МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ЗАСТОСУВАННЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ДІАГНОСТИКИ ХВОРОБИ АЛЬЦГЕЙМЕРА

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21810106

Виконав студент 4-го курсу, групи 401
_____ М. Ю. Галущак
«20» червня 2022 р.
Керівник: к.т.н., доцент
_____Г. В. Кондратенко
«20» червня 2022 р.

Миколаїв – 2022

Чорноморський національний університет ім. Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**

Спеціальність **122 «Комп'ютерні науки»**

(шифр і назва)

Галузь знань **12 «Інформаційні технології»**

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.

_____ Ю. П. Кондратенко

«____» _____2021 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Галущаку Максиму Юрійовичу.

1. Тема кваліфікаційної роботи «Застосування згорткових нейронних мереж для діагностики хвороби Альцгеймера».

Керівник роботи Кондратенко Галина Володимирівна, к.т.н., доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: знімки магнітно-резонансної томографії з різним ступенем розвитку хвороби Альцгеймера.

Очікуваний результат: система діагностування хвороби Альцгеймера за допомогою згорткових нейронних мереж.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз сучасного стану задачі систем діагностування захворювань Альцгеймера за допомогою нейромережевих технологій;

– огляд технологій, методів, підходів, алгоритмів для вирішення задачі діагностування хвороби Альцгеймера;

– моделювання та реалізація нейронних мереж;

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «формування у майбутніх фахівців відповідних знань щодо особливостей впливу несприятливих виробничих чинників на комп'ютеризованих робочих місцях»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Макарова О.В., ст. викладач кафедри екології	

Керівник роботи __ к.т.н., доцент Кондратенко Г. В._____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Галущак М. Ю._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_26_» _____листопада____ 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Застосування згорткових нейронних мереж для діагностики хвороби Альцгеймера _____

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	28.10.2021	30.10.2021	
2	Отримання завдання на виконання БКР	24.11.2021	24.11.2021	
3	Складання календарного плану роботи на весь період виконання БКР	08.12.2021	09.12.2021	
4	Отримання завдання на переддипломну практику	20.05.2022	20.05.2022	
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	23.05.2022	04.06.2022	
6	Розробка звіту з переддипломної практики	04.06.2022	06.06.2022	
7	Виконання БКР: аналіз сучасного стану задачі відслідковування очей, огляд існуючих технологій, розробка ПЗ	28.02.2022 та 06.06.2022	27.03.2022 та 19.06.2022	
8	Попередній захист БКР на засіданні комісії кафедри	30.05.2022	31.05.2022	
9	Доробка та остаточне оформлення БКР	02.06.2022	20.06.2022	
10	Подання БКР рецензенту	16.06.2022	18.06.2022	
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	20.06.2022	22.06.2022	
12	Захист БКР перед екзаменаційною комісією (ЕК)	27.06.2022	27.06.2022	

Розробив студент __Галущак М. Ю._____ _____

(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ к.т.н., доцент. Кондратенко Г. В. ____ _____

(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 2021 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили

Галущак Максима Юрійовича

Тема: «Застосування згорткових нейронних мереж для діагностики хвороби Альцгеймера»

Об'єкт роботи – процеси діагностування хвороби Альцгеймера за допомогою згорткових нейронних мереж.

Предмет роботи – згорткова нейронна мережа для визначення хвороби Альцгеймера на різних стадіях.

Метою бакалаврської кваліфікаційної роботи є діагностування захворювань Альцгеймера на основі розроблених нейронних мереж з використанням різних моделей нейронних мереж, які здатні робити точні прогнози.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, трьох розділів, висновків та додатків.

У першому розділі розглядається аналіз сучасного стану задачі систем діагностування захворювань Альцгеймера за допомогою нейромережевих технологій.

У другому розділі проведено огляд технологій, методів, підходів, алгоритмів для вирішення задачі діагностування хвороби Альцгеймера.

У третьому розділі описано моделювання та реалізація нейронних мереж на основі моделей: AlexNet, ZFNet, LeNet.

В результаті розроблено систему діагностування захворювань Альцгеймера з використанням нейромережевих технологій з точністю близько 80%.

Бакалаврська кваліфікаційна робота містить __ сторінок, __ рисунків, __ таблиць, __ використаних джерел та __ додатків.

Ключові слова: нейроні мережі, штучний інтелект, машинне навчання, хвороби Альцгеймера, діагностування, медицина.

ABSTRACT

bachelor's degree work of a student of group 401 ChNU. Petra Mogili

Halushchak Maxim Yuriovich

Topic: "The use of convolutional neural networks for the diagnosis of Alzheimer's disease"

The object of work - the process of diagnosing Alzheimer's disease using convolutional neural networks.

The subject of the work is a convolutional neural network for determining Alzheimer's disease at different stages.

The purpose of the bachelor's thesis is to diagnose Alzheimer's disease on the basis of developed neural networks using different models of neural networks that are able to make accurate predictions.

The work consists of a professional section and a special section on labor protection. The explanatory note consists of an introduction, three sections, conclusions and appendices.

The first section considers the analysis of the current state of the problem of systems for diagnosing Alzheimer's disease using neural network technologies.

The second section reviews technologies, methods, approaches, algorithms for solving the problem of diagnosing Alzheimer's disease.

The third section describes the modeling and implementation of neural networks based on models: AlexNet, ZFNet, LeNet.

As a result, a system for diagnosing Alzheimer's disease was developed using neural network technologies with an accuracy of about 80%.

The bachelor's thesis contains __ pages, __ figures, __ tables, __ used sources and __ appendices.

Key words: neural networks, artificial intelligence, machine learning, Alzheimer 's disease, diagnosis, medicine.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 4

ВСТУП 5

1 АНАЛІЗ СУЧАСНОГО СТАНУ ЗАДАЧІ СИСТЕМ ДІАГНОСТУВАННЯ ЗАХВОРЮВАНЬ АЛЬЦГЕЙМЕРА ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ 7

1.1 Огляд предметної сфери 7

1.2 Останні дослідження та публікації 10

1.3 Постановка задачі 14

2 ТЕХНОЛОГІЇ, МЕТОДИ, ПІДХОДИ, АЛГОРИТМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ДІАГНОСТУВАННЯ ХВОРОБИ АЛЬЦГЕЙМЕРА 16

2.1 Методи для вирішення задачі 16

2.2 Технології розробки системи 31

3 МОДЕЛЮВАННЯ ТА РЕАЛІЗАЦІЯ НЕЙРОННИХ МЕРЕЖ 38

3.1 Опис датасету 38

3.2 Підготовка датасету 39

3.3 Архітектура системи на AlexNet 42

3.4 Реалізація системи на AlexNet 46

3.5 Архітектура системи на LeNet 50

3.6 Реалізація системи на LeNet 54

3.7 Архітектура системи на ZFNet 57

3.8 Реалізація системи на ZFNet 60

4 ОХОРОНА ПРАЦІ 62

ВИСНОВКИ 73

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 74

ДОДАТОК А Код для реалізації AlexNet 77

ДОДАТОК Б Код для реалізації AlexNet 82

ДОДАТОК В Код для реалізації AlexNet 86

ПЕРЕЛІК СКОРОЧЕНЬ

ANN (англ. artificial neural networks) - штучні нейронні мережі

SNN (англ. simulated neural networks) - змодельовані нейронні мережі

MRI (англ. magnetic resonance imaging) - магнітно-резонансна томографія

fMRI (англ. functional magnetic resonance imaging) - функціональна магнітно-резонансної томографії

AD (англ. Alzheimer's disease) - хвороба Альцгеймера

ML (англ. machine learning) - машинне навчання

SVM (англ. support vector machine) - машина опорного вектора

FC (англ. functional connectivity) - функціональна зв’язність

DL (англ. deep learning) - глибоке навчання

K-NN (англ. k-middle nearest neighbors) - k-середніх найближчих сусідів

RBM (англ. restricted Boltzmann machine) - обмежена машина Больцмана

CNN (англ. convolutional neural network) - згорткова нейронна мережа

SNUBH (англ. Seoul National University Bundang Hospital) - Бунданг Сеульській національний університет

ADNI (англ. Alzheimer's Disease Neuroimaging Initiative) - ініціатива нейровізуалізації хвороби Альцгеймера

ВДТ – Візуальний дисплейний термінал

ПЕОМ – Персональні електронні обчислювальні машини

КПО – Коефіцієнт природної освітленості

ПК – Персональний комп’ютер

ЕОМ – Електронна обчислювальна машина

ССБП – Система стандартів безпеки праці

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«ЗАСТОСУВАННЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ДІАГНОСТИКИ ХВОРОБИ АЛЬЦГЕЙМЕРА»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21810106

Виконав студент 4-го курсу, групи 401

_____ *Галущак М.Ю.*_____

(підпис, ініціали та прізвище)

«___» _____ 2022 р.

Керівник: *к.т.н., доцент. Кондратенко Г. В.*

(наук. ступінь, вчене звання)

_____

(підпис, ініціали та прізвище)

«___» _____ 2022 р.

Миколаїв – 2022

ВСТУП

Нейронні мережі, також відомі як штучні нейронні мережі (ANN) або змодельовані нейронні мережі (SNN), є підмножиною машинного навчання і лежать в основі алгоритмів глибокого навчання. Їх назва та структура натхненні людським мозком, імітуючи спосіб, яким біологічні нейрони сигналізують один одному. Штучні нейронні мережі складаються з шарів вузлів, що містять вхідний шар, один або кілька прихованих шарів і вихідний шар. Кожен вузол або штучний нейрон з'єднується з іншим і має відповідну вагу та поріг. Якщо вихід будь-якого окремого вузла перевищує вказане порогове значення, цей вузол активується, надсилаючи дані на наступний рівень мережі. В іншому випадку дані не передаються на наступний рівень мережі. Нейронні мережі покладаються на навчальні дані, щоб з часом вивчати та покращувати свою точність. Проте, як тільки ці алгоритми навчання будуть налаштовані на точність, вони стануть потужними інструментами в інформатиці та штучному інтелекті, що дозволить нам класифікувати та кластерувати дані з високою швидкістю. Завдання розпізнавання мовлення або зображення можуть займати хвилини та години, якщо порівнювати з ручною ідентифікацією експертів. Однією з найвідоміших нейронних мереж є пошуковий алгоритм Google [1].

Актуальність теми. Хвороба Альцгеймера є основною причиною деменції. Однак ні хвороба Альцгеймера, ні деменція не є неминучим наслідком старіння. Хоча в даний час не існує засобів або факторів, здатних анулювати патологічні зміни, але, є багато перспектив запобігти або відстрочити розвиток деменції у частини населення, змінивши вплив загальних факторів ризику. Для більшості людей раннє діагностування захворювання або ризику захворювання все є важливою частиною життя, задля того, щоб особа та її опікуни мали час зробити вибір і спланувати майбутнє, а також надати доступ до лікування, яке може допомогти впоратися з симптомами. Фахівці первинної медичної допомоги відіграють важливу роль у розпізнаванні осіб, які знаходяться в групі ризику,

рекомендуючи зміни способу життя в середині дорослого життя, які можуть запобігти або уповільнити захворювання, а також у своєчасній діагностиці. Раннє втручання є оптимальною стратегією, оскільки рівень функції пацієнта зберігається довше.

Висока точність функціонування нейромережевих діагностичних моделей, свідчить про перспективність застосування штучних нейронних мереж у різних галузях медицини для діагностики та прогнозування захворювань. Впровадження в клінічну практику нейромережевих діагностичних моделей може надати ефективну допомогу у прийнятті лікарських рішень, сприяти підвищенню якості та точності діагностики захворювань, скоротити час на обстеження пацієнта. Варто також зазначити, що штучні нейронні мережі можуть використовуватися як математичні моделі предметної області. Змінюючи вхідні параметри нейромережевої моделі, спостерігаючи за поведінкою вихідних сигналів, можна вивчати предметну область, що розглядається, виявляти і досліджувати медичні закономірності, які витягла штучна нейронна мережа при навчанні. Отримані відомості дозволять розширити теоретичні знання у різних галузях медицини.

Об'єктом розробки є процеси діагностування хвороби Альцгеймера за допомогою згорткових нейронних мереж.

Предметом роботи є згорткова нейронна мережа для визначення хвороби Альцгеймера на різних стадіях.

Мета роботи полягає у діагностування захворювання Альцгеймера на різних стадіях на основі розроблених нейронних мереж.

Для досягнення вказаної мети роботи треба виконати основні завдання:

– проаналізувати сучасний стан задачі діагностування захворювань Альцгеймера;

– окреслити існуючі технології для вирішення поставленої задачі;

– реалізація штучної нейронної мережі та її налаштування для діагностування захворювань Альцгеймера з використанням фреймворків;

– навчання нейронної мережі, використовуючи дані пацієнтів.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ЗАДАЧІ СИСТЕМ ДІАГНОСТУВАННЯ ЗАХВОРЮВАНЬ АЛЬЦГЕЙМЕРА ЗА ДОПОМОГОЮ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

1.1 Огляд предметної сфери

На сьогоднішній день аналіз даних нейровізуалізації, таких як ті, що були отримані за допомогою магнітно-резонансної томографії (MRI), позитронно-емісійної томографії, функціональної МРТ (fMRI) та дифузійного тензорного зображення, в основному виконувався експертами, такими як радіологи та лікарі, що вимагає високий ступінь спеціалізації. Хвороба Альцгеймера (AD), що характеризується прогресуючим порушенням когнітивних функцій і функцій пам’яті, є найпоширенішим типом деменції, який часто з’являється у осіб старше 65 років. Для уповільнення прогресування деменції вирішальне значення має своєчасне лікування, яке вимагає ранньої діагностики AD та її продромальної стадії, легкого когнітивного порушення.

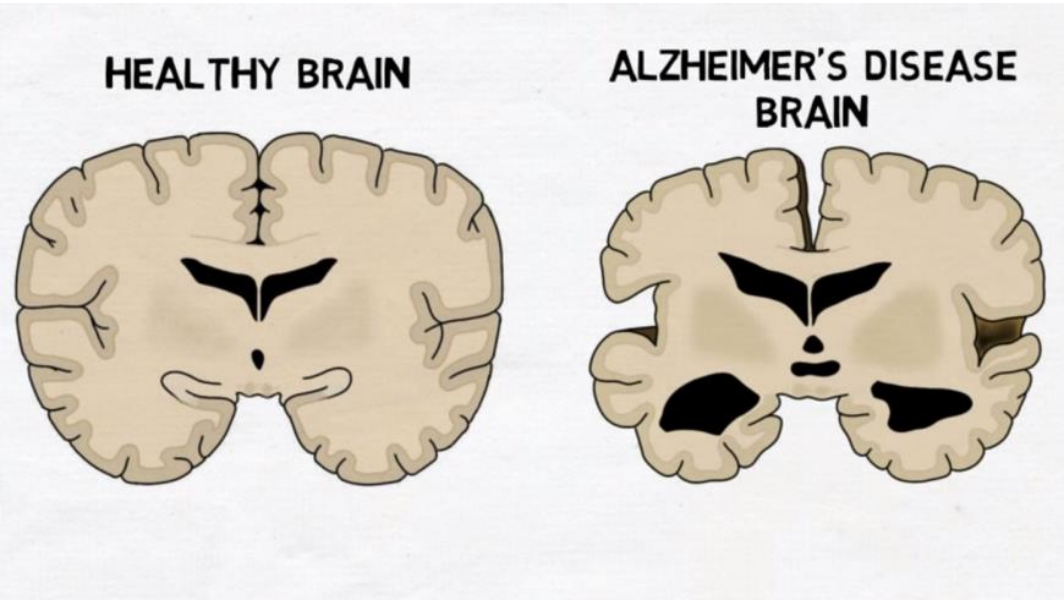

Рисунок 1.1 - Візуалізація хвороби Альцгеймера.

З цією метою необхідна надійна діагностика на основі візуалізації мозку, а надійна діагностична система за допомогою аналізу даних нейровізуалізації дозволила б отримати більш інформативний та надійний підхід, а також потенційно підвищити точність діагностики. Традиційні аналітичні методи дослідження біомаркерів нейровізуалізації для аналізу нервово-психічних розладів були засновані на масовій однофакторній статистиці з припущенням, що різні ділянки мозку діють незалежно. Однак це припущення не підходить з огляду на наше сучасне розуміння функціонування мозку.

Останнім часом методи машинного навчання (ML), які можуть враховувати взаємозв'язок між регіонами, стали привабливим і фундаментальним елементом комп'ютерних аналітичних методів і широко використовуються для автоматизованої діагностики та аналізу нервово-психічних розладів. Незважаючи на те, що різні моделі машинного навчання використовувалися для автоматизованого прогнозування неврологічних розладів, два основних напрямки дослідження включають моделі діагностики на основі машини опорного вектора (SVM) і глибокого навчання (DL). У зв'язку з цим були опубліковані великі огляди, пов'язані з медичною візуалізації з використанням методів машинного навчання. Автоматизовані діагностичні моделі нервово-психічних розладів, засновані на SVM, як правило, використовують функції, створені вручну через їх нездатність виділити адаптивні особливості. Патерни функціональної зв'язності у (FC), що представляють кореляції областей мозку, є популярною особливістю існуючих моделей діагностики на основі SVM. Окремі моделі FC виділяються для пар сегментованих ділянок мозку, визначених за допомогою автоматизованого анатомічного маркування. Незважаючи на його популярність, SVM критикували за низьку продуктивність на необроблених даних і за потребу експертного використання методів проектування для отримання інформаційних функцій.

Навпаки, моделі DL дозволяють системі використовувати вихідні дані як вхідні дані, що дозволяє їм автоматично виявляти дуже дискримінаційні ознаки в даному наборі навчальних даних. Ця філософія проектування наскрізного навчання

є фундаментальною основою DL. Основна перевага наскрізного навчання полягає в тому, що всі етапи конвеєра обробки одночасно оптимізуються, що потенційно може призвести до оптимальної продуктивності.

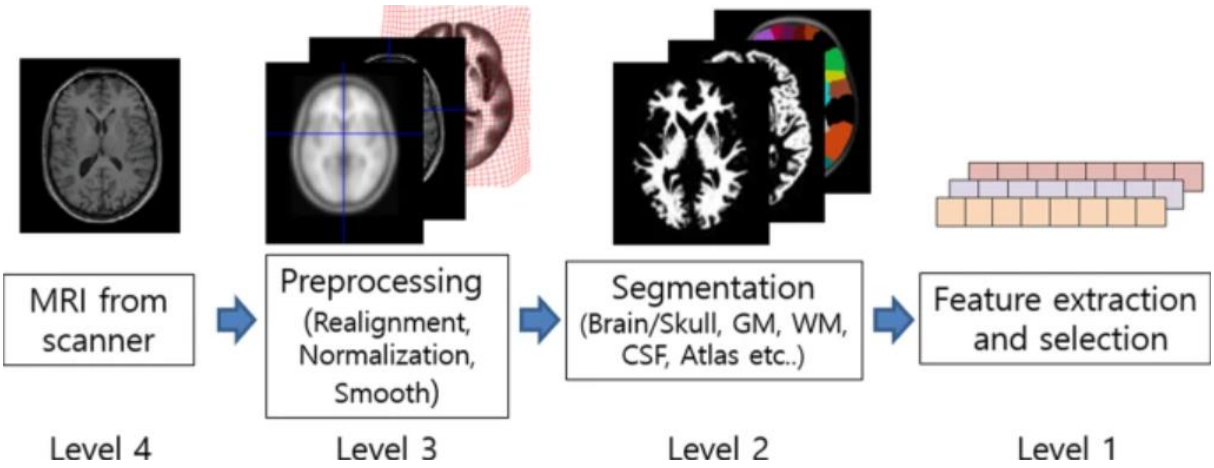

Рисунок 1.2 - Поняття наскрізних рівнів навчання

Рівні ієрархії варіюються від 1 (немає) до 4 (повний). Більшість існуючих досліджень використовує рівень 1 або рівень 2, продуктивність яких сильно залежить від конкретного програмного забезпечення, а іноді навіть від налаштування гіперпараметрів і ручного видалення шуму. Через ці залежності оцінка продуктивності в цих дослідженнях використовувала лише підмножину вихідних наборів даних, виключаючи видимі викиди та ускладнюючи справедливе порівняння продуктивності. Ще одна перевага наскрізного навчання полягає в тому, що можливе ефективне візуальне пояснення того, чому CNN прийняла рішення щодо класифікації. Пояснення допомагає лікарю зрозуміти поведінку CNN і виявити нові біомаркери. На рівні 2 пояснення обмежується сегментованою частиною, що може блокувати можливість виявлення нових біомаркерів, розташованих у вилученій області. На рівні 1 пояснення неможливе або ненадійне, оскільки зворотне відображення в тривимірному просторі повністю порушено на вхідному рівні.

Ранні підходи на основі DL працюють на рівні 1 і далекі від автоматичного вилучення функцій. Наприклад, тривимірні (3D) або 4D дані об'єму перегруповуються в одновимірну векторну форму для використання вхідних даних мережі DL, таких як обмежена машина Больцмана (RBM) і мережа глибоких переконань (DBN). Залежність від ручної роботи можна віднести до дефіциту даних і високої розмірності, які є ендемічними характеристиками медичних даних. Наприклад, набір даних ADNI містить лише кілька сотень зображень, тоді як кожне зображення має понад 11 мільйонів розмірів ($256 \times 256 \times 170$ вокселів). Важливо відзначити, що вищезгадані методи спотворюють сусідні відносини (просторову локальність) у даних візуалізації мозку на етапі виділення ознак. Без збереження просторових відносин важко очікувати надійного пояснення того, як мережа досягає рішення про класифікацію.

1.2 Останні дослідження та публікації

Системи аналізу зображень, за допомогою методів машинного навчання (ML), швидко розвиваються в останні роки. Методи машинного навчання включають у себе навчання дерева рішень, кластерізацію, методи опорних векторів (SVM), k-середніх найближчих сусідів (K-NN), обмежені машини Больцмана (RBM) та випадкові ліси. Фактором для ефективної роботи методів ML є вилучення дискримінантних визнань. Ці функції, як правило, невідомі, а також є дуже складною задачею, особливо для доданих, пов'язаних із зображеннями, і все ще є предметом досліджень. Логічним кроком для подолання було створення інтелектуальних машин, які могли б використовувати функції, необхідні для аналізу зображень та витягувати їх самостійно. Однією із таких інтелектуальних моделей є згорткова нейронна мережа (CNN), яка автоматично вивчає необхідні функції та витягує їх для аналізу медичних зображень.

Модель CNN складається зі зворотних фільтрів, основною функцією яких є вивчення необхідних характеристик для ефективного спостереження медичних зображень. CNN почав набирати популярність у 2012 році завдяки AlexNet, модель

CNN, яка здобула всі інші моделі з рекордною точністю та низькими рівнями помилок у виклику imageNet 2012.

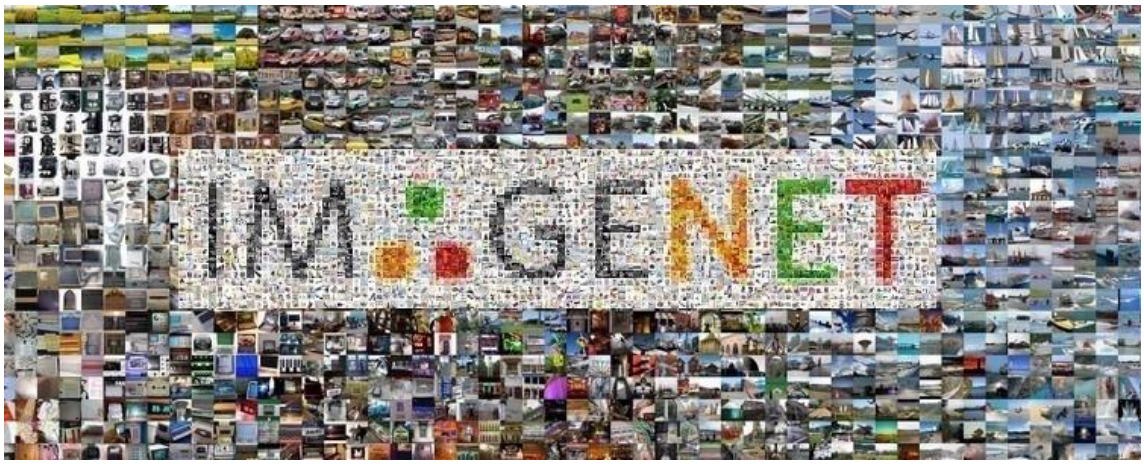

Рисунок 1.3 - Лого ImageNet

CNN використовувала корпоративні гіганти для надання інтернет-послуг, автоматичної помітки зображень, рекомендацій щодо продуктів. Основними функціями CNN є обробка зображень і сигналів, та аналіз даних. CNN мала великий прорив, коли GoogleNet використав його для виявлення захворювання на рак з точністю 89%, у той час як патологічної анатомії змогла отримати точність лише 70%.

CNN також домінували в області виявлення COVID-19 за допомогою рентгенографії грудної клітки/комп'ютерної томографії. Дослідження з використанням CNN у цей час є домінуючою темою великих конференціях. Крім того, у відомих журналах зарезервовані спеціальні випуски для вирішення завдань із використанням моделей глибокого навчання. Величезна кількість доступної літератури по CNN свідчить про їх ефективність та широке використання. Тим не менш, різні дослідницькі спільноти одночасно розробляють ці додатки, і результати розповсюдження розкидані по широкому та різноманітному колу матеріалів конференцій та журналів.

У цьому дослідженні[2] группа вчених під керівництвом Чон Бин Бэ розробили алгоритм класифікації бронхіальної астми на основі згорткової нейронної мережі (CNN) з використанням магнітно-резонансної томографії пацієнтів з бронхіальної астми і відповідних за віком/статтю когнітивно нормальних контролів з двох груп населення, що відрізняються за етнічною належністю та рівнем освіти. Ці групи приходять з лікарні Бунданг Сеульського національного університету (SNUBH) та ініціативи нейровізуалізації хвороби Альцгеймера (ADNI). Для кожної популяції вони навчили CNN на п'яти підмножинах, використовуючи корональні зрізи T1 зважених зображень, які охоплюють медіальну скроневу частку. Вони оцінили моделі на підмножинах перевірки як з однієї й тієї самої сукупності (перевірка всередині набору даних), так і з іншої сукупності (перевірка між наборами даних). Їх моделі досягли середніх площ під кривими 0,91-0,94 для перевірки всередині набору даних та 0,88-0,89 для перевірки між наборами даних. Середній час обробки на людину становив 23-24 с. Характеристики всередині набору даних та між наборами даних були співставні між моделями, отриманими з ADNI, та моделями, отриманими з SNUBH. Ці результати демонструють узагальнення наших моделей для різних пацієнтів з різною етнічною приналежністю та рівнем освіти, а також їх потенціал для розгортання як швидкі та точні інструменти діагностичної підтримки при AD. 94 для перевірки всередині набору даних та 0,88–0,89 для перевірки між наборами даних. Середній час обробки на людину становив 23-24 с. Характеристики всередині набору даних та між наборами даних були співставні між моделями, отриманими з ADNI, та моделями, отриманими з SNUBH. Ці результати демонструють узагальнення їх моделей для різних пацієнтів з різною етнічною приналежністю та рівнем освіти, а також їх потенціал для розгортання як швидкі та точні інструменти діагностичної підтримки при AD. 94 для перевірки всередині набору даних та 0,88–0,89 для перевірки між наборами даних. Середній час обробки на людину становив 23-24 с. Характеристики всередині набору даних та між наборами даних були співставні між моделями, отриманими з ADNI, та моделями,

отриманими з SNUBH. Ці результати демонструють узагальнення їх моделей для різних пацієнтів з різною етнічною приналежністю та рівнем освіти, а також їх потенціал для розгортання як швидкі та точні інструменти діагностичної підтримки при AD.

В іншому дослідженні[3] команда під керівництвом Гильерме Фолего покладалися на 3D CNN з даними, в першу чергу наданими ADNI, і оцінювалися на виклику CADDementia[4]. Їх рішення також включає механізм підзвітності, який дозволяє розуміти його рішення. Наші експерименти проводилися у сценарії, схожому на реальні умови, в яких система САПР використовується з набором даних, відмінним від того, що використовувався для навчання.

Основні завдання та внесок дослідження полягали у розробці рішення для глибокого навчання, повністю автоматичного та порівняно швидкого, а також представлення конкурентоспроможних результатів без використання будь-яких знань у предметній галузі. Їх метод, названий ADNet, дає значний виграш у точності, перевершуючи кілька інших систем попереднього рівня техніки, всі з яких вимагають попередніх знань про захворювання, таких як певні сфери інтересу з вхідних зображень. Крім того, ця система не вимагає ручного втручання, клінічної інформації або вибраних ділянок мозку. Основна причина відмови від використання будь-якої інформації про хворобу полягає в тому, щоб дати системі можливість автоматично вивчати та вилучати відповідні патерни з областей мозку та, зрештою, дозволити їй підтримувати поточні стандарти діагностики відомих чи нових захворювань. Крім того, він працює в середньому у 80 разів швидше, ніж сучасний.

Створені моделі ADNet та ADNet-DA, а також допоміжний код знаходяться у відкритому доступі. для використання або навчання нових даних. За допомогою цієї роботи вони випустили одну з перших моделей, готових до використання, заохочуючи відкриту науку та відтворювані дослідження, а також встановлюючи відправну точку для дослідників, що працюють із 3D-MRI.

У дослідженні[5] групи дослідників під керівництвом Моніки Сетхі реалізовано та порівнюється декілька глибоких моделей та конфігурацій, включаючи двовимірні (2D) та тривимірні (3D) CNN та рекурентні нейронні мережі (RNN). Щоб використовувати 2D CNN для 3D-об'ємів MRI, кожне сканування MRI розбивається на 2D-зрізи, нехтуючи зв'язком між фрагментами 2D-зображення в об'ємі MRI. Натомість за моделлю CNN може слідувати RNN таким чином, щоб модель 2D CNN + RNN могла зрозуміти зв'язок між послідовністю фрагментів 2D зображення для MRI. Проблема в тому, що етап вилучення ознак у 2D CNN не залежить від класифікації в RNN. Щоб вирішити це, 3D CNN можна використовувати замість 2D CNN для прийняття рішень на основі вокселів. Основний внесок дослідження полягає в тому, щоб запровадити передачу навчання з набору даних 2D-зображень до 3D CNN. Результати: результати нашого набору даних MRI показують, що рішення на основі послідовності покращують точність рішень на основі зрізів на 2% при класифікації пацієнтів з AD від здорових осіб. Також метод на основі 3D-вокселів із трансферним навчанням перевершує інші методи з точністю 96,88%, чутливістю 100% і специфічністю 94,12%. Кілька реалізацій та експериментів із використанням CNN на MRI-скануваннях для виявлення AD продемонстрували, що метод на основі вокселів із перенесенням навчання з ImageNet у набори даних МРТ із використанням 3D CNN значно покращив результати порівняно з іншими.

1.3 Постановка задачі

Тому згідно з актуальністю теми медицини та штучного інтелекту об'єктом дослідження обрано процес діагностування хвороби Альцгеймера за допомогою згорткових нейронних мереж. Згідно з цим предметом дослідження є автоматизація процесу визначення хвороби Альцгеймера на різних стадіях.

Метою бакалаврської кваліфікаційної роботи є діагностування захворювань Альцгеймера на основі розроблених нейронних мереж з використанням різних моделей нейронних мереж, які здатні робити точні прогнози.

Для досягнення вказаної мети роботи треба виконати основні завдання:

– проаналізувати сучасний стан задачі діагностування захворювань Альцгеймера;

– окреслити існуючі технології для вирішення поставленої задачі;

– реалізація штучної нейронної мережі та її налаштування для діагностування захворювань Альцгеймера з використанням фреймворків;

– навчання нейронної мережі, використовуючи дані пацієнтів.

2 ТЕХНОЛОГІЇ, МЕТОДИ, ПІДХОДИ, АЛГОРИТМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ДІАГНОСТУВАННЯ ХВОРОБИ АЛЬЦГЕЙМЕРА

2.1 Методи для вирішення задачі

CNN — це тип моделі глибокого навчання для обробки даних, що мають сітку, наприклад, зображення, яка надихається організацією зорової кори тварин і призначена для автоматичного та адаптивного вивчення просторової ієрархії функцій, від низьких - до візерунків високого рівня. CNN — це математична конструкція, яка зазвичай складається з трьох типів шарів (або будівельних блоків): шарів згортки, об’єднання та повністю пов’язаних шарів. Перші два, шари згортки та об’єднання, виконують вилучення ознак, тоді як третій, повністю пов’язаний шар, відображає вилучені об’єкти в кінцевий результат, наприклад класифікацію. Рівень згортки відіграє ключову роль у CNN, який складається з набору математичних операцій, таких як згортка, спеціалізований тип лінійної операції.

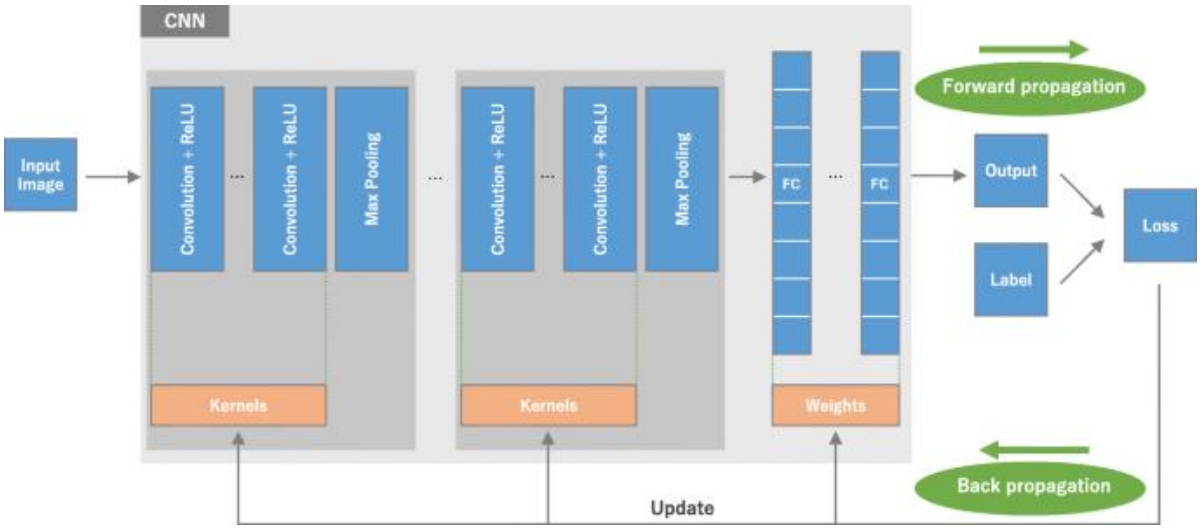

Рисунок 2.1 - Архітектура CNN

У цифрових зображеннях значення пікселів зберігаються в двовимірній (2D) сітці, тобто в масиві чисел, а невелика сітка параметрів, що називається ядром,

оптимізований екстрактор функцій, застосовується до кожної позиції зображення. , що робить CNN високоефективними для обробки зображень, оскільки функція може зустрічатися в будь-якому місці зображення. Оскільки один шар передає свій вихід на наступний шар, витягнуті об’єкти можуть ієрархічно та поступово ставати все більш складними. Процес оптимізації таких параметрів, як ядра, називається навчанням, яке виконується таким чином, щоб мінімізувати різницю між виходами та основними мітками істини за допомогою алгоритму оптимізації, який називається зворотним поширенням і градієнтним спуском, серед інших.

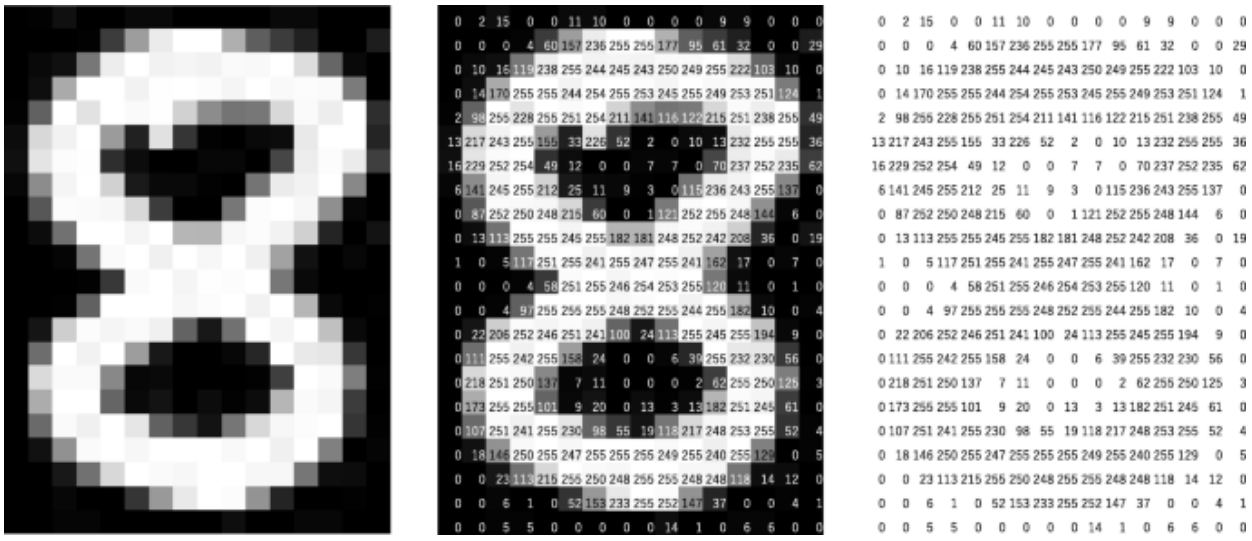

Рисунок 2.2 - Процес надання вагів

У останніх дослідженнях використовуються створені вручну методи виділення ознак, такі як аналіз текстури, а потім звичайні класифікатори машинного навчання, такі як випадкові ліси та машини опорних векторів. Між такими методами та CNN є кілька відмінностей. По-перше, CNN не вимагає ручного вилучення функцій. По-друге, архітектури CNN не обов’язково вимагають сегментації пухлин або органів експертами-людями. По-третє, CNN набагато більше потребує даних через мільйони параметрів, які можна вивчати для оцінки, і, таким чином, є більш дорогим з точки зору обчислень, що призводить до необхідності графічних процесорів (GPU) для навчання моделі.

Архітектура CNN[6] включає в себе кілька будівельних блоків, таких як шари згортки, шари об’єднання та повністю пов’язані шари. Типова архітектура складається з повторень стека з кількох шарів згортки та шару об’єднання, за яким слідують один або кілька повністю пов’язаних шарів. Крок, на якому вхідні дані перетворюються у вихідні через ці шари, називається прямим поширенням.

Рівень згортки є основним компонентом архітектури CNN, який виконує вилучення ознак, яке зазвичай складається з комбінації лінійних і нелінійних операцій, тобто операції згортки та функції активації.

Згортка — це спеціалізований тип лінійної операції, що використовується для вилучення ознак, де невеликий масив чисел, який називається ядром, застосовується до вхідних даних, який є масивом чисел, який називається тензором. Поелементний добуток між кожним елементом ядра та вхідним тензором обчислюється в кожному місці тензора і підсумовується, щоб отримати вихідне значення у відповідній позиції вихідного тензора, яке називається картою ознак.

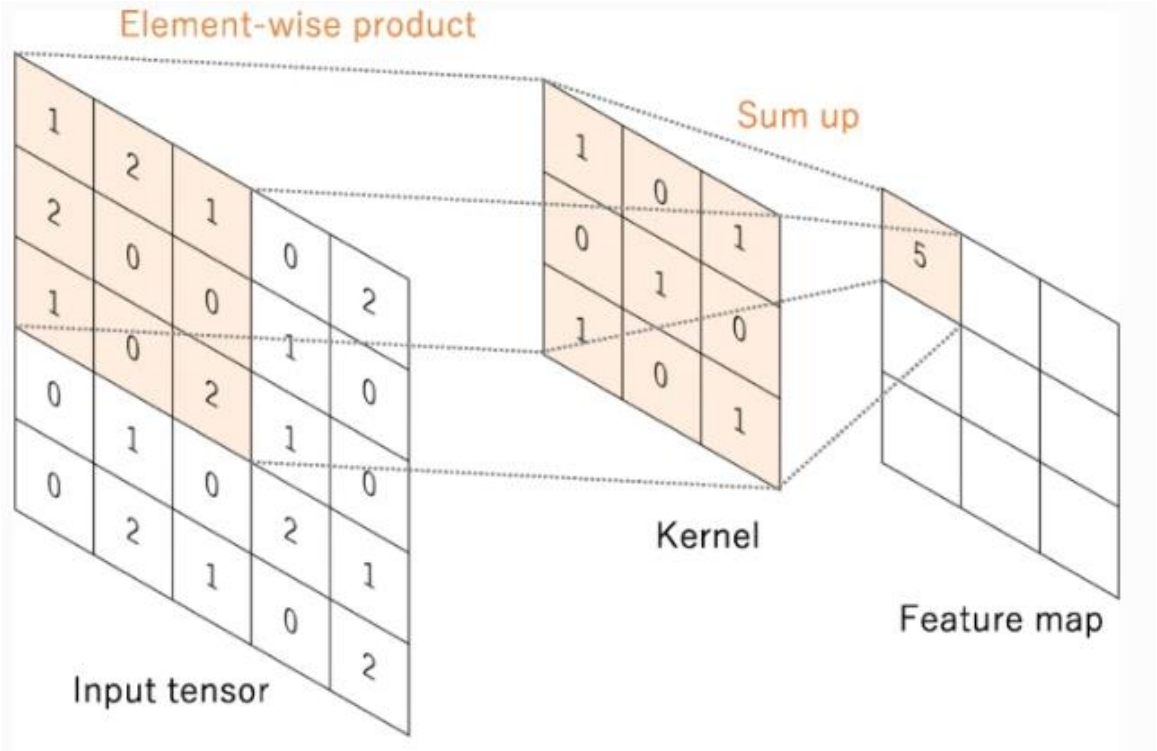

Рисунок 2.3 - Карта ознак

Ця процедура повторюється із застосуванням кількох ядер для формування довільної кількості карт ознак, які представляють різні характеристики вхідних тензорів; Таким чином, різні ядра можна розглядати як екстрактори різних ознак. Двома ключовими гіперпараметрами, які визначають операцію згортки, є розмір і кількість ядер. Перший зазвичай становить 3 × 3, але іноді 5 × 5 або 7 × 7. Останній є довільним і визначає глибину вихідних карт об’єктів.

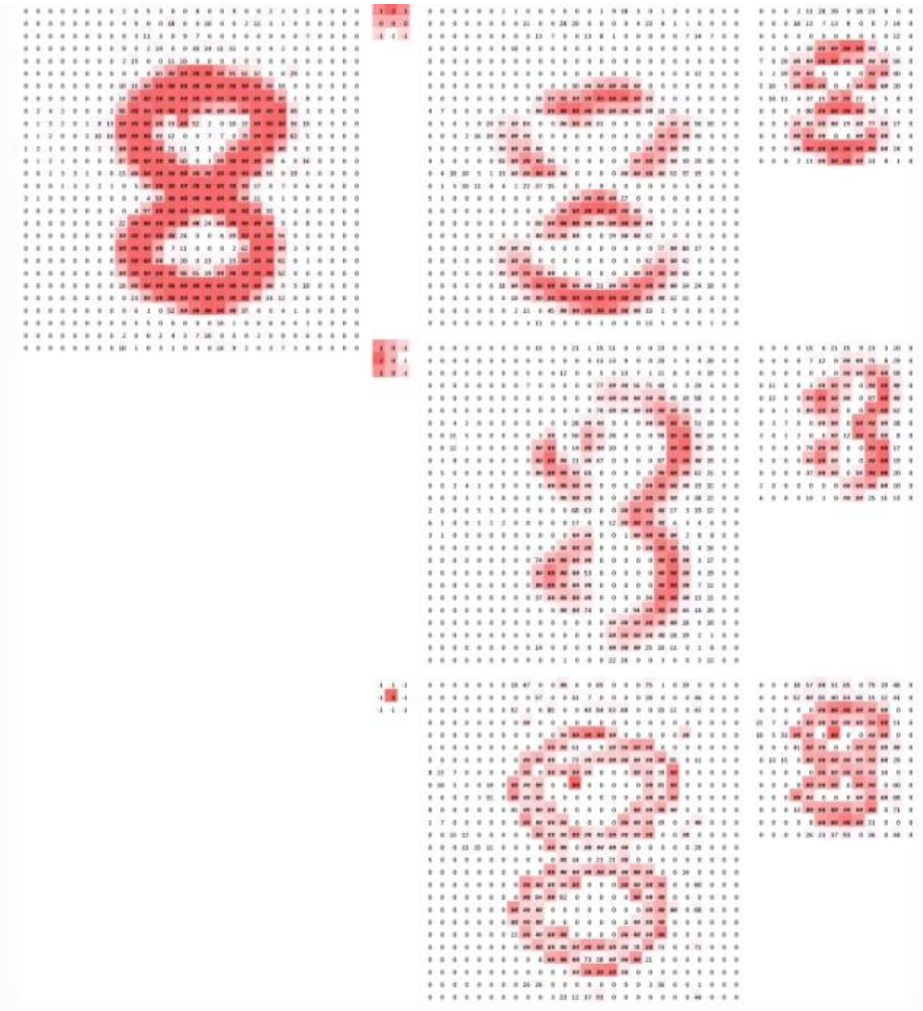

Рисунок 2.4 - Операція згортки

Операція згортки, описана вище, не дозволяє центру кожного ядра перекривати крайній елемент вхідного тензора і зменшує висоту та ширину вихідної карти об’єктів порівняно з вхідним тензором. Заповнення, як правило, нульове заповнення, є методом вирішення цієї проблеми, коли рядки та стовпці

нулів додаються з кожної сторони вхідного тензора, щоб умістити центр ядра на крайньому зовнішньому елементі та зберегти ту саму площину. розмір через операцію згортки. Сучасні архітектури CNN зазвичай використовують нульове заповнення, щоб зберегти розміри в площині, щоб застосувати більше шарів. Без нульового заповнення кожна наступна карта об’єктів буде зменшуватися після операції згортки.

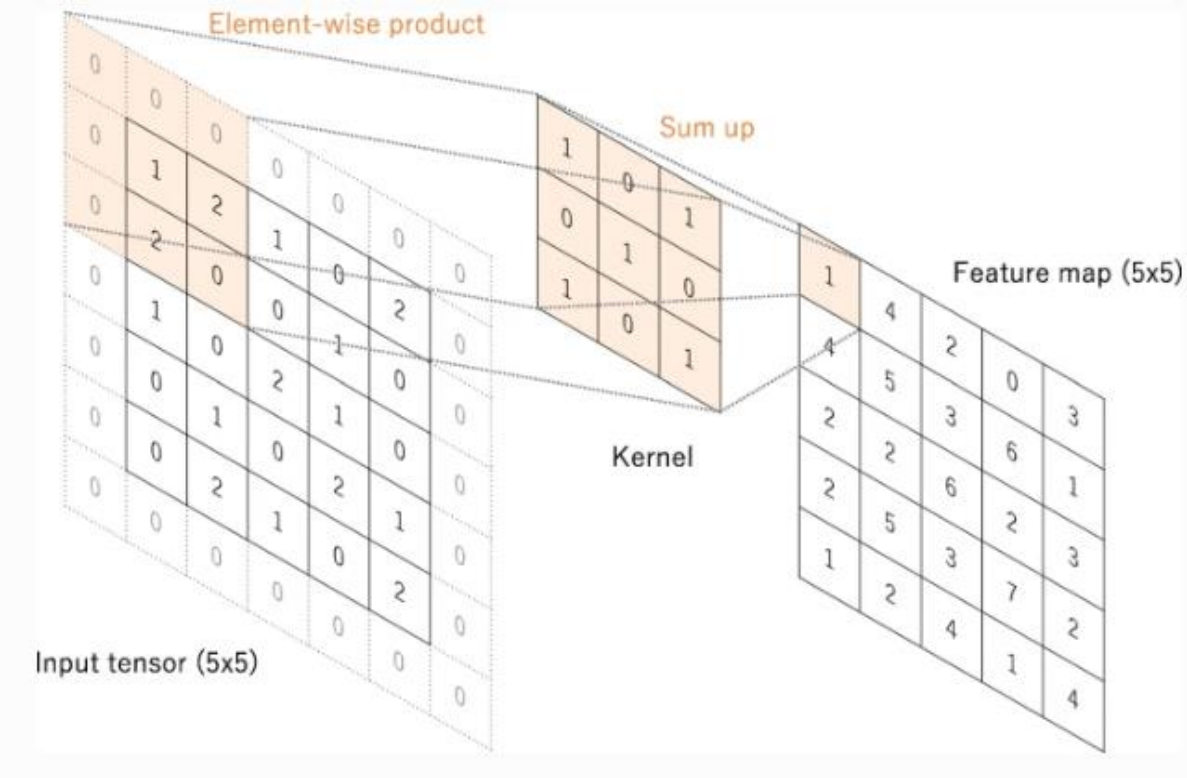

Рисунок 2.5 - Операція згортки

Відстань між двома послідовними положеннями ядра називається кроком, який також визначає операцію згортки. Загальний вибір кроку – 1; однак крок, більший за 1, іноді використовується для досягнення зниження дискретизації карт ознак. Альтернативною технікою виконання пониження дискретизації є операція об'єднання, як описано нижче.

Ключовою особливістю операції згортки є розподіл ваги: ядра використовуються спільно для всіх позицій зображення. Розподіл ваги створює такі

характеристики операцій згортки: дозволяючи шаблонам локальних ознак, витягнутим за допомогою трансляції ядра b, бути інваріантними, оскільки ядра переміщуються по всіх позиціях зображення та виявляють вивчені локальні шаблони, вивчення просторової ієрархії шаблонів ознак шляхом зменшення дискретизації в у поєднанні з операцією об'єднання, що призводить до захоплення все більшого поля зору та підвищення ефективності моделі за рахунок зменшення кількості параметрів для вивчення в порівнянні з повністю підключеними нейронними мережами.

Як описано пізніше, процес навчання моделі CNN щодо шару згортки полягає у визначенні ядер, які найкраще працюють для даного завдання на основі заданого набору навчальних даних. Ядра є єдиними параметрами, які автоматично вивчаються під час процесу навчання в шарі згортки; з іншого боку, розмір ядер, кількість ядер, відступи та кроки є гіперпараметрами, які необхідно встановити до початку процесу навчання.

Вихідні дані лінійної операції, наприклад згортки, потім передаються через нелінійну функцію активації. Хоча гладкі нелінійні функції, такі як сигмовидна або гіперболічна тангенс (tanh), використовувалися раніше, оскільки вони є математичним уявленням поведінки біологічного нейрона, найпоширенішою нелінійною функцією активації, яка використовується зараз, є випрямлена лінійна одиниця (ReLU), яка просто обчислює функція: $f(x) = \max(0, x)$.

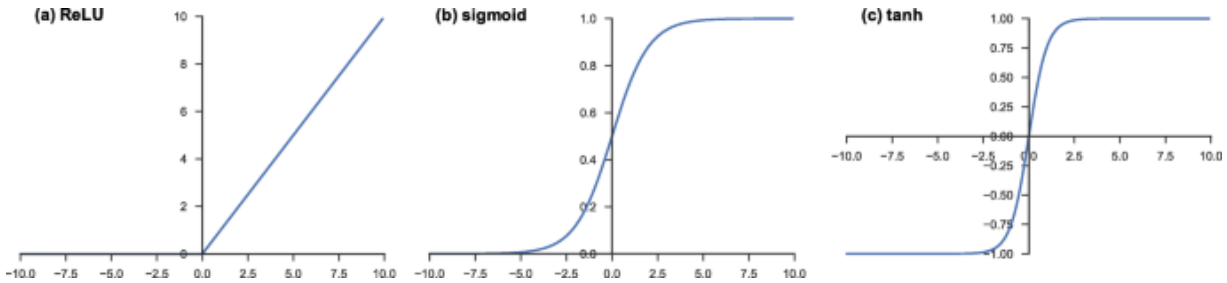

Рисунок 2.6 - Графіки функцій активізацій

Рівень об’єднання забезпечує типову операцію зниження дискретизації, яка зменшує розмірність у площині карт ознак, щоб ввести інваріантність трансляції до невеликих зсувів і спотворень, а також зменшити кількість наступних параметрів, які можна вивчати. Слід зазначити, що в жодному з шарів об’єднання немає параметрів, які можна вивчати, тоді як розмір фільтра, крок і заповнення є гіперпараметрами в операціях об’єднання, подібними до операцій згортки.

Найпопулярнішою формою операції об’єднання є максимальний пул, який витягує латки з вхідних карт об’єктів, виводить максимальне значення в кожному патчі та відкидає всі інші значення (рис. 6). На практиці зазвичай використовується максимальне об’єднання з фільтром розміром 2×2 з кроком 2. Це зменшує розмірність у площині карт об’єктів у 2 рази. На відміну від висоти та ширини, розмір глибини карт об’єктів залишається незмінним.

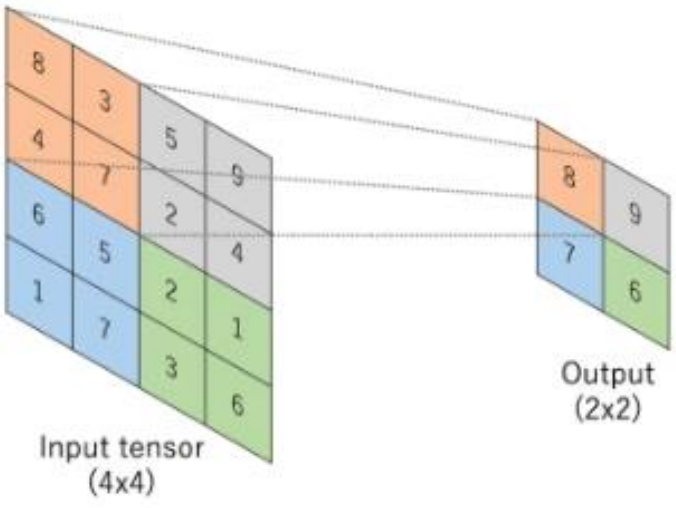

Рисунок 2.7 - Операція об’єднання

Ще одна операція об’єднання, на яку варто звернути увагу, – це глобальне середнє об’єднання. Глобальне середнє об’єднання виконує екстремальний тип зниження дискретизації, коли карта об’єктів розміром висота $\times$ ширина зменшується до масиву 1×1, просто взявши середнє значення всіх елементів у кожній карті об’єктів, тоді як глибина карт об’єктів дорівнює зберігається. Ця

операція зазвичай застосовується лише один раз перед повністю з'єднаними шарами. Переваги застосування глобального середнього пулу полягають у наступному: зменшує кількість параметрів, які можна вивчати, і дає змогу CNN приймати вхідні дані змінного розміру.

Вихідні карти об'єктів остаточного шару згортки або об'єднання зазвичай згладжуються, тобто перетворюються в одновимірний (1D) масив чисел (або вектор) і з'єднуються з одним або кількома повністю пов'язаними шарами, також відомими як щільні шари, в якому кожен вхід пов'язаний з кожним виходом за допомогою ваги, яку можна вивчати. Після того, як об'єкти, виділені шарами згортки та зменшені шарами об'єднання, створені, вони відображаються підмножиною повністю пов'язаних шарів на кінцеві вихідні дані мережі, наприклад, ймовірності для кожного класу в завданнях класифікації. Останній повністю підключений шар зазвичай має таку ж кількість вихідних вузлів, як і кількість класів. За кожним повністю підключеним шаром слідує нелінійна функція, така як ReLU, як описано вище.

Функція активації[7], застосована до останнього повністю підключеного шару, зазвичай відрізняється від інших. Відповідно до кожного завдання необхідно вибрати відповідну функцію активації. Функція активації, застосована до завдання мультикласової класифікації, є функцією softmax, яка нормалізує вихідні реальні значення з останнього повністю підключеного шару до ймовірностей цільового класу, де кожне значення знаходиться в діапазоні від 0 до 1, а сума всіх значень дорівнює 1.

Навчання мережі[8] – це процес пошуку ядер у шарах згортки та ваг у повністю зв'язаних шарах, що мінімізує відмінності між вихідними передбаченнями та заданими основними мітками істинності в наборі навчальних даних. Алгоритм зворотного поширення — це метод, який зазвичай використовується для навчання нейронних мереж, де важливу роль відіграють функція втрат і алгоритм оптимізації градієнтного спуску. Продуктивність моделі для певних ядер і ваг розраховується функцією втрат шляхом прямого поширення

на наборі даних для навчання, а параметри, які можна вивчати, а саме ядра та ваги, оновлюються відповідно до значення втрат за допомогою алгоритму оптимізації, який називається зворотним поширенням і градієнтним спуском, серед інших.

Функція втрат[9], яку також називають функцією вартості, вимірює сумісність між вихідними передбаченнями мережі через пряме поширення та заданими наземними мітками істинності. Зазвичай функцією втрат для багатокласової класифікації є перехресна ентропія, тоді як середня квадратична помилка зазвичай застосовується до регресії до безперервних значень. Тип функції втрат є одним із гіперпараметрів і потребує визначення відповідно до поставлених завдань.

Градієнтний спуск зазвичай використовується як алгоритм оптимізації, який ітеративно оновлює доступні для навчання параметри, тобто ядра та ваги, мережі, щоб мінімізувати втрати. Градієнт функції втрат дає нам напрямок, у якому функція має найбільшу швидкість зростання, і кожен параметр, який можна вивчати, оновлюється в негативному напрямку градієнта з довільним розміром кроку, визначеним на основі гіперпараметра, який називається швидкістю навчання. Математично градієнт є частковою похідною від втрати по відношенню до кожного параметра, що вивчається, і одноразове оновлення параметра формулюється таким чином:

$$w := w - \alpha * \frac{\partial L}{\partial w}$$

Рисунок 2.8 - Функція градієнта

де w означає кожен параметр, який можна вивчати, α означає швидкість навчання, а L означає функцію втрат. Слід зазначити, що на практиці швидкість навчання є одним із найважливіших гіперпараметрів, які необхідно встановити перед початком навчання. На практиці, з таких причин, як обмеження пам’яті,

градієнти функції втрат щодо параметрів обчислюються за допомогою підмножини навчального набору даних, що називається міні-пакетом, і застосовуються до оновлення параметрів. Цей метод називається градієнтним спуском міні-партії, який також часто називають стохастичним градієнтним спуском (SGD), а розмір міні-партії також є гіперпараметром. Крім того, було запропоновано і широко використовується багато вдосконалень алгоритму градієнтного спуску, таких як SGD з імпульсом, RMSprop і Adam.

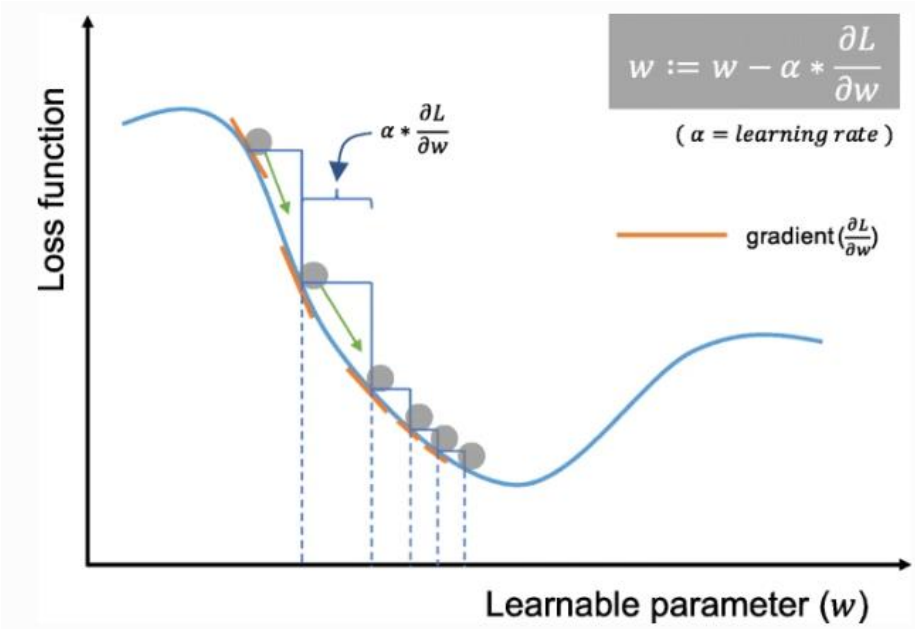

Рисунок 2.9 - Графік функції градієнта

Дані та основні позначки істини є найважливішими компонентами в дослідженнях із застосуванням глибокого навчання або інших методів машинного навчання. Як зазначає відоме прислів'я, що виникло з інформатики: «Сміття входить, сміття виходить». Для успішного проекту глибокого навчання обов'язковим є ретельний збір даних і базових міток істини, за допомогою яких можна тренувати й тестувати модель, але отримання високоякісних позначених даних може бути дорогим і тривалим. Незважаючи на те, що може бути багато наборів даних медичних зображень, відкритих для громадськості, особливу увагу в цих випадках слід приділяти якості основних позначок істини.

Доступні дані зазвичай поділяються на три набори: навчальний, перевірочний і тестовий набір, хоча є деякі варіанти, наприклад перехресна перевірка. Для навчання мережі використовується навчальний набір, де значення втрат обчислюються за допомогою прямого поширення, а параметри, які можна вивчати, оновлюються за допомогою зворотного поширення. Набір перевірки використовується для оцінки моделі під час процесу навчання, точного налаштування гіперпараметрів і виконання вибору моделі. Тестовий набір в ідеалі використовується лише один раз в самому кінці проекту, щоб оцінити продуктивність остаточної моделі, яка була точно налаштована та обрана в процесі навчання з наборами навчання та перевірки.

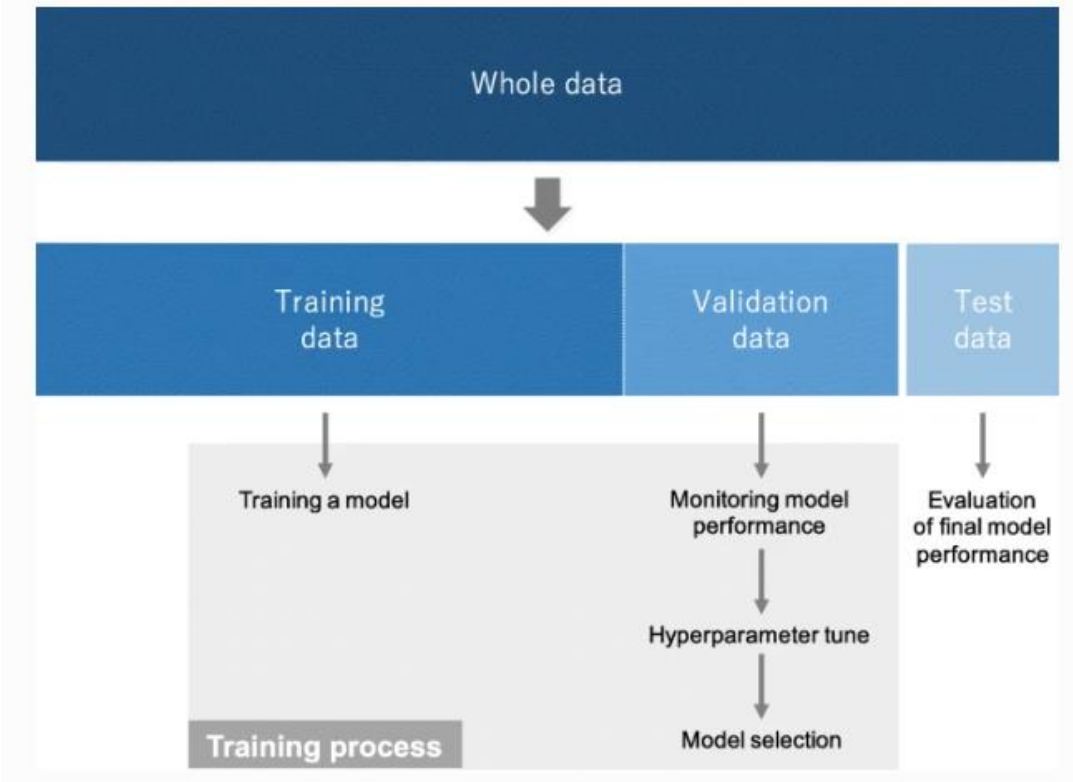

Рисунок 2.10 - Розподілення даних

Потрібні окремі набори перевірки та тестування, оскільки навчання моделі завжди передбачає тонке налаштування її гіперпараметрів і виконання вибору моделі. Оскільки цей процес виконується на основі продуктивності набору

перевірки, деяка інформація про цей набір перевірки просочується в саму модель, тобто переналаштовується на набір перевірки, навіть якщо модель ніколи не навчається безпосередньо на ньому для параметрів, які можна вивчати. З цієї причини гарантується, що модель із точно налаштованими гіперпараметрами у наборі перевірки буде добре працювати на цьому самому наборі перевірки. Таким чином, для відповідної оцінки продуктивності моделі необхідний абсолютно невидимий набір даних, тобто окремий набір тестів, оскільки ми піклуємося про продуктивність моделі на ніколи не бачених даних, тобто про узагальненість.

Варто згадати, що термін «валідація»[10] по-різному використовується в медицині та сфері машинного навчання. Як описано вище, у машинному навчанні термін «перевірка» зазвичай відноситься до кроку для точного налаштування та вибору моделей під час процесу навчання. З іншого боку, в медицині «валідація» зазвичай означає процес перевірки ефективності моделі прогнозування, що є аналогом терміну «тест» у машинному навчанні. Щоб уникнути цієї плутанини, слово «набір розробки» іноді використовується як заміна «набору для перевірки».

Переобладнання відноситься до ситуації, коли модель вивчає статистичні закономірності, специфічні для навчального набору, тобто в кінцевому підсумку запам’ятовує невідповідний шум замість того, щоб вивчати сигнал, і, отже, працює менш добре на наступному новому наборі даних. Це одна з головних проблем машинного навчання, оскільки переобладнану модель неможливо узагальнити на дані, які ніколи не бачили. У цьому сенсі тестовий набір відіграє ключову роль у належній оцінці продуктивності моделей машинного навчання, як обговорювалося в попередньому розділі. Звичайна перевірка на розпізнавання переобладнання навчальним даним полягає в моніторингу втрат і точності на наборах навчання та перевірки. Якщо модель добре працює на навчальному наборі порівняно з набором перевірки, то, ймовірно, модель була переповнена навчальними даними. Для мінімізації переобладнання було запропоновано кілька методів. Найкраще рішення для зменшення переобладнання – отримати більше даних про тренування. Модель, навчена на більшому наборі даних, як правило, краще узагальнює, хоча це не

завжди досягається в медичній візуалізації. Інші рішення включають регуляризацію з випаданням або зменшенням ваги, нормалізацію пакетів і збільшення даних, а також зниження архітектурної складності. Dropout[11] — це нещодавно введена методика регуляризації, коли випадково вибрані активації встановлюються в 0 під час навчання, так що модель стає менш чутливою до певних ваг у мережі. Зниження ваги, також зване регуляризацією L2, зменшує переобладнання, штрафуючи ваги моделі, щоб ваги приймали лише невеликі значення. Пакетна нормалізація — це тип додаткового рівня, який адаптивно нормалізує вхідні значення наступного шару, пом'якшуючи ризик переобладнання, а також покращуючи градієнтний потік через мережу, забезпечуючи більш високу швидкість навчання та зменшуючи залежність від ініціалізації. Збільшення даних також ефективне для зменшення переобладнання, що є процесом модифікації навчальних даних за допомогою випадкових перетворень, таких як гортання, переклад, обрізання, обертання та випадкове стирання, так що модель не бачитиме точно однакові вхідні дані під час навчальні ітерації. Незважаючи на ці зусилля, все ще існує занепокоєння щодо переобладнання набору перевірки, а не до навчального набору через витік інформації під час тонкого налаштування гіперпараметрів і процесу вибору моделі. Тому звіт про продуктивність остаточної моделі на окремому (невидимому) наборі тестів, а в ідеалі — на зовнішніх наборах даних перевірки, якщо це можливо, є вирішальним для перевірки узагальнюваності моделі.

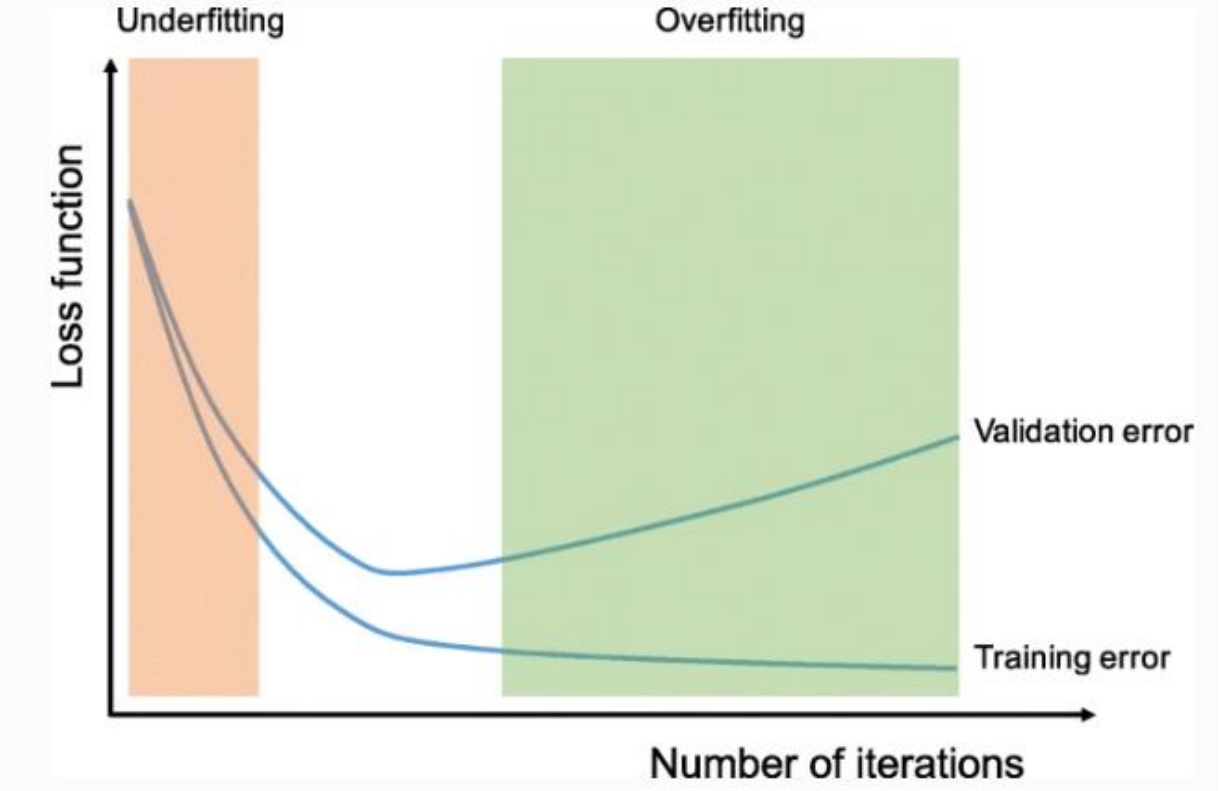

Рисунок 2.11 - Графік навчання моделі

Велика кількість добре позначених даних у медичній візуалізації є бажаною, але рідко доступною через вартість та необхідне навантаження експертів з радіології. Існує кілька доступних методів для ефективного навчання моделі на меншому наборі даних: збільшення даних і навчання з перенесенням. Оскільки розширення даних було коротко розглянуто в попередньому розділі, цей розділ зосереджено на навчанні з перенесенням.

Трансферне навчання[12] – це поширена та ефективна стратегія навчання мережі на невеликому наборі даних, де мережу попередньо тренують на надзвичайно великому наборі даних, наприклад ImageNet, який містить 1,4 мільйона зображень з 1000 класами, потім повторно використовується та застосовується до заданого завдання. інтерес. Основне припущення трансферного навчання полягає в тому, що загальні характеристики, засвоєні на достатньо великому наборі даних, можуть бути спільними між, здавалося б, розрізненими наборами даних. Ця переносимість вивчених загальних функцій є унікальною

перевагою глибокого навчання, яке стає корисним у різних предметних задачах з невеликими наборами даних. На даний момент багато моделей, попередньо навчені на наборі даних ImageNet Challenge, відкриті для громадськості та легкодоступні, а також їхні вивчені ядра та ваги, такі як AlexNet , VGG , ResNet, Inception, і DenseNet. На практиці існує два способи використання попередньо навченої мережі: виділення фіксованих ознак і тонке налаштування.

Метод вилучення фіксованих ознак[13] — це процес видалення повністю підключених шарів із мережі, попередньо натренованої на ImageNet, із збереженням мережі, що залишилася, яка складається з серії шарів згортки та об’єднання, званих базою згортки, як екстрактор фіксованих ознак. У цьому сценарії будь-який класифікатор машинного навчання, такий як випадкові ліси та машини опорних векторів, а також звичайні повністю підключені шари в CNN, можуть бути додані поверх екстрактора фіксованих ознак, в результаті чого навчання обмежується доданим класифікатором на заданий набір даних, що цікавить. Цей підхід не є поширеним у дослідженнях глибокого навчання медичних зображень через несхожість між ImageNet і наданими медичними зображеннями.

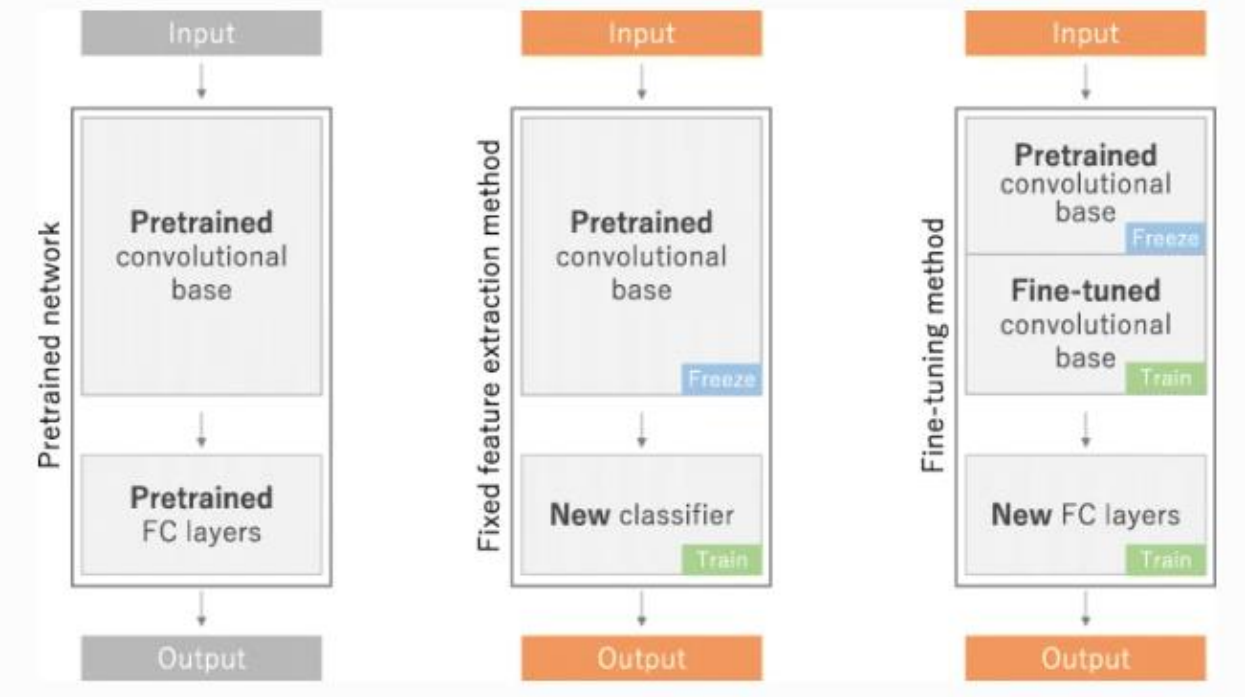

Рисунок 2.12 - Візуалізація методу вилучення фіксованих ознак

Метод точного налаштування, який частіше застосовується до радіологічних досліджень, полягає не тільки в заміні повністю пов’язаних шарів попередньо навченої моделі новим набором повністю пов’язаних шарів для перенавчання на заданому наборі даних, а й у точному налаштуванні повністю або частково. ядер у попередньо навченій згортковій основі за допомогою зворотного поширення. Усі шари в згортковій основі можна точно налаштувати або, як альтернатива, деякі більш ранні шари можна зафіксувати під час точного налаштування решти більш глибоких шарів. Це мотивується спостереженням, що функції раннього рівня здаються більш загальними, включаючи такі функції, як краї, застосовні до різноманітних наборів даних і завдань, тоді як пізніші функції поступово стають більш специфічними для певного набору даних або завдання.

Одним з недоліків трансферного навчання є його обмеження на вхідні розміри. Вхідне зображення має бути 2D з трьома каналами, що відповідають RGB, оскільки набір даних ImageNet[14] складається з двовимірних кольорових зображень, які мають три канали (RGB: червоний, зелений і синій), тоді як медичні зображення у відтінках сірого мають лише один канал (рівні сірого). З іншого боку, висота та ширина вхідного зображення можуть бути довільними, але не надто малими, шляхом додавання шару глобального об’єднання між згортковою базою та доданими повністю зв’язаними шарами.

Також зростає інтерес до використання нерозмічених даних, тобто напівконтрольованого навчання, для подолання проблеми з малими даними. Приклади цієї спроби включають псевдомітку та включення генеративних моделей, таких як генеративні змагальні мережі (GAN) . Однак, чи дійсно ці методи можуть допомогти покращити ефективність глибокого навчання в радіології, не зрозуміло і залишається сферою активного дослідження.

2.2 Технології розробки системи

Python [15] — це широко використовувана мова програмування загального призначення високого рівня. Спочатку він був розроблений Гвідо ван Россумом у

1991 році і розроблений Python Software Foundation. В основному він був розроблений для акцентування уваги на читабельності коду, а його синтаксис дозволяє програмістам висловлювати поняття в меншій кількості рядків коду.

Наприкінці 1980-х років мала бути написана історія. Саме тоді почалася робота над Python. Незабаром після цього в грудні 1989 року Гвідо Ван Россум почав працювати з додатками в Centrum Wiskunde & Informatica (CWI), розташованому в Нідерландах. Спершу це було розпочато як хобі, тому що він шукав цікавий проект, щоб зайняти його під час Різдва. Мова програмування, в якій, як кажуть, досяг успіху Python, це мова програмування ABC, яка мала взаємодію з операційною системою Amoeba і мала функцію обробки винятків. Він уже допоміг створити ABC на початку своєї кар'єри, і він бачив деякі проблеми з ABC, але йому сподобалися більшість функцій. Після цього те, що він зробив, було справді дуже розумним. Він взяв синтаксис ABC і деякі його хороші риси. Це також викликало багато скарг, тому він повністю виправив ці проблеми і створив хорошу мову сценаріїв, яка усунула всі недоліки. Натхненням для створення назви послужило телешоу BBC – «Літаючий цирк Монті Пайтона», оскільки він був великим шанувальником телешоу, а також хотів коротку, унікальну і трохи загадкову назву для свого винаходу, тому він назвав його Python! Він був «Довічним диктатором» (BDFL), поки не пішов у відставку з посади лідера 12 липня 2018 року. Довгий час він працював у Google, але зараз працює в Dropbox.

Нарешті, мова була випущена в 1991 році. Коли вона була випущена, вона використовувала набагато менше кодів для вираження концепцій, якщо порівнювати її з Java, C++ і C. Його філософія дизайну також була досить хорошою. Його головна мета — забезпечити читабельність коду та покращити продуктивність розробників. Коли він був випущений, він мав більш ніж достатньо можливостей для надання класам успадкування, обробки винятків кількох основних типів даних і функцій.

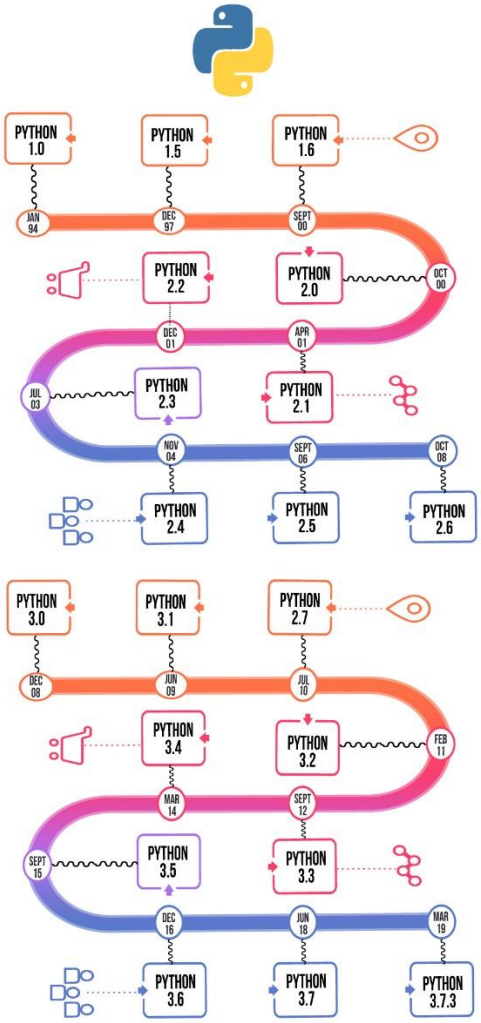

Рисунок 2.13 - Візуалізація версій Python

Дві з найбільш використовуваних версій мають Python 2.x і 3.x. Між ними велика конкуренція, і вони, схоже, мають досить багато різних шанувальників.

Для різних цілей, таких як розробка, створення сценаріїв, генерація та тестування програмного забезпечення, ця мова використовується. Завдяки своїй елегантності та простоті найкращі технологічні організації, такі як Dropbox, Google, Quora, Mozilla, Hewlett-Packard, Qualcomm, IBM і Cisco, впровадили Python.

Python пройшов довгий шлях, щоб стати найпопулярнішою мовою кодування у світі. Python щойно виповнилося 30, і зовсім нещодавно на pycon22 (python confrence) фонд Anaconda випустив нову функцію, відому як pyscript, і тепер Python

можна писати та запускати у браузері, як javascript, що раніше було неможливо, але все ще є той невідомий шарм і фактор X, який добре видно з того факту, що користувачі Google постійно шукали Python набагато більше, ніж вони шукали Кім Кардашян, Дональда Трампа, Тома Круза тощо.

Keras — високорівневий API глибокого навчання, розроблений Google для впровадження нейронних мереж. Він написаний на Python і використовується для полегшення реалізації нейронних мереж. Він також підтримує обчислення кількох серверних нейронних мереж.

Рисунок 2.14 - Лого Keras

Keras відносно легко вивчати та працювати з ним, оскільки він забезпечує інтерфейс Python з високим рівнем абстракції, маючи можливість кількох серверних елементів для обчислень. Це робить Keras повільнішим, ніж інші фреймворки глибокого навчання, але надзвичайно зручним для початківців.

Keras дозволяє перемикатися між різними бекендами. Фреймворки, які підтримує Keras:

1. Tensorflow;
2. Теано;
3. PlaidML;
4. MXNet;
5. CNTK (Microsoft Cognitive Toolkit).

З цих п’яти фреймворків TensorFlow прийняв Keras як офіційний високорівневий API. Keras вбудований в TensorFlow і може використовуватися для швидкого глибокого навчання, оскільки він надає вбудовані модулі для всіх обчислень нейронної мережі. У той же час обчислення, що включають тензори, графіки обчислень, сеанси тощо, можна зробити на замовлення за допомогою Tensorflow Core API, що надає вам повну гнучкість і контроль над вашим додатком, а також дозволяє реалізувати свої ідеї за відносно короткий час.

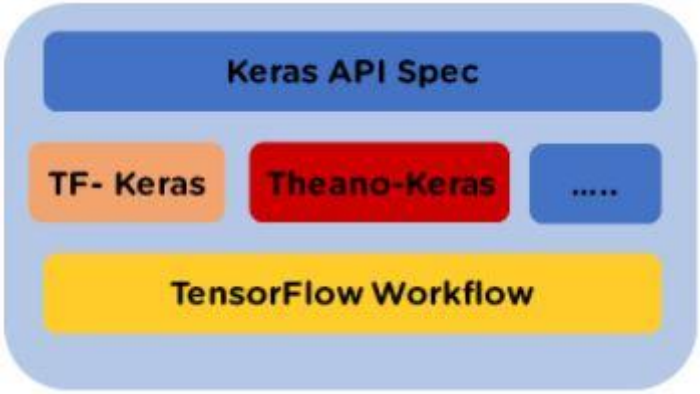

Рисунок 2.15 - Структура Keras

TensorFlow дозволяє розробникам створювати графіки потоків даних — структури, які описують, як дані переміщуються по графіку або ряду вузлів обробки. Кожен вузол на графі представляє математичну операцію, а кожне з’єднання або ребро між вузлами є багатовимірним масивом даних або тензором.

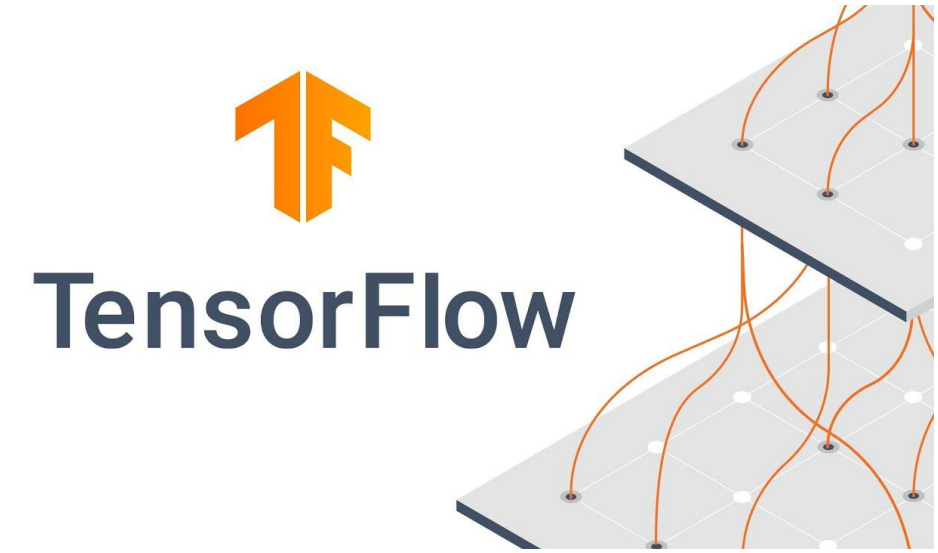

Рисунок 2.16 - Лого TensorFlow

Програми TensorFlow можна запускати практично на будь-якій зручній цілі: локальній машині, кластері в хмарі, пристроях iOS і Android, процесорах або графічних процесорах. Якщо ви використовуєте власну хмару Google, ви можете запустити TensorFlow на спеціальному силіконовому блокі обробки TensorFlow (TPU) від Google для подальшого прискорення. Однак отримані моделі, створені TensorFlow, можна розгорнути на будь-якому пристрої, де вони будуть використовуватися для прогнозування.

TensorFlow 2.0, випущений у жовтні 2019 року, оновив фреймворк багатьма способами на основі відгуків користувачів, щоб полегшити роботу (наприклад, за допомогою відносно простого API Keras для навчання моделей) і зробити його більш продуктивним. Розподілене навчання легше виконувати завдяки новому API, а підтримка TensorFlow Lite дає змогу розгортати моделі на більшій різноманітності платформ. Однак код, написаний для попередніх версій TensorFlow, потрібно переписати — іноді лише трохи, іноді значно — щоб максимально використати нові можливості TensorFlow 2.0.

TensorFlow надає все це для програміста за допомогою мови Python. Python легко вивчати та працювати з ним, і він надає зручні способи виразити, як абстракції високого рівня можна об’єднати разом. TensorFlow підтримується на

Python версії 3.7–3.10, і хоча він може працювати на попередніх версіях Python, він не гарантує, що це зробить.

Вузли та тензори в TensorFlow є об’єктами Python, а програми TensorFlow самі є додатками Python. Однак фактичні математичні операції не виконуються в Python. Бібліотеки перетворень, які доступні через TensorFlow, записуються як високопродуктивні двійкові файли C++. Python просто спрямовує трафік між частинами і надає високорівневі абстракції програмування, щоб з’єднати їх разом.

Робота високого рівня в TensorFlow — створення вузлів і шарів і їх зв’язування разом — використовує бібліотеку Keras. API Keras зовні простий; базова модель з трьома шарами може бути визначена менш ніж за 10 рядків коду, а навчальний код для цього ж займає лише кілька рядків коду більше. Але якщо ви хочете «підняти капот» і виконувати більш детальну роботу, наприклад, написати свій власний цикл навчання.

3 МОДЕЛЮВАННЯ ТА РЕАЛІЗАЦІЯ НЕЙРОННИХ МЕРЕЖ

3.1 Опис датасету

За останні чотири десятиліття візуалізація відігравала різноманітну роль у вивченні хвороби Альцгеймера. Спочатку комп'ютерна томографія, а потім магнітно-резонансна томографія використовувалися для діагностики, щоб виключити інші причини деменції. Зовсім недавно різноманітні методи візуалізації, включаючи структурну та функціональну МРТ та позитронно-емісійну томографію, дослідження мозкового метаболізму за допомогою фтор-дезокси-d-глюкози та амілоїдних індикаторів, таких як Піттсбургська сполука-В (PiB), показали характерні зміни в головному мозку пацієнтів з хворобою Альцгеймера, а також у продромальних і навіть пресимптоматичних станах, які можуть допомогти врегулювати патофізіологічний процес хвороби Альцгеймера. Жоден метод зображення не може служити всім цілям, оскільки кожен має унікальні сильні та слабкі сторони. Ці способи та їх особливі корисності обговорюються в цій статті. Завдання майбутнього полягатиме в тому, щоб об'єднати біомаркери візуалізації для найефективнішого полегшення діагностики, визначення стадії захворювання та, найголовніше, розробки ефективних методів лікування, що модифікують захворювання.

Обраний датасет Alzheimer's Dataset. Дані збираються вручну з різних веб-сайтів з перевіркою кожного класу. Дані складаються з зображень МРТ. Дані мають чотири класи зображень як для навчання, так і для тестового набору:

1. Mild Demented;
2. Moderate Demented;
3. Non Demented;
4. Very Mild Demented.

3.2 Підготовка датасету

Для реалізації вирішено використовувати набір даних Kaggle Alzheimer. tf.keras має нову функцію попередньої обробки, яка може легко завантажувати зображення для каталогу. Щоб ця функція працювала, дані мають бути структуровані у форматі каталогу файлів.

```
main_directory/
    class1/
        class1_images
    class2/
        class2_images
```

Рисунок 3.1 - Структура каталогів файлів

Введення main_directory у функцію tf.keras, вона зрозуміє структуру каталогів. У цьому випадку каталог train є основним каталогом.

Також визначається розділення 80:20 для наборів даних для навчання та перевірки.

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "../input/alzheimers-dataset-4-class-of-images/Alzheimer_s Dataset/train",
    validation_split=0.2,
    subset="training",
    seed=1337,
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)
```

Рисунок 3.2 - Розділення 80:20 для наборів даних

```
Found 5121 files belonging to 4 classes.
Using 4097 files for training.
Found 5121 files belonging to 4 classes.
Using 1024 files for validation.
```

Рисунок 3.3 - Виведення у консоль відповідного логу

Після того, як дані було завантажено, наступним кроком є візуалізація зображень. Це допомагає зрозуміти, що використовується як вхідні дані для моделі. Це також служить перевіркою, чи правильно завантажено зображення.

```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(train_ds.class_names[labels[i]])
    plt.axis("off")
```

Рисунок 3.4 - Реалізація візуалізації вхідних даних

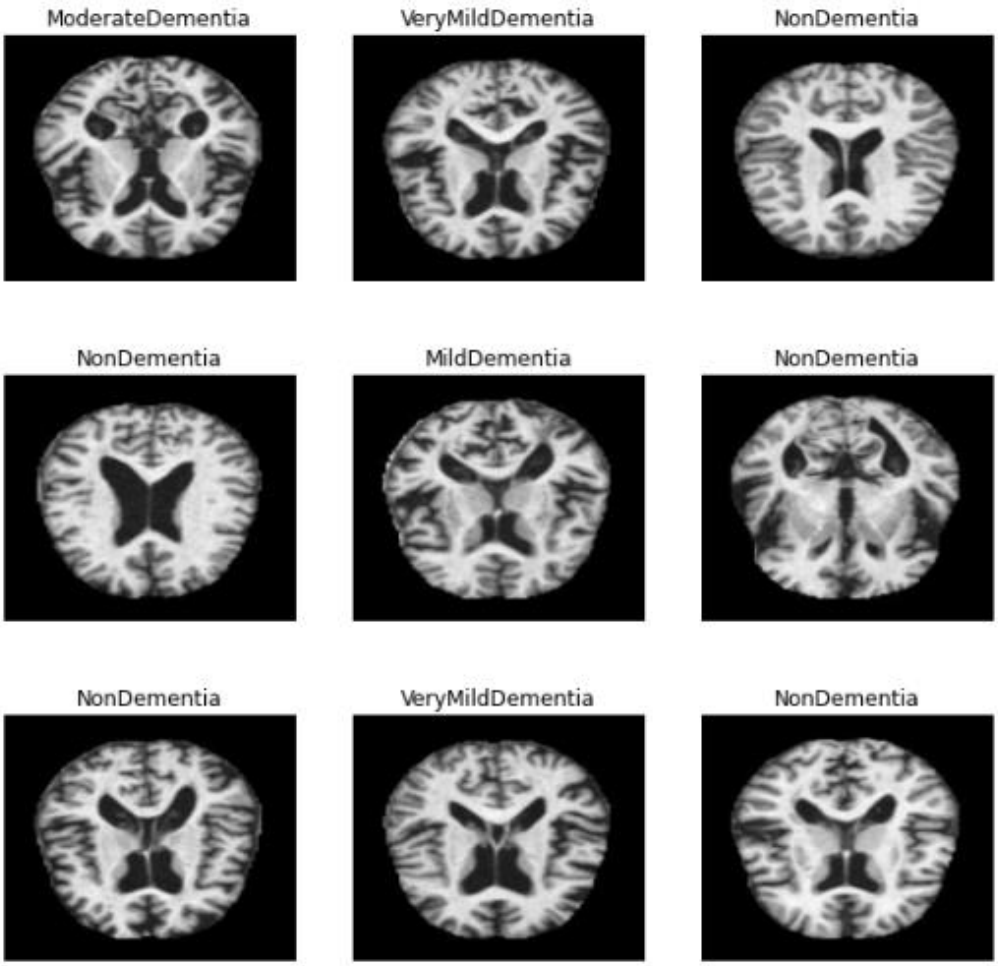

Рисунок 3.5 - Виведення вхідних даних

Оскільки датасет має категоричні та неперервнимі дані, треба перетворити модель в one-hot encodings. One-hot encodings— це спосіб для моделі зрозуміти, що обробляються категоріальні, а не безперервні дані. Перетворення функцій, щоб вони були більш зрозумілими, називається інженерією функцій.

```
def one_hot_label(image, label):
    label = tf.one_hot(label, NUM_CLASSES)
    return image, label

train_ds = train_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
val_ds = val_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
```

Рисунок 3.6 - Реалізація one-hot encodings

Найбільш звичайним показником для використання є accuracy. Однак accuracy не може використовуватися для незбалансованих наборів даних. Перевіримо, скільки зображень у кожному класі для навчальних даних.

```
NUM_IMAGES = []

for label in class_names:
    dir_name = "../input/alzheimers-dataset-4-class-of-images/Alzheimer_s Dataset/train/" + label[:-2] + 'ed'
    NUM_IMAGES.append(len([name for name in os.listdir(dir_name)]))
```

Рисунок 3.7 - Перевірка навчальних даних на збалансованість

```
[717, 52, 2560, 1792]
```

Рисунок 3.8 - Демонстрація незбалансованості

3.3 Архітектура системи на AlexNet

AlexNet[16] став переможцем у ILSVRC 2012. Він вирішує проблему класифікації зображень, де вхідним є зображення одного з 1000 різних класів (наприклад, кішки, собаки тощо), а вихідним є вектор із 1000 чисел. I-й елемент вихідного вектора інтерпретується як ймовірність того, що вхідне зображення належить до i-го класу. Отже, сума всіх елементів вихідного вектора дорівнює 1.

Вхідним для AlexNet є зображення RGB розміром 256×256. Це означає, що всі зображення в навчальному наборі та всі тестові зображення повинні мати розмір 256×256.

Якщо вхідне зображення не 256×256, його потрібно перетворити на 256×256, перш ніж використовувати його для навчання мережі. Щоб досягти цього, розмір меншого розміру змінюється до 256, а потім отримане зображення обрізається, щоб отримати зображення розміром 256×256. На малюнку нижче показано приклад.

Рисунок 3.9 - Візуалізація операції Crop

Якщо вхідне зображення має відтінки сірого, воно перетворюється на зображення RGB шляхом реплікації одного каналу для отримання 3-канального зображення RGB. Випадкові кадри розміром 227×227 були згенеровані всередині зображень 256×256 для живлення першого шару AlexNet.

AlexNet є набагато більшим, ніж попередні CNN, які використовувалися для задач комп'ютерного зору. Він має 60 мільйонів параметрів і 650 000 нейронів, і на навчання на двох графічних процесорах GTX 580 3 ГБ знадобилося п'ять-шість днів. Сьогодні існують набагато складніші CNN, які можуть дуже ефективно працювати на швидших графічних процесорах навіть на дуже великих наборах даних.

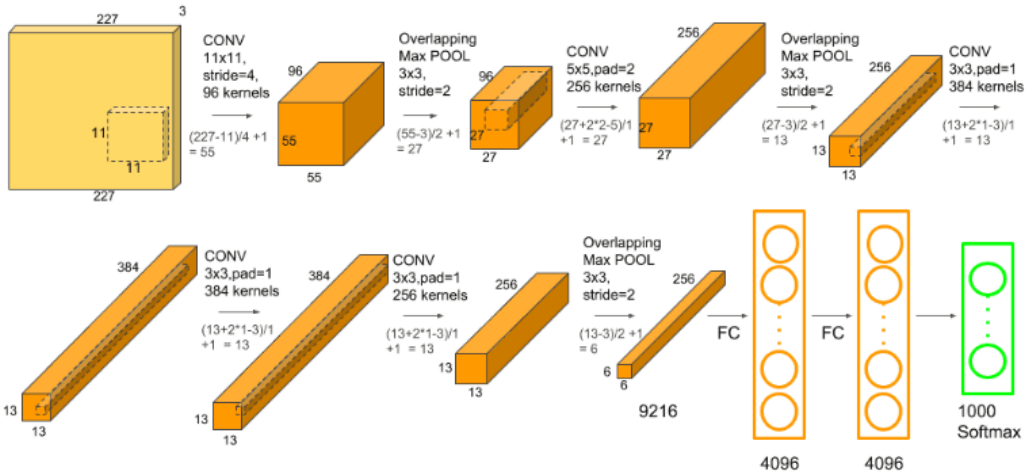

Рисунок 3.10 - Архітектура AlexNet

Кілька згорткових ядер виділяють цікаві особливості зображення. В одному згортковому шарі зазвичай є багато ядер однакового розміру. Наприклад, перший Conv Layer AlexNet містить 96 ядер розміром 11x11x3. Ширина і висота ядра зазвичай однакові, а глибина така ж, як і кількість каналів.

За першими двома згортковими шарами слідують шари максимального перекриття, які ми описуємо далі. Третій, четвертий і п'ятий згорткові шари з'єднані безпосередньо. За п’ятим згортковим шаром слід шар перекривання максимального об’єднання, вихід якого переходить у серію з двох повністю з’єднаних шарів. Другий повністю підключений шар подається в класифікатор softmax з 1000 мітками класів.

Нелінійність ReLU застосовується після всіх шарів згортки та повністю зв’язаних. Після нелінійності ReLU першого та другого шарів згортки слід локальний крок нормалізації перед виконанням об’єднання. Але пізніше дослідники не знайшли нормалізацію дуже корисною. Шари максимального об’єднання зазвичай використовуються для зменшення ширини та висоти тензорів, зберігаючи глибину однаковою.

Шари Max Pool, що перекриваються, подібні до шарів Max Pool, за винятком того, що сусідні вікна, для яких обчислюється максимальний пул, перекривають одне одного. Такий перекриваючий характер об’єднання допомогає зменшити частоту помилок першої категорії на 0,4% і частоту помилок топ-5 відповідно на 0,3% у порівнянні з використанням вікон об’єднання, що не перекриваються, розміром 2×2 з кроком 2, що дасть однаковий результат.

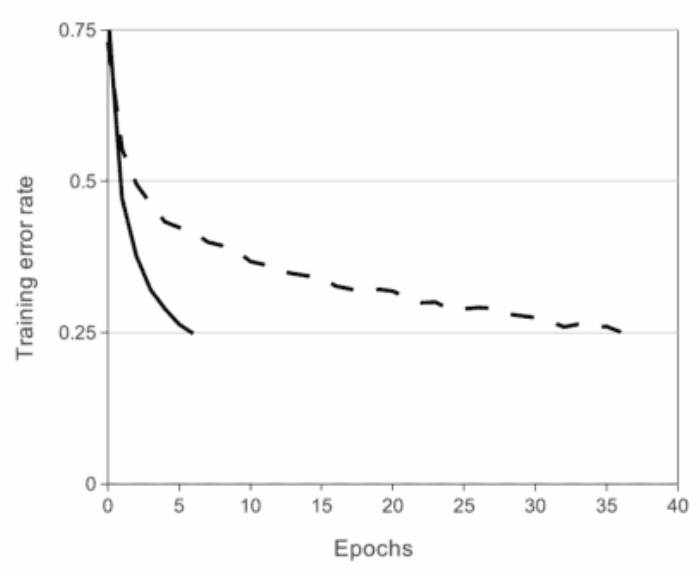

Рисунок 3.11 - Графік помилок

Шари максимального об’єднання зазвичай використовуються для зменшення ширини та висоти тензорів, зберігаючи глибину однаковою. Шари Max Pool, що перекриваються, подібні до шарів Max Pool, за винятком того, що сусідні вікна, для яких обчислюється максимальний пул, перекривають одне одного. Автори використовували об’єднані вікна розміром 3×3 із кроком 2 між сусідніми вікнами. Такий перекриваючий характер об’єднання допоміг зменшити частоту помилок першої категорії на 0,4% і частоту помилок топ-5 відповідно на 0,3% у порівнянні з використанням вікон об’єднання, що не перекриваються, розміром 2×2 з кроком 2, що дасть однаковий результат. розміри.

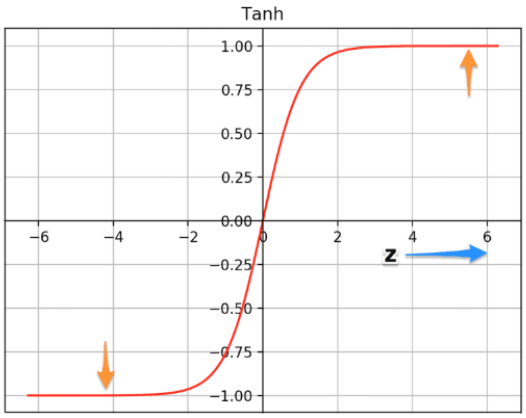

Рисунок 3.12 - Графік tanh

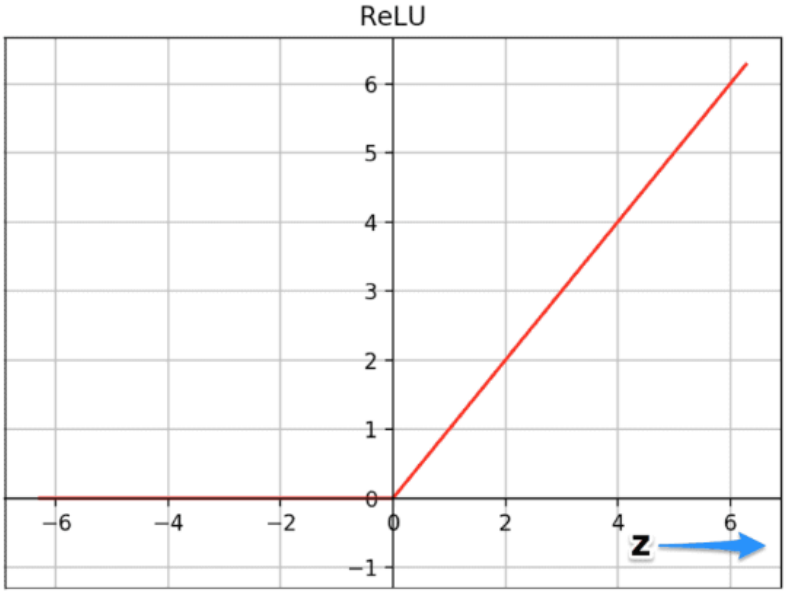

Рисунок 3.13 - Графік ReLu

Вище наведені графіки двох функцій – tanh і ReLU. Функція tanh насичується при дуже високих або дуже низьких значеннях z. У цих областях нахил функції дуже близький до нуля. Це може уповільнити градієнтний спуск. З іншого боку, нахил функції ReLU не близький до нуля для більш високих додатних значень z. Це допомагає оптимізації збігатися швидше. Для негативних значень z нахил все ще дорівнює нулю, але більшість нейронів нейронної мережі зазвичай мають позитивні значення. ReLU також перемагає над сигмовидною функцією з тієї ж причини.

Розмір нейронної мережі — це її здатність до навчання, але якщо ви не будете обережні, вона спробує запам’ятати приклади в даних навчання, не розуміючи концепції. В результаті нейронна мережа буде надзвичайно добре працювати з навчальними даними, але вони не можуть засвоїти справжню концепцію. Він не зможе добре працювати з новими та невидимими тестовими даними. Це називається переобладнанням.

3.4 Реалізація системи на AlexNet

Для реалізації буде використовуватися архітектура моделі AlexNet.

Використовуючи tf.keras, можливо легко створити шари CNN.

```
model = tf.keras.Sequential()
    model.add(layers.Conv2D(filters=96, kernel_size=(11, 11),
                            strides=(4, 4), activation="relu",
                            input_shape=(*IMAGE_SIZE, 3)))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)))
    model.add(layers.Conv2D(filters=256, kernel_size=(5, 5),
                            strides=(1, 1), activation="relu",
                            padding="same"))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)))
    model.add(layers.Conv2D(filters=384, kernel_size=(3, 3),
                            strides=(1, 1), activation="relu",
                            padding="same"))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(filters=384, kernel_size=(3, 3),
                            strides=(1, 1), activation="relu",
                            padding="same"))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(filters=256, kernel_size=(3, 3),
                            strides=(1, 1), activation="relu",
                            padding="same"))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation="relu"))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(4, activation="softmax"))
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer=tf.optimizers.SGD(lr=0.001),
                  metrics=['accuracy'])
    model.summary()
```

Рисунок 3.14 - Реалізація моделі AlexNet на Python

Щоб ефективніше навчати модель. Будуть використовуватись зворотні виклики, щоб налаштувати швидкість навчання та зупинити модель, коли вона збіжиться.

Швидкість навчання є дуже важливим гіперпараметром у моделі. Занадто високий LR запобіжить зближенню моделі. Занадто повільний LR зробить процес занадто довгим. Раннє зупинення моделі є одним із механізмів, що запобігає переобладнанню.

```
def exponential_decay(lr0, s):
    def exponential_decay_fn(epoch):
        return lr0 * 0.1 **(epoch / s)
    return exponential_decay_fn

exponential_decay_fn = exponential_decay(0.01, 20)

lr_scheduler = tf.keras.callbacks.LearningRateScheduler(exponential_decay_fn)

checkpoint_cb = tf.keras.callbacks.ModelCheckpoint("alzheimer_model.h5",
                                                   save_best_only=True)

early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10,
                                                     restore_best_weights=True)
```

Рисунок 3.15 - Реалізація раннього зупинення моделі

Процесс навчання моделі складається з 100 епох, але у зв’язку з використанням раннього зупинення моделі, їх може бути менше.

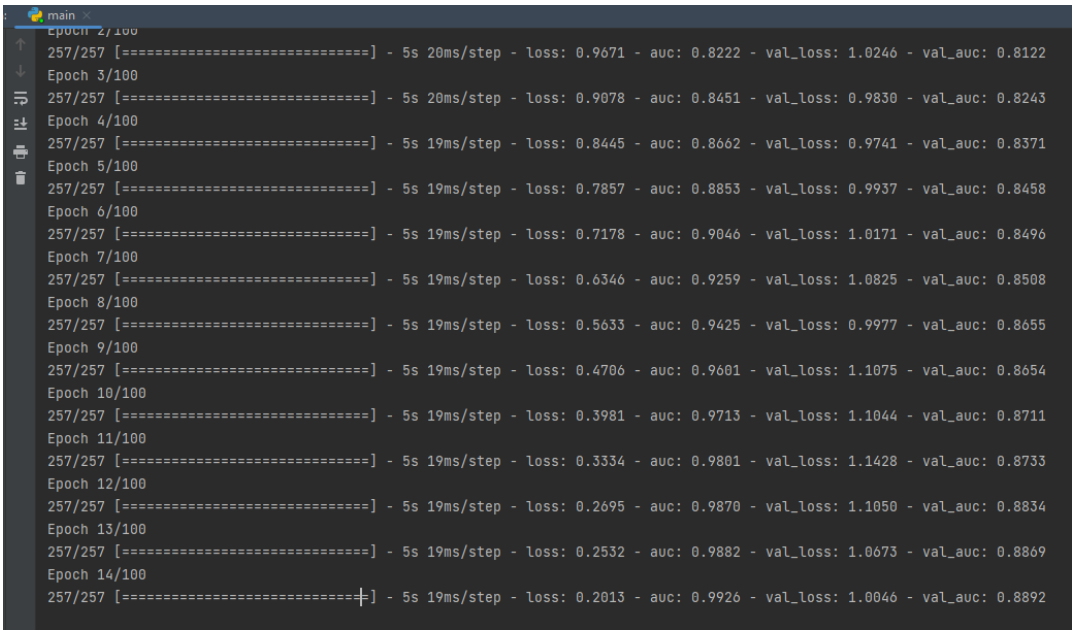

Рисунок 3.16 - Процес навчання моделі AlexNet

Побудовано графік метрики ROC AUC і втрати після кожної епохи для даних навчання та перевірки. Хоча не було використане випадкове початкове значення для нашого блокнота, результати можуть дещо відрізнятися, загалом бали для даних перевірки подібні, якщо не кращі, ніж навчальні дані.

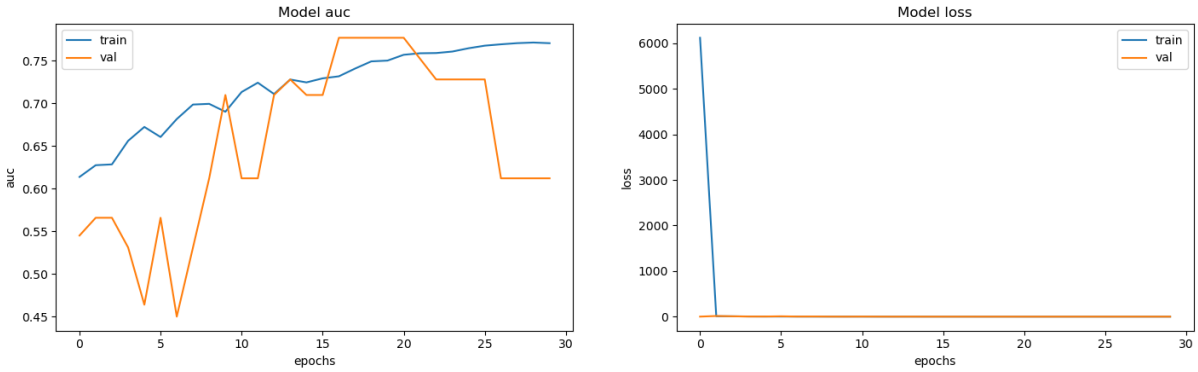

Рисунок 3.17 - Графік метрики ROC AUC для моделі AlexNet

Хоча було використано набір даних перевірки для постійної оцінки моделі, також є окремий набір даних тестування. Підготуємо набір даних для тестування.

```
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "../input/alzheimers-dataset-4-class-of-images/Alzheimer_s Dataset/test",
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)
```

Рисунок 3.18 - Підготування тестових даних

```
Found 1279 files belonging to 4 classes.
80/80 [==============================] - 2s 22ms/step - loss: 1.0389 - auc: 0.7806
```

Рисунок 3.19 - Результат перевірки тестовими даними

Параметр loss дорівнює 1.03, що є в цілому задовільно для тестового набору даних. Параметр AUC дорівнює 0.78, що також є задовільним для тестового набору даних.

Спробуємо на основі отриманої моделі отримати прогноз по одній МРТ, яка приналежить класу NonDementia.

```
This image most likely belongs to NonDementia with a 30.63 percent confidence.

Process finished with exit code 0
```

Рисунок 3.20 - Результат прогнозу AlexNet

Модель вірно прогнозувала клас MPT, але має доволі низький процент впевненості.

3.5 Архітектура системи на LeNet

LeNet-5[17] була однією з найперших згорткових нейронних мереж, яка пропагувала подію глибокого навчання. Після численних років аналізу та безлічі переконливих ітерацій кінцевий результат отримав назву LeNet-5 у 1988 році.

У 1989 році Yann LeCun et al. У Bell Labs спочатку застосовували правило зворотного поширення до всіх розумних додатків і вважали, що гнучкість, яка забезпечується узагальненням мережі, може бути значно збільшена, якщо надати обмеження з області завдань.

Він об’єднав згорткову нейронну мережу, навчену алгоритмами зворотного поширення, для сканування письмових чисел, які пізніше використав для виявлення різних рукописних поштових індексів, наданих Службою зв’язку США. Це був елементарний приклад того, що пізніше стали називати LeNet.

Протягом того ж року Лекун пояснив крихітну аномалію розпізнавання рукописних цифр в іншій роботі і довів, що навіть якщо припустити, що проблема лінійно розділена, одношарові мережі, як правило, мають погані навички узагальнення.

Після того, як аномалію видалено за допомогою інваріантних детекторів ознак у багатошаровій неприродній мережі, модель може працювати надзвичайно добре. Він вважав, що ці результати підтвердили, що мінімізація кількості вільних параметрів у нейронній мережі може посилити узагальнення того ж самого.

У 1989 році Ян Лекун представив згорткову нейронну мережу під назвою LeNet. Загалом, LeNet відноситься до LeNet-5 і є простою згортковою нейронною мережею.

Згорткові нейронні мережі є формою нейронної мережі з прямим зв'язком, чиї штучні нейрони відповідатимуть місцевості оточуючих клітин у межах покриття, які варіюються та добре працюють у великомасштабній обробці зображень.

LeNet-5 означає появу CNN і визначає основні елементи CNN. Однак на той момент це було зовсім не в моді через нестачу апаратного обладнання, зокрема GPU (Graphics Process Unit, спеціалізована електронна схема, призначена для зміни пам'яті для прискорення створення зображень під час буфера, призначеного для виведення на демонстраційний пристрій) і інший алгоритм, цінують SVM, здатні робити подібні ефекти або, можливо, перевищують LeNet.

LeNet-5 широко використовувався у масових масштабах для звичайної класифікації рукописних цифр у фінансових установах, таких як банки. CNN є музою сучасного глибокого навчання, яке в першу чергу базується на комп'ютерному баченні.

Ці мережі побудовані на основі трьох основних ідей: сприйнятливих полів сусідства, спільних ваг і просторової підвибірки. Рецептивні поля сусідства зі спільними вагами є квінтесенцією згорткового шару, і більшість архітектур, визначених під використанням згорткових шарів в одній чи іншій формі.

Іншою причиною, чому LeNet є важливою структурою, є те, що раніше, ніж він був винайдений, розпізнавання символів здійснювалося в більшості випадків за допомогою використання досвіду розробки функцій вручну, а також використання моделі машинного навчання для виявлення способів класифікації. можливості, створені вручну.

LeNet зробив зайвими можливості ручної інженерії через те, що спільнота миттєво вивчає першокласну внутрішню ілюстрацію з необроблених зображень.

Мережа має 5 рівнів з параметрами, які можна вивчати, і тому названа Ленет-

5. Він має три набори шарів згортки з комбінацією середнього об’єднання. Після шарів згортки та середнього об’єднання ми маємо два повністю пов’язані шари. Нарешті, класифікатор Softmax, який класифікує зображення у відповідний клас.

Мережа має 5 рівнів з параметрами, які можна вивчати, і тому названа LeNet-5. Він має три набори шарів згортки з комбінацією середнього об’єднання. Після шарів згортки та середнього об’єднання ми маємо два повністю пов’язані шари. Нарешті, класифікатор Softmax, який класифікує зображення у відповідний клас.

Рисунок 3.21 - Візуалізація input

Потім застосовується перша операція згортки з розміром фільтра 5X5 і маємо 6 таких фільтрів. В результаті отримується карту ознак розміром 28X28X6. Тут кількість каналів дорівнює кількості застосованих фільтрів.

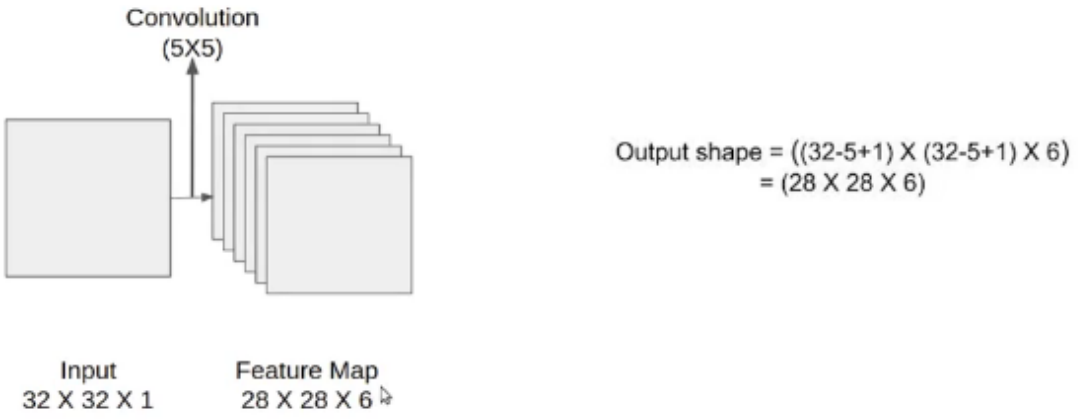

Рисунок 3.22 - Візуалізація першої операції

Після першої операції об’єднання застосовується середнє об’єднання, і розмір карти об’єктів зменшується вдвічі. Кількість каналів залишається незмінною.

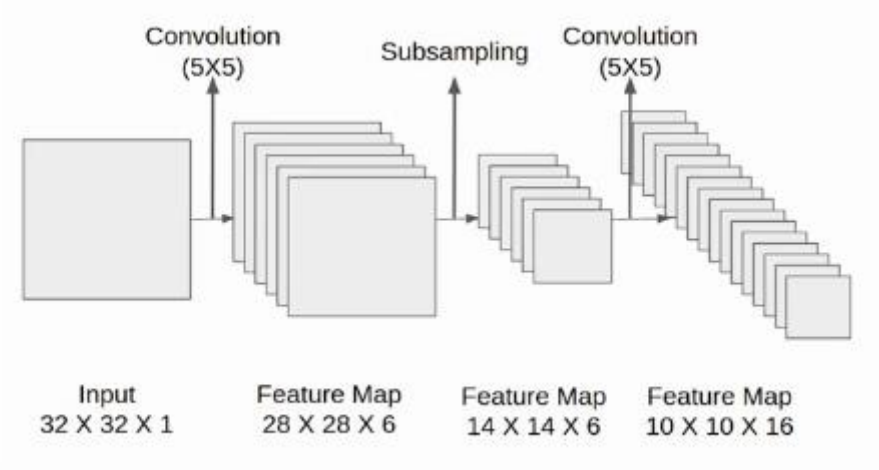

Рисунок 3.23 - Візуалізація другої операції

Далі шар згортки з шістнадцятьма фільтрами розміром 5X5. Знову карта об’єктів змінена, вона становить 10X10X16. Аналогічно розраховується вихідний розмір. Після цього знову застосовується середній шар об’єднання або підвибірки, які знову зменшили розмір карти об’єктів вдвічі, тобто 5X5X16.

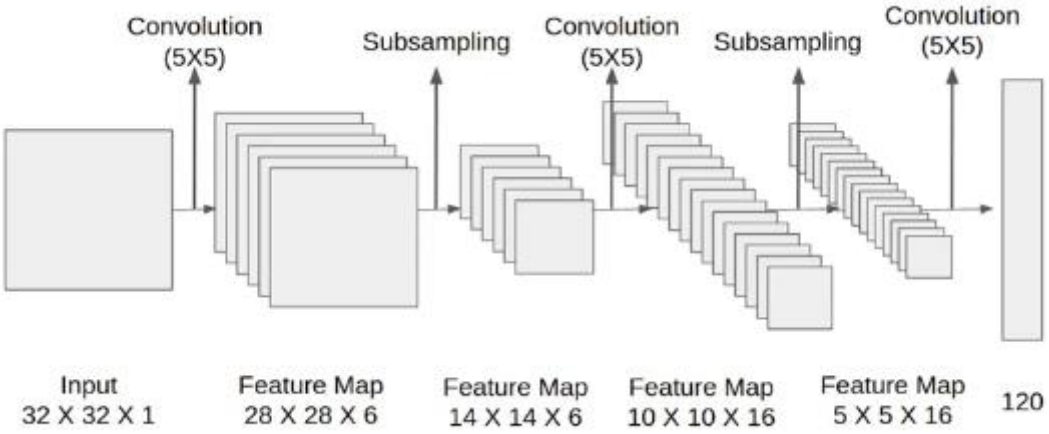

Рисунок 3.24 - Візуалізація третьої операції

Остаточний шар згортки розміром 5X5 зі 120 фільтрами. Як показано на зображенні вище. Залишивши карту об’єктів розміром 1X1X120. Після цього

результат вирівнювання становить 120 значень.

Після цих шарів згортки є повністю пов’язаний шар із вісімдесяти чотирма нейронами. Вихідний шар із десятьма нейронами, оскільки дані мають десять класів.

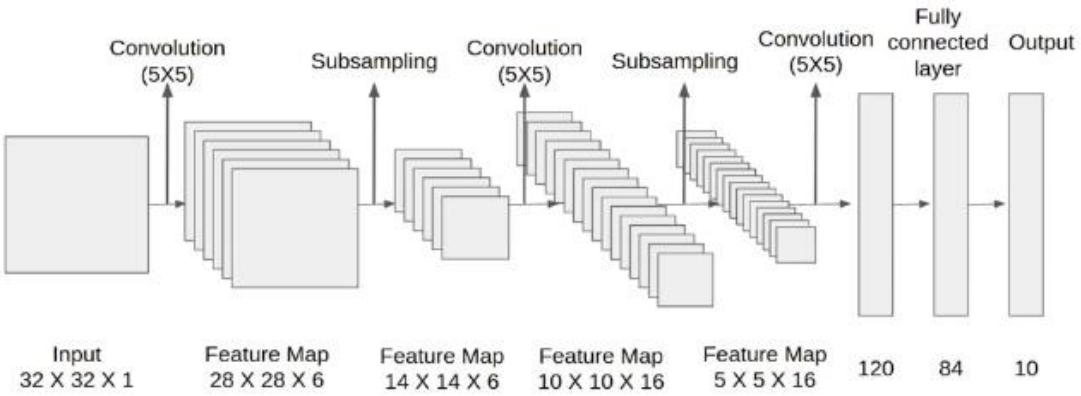

Рисунок 3.25 - Візуалізація архітектуру LeNet

3.6 Реалізація системи на LeNet

Для реалізації буде використовуватися архітектура моделі LeNet. Використовуючи tf.keras, можливо легко створити шари CNN.

```
model = tf.keras.Sequential()
    model.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu', input_shape=(*IMAGE_SIZE, 3)))
    model.add(layers.AveragePooling2D())
    model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
    model.add(layers.AveragePooling2D())
    model.add(layers.Flatten())
    model.add(layers.Dense(units=120, activation='relu'))
    model.add(layers.Dense(units=84, activation='relu'))
    model.add(layers.Dense(units=4, activation='softmax'))
    model.summary()
```

Рисунок 3.26 - Реалізація моделі LeNet на Python

Щоб ефективніше навчати модель. Будуть використовуватись зворотні виклики, щоб налаштувати швидкість навчання та зупинити модель, коли вона збіжиться.

Швидкість навчання є дуже важливим гіперпараметром у моделі. Занадто високий LR запобіжить зближенню моделі. Занадто повільний LR зробить процес занадто довгим. Раннє зупинення моделі є одним із механізмів, що запобігає переобладнанню.

```
def exponential_decay(lr0, s):
    def exponential_decay_fn(epoch):
        return lr0 * 0.1 **(epoch / s)
    return exponential_decay_fn

exponential_decay_fn = exponential_decay(0.01, 20)

lr_scheduler = tf.keras.callbacks.LearningRateScheduler(exponential_decay_fn)

checkpoint_cb = tf.keras.callbacks.ModelCheckpoint("alzheimer_model.h5",
                                                   save_best_only=True)

early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10,
                                                     restore_best_weights=True)
```

Рисунок 3.27 - Реалізація раннього зупинення моделі

Процесс навчання моделі складається з 100 епох, але у зв’язку з використанням раннього зупинення моделі, їх може бути менше.

```
257/257 [==============================] - 18s 71ms/step - loss: 1.2560 - auc: 0.7406 - val_loss: 1.0858 - val_auc: 0.7767
Epoch 19/100
257/257 [==============================] - 19s 73ms/step - loss: 1.2020 - auc: 0.7490 - val_loss: 1.0638 - val_auc: 0.7767
Epoch 20/100
257/257 [==============================] - 19s 72ms/step - loss: 1.2948 - auc: 0.7500 - val_loss: 1.0433 - val_auc: 0.7767
Epoch 21/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1841 - auc: 0.7568 - val_loss: 1.0629 - val_auc: 0.7767
Epoch 22/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1678 - auc: 0.7585 - val_loss: 1.0627 - val_auc: 0.7523
Epoch 23/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1610 - auc: 0.7588 - val_loss: 1.0649 - val_auc: 0.7279
Epoch 24/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1432 - auc: 0.7605 - val_loss: 1.1224 - val_auc: 0.7279
Epoch 25/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1279 - auc: 0.7644 - val_loss: 1.1609 - val_auc: 0.7279
Epoch 26/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1153 - auc: 0.7674 - val_loss: 1.2134 - val_auc: 0.7279
Epoch 27/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1075 - auc: 0.7690 - val_loss: 1.3513 - val_auc: 0.6120
Epoch 28/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1027 - auc: 0.7704 - val_loss: 1.4711 - val_auc: 0.6120
Epoch 29/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1020 - auc: 0.7711 - val_loss: 1.4885 - val_auc: 0.6120
Epoch 30/100
257/257 [==============================] - 18s 71ms/step - loss: 1.1075 - auc: 0.7704 - val_loss: 1.4673 - val_auc: 0.6120
```

Рисунок 3.28 - Процес навчання моделі LeNet

Побудовано графік метрики ROC AUC і втрати після кожної епохи для даних навчання та перевірки. Хоча не було використане випадкове початкове значення для нашого блокнота, результати можуть дещо відрізнятися, загалом бали для даних перевірки подібні, якщо не кращі, ніж навчальні дані.

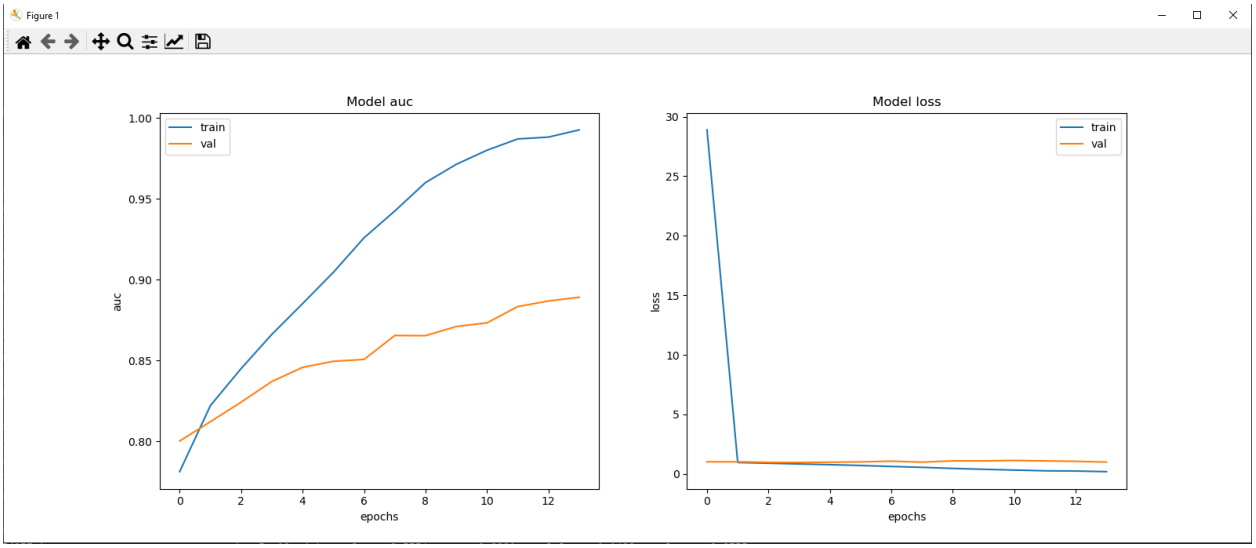

Рисунок 3.29 - Графік метрики ROC AUC для моделі LeNet

Хоча було використано набір даних перевірки для постійної оцінки моделі, також є окремий набір даних тестування. Підготуємо набір даних для тестування.

```
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "../input/alzheimers-dataset-4-class-of-images/Alzheimer_s Dataset/test",
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)
```

Рисунок 3.30 - Підготування тестових даних

```
Found 1279 files belonging to 4 classes.
80/80 [==============================] - 1s 12ms/step - loss: 1.0820 - auc: 0.8012
```

Рисунок 3.31 - Результат перевірки тестовими даними

Параметр loss дорівнює 1.08, що є в цілому задовільно для тестового набору даних. Параметр AUC дорівнює 0.8, що також є задовільним для тестового набору даних.

Спробуємо на основі отриманої моделі отримати прогноз по одній МРТ, яка приналежить класу NonDementia.

```
This image most likely belongs to VeryMildDementia with a 27.34 percent confidence.

Process finished with exit code 0
```

Рисунок 3.32 - Результат перевірки тестовими даними

```
This image most likely belongs to VeryMildDementia with a 27.34 percent confidence.

Process finished with exit code 0
```

Рисунок 3.33 - Результат прогнозу LeNet

Модель невірно прогнозувала клас МРТ, це пов`язано з низькою точністю за частиною тренувальних даних. які відповідають за валідацію.

3.7 Архітектура системи на ZFNet

ZFNet[18] використовує деконволюційну нейронну мережу для візуалізації. Деконволюційна нейронна мережа приєднана до всіх шарів CNN, які ми хочемо візуалізувати.

Зображення подається в CNN як вхідні дані, а функції витягуються за допомогою різних шарів. У підсумку створюється карта об’єктів. Щоб перевірити цю CNN, всі функції активації встановлюються на нуль, а карта ознак передається як вхід до деконволюційної нейронної мережі. Потім усі дії, що виконуються різними шарами, змінюються. Ми роз’єднуємо, виправляємо та фільтруємо вхідні дані, щоб визначити вихід попереднього рівня, який активував вибрану активацію. Це повторюється, поки не буде досягнуто вхідний піксель.

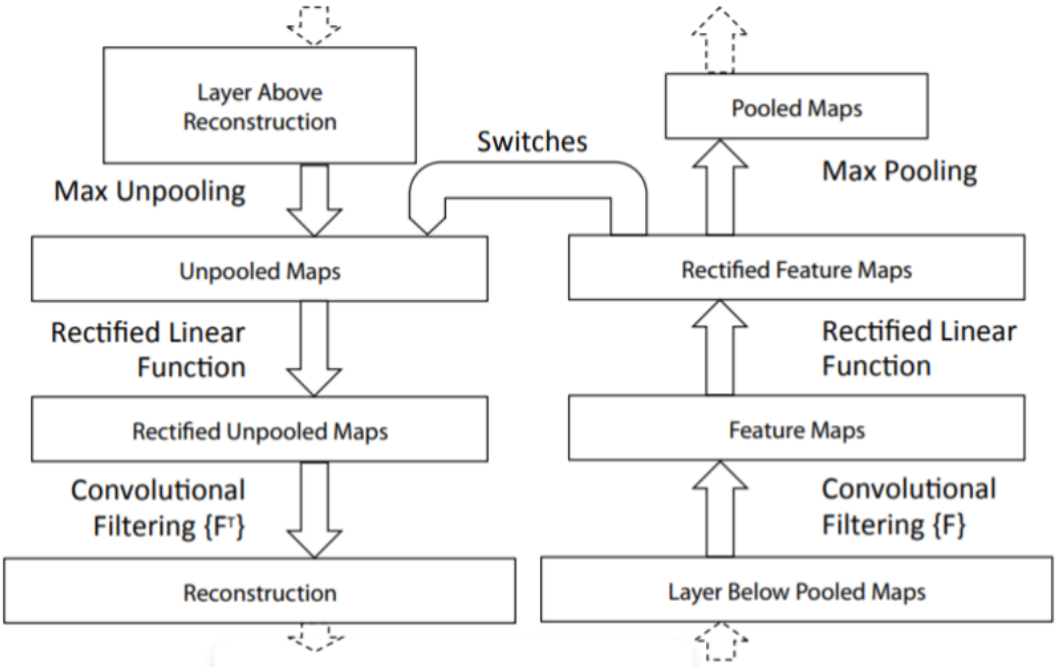

Рисунок 3.34 - Візуалізація підключення деконволюційної нейронної мережі до CNN

Операцію максимального об’єднання не можна змінити. Однак ми можемо отримати приблизне значення, відстежуючи максимальне значення пікселя, який активував поточний шар.

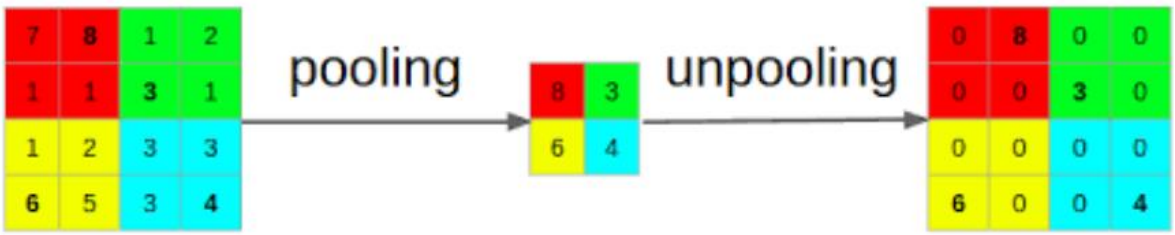

Рисунок 3.35 - Візуалізація операції максимального об’єднання

CNN може мати різні функції активації відповідно до потреб. Дуже поширеною функцією активації є ReLU. Це гарантує, що карта об’єктів завжди буде позитивною.

CNN використовує фільтри для вилучення функцій. Щоб уникнути цього, деконволюційна нейронна мережа застосовує фільтри до виправлених карт. Він

використовує іншу версію тих самих фільтрів, що використовуються в CNN. Він транспонує фільтри, тобто повертає кожен фільтр по вертикалі та горизонталі.

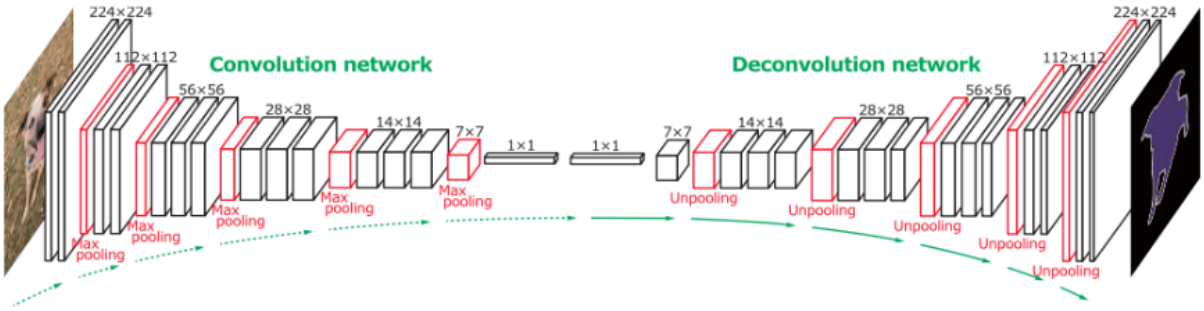

Рисунок 3.36 - Візуалізація фільтрів

Після роз'єднання отримується нерозбірні карти, які передаються функції активації ReLU, щоб отримати виправлену карту об'єктів. Потім він пропускається через шар деконволюційної фільтрації, що призводить до реконструкції карти ознак.

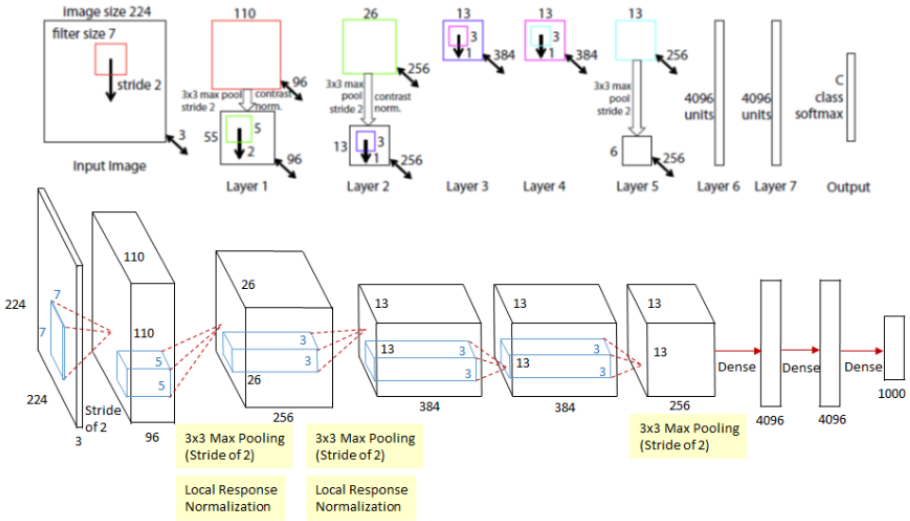

Рисунок 3.37 - Візуалізація архітектури ZFNet

3.8 Реалізація системи на ZFNet

Для реалізації буде використовуватися архітектура моделі LeNet. Використовуючи tf.keras, можливо легко створити шари CNN.

```
model = tf.keras.models.Sequential([

            tf.keras.layers.Conv2D(96, (7, 7), strides=(2, 2), activation='relu',
                    input_shape=(224, 224, 3)),
            tf.keras.layers.MaxPooling2D(3, strides=2),
    tf.keras.layers.Lambda(lambda x: tf.image.per_image_standardization(x)),

            tf.keras.layers.Conv2D(256, (5, 5), strides=(2, 2), activation='relu'),
            tf.keras.layers.MaxPooling2D(3, strides=2),
    tf.keras.layers.Lambda(lambda x: tf.image.per_image_standardization(x)),

            tf.keras.layers.Conv2D(384, (3, 3), activation='relu'),

            tf.keras.layers.Conv2D(384, (3, 3), activation='relu'),

            tf.keras.layers.Conv2D(256, (3, 3), activation='relu'),

            tf.keras.layers.MaxPooling2D(3, strides=2),

    tf.keras.layers.Flatten(),

            tf.keras.layers.Dense(4096),

            tf.keras.layers.Dense(4096),

            tf.keras.layers.Dense(10, activation='softmax')
        ])
```

Рисунок 3.38 - Реалізація моделі ZFNet на Python

Щоб ефективніше навчати модель. Будуть використовуватись зворотні виклики, щоб налаштувати швидкість навчання та зупинити модель, коли вона збіжиться.

Швидкість навчання є дуже важливим гіперпараметром у моделі. Занадто високий LR запобіжить зближенню моделі. Занадто повільний LR зробить процес занадто довгим. Раннє зупинення моделі є одним із механізмів, що запобігає переобладнанню.

```
def exponential_decay(lr0, s):
    def exponential_decay_fn(epoch):
        return lr0 * 0.1 **(epoch / s)
    return exponential_decay_fn

exponential_decay_fn = exponential_decay(0.01, 20)

lr_scheduler = tf.keras.callbacks.LearningRateScheduler(exponential_decay_fn)

checkpoint_cb = tf.keras.callbacks.ModelCheckpoint("alzheimer_model.h5",
                                                   save_best_only=True)

early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10,
                                                     restore_best_weights=True)
```

Рисунок 3.39 - Реалізація раннього зупинення моделі

Процесс навчання моделі складається з 100 епох, але у зв’язку з використанням раннього зупинення моделі, їх може бути менше.

```
Epoch 33/100
257/257 [==============================] - 13s 51ms/step - loss: 0.4478 - auc: 0.9633 - val_loss: 0.6602 - val_auc: 0.9341
Epoch 34/100
257/257 [==============================] - 14s 53ms/step - loss: 0.4072 - auc: 0.9692 - val_loss: 0.5814 - val_auc: 0.9492
Epoch 35/100
257/257 [==============================] - 13s 51ms/step - loss: 0.4022 - auc: 0.9703 - val_loss: 1.2411 - val_auc: 0.9007
Epoch 36/100
257/257 [==============================] - 13s 51ms/step - loss: 0.3545 - auc: 0.9768 - val_loss: 4.7571 - val_auc: 0.7865
Epoch 37/100
257/257 [==============================] - 13s 51ms/step - loss: 0.3302 - auc: 0.9798 - val_loss: 0.8975 - val_auc: 0.9338
Epoch 38/100
257/257 [==============================] - 13s 51ms/step - loss: 0.3161 - auc: 0.9814 - val_loss: 2.4540 - val_auc: 0.8513
Epoch 39/100
257/257 [==============================] - 13s 51ms/step - loss: 0.3053 - auc: 0.9826 - val_loss: 1.1380 - val_auc: 0.9230
Epoch 40/100
257/257 [==============================] - 13s 51ms/step - loss: 0.2776 - auc: 0.9860 - val_loss: 1.6154 - val_auc: 0.8982
Epoch 41/100
257/257 [==============================] - 13s 51ms/step - loss: 0.2506 - auc: 0.9883 - val_loss: 0.6958 - val_auc: 0.9518
Epoch 42/100
257/257 [==============================] - 13s 51ms/step - loss: 0.2489 - auc: 0.9882 - val_loss: 0.9441 - val_auc: 0.9392
Epoch 43/100
257/257 [==============================] - 13s 51ms/step - loss: 0.2301 - auc: 0.9902 - val_loss: 0.6780 - val_auc: 0.9559
Epoch 44/100
257/257 [==============================] - 13s 51ms/step - loss: 0.2186 - auc: 0.9912 - val_loss: 0.6709 - val_auc: 0.9539
```

Рисунок 3.40 - Процес навчання моделі ZFNet

Побудовано графік метрики ROC AUC і втрати після кожної епохи для даних навчання та перевірки. Хоча не було використане випадкове початкове значення для нашого блокнота, результати можуть дещо відрізнятися, загалом бали для даних перевірки подібні, якщо не кращі, ніж навчальні дані.

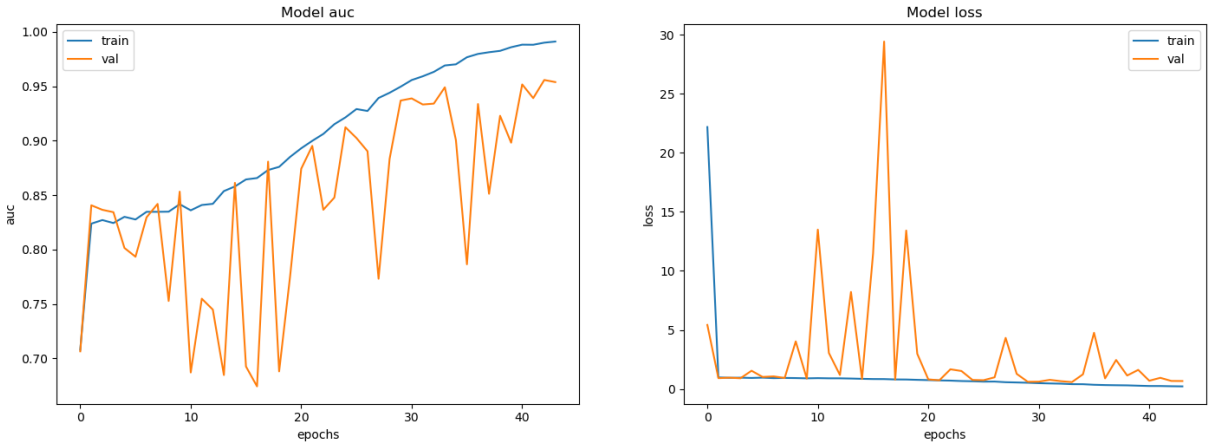

Рисунок 3.41 - Графік метрики ROC AUC для моделі ZFNet

Хоча було використано набір даних перевірки для постійної оцінки моделі, також є окремий набір даних тестування. Підготуємо набір даних для тестування.

```
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "../input/alzheimers-dataset-4-class-of-images/Alzheimer_s Dataset/test",
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)
```

Рисунок 3.42 - Підготування тестових даних

```
Found 1279 files belonging to 4 classes.
80/80 [==============================] - 1s 19ms/step - loss: 5.1833 - auc: 0.7936
```

Рисунок 3.43 - Результат перевірки тестовими даними

Параметр loss дорівнює 5.08, що є забагато для тестового набору даних. Параметр AUC дорівнює 0.8, є задовільним для тестового набору даних.

Спробуємо на основі отриманої моделі отримати прогноз по одній МРТ, яка приналежить класу NonDementia.

```
This image most likely belongs to VeryMildDementia with a 27.34 percent confidence.

Process finished with exit code 0
```

Рисунок 3.44 - Результат прогнозу ZFNet

Модель вірно прогнозувала клас МРТ, а також має великий процент впевненості

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«ЗАСТОСУВАННЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ДІАГНОСТИКИ ХВОРОБИ АЛЬЦГЕЙМЕРА»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21810106

Виконав студент 4-го курсу, групи 401

_____

(підпис, ініціали та прізвище)

«___» _____ 2022 р.

Консультант_____

(наук. ступінь, вчене звання)

_____

(підпис, ініціали та прізвище)

«___» _____ 2022 р.

Миколаїв – 2022

Вступ

Охорона праці - це система законодавчих, організаційно-технічних, соціально-економічних, санітарно-гігієнічних і лікувально-профілактичних мір і засобів, спрямованих на збереження життя, здоров'я й працездатності людини в процесі праці. Завдання охорони праці полягає в тому, щоб звести до мінімуму ймовірність поразки працюючого під дією небезпечного виробничого фактора або захворювання під дією шкідливого виробничого фактора з одночасним забезпеченням комфортних умов при максимальній продуктивності праці. Закон України "Про охорону праці" визначає основні положення по реалізації конституційного права громадян на охорону їх життя і здоров'я в процесі трудової діяльності; регулює взаємини між адміністрацією і працівником в незалежності від форм власності; встановлює єдиний порядок організації охорони праці в Україні [19].

Завданням законодавства про охорону навколишнього природного середовища є регулювання відносин у галузі охорони, використання і відтворення природних ресурсів, забезпечення екологічної безпеки, запобігання і ліквідації негативного впливу господарської та іншої діяльності на навколишнє природне середовище, збереження природних ресурсів, генетичного фонду живої природи, ландшафтів та інших природних комплексів, унікальних територій та природних об'єктів, пов'язаних з історико-культурною спадщиною [20]. Згідно закону України «Про підприємства в Україні» усі роботодавці повинні турбуватись про дотримання у своєї діяльності вимог законів України стосовно охорони праці та навколишнього природного середовища.

У даній дипломній роботі питання охорони праці розглядаються стосовно робочого приміщення, де виконується безпосередньо робота за напрямом диплому та за умовами праці які визначені завданням.

4.1 Міжнародні та державні норми з питань безпеки праці користувачів інформаційних технологій

Вивчаючи цю тему, необхідно пам’ятати, що державна політика України в галузі безпеки праці спрямована на створення безпечних і здорових умов праці, запобігання нещасним випадкам та професійним захворюванням. Вона базується на ряді принципів, основними з яких є пріоритет життя і здоров’я працівників, повна відповідальність роботодавця за створення безпечних та належних умов праці, підвищення рівня промислової безпеки, комплексне розв’язання завдань з охорони праці, соціальний захист працівників, повне відшкодування шкоди особам, які потерпіли від нещасних випадків на виробниіцтві та професійних захворювань.

Україна є членом Міжнародної організації праці (МОП). Вона ратифікувала 63 конвенції МОП, із них 14 – за роки незалежності. Положення цих конвенцій лягли в основу чинного в Україні законодавства, що регулює соціально-трудові відносини.

Законодавство Євросоюзу у сфері охорони праці можна умовно розділити на дві групи:

– директиви ЄС щодо захисту працівників;

– директиви ЄС щодо випуску товарів на ринок (включаючи обладнання, устаткування, машини, засоби колективного та індивідуального захисту, які використовують працівники на робочому місці). [20]

Необхідно вивчити міжнародну систему охорони безпеки та здоров’я, основою якої є міжнародний стандарт OHSAS 18001 «Система менеджменту охорони здоров’я та безпеки персоналу» будується на принципі добровільного дотримання вимог охорони праці, виходячи із встановленого законодавством допустимого рівня ризику для життя і здоров’я працівників. Міжнародний стандарт OHSAS так само, як і система управління якістю ISO 9000 і система управління

охороною навколишнього середовища ISO 14000, побудований на основі циклу Демінга. Вивчаючи цю тему, треба знати вимоги, що встановлює підхід OHSAS.

Вимоги про те, якою має бути ергономіка в країнах ЄС, закріплено в законодавстві Союзу й, насамперед, у директиві № 89/391 «Про заходи щодо поліпшення безпеки й здоров'я трудящих». Відповідно до неї роботодавці зобов'язані оцінювати виробничі ризики й забезпечувати адекватні захисні і профілактичні заходи, гарантувати відповідне навчання й інструктаж працівників з дотримання заходів безпеки, а також надавати працівникам інформацію та консультації і дозволяти їм брати участь в обговоренні всіх питань із забезпечення безпеки й гігієни праці. [21]

Розробкою загальних єдиних нормативних документів для користувачів інформаційних технологій займаються декілька міжнародних організацій:

– International Organization for Standardization (1S0) – міжнародна організація зі стандартизації;

– Ergonomics committee ISO (TC 159) – комітет з ергономіки міжнародної організації зі стандартизації;

– European Standardization Organization (CEN) – європейська організація зі стандартизації.

Серед низки розроблених нормативних документів з використовування ВДТ найбільш часто застосовуються такі стандарти:

– IS0 9241-3, який регламентує ергономічні вимоги за умовами праці і охорони здоров'я користувачів;

– IS0 9001, який визначає якість і рівень виробництва апаратури;

– IS0 DIN 9995, який встановлює принципи розміщення елементів клавіатури для роботи з текстом в офісних системах;

– IEC 950, який визначає норми безпеки електротехнічного устаткування.

Треба мати на увазі, що основним нормативним документом України із забезпечення охорони праці користувачів комп'ютерно-інформаційних технологій

є НПАОП 0.00-1.28-2010 [21] «Правила охорони праці при експлуатації електронно-обчислювальних машин».

4.2 Загальна характеристика приміщення та робочого місця

Робоче приміщення, в якому проводяться дослідження та випробування за завданням наведені у таблиці 4.1.

Таблиця 4.1 – Загальна характеристика умов праці

Шкідливі та небезпечні фактори на робочому місці	Джерела утворювання небезпек	Примітка (данні наведені для технічного відділу)
Електрична напруга вище 127 В; Шум; Випромінювання – електромагнітні, радіаційні, теплові; Статична електрика; Іонізація повітря; Пожежна небезпека у приміщенні; Не якісне освітлення.	Кондиціонер, 2-ПЕОМ, Папір, Світильники (лампи)	Розміри приміщення (м) : Довжина – 10; Ширина – 5; Висота – 3. Кількість працюючих – 1

Згідно з НПАОП 0.00-1.28-2010 [21] в лабораторії може перебувати 6 працівників. Мінімальна припустима площа приміщення на 1 людину повинна складати не менш 6,0 м2. Висновок - за умовами завдання це виконується повністю. В приміщенні відсутні умови, які можуть створювати підвищену або особливо підвищену небезпеку, тому воно відноситься до класу звичайних приміщень (згідно ПУЕ[22]). Джерелом живлення є трифазна мережа напруги 380/220 В з

глухо заземленою нейтралю, з частотою 50 Гц (згідно НПАОП 0.00-1.28-2010 [21]). За пожеже вибухонебезпекою приміщення лабораторії відноситься до класу В. У таблиці 4.1 наведена загальна характеристика приміщення щодо вибухопожежної небезпеки та за важкістю робіт.

Таблиця 4.2 - Загальна характеристика приміщення щодо вибухопожежної

Характеристика приміщень за вибухопожежною категорією та класом зони	Загальна характеристика приміщення	Категорія за важкістю робіт згідно **ГН 3.3.5-8.6.6.1-2002**
В – пожеже небезпечна, Клас П-II	Звичайне, без ознак хімічного забруднення та нормальної вологості за санітарними вимогами	1а ……..до 139 Вт/м^2 1б ……..140-174 Вт/м^2 Клас умов праці - Оптимальний Окремі показники напруженості трудового процесу – ступінь ризику для власного життя – виключено; ступінь відповідальності за безпеку інших осіб – виключено. Ступінь відповідальності за результат своєї діяльності. Значущість помилки - допустимий: (напруженість праці середнього ступеня), а саме – несе відповідальність за функціональну якість допоміжних

		завдань. Вимагає додаткових зусиль з боку керівництва (керівника дипломної роботи); спостереження за екраном відео терміналу (годин на зміну) 2-3.

4.3 Метеорологічні параметри робочої зони

Під час роботи з ПЕОМ необхідно дотримувати оптимальні метеорологічні умови. Оптимальні метеорологічні умови - сполучення параметрів, які при тривалому й систематичному впливі на людину забезпечують збереження нормального функціонального й теплового стану організму без напруження реакцій терморегуляції. Параметри мікроклімату в приміщенні повинні відповідати ГН 3.3.5-8-6.6.1-2002 [23]. Із урахуванням категорії роботи за енерговитратами повинні дотримуватися параметри мікроклімату, наведені в табл. 4.3

Таблиця 4.3 - Оптимальні параметри мікроклімату

Категорія робіт	Період року	Температура, °С	Відносна вологість, %	Швидкість руху повітря, м/с
Легка (Іб)	холодний	21-23	40-60	не більше 0,1
Легка (Іб)	теплий	22-24	40-60	не більше 0,2

Для підтримки в приміщенні оптимального температурного режиму відповідно до вимоги ДБН В.2.5-67:2013 [24] є централізоване опалювання і вентиляція. У теплий період року використовується кондиціювання.

4.4. Освітлення

Особливістю роботи за дисплеєм ЕОМ є постійна й значна напруга функцій зорового аналізатора, обумовленого необхідністю розходження самосвітних об'єктів (символів, знаків і т.п.) при наявності відблисків на екрані, рядковій структурі екрана, мерехтіння зображення, недостатньою чіткістю об'єктів розходження. Для забезпечення нормального освітлення застосовуються природне бокове одностороннє й штучне освітлення, які нормуються ДБН В.2.5-28-2006 [25] та НПАОП 0.00-1.28-2010 [21]. По характеру зорової роботи, робота відноситься до високої точності, розряд зорової роботи III, підрозряд г. Раціональне освітлення приміщення сприяє кращому виконанню виробничого завдання і забезпеченню комфорту при роботі. Для забезпечення нормального освітлення застосовуються природне, однобічне, бічне і штучне освітлення, а також сполучене, які нормуються санітарними нормами й правилами ДБН В.2.5-28-2006 [25]. Дані по нормах освітлення наведені в табл. 4.4.

Таблиця 4.4 - Норми природного й штучного освітлення

Мінімальний розмір об'єкта розрізнювання, мм	Фон	Контраст	Розряд, під розряд зорової праці	Природне освітлення КПО, %	Штучне освітлення	
					Емін, лк	Емін, лк
Від 0,3 до 0,5	Світлий	Середній	III г	1,5	300	Тип ламп

Приміщення з постійним перебуванням людей повинно мати, як правило, природне освітлення. При виконанні роботи використовувалося природне одностороннє бокове й штучне освітлення. Нормативне значення КПО повинно бути не менш 1,5% при роботі з ПЕОМ, тому потрібно застосовувати штучне освітлення (згідно ДБН В.2.5-28-2006 [25]).

4.5 Шум та вібрація у робочому приміщенні

У приміщенні технічного відділу причинної шуму і вібрації являються апарати, прилади і устаткування: друкуючі пристрої, комп'ютери, вентилятори, кондиціонер та ін. При їхній роботі рівень вібрації не вище 33 дБ, рівень шуму не повинен перевищувати 50 дБА, що є нормою для даного виду діяльності відповідно до НПАОП 0.00-1.28-2010 [21]. Заходи по забезпечення встановлених норм: використання спеціальних шум-поглинаючих перегородок, застосування меблів, які сприяють зменшенню шуму і вібрації, установка апаратів і приладів на спеціальні амортизуючи підкладки.

4.6 Електробезпека

Для живлення устаткування (ПЕОМ, освітлювальні прилади) які є однофазними споживачами використовується трифазна мережа 380/220В частотою 50Гц з глухо заземленої нейтралі. Із цієї причини при роботі з електроприладами існує потенційна небезпека ураження людини електричним струмом, тому в правилах устрою електроустановок (згідно ПУЕ [22]) передбачені наступні заходи електробезпеки: конструктивні, схемно-конструктивні й експлуатаційні. Конструктивні - вимоги що забезпечують захист від доторкання персоналу до струмоведучих частин. ПЕОМ мають ступінь захисту IP44. Прилади освітлення IP-23. Схемно-конструктивним заходом захисту є занулення електрообладнання у приміщенні. Для користувача ПЕОМ важливим є дотримання правил безпеки експлуатації електрообладнання. Так, заборонено доторкатися до дротів та з’єднань при наявності напруги в мережі, а також самостійно проводити ремонт електрообладнання. Усі питання щодо ремонту налагодження та інше, можуть виконувати тільки електрики та відповідні фахівці, які мають допуск до роботи із електрообладнанням певної категорії.

4.7 Електробезпека

Робоче місце оператора ЕОМ обладнується робочим столом, кріслом і підставкою для ніг. Висота робочого стола регулюється в межах 0,68—0,80 м, а при відсутності такої можливості має складати 0,72 м. Мінімальна ширина стола 0,6 м, поверхня стола не блискуча. Робоче крісло оператора забезпечується підіймально-поворотним пристроєм з регулюванням висоти сидіння та спинки. Розміри підставки для ніг довжина 0,4 м, ширина не менше 0,30 м. На одного працюючого з урахуванням роботи з ПЕОМ має відводитись не менше 6,0 м2 та не менше 20 м2 об’єму приміщення згідно НПАОП 0.00-1.28-2010 [21].

Висновки

Під час виконання спеціальної частини з охорони праці було досліджено умови праці людини на підприємствах та установах, а також роботу при користуванні електронними пристроями.

Правила, які прописані у нормативних документах є актуальними на сьогодні через те, що майже в кожному підприємстві є працівники, які працюють з електронними пристроями. Під час роботи за персональним комп'ютером людина швидко втомлюється та від цього страждає опорно-руховий, зоровий апарат, нервова система и в окремих випадках психічне здоров'я.

На сьогодні існує багато шкідливих умов, які погіршують процес трудової діяльності людини, такі як: шум, вібрація та інші умови. Було проаналізовано умови праці на робочому місці або у виробничому приміщенні.

Вимоги та нормативи, щодо охорони праці мають дотримуватися на підприємствах під час трудової діяльності людини. Дотримання поставлених вимог до працівників та власників підприємств дозволить мінімізувати шкідливі наслідки, які мають вплив на здоров'я людини.

ВИСНОВКИ

Під час написання кваліфікаційної роботи проведено аналіз використання методів штучного інтелекту в світі. Виявлено поширення та популярність нейронних мереж, як одного із методів машинного навчання. Досліджено принципи роботи нейромережевих технологій. Проаналізовано становлення згорткових нейронних мереж. Досліджено архітектури таких нейромережевих моделей: AlexNet, LeNet, ZFNet. Налаштовано 3 нейронних мереж використовуючи мову програмування Python для діагностування хвороби Альцгеймера. Засвоєно також налаштування бібліотек штучного інтелекту для роботи з цими моделями. Проведено аналіз та порівняння різних мереж таким характеристикам: loss та auc.

Отже згідно з поставленим завданням та метою виконано усі пункти. Створено реалізацію 3 нейронних мереж по моделям: AlexNet, LeNet, ZFNet. З нуля налаштовано роботу з різними бібліотеками. Було налаштовано датасет для оптимальної роботи з кожною моделлю. Друга з трьох створених моделей мають достатню впевненість для прогнозування класу, тому їх можливо використовувати з достатньою довірою до результатів.

Створені системи можливо інтегрувати в будь-який робочій термінал. А завдяки цьому можливо надати експертам програмне забезпечення для прогнозування захворювання за допомогою МРТ. Це допоможе робити раннє діагностування хвороби, і як результат пришвидшити діагностування хвороби Альцгеймера.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google's ANN service. - Vertex AI: веб-сайт. URL: https://cloud.google.com/vertex-ai/docs/matching-engine/ann-service-overview (дата звернення: 01.05.2022).

2. COPD identification and grading based on deep learning of lung parenchyma and bronchial wall in chest CT images. - Researchgate: веб-сайт. URL: https://www.researchgate.net/publication/358513255_COPD_identification_and_grading_based_on_deep_learning_of_lung_parenchyma_and_bronchial_wall_in_chest_CT_images (дата звернення: 02.05.2022).

3. Reproducible and interpretable deep learning for the diagnosis, prognosis and subtyping of Alzheimer's disease from neuroimaging data – Wikipedia: веб-сайт. URL: https://en.wikipedia.org/wiki/Comparison_of_code_generation_tools (дата звернення: 03.05.2022).

4. Computer-Aided Diagnosis of Dementia based on structural MRI data. – Grand Challenge: веб-сайт. URL: https://caddementia.grand-challenge.org/ (дата звернення: 04.05.2022).

5. Deep Convolutional Neural Networks for Automated Diagnosis of Alzheimer's Disease and Mild Cognitive Impairment Using 3D Brain MRI – CodeSmith Researchgate: веб-сайт. URL: https://www.researchgate.net/figure/Deep-CNN-architecture-used-for-the-proposed-Alzheimers-Disease-diagnosis-framework_fig5_332212620 (дата звернення: 06.05.2022).

6. Convolutional neural network – Wikipedia: веб-сайт. URL: https://en.wikipedia.org/wiki/Convolutional_neural_network#:~:text=A%20CNN%20architecture%20is%20formed,of%20layers%20are%20commonly%20used. (дата звернення: 09.05.2022).

7. Activation Functions in Neural Networks – Towards Data Science: веб-сайт. URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (дата звернення: 10.05.2022).

8. Нейронні мережі - шлях до глибинного навчання – Codeguida: веб-сайт. URL: https://codeguida.com/post/739 (дата звернення: 20.05.2022).

9. Функція втрат – Wikipedia: веб-сайт. URL: https://en.wikipedia.org/wiki/Loss_function (дата звернення: 22.05.2022).

10. Validation – Wikipedia: веб-сайт. URL: https://en.wikipedia.org/wiki/Validation (дата звернення: 25.05.2022).

11. Dilution (neural networks) – Wikipedia: веб-сайт. URL: https://en.wikipedia.org/wiki/Dilution_(neural_networks) (дата звернення: 27.05.2022).

12. Трансферне навчання – Wikipedia: веб-сайт. URL: https://uk.upwiki.one/wiki/Transfer_learning (дата звернення: 01.06.2022).

13. Метод вилучення – Wikipedia: веб-сайт. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B2%D0%B8%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%BD%D1%8F (дата звернення: 05.06.2022).

14. ImageNet – Wikipedia: веб-сайт. URL: https://en.wikipedia.org/wiki/ImageNet (дата звернення: 06.06.2022).

15. Python – Python: веб-сайт. URL: https://www.python.org/ (дата звернення: 06.06.2022).

16. Introduction to The Architecture of Alexnet – Analytics Vidhya: веб-сайт. URL: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/ (дата звернення: 08.06.2022).

17. The Architecture of Lenet-5 – Analytics Vidhya: веб-сайт. URL: https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/ (дата звернення: 08.06.2022).

18. ZFNet – Papers With Code: веб-сайт. URL: https://en.wikipedia.org/wiki/ImageNet (дата звернення: 06.06.2022).

19. Закон України “Про охорону праці” / Законодавство України про охорону праці. - К. Нова редакція 2002 р.

20. Закон України “Про охорону навколишнього природного середовища” – К.: Україна. – 1991. - 59 с. (з усіма редакціями до 2017 року)

21. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин/ Зареєстровано в Міністерстві юстиції України 19 квітня 2010 р. за N 293/17588

22. Правила улаштування электроустановок. ПУЕ.– Харків.: Форт – 2011 – 728 с.

23. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу. Гігієнічні нормативи ГН 3.3.5-8-6.6.1 2002 р. Видання офіційне Київ, 2001 рік – 46 с.

24. ДБН В.2.5-67:2013. Опалення, вентиляція та кондиціювання . -К.: Мінрегіон України, 2013.-147 с

25. ДБН.В.2.5 – 28-2006 . Природне і штучне освітлення. – К.: Мінбуд України, - 2008 – 74 с.

ДОДАТОК А

Код для моделі AlexNet

```
AUTOTUNE = tf.data.experimental.AUTOTUNE
BATCH_SIZE = 16 * strategy.num_replicas_in_sync
IMAGE_SIZE = [176, 208]
EPOCHS = 100
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
   "input/Alzheimer_s Dataset/train",
   validation_split=0.2,
   subset="training",
   seed=1337,
   image_size=IMAGE_SIZE,
   batch_size=BATCH_SIZE,
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
   "input/Alzheimer_s Dataset/train",
   validation_split=0.2,
   subset="validation",
   seed=1337,
   image_size=IMAGE_SIZE,
   batch_size=BATCH_SIZE,
)
class_names = ['MildDementia', 'ModerateDementia', 'NonDementia',
'VeryMildDementia']
train_ds.class_names = class_names
val_ds.class_names = class_names
NUM_CLASSES = len(class_names)
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
```

```
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(train_ds.class_names[labels[i]])
        plt.axis("off")
plt.show()
def one_hot_label(image, label):
    label = tf.one_hot(label, NUM_CLASSES)
    return image, label
train_ds = train_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
val_ds = val_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
NUM_IMAGES = []
for label in class_names:
    dir_name = "input/Alzheimer_s Dataset/train/" + label[:-2] + 'ed'
    NUM_IMAGES.append(len([name for name in os.listdir(dir_name)]))
def build_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(filters=96, kernel_size=(11, 11),
                            strides=(4, 4), activation="relu",
                            input_shape=(*IMAGE_SIZE, 3)))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)))
    model.add(layers.Conv2D(filters=256, kernel_size=(5, 5),
                            strides=(1, 1), activation="relu",
                            padding="same"))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)))
```

```
    model.add(layers.Conv2D(filters=384, kernel_size=(3, 3),
                            strides=(1, 1), activation="relu",
                            padding="same"))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(filters=384, kernel_size=(3, 3),
                            strides=(1, 1), activation="relu",
                            padding="same"))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(filters=256, kernel_size=(3, 3),
                            strides=(1, 1), activation="relu",
                            padding="same"))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation="relu"))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(4, activation="softmax"))
    model.compile(loss='sparse_categorical_crossentropy',
                  optimizer=tf.optimizers.SGD(lr=0.001),
                  metrics=['accuracy'])
    model.summary()
    return model
with strategy.scope():
    model = build_model()
    METRICS = [tf.keras.metrics.AUC(name='auc')]
    model.compile(
      optimizer='adam',
      loss=tf.losses.CategoricalCrossentropy(),
      metrics=METRICS
```

```
    )
def exponential_decay(lr0, s):
    def exponential_decay_fn(epoch):
        return lr0 * 0.1 ** (epoch / s)
    return exponential_decay_fn
exponential_decay_fn = exponential_decay(0.01, 20)
lr_scheduler = tf.keras.callbacks.LearningRateScheduler(exponential_decay_fn)
checkpoint_cb = tf.keras.callbacks.ModelCheckpoint("LeNet.h5",
                                                   save_best_only=True)
early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10,
                                                     restore_best_weights=True)
history = model.fit(
    train_ds,
    validation_data=val_ds,
    callbacks=[checkpoint_cb, early_stopping_cb, lr_scheduler],
    epochs=EPOCHS
)
fig, ax = plt.subplots(1, 2, figsize=(20, 3))
ax = ax.ravel()
for i, met in enumerate(['auc', 'loss']):
    ax[i].plot(history.history[met])
    ax[i].plot(history.history['val_' + met])
    ax[i].set_title('Model {}'.format(met))
    ax[i].set_xlabel('epochs')
    ax[i].set_ylabel(met)
    ax[i].legend(['train', 'val'])
plt.show()
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "input/Alzheimer_s Dataset/test",
```

```
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)
image = tf.keras.preprocessing.image.load_img("input/Alzheimer_s
Dataset/train/MildDemented/mildDem0.jpg")
input_arr = tf.keras.preprocessing.image.img_to_array(image)
input_arr = np.array([input_arr])
input_arr = input_arr.astype('float32') / 255.

test_ds = test_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)

_ = model.evaluate(test_ds)
savedModel = keras.models.load_model("LeNet.h5")
predictions = savedModel.predict(input_arr)
score = tf.nn.softmax(predictions)
savedModel.summary()
print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
        .format(class_names[np.argmax(score)], 100 * np.max(score))
```

ДОДАТОК Б

Код для моделі LeNet

```
AUTOTUNE = tf.data.experimental.AUTOTUNE
BATCH_SIZE = 16 * strategy.num_replicas_in_sync
IMAGE_SIZE = [176, 208]
EPOCHS = 100
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
   "input/Alzheimer_s Dataset/train",
   validation_split=0.2,
   subset="training",
   seed=1337,
   image_size=IMAGE_SIZE,
   batch_size=BATCH_SIZE,
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
   "input/Alzheimer_s Dataset/train",
   validation_split=0.2,
   subset="validation",
   seed=1337,
   image_size=IMAGE_SIZE,
   batch_size=BATCH_SIZE,
)
class_names = ['MildDementia', 'ModerateDementia', 'NonDementia',
'VeryMildDementia']
train_ds.class_names = class_names
val_ds.class_names = class_names
NUM_CLASSES = len(class_names)
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
```

```
        for i in range(9):
            ax = plt.subplot(3, 3, i + 1)
            plt.imshow(images[i].numpy().astype("uint8"))
            plt.title(train_ds.class_names[labels[i]])
            plt.axis("off")
    plt.show()
    def one_hot_label(image, label):
        label = tf.one_hot(label, NUM_CLASSES)
        return image, label
    train_ds = train_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
    val_ds = val_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
    train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
    val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
    NUM_IMAGES = []
    for label in class_names:
        dir_name = "input/Alzheimer_s Dataset/train/" + label[:-2] + 'ed'
        NUM_IMAGES.append(len([name for name in os.listdir(dir_name)]))
    def build_model():
    model = tf.keras.Sequential()
        model.add(layers.Conv2D(filters=6, kernel_size=(3, 3), activation='relu',
input_shape=(*IMAGE_SIZE, 3)))
        model.add(layers.AveragePooling2D())
        model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
        model.add(layers.AveragePooling2D())
        model.add(layers.Flatten())
        model.add(layers.Dense(units=120, activation='relu'))
        model.add(layers.Dense(units=84, activation='relu'))
        model.add(layers.Dense(units=4, activation='softmax'))
        model.summary()
```

```
with strategy.scope():
    model = build_model()
    METRICS = [tf.keras.metrics.AUC(name='auc')]
    model.compile(
        optimizer='adam',
        loss=tf.losses.CategoricalCrossentropy(),
        metrics=METRICS
    )
def exponential_decay(lr0, s):
    def exponential_decay_fn(epoch):
        return lr0 * 0.1 ** (epoch / s)
    return exponential_decay_fn
exponential_decay_fn = exponential_decay(0.01, 20)
lr_scheduler = tf.keras.callbacks.LearningRateScheduler(exponential_decay_fn)
checkpoint_cb = tf.keras.callbacks.ModelCheckpoint("LeNet.h5",
                                                   save_best_only=True)
early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10,
                                                     restore_best_weights=True)
history = model.fit(
    train_ds,
    validation_data=val_ds,
    callbacks=[checkpoint_cb, early_stopping_cb, lr_scheduler],
    epochs=EPOCHS
)
fig, ax = plt.subplots(1, 2, figsize=(20, 3))
ax = ax.ravel()
for i, met in enumerate(['auc', 'loss']):
    ax[i].plot(history.history[met])
    ax[i].plot(history.history['val_' + met])
```

```
        ax[i].set_title('Model {}'.format(met))
        ax[i].set_xlabel('epochs')
        ax[i].set_ylabel(met)
        ax[i].legend(['train', 'val'])
    plt.show()
    test_ds = tf.keras.preprocessing.image_dataset_from_directory(
        "input/Alzheimer_s Dataset/test",
        image_size=IMAGE_SIZE,
        batch_size=BATCH_SIZE,
    )
    image = tf.keras.preprocessing.image.load_img("input/Alzheimer_s Dataset/train/MildDemented/mildDem0.jpg")
    input_arr = tf.keras.preprocessing.image.img_to_array(image)
    input_arr = np.array([input_arr])
    input_arr = input_arr.astype('float32') / 255.

    test_ds = test_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
    test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)

    _ = model.evaluate(test_ds)
    savedModel = keras.models.load_model("LeNet.h5")
    predictions = savedModel.predict(input_arr)
    score = tf.nn.softmax(predictions)
    savedModel.summary()
    print(
        "This image most likely belongs to {} with a {:.2f} percent confidence."
            .format(class_names[np.argmax(score)], 100 * np.max(score))
```

ДОДАТОК В

Код для моделі ZFNet

```
AUTOTUNE = tf.data.experimental.AUTOTUNE
BATCH_SIZE = 16 * strategy.num_replicas_in_sync
IMAGE_SIZE = [176, 208]
EPOCHS = 100
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "input/Alzheimer_s Dataset/train",
    validation_split=0.2,
    subset="training",
    seed=1337,
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "input/Alzheimer_s Dataset/train",
    validation_split=0.2,
    subset="validation",
    seed=1337,
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)
class_names = ['MildDementia', 'ModerateDementia', 'NonDementia',
'VeryMildDementia']
NUM_CLASSES = len(class_names)
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
```

```
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(train_ds.class_names[labels[i]])
        plt.axis("off")
plt.show()
def one_hot_label(image, label):
    label = tf.one_hot(label, NUM_CLASSES)
    return image, label
train_ds = train_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
val_ds = val_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
NUM_IMAGES = []
for label in class_names:
    dir_name = "input/Alzheimer_s Dataset/train/" + label[:-2] + 'ed'
    NUM_IMAGES.append(len([name for name in os.listdir(dir_name)]))
def build_model():
model = tf.keras.models.Sequential([
                tf.keras.layers.Conv2D(96, (7, 7), strides=(2, 2), activation='relu',
                        input_shape=(224, 224, 3)),
                tf.keras.layers.MaxPooling2D(3, strides=2),
    tf.keras.layers.Lambda(lambda x: tf.image.per_image_standardization(x)),
                tf.keras.layers.Conv2D(256, (5, 5), strides=(2, 2), activation='relu'),
                tf.keras.layers.MaxPooling2D(3, strides=2),
    tf.keras.layers.Lambda(lambda x: tf.image.per_image_standardization(x)),
                tf.keras.layers.Conv2D(384, (3, 3), activation='relu'),
                tf.keras.layers.Conv2D(384, (3, 3), activation='relu'),

                tf.keras.layers.Conv2D(256, (3, 3), activation='relu'),
                tf.keras.layers.MaxPooling2D(3, strides=2),
```

```
    tf.keras.layers.Flatten(),
                tf.keras.layers.Dense(4096),
                tf.keras.layers.Dense(4096),
                tf.keras.layers.Dense(10, activation='softmax')
        ])with strategy.scope():
    model = build_model()
    METRICS = [tf.keras.metrics.AUC(name='auc')]
    model.compile(
        optimizer='adam',
        loss=tf.losses.CategoricalCrossentropy(),
        metrics=METRICS
    )
def exponential_decay(lr0, s):
    def exponential_decay_fn(epoch):
        return lr0 * 0.1 ** (epoch / s)
    return exponential_decay_fn
exponential_decay_fn = exponential_decay(0.01, 20)
lr_scheduler = tf.keras.callbacks.LearningRateScheduler(exponential_decay_fn)
early_stopping_cb = tf.keras.callbacks.EarlyStopping(patience=10,
                                        restore_best_weights=True)
history = model.fit(
    train_ds,
    validation_data=val_ds,
    callbacks=[checkpoint_cb, early_stopping_cb, lr_scheduler],
    epochs=EPOCHS
)
fig, ax = plt.subplots(1, 2, figsize=(20, 3))
ax = ax.ravel()
for i, met in enumerate(['auc', 'loss']):
```

```
    ax[i].plot(history.history[met])
    ax[i].plot(history.history['val_' + met])
    ax[i].set_title('Model {}'.format(met))
    ax[i].set_xlabel('epochs')
    ax[i].set_ylabel(met)
    ax[i].legend(['train', 'val'])
plt.show()
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "input/Alzheimer_s Dataset/test",
    image_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
)
image = tf.keras.preprocessing.image.load_img("input/Alzheimer_s
Dataset/train/MildDemented/mildDem0.jpg")
input_arr = tf.keras.preprocessing.image.img_to_array(image)
input_arr = np.array([input_arr])
input_arr = input_arr.astype('float32') / 255.

test_ds = test_ds.map(one_hot_label, num_parallel_calls=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)

_ = model.evaluate(test_ds)
savedModel = keras.models.load_model("LeNet.h5")
predictions = savedModel.predict(input_arr)
score = tf.nn.softmax(predictions)
savedModel.summary()
print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
        .format(class_names[np.argmax(score)], 100 * np.max(score))
```