НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
(КПІ ІМ. ІГОРЯ СІКОРСЬКОГО)

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

«До захисту допущено»

В.о. завідувач кафедри БМК

_____ **Євген НАСТЕНКО**

“___”_____2023р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Комп’ютерні технології в біології та медицині»
спеціальності *122 «Комп’ютерні науки»*

на тему: **Програмний додаток для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі**

Виконав (-ла): студент (-ка) **IV** курсу, групи _БС-93_

ОНИЩУК ЯРОСЛАВ ОЛЕКСАНДРОВИЧ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник: доцент каф. біомедичної кібернетики (БМК)
доцент, к.п.н., Добровська Людмила Миколаївна _____
(посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові) (підпис)

Рецензент: *проф. каф. біомедичної інженерії, проф., д.т.н.*
Лебедєв Олексій Володимирович _____
(посада, науковий ступінь, вчене звання, прізвище, ім’я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка)

(підпис)

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет біомедичної інженерії
Кафедра біомедичної кібернетики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні технології в біології та медицині»

ЗАТВЕРДЖУЮ
В.о. завідувач кафедри БМК
_____ Євген НАСТЕНКО
«_*30*_» __*травня*___2023 р.

ЗАВДАННЯ
на дипломну роботу студентці

ОНИЩУК ЯРОСЛАВ ОЛЕКСАНДРОВИЧ
(прізвище, ім'я, по батькові)

1. Тема роботи **Програмний додаток для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі**

Керівник роботи
Добровська Людмила Миколаївна, к.п.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31_»_*травня*_ 2023 р. №***2106-с***

2. Термін подання студентом роботи ***06-08 червня 2023р.***
3. Вихідні дані до роботи: *КТ знімки головного мозку, мова програмування Python, фреймворки TenzorFlow, PyQt5, OpenCV, Keras, Matplotlib*
4. Зміст роботи: *підбір кращої архітектури нейронної мережі, тестування нейронної мережі, побудова програмного додатку для користування нейронною мережею*.
5. Перелік ілюстративного матеріалу: 12 слайдів
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ***30 травня 2023 року***

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роб оти	Примітка
1	Отримати завдання за темою ДР на практику	До 15.02.2023р.	*виконано*
2	Переддипломна практика	За графіком	*виконано*
3	Виконання розділів ДР (Вступ, аналітичний огляд літературних джерел, теоретична частина)	До кінця практики	*виконано*
4	Виконання розділів ДР (практична частина, загальні висновки, список джерел)	Не пізніше 1 тижня до засідання каф-ри	*виконано*
5	Перевірка ДР науковим керівником	Не пізніше 1 тижня до засідання каф-ри	*виконано*
6	Подання в електронному вигляді ДР та анотації до неї на перевірку нормоконтролера та плагіат (UNICHECK).	---- « ------	*виконано*
7	Надання документів на засідання кафедри	За день до засідання	*виконано*
8	Предзахист ДР та допуск до захисту дисертації	Згідно плану каф.	*виконано*
9	Подання ДР рецензенту. Отримання рецензії.	До подання пакету документів до ЕК	*виконано*
10	Подання пакету документів по ДР та супровідних до неї документів до захисту в ЕК[1]	За 5 днів до дати захисту ДР за графіком	*виконано*
11	Захист ДР в ЕК		

Студент _____ **Ярослав ОНИЩУК**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник ДР _____ **Людмила ДОБРОВСЬКА**
(підпис) (ім'я, ПРІЗВИЩЕ)

Нормоконтролер _____ **Галина КОРНІЄНКО**
(підпис) (ім'я, ПРІЗВИЩЕ)

[1] не пізніше ніж за 5 днів до затвердженої дати захисту ДР в ЕК

АНОТАЦІЯ

Дипломна робота за темою «Програмний додаток для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі» виконана студентом кафедри біомедичної кібернетики ФБМІ Онищуком Ярославом Олександровичем *зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та* складається зі: вступу; 3 розділів (Аналітичний огляд літературних джерел, Теоретична частина, Практична частина), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 19 джерела та додатків. Загальний обсяг роботи 69 сторінок.

Актуальність теми. Завдяки розвитку сучасних технологій, включаючи глибинне навчання та машинний зір, стало можливою автоматизація виявлення пухлин. Одним із сучасних методів, що можуть бути використані у діагностиці це використання згорткових нейронних мереж на медичних зображеннях МРТ головного мозку. Згорткові нейронні мережі є одними з найефективніших алгоритмів машинного навчання для розв'язання завдань комп'ютерного зору та обробки зображень. Отже, створення програмного додатку для автоматичного виявлення пухлин за допомогою моделей комп'ютерного може значно полегшити та поліпшити процес діагностики даного захворювання. Автоматизація процесів завжди слідувала за розвитком технологій, що допомагає оптимізувати знання та набутий досвід.

Мета і завдання роботи.

Мета даної дипломної роботи — розробка програмного додатку з використанням згорткової нейронної мережі для автоматичної ідентифікації пухлин головного мозку на основі зображень МРТ головного мозку.

Для досягнення цієї мети, необхідно:

1. Розглянути теоретичні засади згорткових нейронних мереж.
2. Вивчити методи глибинного навчання та машинного зору.
3. Побудувати архітектуру нейронної мережі та натренувати її на дата

сеті. Перевірити її роботу на тестових данних.

4. Розробити програмний додаток, який може аналізувати зображення головного мозку та автоматично виявляти наявність пухлин.

5. Сформувати загальні висновки щодо проведеної роботи.

Використані методи. У роботі було використано декілька методів:

1. Збір та підготовка даних: Було зібрано набір даних, що містить зображення головного мозку з різних джерел, таких як МРТ або КТ сканування. Дані було попередньо оброблено, включаючи масштабування, нормалізацію та вирівнювання.

2. Архітектура згорткової нейронної мережі (ЗНМ): Було вибрано та налаштовано відповідну архітектуру ЗНМ для розпізнавання пухлин головного мозку. Це може включати в себе різні комбінації згорткових шарів, шарів пулінгу, повнозв'язаних шарів, функцій активації та шарів розпізнавання.

3. Тренування ЗНМ: Було використано тренувальний набір даних для навчання ЗНМ. Це включало в себе подачу зображень через ЗНМ, обчислення втрати (кросс-ентропійна функція втрати) та виконання зворотного поширення помилки для оновлення ваг моделі. Процес тренування може вимагати багато епох для досягнення задовільних результатів.

4. Перевірка та оцінка моделі: Після тренування моделі було використано набір даних для перевірки та оцінки її продуктивності.

Отримані результати. **.** Засвоєно основні принципи розробки програмного додатку для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі.

- Здобуто навички в роботі з згортковими нейронними мережами та обробці медичних зображень.

- Проведено практичне застосування придбаних знань шляхом розробки імплементації програмного додатку, який здатний ідентифікувати пухлини головного мозку на основі згорткової нейронної мережі.

- Реалізовано програмний додаток з наступними властивостями:

1. Зчитування та обробка медичних зображень головного мозку.

2. Застосування згорткової нейронної мережі для ідентифікації пухлин на зображеннях.

3. Візуалізація результатів ідентифікації та надання відповіді користувачеві.

- реалізація поставлених задач, таких як розробка програмного додатку для ідентифікації пухлин головного мозку, обробка медичних зображень та застосування згорткової нейронної мережі.

Публікації. Не передбачено.

Ключові слова. пухлина головного мозку, нейрона мережа, згортковий шар, програмний додаток, медичні зображення.

Бібліографічний опис ДР

Онищук Я.О. Програмний додаток для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі : дипломна роб. бакалавра : 122 Комп'ютері науки / Онищук Ярослав Олександрович. – Київ, 2023. – 69 с.

ABSTRACT

The thesis entitled "Software Application for Brain Tumor Identification Using Convolutional Neural Network" was carried out by Yaroslav Oleksandrovich Onyshchuk, a student of the Biomedical Cybernetics Department at the Faculty of Biomedical Engineering and Computer Science, specializing in Computer Science. The thesis consists of an introduction, three chapters (Literature Review, Theoretical Part, Practical Part), conclusions for each of these chapters, general conclusions, a list of references comprising 19 sources, and appendices. The total length of the thesis is 64 pages.

Relevance of the Topic: Advances in modern technologies, including deep learning and computer vision, have made it possible to automate tumor detection. One of the contemporary methods that can be used in diagnosis is the utilization of convolutional neural networks (CNNs) on magnetic resonance imaging (MRI) brain scans. CNNs are among the most effective machine learning algorithms for computer vision tasks and image processing. Therefore, developing a software application for automatic tumor detection using computer models can significantly facilitate and improve the diagnostic process. Automation has always followed technological advancements, helping to optimize knowledge and acquired experience.

Objective and Tasks: The goal of this thesis is to develop a software application using a convolutional neural network for automatic identification of brain tumors based on MRI brain images. To achieve this goal, the following tasks were pursued:

1. Study the theoretical foundations of convolutional neural networks.
2. Explore deep learning and computer vision methods.
3. Construct the architecture of a neural network and train it on a dataset. Validate its performance on test data.
4. Develop a software application capable of analyzing brain images and automatically detecting tumor presence.
5. Formulate general conclusions regarding the conducted work.

Methods Used: The thesis employed several methods:

1. Data Collection and Preparation: A dataset containing brain images from various sources, such as MRI or CT scans, was collected. The data underwent preprocessing, including scaling, normalization, and alignment.

2. Convolutional Neural Network (CNN) Architecture: A suitable CNN architecture was selected and configured for brain tumor recognition. This may involve different combinations of convolutional layers, pooling layers, fully connected layers, activation functions, and classification layers.

3. CNN Training: A training dataset was used to train the CNN. This involved feeding images through the CNN, calculating the loss (cross-entropy loss function), and performing backpropagation to update the model's weights. The training process may require multiple epochs to achieve satisfactory results.

4. Model Testing and Evaluation: After training the model, a dataset was used to verify and evaluate its performance.

Results Obtained: The following results were achieved:

- Mastery of the fundamental principles of developing a software application for brain tumor identification using a convolutional neural network.
- Acquisition of skills in working with CNNs and processing medical images.
- Practical application of acquired knowledge through the implementation of a software application capable of identifying brain tumors based on a convolutional neural network.
- Implementation of the software application with the following features:

1. Reading and processing of brain MRI images.
2. Application of a convolutional neural network for tumor identification in the images.
3. Visualization of identification results and presentation of the output to the user.

- Accomplishment of the set tasks, such as the development of a software application for brain tumor identification, medical image processing, and utilization of a convolutional neural network.

Publications: None anticipated.

Keywords: brain tumor, neural network, convolutional layer, software application, medical images.

Bibliographic Description of the Thesis: Onyshchuk, Y.O. (2023). Software Application for Brain Tumor Identification Using Convolutional

ЗМІСТ

ВСТУП

У час постійних пандемій та війн, медицина є однією з найважливіших галузей, яка постійно розвивається та вдосконалюється завдяки новітнім технологіям та науковим дослідженням. Швидке виявлення захворювання та його зцілення дає змогу запобігти важких ускладнень. Саме тому найбільш актуальними та першочерговими у цій галузі є діагностика та лікування. У сучасному світі гострою проблемою є збільшення випадків злоякісних пухлин, одна з найнебезпечніших видів для людини є пухлина головного мозку. На жаль, досі існує недостатня кількість точних та швидких методів діагностики цього захворювання. Тому своєчасна ідентифікація пухлини є важливою задачею для його успішної діагностики та лікування.

Застосування та відкриття нових методів діагностики допоможе своєчасному виявленню пухлин та ефективному лікуванню. Одним із таких підходів може бути використання згорткових нейронних мереж для ідентифікації пухлин головного мозку на зображеннях, отриманих різними методами обстеження (магнітно-резонансна томографія, комп’ютерна томографія тощо). Згорткові нейронні мережі є одними з найефективніших алгоритмів машинного навчання для вирішення завдань комп’ютерного зору та обробки зображень. Для зручного використання моделі її можна інтегрувати у зручний UI (User Interface) та використовувати як програмний додаток.

Розробка та поширення подібних програмних додатків зможе суттєво пришвидшити початкові етапи виявлення хвороби, які зазвичай займають невиправдано велику кількість часу. Це може допомогти зберегти життя та полегшити роботу лікарів у ході діагностики. По-перше, це додає мобільності спеціалістам, що може суттєво економити найважливіший в цьому питанні ресурс — час. По-друге, це може служити додатковою перевіркою діагнозу та результатів, вказаних лікарем.

Завдяки розвитку сучасних технологій, включаючи глибинне навчання та

машинний зір, стало можливою автоматизація виявлення пухлин. Одним із сучасних методів, що можуть бути використані у діагностиці це використання згорткових нейронних мереж на медичних зображеннях МРТ головного мозку. Згорткові нейронні мережі є одними з найефективніших алгоритмів машинного навчання для розв'язання завдань комп'ютерного зору та обробки зображень. Отже, створення програмного додатку для автоматичного виявлення пухлин за допомогою моделей комп’ютерного може значно полегшити та поліпшити процес діагностики даного захворювання. Автоматизація процесів завжди слідувала за розвитком технологій, що допомагає оптимізувати знання та набутий досвід.

Мета і завдання роботи

Мета даної дипломної роботи — розробка програмного додатку з використанням згорткової нейронної мережі для автоматичної ідентифікації пухлин головного мозку на основі зображень МРТ головного мозку.

Для досягнення цієї мети, необхідно:

Розглянути теоретичні засади згорткових нейронних мереж.

Вивчити методи глибинного навчання та машинного зору.

Побудувати архітектуру нейронної мережі та натренувати її на дата сеті. Перевірити її роботу на тестових даних.

Розробити програмний додаток, який може аналізувати зображення головного мозку та автоматично виявляти наявність пухлин.

Сформувати загальні висновки щодо проведеної роботи.

Її досягнення передбачає вирішення наступних *завдань*:

Аналіз вітчизняних та зарубіжних джерел.

Побудувати архітектуру нейронної мережі для класифікації зображень.

Розробити графічний інтерфейс для моделі

Використані методи. У роботі було використано декілька методів:

Збір та підготовка даних: Було зібрано набір даних, що містить зображення головного мозку з різних джерел, таких як МРТ або КТ сканування. Дані було попередньо оброблено, включаючи масштабування, нормалізацію та вирівнювання.

Архітектура згорткової нейронної мережі (ЗНМ): Було вибрано та налаштовано відповідну архітектуру ЗНМ для розпізнавання пухлин головного мозку. Це може включати в себе різні комбінації згорткових шарів, шарів пулінгу, повнозв'язаних шарів, функцій активації та шарів розпізнавання.

Тренування ЗНМ: Було використано тренувальний набір даних для навчання ЗНМ. Це включало в себе подачу зображень через ЗНМ, обчислення втрати (кросс-ентропійна функція втрати) та виконання зворотного поширення помилки для оновлення ваг моделі. Процес тренування може вимагати багато епох для досягнення задовільних результатів.

Перевірка та оцінка моделі: Після тренування моделі було використано набір даних для перевірки та оцінки її продуктивності.

Отримані результати. Засвоєно основні принципи розробки програмного додатку для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі.

- Здобуто навички в роботі з згортковими нейронними мережами та обробці медичних зображень.

- Проведено практичне застосування придбаних знань шляхом розробки імплементації програмного додатку, який здатний ідентифікувати пухлини головного мозку на основі згорткової нейронної мережі.

- Реалізовано програмний додаток з наступними властивостями:

1. Зчитування та обробка медичних зображень головного мозку.

Застосування згорткової нейронної мережі для ідентифікації пухлин на зображеннях.

Візуалізація результатів ідентифікації та надання відповіді користувачеві.

- реалізація поставлених задач, таких як розробка програмного додатку для ідентифікації пухлин головного мозку, обробка медичних зображень та застосування згорткової нейронної мережі.

Публікації. Не передбачено.

Структура роботи

Дипломна робота за темою «Програмний додаток для ідентифікації пухлин

головного мозку з використанням згорткової нейронної мережі» виконана студентом Онищуком Ярославом Олександровичем *зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині»*, побудована за класичним типом та викладена на 69 сторінках машинописного тексту. Вона складається з: вступу; 3 розділів (Аналітичний огляд літературних джерел, Теоретична частина, Практична частина), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 19 джерел. В роботі представлено 30 рисунків і 2 таблиць.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1. Оглад літературних джерел

1. Aurélien, Géron. "Hands-on machine learning with scikit-learn & tensorflow." Geron Aurelien (2017). Це детальний підручник з машинного навчання, який охоплює багато тем, від введення до машинного навчання до більш складних технік. Серед значних переваг джерела є те, що воно надає читачам можливість зрозуміти основи машинного навчання, навіть якщо у них немає попереднього досвіду з програмуванням або статистикою. На початку книги є введення до машинного навчання та його застосування в різних областях, таких як медицина, фінанси, наука про матеріали та інші. Також, ключовим аспектом книги є практичні приклади, що допомагають легше зрозуміти, як використовувати бібліотеки машинного навчання Scikit-Learn та TensorFlow. Кожен розділ містить приклади коду на Python, що дозволяє підійти до ознайомлення ще й з практичної частини (побачити та перевірити, як саме це працює).

Книга охоплює різноманітні теми, такі як класифікація, регресія, кластеризація, нейронні мережі та глибинне навчання. Наявні розділи про обробку природньої мови та рекомендаційні системи. Автор також зосереджується на практичних аспектах машинного навчання, таких як налаштування моделі та оптимізація гіперпараметрів. Крім того, часто включені поради реальних застосувань машинного навчання, що дає змогу зрозуміти, як можна використовувати ці техніки для вирішення реальних проблем.

Одним із незначних недоліків книги є те, що була вперше опублікована в 2017 році, тому деякі частини можуть бути застарілими в контексті останніх тенденцій машинного навчання та нових версій Scikit-Learn та TensorFlow.

Загалом, «Hands-On Machine Learning with Scikit-Learn & TensorFlow» є цінним ресурсом, оскільки має багато узагальнених та практичних знань, що подані досить зрозуміло.

2. Trask, Andrew W. Grokking deep learning. Simon and Schuster, 2019. Це книга, яка призначена для людей, що хочуть зрозуміти основи глибокого машинного навчання, не заглиблюючись у складну математику та програмування. Тут все написано легко та зрозуміло, та з наведенням багатьох прикладів, які допомагають читачам зрозуміти основні концепції машинного навчання. Одним з переваг даного джерела є те, що у поясненнях постійно використовує неформальна математика, як каже сам автор: «Все має бути настільки простим, на скільки це тільки можливо». Крім того, у тексті відсутні складні мовні конструкції, що робить книгу доступною для широкого кола читачів, які не мають великих знань у математиці та програмуванні. Ще одна з переваг, це багато зображень та ілюстрацій для наочного пояснення складних моментів, вони роблять їх більш зрозумілими. Тут є приклади коду на мові програмування Python, що дає змогу отримати практичні навички на поставлених задачах. Сам код зрозуміло написаний та належним чином пояснений

В цілому, недоліком як і плюсом даного джерела може бути його простота. Для деяких читачів, які мають певний досвід у машинному навчанні, ця книга може бути занадто спрощеною. Також, хоча книга охоплює багато основних концепцій глибокого навчання, вона не включає в себе більш складні теми, які можуть бути корисні для досвідчених інженерів машинного навчання. У ній не містяться приклади з використанням найпотужніших бібліотек глибокого навчання, таких як PyTorch та Tenzorflow, Keras. Однак, це не впливає на загальну цінність книги, бо майже всі підходи, які описані у ній, можуть бути застосовані незалежно від бібліотек.

Загалом, цей літературний ресурс є корисним для тих, хто бажає зрозуміти основи машинного навчання при відсутності будь-яких хороших знань у галузі математики та програмування. Прочитання книги «Grokking Deep Learning» є гарним стартом у вивченні штучного інтелекту та його особливостей.

3. Синєглазов, Віктор, та Олена Чумаченко. "Глибокі Нейронні Мережі для Вирішення Завдань Розпізнавання і Класифікації Зображення." Інформаційні технології та комп’ютерне моделювання 2017: 15-20. У цій статті автори

розглядають застосування глибокого машинного навчання з використанням нейронних мереж для розпізнавання та класифікації зображень. Тут детально пояснюються архітектура та її основні компоненти (шари), такі як вхідні, приховані та вихідні шари.

Особливу увагу у статті виділено згортковим нейронним мережам, чітко пояснена їх особливість роботи та приклади використання для задачі класифікації зображень. Наведені переваги над іншими типами шарів, таких як повнозв'язний та інші, для поставленої задачі , та недоліки у використанні. Описуються алгоритми усунення недоліків та оптимізація, параметризація згорткових слоїв нейронної мережі.

Отже, стаття "Глибокі Нейронні Мережі для Вирішення Завдань Розпізнавання і Класифікації Зображення" є корисним джерелом для пізнання у роботі моделей комп'ютерного зору, їх застосування на задачах класифікації зображень. Стаття надає докладний огляд основних понять та методів, які використовуються в глибокому навчанні та зосереджується на застосуванні згорткових нейронних шарів глибоких нейронних мереж.

4. Джерело "Brain tumor segmentation with Deep Neural Networks" авторства Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., & Pal, C. (2017) є статтею, опублікованою в журналі Medical Image Analysis. Оглянувши цю статтю, можна отримати інформацію про застосування глибоких нейронних мереж для сегментації пухлин головного мозку. Стаття пропонує метод, який використовує глибоку згорткову нейронну мережу для сегментації пухлин головного мозку на медичних зображеннях. Дослідники використали набір даних BRATS (The Multimodal Brain Tumor Segmentation Challenge), що містить зображення МРТ пацієнтів з пухлинами головного мозку. Автори статті описують архітектуру своєї нейронної мережі, яка використовує згорткові шари для виявлення різних ознак на зображеннях та піксельну класифікацію для сегментації пухлин. Вони також використали методи для зменшення перенавчання, такі як випадкові деформації та аугментацію даних. Автори провели експерименти з використанням набору даних BRATS, де їх метод показав добрі результати в порівнянні з іншими методами

сегментації пухлин головного мозку. Вони оцінювали точність сегментації, включаючи показники DICE та чутливість.

5. Джерело "Single subject prediction of brain disorders in neuroimaging: Promises and pitfalls" авторства Arbabshirani, M. R., Plis, S., Sui, J., & Calhoun, V. D. (2017) є статтею, опублікованою в журналі NeuroImage. У цій статті розглядаються можливості та проблеми передбачення розладів мозку у одного окремого суб'єкта на основі нейроіміджингу.

Дослідники описують використання методів машинного навчання для передбачення різних розладів мозку, таких як шизофренія, депресія, розлади аутистичного спектра та інші. Вони обговорюють можливості застосування класифікаційних моделей та регресійних підходів до нейроіміджингових даних з метою індивідуального прогнозування розладів мозку.

Автори статті також висвітлюють питання та потенційні проблеми, пов'язані з цим підходом. Вони звертають увагу на необхідність великих обсягів даних для надійних передбачень, а також на необхідність уточнення протоколів обробки даних та стандартизації методів.

Загальний висновок статті полягає в тому, що передбачення розладів мозку на основі нейроіміджингу для окремих суб'єктів має великий потенціал, але вимагає додаткових досліджень та розвитку методологічних підходів. Дослідження в цьому напрямку можуть сприяти покращенню діагностики та індивідуального планування лікування розладів мозку.

Висновок до роділу 1

У цьому розділі було розглянуто корисні літературні ресурси, які дають змогу краще поглибитися у розуміння основ машинного навчання при відсутності будь-яких хороших знань у цій галузі та здобути потрібні навички для виконання практичної частини побудови та валідації мережі для виявлення пухлин головного мозку на медичних знімках. Описані джерела надали теоретичні навички та кращі підходи, а саме архітектури нейронних мереж та кращі фреймворки, мови

програмування та середовища розробки для їх побудови, валідації та дослідження, які будуть використані у роботі.

РОЗДІЛ 2
ТЕОРЕТИЧНА ЧАСТИНА

2.1. Загальні відомості. Рак головного мозку

Рак головного мозку - це злоякісна пухлина, що розвивається в головному мозку або в околицях нього. Цей вид раку може розвиватися в будь-якій частині головного мозку, включаючи мозочок, між мозкову борозну, зоровий нерв, півкуля головного мозку. Симптоми раку головного мозку можуть варіюватися в залежності від розташування пухлини, але деякі загальні ознаки включають: головний біль, нудоту, блювання, проблеми зі зором, проблеми з координацією рухів та слабкість у кінцівках.[10]

Діагностика раку головного мозку може бути складною, оскільки симптоми можуть бути схожі на інші захворювання, а також через те, що головний мозок знаходиться в кістковій порожнині, що ускладнює проведення обстежень.

Основним методом діагностики раку головного мозку є обстеження за допомогою зображувальних методів, таких як магнітно-резонансна томографія (МРТ) та комп'ютерна томографія (КТ). Ці методи дозволяють отримати детальні зображення головного мозку та визначити розмір та розташування пухлини. Крім зображувальних методів, можуть використовуватися інші методи діагностики, такі як електроенцефалографія (ЕЕГ), яка дозволяє виявити відхилення в роботі мозку, та біопсія, яка забезпечує можливість отримання зразків тканини для подальшого аналізу.

Діагноз раку головного мозку може бути встановлений лише після аналізу отриманих даних та вивчення клінічних симптомів пацієнта. Якщо діагноз раку головного мозку підтверджується, лікарі проводять подальшу оцінку розміру та стадії пухлини, що допомагає визначити план лікування. Важливо зазначити, що раннє виявлення пухлини може значно покращити прогноз хвороби та збільшити шанси на успішне лікування.

Оскільки головний мозок відповідає за функціонування всього організму, рак

головного мозку є дуже серйозним захворюванням, яке може мати значний вплив на якість життя пацієнта. Шанси на успішне вилікування від раку головного мозку можуть бути покращені раннім виявленням пухлини та швидким лікуванням.

2.1.1. Нейрона мережа

Нейронні мережі – це математичне моделювання біологічних нейронних мереж. Вони містять у собі з'єднані між собою нейрони, які приймають вхідні дані, обробляють їх та видають вихідні дані. Нейрон складається з вхідних даних, вагових коефіцієнтів та функцій активації. Вхідні дані - це інформація, яка подаються на вхід нейрона. Вагові коефіцієнти - це числа, на які множаться вхідні дані та передаються до функції активації. Функція активації – це функція, яка приймає значення, обробляє його та виводить результат.[2]

Всі нейронні мережі складаються з шарів нейронів, кожен з яких має вхідні та вихідні дані (рис. 2.1). Шари поділяються на зовнішні та внутрішні. Зовнішні - це перший та останній шар, а внутрішні, чи як їх ще називають – приховані, це все що знаходиться між зовнішніми. Кожен шар мережі має свої вагові коефіцієнти та функцію активації, що налаштовуються під час навчання моделі.

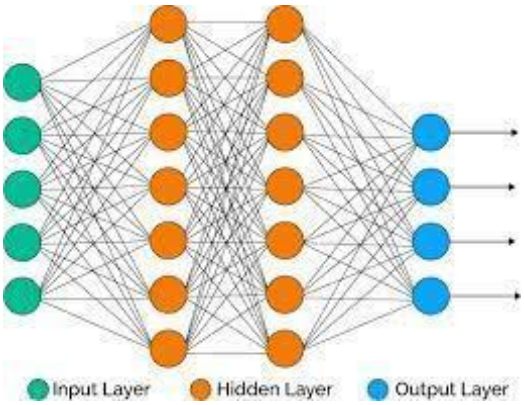


Рисунок 2.1 - Схема нейронних мереж

Сам процес навчання мереж полягає в знаходженні оптимальних значень вагових коефіцієнтів кожного шару. Це досягається шляхом використання алгоритмів оптимізації, таких як зворотне поширення помилки та градієнтного спуску. При цьому, мережа навчається за допомогою тренувального набору даних,

де відомі вхідні та вихідні дані, підбираючи коефіцієнти, які мінімізують помилку (втрати), тобто різницю між справжнім значенням та виводом моделі.[2]

Такі моделі машинного навчання широко використовуються для розв'язання різних задач, таких як класифікація, регресія, завдання комп’ютерного зору, генерація тексту та інше. Нейронні мережі можуть розв'язувати задачі, які неможливо виконати за допомогою традиційних алгоритмів програмування. Вони мають багато різновидів архітектур, одні з найпопулярніших - це одношарові та багатошарові повнозв’язні слої, згорткові мережі, рекурентні мережі, а також їх комбінації. У цій сфері є багато місця для експериментування, створення правильної архітектури є важливою частиною успішного виконання поставленої задачі. Одношарові персептрони можуть розв'язувати прості задачі класифікації та регресії, які не потребують великої складності архітектури. Багатошарові персептрони мають велику точність та можуть розв'язувати більш складні задачі, навіть такі як задачі комп’ютерного зору та задачі з текстом. Згорткові мережі ефективні для аналізу зображень, оскільки вони здатні виявляти локальні шаблони та ознаки на зображеннях. Рекурентні мережі ефективні для аналізу послідовностей даних, таких як мова та музика, оскільки вони можуть зберігати та використовувати інформацію з попередніх кроків.[9]

Підсумовуючи, нейронні мережі є основним інструментом машинного навчання, який допомагає вирішувати складні задачі в багатьох галузях, таких як фінанси , медицина, транспорт, енергетика та інші. Однак, їх використання також пов'язано з проблемами, такими як необхідність великих обсягів даних для навчання та складність інтерпретації результатів.

2.1.2. Градієнтний спуск

Градієнтний спуск - це один з найефективніших алгоритмів оптимізації у машинному навчанні та інших сферах, метою якого є знаходження мінімуму або максимуму функції. Основна ідея цього простого алгоритму полягає у тому, щоб знайти напрямок негативного градієнту функції, тобто напрямок, куди прямує функція та найшвидше спадає/зростає.

Робота цього алгоритму полягає у покроковому послідовному оновленні вектора параметрів моделі у напрямку, протилежному до градієнту функції у вибраній точці. Вектор параметрів моделі змінюється на величину, яка дорівнює деякому кроку у сторону, протилежну до самого градієнту. Для цього на кожному кроці обчислюється градієнт функції у поточній точці, та вектор параметрів моделі змінюється на деякий крок у напрямку, протилежному до градієнту. Як це працює можна побачити на рис. 2.2.[2]

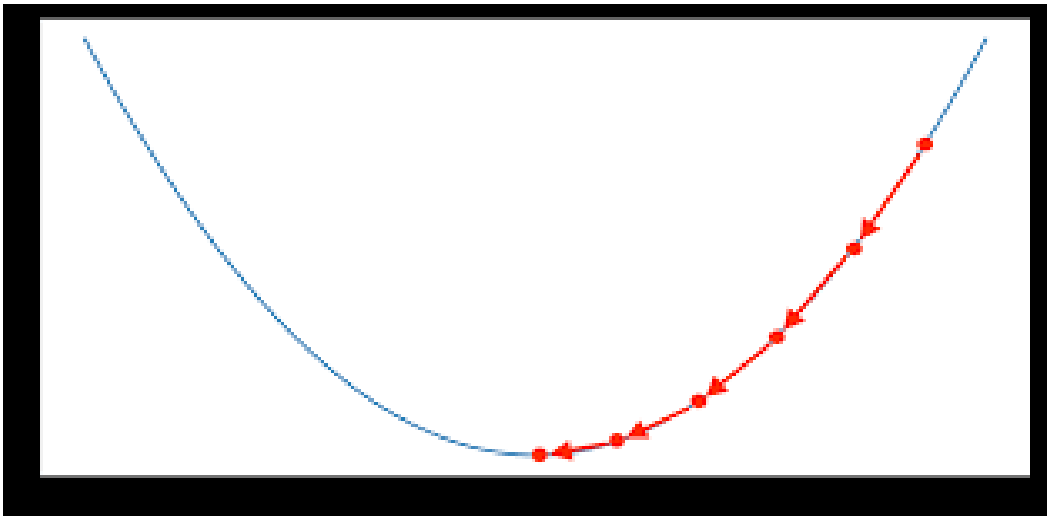


Рисунок 2.2 - Рух градієнта у напрямок зменшення помилки

Швидкість зміни параметрів моделі на кожному кроці залежить від гіперпараметру, який називається learning rate. Він впливає на швидкість навчання, якщо швидкість навчання велика, то можливе переповнення мінімуму функції. Це явище, коли вектор параметрів обходить мінімум і починає шукати його на іншому боці параболи. З іншої сторони при дуже малій швидкості навчання, оптимізація буде відбуватися занадто повільно, а в деяких випадках не зможе досягти кінцевої точки. Тому цей параметр треба підбирати ретельно, бо від нього залежить точність моделі, яку він оптимізує.

Градієнтний спуск працює під капотом та є основою таких популярних алгоритмів машинного навчання як логістична регресія, метод опорних векторів, нейронні мережі. Таке широке використання набув через свою ефективність та

простоту у застосуванні. Він гарно себе показав на великих наборах даних та у складних моделях. Від цього і пішли різні варіації та модернізації цього алгоритму. Їх використовують залежно від поставленої задачі та мети. Один з найбільш поширених варіантів - це стохастичний градієнтний спуск (SGD), який обчислює градієнт на підмножині випадково обраних даних. Це дозволяє знизити вимоги до пам'яті та обчислювальної потужності та прискорити оптимізацію. Інші варіації градієнтного спуску включають метод Нестерова (Nesterov momentum), AdaGrad, Adam, RMSProp. Кожен з цих методів має свої переваги та недоліки, тому вибір підходу залежить від конкретного завдання.[1]

Узагальнюючи, градієнтний спуск є потужним алгоритмом оптимізації, який знаходить широке застосування в машинному навчанні та інших галузях. Він дозволяє знизити значення функції втрат та покращити якість моделі, зменшуючи помилки та збільшуючи точність передбачень.

2.1.3. Зворотнє поширення похибки

Зворотнє поширення похибки є потужним алгоритмом, який використовується для навчання штучних нейронних мереж. Він дозволяє обчислити градієнти ваг для кожного шару мережі, що дає можливість використовувати градієнтний спуск для оптимізації ваг та зменшення функції втрат. Цей алгоритм працює методом надання мережі вхідних даних та вихідних значень, на початку вхідний сигнал прямує через шари мережі, виконуючи операції лінійних комбінацій та нелінійних активаційних функцій. Після цього порівнюється отриманий вихід з очікуваним та обчислюється значення функції втрат, яка визначає різницю між вихідним значенням та очікуваним. Наступним етапом є зворотній прохід, який полягає в обчисленні градієнтів функції втрат по вагам в кожному шарі мережі. Для цього використовується правило ланцюжків, яке дозволяє розбити градієнт функції втрат на множники та обчислити похідну від кожної функції активації. Після обчислення градієнтів можна використовувати градієнтний спуск для оновлення ваг відповідно до градієнту та швидкості навчання. Процес навчання триває до тих пір, поки значення функції втрат не досягне заданого рівня або досягне максимальної

кількості ітерацій. [2]

Зворотнє поширення похибки, так як і градієнтний спуск є основним алгоритмом для навчання штучних нейронних мереж. Їх робота разом допомагає створювати дуже потужні моделі, які здатні розв'язувати складні завдання. Але для досягнення найкращих результатів необхідно враховувати недоліки алгоритму такі як перенавчання.

2.1.4. Функції активації

Функції активації - це математичні функції, які використовуються для застосування нелінійності у нейронних мережах. Вихід кожного шару нейронів, який являє собою зважену суму вхідних сигналів, проходить через функцію активації, яка узагальнює отриманий результат. Таким чином цей метод дає змогу градієнту правильно рухатись та виконувати процес навчання. Нижче наведений приклад найпоширеніших функцій, які використовуються у нейронних мережах, а також їх графіки (рис. 2.3):

1. Функція Sigmoid - ця функція приймає значення від -inf до inf та перетворює їх у діапазон між 0 та 1. Вона чудово підходить для задачі бінарної класифікації, найчастіше використовується на вихідному зовнішньому шару, який повертає мітку класу 0 або 1.

2. Функція ReLU – це найпоширеніша функція активації. Вона приймає значення та повертає 0, якщо воно від’ємне, в іншому випадку повертає вхідне значення. Така проста функція дозволяє моделі навчатися швидше та знижує ризик зниклих градієнтів.

3. Функція Tanh - це функція, яка приводить значення до діапазону між -1 та саме вона набула поширене використання у LSTM-мережах.

4. Функція Softma - це функція, яка приймає вектор значень та перетворює його в розподіл ймовірностей на декількох класах. Часто її можна побачити у вихідних шарах нейронних мереж, що вирішують задачі багатокласової класифікації. [1][2]

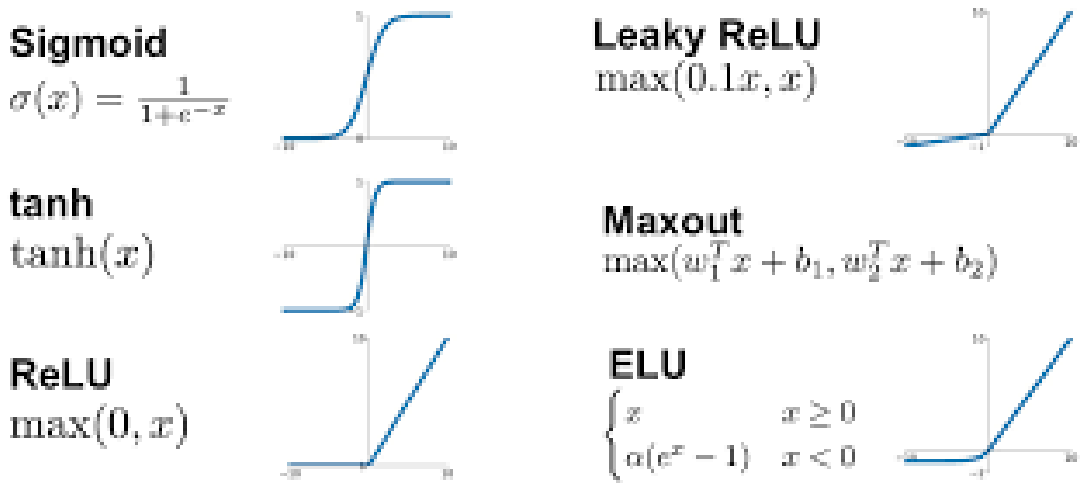


Рисунок 2.3 - Функції активації

Вибір функції активації залежить від конкретної задачі та властивостей даних, з якими має працювати модель. Тому це ще один з важливих параметрів, які потрібно ретельно обирати.

2.1.5. Класифікація зображень в медицині

Класифікація зображень в медицині - це важливий етап у діагностиці та лікуванні пацієнтів. Медичні зображення можуть бути отримані з різних джерел, таких як рентгенівські промені, комп'ютерна томографія, магнітно-резонансна томографія, ультразвук та інші методи. Автоматична обробка цих зображень може значно зменшити час, необхідний для діагностики та лікування пацієнтів, а також допомогти у виявленні ранніх стадій захворювань.[3]

Однією з важливих областей застосування класифікації зображень в медицині є онкологія. Виявлення злоякісних новоутворень на ранніх стадіях дозволяє покращити прогноз лікування та підвищити шанси на успішне вилікування. Автоматична класифікація зображень допомагає виявляти ознаки ракових захворювань на ранніх стадіях та підвищувати ефективність лікування.

Іншою областю застосування класифікації зображень в медицині є кардіологія. Аналіз зображень серця дозволяє виявляти ознаки захворювань, таких як ішемія, аритмії та інші. Автоматична класифікація зображень може допомогти виявити ранні ознаки серцевих захворювань та зменшити ризик виникнення

серйозних ускладнень.

Нейронні мережі та глибинне навчання є основними методами класифікації зображень в медицині. Ці методи дають можливість автоматизувати процес аналізу зображень та зменшити кількість помилок, що допускаються людиною. Однак треба зазначити, що важливо бути уважним у розробці та використанні алгоритмів класифікації зображень в медицині. Наприклад, неправильна класифікація може призвести до неправильної діагностики та лікування пацієнтів, що може бути небезпечним. Тому, розробники повинні мати достатні знання та досвід у медицині, щоб забезпечити точність та надійність результатів.

Також важливо враховувати етичні аспекти використання класифікації зображень в медицині. Наприклад, необхідно дотримуватися принципу конфіденційності та захисту персональних даних пацієнтів. Крім того, потрібно враховувати культурні та соціальні різниці при використанні класифікації зображень в медицині.

Загалом, класифікація зображень в медицині є важливим інструментом у діагностиці та лікуванні пацієнтів. Автоматична обробка медичних зображень може зменшити час та вартість діагностики та лікування, а також покращити прогнози результатів лікування. Але важливо дотримуватися етичних та професійних стандартів при розробці та використанні алгоритмів класифікації зображень в медицині.

2.1.6. Метрики оцінки якості класифікації

Метрики оцінки якості класифікації - це набір інструментів, які використовуються для вимірювання точності моделі класифікації. Ці метрики дозволяють оцінити ефективність алгоритму та зробити правильний вибір між різними моделями. У цьому розділі ми розглянемо найбільш поширені метрики оцінки якості класифікації та їх прикладне використання.[2]

1. **Accuracy (Точність)**

Точність - це метрика, яка вимірює відношення кількості правильно класифікованих елементів до загальної кількості елементів. Формула точності:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

де TP (true positive) - кількість правильно класифікованих елементів, TN (true negative) - кількість правильно не класифікованих елементів, FP (false positive) - кількість неправильно класифікованих елементів, і FN (false negative) - кількість неправильно не класифікованих елементів.[2]

Наприклад, якщо ми маємо 100 елементів, і наша модель правильно класифікувала 80 з них, то точність буде 80/100 = 0.8.

2. **Precision (Точність прогнозування)**

Точність прогнозування - це метрика, яка вимірює відношення кількості правильно класифікованих позитивних елементів до загальної кількості позитивних елементів, що були визначені моделлю. Формула точності прогнозування:

Precision = TP / (TP + FP)

Наприклад, якщо ми маємо 100 елементів, і наша модель правильно визначила 20 позитивних елементів, із яких 15 були справді позитивними, а 5 - негативні, то точність прогнозування буде 15/20 = 0.75.[2]

3. **Recall (Чутливість)**

Чутливість - це метрика, яка вимірює відношення кількості правильно класифікованих позитивних елементів до загальної кількості позитивних елементів. Формула чутливості:

Recall = TP / (TP + FN)

Наприклад, якщо ми маємо 100 елементів, і з них 30 позитивні, і наша модель правильно визначила 25 з них, то чутливість буде 25/30 = 0.83.[2]

4. **F1-Score**

F1-Score - це метрика, яка є гармонічним середнім між точністю прогнозування і чутливістю. Формула F1-Score:

F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

Наприклад, якщо ми маємо 100 елементів, і наша модель має точність прогнозування 0.7 та чутливість 0.8, то F1-Score буде 2 * (0.7 * 0.8) / (0.7 + 0.8) = 0.74.[2]

5. **ROC Curve**

ROC (Receiver Operating Characteristic) Curve - це графік, який відображає відношення між чутливістю та специфічністю моделі. Він зображаєся на площині, де по вісі X - false positive rate (FPR), а по вісі Y - true positive rate (TPR). FPR - це відношення кількості неправильно класифікованих негативних елементів до загальної кількості негативних елементів. TPR - це відношення кількості правильно класифікованих позитивних елементів до загальної кількості позитивних елементів. ROC Curve дозволяє оцінити ефективність моделі при різних порогових значеннях.[2]

6. **AUC (Area Under Curve)**

AUC (Area Under Curve) - це метрика, яка вимірює площу під ROC Curve. Чим більша площа під кривою, тим краще ефективність моделі. AUC може бути в діапазоні від 0 до 1, де значення 0 означає, що модель завжди неправильно класифікує, а значення що означає, що модель завжди правильно класифікує.[2]

7. **Log Loss**

Log Loss - це метрика, яка вимірює рівень помилок моделі, коли вона використовує ймовірності класифікації. Чим менше значення Log Loss, тим краще ефективність моделі. Формула Log Loss:

Log Loss = -1/N * Σ(y * log(y_pred) + (1 - y) * log(1 - y_pred))

де y - правильний клас, y_pred - передбачений клас, N - загальна кількість прикладів.

Наприклад, якщо ми маємо 100 елементів, і наша модель має Log Loss 0.5, то це означає, що в середньому модель зробила 0.5 помилок у передбаченні класів.

Оцінка ефективності моделі класифікації є важливим етапом у машинному навчанні. Метрики оцінки якості класифікації дозволяють визначити, наскільки точно модель передбачає класи об'єктів. Для правильного використання метрик необхідно враховувати особливості задачі та вибрати метрики, які найбільше відповідають потребам проекту.

2.2 Види шарів та нейронних мереж

2.2.1 .Повнозв’язний шар

Повнозв'язний шар - це тип шару нейронної мережі, у якому кожен нейрон пов'язаний з кожним нейроном з попереднього шару. Їх архітектура зображена на рис. 2.4. Це означає, що кожен вихідний сигнал нейрону в попередньому шарі стає вхідним сигналом для кожного нейрону в поточному шарі. Вони мають ваги та зсув, які використовуються для обчислення вихідного значення нейрона. На вхід нейрона надходять зважені сигнали. Вони обчислюються як добуток вхідного сигналу та відповідної ваги, до них додається зсув. Зважена сума подається до функції активації, яка обмежує вихід нейрона в певному діапазоні значень. Функція активації є ключовим елементом повнозв'язного шару, оскільки вона дозволяє нейронній мережі вирішувати задачу. [4]

Навчання повнозв'язного шару зазвичай відбувається за допомогою алгоритму зворотнього поширення помилки (Backpropagation), який дозволяє оновлювати ваги та зсуви відповідно до похибки, яка виникає між вихідними значеннями мережі та бажаними значеннями.

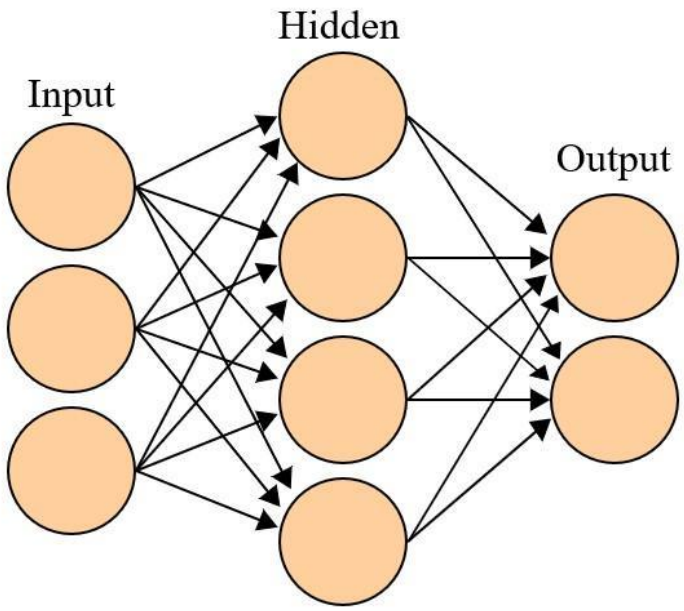


Рисунок 2.4 - Повнозв’язний шар

Оновлення ваг та зсувів здійснюється за допомогою градієнтного спуску,

який мінімізує функцію втрат. Функція втрат - це метрика, яка описує, точність мережі та дає змогу зрозуміти на скільки модель помиляється. Наприклад, для задач класифікації, функції втрат можуть використовувати крос-ентропію або середньоквадратичну помилку.[1]

Повнозв'язний шар може бути використаний в різних типах нейронних мереж. Вони можуть бути присутні у персептроні, згортковій, рекурентній нейронній мережі та інших. Їх будова дозволяє мережі вчитися складнішим функціям шляхом комбінування результатів попередніх шарів, так як кожен нейрон у шарі має зв’язок з кожним нейроном попереднього шару. Таким чином можна виявити складні залежності між вхідними та вихідними даними. Однак існує декілька недоліків, один з яких є наявність великої кількості параметрів. Це може призвести до перенавчання моделі, особливо, якщо вхідні дані досить складні. Друге це нездатність зауважувати локальні особливості у вхідних даних, наприклад об’єкт на фото, тому ці шари погано працюють для вирішення задач комп’ютерного зору.

Загалом, повнозв'язний шар є важливим елементом багатьох типів нейронних мереж, і може бути використаний як для вирішення задач класифікації, так і для регресії. Однак, для більш складних задач, зазвичай використовують більш складні архітектури мережі, які поєднують у собі різні типи шарів.

2.2.2 Згортковий шар

Згорткові (Convolutional) шари є одним з основних типів шарів у глибокому навчанні для обробки зображень. Вони дозволяють визначати локальні особливості в зображеннях, використовуючи малий набір фільтрів, або як їх ще називають, ядер, що пересуваються по всьому зображенню. Кожен фільтр - це набір чисел, які відображають вагу пікселів в зображенні. Ці числа змінюються шляхом зворотного поширення помилки (backpropagation), як і у випадку з іншими шарами нейронних мереж. Перед використанням, фільтри підключаються до зображення за допомогою операції згортки, що призводить до утворення карт признаків, які виконують роль вихідних даних для наступних шарів (рис. 2.5).[2]

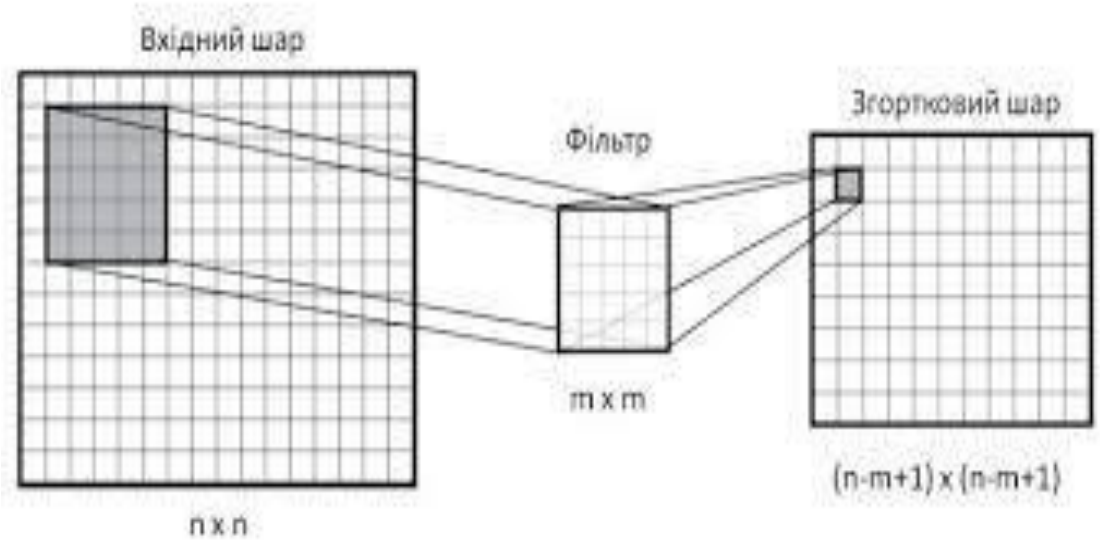


Рисунок 2.5 - Робота фільтру у згортковому шарі

Згорткові шари мають важливу роль, таку як:

1. Виявлення локальних особливостей, шляхом застосування фільтрів. Вони дозволяють виділяти різні характеристики зображення, такі як границі, кути, кольорові плями та інші.

2. Зменшення кількості параметрів. Згорткові шари замість використання окремих нейронів для кожного пікселя зображення, використовують фільтри, що дозволяє зменшити кількість параметрів мережі.

3. Загальний підхід до зображень. Можливість обробляти зображення будь-яких розмірів, оскільки фільтри пересуватися по всій області зображення, та обробляють його.

4. Розпізнавання об'єктів. З використанням цих шарів можна виконувати розпізнавання об'єктів у зображеннях, використовуючи декілька фільтрів для виділення різних характеристик.

5. Зменшення розмірності зображення. Разом зі згортковими шарами використовують операції пулінгу, що дозволяють зменшити кількість параметрів мережі та покращити її швидкість роботи.

6. Стійкість до зміни положення. Мережі не важливо де саме на зображенні знаходиться потрібний об’єкт, бо фільтри що ходять по зображенню, діють на всю

площу пікселів на картинці, а не на частини.

7. У згорткових шарах часто використовуються додаткові операції, такі як активаційні функції, які допомагають забезпечити нелінійність мережі, а також регуляризацію, яка допомагає зменшити перенавчання мережі.

Згорткові шари є основою багатьох архітектур нейронних мереж для обробки зображень, таких як LeNet, AlexNet, VGG, ResNet та інші. Вони дозволяють досягти високої точності розпізнавання зображень та забезпечити ефективну обробку великих об'ємів даних.

2.2.3 MaxPooling та MeanPooling

Max Pooling та Mean Pooling є двома основними операціями пулінгу, які використовуються в згорткових нейронних мережах для зменшення розміру зображення та забезпечення інваріантності до зсуву. Ці два алгоритми мають ідентичну логіку роботи. Max Pooling вибирає максимальне значення з певної області зображення та замінює цю область на максимальне значення. Ця операція дозволяє забезпечити інваріантність до малих зсувів об'єктів на зображенні, оскільки максимальне значення буде зберігатися незалежно від положення цього об'єкта на зображенні. Крім того, Max Pooling допомагає зменшити кількість параметрів мережі та покращити її швидкість роботи. В свою чергу Mean Pooling вибирає середнє значення з певної області зображення та замінює цю область на її середнє значення. Ця операція дозволяє також зменшити розмір зображення та кількість параметрів мережі, але на відміну від Max Pooling, вона не забезпечує інваріантності до малих зсувів об'єктів на зображенні. В основному, Max Pooling використовують для зменшення розміру зображення та забезпечення інваріантності до зсуву, а Mean Pooling - для зменшення кількості параметрів мережі та запобігання перенавчанню. В залежності від поставленої задачі, це 2 основних видів масштабування зображення для згорткових нейронних мереж. Їх робота зображена на рис. 2.6.[2]

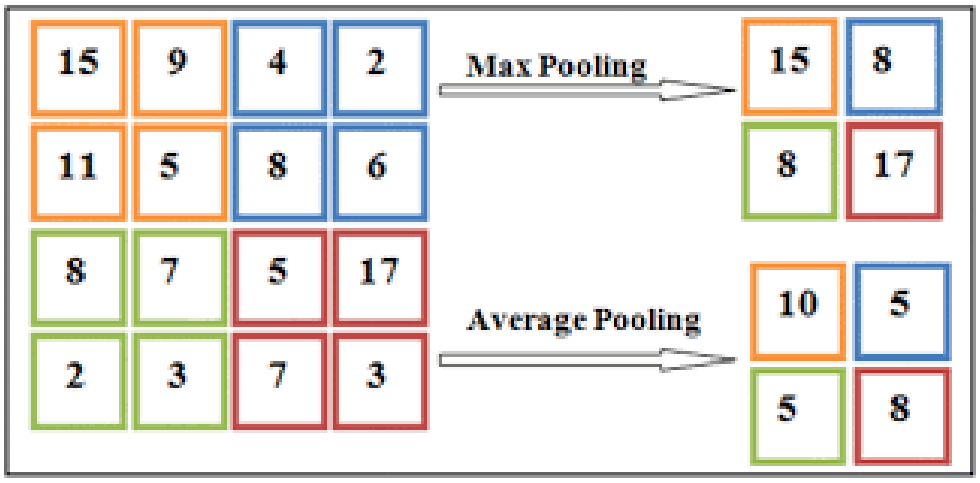


Рисунок 2.6 - Результати роботи Max Pooling та Mean Pooling

2.2.4 **Flatten**

Flatten - це операція в згорткових нейронних мережах, яка перетворює багатовимірні масиви на вектори. Ця операція зазвичай використовується для перетворення вихідних даних зі згорткових шарів на вхідні дані для повнозв’язного шару (рис. 2.7). У згорткових шарах, вихідні дані мають форму (batch_size, height, width, channels), де height та width - розміри зображення, channels - кількість каналів. Після проходження через згортковий шар та операції пулінгу, розмір даних може бути зменшений, але форма даних залишається (batch_size, new_height, new_width, channels). Для передачі цих даних на повнозв’язний шар, який очікує одномірний вектор на вході, дані потрібно перетворити на вектор. Це досягається за допомогою операції Flatten, яка просто вирівнює многовимірний масив в одномірний вектор. Таким чином, форма даних стає (batch_size, flattened_size), де flattened_size - це добуток розмірів зображення та кількості каналів. Після операції Flatten можна передавати вектор на повнозв'язаний шар для подальшої обробки та класифікації. Використання операції Flatten є необхідним, коли згорткова нейронна мережа використовується для класифікації зображень або іншого виду вхідних даних, які мають багатовимірну структуру.[11]

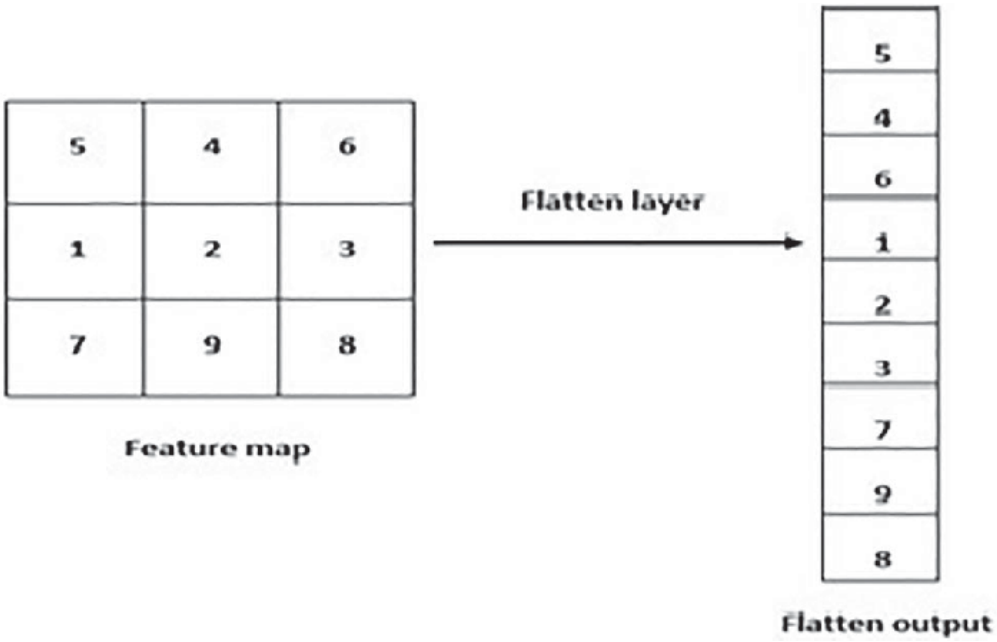


Рисунок 2.7 - Результати роботи Flatten

2.3 Мова програмування та фреймворки

2.3.1 Python

Python - це інтерпретована, високорівнева, об'єктно-орієнтована мова програмування, яку створив Гвідо ван Россум у 1991 році. Вона має простий та зрозумілий синтаксис, що робить його дуже легко засвоювати для початківців. Завдяки своїй простоті та зручності, Python здобув широку популярність в галузі науки про дані, машинного навчання, веб-розробки, аналізу даних, наукових обчислень та інших областях.[13]

Особливості мови програмування Python:

1. це інтерпретована мова, що означає, що код програми виконується без попередньої компіляції.
2. це об'єктно-орієнтована мова програмування. Всі дані в Python є об'єктами, а всі функції є методами.
3. Ця мова має динамічну типізацію, що дозволяє змінювати тип даних змінної в ході виконання програми.
4. Наявність широкого діапазону стандартних бібліотек, що дозволяє виконувати багато різних завдань без додаткової установки сторонніх бібліотек.

5. підтримує багато парадигм програмування, включаючи процедурне програмування, функціональне програмування та об'єктно-орієнтоване програмування.

6. має велику та активну спільноту користувачів та розробників, що створює безліч додаткових бібліотек, модулів та інструментів для програмування.

Python використовується в багатьох відомих проектах, включаючи Google, YouTube, Dropbox, Instagram та інші. Він також є дуже популярним в науці про дані та машинному навчанні, тому що має багато бібліотек для обробки та аналізу даних, такі як NumPy, Pandas, Matplotlib, SciPy, Scikit-learn та багато інших. Ще одною сферою використання є веб-розробка, де фреймворки, такі як Django та Flask, дозволяють розробляти веб-додатки та API. Також є можливість створення та розробки мобільних додатків за допомогою фреймворку Kivy. Ця мова гарно підтримує роботу з базами даних, що робить її універсальною.

Python має високий рівень абстракції, що дозволяє програмістам працювати з високорівневим кодом. Це зменшує об’єм написання програми, та полегшує розуміння коду для інших розробників. Кросплатформеність є одним з переваг Python, бо він дає змогу писати програми, що працюють на різних платформах, включаючи Windows, Mac та Linux. Це зменшує залежність програми від конкретної операційної системи та збільшує її переносимість.

Узагальнюючи, Python - це потужна та дуже популярна мова програмування, яка має широкий спектр застосування в різних галузях. Вона є простою та зрозумілою для новачків, а також має велику кількість бібліотек для роботи з даними та іншими завданнями, що робить її відмінним вибором для різних проектів. Python також є мовою з відкритим кодом, що дозволяє розробникам вносити свої внески до мови та розширювати її можливості.

2.3.2 NumPy

NumPy (Numerical Python) - це бібліотека мови програмування Python, яка надає структури даних та інструменти для роботи з числовими даними. Вона забезпечує швидке та ефективне обчислення, що робить її ідеальним інструментом

для наукових обчислень та аналізу великих даних. Структура даних NumPy, є більш ефективною для обчислень за звичайні списки Python. Тут є можливість виконувати швидкі обчислення з числовими даними, такими як вектори та матриці (рис. 2.8).

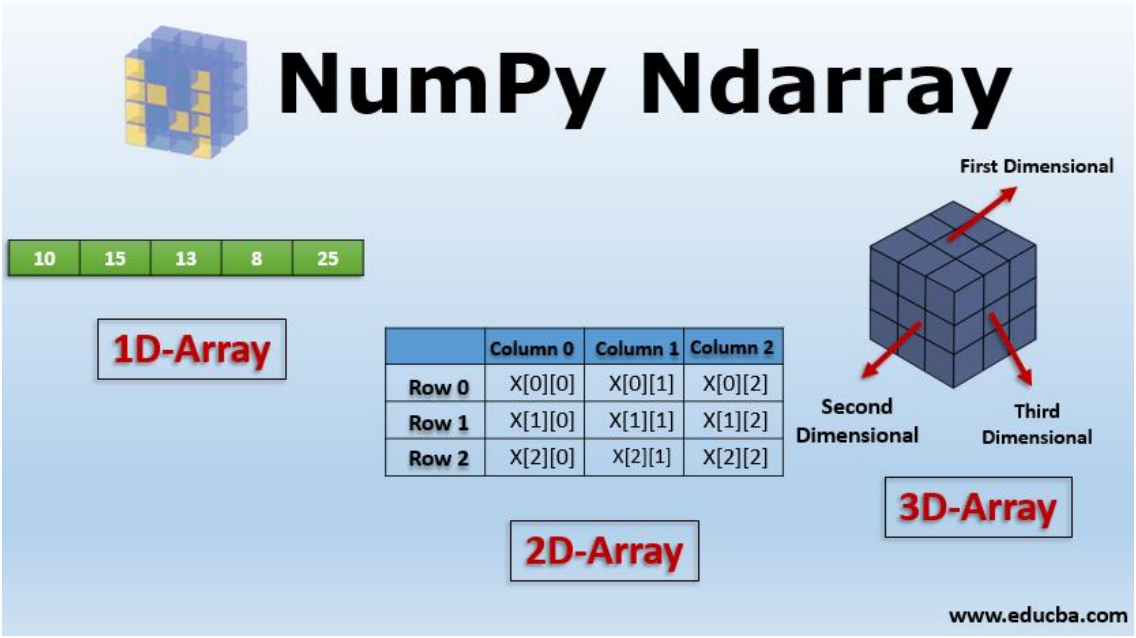


Рисунок 2.8 - Вектори NumPy

Наявність великої кількості інструментів для операцій з масивами, включаючи математичні операції, індексацію, зрізання, злиття та розбиття масивів, робить цю бібліотеку незамінною у розробці алгоритмів, від яких потребується швидке виконання роботи. У цій бібліотеці присутні потужні функції лінійної алгебри, такі як розв'язання систем лінійних рівнянь, обчислення власних значень та векторів, обернення матриць та інші.

Оскільки NumPy є основною бібліотекою для наукових обчислень та обробки даних у мові програмування Python, вона є необхідною для будь-якого, хто працює з числовими даними. Більшість бібліотек для наукових обчислень та аналізу даних в мові Python побудовані на основі NumPy, тому знання цієї бібліотеки є важливим для розуміння та використання більш складних інструментів.[12]

2.3.3 Pandas

Pandas - це потужна бібліотека для аналізу та обробки даних у мові

програмування Python. Вона надає широкі можливості для маніпулювання структурованими даними, такими як таблиці, рядки та стовпці, які називаються об'єктами DataFrame.[14]

Основні структури даних в Pandas:

1. DataFrame: це двовимірна таблиця (рис. 2.9), яка складається з рядків та стовпців. Кожен стовпець у DataFrame представляє собою об'єкт Series, що містить дані одного типу.

2. Series: це одновимірний масив мітокованих даних. Він може містити дані будь-якого типу (числові, рядкові, булеві тощо) та має індекс для ідентифікації кожного елемента.

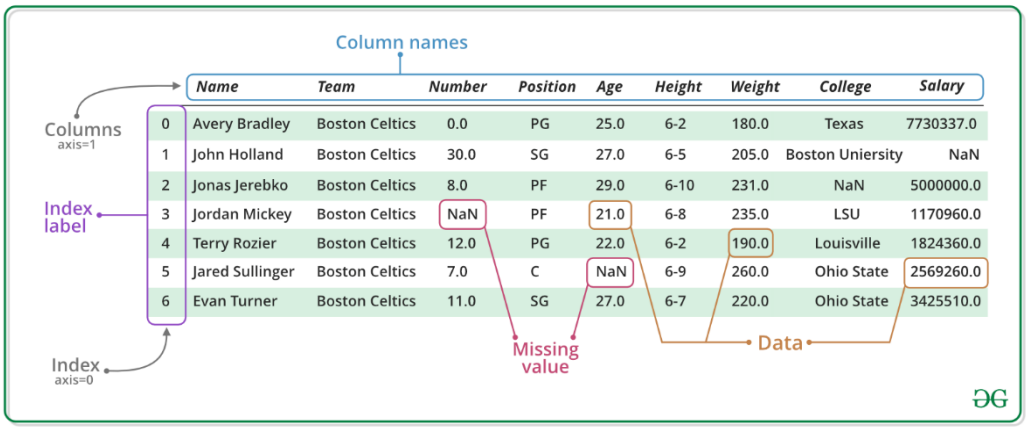


Рисунок 2.9 - Таблиця Pandas

Основні функціональні можливості Pandas:

1. Завантаження та збереження даних: Pandas може читати дані з різних джерел, таких як CSV-файли, бази даних, Excel-файли тощо. Вона також надає можливість зберігати дані у різних форматах.

2. Обробка та маніпулювання даними: Pandas дозволяє виконувати різноманітні операції з даними, такі як вибірка, фільтрація, сортування, об'єднання, групування, агрегування, перетворення та багато інших. Завдяки цьому можна ефективно виконувати аналіз та підготовку даних для моделювання.

3. Обробка пропущених даних: Pandas надає зручні методи для виявлення та роботи з пропущеними даними, такими як заповнення пропусків, видалення

рядків або стовпців з пропущеними значеннями.

4. Візуалізація даних: Pandas інтегрується з бібліотекою візуалізації Matplotlib, що дозволяє створювати високоякісні графіки та діаграми для візуалізації даних.

5. Обробка дат та часу: Pandas надає потужні засоби для роботи з датами та часом, зокрема для перетворення, індексування та аналізу часових рядів.

Pandas є однією з найпопулярніших бібліотек для аналізу даних у середовищі Python, оскільки вона проста у використанні, має велику кількість функціональних можливостей та добре поєднується з іншими інструментами для наукових обчислень та машинного навчання.

2.3.4 Scikit-learn

Scikit-learn - це бібліотека для машинного навчання написана для мови програмування Python. Тут містяться основні інструменти для класифікації, регресії, кластеризації, аналізу текстів та зображень. Всередині Scikit-learn обчислення виконуються за допомогою пакетів NumPy та SciPy, що забезпечує швидкість та ефективність роботи.[15]

У Scikit-learn містяться:

1) багато популярних моделей машинного навчання, такі як наївний баєсівський класифікатор, лінійні моделі, алгоритми на основі дерев, метод опорних векторів та інші.

2) інструменти для перехресної перевірки, що дозволяє оцінити ефективність моделей на даних, які не використовувалися для їх тренування.

3) інструменти для попередньої обробки даних, таких як видалення відсутніх значень, масштабування та обробки категоріальних змінних.

4) інструменти для автоматичного вибору оптимальних параметрів моделей та їх тестування на тренувальних даних, що дозволяє покращити їх ефективність.

Scikit-learn є основною бібліотек машинного навчання. Наявність багатьох алгоритмів та їх глибоке налаштування робить її потужним інструментів, який

спрощує життя у роботі з класичними моделями.

2.3.5 TenzorFlow та Keras

TensorFlow та Keras - це дві бібліотеки глибокого машинного навчання в мові програмування Python. TensorFlow - це відкрите програмне забезпечення для машинного навчання та штучного інтелекту, розроблене компанією Google. Вона дозволяє швидко створювати та навчати моделі машинного навчання, використовуючи граф обчислень. TensorFlow забезпечує підтримку для різних типів моделей, включаючи глибокі нейронні мережі, рекурентні нейронні мережі, кластеризацію та багато іншого. Ця бібліотека підтримує використання GPU для обчислень, яке прискорило всі розрахунки, що значно просунуло вперед розвиток нейронних мереж з великою архітектурою. Процес побудови моделей за допомогою цих фреймворків показаний на рис. 2.10 [1]

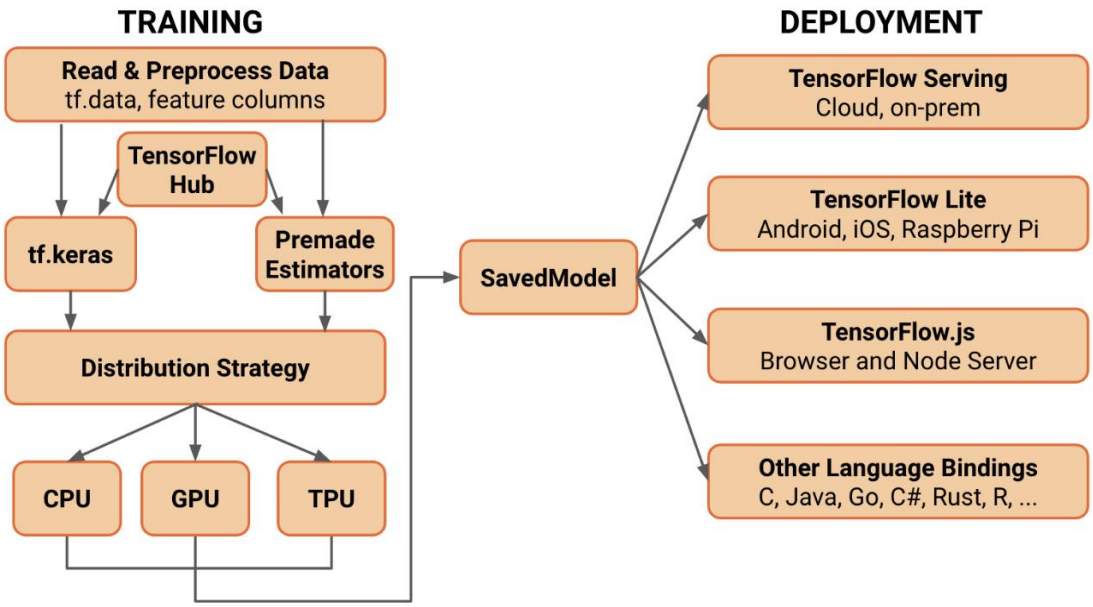


Рисунок 2.10 - Побудова та деплой моделей

Keras - це високорівневий інтерфейс для машинного навчання, розроблений для спрощення процесу створення та навчання моделей машинного навчання. Ця бібліотека є обгорткою TensorFlow, яка зберігає майже весь функціонал основного інструменту. Вона надає легкий та зрозумілий інтерфейс для створення моделей

машинного навчання та їх навчання. Keras має широкий спектр функціональності, включаючи підтримку для глибоких нейронних мереж, рекурентних нейронних мереж, згорткових нейронних мереж та багато іншого.[16]

TensorFlow та Keras мають багато переваг. Одним з них є наявність багатьох вбудованих архітектур нейронних мереж та слоїв для побудови своїм моделей. Вони можуть працювати з різними типами даних, включаючи зображення, текст та числові дані. TensorFlow є більш складною, але використання GPU для прискорення обчислень є суттєвою перевагою. На сам перед Keras надає простий та зрозумілий інтерфейс для створення моделей машинного навчання, що дозволяє легко розуміти та налаштовувати моделі.

Узагальнюючи, TensorFlow та Keras - це потужні та популярні бібліотеки машинного навчання в мові програмування Python, які дозволяють створювати та навчати моделі машинного навчання різного типу з легкістю. Обидві бібліотеки мають велику та активну спільноту розробників та підтримуються відомими компаніями, такими як Google та Microsoft.

2.3.6 PyQt5

PyQt5 - це бібліотека Python для створення графічного інтерфейсу користувача (GUI), яка базується на фреймворку Qt. Вона дозволяє створювати різноманітні інтерактивні додатки, включаючи програми з графічним інтерфейсом, інструменти для аналізу даних та інші інтерактивні застосунки. Одною з великих переваг це підтримка різних операційних систем, такі як Windows, macOS, Linux. PyQt5 є вільним та відкритим програмним засобом, що дозволяє користувачам змінювати та розповсюджувати код. Простий та зрозумілий інтерфейс дозволяє швидко створювати та налаштовувати інтерфейс користувача. Бібліотека надає доступ до багатьох візуальних елементів, таких як кнопки, текстові поля, списки та інші, що дозволяє створювати інтерактивні програми.[17]

Узагальнюючи, PyQt5 є потужною та популярною бібліотекою Python для створення графічного інтерфейсу користувача. Вона дозволяє легко створювати та налаштовувати інтерфейс користувача з використанням різних візуальних

елементів та має багато інших корисних функцій, таких як захист програми. Вона має добре розроблену документацію та велике співтовариство користувачів, що допомагає вирішувати проблеми та знаходити відповіді на питання.

2.4 Середовища розробки

2.4.2 Jupyter Notebook

Jupyter Notebook - це інтерактивна середовище для програмування та обчислювальної науки, яке дозволяє користувачам створювати та ділитися документами, які містять живий код, текстові коментарі, графіки та інші типи вмісту. Він дає змогу користувачам легко створювати та відтворювати програми, що забезпечує швидкий та ефективний процес розробки. Jupyter Notebook заснований на веб-технологіях та підтримує декілька мов програмування, включаючи Python, R, Julia, та інші. Можна виконувати код безпосередньо в документі та відображати результати обчислень у вигляді графіків, таблиць, та інших типів вмісту. Крім того, користувачі можуть використовувати його для створення презентацій та демонстрацій.[18]

Кожен документ у Jupyter Notebook складається з послідовності клітинок, які можуть містити код, текстові коментарі, математичні формули, та інші типи вмісту. Користувачі можуть виконувати окремі клітинки коду окремо, що дозволяє їм тестувати та налагоджувати код по частинах, а не весь проект в цілому.

Jupyter Notebook має ряд переваг для програмістів та дослідників:

1. Інтерактивність: Jupyter Notebook дозволяє користувачам взаємодіяти з кодом та відображати результати обчислень безпосередньо у документі.
2. Розширюваність: Jupyter Notebook підтримує багато різних мов програмування та дозволяє встановлювати різноманітні бібліотеки та плагіни.
3. Переносимість: Jupyter Notebook можна використовувати на різних платформах та операційних системах, що забезпечує легкий перехід між ними.
4. Розподіленість: Jupyter Notebook дозволяє декільком користувачам працювати над одним документом одночасно, що полегшує співпрацю та ділиться результатами.

5. Відкритість: Jupyter Notebook - це відкрите програмне забезпечення, що означає, що користувачі можуть переглядати та редагувати вихідний код, що дозволяє додавати нові функції та покращувати існуючі.

6. Повторюваність: Jupyter Notebook дозволяє повторювати обчислення та аналіз з легкістю, що полегшує розв'язання проблем та відповіді на запитання у майбутньому.

7. Візуалізація: Jupyter Notebook дозволяє відображати результати обчислень у вигляді графіків, діаграм, та інших типів візуалізації, що дозволяє користувачам легко сприймати та аналізувати дані.

8. Розробка та навчання: Jupyter Notebook дозволяє студентам, викладачам та дослідникам ефективно використовувати живий код, що полегшує розробку та навчання.

9. Документація: Jupyter Notebook дозволяє створювати документацію та звіти, що допомагає користувачам зберігати та організовувати свої дослідження та проекти.

Загалом, Jupyter Notebook - це потужний інструмент для програмістів та дослідників, який дозволяє ефективно працювати з живим кодом та відображати результати обчислень у вигляді візуалізацій та іншого типу вмісту. Jupyter Notebook підтримує багато різних мов програмування та дозволяє встановлювати різноманітні бібліотеки та плагіни, що дозволяє користувачам пристосувати середовище до їх потреб та завдань. Jupyter Notebook також дозволяє зберігати та ділитися з іншими користувачами своїми проектами та дослідженнями, що сприяє співпраці та обміну знаннями.

Jupyter Notebook - це потужний інструмент для розробки та аналізу програмного забезпечення, який дозволяє працювати з живим кодом та візуалізувати результати обчислень. Він підтримує багато різних мов програмування та дозволяє пристосувати середовище до потреб користувача. Також це середовище може бути використаний для викладання та навчання, оскільки він дозволяє створювати документи, які містять не тільки код, а й текстову та графічну інформацію. Загалом, це є потужним та універсальним інструментом, який варто

розглянути для використання в програмуванні, дослідженнях та навчанні.

2.4.3 PyCharm

PyCharm - це інтегроване середовище розробки (IDE) для мови програмування Python, яке надає розширений набір інструментів для зручного і продуктивного програмування. Розроблене компанією JetBrains, PyCharm стало популярним вибором серед розробників Python завдяки своїм функціям, що полегшують розробку, налагодження та керування проектами.[19]

Основні особливості PyCharm:

1. Редактор коду: PyCharm має потужний редактор коду з відмінною підсвіткою синтаксису, автодоповненням, перевіркою на відповідність стилю коду та багатьма іншими корисними функціями. Він також підтримує редагування коду з використанням віртуальних середовищ, що полегшує роботу з різними версіями Python.

2. Управління проектами: PyCharm дозволяє зручно створювати, керувати та відкривати проекти Python. Він підтримує різні типи проектів, включаючи консольні програми, веб-додатки, наукові проекти та інше. Інтеграція з системами контролю версій, такими як Git, дозволяє легко відстежувати та керувати змінами в коді.

3. Аналіз коду та відлагодження: PyCharm має потужні інструменти для аналізу коду, включаючи перевірку помилок, підказки про можливі вдосконалення та оптимізації коду. Він також підтримує відлагодження коду, що дозволяє крок за кроком виконувати програму та відстежувати значення змінних, допомагаючи виявляти та виправляти помилки.

4. Підтримка веб-розробки: PyCharm надає можливості для розробки веб-додатків на базі Python, включаючи підтримку Django, Flask, Pyramid та інших популярних веб-фреймворків. Він має функції, такі як автоматичне завершення коду, підказки про структуру проекту та інструменти для роботи з базами даних.

5. Інтеграція з інструментами та зовнішніми сервісами: PyCharm підтримує інтеграцію з популярними інструментами розробки, такими як IPython,

Jupyter Notebook, Docker та іншими. Він також надає підтримку для використання систем віртуальних середовищ, дистанційного відладження та інших розширень.

6. Тестування та профілювання: PyCharm має вбудовані інструменти для написання та виконання модульних тестів. Він також надає можливості для профілювання коду з метою виявлення його швидкодії та виправлення проблем ефективності.

7. Підтримка інших мов та фреймворків: PyCharm не обмежується лише Python і підтримує інші мови програмування, такі як JavaScript, HTML, CSS, SQL та інші. Він також надає функціональність для розробки проектів, що використовують інші фреймворки, наприклад, Angular, React та інші.

PyCharm доступний в двох версіях: Community Edition (безкоштовна) та Professional Edition (платна). Версія Professional має більшу кількість функцій, включаючи додаткову підтримку фреймворків та інструментів. Однак, обидві версії надають потужні можливості для розробки проектів на Python.

Висновок до розділу 2

У даному розділі теоретичної частини дипломної роботи був проведений детальний огляд теоретії потрібної для програмного додатку для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі. Досліджено основні концепції, алгоритми та методи, які застосовуються у сфері медичного зображення та машинного навчання. Перш за все, були розглянуті основні принципи функціонування головного мозку та різні типи пухлин, що можуть виникати у ньому. Відзначено, що точна та швидка ідентифікація пухлин є критично важливою для успішного лікування та прогнозування хвороби. Подальше дослідження зосереджено на згорткових нейронних мережах (ЗНМ), які є потужними інструментами для обробки медичних зображень. Описано структуру та принципи роботи ЗНМ, зокрема їх здатність до автоматичного вивчення важливих ознак зображень та здатність до класифікації об'єктів. Відзначено, що ЗНМ вже успішно використовуються в медицині для розпізнавання та діагностики

пухлин різних органів, включаючи головний мозок. Також були проаналізовані різні підходи до побудови ЗНМ для ідентифікації пухлин головного мозку. Розглянуто різні слої, що зі значними об'ємами даних може допомогти досягти кращих результатів у задачах класифікації пухлин головного мозку. Загальний висновок з теоретичної частини полягає у тому, що розробка програмного додатку для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі є актуальною і перспективною задачею. Використання ЗНМ дозволяє досягти високої точності та швидкодії при класифікації пухлин.

РОЗДІЛ 3
ПРАКТИЧНА ЧАСТИНА

3.1. Побудова та тестування моделей

Побудова та тестування моделей для ідентифікації пухлин головного мозку на зображеннях є важливим завданням в сфері медичного образування та діагностики. Пухлини головного мозку є серйозними захворюваннями, які можуть вимагати негайного втручання та лікування. Вірна та точна ідентифікація пухлин дозволяє швидше та ефективніше встановити діагноз, визначити підходящий план лікування та покращити прогнози для пацієнтів.

За допомогою передових технологій обробки зображень та машинного навчання, вже досягнуто значних успіхів у розпізнаванні та класифікації пухлин головного мозку на медичних зображеннях, таких як комп'ютерні томографії (КТ) та магнітно-резонансні томографії (МРТ). Використання алгоритмів та моделей глибокого навчання дозволяє автоматизувати процес аналізу та допомагає лікарям зробити швидкі та точні висновки з медичних зображень.

У цій практичній частині ми зосередимось на побудові та тестуванні моделей для ідентифікації пухлин головного мозку на зображеннях. Ми будемо використовувати зібрані дані з медичних досліджень, які містять реальні зображення пацієнтів з різними видами пухлин головного мозку. Під час цієї практичної роботи ми будемо досліджувати та використовувати різні підходи до обробки зображень та моделювання. Ми будемо розглядати різні архітектури глибоких нейронних мереж, такі як згорткові нейронні мережі (CNN), які ефективно працюють з зображеннями, та будемо налаштовувати їх параметри для досягнення найкращих результатів. Крім того, ми розглянемо методи підготовки та попередньої обробки даних зображень та нормалізацію. Також ми проведемо оцінку моделей, використовуючи метрики якост.

Ця практична робота надасть вам практичні навички у побудові та тестуванні моделей для ідентифікації пухлин головного мозку на зображеннях. Ми будемо

мати можливість експериментувати з різними підходами та параметрами моделей, а також вивчити процес підготовки даних та оцінку моделей. Придбані знання та навички можуть бути застосовані в реальних медичних дослідженнях та діагностичних процедурах, що сприятимуть поліпшенню медичної практики та збереженню людських життів.

Підготовка зображень для моделі

Підготовка зображення до моделі включає кілька кроків для оптимального представлення зображення у вигляді числових даних, які можуть бути подані моделі для обробки. Основні кроки підготовки зображення включають наступне:

1. Завантаження зображення: Першим кроком є завантаження зображення з файлу або отримання його з іншого джерела, наприклад, з веб-камери.

2. Зміна розміру зображення: Залежно від вимог моделі, може бути потрібно змінити розмір зображення. Це може бути досягнуто зміною ширини та висоти зображення за допомогою методу зміни розміру або обрізання зображення.

3. Нормалізація пікселів: Часто зображення нормалізують, щоб відрізняти його пікселів у межах відповідного діапазону значень. Зазвичай пікселі нормалізуються у діапазоні від 0 до 1 або у діапазоні від -1 до 1. Це може допомогти поліпшити стійкість та швидкість навчання моделі.

4. Перетворення кольорового простору: В деяких випадках може бути потрібно змінити кольоровий простір зображення. Наприклад, зображення може бути перетворене у відтінки сірого або перетворене з RGB у HSV (або навпаки), якщо це відповідає потребам моделі.

5. Додавання розмірності: Деякі моделі можуть вимагати вхідних зображень з певною кількістю розмірностей. Наприклад, згорткові нейронні мережі очікують вхідні зображення у форматі (ширина, висота, канали). Тому, якщо зображення не має необхідних розмірностей, може бути потрібно додати або видалити деякі розмірності, щоб вони відповідали вимогам моделі.

6. Приведення до числового представлення: Нарешті, зображення перетворюється у числове представлення, яке може бути подано моделі. Зазвичай це є числовий масив або тензор, де кожен елемент представляє собою значення

пікселя.

Ці кроки підготовки зображення можуть варіюватися залежно від потреб конкретної моделі та завдання, яке ви розв'язуєте. Використання бібліотеки для обробки зображень, такої як OpenCV або PIL (Python Imaging Library), може значно спростити цей процес.

Наш датасет складається з КТ знімків головного мозку здорових людей та хворих на рак. Ми маємо 4600 унікальних екземплярів з яких 55% є знімки хворих та 45% здорових людей. Перед нами стоїть задача класифікації. Для зручної розробки та тестування моделі використовувалося середовище Jupyter Notebook. Було розроблено декілька архітектур, та досліджена їх точність на тренувальному датасеті та валідаційному.

Першим що хотілося перевірити, як справляється з задачею класифікації зображень череда повнозв’язних шарів. Архітектура зображена на рис. 3.1.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_3 (Flatten)         (None, 67500)             0

 dense_9 (Dense)             (None, 128)               8640128

 dense_10 (Dense)            (None, 64)                8256

 dense_11 (Dense)            (None, 32)                2080

 dense_12 (Dense)            (None, 1)                 33

=================================================================
Total params: 8,650,497
Trainable params: 8,650,497
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.1 - Архітектура повнозв’язних шарів

Архітектура складається із першого шару Flatten, який переводить зображення у одномірний масив, для передачі на вхід повнозв’язним шарам, та з 4

повнозв'язних шарів з різною кількістю нейронів. Така архітектура має велику кількість змінних параметрів, яка сягає 8 650 497 одиниць. Сама по собі архітектура не може досліджувати певні елементи на зображенні, а бере зображення повністю, що зменшує ефективність та точність архітектури. Перейдемо до результатів (рис. 3.2 – рис. 3.4).

```
Epoch 1/10
15/15 [==============================] - 22s 1s/step - loss: 2.4000 - accuracy: 0.5422 - val_loss: 0.5689 - val_accuracy: 0.776
9
Epoch 2/10
15/15 [==============================] - 20s 1s/step - loss: 0.6010 - accuracy: 0.6732 - val_loss: 0.5434 - val_accuracy: 0.731
2
Epoch 3/10
15/15 [==============================] - 20s 1s/step - loss: 0.5666 - accuracy: 0.7039 - val_loss: 0.5595 - val_accuracy: 0.716
0
Epoch 4/10
15/15 [==============================] - 20s 1s/step - loss: 0.5687 - accuracy: 0.7118 - val_loss: 0.4554 - val_accuracy: 0.803
0
Epoch 5/10
15/15 [==============================] - 20s 1s/step - loss: 0.5187 - accuracy: 0.7596 - val_loss: 0.5722 - val_accuracy: 0.700
8
Epoch 6/10
15/15 [==============================] - 20s 1s/step - loss: 0.4449 - accuracy: 0.8079 - val_loss: 0.5066 - val_accuracy: 0.760
6
Epoch 7/10
15/15 [==============================] - 20s 1s/step - loss: 0.4159 - accuracy: 0.8286 - val_loss: 0.3945 - val_accuracy: 0.831
3
Epoch 8/10
15/15 [==============================] - 20s 1s/step - loss: 0.3854 - accuracy: 0.8514 - val_loss: 0.5001 - val_accuracy: 0.775
8
Epoch 9/10
15/15 [==============================] - 20s 1s/step - loss: 0.3603 - accuracy: 0.8601 - val_loss: 0.3795 - val_accuracy: 0.827
0
Epoch 10/10
15/15 [==============================] - 20s 1s/step - loss: 0.3373 - accuracy: 0.8723 - val_loss: 0.4383 - val_accuracy: 0.819
4
```

Рисунок 3.2 - Результати навчання архітектури 1

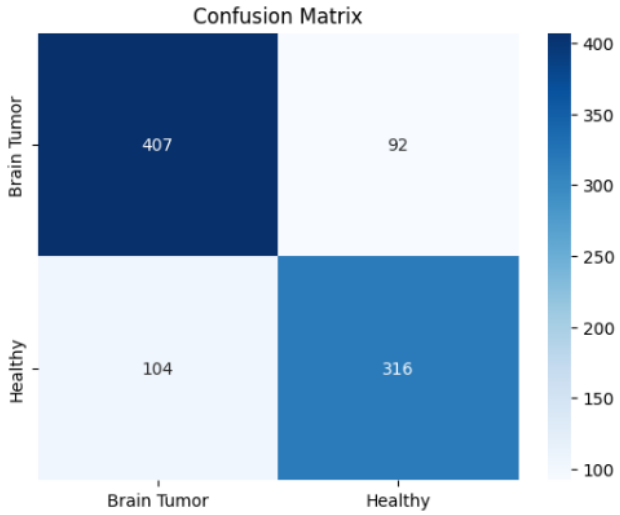


Рисунок 3.3 - Матриця конфузій

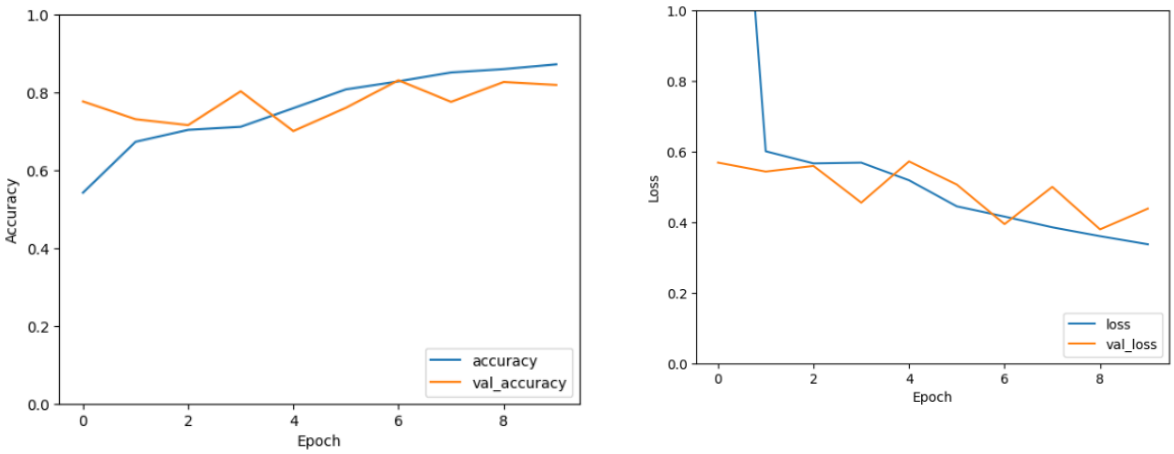


Рисунок 3.4 - Графіки точності та помилки під час навчання

Можна побачити що архітектура повнозв'язних шарів досить не погано справилася з завданням, але точність у 81% не той результат, який ми хочемо досягти. Також наявність великої кількості змінних параметрів робить складним підтримку та інтеграцію моделі. Тому цей варіант нам не підходить.

Друга архітектура вже містила згортковий шар, який мав би краще впоратися з класифікацією зображень. Тому була створена проста архітектура, яка складається з одного згорткового та двох повнозв'язниз слоїв. Між згортковим та повнозв'язним слоєм присутній Flatten (рис. 3.5).

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)            (None, 150, 150, 64)      832

 flatten_1 (Flatten)          (None, 1440000)           0

 dense_3 (Dense)              (None, 128)               184320128

 dense_4 (Dense)              (None, 1)                 129

=================================================================
Total params: 184,321,089
Trainable params: 184,321,089
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.5 - Друга архітектура зі згортковим слоєм

Ця мережа містить ще більше параметрів, тому її навчання займає довгий час.

Це тому що був використаний Dense шар без оптимізаторів таких як dropout, тому присутня велика кількість параметрів для тренування. Перейдемо до результатів (рис. 3.6. – рис. 3.8).

```
Epoch 1/10
15/15 [==============================] - 54s 4s/step - loss: 1.6393 - accuracy: 0.7120 - val_loss: 2.1028 - val_accuracy: 0.620
2
Epoch 2/10
15/15 [==============================] - 59s 4s/step - loss: 0.6227 - accuracy: 0.8280 - val_loss: 0.3156 - val_accuracy: 0.895
5
Epoch 3/10
15/15 [==============================] - 53s 3s/step - loss: 0.1887 - accuracy: 0.9351 - val_loss: 0.1727 - val_accuracy: 0.933
6
Epoch 4/10
15/15 [==============================] - 50s 3s/step - loss: 0.0691 - accuracy: 0.9813 - val_loss: 0.0827 - val_accuracy: 0.970
6
Epoch 5/10
15/15 [==============================] - 50s 3s/step - loss: 0.0356 - accuracy: 0.9908 - val_loss: 0.0819 - val_accuracy: 0.965
2
Epoch 6/10
15/15 [==============================] - 50s 3s/step - loss: 0.0206 - accuracy: 0.9973 - val_loss: 0.0343 - val_accuracy: 0.991
3
Epoch 7/10
15/15 [==============================] - 50s 3s/step - loss: 0.0108 - accuracy: 0.9995 - val_loss: 0.0296 - val_accuracy: 0.989
1
Epoch 8/10
15/15 [==============================] - 50s 3s/step - loss: 0.0066 - accuracy: 0.9997 - val_loss: 0.0291 - val_accuracy: 0.990
2
Epoch 9/10
15/15 [==============================] - 50s 3s/step - loss: 0.0046 - accuracy: 1.0000 - val_loss: 0.0317 - val_accuracy: 0.990
2
Epoch 10/10
15/15 [==============================] - 50s 3s/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.0232 - val_accuracy: 0.991
3
```

Рисунок 3.6 - Результати навчання

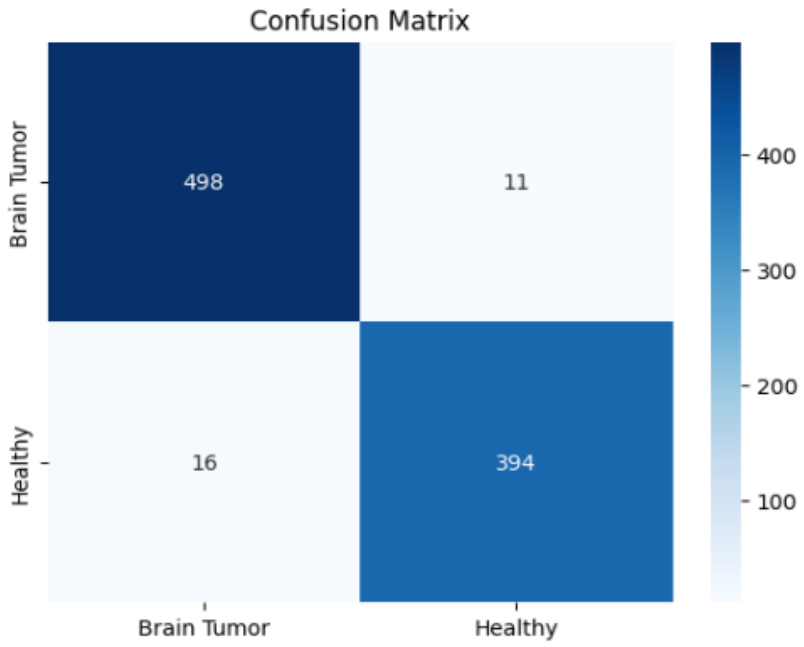


Рисунок 3.7 - Матриця конфузій

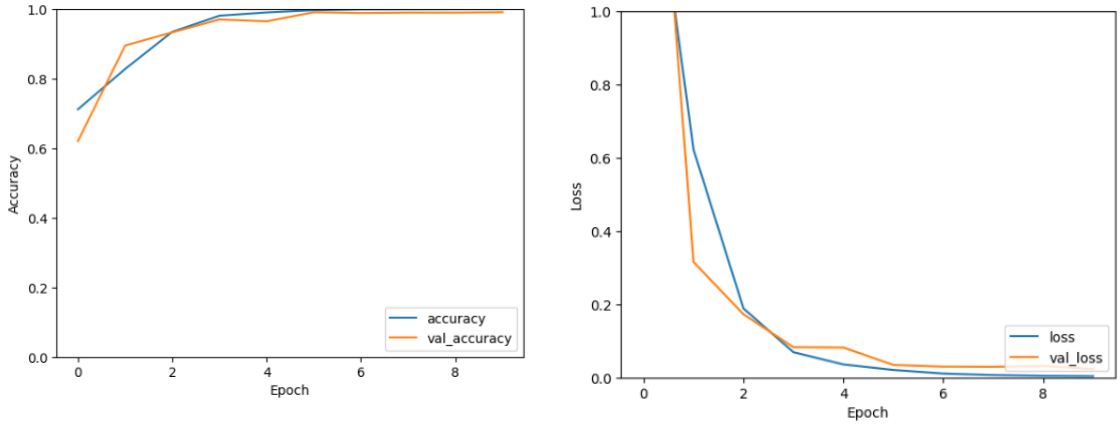


Рисунок 3.8 - Графіки точності та помилки під час навчання

Бачимо що мережа досить гарно навчилася класифікувати картинки, але наявність великої кількості параметрів є значним мінусом.

Перейдемо до 3 архітектури, яка краще оптимізована та має гарний результат. Вона включає в себе згорткові, повнозв'язні шари, макспулінги та дропаути. Така архітектура дає можливість узагальнити знімок та виявити загальні тенденції для визначення класу. Це здобувається за допомогою MaxPooling та Dropout, вони виступають у ролі регуляризаторів, які не дають мережі перенавчитися (рис. 3.9).

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 150, 150, 32)      896

 max_pooling2d_6 (MaxPooling  (None, 75, 75, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 75, 75, 64)        18496

 max_pooling2d_7 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 flatten_4 (Flatten)         (None, 87616)             0

 dense_13 (Dense)            (None, 128)               11214976

 batch_normalization_6 (Batc  (None, 128)              512
 hNormalization)

 dropout_4 (Dropout)         (None, 128)               0

 dense_14 (Dense)            (None, 64)                8256

 batch_normalization_7 (Batc  (None, 64)               256
 hNormalization)

 dropout_5 (Dropout)         (None, 64)                0

 dense_15 (Dense)            (None, 1)                 65

=================================================================
Total params: 11,243,457
Trainable params: 11,243,073
Non-trainable params: 384
```

Рисунок 3.9 - Третя архітектура зі згортковим слоєм

Перейдемо до результатів останньої мережі, вона показала гарний результат та швидкість роботи, що є оптимальним вибором для нашого додатку (рис. 3.10 – рис. 3.12). Це було досягнуто за допомогою шарів dropout та maxpooling, що значно обмежили кількість тренувальних параметрів, при цьому суттєво не зменшивши точність нашої моделі.

```
Epoch 1/10
15/15 [==============================] - 49s 3s/step - loss: 0.6494 - accuracy: 0.6919 - val_loss: 0.9060 - val_accuracy: 0.731
2
Epoch 2/10
15/15 [==============================] - 48s 3s/step - loss: 0.4214 - accuracy: 0.8185 - val_loss: 0.3489 - val_accuracy: 0.854
2
Epoch 3/10
15/15 [==============================] - 49s 3s/step - loss: 0.2537 - accuracy: 0.8981 - val_loss: 0.3128 - val_accuracy: 0.877
0
Epoch 4/10
15/15 [==============================] - 49s 3s/step - loss: 0.1609 - accuracy: 0.9473 - val_loss: 0.2432 - val_accuracy: 0.955
4
Epoch 5/10
15/15 [==============================] - 43s 3s/step - loss: 0.1005 - accuracy: 0.9728 - val_loss: 0.1654 - val_accuracy: 0.965
2
Epoch 6/10
15/15 [==============================] - 45s 3s/step - loss: 0.0614 - accuracy: 0.9867 - val_loss: 0.1144 - val_accuracy: 0.989
1
Epoch 7/10
15/15 [==============================] - 47s 3s/step - loss: 0.0415 - accuracy: 0.9919 - val_loss: 0.1219 - val_accuracy: 0.991
3
Epoch 8/10
15/15 [==============================] - 51s 3s/step - loss: 0.0281 - accuracy: 0.9954 - val_loss: 0.1040 - val_accuracy: 0.996
7
Epoch 9/10
15/15 [==============================] - 48s 3s/step - loss: 0.0194 - accuracy: 0.9978 - val_loss: 0.0754 - val_accuracy: 0.995
6
Epoch 10/10
15/15 [==============================] - 42s 3s/step - loss: 0.0168 - accuracy: 0.9978 - val_loss: 0.0756 - val_accuracy: 0.993
5
```

Рисунок 3.10 - Результати навчання

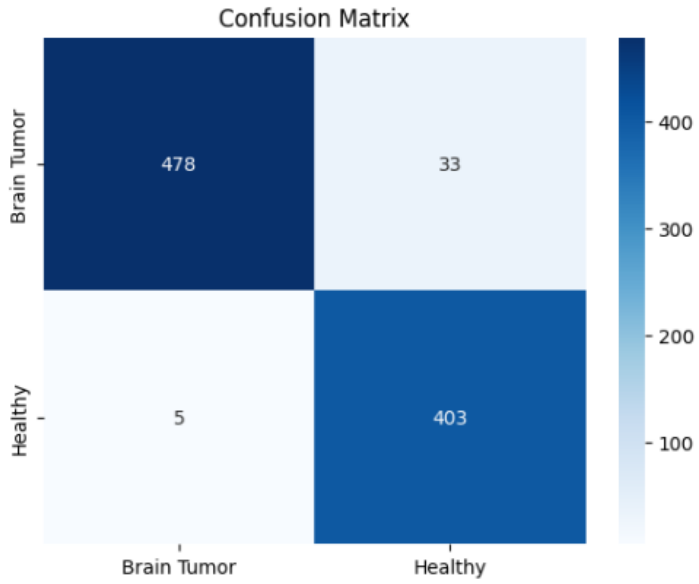


Рисунок 3.11 - Матриця конфузій

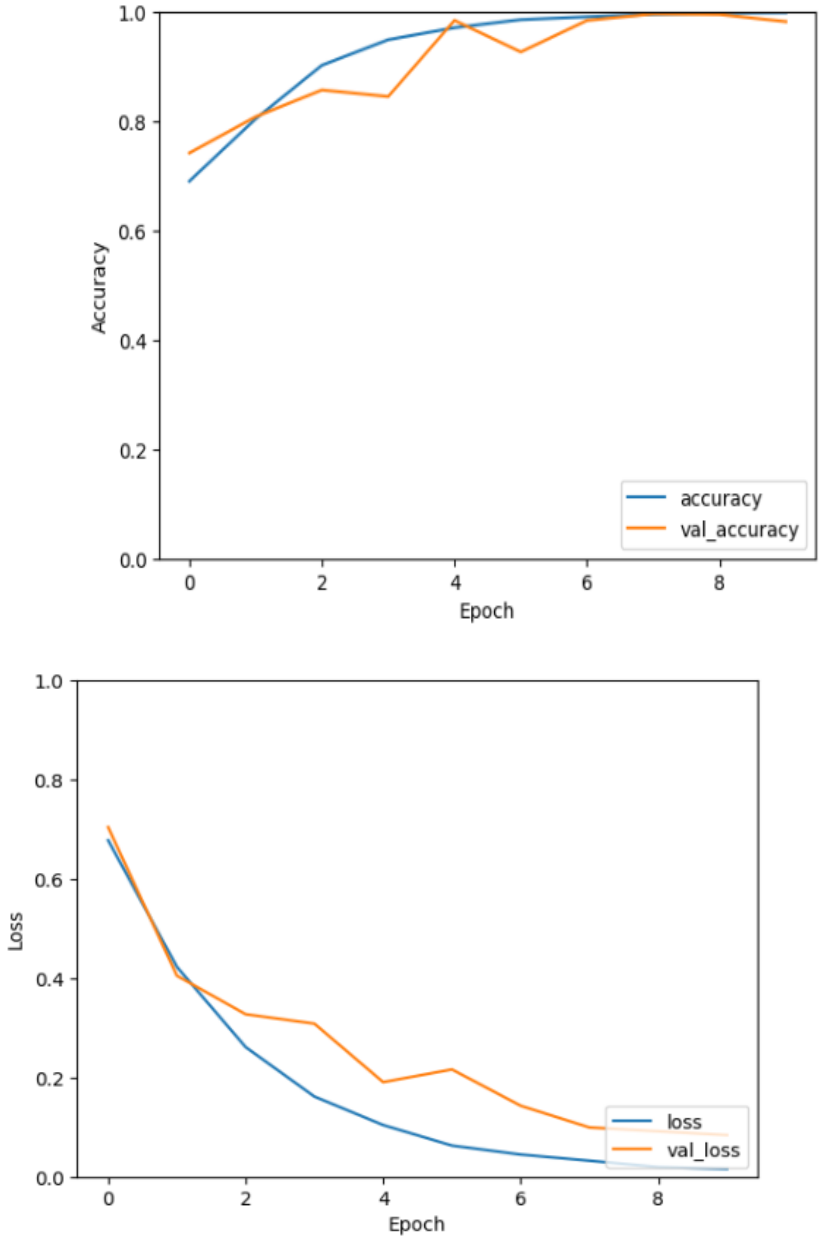


Рисунок 3.12 - Графіки точності та помилки під час навчання

Маємо гарну точність та оптимальну швидкість навчання, тому це фінальний вибір архітектури.

3.2. Розробка графічного інтерфейсу

3.2.1. Загальний огляд додатку

Побудова додатку для ідентифікації пухлин головного мозку вимагає створення інтерфейсу користувача, який дозволяє легко та ефективно взаємодіяти з

програмою для аналізу медичних зображень. Основним завданням цього інтерфейсу є забезпечення зручного способу завантаження зображень та відображення результатів.

Ось кілька ключових елементів, які можна включити до інтерфейсу користувача для практичного розділу побудови додатку для ідентифікації пухлин головного мозку:

1. Панель навігації, де можна розмістити основні функції додатку, такі як завантаження зображень, запуск аналізу та доступ до результатів. Цей елемент може бути розташований зверху або зліва на інтерфейсі.

2. Завантаження зображень. Можливість завантажувати зображення з комп'ютера користувача або з інших джерел, таких як медичні бази даних. Для зручності використання, додайте підтримку для перетягування та випадаючих списків для вибору зображень.

3. Візуалізація результатів. Після завершення аналізу, відобразіть результати на інтерфейсі. Це може бути графічне представлення зображення головного мозку з позначеними виявленими пухлинами або вивід текстової інформації.

4. Експорт та збереження. Додайте можливість експортувати результати аналізу, наприклад, у вигляді зображень або даних у форматі, зручному для подальшого використання або обробки.

Важливо врахувати, що інтерфейс користувача повинен бути інтуїтивно зрозумілим та легким у використанні. Колірна палітра та організація елементів на екрані також мають сприяти зручному сприйняттю та навігації користувача. Також треба звернути увагу на розміщення та розмір елементів інтерфейсу, щоб забезпечити їх зручний доступ.

У процесі розробки інтерфейсу користувача для побудови додатку для ідентифікації пухлин головного мозку, варто також провести тестування з користувачами, щоб отримати зворотний зв'язок і вдосконалити дизайн та функціональність.

Для зручного користування моделлю, було прийняте рішення розробити спеціальний програмний додаток, який дозволить, не заглиблюючись у теорію

нейронних мереж, отримувати результати діагнозу по КТ знімкам головного мозку. Інтерфейс додатку має бути зрозумілим та легким у використанні, тому під час розробки ставала задача створити просту обгортку з усім потрібним для моделі.

На рис. 3.13 можна побачити головне вікно програми.

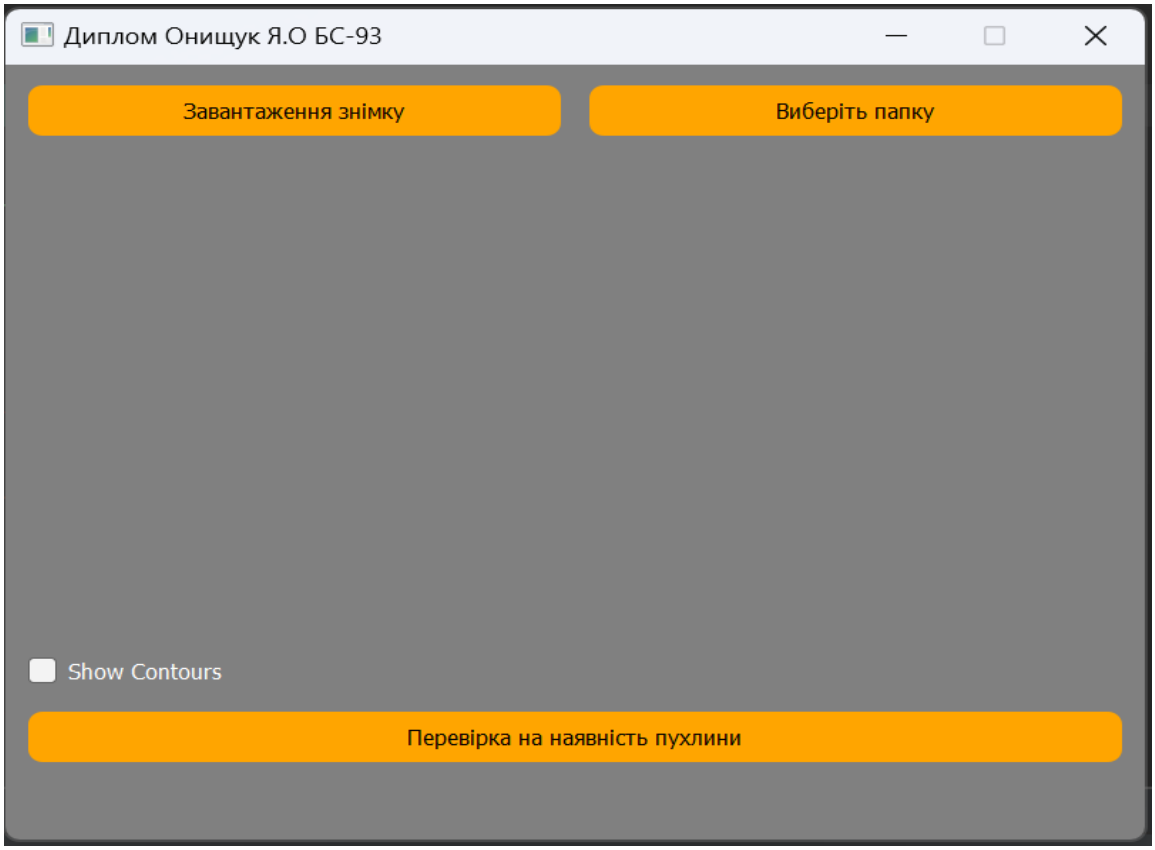


Рисунок 3.13 - Головне вікно

У головному вікні можна побачити 3 основні кнопки. Кнопка «Завантаження знімку» дозволяє завантажити знімок з комп'ютера для визначення діагнозу. Кнопка перевірка на наявність пухлини виводить результат по вибраному зображенню. Кнопка «Виберіть папку» дозволяє вибрати директорію зі знімками та отримати результат по кожному з них у Excel таблиці.

3.2.2. Демонстрація роботи програми

Під час натискання кнопки «Завантаження знімку» відкривається провідник, у якому можна обрати знімок для перевірки. Можна обрати будь-яке місце на комп'ютері, де знаходиться знімок, та завантажити його (рис. 3.14).

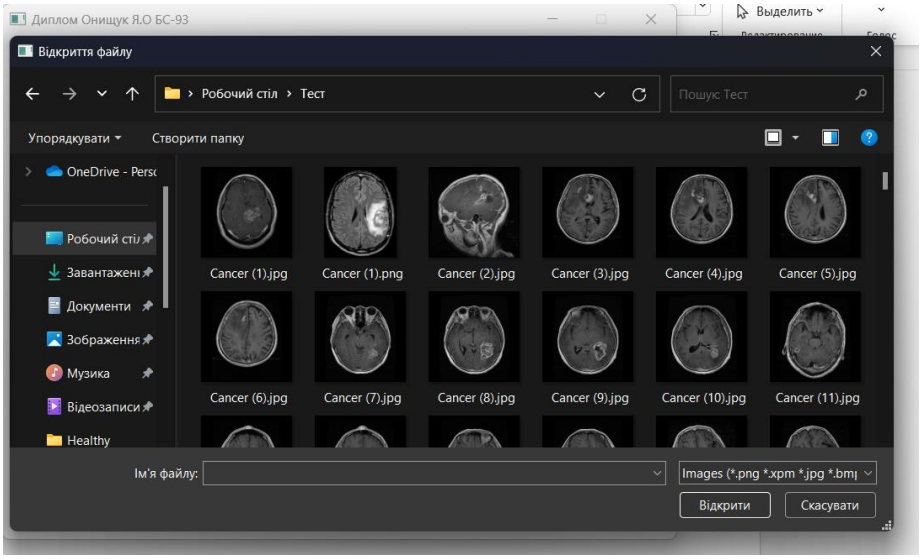


Рисунок 3.14 – Провідник

Після вибору знімка, у додатку ми його побачимо на місці де має бути завантажений знімок (рис. 3.15).

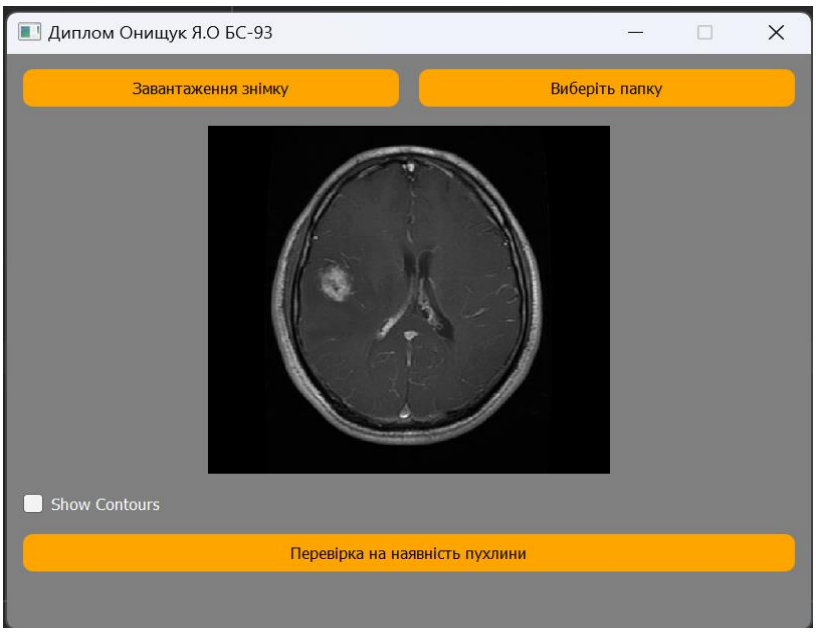


Рисунок 3.15 Відображення вибраного знімку

Далі треба натиснути кнопку «Перевірка на наявність пухлини», яка виведе результат по цьому знімку (рис. 3.16).

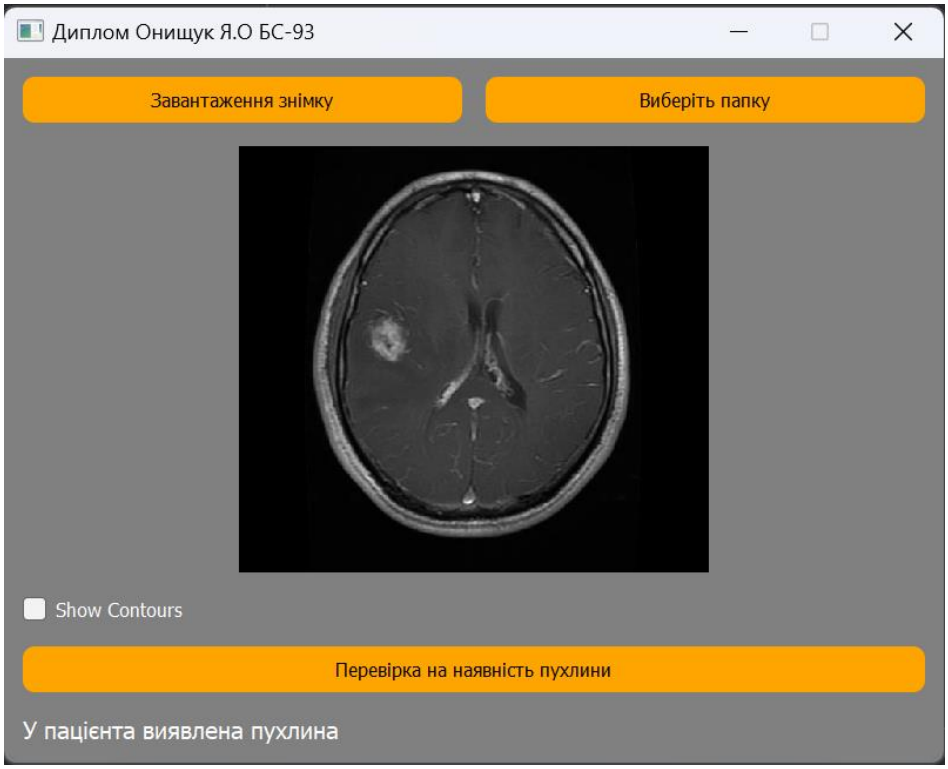


Рисунок 3.16 - Результат моделі

Флаг «Show Contours» обводить основні лінії на маюнку, це дає змогу перевірити де саме знаходиться пухлина, якщо її видно на КТ знімку (рис. 3.17).

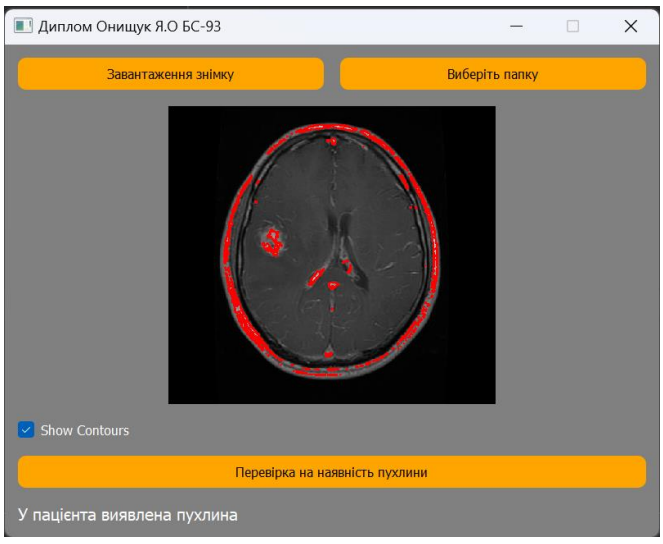


Рисунок 3.17 - Результат роботи «Show Contours»

Переходимо до останньої кнопки вибору папки, вона працює схоже з першою кнопкою, але дозволяє за раз перевірити багато зображень, та занести всі результати

у Excel таблицю (рис. 3.18 – рис. 3.19).

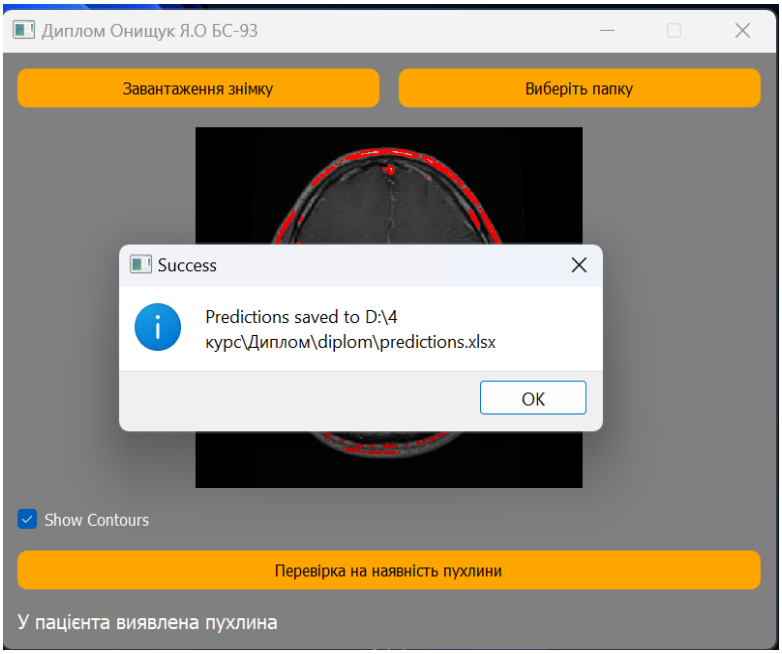
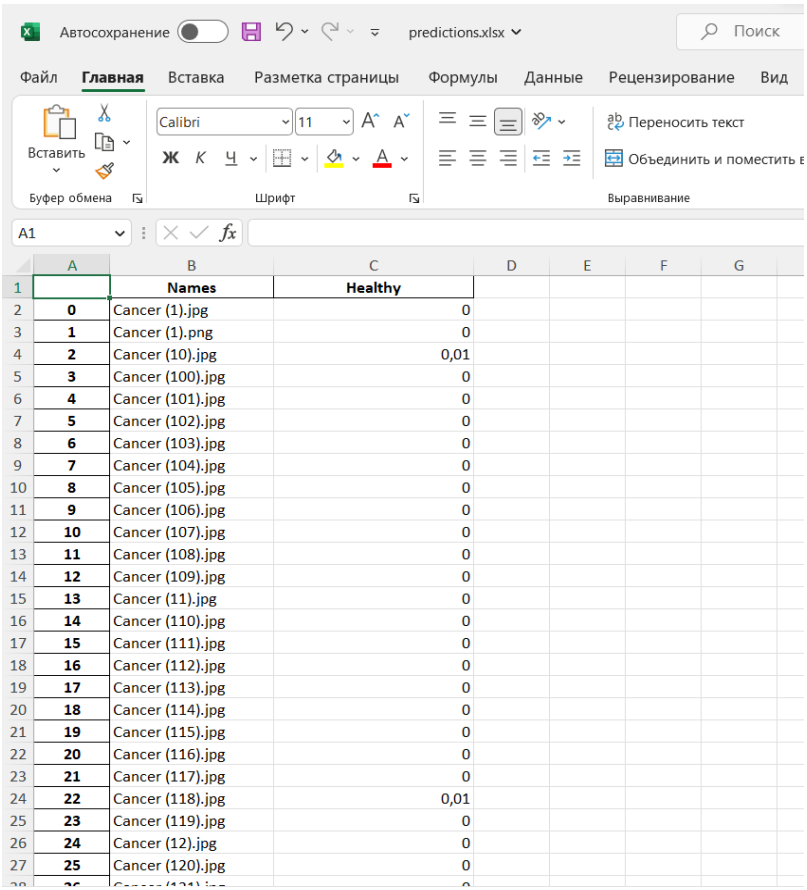


Рисунок 3.18 - Повідомлення про завершення роботи моделі



predictions.xlsx

	A	B	C
1		Names	Healthy
2	0	Cancer (1).jpg	0
3	1	Cancer (1).png	0
4	2	Cancer (10).jpg	0,01
5	3	Cancer (100).jpg	0
6	4	Cancer (101).jpg	0
7	5	Cancer (102).jpg	0
8	6	Cancer (103).jpg	0
9	7	Cancer (104).jpg	0
10	8	Cancer (105).jpg	0
11	9	Cancer (106).jpg	0
12	10	Cancer (107).jpg	0
13	11	Cancer (108).jpg	0
14	12	Cancer (109).jpg	0
15	13	Cancer (11).jpg	0
16	14	Cancer (110).jpg	0
17	15	Cancer (111).jpg	0
18	16	Cancer (112).jpg	0
19	17	Cancer (113).jpg	0
20	18	Cancer (114).jpg	0
21	19	Cancer (115).jpg	0
22	20	Cancer (116).jpg	0
23	21	Cancer (117).jpg	0
24	22	Cancer (118).jpg	0,01
25	23	Cancer (119).jpg	0
26	24	Cancer (12).jpg	0
27	25	Cancer (120).jpg	0

Рисунок 3.19 - Результат виконання

3.3. Розрахунок економічного ефекту

За темою роботи проведена оцінка основних функціональних та вартісних характеристик програмного застосунку, розробленого для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі. Програмний застосунок призначений для використання на персональних комп'ютерах під управлінням операційної системи Windows.

Морфологічна карта системи зображена на рисунку 3.20. показує різні варіанти реалізації та покриває всі можливі комбінації.

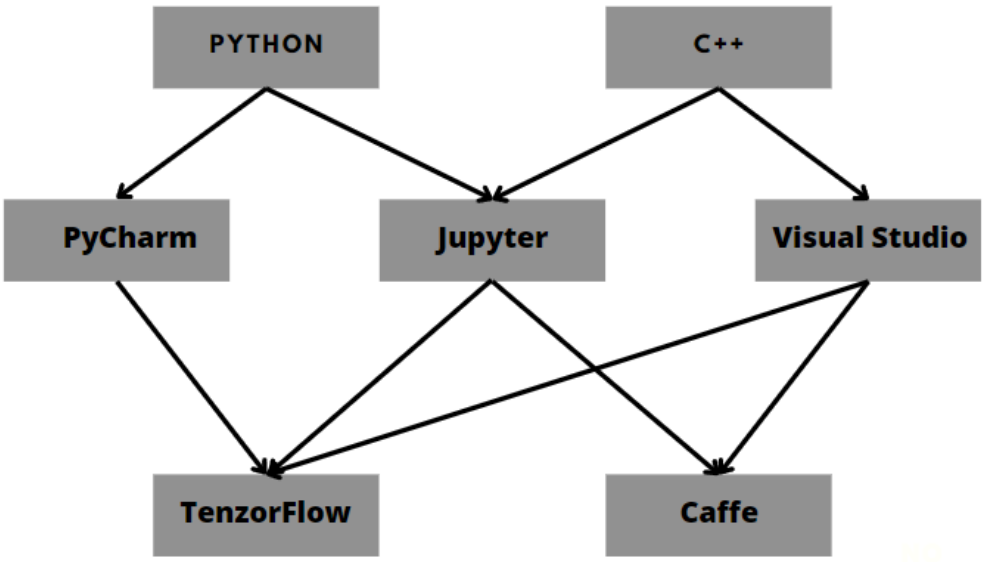


Рисунок 3.20 - Морфологічна карта варіантів реалізації функцій

На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (табл. 3.1.).

Таблиця 3.1

Позитивно-негативна матриця варіантів основних функцій

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	А	Кросплатформений, займає менше часу для написання коду, має багато зручних фреймворків для ML.	Повільність виконання коду
	Б	Висока швидкодія	Складність написання коду
F2	А	Висока гнучкість та універсальність, що дозволяє виконувати різноманітні задачі глибинного навчання. Має потужну підтримку для GPU, що робить його більш ефективним для роботи з великими обсягами даних.	Більш складний у використанні та налаштуванні, що може вимагати більшого досвіду в глибинному навчанні.
	Б	Простота використання та налаштування. Має готові моделі для різноманітних задач комп'ютерного зору та навчання машин.	Менш гнучкий та менш універсальний. Менш розвинута підтримка GPU.
F3	А	Забезпечує широкий спектр інструментів та можливостей для Python-розробки. Має потужні функції рефакторингу та автоматичного завершення коду, що полегшує процес розробки. Підтримує інтеграцію з віртуальними середовищами та системами контролю версій.	Часто може вимагати значну кількість ресурсів системи, що може призвести до сповільнення роботи.
	Б	Забезпечує інтерактивну розробку та можливість візуалізації даних в режимі реального часу. Можливість додавати текстові блоки та графічні елементи до ноутбуку.	Не дуже зручний для написання складних проектів. Немає функцій рефакторингу та автоматичного завершення коду, що робить процес розробки менш продуктивним.

Продовж. табл. 3.1

Основні функції	Варіанти реалізації	Переваги	Недоліки
	В	Забезпечує інтеграцію з різноманітними мовами програмування та технологіями, що дозволяє розробляти різноманітні проекти. Підтримує візуальну розробку та створення інтерфейсів.	Важка програма, яка вимагає багато ресурсів комп'ютера.

Розрахунок показників рівня якості представлено відповідно в табл. 3.2.

Таблиця 3.2

Розрахунок показників якості

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	2	10	0,208	2,08
F2	А	X2	64	5	0,158	0,79
		X3	1000	10	0,274	2,74
	Б	X2	64	5	0,158	0,79
		X3	3000	5	0,274	1,37
F3	А	X4	500	10	0,361	3,61
	Б	X4	500	10	0,361	3,61

За цими даними визначаємо рівень якості кожного з варіантів:

F1(А) – F2(А) – F3(А) = 2,08+ 3,53 + 3,61 = 9,22

F1(А) – F2(А) – F3(Б) = 2,08 + 3,53 + 3,61 = 9,22

F1(А) – F2(Б) – F3(А) = 2,08+ 2,16 + 3,61 = 5,77

F1(А) – F2(Б) – F3(Б) = 2,08+ 2,16 + 3,61 = 5,77

Отже, найкращим є перший та другий варіанти, для якого коефіцієнт

технічного рівня має найбільше значення.

Висновок до розділу 3

У цьому розділі була продемонстрована робота додатку для виявлення пухлин головного мозку. Додаток був розроблений з використанням фреймвоку PyQt5. Результати дослідження показали, що розроблений додаток має великий потенціал у виявленні пухлин головного мозку з високою точністю. Нейронна мережа, на якій ґрунтується додаток, була навчена на великому наборі даних, що дозволило їй виявляти різні типи пухлин та розрізняти їх від нормальних тканин. Під час тестування додатку на реальних клінічних зображеннях пацієнтів було отримано високу точність виявлення пухлин. Додаток здатний швидко аналізувати зображення головного мозку та виділяти можливі пухлини, що допомагає лікарям у попередньому діагнозі та прийнятті важливих рішень щодо подальшого лікування. Застосування такого додатку може значно полегшити роботу фахівців у галузі нейрохірургії та діагностики пухлин головного мозку. Він може стати потужним інструментом у виявленні пухлин на ранніх стадіях, що дозволяє своєчасно почати лікування та покращити прогнози для пацієнтів. Результати дослідження вказують на важливість та перспективи застосування нейронних мереж у медицині, зокрема в діагностиці пухлин головного мозку. Додаток для виявлення пухлин головного мозку, розроблений на основі нейронної мережі, може стати важливим інструментом для покращення діагностики та лікування цього серйозного захворювання.

Під час виконання економічного дослідження було створено систематичний огляд теоретичних знань у галузі економіки та організації виробництва. Ці знання були використані для техніко-економічного обґрунтування розробки методом функціонально-вартісного аналізу. За допомогою цього аналізу були визначені чотири найбільш перспективні варіанти реалізації програмного продукту на основі його основних функцій. Серед цих варіантів, перший виявився найбільш ефективним, з максимальним коефіцієнтом техніко-економічного рівня та вартістю

витрат 20 692 20 692 грн. Цей варіант передбачає використання мови програмування Python, інтерфейсу користувача на базі Tensor Flow та має найвищу точність розрахунків.

ЗАГАЛЬНІ ВИСНОВОК

У ході виконання дипломної роботи на тему "Програмний додаткок для ідентифікації пухлин головного мозку з використанням згорткової нейронної мережі" були розглянуті основні теоретичні засади згорткових нейронних мереж, вивчені та досліджені методи глибинного навчання та машинного зору. Після цього було вдало підібрано архітектуру нейронної мережі, налаштовано її параметри та протестовано. У ході розробки мережі досягнуто великої точності на тестовому датасеті, близько 99%, що являється гарним показником для поставленої задачі. Останнім кроком було створення графічного інтерфейсу та вбудову у нього нейронної мережі. Програмний додаток включає такі властивості, як зчитування та обробка медичних зображень головного мозку, а також візуалізацію результатів ідентифікації та надання відповіді користувачеві. Практична реалізація поставлених завдань показала, що розроблений програмний додаток успішно виконує ідентифікацію пухлин головного мозку. Це відкриває широкі можливості для використання такого додатку в медичних дослідженнях та практичній медицині для швидкого та точного виявлення пухлин та постановки діагнозу.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Aurélien, Géron. "Hands-on machine learning with scikit-learn & tensorflow." Geron Aurelien (2017).
2. Trask, Andrew W. Grokking deep learning. Simon and Schuster, 2019
3. Синєглазов, Віктор, та Олена Чумаченко. "Глибокі Нейронні Мережі для Вирішення Завдань Розпізнавання і Класифікації Зображення." Інформаційні технології та комп'ютерне моделювання 2017: 15-20.
4. Теорія та практика нейронних мереж : навч. посіб. / Л. М. Добровська, І. А. Добровська. – К. : НТУУ «КПІ» Вид-во «Політехніка», 2015. – 396 с
5. TensorFlow: https://www.tensorflow.org/
6. PyTorch: https://pytorch.org/
7. GitHub: https://github.com/
8. Arbabshirani, M. R., Plis, S., Sui, J., & Calhoun, V. D. (2017). Single subject prediction of brain disorders in neuroimaging: Promises and pitfalls. NeuroImage, 145, 137-165.
9. Аліна Присяжнюк. "Як працює machine learning та його застосування на практиці" 2019.
10. Макеєв, СЕРГІЙ СЕРГІЙОВИЧ, Д. С. Мечев, and В. Д. Розуменко. "Однофотонна емісійна компютерна томографія у діагностиці пухлин головного мозку." *К.: Інтерсервіс* 202 (2012).
11. Jin, Jonghoon, Aysegul Dundar, and Eugenio Culurciello. "Flattened convolutional neural networks for feedforward acceleration." arXiv preprint arXiv:1412.5474 (2014).
12. Oliphant, Travis E. A guide to NumPy. Vol. 1. USA: Trelgol Publishing, 2006.
13. Python, Why. "Python." Python Releases Wind 24 (2021).
14. McKinney, Wes, and P. D. Team. "Pandas-Powerful python data analysis toolkit." Pandas—Powerful Python Data Analysis Toolkit 1625 (2015).
15. Kramer, Oliver, and Oliver Kramer. "Scikit-learn." Machine learning for evolution strategies (2016): 45-53.

16. Ketkar, Nikhil, and Nikhil Ketkar. "Introduction to keras." Deep learning with python: a hands-on introduction (2017): 97-111.
17. Willman, Joshua, and Joshua Willman. "Overview of pyqt5." Modern PyQt: Create GUI Applications for Project Management, Computer Vision, and Data Analysis (2021): 1-42.
18. Randles, Bernadette M., et al. "Using the Jupyter notebook as a tool for open science: An empirical study." 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL). IEEE, 2017.
19. Hu, Qiang, Lei Ma, and Jianjun Zhao. "DeepGraph: A PyCharm tool for visualizing and understanding deep learning models." 2018 25th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2018.