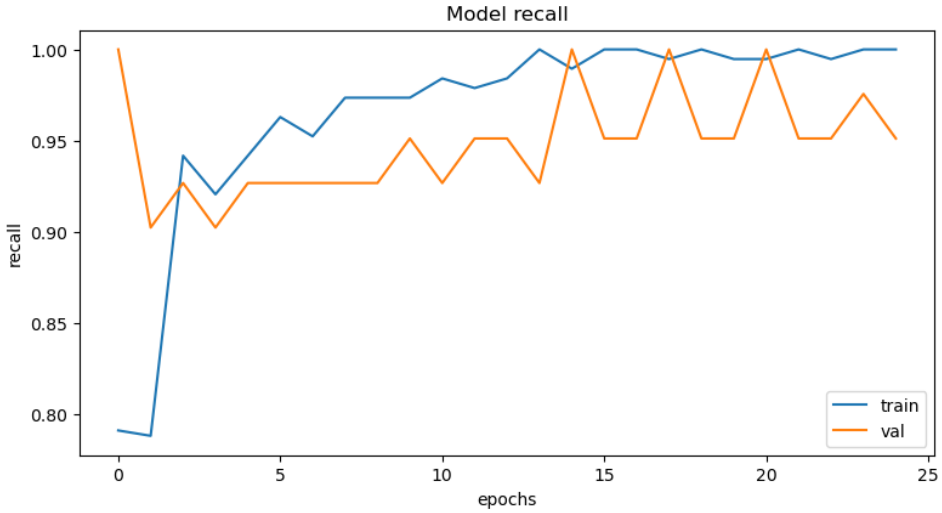


Рисунок Е.2 – Графік зміни показника recall для моделі DenseNet

ДОДАТОК Е (продовження)

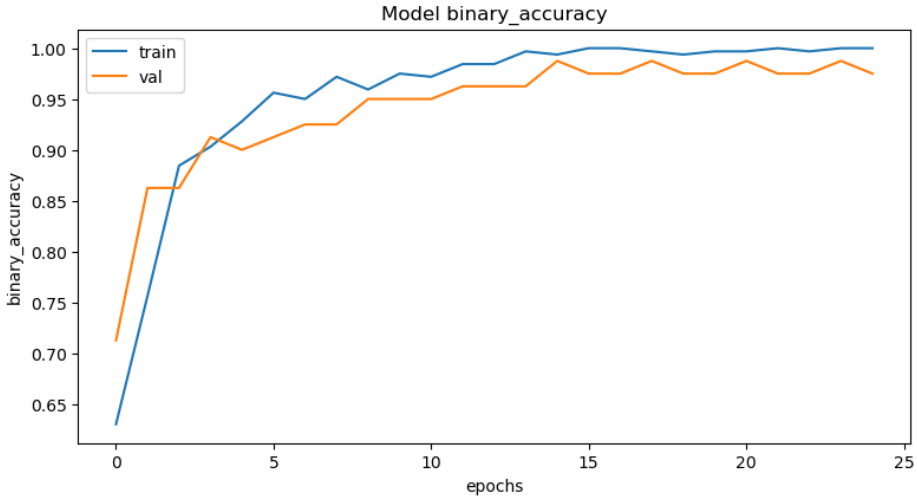


Рисунок Е.3 – Графік зміни показника accuracy для моделі DenseNet

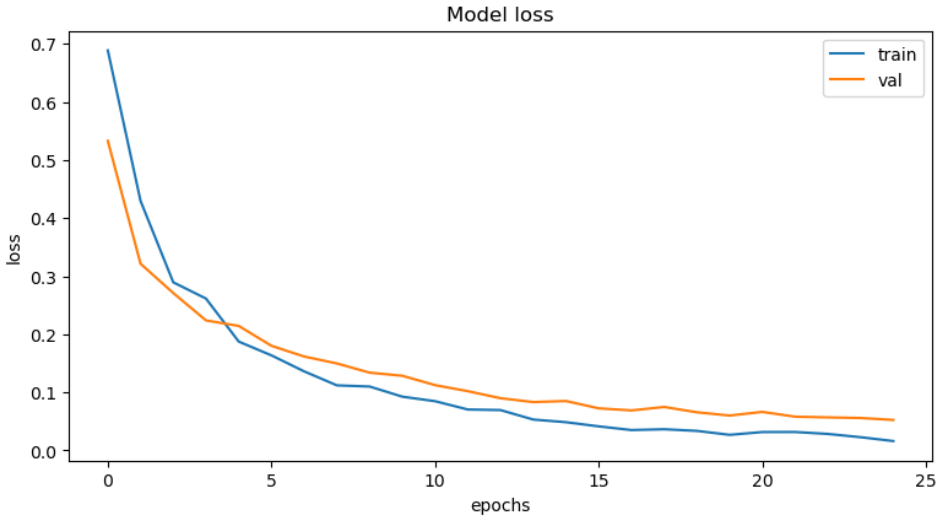


Рисунок Е.4 – Графік зміни показника loss для моделі DenseNet

ДОДАТОК Ж

Графіки змін метрик та функції втрат для моделі MobileNet у другому експерименті

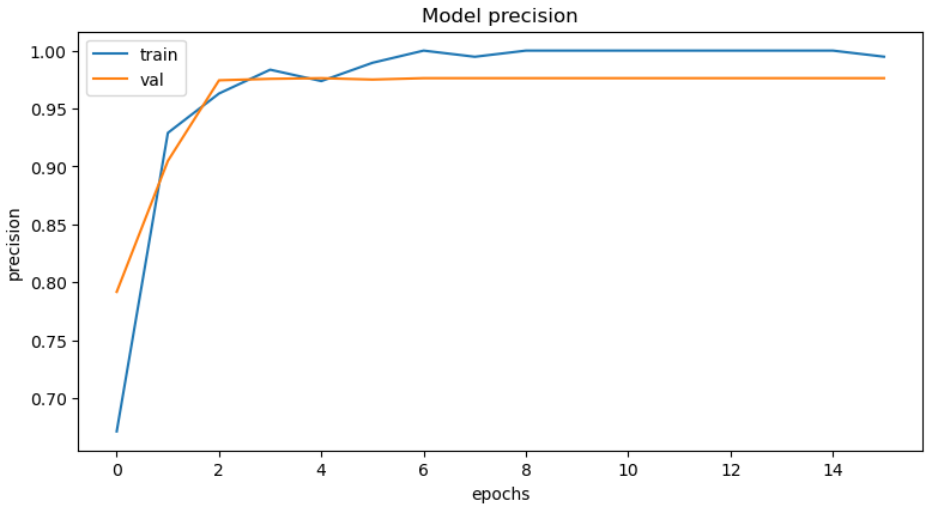


Рисунок Ж.1 – Графік зміни показника prescision для моделі MobileNet

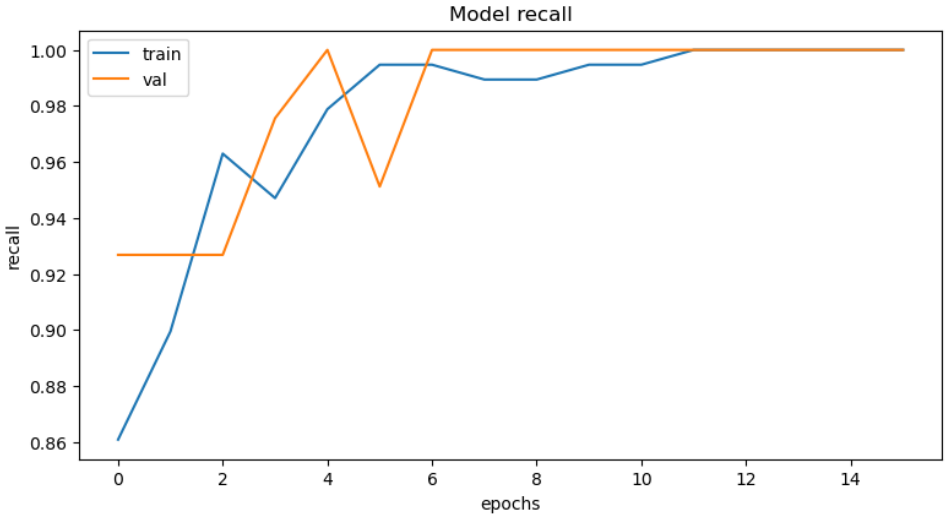


Рисунок Ж.2 – Графік зміни показника recall для моделі MobileNet

ДОДАТОК Ж (продовження)

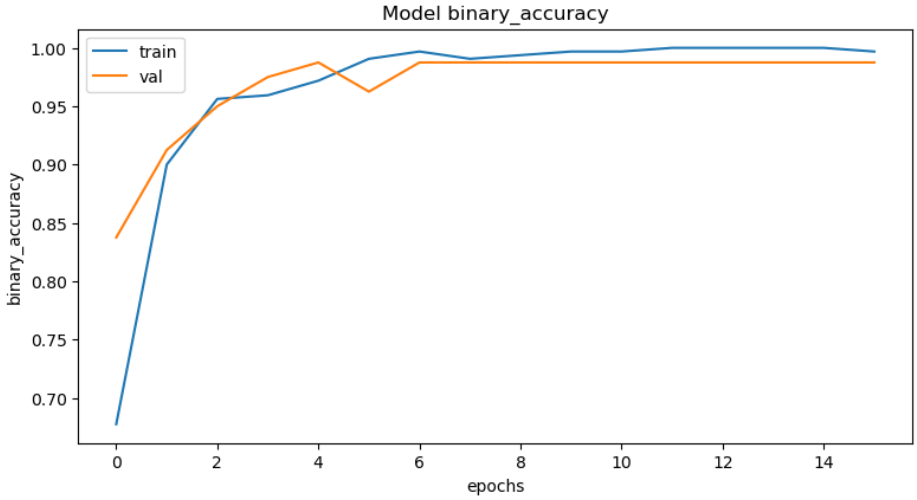


Рисунок Ж.3 – Графік зміни показника accuracy для моделі MobileNet

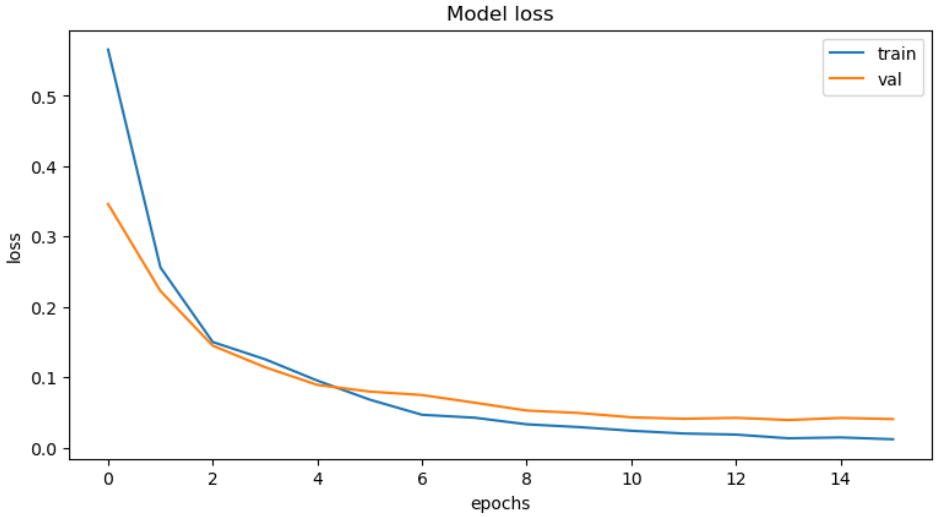


Рисунок Ж.4 – Графік зміни показника loss для моделі MobileNet

ДОДАТОК И

Лістинг програми використаної для проведення експериментів з оцінки ефективності моделей нейронних мереж

```
import numpy as np
import pandas as pd
import os, cv2
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
import shutil
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score, log_loss
from sklearn.metrics import ConfusionMatrixDisplay
from numpy import savetxt
try:
  tpu = tf.distribute.cluster_resolver.TPUClusterResolver.connect()
  print("Device:", tpu.master())
  strategy = tf.distribute.TPUStrategy(tpu)
except:
  strategy = tf.distribute.get_strategy()
print("Number of replicas:", strategy.num_replicas_in_sync)

# Передобробка даних
# копіюємо потрібні папки в окрему частину, щоб їх можна було зручно прочитати в майбутньому
shutil.copytree("/kaggle/input/brain-tumor-detection-mri/Brain_Tumor_Detection/no", "data/no")
shutil.copytree("/kaggle/input/brain-tumor-detection-mri/Brain_Tumor_Detection/yes", "data/yes")
DIRECTORY = '/kaggle/working/data'
def get_images(data_path):
  img_width, img_height = 128, 128
  k = 0
  # рахуємо кількість зображень у наборі даних
  for sub_dir in os.listdir(data_path):
    target_dir = data_path + '/' + sub_dir
    files = os.listdir(target_dir)
    count = len(files)
    k += count
  # створюємо порожній список для зберігання зображень
  images = np.ndarray((k,128,128,3),dtype=np.float32)
  # створюємо порожній список для зберігання міток
  labels = np.ndarray((k), dtype=np.uint)
  # цикл по класах (припускаючи, що каталог містить один підкаталог на клас)
  for i, class_name in enumerate(sorted(os.listdir(data_path))):
    print(class_name, i)
    # створюємо шлях до підкаталогу для поточного класу
    class_path = os.path.join(data_path, class_name)
    l=k-len(os.listdir(class_path))
    # переходимо до зображень у підкаталозі
    for j, img_name in enumerate(os.listdir(class_path)):
      # створюємо шлях до поточного зображення
```

```
            img_path = os.path.join(class_path, img_name)
            # відкриваємо зображення та змін.ємо його до потрібних розмірів
            img = cv2.imread(img_path, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, (img_width, img_height))

            if img.shape != (128,128,3):
               img = np.expand_dims(img, axis=2)
               img = np.concatenate([img]*3, axis=2)
            # конвертуємо зображення в масив numpy
            img_array = np.array(img) / 255
            # додаємо зображення до списку зображень
            images[(i*l) + j] = img_array
            # додаємо мітку до списку міток
            labels[(i*l) + j] = i

    return images, labels

images, labels = get_images(DIRECTORY)
# виконуємо розбиття на тренувальний та тестовий набір
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42,
shuffle=True)

# Створення моделей
# Вказуємо оцінку точностей
METRICS = [
    tf.keras.metrics.BinaryAccuracy(),
    tf.keras.metrics.Precision(name="precision"),
    tf.keras.metrics.Recall(name="recall"),
]

INPUT_SHAPE=(128, 128, 3)

# Модель MobileNet

# Імпортуємо модель
pretrained_model_mobilenet = tf.keras.applications.MobileNetV2(
    include_top=False,
    weights="imagenet",
    input_shape=INPUT_SHAPE,
)

# Сама модель не буде тренуватись
pretrained_model_mobilenet.trainable = False

# Додаємо нові шари зверху до готової моделі
model_mobilenet = keras.Sequential([
    pretrained_model_mobilenet,

    layers.GlobalAveragePooling2D(),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(1, activation='sigmoid')
])
```

```
# Компілюємо моделі
model_mobilenet.compile(optimizer = 'adam',
          loss='binary_crossentropy',
          metrics=METRICS)

model_mobilenet.summary()

# Модель DenseNet

pretrained_model_densenet = tf.keras.applications.DenseNet121(
   include_top=False,
   weights="imagenet",
   input_shape=INPUT_SHAPE,
   classifier_activation="softmax",
)

pretrained_model_densenet.trainable = False

model_densenet = keras.Sequential([
   pretrained_model_densenet,

   layers.GlobalAveragePooling2D(),
   layers.Dense(64, activation='relu'),
   layers.Dropout(0.2),
   layers.Dense(1, activation='sigmoid')
])

model_densenet.compile(optimizer = 'adam',
          loss='binary_crossentropy',
          metrics=METRICS)

model_densenet.summary()

# Модель Depthwise

depthwise_model = keras.Sequential([
   layers.Input(shape=INPUT_SHAPE, name='Input'),

   layers.Conv2D(64, (3,3), activation='relu', padding='same', name='Conv1_1'),
   layers.Conv2D(64, (3,3), activation='relu', padding='same', name='Conv1_2'),
   layers.MaxPooling2D((2,2), name='MaxPooling1'),

   layers.SeparableConv2D(128, (3,3), activation='relu', padding='same', name='Conv2_1'),
   layers.SeparableConv2D(128, (3,3), activation='relu', padding='same', name='Conv2_2'),
   layers.MaxPooling2D((2,2), name='MaxPooling2'),

   layers.SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_1'),
   layers.BatchNormalization(name='Batch_normalization3_1'),
   layers.SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_2'),
   layers.BatchNormalization(name='Batch_normalization3_2'),
   layers.MaxPooling2D((2,2), name='MaxPooling3'),
```

```
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(1, activation='sigmoid')

])

depthwise_model.compile(optimizer = 'adam',
            loss='binary_crossentropy',
            metrics=METRICS)

depthwise_model.summary()

# Тренування моделей

# Функція для виведення результатів
def plot_results(history):
    fig, ax = plt.subplots(2, 2, figsize=(20, 10))
    ax = ax.ravel()

    for i, met in enumerate(["precision", "recall", "binary_accuracy", "loss"]):
        ax[i].plot(history.history[met])
        ax[i].plot(history.history["val_" + met])
        ax[i].set_title("Model {}".format(met))
        ax[i].set_xlabel("epochs")
        ax[i].set_ylabel(met)
        ax[i].legend(["train", "val"])

# Додаємо попереднє закінчування для тренування
early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True)
early_stopping_cb_pretrained = tf.keras.callbacks.EarlyStopping(patience=2, restore_best_weights=True)

# Тренуємо модель Depthwise
history_depthwise_model = depthwise_model.fit(
    X_train, y_train,
    batch_size=64,
    validation_data=(X_test, y_test),
    epochs=100,
    shuffle=False,
    callbacks=[early_stopping_cb],
)
# Зберігаємо модель
depthwise_model.save('models/depthwise_model.h5')

# Виводимо результати
plot_results(history_depthwise_model)

# Тренуємо модель Densenet
history_densenet = model_densenet.fit(
    X_train, y_train,
    batch_size=64,
    validation_data=(X_test, y_test),
```

```
    epochs=25,
    shuffle=False,
    callbacks=[early_stopping_cb_pretrained],
)

model_densenet.save('models/model_densenet.h5')

# Виводимо результати
plot_results(history_densenet)

# Тренуємо модель Mobilenet
history_mobilenet = model_mobilenet.fit(
    X_train, y_train,
    batch_size=64,
    validation_data=(X_test, y_test),
    epochs=25,
    shuffle=False,
    callbacks=[early_stopping_cb_pretrained],
)

model_mobilenet.save('models/model_mobilenet.h5')

# Виводимо результати
plot_results(history_mobilenet)

# Побудова ансамблів моделей
# Читаємо готові моделі
model_one = keras.models.load_model('models/depthwise_model.h5')
model_one._name = 'depthwise_model'

model_two = keras.models.load_model('models/model_densenet.h5')
model_two._name = 'model_densenet'

model_three = keras.models.load_model('models/model_mobilenet.h5')
model_three._name = 'model_mobilenet'

# Об'єднуємо моделі з використанням усереднення передбачень
models = [model_one, model_two, model_three]

model_input = tf.keras.Input(shape=(128, 128, 3))
model_outputs = [model(model_input) for model in models]

# Ансамбль моделей з використанням усереднення
# Будуємо модель з усередненням значень передбачень
ensemble_avg_output = tf.keras.layers.Average()(model_outputs)
ensemble_model_avg = tf.keras.Model(inputs=model_input, outputs=ensemble_avg_output,
name='ensamble_avg')

# Компілюємо модель
ensemble_model_avg.compile(optimizer = 'adam',
              loss='binary_crossentropy',
              metrics=METRICS)
```

```
ensemble_model_avg.summary()

# Зберігаємо модель
ensemble_model_avg.save('models/ensemble_model_avg.h5')

tf.keras.utils.plot_model(ensemble_model_avg,
                  to_file="ensemble_model_avg.png",
                  show_shapes=True)

# Ансамбль моделей з використанням ваг
# Функція для розрахунку ваг
def get_weights(models, X_test, y_test):
   weights = []

   for model in models:
      result = model.evaluate(X_test, y_test, return_dict=True)
      weight = result['binary_accuracy']
      weights.append(weight)
   return weights

weights = get_weights(models, X_test, y_test)
savetxt('weights.csv', weights, delimiter=';')

Weighted = sum([model_outputs[i]*weights[i] for i in range(len(models))])/sum(weights)
ensemble_model_weights = tf.keras.Model(inputs=model_input, outputs=Weighted,
name='ensamble_with_weights')

ensemble_model_weights.summary()

ensemble_model_weights.compile(optimizer = 'adam',
            loss='binary_crossentropy',
            metrics=METRICS)

ensemble_model_weights.save('models/ensemble_with_weights.h5')

tf.keras.utils.plot_model(ensemble_model_weights,
                  to_file="ensemble_model_weights.png",
                  show_shapes=True)

# Оцінювання моделей
# Функція для оцінки ефективності
def show_eval_models(y_pred):

   accuracy = accuracy_score(y_test, y_pred)
   precision = precision_score(y_test, y_pred)
   recall = recall_score(y_test, y_pred)

   print(f"Binary accuracy: {accuracy}")
   print(f"Precision: {precision}")
   print(f"Recall: {recall}")

print("Depthwise:")
y_pred_depthwise = model_one.predict(X_test)
```

```
y_pred_depthwise_bin = (y_pred_depthwise > 0.5).astype(np.int_)
show_eval_models(y_pred_depthwise_bin)
loss_dep = log_loss(y_test, y_pred_depthwise.astype("float64"))
print(f"Loss: {loss_dep}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_depthwise_bin)
plt.show()

print("DenseNet:")
y_pred_densenet = model_two.predict(X_test)
y_pred_densenet_bin = (y_pred_densenet > 0.5).astype(np.int_)
show_eval_models(y_pred_densenet_bin)
loss_den = log_loss(y_test, y_pred_densenet.astype("float64"))
print(f"Loss: {loss_den}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_densenet_bin)
plt.show()

print("MobileNet:")
y_pred_mobilenet = model_three.predict(X_test)
y_pred_mobilenet_bin = (y_pred_mobilenet > 0.5).astype(np.int_)
show_eval_models(y_pred_mobilenet_bin)
loss_mob = log_loss(y_test, y_pred_mobilenet.astype("float64"))
print(f"Loss: {loss_mob}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_mobilenet_bin)
plt.show()

print("Average:")
y_pred_avg = ensemble_model_avg.predict(X_test)
y_pred_avg_bin = (y_pred_avg > 0.5).astype(np.int_)
show_eval_models(y_pred_avg_bin)
loss_avg = log_loss(y_test, y_pred_avg.astype("float64"))
print(f"Loss: {loss_avg}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_avg_bin)
plt.show()

# Виконуємо передбачення з різними вагами для моделей
print("Weighted:")
y_pred_weights = ensemble_model_weights.predict(X_test)
y_pred_weights_bin = (y_pred_weights > 0.5).astype(np.int_)
show_eval_models(y_pred_weights_bin)
loss_weights = log_loss(y_test, y_pred_weights.astype("float64"))
print(f"Loss: {loss_weights}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_weights_bin)
plt.show()

# Візуалізація результатів передбачень
length = int(len(X_test)/10)
x = list(range(length))
plt.figure(figsize=(20, 10))
plt.plot(x, y_test[:length], 'o', label="real values", mec='b', mew=1, ms=10)
plt.plot(x, y_pred_avg[:length], color='g', label="pred_avg", lw=2)
plt.plot(x, y_pred_weights[:length], 'o', label="pred_weights", mec='m', mew=1, ms=5)
plt.xlabel('Number', fontsize=14)
plt.ylabel('Prediction', fontsize=14)
```

```
plt.title('Comparison of predictions',fontsize=20)
plt.legend()

# Ручна перевірка на окремих зображеннях

# Обираємо зображення для виконання передбачення
#file_path = input('Enter a file path: ')
file_path = '/kaggle/input/mri-based-brain-tumor-images/Brain_tumor_images/Tumor/Tumor (104).jpg'

def prediction(file_path, models=models, weights=weights):
  img = cv2.imread(file_path, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img, (128, 128))
  if img.shape != (128,128,3):
    img = np.expand_dims(img, axis=2)
    img = np.concatenate([img]*3, axis=2)
  # конвертуємо зображення в масив numpy
  img_array = np.array(img) / 255
  img1 = img_array.reshape(1,128,128,3)
  y_pred = ensemble_model_weights.predict(img1)
  y_pred = (y_pred > 0.5).astype(np.int_)
  if y_pred[0] == 0:
    msg = 'No'
  else:
    msg = 'Yes'
  plt.imshow(img)
  plt.title(msg, fontweight ="bold")
  plt.show()

prediction(file_path)
```

ДОДАТОК К

Лістинг програми для поєднання у ансамбль базових моделей з точністю більшою за 95%

```
import numpy as np
import pandas as pd
import os, cv2
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
import shutil
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score, log_loss
from sklearn.metrics import ConfusionMatrixDisplay
from numpy import savetxt

try:
  tpu = tf.distribute.cluster_resolver.TPUClusterResolver.connect()
  print("Device:", tpu.master())
  strategy = tf.distribute.TPUStrategy(tpu)
except:
  strategy = tf.distribute.get_strategy()
print("Number of replicas:", strategy.num_replicas_in_sync)

# копіюємо потрібні папки в окрему частину, щоб їх можна було зручно прочитати в майбутньому
shutil.copytree("/kaggle/input/brain-tumor-detection-mri/Brain_Tumor_Detection/no", "data/no")
shutil.copytree("/kaggle/input/brain-tumor-detection-mri/Brain_Tumor_Detection/yes", "data/yes")
DIRECTORY = '/kaggle/working/data'
def get_images(data_path):
  img_width, img_height = 128, 128
  k = 0
  # рахуємо кількість зображень у наборі даних
  for sub_dir in os.listdir(data_path):
    target_dir = data_path + '/' + sub_dir
    files = os.listdir(target_dir)
    count = len(files)
    k += count
  # створюємо порожній список для зберігання зображень
  images = np.ndarray((k,128,128,3),dtype=np.float32)
  # створюємо порожній список для зберігання міток
  labels = np.ndarray((k), dtype=np.uint)
  # цикл по класах (припускаючи, що каталог містить один підкаталог на клас)
  for i, class_name in enumerate(sorted(os.listdir(data_path))):
    print(class_name, i)
    # створюємо шлях до підкаталогу для поточного класу
    class_path = os.path.join(data_path, class_name)
    l=k-len(os.listdir(class_path))
    # переходимо до зображень у підкаталозі
    for j, img_name in enumerate(os.listdir(class_path)):
      # створюємо шлях до поточного зображення
```

```
            img_path = os.path.join(class_path, img_name)
            # відкриваємо зображення та змін.ємо його до потрібних розмірів
            img = cv2.imread(img_path, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, (img_width, img_height))

            if img.shape != (128,128,3):
               img = np.expand_dims(img, axis=2)
               img = np.concatenate([img]*3, axis=2)
            # конвертуємо зображення в масив numpy
            img_array = np.array(img) / 255
            # додаємо зображення до списку зображень
            images[(i*l) + j] = img_array
            # додаємо мітку до списку міток
            labels[(i*l) + j] = i

    return images, labels

images, labels = get_images(DIRECTORY)
# виконуємо розбиття на тренувальний та тестовий набір
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42,
shuffle=True)

# Створюємо моделі

# Вказуємо оцінку точностей
METRICS = [
    tf.keras.metrics.BinaryAccuracy(),
    tf.keras.metrics.Precision(name="precision"),
    tf.keras.metrics.Recall(name="recall"),
]

INPUT_SHAPE=(128, 128, 3)

# Модель MobileNet
# Імпортуємо модель
pretrained_model_mobilenet = tf.keras.applications.MobileNetV2(
    include_top=False,
    weights="imagenet",
    input_shape=INPUT_SHAPE,
)

# Сама модель не буде тренуватись
pretrained_model_mobilenet.trainable = False

# Додаємо нові шари зверху до готової моделі
model_mobilenet = keras.Sequential([
    pretrained_model_mobilenet,

    layers.GlobalAveragePooling2D(),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(1, activation='sigmoid')
])
```

```
# Компілюємо моделі
model_mobilenet.compile(optimizer = 'adam',
          loss='binary_crossentropy',
          metrics=METRICS)

model_mobilenet.summary()

# Модель DenseNet
pretrained_model_densenet = tf.keras.applications.DenseNet121(
   include_top=False,
   weights="imagenet",
   input_shape=INPUT_SHAPE,
   classifier_activation="softmax",
)

pretrained_model_densenet.trainable = False

model_densenet = keras.Sequential([
   pretrained_model_densenet,

   layers.GlobalAveragePooling2D(),
   layers.Dense(64, activation='relu'),
   layers.Dropout(0.2),
   layers.Dense(1, activation='sigmoid')
])

model_densenet.compile(optimizer = 'adam',
          loss='binary_crossentropy',
          metrics=METRICS)

model_densenet.summary()

# Модель Depthwise
# Будуємо широку глибинну модель класифікації
depthwise_model = keras.Sequential([
   layers.Input(shape=INPUT_SHAPE, name='Input'),

   layers.Conv2D(64, (3,3), activation='relu', padding='same', name='Conv1_1'),
   layers.Conv2D(64, (3,3), activation='relu', padding='same', name='Conv1_2'),
   layers.MaxPooling2D((2,2), name='MaxPooling1'),

   layers.SeparableConv2D(128, (3,3), activation='relu', padding='same', name='Conv2_1'),
   layers.SeparableConv2D(128, (3,3), activation='relu', padding='same', name='Conv2_2'),
   layers.MaxPooling2D((2,2), name='MaxPooling2'),

   layers.SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_1'),
   layers.BatchNormalization(name='Batch_normalization3_1'),
   layers.SeparableConv2D(256, (3,3), activation='relu', padding='same', name='Conv3_2'),
   layers.BatchNormalization(name='Batch_normalization3_2'),
   layers.MaxPooling2D((2,2), name='MaxPooling3'),

   layers.GlobalAveragePooling2D(),
```

```
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(1, activation='sigmoid')

])


depthwise_model.compile(optimizer = 'adam',
            loss='binary_crossentropy',
            metrics=METRICS)

depthwise_model.summary()

# Тренування та результати
# Функція для виведення результатів
def plot_results(history):
    fig, ax = plt.subplots(2, 2, figsize=(20, 10))
    ax = ax.ravel()

    for i, met in enumerate(["precision", "recall", "binary_accuracy", "loss"]):
        ax[i].plot(history.history[met])
        ax[i].plot(history.history["val_" + met])
        ax[i].set_title("Model {}".format(met))
        ax[i].set_xlabel("epochs")
        ax[i].set_ylabel(met)
        ax[i].legend(["train", "val"])

# Додаємо попереднє закінчування для тренування
early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True)
early_stopping_cb_pretrained = tf.keras.callbacks.EarlyStopping(patience=2, restore_best_weights=True)

depthwise_acc=0
while (depthwise_acc < 0.95 ):
# Тренуємо модель
    history_depthwise_model = depthwise_model.fit(
        X_train, y_train,
        batch_size=64,
        validation_data=(X_test, y_test),
        epochs=100,
        shuffle=False,
        callbacks=[early_stopping_cb],
    )
    result = depthwise_model.evaluate(X_test, y_test, return_dict=True)
    depthwise_acc = result['binary_accuracy']
# Зберігаємо модель
depthwise_model.save('models/depthwise_model.h5')

# Виводимо результати
plot_results(history_depthwise_model)

densenet_acc=0
while (densenet_acc < 0.95 ):
```

```
    history_densenet = model_densenet.fit(
        X_train, y_train,
        batch_size=64,
        validation_data=(X_test, y_test),
        epochs=25,
        shuffle=False,
        callbacks=[early_stopping_cb_pretrained],
    )
    result = model_densenet.evaluate(X_test, y_test, return_dict=True)
    densenet_acc = result['binary_accuracy']

model_densenet.save('models/model_densenet.h5')

plot_results(history_densenet)

mobilenet_acc=0

while (mobilenet_acc < 0.95 ):
    history_mobilenet = model_mobilenet.fit(
        X_train, y_train,
        batch_size=64,
        validation_data=(X_test, y_test),
        epochs=25,
        shuffle=False,
        callbacks=[early_stopping_cb_pretrained],
    )
    result = model_mobilenet.evaluate(X_test, y_test, return_dict=True)
    mobilenet_acc = result['binary_accuracy']

model_mobilenet.save('models/model_mobilenet.h5')

plot_results(history_mobilenet)

# Об'єднання моделей у ансамблі

# Читаємо готові моделі
model_one = keras.models.load_model('models/depthwise_model.h5')
model_one._name = 'depthwise_model'

model_two = keras.models.load_model('models/model_densenet.h5')
model_two._name = 'model_densenet'

model_three = keras.models.load_model('models/model_mobilenet.h5')
model_three._name = 'model_mobilenet'

# Об'єднуємо моделі
models = [model_one, model_two, model_three]
model_input = tf.keras.Input(shape=(128, 128, 3))
model_outputs = [model(model_input) for model in models]

# Ансамбль моделей з використанням усереднення
# Будуємо модель з усередненням значень передбачень
ensemble_avg_output = tf.keras.layers.Average()(model_outputs)
```

```
ensemble_model_avg = tf.keras.Model(inputs=model_input, outputs=ensemble_avg_output,
name='ensamble_avg')

# Компілюємо модель
ensemble_model_avg.compile(optimizer = 'adam',
              loss='binary_crossentropy',
              metrics=METRICS)

ensemble_model_avg.summary()

# Зберігаємо модель
ensemble_model_avg.save('models/ensemble_model_avg.h5')

# Ансамбль моделей з використанням ваг
# Функція для розрахунку ваг
def get_weights(models, X_test, y_test):
   weights = []

   for model in models:
      result = model.evaluate(X_test, y_test, return_dict=True)
      weight = result['binary_accuracy']
      weights.append(weight)
   return weights

weights = get_weights(models, X_test, y_test)
savetxt('weights.csv', weights, delimiter=';')

Weighted = sum([model_outputs[i]*weights[i] for i in range(len(models))])/sum(weights)
ensemble_model_weights = tf.keras.Model(inputs=model_input, outputs=Weighted,
name='ensamble_with_weights')

ensemble_model_weights.summary()

ensemble_model_weights.compile(optimizer = 'adam',
             loss='binary_crossentropy',
             metrics=METRICS)

ensemble_model_weights.save('models/ensemble_with_weights.h5')

# Оцінювання моделей
# Функція для оцінки точності
def show_eval_models(y_pred):

   accuracy = accuracy_score(y_test, y_pred)
   precision = precision_score(y_test, y_pred)
   recall = recall_score(y_test, y_pred)

   print(f"Binary accuracy: {accuracy}")
   print(f"Precision: {precision}")
   print(f"Recall: {recall}")

print("Depthwise:")
y_pred_depthwise = model_one.predict(X_test)
```

```
y_pred_depthwise_bin = (y_pred_depthwise > 0.5).astype(np.int_)
show_eval_models(y_pred_depthwise_bin)
loss_dep = log_loss(y_test, y_pred_depthwise.astype("float64"))
print(f"Loss: {loss_dep}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_depthwise_bin)
plt.show()

print("DenseNet:")
y_pred_densenet = model_two.predict(X_test)
y_pred_densenet_bin = (y_pred_densenet > 0.5).astype(np.int_)
show_eval_models(y_pred_densenet_bin)
loss_den = log_loss(y_test, y_pred_densenet.astype("float64"))
print(f"Loss: {loss_den}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_densenet_bin)
plt.show()

print("MobileNet:")
y_pred_mobilenet = model_three.predict(X_test)
y_pred_mobilenet_bin = (y_pred_mobilenet > 0.5).astype(np.int_)
show_eval_models(y_pred_mobilenet_bin)
loss_mob = log_loss(y_test, y_pred_mobilenet.astype("float64"))
print(f"Loss: {loss_mob}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_mobilenet_bin)
plt.show()

print("Average:")
y_pred_avg = ensemble_model_avg.predict(X_test)
y_pred_avg_bin = (y_pred_avg > 0.5).astype(np.int_)
show_eval_models(y_pred_avg_bin)
loss_avg = log_loss(y_test, y_pred_avg.astype("float64"))
print(f"Loss: {loss_avg}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_avg_bin)
plt.show()

# Ансамбль моделей з використанням ваг
# Виконуємо передбачення з різними вагами для моделей
print("Weighted:")
y_pred_weights = ensemble_model_weights.predict(X_test)
y_pred_weights_bin = (y_pred_weights > 0.5).astype(np.int_)
show_eval_models(y_pred_weights_bin)
loss_weights = log_loss(y_test, y_pred_weights.astype("float64"))
print(f"Loss: {loss_weights}")
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_weights_bin)
plt.show()
```