НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Наталія АУШЕВА
«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра
спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Система класифікації знімків комп'ютерної томографії з використанням нейронних мереж**»**

Виконав :
студент IV курсу, групи ТР-82
Журавльов Олег Володимирович _____

Керівник:
Старший викладач
Мірошниченко Іван Володимирович. _____

Консультант з нейронних мереж:
Доцент
Залевська Ольга Валеріївна _____

Рецензент:
к.т.н., доцент
Фіногенов Олексій Дмитрович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший

спеціальність 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
”___”_____2022р.

ЗАВДАННЯ
на дипломну роботу студенту

Журавльову Олегу Володимировичу
(прізвище, ім’я, по батькові)

1. Тема роботи : «Система класифікації знімків комп’ютерної томографії з використанням нейронних мереж»
керівник роботи Старший викладач Мирошниченко І.В.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__” травня 2022р. № _____

2. Строк подання студентом роботи 10 червня 2022 р.

3. Вихідні дані до роботи: мова програмування Python , середовище PyCharm.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд літературних джерел за темою дослідження. Ознайомлення з медичними форматами даних та приведення їх до відповідного формату. Розробка алгоритму та архітектури нейронної мережі для бінарної класифікації та реалізація її навчання. Застосування та аналізи розробленого програмного продукту для обробки знімків медичних зображень з використанням алгоритмів навчання з вчителем та без нейронних мереж. Проведення порівняльної характеристики Підготовка готової моделі до використання у зазначених цілях

5. Перелік ілюстративного матеріалу:
1) Актуальність; 2) Мета; 3) Основні задачі 4) Збір даних 5) Підготовка даних 6) Навчання моделі 7) Засоби програмної реалізації 8) Інтерфейс 9)Архітектура проекту 10) Діаграма прецедентів 11) Алгоритм роботи системи 12) Інсталяція та системні вимоги; 13) Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1,2	Доцент Залевська Ольга Валеріївна		

7. Дата видачі завдання 10 вересня 2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	25.05.2022	
2.	Вивчення та аналіз задачі	02.03.2022-08.03.2022	
3.	Розробка архітектури та загальної структури системи	09.03.2022-15.03.2022	
4.	Розробка структур окремих підсистем	20.03.2022-29.03.2022	
5.	Програмна реалізація системи	01.04.2022-16.04.2022	
6.	Оформлення пояснювальної записки	05.05.2022-19.05.2022	
7.	Захист програмного продукту	20.05.2022	
8.	Передзахист	06.06.2022-09.06.2022р	
9.	Захист	10.06.2022	

Студент _____ Журавльов О.В.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Мирошниченко І.В.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Обсяг роботи становить 102 сторінки, 34 ілюстрації, 9 таблиць, 13 джерел літератури.

Метою програмного продукту є бінарна класифікація медичних зображень з використанням нейронної мережі.

Програмний продукт створений для використання в медичних закладах, та для зниження навантаження на лікарів ,які займаються діагностуванням запалення легень.

Вхідною інформацією є результати комп'ютерної томографії ,які можуть бути представлені в різному виді та форматі.

Вихідною інформацією є класифікація кожного знімку, та встановлення таких знімків, що потребують додаткового огляду лікарем для встановлення кінцевого діагнозу.

Пояснювальна записка містить аркушів, ілюстрацій, додатків та використаних джерел.

ANNOTATION

The volume of work is 102 pages, 34 illustrations, 9 tables, 13 sources of literature.

The purpose of the software product is binary classification of medical images using a neural network.

The software product is designed for use in medical institutions, and to reduce the burden on doctors who diagnose pneumonia.

The input information is computed tomography images, which can be presented in different types and formats.

The source information is the diagnosis for each image, and the overall diagnosis.

The explanatory note contains sheets, illustrations, appendices and sources use

Зміст

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ – програмне забезпечення;

DNN- (deep neural network)- глибока нейронна мережа

CNN, або ConvNet – (Convolutional neural network) - Згорткова нейронна мережа

КТ- комп'ютерна томографія

AI- (artificial intelligence) - штучний інтелект

DL-(Deep learning)- Глибинне навчання

COVID-19- (CoronaVirus Disease 2019) - Коронавірусна хвороба

DICOM (Digital Imaging and Communications in Medicine) — медичний стандарт зберігання зображень

ВСТУП

Коронавірусна хвороба 2019 (COVID-19) є надзвичайно швидко розповсюджується та має серйозними і глобальними наслідками для здоров'я. Станом на 26 квітня 2022 року в усьому світі було зареєстровано 510 мільйони підтверджених захворювань, що забрали понад 6.2 мільйона життів[1] . Основною перешкодою в боротьбі з COVID-19 є доступність своєчасного обстеження та моніторингових тестів.

Комп'ютерна томографія (КТ) зазвичай використовується в клінічній практиці для діагностики, скринінгу та лікування COVID-19 у всьому світі. Рентгенологи мусять аналізувати велику кількість знімків, що в свою чергу гальмує своєчасний діагноз, та лікування. Крім того, у багатьох слаборозвинених сільських регіонах обмежений доступ до добре підготовлених рентгенологів з достатнім знанням у сфері діагностування COVID-19. У сукупності вони вимагають рішень на основі штучного інтелекту (AI). Серед методологій (AI), мережі глибокого навчання (DL) набули значної популярності в порівнянні з традиційними методами машинного навчання . На відміну від методів машинного навчання, усі етапи виділення ознак, вибору та класифікації в моделях DL виконуються автоматично.

У цій роботі рохзглянуто застосування методів DL для діагностики COVID-19 та автоматизованого аналізу КТ-зображення.

В ході роботи будуть вирішені такі задачі:

- Обробка вхідних даних та зведення їх до одного формату
- Розробка архітектури нейронної мережі для бінарної класифікації та її навчання
- Підготовка готової моделі до використання у зазначених цілях

1 ПОСТАНОВКА ЗАДАЧІ

Дослідити ефективність застосування нейронних мереж у цій задачі, та надати експертам (медичним чи іншим) і технікам нове уявлення, про те, як методи глибокого навчання використовуються у цьому відношенні та як вони потенційно працюють у боротьбі зі спалахом COVID-19. Написати програмне забезпечення, яке має виконувати функцію доступного та функціонального інструменту для аналізу знімків комп'ютерної томографії.

На вході система отримує файл або декілька файлів, які містять в собі зображення сканів. Далі іде стадія попередньої обробки даних, де дається можливість оптимізувати данні, для подальшого аналізу.

На виході система має надати такі результати:

- Оброблені знімки
- Діагноз по кожному знімку
- Загальний діагноз пацієнта (якщо за один раз обробляються декілька пацієнтів, то загальний діагноз слід ігнорувати)
- Можливість довчити наявну модель, на знімках діагноз яких вірний
- Готова модель

Потенційним користувачем ПО є лікар рентгенолог.

2 ТЕХНІКА ГЛИБОГО НАВЧАННЯ ДЛЯ ДІАГНОСТУВАННЯ COVID-19

У системі класифікації головною метою є ідентифікація хворих на COVID-19, що передбачає процес вилучення ознаки, вибір об’єктів та класифікація за допомогою глибоких шарів. Загалом, системи на основі глибокого навчання складаються з кількох етапів, таких як збір даних, підготовка даних, виділення та класифікація ознак, а також оцінка продуктивності. Загальний конвеєр системи діагностики COVID-19 на основі глибокого навчання показано на рисунку 2.1.

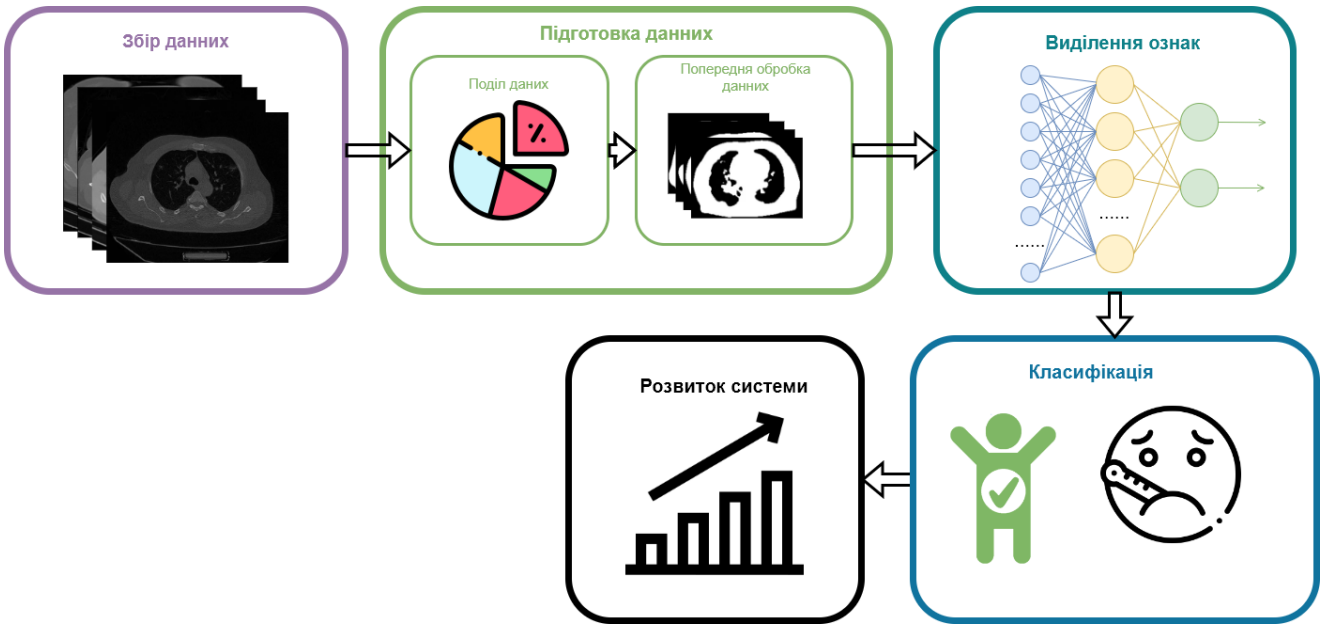

Рисунок 2.1 - Загальний конвеєр системи діагностики COVID-19 на основі глибокого навчання.

Етап збору даних полягає в знаходженні та об’єднанні даних з різних джерел. Крок розподілу даних розбиває дані на набори для навчання, перевірки та тестування . Наступним кроком є підготовка даних, яка перетворює дані у відповідний формат. На цьому кроці попередня обробка даних включає деякі операції, такі як видалення шуму, зміна розміру, збільшення тощо. Основним етапом діагностики COVID-19 на основі глибокого навчання є виділення та класифікація ознак. На цьому етапі техніка глибокого навчання автоматично

витягує функцію, виконуючи кілька операцій повторно, і, нарешті, класифікація здійснюється на основі міток класу (здоровий або хворий). Нарешті, розроблена система готова для навчання.

2.1 Збір даних

Загалом вдалося зібрати 67592 знімків з хворими легенями, та 36683 зі здоровими.

Цей набір даних[2] складається з КТ грудної клітки без покращення від 1000+ пацієнтів з підтвердженими інфекціями COVID-19. Розподіл за віком пацієнтів, яким було проведено КТ, становив 47,18 ± 16,32 (середнє ± стандартне відхилення) років, а віковий діапазон – від 6 до 89 років. Гендерний розподіл становив 60,9% чоловіків і 39,1% жінок. Найпоширенішими супутніми хворобами, серед пацієнтів були гіпертонія або ішемічна хвороба серця, цукровий діабет та інтерстиціальна пневмонія або емфізема. Зображення були отримані в період з березня 2020 року по січень 2021 року.

Набори даних[3,4,5] включає в себе знімки хворих раком легень, та здорових людей. У цій роботі будуть використовуватись тільки данні здорових людей.

2.2 Підготовка даних

Цей процес забезпечує контекст, у якому ми можемо розглянути підготовку даних, необхідну для проекту, на основі визначення проекту, виконаного до підготовки даних.

2.2.1 Поділ даних

Поділ даних включає в себе не тільки виділення наборів даних для навчання, перевірки та тестування, а й відкидання непотрібних чи не підходящих даних (усі набори даних містять знімки усього тулуба людини, а аналізу підлягають лише легені, тож все що вище, нижче або має не підходящий фільтр, відкидалося).

2.2.2 Попередня обробка даних

Щоб розуміти як проводилася обробка даних розглянемо структуру DICOM файлу.

Файл, що зберігає одне зображення в стандарті DICOM є складною структурою даних, що включає не тільки безпосередньо зображень, але й супутню інформацію, таку як: дані про обладнання, на якому проводилося дослідження; опис проведеного дослідження; параметри та опис серії; системи координат пов'язаної із зображенням; атрибути, що визначають саме зображення; текстово-графічні елементи, графіки та коментарі, що виконуються медичним персоналом та атрибути, що описують перетворення над отриманими даними тощо.

Окремий DICOM файл містить як заголовок (який зберігає інформацію про ім'я пацієнта, тип дослідження, розмір зображення тощо), так і всі дані зображення (які можуть містити інформацію у трьох вимірах).

Для більшості МРТ та КТ використовується фотометрична інтерпретація монохромна (наприклад, зазвичай зображена пікселями у сірій шкалі). У DICOM цим монохромним зображенням зіставляють фотометричну інтерпретацію 'MONOCHROME1' (малі темні значення, великі значення dim) або 'MONOCHROME2' (низькі значення темні, великі значення яскраві). Однак, ціла низка медичних зображень (УЗД, позитронно-емісійні томограми) містять колір, тому вони описуються різними фотометричними інтерпретаціями (наприклад,

палітра, RGB, CMYK, YBR тощо). Деякі кольорові зображення (наприклад, RGB) зберігають 3 кольори в кожному пікселі (червоний, зелений та синій), тоді як монохромні та палітрові зображення зазвичай зберігають лише один зразок на зображення. Кожне зображення зберігає 8 біт (256 рівнів) або 16 бітів (65 535 рівнів), хоча деякі сканери зберігають дані в 12 або 32 бітному роздільній здатності. Так що RGB зображення, що зберігає 3 зразки в пікселі в 8 бітах, може потенційно описати 16 мільйонів кольорів (256^3).

Також слід зауважити, що для роботи з DICOM файлами була використана стороння бібліотека, за допомогою якої всі знімки були відсортовані по глибині зрізу та переведенні в піксельні данні. Після чого всі знімки були перетворені в відео, що ще раз допомагає зменшити розмір файлів на диску. Початковий розмір файлів з усіх датасетів становив 150 гігабайт, після усіх маніпуляцій він зменшився до 500 мегабайт. Щоб використовувати дані надалі файли відео будуть зчитуватися покадрово.

Наступними кроком обробки були застосування декількох методів по уніфікації та оптимізації даних спрямованих на зменшення шумів та позбавлення від зайвих деталей:

- Знімки були зменшені в розмірі з 512× 512 пікселів до 128 ×128 пікселів.
- Були обрізані нижні 20 пікселів зображення. Це рішення обґрунтоване тим що там знаходиться стіл аналіз якого не має сенсу.
- Для позбавлення від шумів, зернистості та сторонніх деталей було застосоване розмивання Гауса.
- Також для сегментації був використаний поріг. Тобто якщо значення пікселя менше за поріг, воно встановлюється на 0, в іншому випадку встановлюється максимальне значення 255.

Порівняння знімків до(рисунок 2.2) обробки та після(рисунок 2.1):

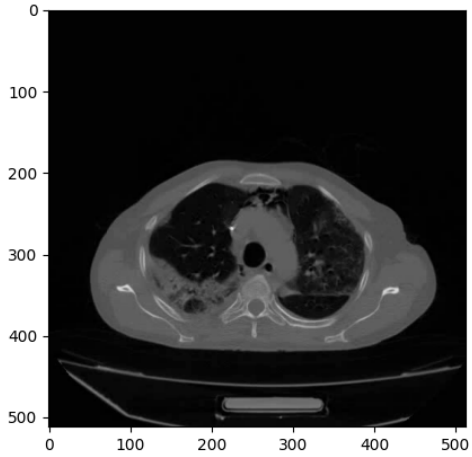

Рисунок 2.2 - Початкове зображення

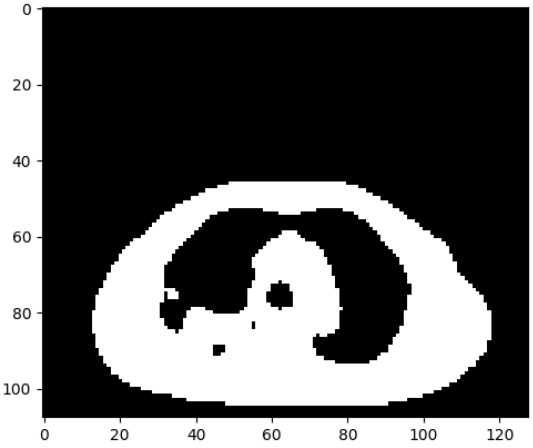

Рисунок 2.3 - Готове зображення

2.3 Навчання моделі

Загальна архітектура нейронної мережі складається з двох згорткових шарів, двох агрегувальних шарів та повнозвязної мережі.

На вхід згорткового шару до кожного нейрона поступає частина матриці зображення(розмір частини визначається ядром згортки). Після проходження через згортковий шар зображення абстрагується до карти об’єктів. Суть описаної системи в тому, що група нейронів активується, тільки тоді коли, на фрагменті зображення з'являється об'єкт, який підходить під їх ядра [6].

Агрегувальний шар спрямований на виділення загальних деталей на зображенні.

Повністю пов’язані шари з’єднують кожен нейрон в одному шарі з кожним нейроном в іншому шарі.

При навчанні нейронної мережі були використані 30000 знімків здорових легень, та 50000 хворих взяті з декількох датасетів.

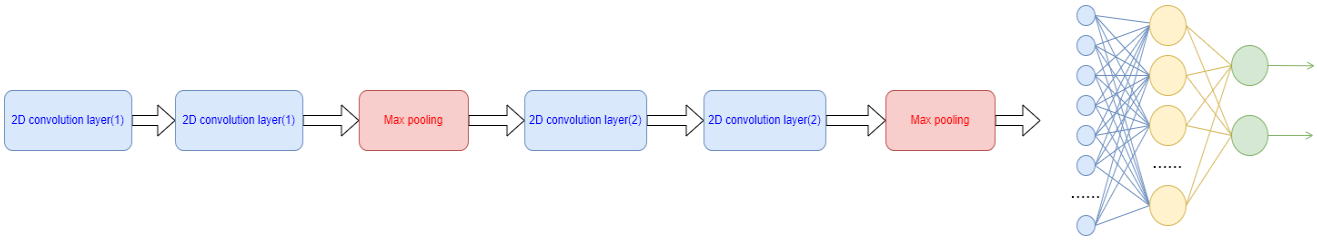

Рисунок 2.4 - Архітектура 2D-CNN, яка використовується для виявлення COVID-19

Детальні параметри нейронної мережі:

- Перші згорткові шари – кількість ядер 32, розмір ядра 3×3, функція активації ReLU
- Другі згорткові шари – кількість ядер 64, розмір ядра 3×3, функція активації ReLU
- Агрегувальний шар(max pooling) – розмір вікна для якого потрібно взяти максимальне значення 2× 2
- Повноз'єднаний шар містить 128 нейронів
- Оптимізатор – Adam
- Функція втрат – Бінарна крос-ентропія
- Метрика – Бінарна точність
- Кількість епох – 3

Усього в нейронній мережі 7,143,266 зв'язків.

Таблиця 2.1 - Кількість зв’язків нейронної мережі

Назва шару	Кількість зв’язків
Conv2D (1)	320
Conv2D (1)	9248
MaxPooling2D(1)	0
Conv2D(2)	18496
Conv2D(2)	36928

MaxPooling2D(2)	0
Dense(1)	7078016
Dense(2)	258

Якщо дивитися на точність при навчанні, то вона буде близькою до ідеальної, що не відповідає дійсності [8,9]. Таку високу точність при валідації можна пояснити тим, що для неї використовувались знімки сусідніх зрізів, тобто дуже схожі на ті на яких тренували нейронну мережу. Якщо передбачити діагноз для знімків, які ніяк не схожі на навчальну вибірку, то точність буде трохи нижчою: для здорових 99.4% ,для хворих 90.67%.

Висновки до розділу

У даному розділі була розглянута загальну структуру, та спосіб навчання згорткових нейронних мереж, показані детальні методи попередньої обробки інформації та детальна архітектура використаної нейронної мережі.

3 ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Мова програмування

Для проекту було вибрано мову програмування Python. Цей вибір обґрунтовано тим, що це найпопулярніша та найзручніша мова для роботи нейронними мережами. Також для Python існує багато сторонніх бібліотек модулів та фреймворків які роблять цю мову дуже універсальною.

3.2 Головні модулі

У даному розділі будуть розглянуті головні стороні пакети для Python, які використовувались в цій роботі.

3.2.1 opencv-python

OpenCV (Open Source Computer Vision Library) — це бібліотека функцій програмування, в основному спрямованих на комп'ютерний зір у реальному часі .

OpenCV написаний на C++ , а його основний інтерфейс — на C++, але він все ще зберігає менш повний, хоча й розширений інтерфейс старого C. Всі нові розробки та алгоритми з'являються в інтерфейсі C++. Існують прив'язки в Python, тож в даній бібліотеці ми працюємо лише з API.

3.2.2 ffmpeg

FFmpeg — набір вільних бібліотек з відкритим вихідним кодом, які дозволяють записувати, конвертувати та передавати цифрові аудіо- та відеозаписи

у різних форматах. Він включає libavcodec - бібліотеку кодування та декодування аудіо та відео, і libavformat - бібліотеку мультиплексування та демультиплексування в медіаконтейнер[11].

3.2.3 ffmpeg-python

API для роботи з ffmpeg в Python цей модуль виділяється тим шо він вміє працювати з фільтрами, а також дозволяє створювати канали по яким можна швидко перетворити відео файл в масив даних [12].

3.2.4 numpy

NumPy це open-source модуль для python, який надає загальні математичні та числові операції у вигляді пре-компільованих, швидких функцій. Вони поєднуються у високо рівневі пакети. Вони забезпечують функціонал, який можна порівняти із функціоналом MatLab. NumPy (Numeric Python) надає базові методи для маніпуляції з великими масивами та матрицями.

3.2.5 pydicom

Pynetdicom — це чистий пакет Python 3.7+, який реалізує мережевий протокол DICOM. Працюючи з pydicom, він дозволяє легко створювати користувачів класу обслуговування DICOM і постачальників класу послуг.

Як чистий пакет Python, pydicom може працювати скрізь, де працює Python.

3.2.6 PyQt5

Qt — це набір крос-платформних бібліотек C++, які реалізують високорівневі API для доступу до багатьох аспектів сучасних настільних і

мобільних систем.

PyQt5 — це повний набір прив’язок Python для Qt v5. Він реалізований у вигляді більш ніж 35 модулів розширення і дозволяє використовувати Python як альтернативну мову розробки програм C++ на всіх підтримуваних платформах.

PyQt5 також може бути вбудований в програми на основі C++, щоб дозволити користувачам цих програм налаштовувати або покращувати функціональність цих програм[13].

3.2.7 Scikit-learn

Scikit-learn — це бібліотека машинного навчання з відкритим вихідним кодом, яка підтримує контрольоване та неконтрольоване навчання. Він також надає різні інструменти для підгонки моделі, попередньої обробки даних, вибору моделі, оцінки моделі та багатьох інших утиліт.

Scikit-learn в основному написаний на Python і широко використовує NumPy для високопродуктивної лінійної алгебри та операцій з масивами. Крім того, деякі основні алгоритми написані на Cython для підвищення продуктивності. Машини опорних векторів реалізовані за допомогою обгортки Cython навколо LIBSVM; логістичної регресії та лінійної опорної векторної машини за аналогічною обгорткою навколо LIBLINEAR. У таких випадках розширення цих методів за допомогою Python може бути неможливим.

Scikit-learn добре інтегрується з багатьма іншими бібліотеками Python, такими як Matplotlib і plotly для побудови графіків, NumPy для векторизації масивів, Pandas, SciPy та багатьма іншими.

3.2.8 Matplotlib

Matplotlib — бібліотека мовою програмування Python для візуалізації даних двовимірною (2D) графікою (3D графіка також підтримується). Отримані

зображення можуть бути використані як ілюстрації в публікаціях.

Matplotlib є гнучким, легко конфігурованим пакетом, який разом з NumPy, SciPy та IPython надає можливості, подібні до MATLAB. В даний час пакет працює з декількома графічними бібліотеками, включаючи Windows і PyGTK.

3.2.9 Tensorflow

TensorFlow це наскрізна платформа з відкритим вихідним кодом для машинного навчання. Він має комплексну та гнучку екосистему інструментів, бібліотек та ресурсів спільноти, яка дозволяє дослідникам впроваджувати найсучасніші технології машинного навчання, а розробникам легко створювати та розгортати програми на основі машинного навчання.

Його гнучка архітектура дозволяє легко розгортати обчислення на різних платформах (CPU, GPU, TPU) і від настільних комп'ютерів до кластерів серверів до мобільних і граничних пристроїв. Обчислення TensorFlow виражаються у вигляді графіків потоку даних із збереженням стану .

3.2.10 Keras

Keras — це API, розроблений для людей, а не для машин. Keras дотримується найкращих практик щодо зниження когнітивного навантаження: він пропонує послідовні та прості API, мінімізує кількість дій користувача, необхідних для поширених випадків використання, а також надає чіткі й ефективні повідомлення про помилки. Він також містить велику документацію та посібники для розробників.

Висновки до розділу

У цьому розділі були розглянуті основні засоби програмної реалізації проекту. Тут представлені лише основні пакти, переглянути список всіх пакетів зі вказаними версіями можна в файлі GUI/ requirements.txt. Щоб встановити всі пакети зі списку потрібно виконати команду pip install -r GUI/requirements.txt.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Інтерфейс користувача

Загальна структура проекту

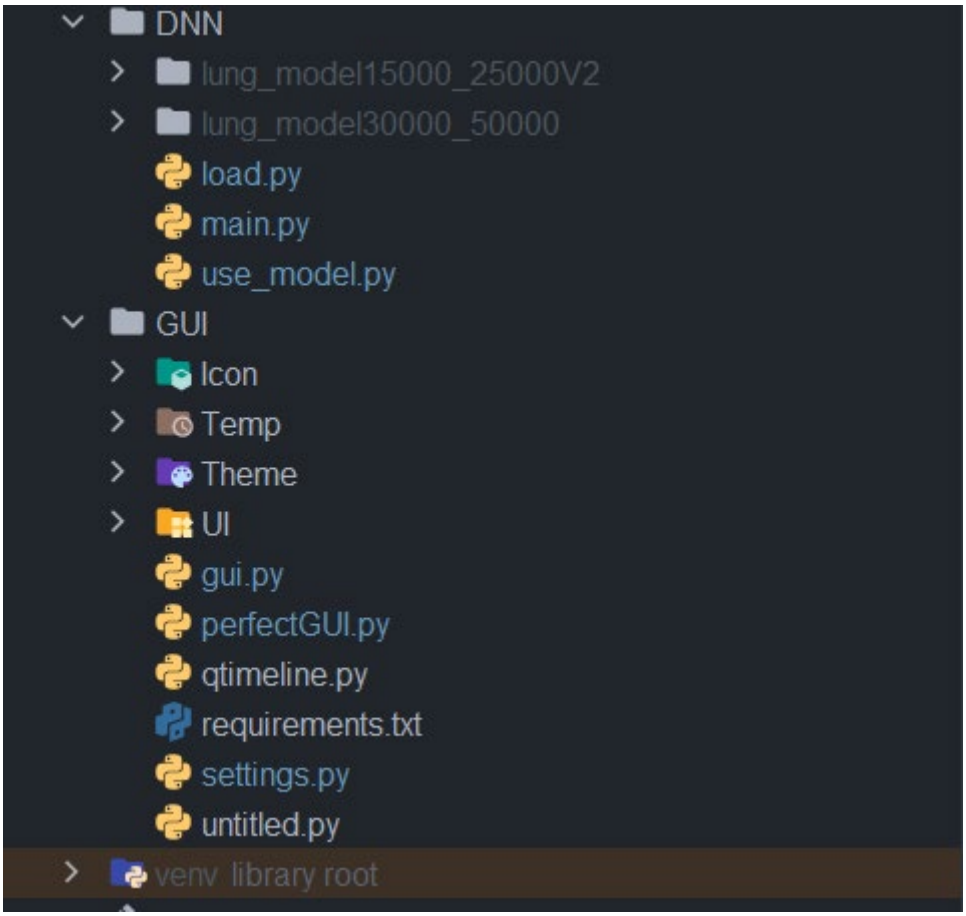

Рисунок 4.1 - Структура проекту

4.1.1 Загальна інформація про інтерфейс

Інтерфейс прогами створений за допомогою Qt Designer, який постачається разом з пакетом pyqt5-tools, і представляє собою головне вікно з центральним віджетом QStackedWidget, та допоміжне вікно налаштувань. Також була використана таблиця стилів яка зберігається в файлі ElegantDark.qss.

4.1.2 Опис програмної частини

Вихідними файлами з Qt Designer є GUI/UI/ perfectGUI.ui (головне вікно), та GUI/UI/ settings.ui (вікно налаштувань). Ці файли представляють собою XML таблиці з описом усіх віджетів (рисунок 4.1).

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>CT_main_win</class>
 <widget class="QMainWindow" name="CT_main_win">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>890</width>
    <height>669</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>CT analyzer</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <property name="acceptDrops">
    <bool>false</bool>
   </property>
   <layout class="QGridLayout" name="gridLayout_2">
    <item row="0" column="0">
     <widget class="QStackedWidget" name="stackedWidget">
      <property name="enabled">
       <bool>true</bool>
      </property>
      <property name="layoutDirection">
       <enum>Qt::LeftToRight</enum>
      </property>
      <property name="frameShape">
       <enum>QFrame::NoFrame</enum>
      </property>
```

Рисунок 4.2 - Приклад файлу

Дані файли були перетворені в Python код, командою -x [filename].ui -o [filename].py, так були створенні GUI/perfectGUI.py рисунок 4.3 та GUI/ settings.py. Таке рішення пояснюється тим, що парсинг ui файлу займає багато часу.

```
class Ui_CT_main_win(object):
    def setupUi(self, CT_main_win):
        CT_main_win.setObjectName("CT_main_win")
        CT_main_win.resize(1280, 720)
        CT_main_win.setWindowIcon(QtGui.QIcon('Icon/PngItem_320012.png'))
        self.mediaPlayer = QMediaPlayer(None, QMediaPlayer.VideoSurface)
        videoWidget = QVideoWidget()
        self.mediaPlayer.setVideoOutput(videoWidget)
        self.centralwidget = QtWidgets.QWidget(CT_main_win)
        self.centralwidget.setAcceptDrops(False)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout_2 = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.stackedWidget = QtWidgets.QStackedWidget(self.centralwidget)
        self.stackedWidget.setEnabled(True)
        self.stackedWidget.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.stackedWidget.setFrameShape(QtWidgets.QFrame.NoFrame)
        self.stackedWidget.setObjectName("stackedWidget")
        self.select = QtWidgets.QWidget()
        self.select.setObjectName("select")
        self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.select)
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setSizeConstraint(QtWidgets.QLayout.SetDefaultConstraint)
        self.verticalLayout_2.setContentsMargins(-1, -1, -1, 20)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_2.addItem(spacerItem)
        self.label = QtWidgets.QLabel(self.select)
        font = QtGui.QFont()
        font.setPointSize(21)
        self.label.setFont(font)
        self.label.setAlignment(QtCore.Qt.AlignBottom | QtCore.Qt.AlignHCenter)
```

Рисунок 4.3 - Вміст GUI/perfectGUI.py

Надалі ці файли змінюватись не будуть, а використовувати їх та їхні об’єкти буть у головному файлі gui.py. Для цього використовується метод setupUi (рисунок 4.4).

```
class mywindow(QtWidgets.QMainWindow):

    def __init__(self):
        super(mywindow, self).__init__()
        self.ui = Ui_CT_main_win()
        self.ui.setupUi(self)
```

Рисунок 4.4 - Виклик головного вікна

Був підвантажений та використаний файл qss (рисунок 4.5).

```
339     File = open("Theme/ElegantDark.qss", 'r')
340     with File:
341         qss = File.read()
342         app.setStyleSheet(qss)
```

Рисунок 4.5 - Застосування таблиці стилів

4.1.3 Приклад інтерфейсу

Приклад головного вікна рисунок 4.6

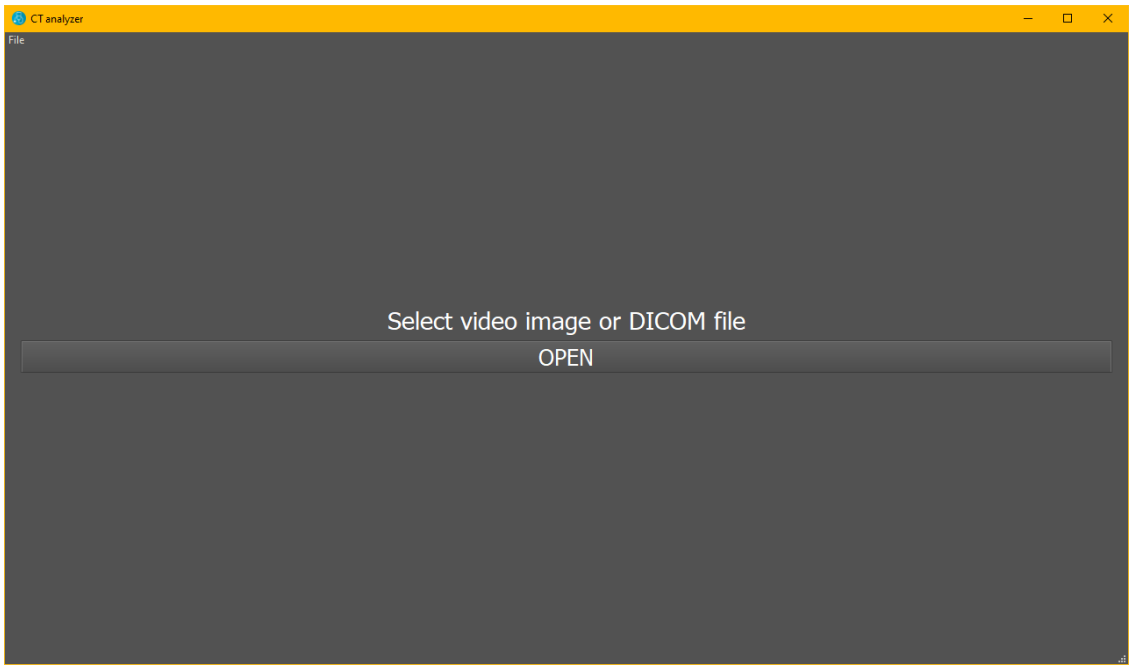

Рисунок 4.6 - Головне вікно

Приклад вікна налаштувань рисунок 4.7

Рисунок 4.7 - Вікно налаштувань

4.2 Реалізація підготовки даних

Ціллю цього процесу є позбавлення від шумів та уніфікація даних.

Дані можуть поступати в різних форматах ,та різних розмірах тож якщо данні надаються в окремих файлах , то вони спочатку будуть склеєні в одне відео, яке буде записано в GUI/Temp/ comp_video.mp4.

```
out = cv2.VideoWriter(r'Temp\comp_video.mp4', cv2.VideoWriter_fourcc(*'mp4v'), 10, (512, 512))
for i in self.fileName:
    name=os.path.basename(i)
    n, e = os.path.splitext(name)
    if e==".jpg" or e==".png" or e==".PNG"or e==".JPG":
        img = cv2.imdecode(np.fromfile(i, dtype=np.uint8), cv2.IMREAD_COLOR)
        img = cv2.resize(img, (512,512))
        out.write(img)
    if e==".mp4":
        cap = cv2.VideoCapture(i)
        r = True
        while r:
            r, img = cap.read()
            if r:
                img = cv2.resize(img, (512, 512))
                out.write(img)
            else:
                break
        cap.release()
    if e==".dcm":
        frames.append(pydicom.dcmread(i))
slices = []
for f in frames:
    if hasattr(f, 'SliceLocation'):
        slices.append(f)
slices = sorted(slices, key=lambda s: s.SliceLocation)
image_2d_scaled = []
for s in slices:
    img = (s.pixel_array.astype(float))
    image_2d_scaled.append((np.maximum(img, 0) / img.max()) * 255.0)
image_2d_scaled = np.uint8(image_2d_scaled)
for i in image_2d_scaled :
    out.write(i)

out.release()
```

Рисунок 4.8 - Код відповідальний за об'єднання даних

Для подальшого використання інформації з відео її потрібно перетворити в багатомірний масив.

```
        out, _ = (
            ffmpeg
                .input(self.fileName)
                .trim(start_frame=int(cut_start / 100 * self.frame_count),
                      end_frame=int(cut_end / 100 * self.frame_count))
                .setpts('PTS-STARTPTS')
                .filter('scale', 128, 128)
                .output('pipe:', format='rawvideo', pix_fmt='rgb24')
                .run(capture_stdout=True)
        )
        self.video = (
            np
                .frombuffer(out, np.uint8)
                .reshape([-1, 128, 128, 3])
        )

```

Рисунок 4.9 - Перетворення в масив

Вихідний масив має таку форму (кількість кадрів, висота, ширина, 3).

Перед аналізом знімків кожен з них проходить процедури описані в розділі 2.2.2, тобто відрізається нижній поріг, накладається розмивання Гауса та встановлюється поріг яскравості.

```
        for i in self.video:
            im_g = i[:-20, :]
            im_g = cv2.cvtColor(im_g, cv2.COLOR_RGB2GRAY)

            im_g = cv2.GaussianBlur(im_g, (
                int(self.ui_settings.doubleSpinBox_Blur.value()), int(self.ui_settings.doubleSpinBox_Blur.value())), 5,
                                    cv2.BORDER_DEFAULT)

            ret, thresh1 = cv2.threshold(im_g, int(self.ui_settings.spinBox_thresholdValue.value()), 255,
                                         cv2.THRESH_BINARY)
```

Рисунок 4.10 - Попередня обробка даних

4.3 Реалізація нейронної мережі

Нейрона мережа навчається окремо від головного проекту та міститься в папці DNN . Тут міститься файл самої нейронної мережі main.py файл в якому реалізована допоміжна функція підгруздки даних з диску – load.py та файл який використовує нейронну мережу. Також є 2 навчені нейронні мережі: lung_model15000_25000V2 – це нейронна мережа навчена на 40000

зображеннях(15000 здорових та 25000 хворих) формою 216×256, lung_model30000_50000 – це мережа яка навчалася на 80000 зображень (30000 здорових та 50000 хворих) формою 108×128 .

Досліди на цих мережах показали, що нейронна мережа точніша при розмірі зображень 216×256, але це твердження вірне лише при однаковому розмірі навчальних вибірок. За рахунок в двічі більшої навчальної вибірки lung_model30000_50000 показує більшу точність. Слід знати що навчання нейронної мережі виконувалось на комп'ютері з наступною конфігурацією:

- Процесор Ryzen 5 3600
- Об'єм ОЗУ 16 гігабайт
- Відеокарта Nvidia 1650 super 4 Гб

Саме обсяг пам'яті є вузьким місцем через яке неможливо навчати нейрону мережу на 80000 зображень формою 216×256. Якщо обсяг ОЗУ збільшити, то можливе перенавчання мережі на більшій вибірці.

Слід зазначити, що навчання мережі виконувалось на процесорі, а при використанні мережі в головному проекті(тобто в графічному інтерфейсі) використовується відеокарта, яка теж має своє обмеження по пам'яті. Для даної відеокарти це обмеження становить приблизно 12000 знімків.

4.4 Архітектура проекту

Загальна архітектура проекту складається з трьох папок:

- Convert містить всі скрити які були використані для конвертації
- DNN містить навчені нейроні мережі, та скрипт для навчання
- GUI містить в собі всі файли які пов'язані з користувацьким інтерфейсом

Розглянимо діаграму класів :

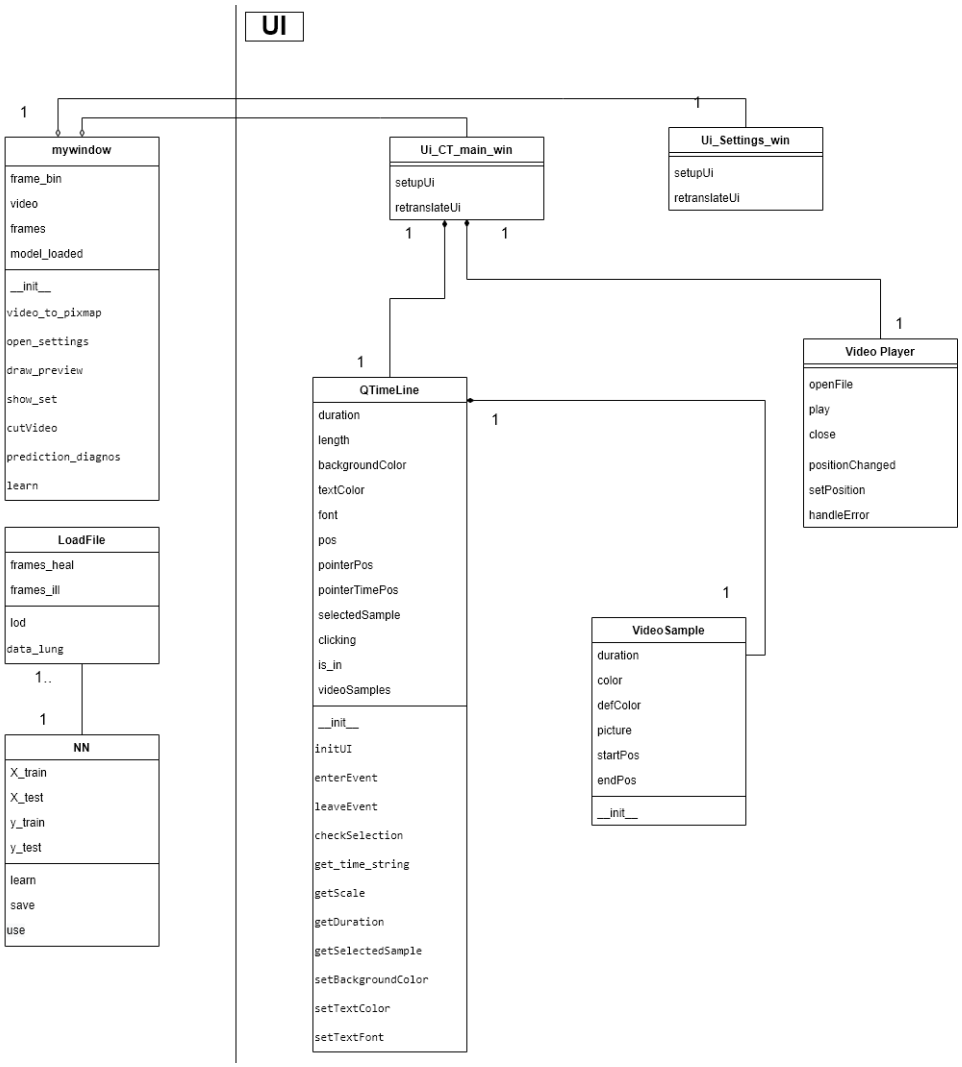

Рисунок 4.11 - Діаграма класів

Як можна побачити з діаграми головними класами є mywindow та NN. В конструкторі класу mywindow визивається метод setupUi класу Ui_CT_main_win після чого робота буде вестися з об’єктом класу .

```
22      def __init__(self):
23          super(mywindow, self).__init__()
24          self.ui = Ui_CT_main_win()
25          self.ui.setupUi(self)
```

Рисунок 4.12 - Створення об’єкту класу

Слід зауважити що клас Ui_Settings_win не являється дочірнім класом Ui_CT_main_win або mywindow, його об’єкт створюється у методі open_settings.

4.5 Опис класів

4.5.1 Опис класу QTimeLine

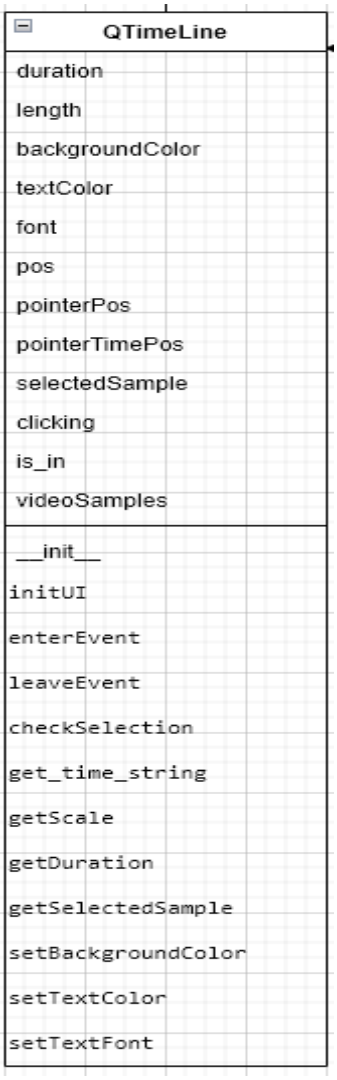

Рисунок 4.13 - Клас QTimeLine

Клас QTimeLine створений для реалізації віджету шкали часу

Таблиця 4.2 - Опис методів класу QTimeLine

Сигнатура	Вхідні	Повернене значення	Призначен

	параметри		ня
def __init__(self, duration, length):	duration, length	-	Конструктор класу
def initUI(self):		-	Ініціалізація
def paintEvent(self, event):	event	-	Намалювати віджет
def mouseMoveEvent(self, e):	e	-	Відстежити рух миші
def enterEvent(self, e):	e	-	Подія входження курсору миші на територію віджету
def leaveEvent(self, e):	e	-	Подія виходу курсору миші на територію віджету
def checkSelection(self, x):	x	-	Подія вибору сегменту
def get_time_string(self,	seconds	"%02d:%02d:%02d" % (h, m, s)	Отримати рядок часу

seconds):			з секунд
def getScale(self):	-	float(self.duration)/float(self.width())	Отримайте масштаб за довжиною
def getDuration(self):	-	self.duration	Отримати тривалість
def getSelectedSample(self):	-	self.selectedSample	Отримати вибраний зразок
def setBackgroundColor(self, color):	color	-	Встановити колір фону
def setTextColor(self, color):	color	-	Встановити колір тексту
def setTextFont(self, font):	font	-	Встановити шрифт

4.5.2 Опис класу VideoSample

Рисунок 4.14 - Клас VideoSample

Даний клас призначений для зберігання зображень попереднього перегляду відео.

Таблиця 4.2 - Опис методів класу VideoSample

Сигнатура	Вхідні параметри	Повернене значення	Призначення
def __init__(self, duration, color=Qt.darkYellow, picture=None, audio=None):	duration, color picture audio	-	Конструктор класу

4.5.3 Опис класу Video Player

Рисунок 4.15 - Клас Video Player

Цей клас є стандартним класом медіа плеєру.

Таблиця 4.3 - Опис методів класу Video Player

Сигнатура	Вхідні параметри	Повернене	Призначення

		значення	
def openFile(self):	-	-	Відкрити файл
def play(self):	-	-	Запустити відео
def positionChanged(self, position):	position	-	Вибір часу на слайдері
def setPosition(self, position):	position	-	Пересунути повзунок з плином часу
def handleError(self):	-	-	Обробка помилок

4.5.4 Опис класу Ui_Settings_win

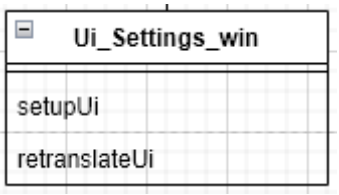

Рисунок 4.16 - Клас Ui_Settings_win

Цей клас містить в собі всі об’єкти головного вікна.

Таблиця 4.4 - Опис методів класу Ui_Settings_win

Сигнатура	Вхідні параметри	Повернене значення	Призначення
def setupUi(*self*,	Settings_win	-	Ініціалізація

Settings_win):			об’єктів вікна
def retranslateUi(*self*, Settings_win):	Settings_win	-	Реалізації логіки багатомовної підтримки

4.5.5 Опис класу Ui_CT_main_win

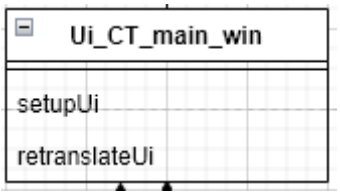

Рисунок 4.17 - Клас Ui_CT_main_win

Даний клас був створений для зберігання об’єктів вікна налаштувань.

Таблиця 4.5 - Опис методів класу Ui_CT_main_win

Сигнатура	Вхідні параметри	Повернене значення	Призначення
def setupUi(self, CT_main_win):	CT_main_win	-	Ініціалізація об’єктів вікна
def retranslateUi(self, CT_main_win):	CT_main_win	-	Реалізації логіки багатомовної підтримки

4.5.6 Опис класу mywindow

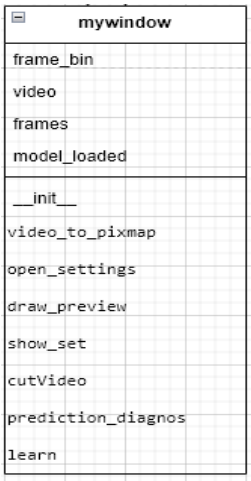

Рисунок 4.18 - Клас mywindow

Цей клас є головним, також він реалізує в собі попередню обробку інформації, аналіз зображень та навчання мережі.

Таблиця 4.6 - Опис методів класу mywindow

Сигнатура	Вхідні параметри	Повернене значення	Призначення
def __init__(self):	-	-	Конструктор класу
def video_to_pixmap(self, fileName):	fileName	-	Намалювати зображення для попереднього перегляду
def open_settings(self, show=True):	show	-	Ініціалізація вікна налаштувань
def draw_preview(self):	-	-	Намалювати зображення для

			попереднього перегляду в вікні налаштувань
def show_set(self, win):	win	-	Показати вікно налаштувань
def cutVideo(self):	-	-	Обрізати відео та підготовити для обробки
def prediction_diagnos(self, frame):	frame	diagnos	Передбачити діагноз для зображень
def learn(self):	-	-	Довчити наявну модель

4.5.7 Опис класу NN

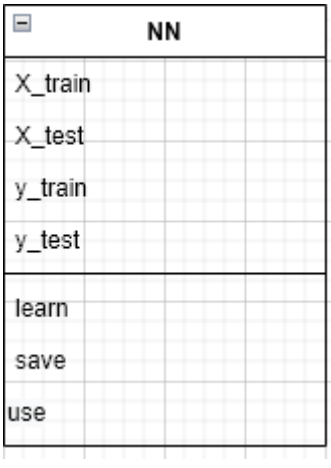

Рисунок 4.19 - Клас NN

Цей клас відповідає за створення нейронної мережі, надає можливість її збереження та приклад використання.

Таблиця 4.7 - Опис методів класу NN

Сигнатура	Вхідні параметри	Повернене значення	Призначення
def learn (self)	-	-	Почати навчання моделі
def save(self, name)	name	-	Зберегти моделі
def use(self, name)	name	-	Приклад використання моделі

4.5.8 Опис класу LoadFile

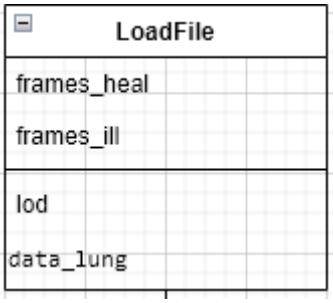

Рисунок 4.20 - Клас LoadFile

Клас LoadFile відповідальний за читання даних з диску та структурування.

Таблиця 4.8 - Опис методів класу LoadFile

Сигнатура	Вхідні параметри	Повернене значення	Призначення
def lod(self, h_name, i_name)	h_name, i_name	frames_heal,frames_ill	Перетворити відео в багатомірні

			масиви
def data_lung(self, frames=40000,testS=0.1)	frames, testS	X_train, X_test, y_train, y_test	Структурувати дані

4.6 Діаграма прецедентів

Діаграма варіантів використання показує різні варіанти використання та різні типи користувачів. Варіанти використання представлені або колами, або еліпсами. Актори часто зображені у вигляді фігурок.

У той час як сам варіант використання може детально розглянути кожну можливість, діаграма прецедентів може допомогти представити систему на більш високому рівні.

Дана діаграма прецедентів демонструє основні варіанти роботи програми. Користувачу надається можливість вибрати оди або декілька файлів, після цього нейронна мережа аналізує обрані знімки та має необов’язкову можливість навчитися на цих знімках. Також надається можливість налаштувати параметри які відповідають за попередньою обробку даних тобто змінити параметри розмиття по Гаусу та поріг яскравості для бінарізації.

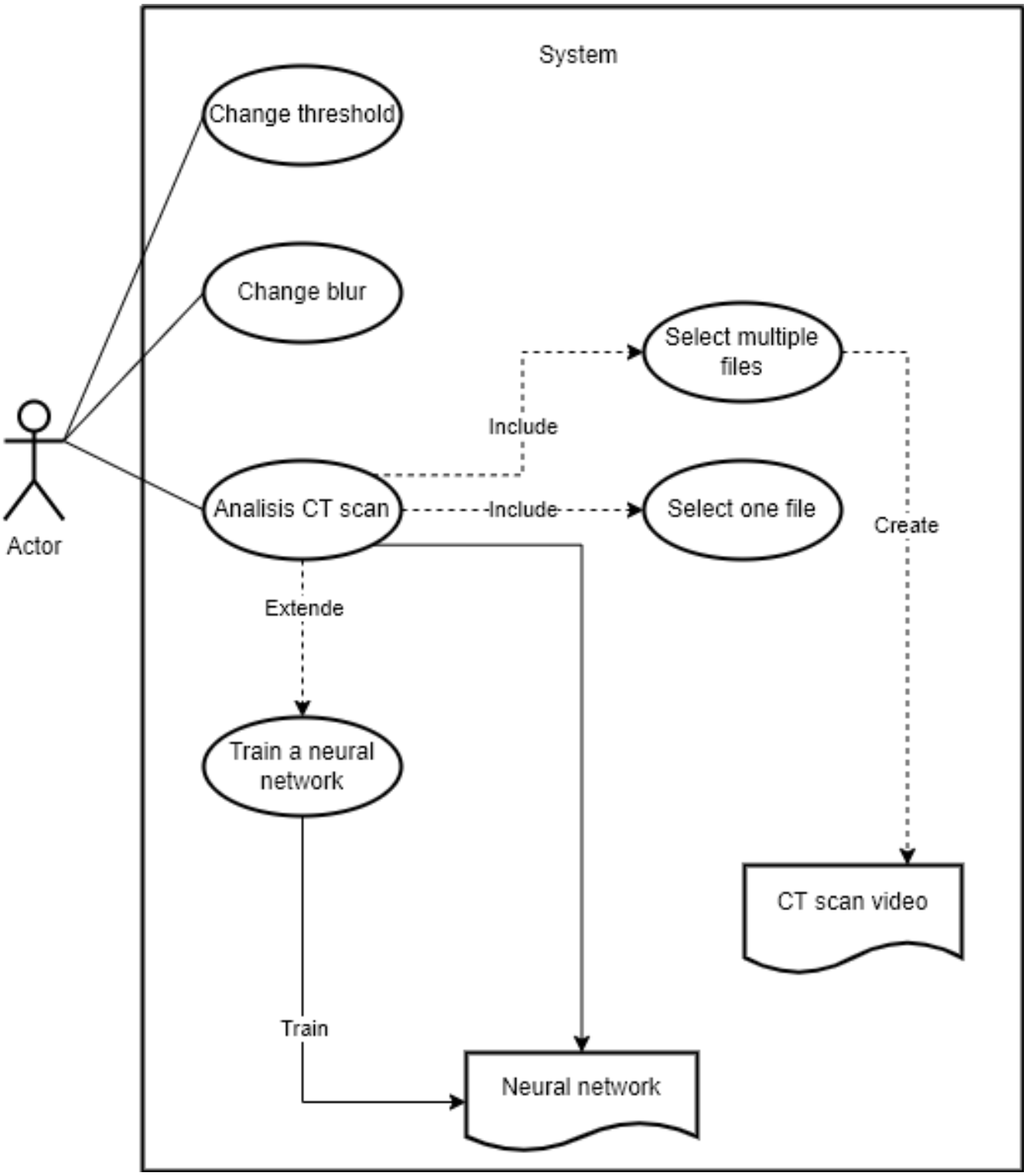

Рисунок 4.21 - Діаграма прецедентів

4.7 Алгоритм роботи системи

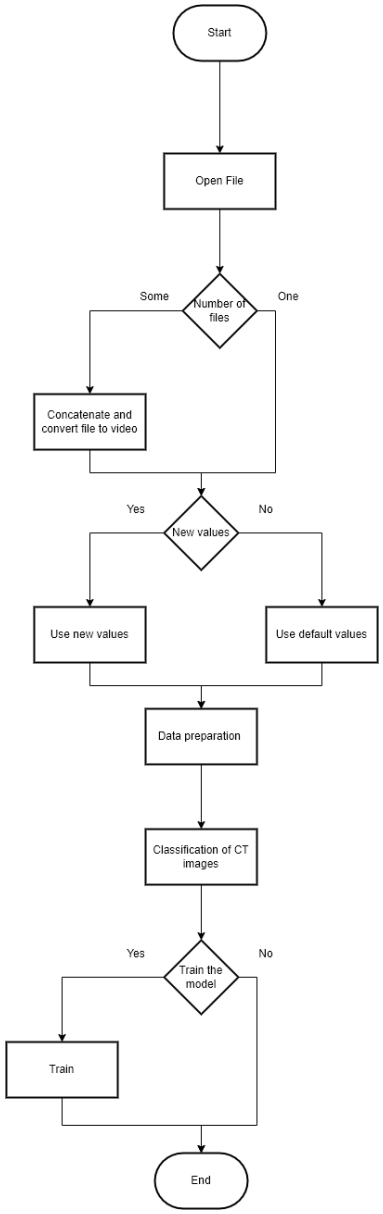

Рисунок 4.22 - Блок схема алгоритму роботи системи

На цій блок схему представлений User flow(алгоритм роботи) - це шлях, який пройде прототип користувача в програмі для виконання завдання. Потік користувачів веде користувача від точки входу через набір кроків до успішного результату та кінцевої дії.

4.7.1 Start

Початкове вікно

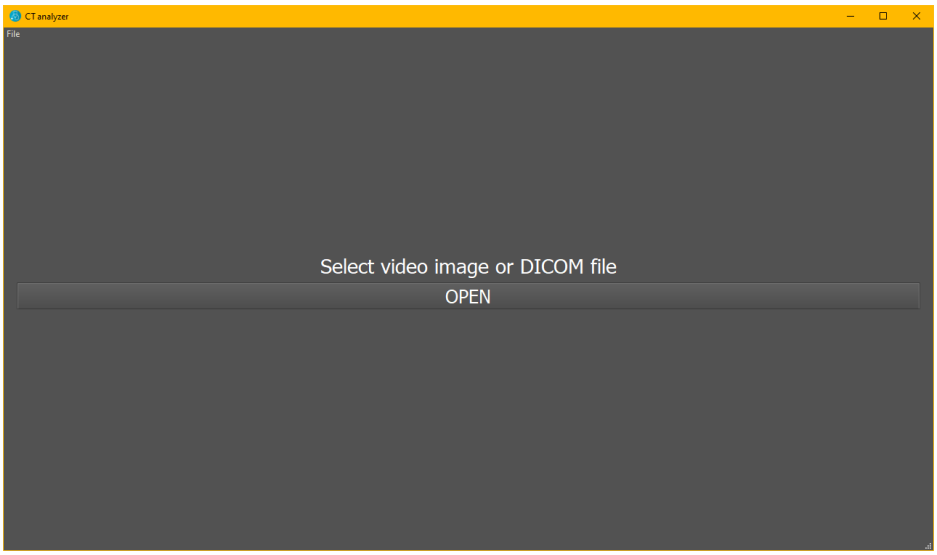

Рисунок 4.23 - Стартове вікно

4.7.2 Open File

Вибір файлу можливий через копку на головному екрані або через меню File -> Open.

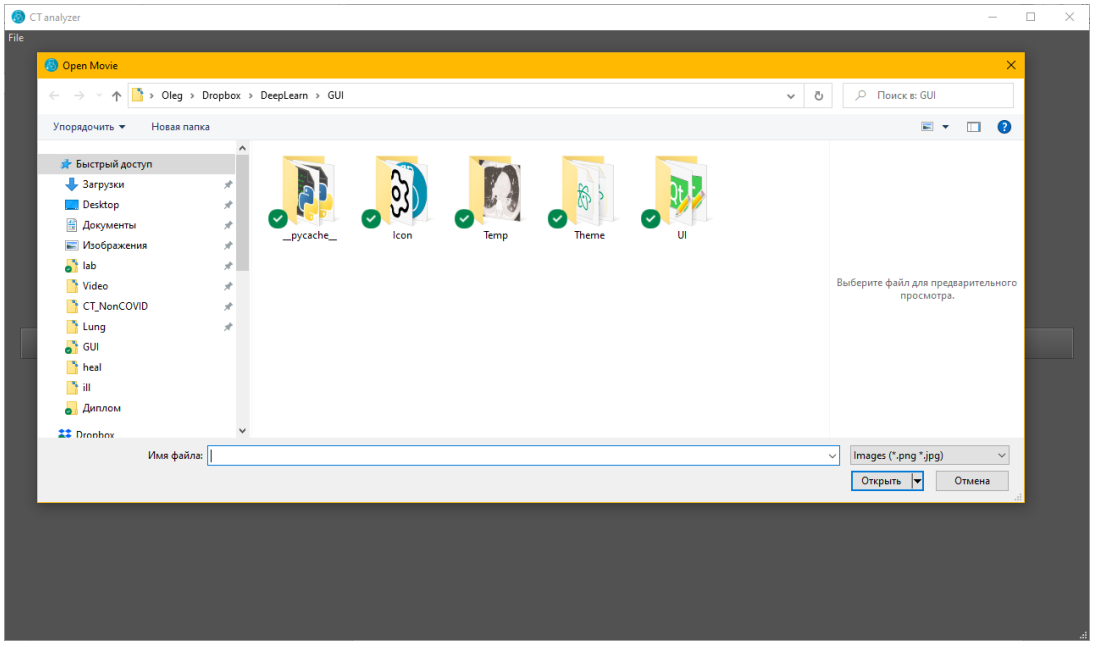

Рисунок 4.24 - Діалогове вікно з вибором файлу

4.7.3 Number of files

Можливий вибір одного або декількох файлів в форматах jpg, png, mp4 або DICOM .

Рисунок 4.25 - Приклад виборі одного або декількох файлів

4.7.4 Concatenate and convert file to video

Якщо були вибрані декілька файлів, то вони будуть склеєні в один файл GUI\Temp\ comp_video.mp4.

Рисунок 4.26 - Приклад відео файлу який був зібраний з DICOM файлів

4.7.5 New values

На цьому етапі дається можливість вибрати сегмент відео який буде аналізуватися та налаштування параметрів розмиття по Гаусу і поріг яскравості, для обробки даних.

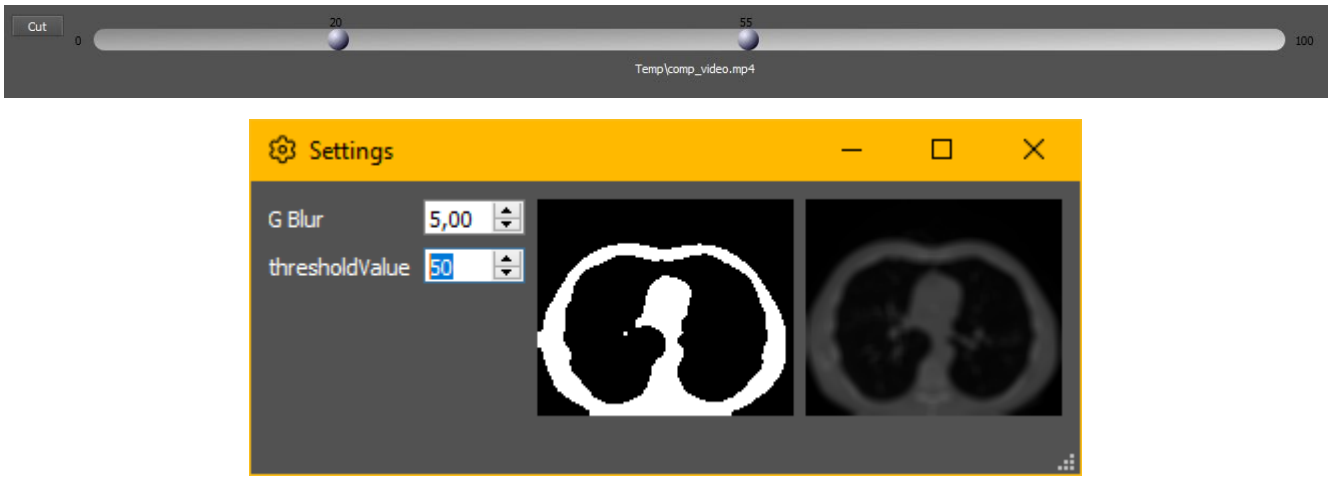

Рисунок 4.27 - Вікно налаштувань та вибір сегменту

4.7.6 Data preparation та Classification of CT images

Дані будуть зведені до загального виду та до кожного знімку буде виставлений свій діагноз. Якщо система зробила помилку є можливість не використовувати цей знімок у подальшому навчанні.

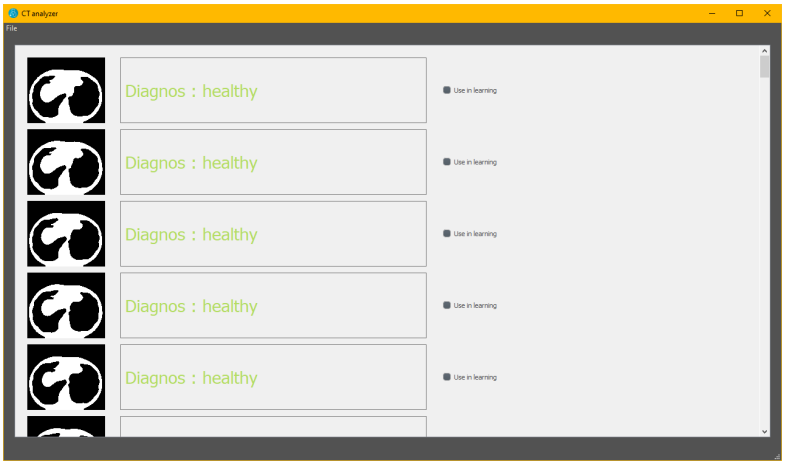

Рисунок 4.28 - Приклад проаналізованих знімків

4.7.7 Train the model

Користувач може відмінити навчання системи за бажанням.

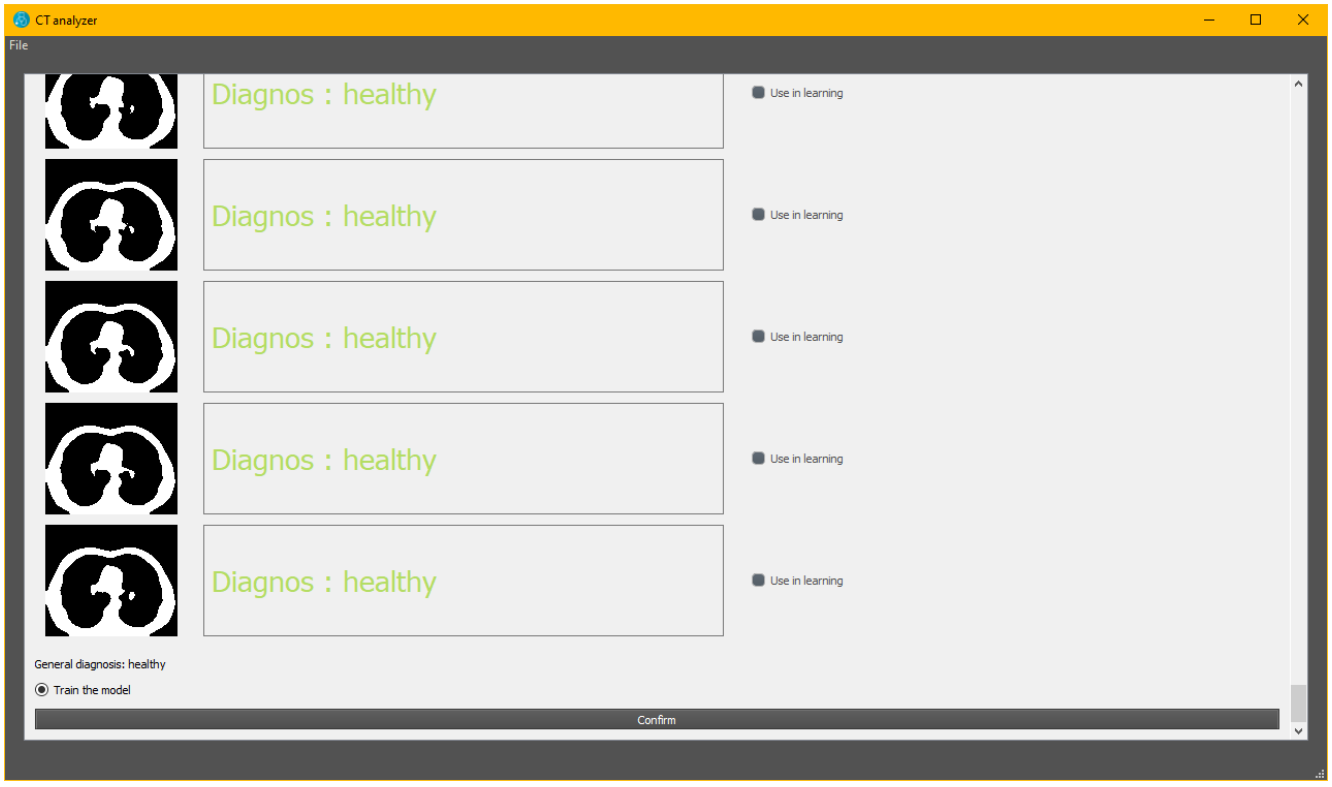

Рисунок 4.29 - Навчати модель чи ні

4.7.8 End

Завершення роботи програми

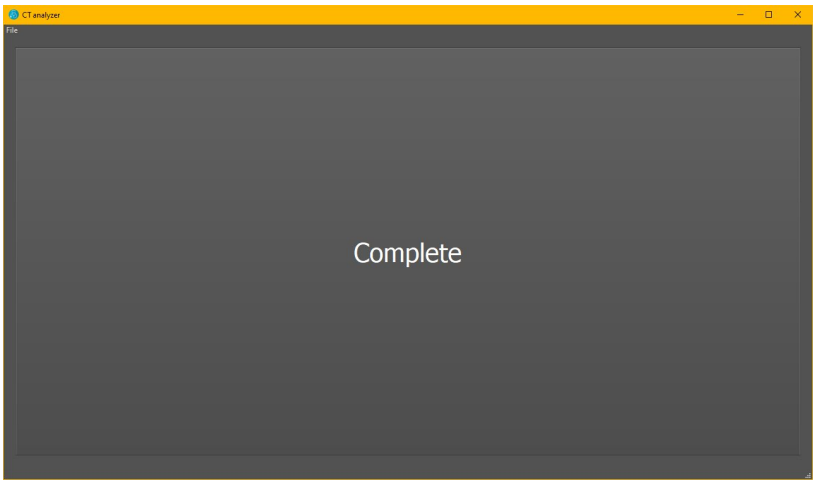

Рисунок 4.30 Завершення роботи

Висновки до розділу

У цьому розділі була детально розглянута структура програми, структура коду, був представлений алгоритм роботи з програмою та можливі способи використання програми.

5 ІНСТАЛЯЦІЯ ТА СИСТЕМНІ ВИМОГИ

Для початку інсталяції на комп’ютері повинен бути встановлений python версії 3.9 або вище. Для встановлення усіх необхідних модулів потрібно виконати команду pip install -r requirements.txt. Рекомендується провести інсталяцію у віртуальному середовищі яке можна створити командою python -m venv .venv та активувати .\venv\Scripts\ activate.

При наявності кати Nvidia можна встановити CUDA Toolkit та cuDNN SDK це пришвидшить аналіз знімків та навчання мережі.

Слід попередити що для великого об’єму знімків необхідно мати великий обсяг пам’яті.

Хоч інсталяцію і можна виконати вручну, але для зручності користувача був створений інсталятор install.exe. Після запуску install.exe будуть завантажені всі необхідні модулі та створений CT_Analyzer.exe у папці GUI. Запуск також можна провести вручну запустивши gui.pyw.

Системні вимоги :

- Мінімальні:
    - Процесор Intel i5 9400 або AMD Ryzen 5 1600x
    - Оперативна пам'ять 8 Гб
    - Відеокарта GeForce GTX 960 або аналог
- Рекомендовані
    - Процесор Intel i7 10700 або AMD Ryzen 7 3700x
    - Оперативна пам'ять 32 Гб
    - Відеокарта GeForce RTX 3060 або аналог

ВИСНОВКИ

У роботі було створено програмне забезпечення для бінарної класифікації знімків комп’ютерної томографії. На даний момент часу згорткові нейронні мережі це найбільш поширені та популярні мережі, і треба сказати що не дарма. Хоч вони і вимагають більших обчислень ніж звичайні мережі, але ж і їхня точність набагато більша. Точність моделі представленої в роботі не ідеальна, вважаю що це не пов’язано з кількістю шарів чи розміром навчальної вибірки. Об'єкти на зображеннях досить прості та однотипні, тож збільшення слоїв або навчальної вибірки не дасть значного результату. Скоріш за все значні зрушення в точності принесе лише покращення якості навчальної вибірки, але без людського втручання це не можливо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Worldometers. [Online]. Available: https://www.worldometers.info/coronavirus/
2. Shakouri, S., Bakhshali, M.A., Layegh, P. et al. COVID19-CT-dataset: an open access chest CT image repository of 1000+ patients with confirmed COVID-19 diagnosis. BMC Res Notes 14, 178 (2021). https://doi.org/10.1186/s13104-021-05592-x
3. https://portal.gdc.cancer.gov/
4. https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images
5. https://wiki.cancerimagingarchive.net/display/Public/RIDER+Lung+CT
6. Krizhevsky, Alex. "ImageNet Classification with Deep Convolutional Neural Networks"
7. M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, “The history began from alexnet: A comprehensive survey on deep learning approaches,” arXiv preprint arXiv:1803.01164, 2018.
8. Alex Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.
9. L. Zhang, M. Wang, M. Liu, and D. Zhang, “A survey on deep learning for neuroimaging-based brain disorder analysis,” arXiv preprint arXiv:2005.04573, 2020.
10. M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, “The history began from alexnet: A comprehensive survey on deep learning approaches,” arXiv preprint arXiv:1803.01164, 2018.
11. Ffmpeg Documentation. https://ffmpeg.org/documentation.html
12. ffmpeg-python Convert video to numpy array https://github.com/kkroening/ffmpeg python/tree/master/examples
13. Advanced PyQt5 e-book https://zetcode.com/ebooks/advancedpyqt5/

ДОДАТОК А

Автоматизована система контролю працездатності ресурсів Приймальної комісії КПІ ім. Ігоря Сікорського

Лістинг програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР82_№8210_ 22Б

Аркушів 42

Київ 2022

```
# Опис модуля gui.py
import os
import sys

import cv2
import ffmpeg
import matplotlib.pyplot as plt
import numpy as np
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import Qt, QSettings
from PyQt5.QtGui import QImage, QPixmap
from PyQt5.QtMultimedia import QMediaContent, QMediaPlayer
from PyQt5.QtWidgets import QFrame, QFileDialog
from tensorflow import keras

from perfectGUI import Ui_CT_main_win
from qtimeline import VideoSample
from settings import Ui_Settings_win
import pydicom

# Головний клас який збирає в себе весь функціонал програми

class mywindow(QtWidgets.QMainWindow):

    def __init__(self):

        super(mywindow, self).__init__()
        self.ui = Ui_CT_main_win()
        self.ui.setupUi(self)
```

```
        self.ui.pushButton.clicked.connect(self.openFile)
        self.ui.pushButton_3.clicked.connect(self.play)
        self.ui.actionOpen.triggered.connect(self.openFile)
        self.ui.actionSettings.triggered.connect(lambda: self.Settings_win.show())
        self.ui.actionClose.triggered.connect(self.close)
        self.ui.actionClose_File.triggered.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.select))
        self.ui.pushButton_2.clicked.connect(self.cutVideo)
        self.ui.pushButton_4.clicked.connect(self.learn)
        self.ui.pushButton_5.clicked.connect(lambda:
self.ui.stackedWidget.setCurrentWidget(self.ui.select))
        self.ui.mediaPlayer.positionChanged.connect(self.positionChanged)
        self.ui.mediaPlayer.setNotifyInterval(30)

        self.frame_bin = []
        self.video = []
        self.frames = []

        self.model_loaded = None
        self.open_settings(False)



    def openFile(self):
        """
        Метод відповідає за відкриття файлу. Якщо файлів декілька вони будуть
склені в один
        :return:
        """
```

```
        self.fileName, _ = QFileDialog.getOpenFileNames(self, "Open Movie", "",
"Images (*.png *.jpg);;Video (*.mp4);;Dicom (*.dcm)")
        if len(self.fileName) > 1:
            print(self.fileName)
            frames=[]
            out = cv2.VideoWriter(r'Temp\comp_video.mp4',
cv2.VideoWriter_fourcc(*'mp4v'), 10,(512, 512))
            for i in self.fileName:
                name=os.path.basename(i)
                n, e = os.path.splitext(name)
                if e==".jpg" or e==".png" or e==".PNG"or e==".JPG":
                    img = cv2.imdecode(np.fromfile(i, dtype=np.uint8),
cv2.IMREAD_COLOR)
                    img = cv2.resize(img, (512,512))
                    out.write(img)
                if e==".mp4":
                    cap = cv2.VideoCapture(i)
                    r = True
                    while r:
                        r, img = cap.read()
                        if r:
                            img = cv2.resize(img, (512, 512))
                            out.write(img)
                        else:
                            break
                    cap.release()
                if e==".dcm":
                    frames.append(pydicom.dcmread(i))
            slices = []
```

```
            for f in frames:
                if hasattr(f, 'SliceLocation'):
                    slices.append(f)
            slices = sorted(slices, key=lambda s: s.SliceLocation)
            image_2d_scaled = []
            for s in slices:
                img = (s.pixel_array.astype(float))
                image_2d_scaled.append((np.maximum(img, 0) / img.max()) * 255.0)
            image_2d_scaled = np.uint8(image_2d_scaled)

            for i in image_2d_scaled[:] :
                im = cv2.cvtColor(i, cv2.COLOR_GRAY2RGB)
                out.write(im)

            out.release()
            self.fileName="Temp\comp_video.mp4"
        elif len(self.fileName) == 1 :
            self.fileName = self.fileName[0]

        self.frame_bin = []
        print(self.fileName)
        if self.fileName != []:

self.ui.mediaPlayer.setMedia(QMediaContent(QtCore.QUrl.fromLocalFile(self.fileNam
e)))
            self.video_to_pixmap(self.fileName)
            self.ui.label_2.setText(self.fileName)
            self.ui.stackedWidget.setCurrentWidget(self.ui.cut)
            for i in reversed(range(self.ui.cheking_grid.count())):
```

```
            self.ui.cheking_grid.itemAt(i).widget().setParent(None)

    def play(self):
        """
        Метод який починає та зупиняє відео
        :return:
        """
        if self.ui.mediaPlayer.state() == QMediaPlayer.PlayingState:
            self.ui.mediaPlayer.pause()
        else:
            self.ui.mediaPlayer.play()



    def positionChanged(self, position):
        """
        Метод для вибору часу на слайдері
        :param position: Час
        :return:
        """
        x = self.ui.qtimeline.x()
        self.ui.qtimeline.pointerPos = x
        self.ui.qtimeline.positionChanged.emit(x)

        self.ui.qtimeline.pointerTimePos = position / 1000
        self.ui.qtimeline.checkSelection(x)
        self.ui.qtimeline.update()
        # self.ui.qtimeline.clicking = True

    def setPosition(self, position):
```

```
        """
        Метод який рухає повзунок з часом
        :param position: Час
        :return:
        """
        self.ui.mediaPlayer.setPosition(position)

    def mousePressEvent(self, e):
        """
        Евент кліку миші по віджету
        :param e: Евент
        :return:
        """
        if e.button() == Qt.LeftButton:
            self.ui.mediaPlayer.pause()
            x = e.pos().x() - self.ui.pushButton_2.width() - 12
            self.ui.qtimeline.pointerPos = x
            self.ui.qtimeline.positionChanged.emit(x)
            self.ui.qtimeline.pointerTimePos = self.ui.qtimeline.pointerPos *
self.ui.qtimeline.getScale()
            self.ui.qtimeline.checkSelection(x)
            self.ui.qtimeline.update()
            self.ui.qtimeline.clicking = True

    def mouseReleaseEvent(self, e):
        """
        Евент відпускання миші
        :param e: Евент
        :return:
```

```
        """
        if e.button() == Qt.LeftButton:
            self.ui.mediaPlayer.setPosition(int(self.ui.qtimeline.pointerTimePos * 1000))
            self.ui.qtimeline.clicking = False

    def handleError(self):
        """
        Обробка помилок
        :return:
        """
        self.ui.pushButton_3.setEnabled(False)
        self.ui.label_2.setText("Error: " + self.ui.mediaPlayer.errorString())

    def video_to_pixmap(self, fileName):
        """
        Створення зображень для попереднього перегляду
        :param fileName: Назва файлу
        :return:
        """
        cap = cv2.VideoCapture(fileName)
        fps = cap.get(cv2.CAP_PROP_FPS)
        self.frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        self.frames = []
        r = True
        while r:
            r, img = cap.read(cv2.IMREAD_GRAYSCALE)
            if r:
                self.frames.append(img)
            else:
```

```
            break
        self.frames = np.array(self.frames)
        videoSample = []

        if self.frames.shape[0] > 30:
            step = self.frames.shape[0] // (self.geometry().width() // 52)
            m = self.frames.shape[0] / (self.geometry().width() // 52)
        else:
            step = 1
            m = 1
        print(self.frames.shape)
        for i in self.frames[::step]:
            im = cv2.cvtColor(i, cv2.COLOR_BGR2GRAY)
            im = np.uint8(im)
            qimage = QImage(im, im.shape[0], im.shape[1], QImage.Format_Indexed8)
            videoSample.append(
                VideoSample(duration=((self.frame_count / fps) / self.frame_count) * (m),
picture=QPixmap(qimage)))

        self.ui.qtimeline.duration = self.frame_count / fps
        self.ui.qtimeline.videoSamples = videoSample

    def open_settings(self, show=True):
        """
        Відкриття вікна налаштувань
        :param show: Від цього параметру залежить чи буде показуватись вікно
        :return:
        """
        self.Settings_win = QtWidgets.QMainWindow()
```

```
        File = open("Theme/ElegantDark.qss", 'r')
        with File:
            qss = File.read()
            app.setStyleSheet(qss)

        self.ui_settings = Ui_Settings_win()
        self.ui_settings.setupUi(self.Settings_win)
        self.ui_settings.doubleSpinBox_Blur.valueChanged.connect(self.draw_preview)

self.ui_settings.spinBox_thresholdValue.valueChanged.connect(self.draw_preview)

        self.Settings_win.closeEvent = lambda e: self.Settings_win.hide()

        self.draw_preview()

        if show:
            self.show_set(self.Settings_win)

    def draw_preview(self):
        """
        Попередній перегляд у вікні налаштувань
        :return:
        """
        if len(self.frames) == 0:
            im = cv2.imread('Icon/PngItem_320012.png')

        else:
            im = self.frames[int(len(self.frames) / 2)]
```

```
        im = cv2.resize(im, (128, 128), interpolation=cv2.INTER_AREA)
        im = im[:-20, :]

        im = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        im = cv2.GaussianBlur(im, (
            int(self.ui_settings.doubleSpinBox_Blur.value()),
int(self.ui_settings.doubleSpinBox_Blur.value())), 5,
                              cv2.BORDER_DEFAULT)
        self.ui_settings.settings_pixmap_ori.setPixmap(
            QPixmap(QImage(im, im.shape[1], im.shape[0], QImage.Format_Indexed8)))
        _, im = cv2.threshold(im, int(self.ui_settings.spinBox_thresholdValue.value()),
255, cv2.THRESH_BINARY)

        im = np.array(im)

        im = np.uint8(im)

        self.ui_settings.settings_pixmap.setPixmap(
            QPixmap(QImage(im, im.shape[1], im.shape[0], QImage.Format_Indexed8)))

    def show_set(self, win):
        """

        :param win:Від цього параметру залежить чи буде показуватись вікно
        :return:
        """
        win.show()
```

```
def cutVideo(self):
    """
    Цей метод обрізає відео виконує аналізує знімки та виводить їх на екран
    :return:
    """
    self.ui.stackedWidget.setCurrentWidget(self.ui.cheking)

    cut_start, cut_end = self.ui.range_slider.value()

    out, _ = (
        ffmpeg
            .input(self.fileName)
            .trim(start_frame=int(cut_start / 100 * self.frame_count),
                  end_frame=int(cut_end / 100 * self.frame_count))
            .setpts('PTS-STARTPTS')
            .filter('scale', 128, 128)
            .output('pipe:', format='rawvideo', pix_fmt='rgb24')
            .run(capture_stdout=True)
    )
    self.video = (
        np
            .frombuffer(out, np.uint8)
            .reshape([-1, 128, 128, 3])
    )

    for i in self.video:
        im_g = i[:-20, :]
        im_g = cv2.cvtColor(im_g, cv2.COLOR_RGB2GRAY)
```

```
            im_g = cv2.GaussianBlur(im_g, (
                int(self.ui_settings.doubleSpinBox_Blur.value()),
int(self.ui_settings.doubleSpinBox_Blur.value())), 5,
                                    cv2.BORDER_DEFAULT)

            ret, thresh1 = cv2.threshold(im_g,
int(self.ui_settings.spinBox_thresholdValue.value()), 255,
                                         cv2.THRESH_BINARY)
            self.frame_bin.append(thresh1)
        self.frame_bin = np.array(self.frame_bin)
        self.dignoses = self.prediction_diagnos(self.frame_bin)

        # DRAW RESULTS

        for n, i in enumerate(self.frame_bin):
            img = QtWidgets.QLabel(self.ui.scrollAreaWidgetContents)
            img.setObjectName("IMG{0}".format(n))
            img.setStyleSheet("color: rgb(0, 0, 0);")

            self.ui.cheking_grid.addWidget(img, n, 0, 1, 1)
            im = np.uint8(i)
            qpixmap = QPixmap(QImage(im, im.shape[1], im.shape[0],
QImage.Format_Indexed8))
            img.setPixmap(qpixmap)
            img.setSizePolicy(QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum,
QtWidgets.QSizePolicy.Maximum))

            diagnos_label = QtWidgets.QLabel(self.ui.scrollAreaWidgetContents)
            diagnos_label.setObjectName("Diagnos{}".format(n))
```

```
        diagnos_label.setFrameShape(QFrame.StyledPanel)
        font = QtGui.QFont()
        font.setPointSize(21)
        diagnos_label.setFont(font)
        if all(self.dignoses[n] == [1, 0]):
            diagnos_label.setStyleSheet("color: rgb(180, 221, 99);")
            diagnos_label.setText("Diagnos : healthy")

        elif all(self.dignoses[n] == [0, 1]):
            diagnos_label.setText("Diagnos : ill")
            diagnos_label.setStyleSheet(" color: rgb(228, 102, 102);")
        else:
            diagnos_label.setText(r"Diagnos : ¯\_(ツ)_/¯")
            diagnos_label.setStyleSheet("color: rgb(0, 0, 0);")
        self.ui.cheking_grid.addWidget(diagnos_label, n, 1, 1, 1)
        checkBox = QtWidgets.QCheckBox(self.ui.scrollAreaWidgetContents)
        checkBox.setText("Use in learning")
        checkBox.setObjectName("checkBox{}".format(n))
        checkBox.setChecked(True)
        checkBox.setStyleSheet(" color:rgb(82,82,82);")
        self.ui.cheking_grid.addWidget(checkBox, n, 2, 1, 1)

def prediction_diagnos(self, frame):
    """
    Метод який завантажує модель та робить передбачення
    :param frame:
    :return:
    """
```

```
        self.model_loaded = keras.models.load_model(os.path.abspath('../') +
r"\DNN\lung_model30000_50000")
        frame = frame / 255
        d = self.model_loaded.predict(np.expand_dims(frame, axis=3))
        d = np.around(d)
        # print(d)
        il = 1
        hl = 1
        for i in d:
            if all(i == [0, 1]):
                il += 1
            elif all(i == [1, 0]):
                hl += 1
        if il / hl >= 0.1:
            self.ui.label_3.setText("General diagnosis: ill")
        else:
            self.ui.label_3.setText("General diagnosis: healthy")
        return d

    def learn(self):
        """
        Медод який навчає модель
        :return:
        """
        use = []
        for i in range(self.ui.cheking_grid.count()):
            if isinstance(self.ui.cheking_grid.itemAt(i).widget(), QtWidgets.QCheckBox):
                use.append(self.ui.cheking_grid.itemAt(i).widget().isChecked())
        use_frame = []
```

```
        for n, i in enumerate(use):
            if i:
                use_frame.append(self.frame_bin[n])
        use_frame = np.array(use_frame)
        print(use_frame.shape)
        if self.ui.radioButton.isChecked():
            self.model_loaded.fit(np.expand_dims(use_frame, axis=3), self.dignoses,
batch_size=4, epochs=3,
                                  validation_split=0.1)
            self.ui.pushButton_5.setEnabled(True)

        self.ui.stackedWidget.setCurrentWidget(self.ui.restart)
        self.ui.pushButton_5.setEnabled(True)



if __name__ == '__main__':
    app = QtWidgets.QApplication([])
    application = mywindow()
    File = open("Theme/ElegantDark.qss", 'r')
    with File:
        qss = File.read()
        app.setStyleSheet(qss)
    application.show()

    sys.exit(app.exec())



# -*- coding: utf-8 -*-
# Опис мудуля perfectGUI.py
```

```
# Form implementation generated from reading ui file 'perfectGUI.ui'



import sys

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtMultimedia import QMediaPlayer
from PyQt5.QtMultimediaWidgets import QVideoWidget
from PyQt5.QtWidgets import QSlider
from qtrangeslider import QLabeledRangeSlider

from qtimeline import QTimeLine
from settings import Ui_Settings_win



# Таблиці стилей для подвійного слайдера
QSS = """
QSlider {
    min-height: 20px;
}

QSlider::groove:horizontal {
    border: 0px;
    background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0 #888, stop:1 #ddd);
    height: 20px;
```

```
    border-radius: 10px;
}

QSlider::handle {
    background: qradialgradient(cx:0, cy:0, radius: 1.2, fx:0.35,
                                fy:0.3, stop:0 #eef, stop:1 #002);
    height: 20px;
    width: 20px;
    border-radius: 10px;
}

QSlider::sub-page:horizontal {
    background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0 #227, stop:1 #77a);
    border-top-left-radius: 10px;
    border-bottom-left-radius: 10px;
}


"""

# Клас з описом обєктів головного вікна
class Ui_CT_main_win(object):

    def setupUi(self, CT_main_win):
        """
        Ініціалізація обєктів вікна
        :param CT_main_win: Вікно
        :return:
        """
```

```
        CT_main_win.setObjectName("CT_main_win")
        CT_main_win.resize(1280, 720)
        CT_main_win.setWindowIcon(QtGui.QIcon('Icon/PngItem_320012.png'))
        self.mediaPlayer = QMediaPlayer(None, QMediaPlayer.VideoSurface)
        videoWidget = QVideoWidget()
        self.mediaPlayer.setVideoOutput(videoWidget)
        self.centralwidget = QtWidgets.QWidget(CT_main_win)
        self.centralwidget.setAcceptDrops(False)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout_2 = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.stackedWidget = QtWidgets.QStackedWidget(self.centralwidget)
        self.stackedWidget.setEnabled(True)
        self.stackedWidget.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.stackedWidget.setFrameShape(QtWidgets.QFrame.NoFrame)
        self.stackedWidget.setObjectName("stackedWidget")
        self.select = QtWidgets.QWidget()
        self.select.setObjectName("select")
        self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.select)
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setSizeConstraint(QtWidgets.QLayout.SetDefaultConstraint)
        self.verticalLayout_2.setContentsMargins(-1, -1, -1, 20)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_2.addItem(spacerItem)
        self.label = QtWidgets.QLabel(self.select)
        font = QtGui.QFont()
```

```
        font.setPointSize(21)
        self.label.setFont(font)
        self.label.setAlignment(QtCore.Qt.AlignBottom | QtCore.Qt.AlignHCenter)
        self.label.setObjectName("label")
        self.verticalLayout_2.addWidget(self.label)
        self.pushButton = QtWidgets.QPushButton(self.select)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.pushButton.sizePolicy().hasHeightForWidth())
        self.pushButton.setSizePolicy(sizePolicy)
        self.pushButton.setMinimumSize(QtCore.QSize(0, 0))
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(19)
        self.pushButton.setFont(font)
        self.pushButton.setObjectName("pushButton")
        self.verticalLayout_2.addWidget(self.pushButton)
        spacerItem1 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_2.addItem(spacerItem1)
        self.verticalLayout_3.addLayout(self.verticalLayout_2)
        self.stackedWidget.addWidget(self.select)
        self.cut = QtWidgets.QWidget()
        self.cut.setObjectName("cut")
        self.verticalLayout_4 = QtWidgets.QVBoxLayout(self.cut)
        self.verticalLayout_4.setSpacing(9)
        self.verticalLayout_4.setObjectName("verticalLayout_4")
```

```
self.gridLayout = QtWidgets.QGridLayout()
self.gridLayout.setSizeConstraint(QtWidgets.QLayout.SetMaximumSize)
self.gridLayout.setObjectName("gridLayout")

self.qtimeline = QTimeLine(24, 20)
self.qtimeline.setObjectName("TimeLine")
self.qtimeline.setMaximumHeight(120)

self.gridLayout.addWidget(self.qtimeline, 1, 1, 1, 1)
self.pushButton_2 = QtWidgets.QPushButton(self.cut)
self.pushButton_2.setObjectName("pushButton_2")
self.gridLayout.addWidget(self.pushButton_2, 2, 0, 1, 1)

self.range_slider = QLabeledRangeSlider()
self.range_slider.setOrientation(QtCore.Qt.Horizontal)
self.range_slider.setObjectName("range_slider")
self.range_slider.setTickPosition(QSlider.TicksBelow)
self.range_slider.setTickInterval(5)
self.range_slider.setMaximumHeight(120)
self.range_slider.setStyleSheet(QSS)
self.range_slider.setRange(0, 100)
self.range_slider.setValue((20,55))

self.gridLayout.addWidget(self.range_slider, 2, 1, 1, 1)
self.pushButton_3 = QtWidgets.QPushButton(self.cut)
self.pushButton_3.setObjectName("pushButton_3")
self.gridLayout.addWidget(self.pushButton_3, 1, 0, 1, 1)
self.horizontalLayout = QtWidgets.QHBoxLayout()
self.horizontalLayout.setObjectName("horizontalLayout")
```

```
        self.horizontalLayout.addWidget(videoWidget)
        self.gridLayout.addLayout(self.horizontalLayout, 0, 1, 1, 1)
        self.label_2 = QtWidgets.QLabel(self.cut)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Minimum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_2.sizePolicy().hasHeightForWidth())
        self.label_2.setSizePolicy(sizePolicy)
        self.label_2.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2.setObjectName("label_2")
        self.gridLayout.addWidget(self.label_2, 3, 1, 1, 1)
        self.gridLayout.setRowStretch(0, 2)
        self.gridLayout.setRowStretch(1, 2)
        self.gridLayout.setVerticalSpacing(10)
        self.gridLayout.setVerticalSpacing(10)
        self.gridLayout.setContentsMargins(10, 10, 10, 10)
        self.verticalLayout_4.addLayout(self.gridLayout)
        self.stackedWidget.addWidget(self.cut)
        self.cheking = QtWidgets.QWidget()
        self.cheking.setObjectName("cheking")
        self.verticalLayout_5 = QtWidgets.QVBoxLayout(self.cheking)
        self.verticalLayout_5.setObjectName("verticalLayout_5")
        self.scrollArea = QtWidgets.QScrollArea(self.cheking)
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")
        self.scrollAreaWidgetContents = QtWidgets.QWidget()
        self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0, 168, 146))
        self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")
```

```
        self.gridLayout_3 = QtWidgets.QGridLayout(self.scrollAreaWidgetContents)
        self.gridLayout_3.setContentsMargins(10, 10, 10, 10)
        self.gridLayout_3.setSpacing(10)
        self.gridLayout_3.setObjectName("gridLayout_3")
        self.pushButton_4 = QtWidgets.QPushButton(self.scrollAreaWidgetContents)
        self.pushButton_4.setObjectName("pushButton_4")
        self.gridLayout_3.addWidget(self.pushButton_4, 4, 0, 1, 1)
        self.radioButton = QtWidgets.QRadioButton(self.scrollAreaWidgetContents)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.radioButton.sizePolicy().hasHeightForWidth())
        self.radioButton.setSizePolicy(sizePolicy)
        self.radioButton.setChecked(True)
        self.radioButton.setAutoRepeat(False)
        self.radioButton.setObjectName("radioButton")
        self.gridLayout_3.addWidget(self.radioButton, 3, 0, 1, 1)
        self.label_3 = QtWidgets.QLabel(self.scrollAreaWidgetContents)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_3.sizePolicy().hasHeightForWidth())
        self.label_3.setSizePolicy(sizePolicy)
        self.label_3.setObjectName("label_3")
        self.label_3.setStyleSheet("color: rgb(0, 0, 0);")
        self.gridLayout_3.addWidget(self.label_3, 2, 0, 1, 1)
        self.cheking_grid = QtWidgets.QGridLayout()
```

```
        self.cheking_grid.setContentsMargins(10, 10, 10, 10)
        self.cheking_grid.setHorizontalSpacing(25)
        self.cheking_grid.setVerticalSpacing(10)
        self.cheking_grid.setObjectName("cheking_grid")

        self.cheking_grid.setColumnStretch(0, 5)
        self.gridLayout_3.addLayout(self.cheking_grid, 0, 0, 1, 1)
        spacerItem2 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.gridLayout_3.addItem(spacerItem2, 1, 0, 1, 1)
        self.scrollArea.setWidget(self.scrollAreaWidgetContents)
        self.verticalLayout_5.addWidget(self.scrollArea)
        self.stackedWidget.addWidget(self.cheking)
        self.restart = QtWidgets.QWidget()
        self.restart.setObjectName("restart")
        self.gridLayout_4 = QtWidgets.QGridLayout(self.restart)
        self.gridLayout_4.setObjectName("gridLayout_4")

        self.pushButton_5 = QtWidgets.QPushButton(self.restart)
        self.pushButton_5.setEnabled(False)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.pushButton_5.sizePolicy().hasHeightForWidth())
        self.pushButton_5.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setPointSize(31)
```

```
self.pushButton_5.setFont(font)
self.pushButton_5.setObjectName("pushButton_5")
self.gridLayout_4.addWidget(self.pushButton_5, 1, 0, 1, 2)
self.stackedWidget.addWidget(self.restart)
self.gridLayout_2.addWidget(self.stackedWidget, 0, 0, 1, 1)
CT_main_win.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(CT_main_win)
self.menubar.setGeometry(QtCore.QRect(0, 0, 890, 21))
self.menubar.setObjectName("menubar")
self.menuFile = QtWidgets.QMenu(self.menubar)
self.menuFile.setObjectName("menuFile")
CT_main_win.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(CT_main_win)
self.statusbar.setObjectName("statusbar")
CT_main_win.setStatusBar(self.statusbar)
self.actionOpen = QtWidgets.QAction(CT_main_win)
self.actionOpen.setObjectName("actionOpen")
self.actionClose = QtWidgets.QAction(CT_main_win)
self.actionClose.setObjectName("actionClose")
self.actionSettings = QtWidgets.QAction(CT_main_win)
self.actionSettings.setObjectName("actionSettings")

self.actionClose_File = QtWidgets.QAction(CT_main_win)
self.actionClose_File.setObjectName("actionClose_File")
self.menuFile.addAction(self.actionOpen)
self.menuFile.addAction(self.actionClose_File)
self.menuFile.addAction(self.actionSettings)
self.menuFile.addAction(self.actionClose)
self.menubar.addAction(self.menuFile.menuAction())
```

```
        self.retranslateUi(CT_main_win)
        self.stackedWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(CT_main_win)

    def retranslateUi(self, CT_main_win):
        _translate = QtCore.QCoreApplication.translate
        CT_main_win.setWindowTitle(_translate("CT_main_win", "CT analyzer"))
        self.label.setText(_translate("CT_main_win", "Select video image or DICOM
file"))
        self.pushButton.setText(_translate("CT_main_win", "OPEN"))
        self.pushButton_2.setText(_translate("CT_main_win", "Cut"))
        self.pushButton_3.setText(_translate("CT_main_win", "   Play    "))
        self.label_2.setText(_translate("CT_main_win", "Cut out the desired segment"))
        self.pushButton_4.setText(_translate("CT_main_win", "Confirm"))
        self.radioButton.setText(_translate("CT_main_win", "Train the model"))
        self.label_3.setText(_translate("CT_main_win", "General diagnosis: "))

        self.pushButton_5.setText(_translate("CT_main_win", "Complete"))
        self.menuFile.setTitle(_translate("CT_main_win", "File"))
        self.actionOpen.setText(_translate("CT_main_win", "Open"))
        self.actionClose.setText(_translate("CT_main_win", "Close"))
        self.actionSettings.setText(_translate("CT_main_win", "Settings"))
        self.actionClose_File.setText(_translate("CT_main_win", "Close File"))


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    CT_main_win = QtWidgets.QMainWindow()
```

```
    File = open("Theme/Darkeum.qss", 'r')
    with File:
        qss = File.read()
        app.setStyleSheet(qss)
    ui = Ui_CT_main_win()
    ui.setupUi(CT_main_win)
    CT_main_win.show()
    sys.exit(app.exec_())


# Опис мудуля settings.py
# Form implementation generated from reading ui file 'settings.ui'
# Created by: PyQt5 UI code generator 5.15.4
#



from PyQt5 import QtCore, QtGui, QtWidgets

# Клас з описом обєктів вікна налаштувань
class Ui_Settings_win(object):

    def setupUi(self, Settings_win):
        """
        Ініціалізація обєктів вікна
        :param CT_main_win: Вікно
        :return:
        """
        Settings_win.setObjectName("Settings_win")
```

```
        Settings_win.resize(410, 320)
        Settings_win.setWindowIcon(QtGui.QIcon('Icon/setting_ico.png'))
        self.centralwidget = QtWidgets.QWidget(Settings_win)
        self.centralwidget.setObjectName("centralwidget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.centralwidget)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.gridLayout = QtWidgets.QGridLayout()
        self.gridLayout.setObjectName("gridLayout")
        self.doubleSpinBox_Blur = QtWidgets.QDoubleSpinBox(self.centralwidget)
        self.doubleSpinBox_Blur.setProperty("value", 1.0)
        self.doubleSpinBox_Blur.setObjectName("doubleSpinBox_Blur")
        self.gridLayout.addWidget(self.doubleSpinBox_Blur, 0, 1, 1, 1)
        self.doubleSpinBox_Blur.setValue(5.0)
        self.doubleSpinBox_Blur.setSingleStep(2)
        self.doubleSpinBox_Blur.setMinimum(1)
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setObjectName("label_2")
        self.gridLayout.addWidget(self.label_2, 0, 0, 1, 1)
        self.spinBox_thresholdValue = QtWidgets.QSpinBox(self.centralwidget)
        self.spinBox_thresholdValue.setMaximum(255)
        self.spinBox_thresholdValue.setProperty("value", 50)
        self.spinBox_thresholdValue.setObjectName("spinBox_thresholdValue")
        self.gridLayout.addWidget(self.spinBox_thresholdValue, 1, 1, 1, 1)
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setObjectName("label_3")
        self.gridLayout.addWidget(self.label_3, 1, 0, 1, 1)
        spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.gridLayout.addItem(spacerItem, 3, 1, 1, 1)
```

```
        self.horizontalLayout.addLayout(self.gridLayout)
        self.settings_pixmap = QtWidgets.QLabel(self.centralwidget)
        self.settings_pixmap.setObjectName("settings_pixmap")
        self.horizontalLayout.addWidget(self.settings_pixmap)
        self.settings_pixmap_ori = QtWidgets.QLabel(self.centralwidget)
        self.settings_pixmap_ori.setObjectName("settings_pixmap_ori")
        self.horizontalLayout.addWidget(self.settings_pixmap_ori)

        self.horizontalLayout.setStretch(1, 3)
        Settings_win.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(Settings_win)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 821, 21))
        self.menubar.setObjectName("menubar")
        Settings_win.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(Settings_win)
        self.statusbar.setObjectName("statusbar")
        Settings_win.setStatusBar(self.statusbar)

        self.retranslateUi(Settings_win)
        QtCore.QMetaObject.connectSlotsByName(Settings_win)

    def retranslateUi(self, Settings_win):
        _translate = QtCore.QCoreApplication.translate
        Settings_win.setWindowTitle(_translate("Settings_win", "Settings"))
        self.label_2.setText(_translate("Settings_win", "G Blur "))
        self.label_3.setText(_translate("Settings_win", "thresholdValue"))
        self.settings_pixmap.setText(_translate("Settings_win", "TextLabel"))
```

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Settings_win = QtWidgets.QMainWindow()
    File = open("Theme/Darkeum.qss", 'r')
    with File:
        qss = File.read()
        app.setStyleSheet(qss)

    ui = Ui_Settings_win()
    ui.setupUi(Settings_win)
    Settings_win.show()
    sys.exit(app.exec_())



#!/usr/bin/python3
# -*- coding: utf-8 -*-

from PyQt5 import QtWidgets
from PyQt5.QtCore import Qt, QPoint, QLine, QRect, QRectF, pyqtSignal
from PyQt5.QtGui import QPainter, QColor, QFont, QBrush, QPalette, QPen,
QPolygon, QPainterPath
from PyQt5.QtWidgets import QWidget

__textColor__ = QColor(187, 187, 187)
__backgroudColor__ = QColor(60, 63, 65)
__font__ = QFont('Decorative', 10)
```

```
class VideoSample:

    def __init__(self, duration, color=Qt.darkYellow, picture=None, audio=None):
        self.duration = duration
        self.color = color  # Floating color
        self.defColor = color  # DefaultColor
        if picture is not None:
            self.picture = picture.scaledToHeight(45)
        else:
            self.picture = None
        self.startPos = 0  # Inicial position
        self.endPos = self.duration  # End position


class QTimeLine(QtWidgets.QMainWindow):

    positionChanged = pyqtSignal(int)
    selectionChanged = pyqtSignal(VideoSample)

    def __init__(self, duration, length):
        super(QWidget, self).__init__()
        self.duration = duration
        self.length = length

        # Set variables
        self.backgroundColor = __backgroudColor__
        self.textColor = __textColor__
        self.font = __font__
        self.pos = None
```

```
        self.pointerPos = None
        self.pointerTimePos = None
        self.selectedSample = None
        self.clicking = False  # Check if mouse left button is being pressed
        self.is_in = False  # check if user is in the widget
        self.videoSamples = []  # List of videos samples

        self.setMouseTracking(True)  # Mouse events
        self.setAutoFillBackground(True)  # background

        self.initUI()
        # self.show()


    def initUI(self):

        self.setGeometry(300, 300, self.length, 200)
        self.setWindowTitle("TESTE")

        # Set Background
        pal = QPalette()
        pal.setColor(QPalette.Background, self.backgroundColor)
        self.setPalette(pal)

    def paintEvent(self, event):
        qp = QPainter()
        qp.begin(self)
        qp.setPen(self.textColor)
        qp.setFont(self.font)
```

```
        qp.setRenderHint(QPainter.Antialiasing)
        w = 0
        # Draw time
        scale = self.getScale()
        while w <= self.width():
            qp.drawText(w - 50, 0, 100, 100, Qt.AlignHCenter, self.get_time_string(w *
scale))
            w += 100
        # Draw down line
        # qp.setPen(QPen(Qt.darkCyan, 5, Qt.SolidLine))
        # qp.drawLine(0, 40, self.width(), 40)

        # Draw dash lines
        point = 0
        qp.setPen(QPen(self.textColor))
        qp.drawLine(0, 40, self.width(), 40)
        while point <= self.width():
            if point % 30 != 0:
                qp.drawLine(3 * point, 40, 3 * point, 30)
            else:
                qp.drawLine(3 * point, 40, 3 * point, 20)
            point += 10

        if self.pos is not None and self.is_in:
            qp.drawLine(self.pos.x(), 0, self.pos.x(), 40)

        if self.pointerPos is not None:
            line = QLine(QPoint(self.pointerTimePos/self.getScale(), 40),
                         QPoint(self.pointerTimePos/self.getScale(), self.height()))
```

```
            poly = QPolygon([QPoint(self.pointerTimePos/self.getScale() - 10, 20),
                             QPoint(self.pointerTimePos/self.getScale() + 10, 20),
                             QPoint(self.pointerTimePos/self.getScale(), 40)])
        else:
            line = QLine(QPoint(0, 0), QPoint(0, self.height()))
            poly = QPolygon([QPoint(-10, 20), QPoint(10, 20), QPoint(0, 40)])

        # Draw samples
        t = 0
        for sample in self.videoSamples:
            # Clear clip path
            path = QPainterPath()
            path.addRoundedRect(QRectF(t / scale, 50, sample.duration / scale, 200), 10,
10)
            qp.setClipPath(path)

            # Draw sample
            path = QPainterPath()
            qp.setPen(sample.color)
            path.addRoundedRect(QRectF(t/scale, 50, sample.duration/scale, 50), 10, 10)
            sample.startPos = t/scale
            sample.endPos = t/scale + sample.duration/scale
            qp.fillPath(path, sample.color)
            qp.drawPath(path)

            # Draw preview pictures
            if sample.picture is not None:
                if sample.picture.size().width() < sample.duration/scale:
                    path = QPainterPath()
```

```
                path.addRoundedRect(QRectF(t/scale, 52.5, sample.picture.size().width(),
45), 10, 10)
                qp.setClipPath(path)
                qp.drawPixmap(QRect(t/scale, 52.5, sample.picture.size().width(), 45),
sample.picture)
            else:
                path = QPainterPath()
                path.addRoundedRect(QRectF(t / scale, 52.5, sample.duration/scale, 45),
10, 10)
                qp.setClipPath(path)
                pic = sample.picture.copy(0, 0, sample.duration/scale, 45)
                qp.drawPixmap(QRect(t / scale, 52.5, sample.duration/scale, 45), pic)
            t += sample.duration

        # Clear clip path
        path = QPainterPath()
        path.addRect(self.rect().x(), self.rect().y(), self.rect().width(), self.rect().height())
        qp.setClipPath(path)

        # Draw pointer
        qp.setPen(Qt.darkCyan)
        qp.setBrush(QBrush(Qt.darkCyan))

        qp.drawPolygon(poly)
        qp.drawLine(line)
        qp.end()

    # Mouse movement
    def mouseMoveEvent(self, e):
```

```
        self.pos = e.pos()

        # if mouse is being pressed, update pointer
        if self.clicking:
            x = self.pos.x()
            self.pointerPos = x
            self.positionChanged.emit(x)
            self.checkSelection(x)
            self.pointerTimePos = self.pointerPos*self.getScale()

        self.update()

    # Mouse pressed
    # Set clicking check to true

    # Mouse release
    # def mouseReleaseEvent(self, e):
    #     print(self)
    #     if e.button() == Qt.LeftButton:
    #         print(int(self.pointerTimePos * 1000))
    #
    #         self.clicking = False  # Set clicking check to false

    # Enter
    def enterEvent(self, e):
        self.is_in = True

    # Leave
    def leaveEvent(self, e):
```

```
    self.is_in = False
    self.update()

# check selection
def checkSelection(self, x):

    # Check if user clicked in video sample
    for sample in self.videoSamples:
        if sample.startPos < x < sample.endPos:
            sample.color = Qt.darkCyan
            if self.selectedSample is not sample:
                self.selectedSample = sample
                self.selectionChanged.emit(sample)
        else:
            sample.color = sample.defColor

# Get time string from seconds
def get_time_string(self, seconds):

    m, s = divmod(seconds, 60)
    h, m = divmod(m, 60)

    return "%02d:%02d:%02d" % (h, m, s)



# Get scale from length
def getScale(self):

    return float(self.duration)/float(self.width())
```

```
    # Get duration
    def getDuration(self):
        return self.duration

    # Get selected sample
    def getSelectedSample(self):
        return self.selectedSample

    # Set background color
    def setBackgroundColor(self, color):
        self.backgroundColor = color

    # Set text color
    def setTextColor(self, color):
        self.textColor = color

    # Set Font
    def setTextFont(self, font):
        self.font = font

# Опис файлу main.py
import os

from matplotlib import pyplot as plt

os.environ["CUDA_VISIBLE_DEVICES"]="-1"
from load import LoadFile
import numpy as np
```

```
from tensorflow import keras
import tensorflow as tf
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

# Клас нейронної мережі

class NN:
    def __init__(self):
        """
        Конструктор
        """
        ld=LoadFile()
        X_train, X_test, y_train, y_test = ld.data_lung()

        self.x_train = X_train / 255
        self.x_test = X_test / 255
        del X_train, X_test
        # print(y_train)
        self.y_train_cat = keras.utils.to_categorical(y_train, 2)
        self.y_test_cat = keras.utils.to_categorical(y_test, 2)
        del y_train, y_test
        self.x_train = np.expand_dims(self.x_train, axis=3)
        self.x_test = np.expand_dims(self.x_test, axis=3)

        print(self.y_train_cat)
        gpus = tf.config.experimental.list_physical_devices('GPU')
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        self._model = keras.Sequential([
```

```
            Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(108, 128,
1)),
            MaxPooling2D((2, 2), strides=2),
            Conv2D(64, (3, 3), padding='same', activation='relu'),
            MaxPooling2D((2, 2), strides=2),
            Flatten(),
            Dense(128, activation='relu'),
            Dense(2, activation='softmax')
        ])

        print(self._model.summary())  # вывод структуры НС в консоль

        self._model.compile(optimizer='adam',
                            loss='binary_crossentropy',
                            metrics=['binary_accuracy'])



    def learn(self):
        """
        Виклик цього методу почне навчання моделі
        :return:
        """
        self.his = self._model.fit(self.x_train, self.y_train_cat, epochs=3, batch_size=128,
                                   validation_data=(self.x_test, self.y_test_cat))  # ,
validation_data=(x_test,y_test_cat)



    def save(self):
```

```
    """
    Цей метод зберігає модель та виводить графік втрат
    :return:
    """
    fig, ax = plt.subplots()
    ax.plot(self.his.history["loss"], label='loss')
    ax.plot(self.his.history["val_loss"], label='val_loss')
    ax.legend()
    plt.show()
    self._model.save("lung_model30000_50000")


def use(self):
    """
    Приклад використаненя моделі
    :return:
    """
    model_loaded = keras.models.load_model("lung_model15000_25000V2")
    print(model_loaded.summary())
    t = 0
    f = 0
    print(self.x_train[:1].shape)
    p = model_loaded.predict(self.x_train)

    for i,j in zip(p,self.y_train_cat):

        if (np.around (i)==np.around(j)).all():
            t+=1
        else:
```

```
            f+=1


        print("True:{0} |   False:{1}".format(t,f))

if __name__ == '__main__':
    nn=NN()
    nn.learn()

# Модуль load.py
import cv2
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split

# Клас для завантаженя та поділу даних
class LoadFile:
    def __init__(self):

        self.cap_heal = cv2.VideoCapture(r"C:\Data set\Video\heal\healreversed.mp4")
        self.cap_ill = cv2.VideoCapture(r"C:\Data set\Video\ill\illreversed.mp4")
        self.s=25000

    def lod(self):
        """
        Завантаження даних
        :return:
        """
        frames_heal = []
```

```
    frames_ill=[]
    n=0

    while n<=15000:
        r,img = self.cap_heal.read()
        if r:
            n += 1
            # img = cv2.resize(img, (128, 108), interpolation=cv2.INTER_AREA)

            frames_heal.append(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY))
            print(n)
        else:



            break
    n=0
    while n<=35000:
        r1, img1 = self.cap_ill.read()
        if r1:
            n += 1
            # img1 = cv2.resize(img1, (128, 108), interpolation=cv2.INTER_AREA)
            frames_ill.append(cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY))
            print(n)
        else:
            break
    return frames_heal,frames_ill

def data_lung(self,frames=50000,testS=0.1):
    """
```

```
        Метод реалізує перемішування та поділ даних
        :param frames: Кількість знімків яку поверне метод
        :param testS: Розмір тестової вибірки в %
        :return: Дані для навчання та перевірки
        """
        frames_heal, frames_ill=self.lod()
        frames_heal=np.array(frames_heal,dtype=np.uint8)
        heal_dignps=np.zeros(len(frames_heal)).astype(np.uint8)
        frames_ill=np.array(frames_ill,dtype=np.uint8)
        ill_dignps=np.ones(len(frames_ill)).astype(np.uint8)
        X=np.vstack((frames_heal,frames_ill))
        del frames_heal,frames_ill
        y=np.append(heal_dignps,ill_dignps)
        del heal_dignps,ill_dignps

        X_train, X_test, y_train, y_test=train_test_split(X, y,
test_size=testS,random_state=12)
        return X_train[:frames],X_test[:frames],y_train[:frames],y_test[:frames]
```

ДОДАТОК Б

Автоматизована система контролю працездатності ресурсів Приймальної комісії КПІ ім. Ігоря Сікорського

Текст публікації

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР82_№8210_ 22Б

Аркушів 28

Київ 2022

УДК 004.4+515.

БІНАРНА КЛАСИФІКАЦІЯ МЕДИЧНИХ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ НЕЙРОННОЇ МЕРЕЖІ

Залевська О.В.,к.т.н.,
o.zalevska@kpi.ua ORCID: 0000-0002-3163-1695
Мірошниченко І.В.
goodgod@ukr.net ORCID: 0000-0001-7383-8013
Журавльов О.В.
oleg95soda@gmail.com ORCID: 0000-0003-2056-4690

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

У роботі досліджується застосування мережі глибокого навчання для бінарної класифікації знімків комп'ютерної томографії.

Коронавірусна хвороба 2019 (COVID-19) є надзвичайно заразною та має серйознимі і глобальнимі наслідками для здоров'я. Станом на 26 квітня 2022 року в усьому світі було зареєстровано 510 мільйони підтверджених захворювань, що забрали понад 6.2 мільйона життів . Основною перешкодою в боротьбі з COVID-19 є доступність своєчасного обстеження та моніторингових тестів.

Комп'ютерна томографія (КТ) зазвичай використовується в клінічній практиці для діагностики, скринінгу та лікування COVID-19 у всьому світі. Рентгенологи мусять аналізувати велику кількість знімків, що в свою чергу гальмує своєчасний діагноз, та лікування. Крім того, у багатьох слаборозвинених сільських регіонах обмежений доступ до добре підготовлених рентгенологів з достатнім знанням у сфері діагностування COVID-19.У цій роботі буде обговорюється застосування методів DL для діагностики COVID-19 та автоматизованого аналізу КТ-зображення.

В ході роботи були вирішені такі задачі:

- *Обробка вхідних даних та зведення їх до одного формату*
- *Розробка архітектури нейронної мережі для бінарної класифікації та її навчання*
- *Підготовка готової моделі до використання у зазначених цілях*

Ключові слова: штучна нейрона мережа, медицина, легені, комп'ютерна томографія

Постановка проблеми: Спалах нового коронавірусу (COVID-19) спричинив жахливу ситуацію в усьому світі та став одним із найгостріших та найважчих захворювань за останні сто років[1,2]. Рівень поширеності COVID-19 стрімко зростає з кожним днем у всьому світі. Станом на 26 квітня 2022 року в усьому світі було зареєстровано 510 мільйони підтверджених захворювань, що забрали понад 6.2 мільйона життів[3]. Хоча вакцини проти цієї пандемії поки не виявлено, методи глибокого навчання виявилися потужним інструментом в арсеналі, який використовується клініцистами для автоматичної діагностики COVID-19.

Ціль статті: Дослідити ефективність застосування нейронних мереж у цій задачі, та надати експертам (медичним чи іншим) і технікам нове уявлення, про те, як методи глибокого навчання використовуються у цьому відношенні та як вони потенційно працюють у боротьбі зі спалахом COVID-19.

Основна частина.Для проведення класифікацій була використана згорткова нейронна мережа(CNN), яка у перше була запропонована Яном Лекуном. Свою популярність дана архітектура здобула коли в 2012р. команда Алекса Крижевськи виграла конкурс ImageNet зі своєю мережею Alexnet [4]. Ідея архітектур таких мереж була підглянута у біологічної зорової системи людини [5]. Дендрити кожного нейрону з'єднуються тільки з декількома рецепторами[6].

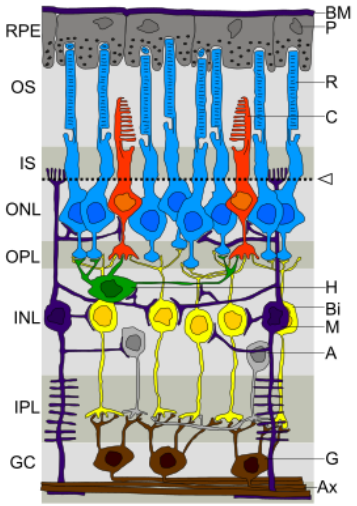

Рис. 2. Шари сітківки

Якщо спростити дану логіку та перенести її на чорно-біле зображення, можна отримати схематичну архітектуру CNN. Зазвичай згорткова нейрона мережа складається з згорткових шарів, агрегувальних шарів, повноз'єднаних шарів.

Згорткові шари намагаються імітувати реакцію нейрону на зоровий стимул. На вхід згорткового шару до кожного нейрона поступає частина матриці зображення(розмір частини визначається ядром згортки). Після проходження через згортковий шар зображення абстрагується до карти об’єктів. Суть описаної системи в тому, що група нейронів активується, тільки тоді коли, на фрагменті зображення з'являється об'єкт, який підходить під їх ядра[7].

Агрегувальний шар срямований на виділення загальних деталей на зображенні.

Повністю пов’язані шари з’єднують кожен нейрон в одному шарі з кожним нейроном в іншому шарі.

При навчанні нейронної мережі були використані 30000 знімків здорових легень, та 50000 хворих взяті з декількох датасетів[8,9,10]. Також слід зауважити, що до знімків були застосовані декілька методів по уніфікації та оптимізації ресурсів:

Знімки були зменшені в розмірі з 512× 512 пікселів до 128 ×128 пікселів.

Були обрізані нижні 20 пікселів зображення. Це рішення обгрунтоване тим що там знаходиться стіл аналіз якого не має сенсу.

Для позбавлення від шумів, зернистості та сторонніх деталей було застосоване розмивання Гауса

Також для сегментації був використаний поріг. Тобто якщо значення пікселя менше за поріг, воно встановлюється на 0, в іншому випадку встановлюється максимальне значення 255.

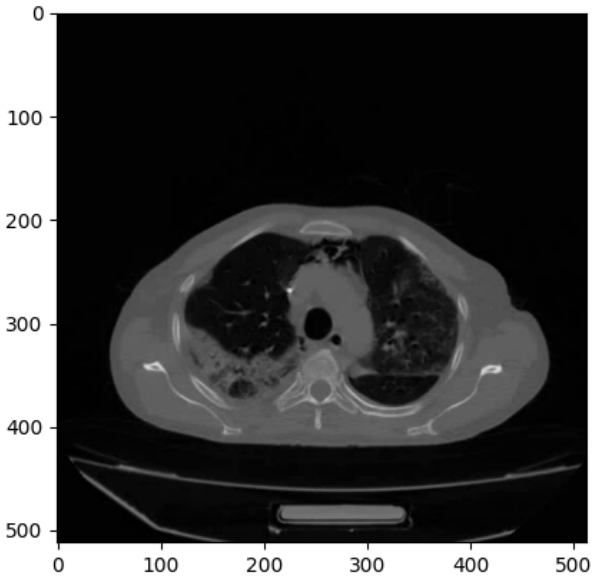

Рис. 3. Початкове зображення

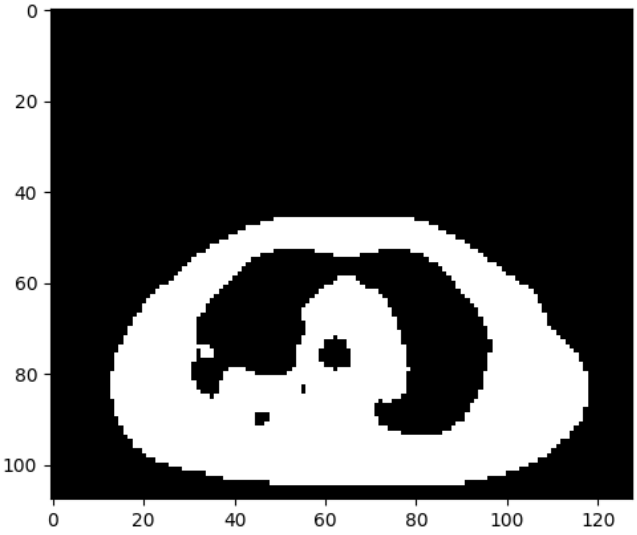

Рис. 4. Готове зображення

Загальна архітектура нейронної мережі складається з двох згорткових шарів, двох агрегувальних шарів та повнозвязної мережі.

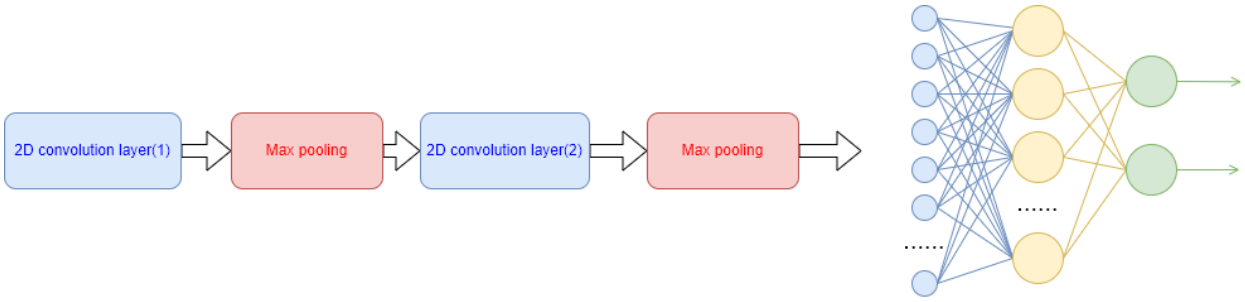

Рис. 5. Типова архітектура 2D-CNN, яка використовується для виявлення COVID-19

Параметри нейронної мережі:

- Перший згортковий шар – кількість ядер 32, розмір ядра 3×3, функція активації ReLU
- Другий згортковий шар – кількість ядер 64, розмір ядра 3×3, функція активації ReLU
- Агрегувальний шар(max pooling) – розмір вікна для якого потрібно взяти максимальне значення 2× 2
- Оптимізатор – Adam
- Функція втрат – Бінарна крос-ентропія
- Метрика – Бінарна точність

- Кількість епох – 3

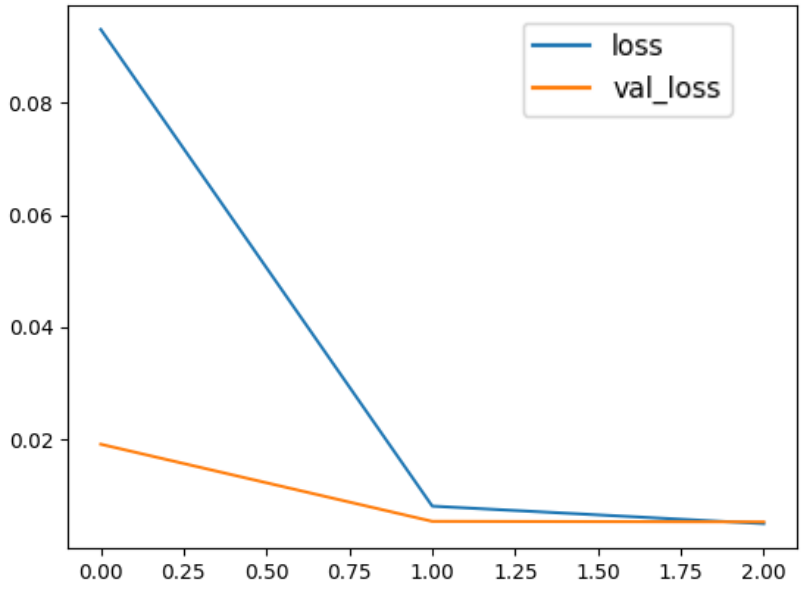

Рис. 6. Точність на кожній епосі

Таку високу точність при валідації можна пояснити тим, що для неї використовувались знімки сусідніх зрізів, тобто дуже схожі на ті на яких тренували нейронну мережу. Якщо передбачити діагноз для знімків, які ніяк не схожі на навчальну вибірку, то точність буде трохи нижчою: для здорових 99.4% ,для хворих 90.67%.

Висновок: Дослідження показало, що нейронні мережі чудово пристосовані для бінарної класифікації КТ знімків .Точність отриманої моделі можна оцінити як задовільну, але варто зазначити, що зображення дає лише часткову інформацію про пацієнтів із COVID-19. Таким чином, важливо поєднувати дані знімків як із клінічними проявами, так і з результатами лабораторних досліджень, щоб допомогти краще виявляти та діагностики COVID-19.

Посилання:

1. F. Wu et al., “A new coronavirus associated with human respiratory disease in China,” Nature, vol. 579, no. 7798, pp. 265–269.
2. D. Cucinotta and M. Vanelli, “WHO Declares COVID-19 a Pandemic.,” Acta Biomed., vol. 91, no. 1, pp. 157–160, 2020.

3. Worldometers. [Online]. Available: https://www.worldometers.info/coronavirus/
4. Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. "ImageNet classification with deep convolutional neural networks"
5. "Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation"
6. Habibi, Aghdam, Hamed . Guide to convolutional neural networks
7. Krizhevsky, Alex. "ImageNet Classification with Deep Convolutional Neural Networks"
8. Shakouri, S., Bakhshali, M.A., Layegh, P. et al. COVID19-CT-dataset: an open-access chest CT image repository of 1000+ patients with confirmed COVID-19 diagnosis. BMC Res Notes 14, 178 (2021). https://doi.org/10.1186/s13104-021-05592-x
9. https://portal.gdc.cancer.gov/
10. https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images

БИНАРНАЯ КЛАССИФИКАЦИЯ МЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННОЙ СЕТИ

Залевская О.В., Журавлев О.В., Мірошниченко І.В.

В работе исследуется использование сети глубокого обучения для бинарной классификации снимков компьютерной томографии.

Коронавирусная болезнь 2019 (COVID-19) чрезвычайно заразна и имеет серьезные и глобальные последствия для здоровья. По состоянию на 26 апреля 2022 года во всем мире было зарегистрировано 510 миллионов подтвержденных заболеваний, унесших более 6.2 миллионов жизней. Основным препятствием в борьбе с COVID-19 есть доступность своевременного обследования и мониторинговых тестов.

Компьютерная томография (КТ) обычно используется в клинической практике для диагностики, скрининга и лечения COVID-19 по всему миру. Рентгенологи должны анализировать большое количество снимков, что в свою очередь тормозит своевременный диагноз и лечение. Кроме того, во многих слаборазвитых сельских регионах ограничен доступ к хорошо подготовленным рентгенологам с достаточным знанием в области диагностики COVID-19. В этой работе будет обсуждаться применение методов DL для диагностики COVID-19 и автоматизированного анализа КТ-изображения.

В ходе работы были решены следующие задачи:
- обработка входных данных и сведение их в один формат
- разработка архитектуры нейронной сети для бинарной классификации и ее обучения

- подготовка готовой модели к использованию в указанных целях

***Ключевые слова**: искусственная нейронная сеть, медицина, легкие, компьютерная томография*

BINARY CLASSIFICATION OF MEDICAL IMAGES USING A NEURAL NETWORK

Zalevskaya O.V., Zhuravlev O.V., Miroshnichenko I.V.

The paper explores the use of a deep learning network for binary classification of computed tomography images.

Coronavirus disease 2019 (COVID-19) is highly contagious and has serious and global health implications. As of April 26, 2022, there were 510 million confirmed diseases worldwide, claiming more than 6.2 million lives. The main obstacle in the fight against COVID-19 is the availability of timely examination and monitoring tests.

Computed tomography (CT) is commonly used in clinical practice for the diagnosis, screening, and treatment of COVID-19 around the world. Radiologists have to analyze a large number of images, which in turn slows down timely diagnosis and treatment. In addition, access to well-trained radiologists with sufficient knowledge of COVID-19 diagnostics is limited in many underdeveloped rural areas. This paper will discuss the application of DL methods for COVID-19 diagnosis and automated CT image analysis.

During the work the following tasks were solved:
- processing input data and bringing them into one format
- development of neural network architecture for binary classification and its training
- preparation of the finished model for use for the specified purposes

***Keywords**: artificial neural network, medicine, lungs, computed tomography*

Referense

1. F. Wu et al., “A new coronavirus associated with human respiratory disease in China,” Nature, vol. 579, no. 7798, pp. 265–269.
2. D. Cucinotta and M. Vanelli, “WHO Declares COVID-19 a Pandemic.,” Acta Biomed., vol. 91, no. 1, pp. 157–160, 2020.
3. Worldometers. [Online]. Available: https://www.worldometers.info/coronavirus/
4. Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. "ImageNet classification with deep convolutional neural networks"

5. "Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation"
6. Habibi, Aghdam, Hamed . Guide to convolutional neural networks
7. Krizhevsky, Alex. "ImageNet Classification with Deep Convolutional Neural Networks"
8. Shakouri, S., Bakhshali, M.A., Layegh, P. et al. COVID19-CT-dataset: an open-access chest CT image repository of 1000+ patients with confirmed COVID-19 diagnosis. BMC Res Notes 14, 178 (2021). https://doi.org/10.1186/s13104-021-05592-x
9. https://portal.gdc.cancer.gov/
10. https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images