МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО ”
Факультет електроніки
Кафедра електронної інженерії

"На правах рукопису"

«До захисту допущено»
Завідувач кафедри

УДК _____

_____ В. І. Тимофєєв
“___”_____ 20___р.

Магістерська дисертація

зі спеціальністі 153 мікро- та наносистемна техніка

на тему «Методи машинного навчання аналізу рентгенівських зображень для діагностики COVID-19»

Виконав(ла) студент 2 курсу, групи ДМ-11мп

Кучер Вячеслав Васильович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доц. каф. ЕІ, к.т.н. Є.С. Карплюк _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент доц. каф. АМЕС ФЕЛ к.т.н. Попович П. В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ - 2022 року

Форма № Н-9.01

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

Факультет електроніки
Кафедра електронної інженерії
Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
за спеціальністю 153 мікро- та наносистемна техніка

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ В. І. Тимофєєв

“___”_____ 20___ р.

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

_____Кучеру Вячеславу Васильовичу_____
(прізвище, ім’я, по батькові)

1. Тема дисертації___Методи машинного навчання аналізу рентгенівських зображень для діагностики COVID-19_____
Науковий керівник _____доц. каф. ЕІ, к.т.н. Є.С. Карплюк _____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “_08_” листопада 2022 року № 4092-с
2. Строк подання студентом роботи 16.12.2022_____
3. Об’єкт дослідження _діагностичні ознаки ренгенівських зображень при ковід COVID-19_____
4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою)_Архітектура нейронних мереж; методи машинного навчання для діагностування COVID-19; (Вихідні данні – набір зображень легень хворих на COVID-19)_____
5. Перелік питань, які потрібно розробити Аналіз предметної галузі та постановка задачі, теоретичні дослідження, практичні дослідження_____
6. Перелік графічного (ілюстративного) матеріалу __Рисунки в тексті_____ пояснювальної записки, ілюстрації до захисту._____
7. Орієнтовний перелік публікацій _____
8. Консультанти розділів дисертації

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

7. Дата видачі завдання___26.10.2022_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів роботи	Примітка
1	Узгодження теми та завдання	26.10.2022-26.10.2022	
2	Ознайомлення з сучасними методами машинного навчання щодо розподіленого моніторингу пацієнта та діагностики COVID-19	26.10.2022-2.11.2022	
3	Огляд архітектури нейронних мереж	2.11.2022-9.11.2022	
4	Дослідження нейронних мереж для діагностики і розподіленого моніторингу захворювання COVID-19.	9.11.2022-16.11.2022	
5	Розробка програмних засобів автоматичного рентгенівських забражень на основі методів машинного навчання та візуалізації результатів дослідження	16.11.2022-23.11.2022	
6	Оформлення пояснювальної записки	23.11.2022-30.11.2022	

Студент _____ ___Кучер В. В._____
(підпис) (прізвище та ініціали)

Керівник роботи _____ ___Карплюк Є. С._____
(підпис) (прізвище та ініціали)

ЗМІСТ

РЕФЕРАТ

Роботу викладено на 95 сторінках, вони містять 3 розділи, 26 ілюстрацій, 1 таблиця, 68 джерел в переліку посилань та 2 додатки.

COVID-19, СИМПТОМИ, АРХІТЕКТУРА, КЛАСИФІКАЦІЯ, ВИЯВЛЕННЯ, ГЛИБОКЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, РЕНТГЕНІВСЬКІ ПРОМЕНІ, МОДЕЛЬ.

Коронавірусна інфекція (COVID-19) – інфекційне захворювання, що викликається вірусом SARS-CoV-2. У більшості хворих інфекція протікає в легкій формі або безсимптомно. Основним методом скринінгу, який використовується для виявлення COVID-19, є полімеразна ланцюгова реакція зі зворотною транскрипцією в режимі реального часу.

Метою роботи є створення інтелектуальної системи діагностування на основі штучних нейронних мереж для аналізу рентгенівських зображень при COVID-19 використовуючи методи машинного навчання.

Об’єктом дослідження є рентгенографії хворих COVID-19. Предметом дослідження є методи машинного навчання для аналізу рентгенівських зображень для діагностики COVID-19.

У першому розділі надані теоритичні відомості про інфекційне захворювання COVID-19, клінічна картина хвороби, основні симптоми, принципи моніторингу та обумовлена рентгенографія грудної клітини при захворюванні COVID-19. У другому розділі проведено аналіз методів машинного навчання, оглянуті CNN, архітектура глибинних нейронних мереж, оглянуті існуючі мережі глибокого навчання, оглянуто метод передавального навчання (transfer learning) та поставлена задача побудови класифікатора рентгенівських знімків. У третьому розділі описано програмні засоби для виконання завдання, описано набір медичних даних які будуть використані в роботі, за допомогою метода передавального навчання було перетреновано три моделі для класифікації рентгенівських знімків.

ABSTRACT

The work is presented on 95 pages, they contain 3 sections, 26 illustrations, 1 table, 68 sources in the list of references.

COVID-19, SYMPTOMS, ARCHITECTURE, CLASSIFICATION, DETECTION, DEEP LEARNING, CONVOLUTIONAL NEURAL NETWORK, COMPUTED TOMOGRAPHY, X-RAYS, MODEL, TRAINING.

In recent years, the most discussed topics are the coronavirus (in the medical community) and deep learning (in the computer vision and machine learning community). A coronavirus (also known as SARS-CoV-2), belonging to the Corona family, is a virus that causes lung infections. The virus is highly contagious, as evidenced by the exponential increase in positive cases worldwide in a short period of time with limited testing. The infection causes serious damage to the lungs, causing pneumonia with associated symptoms: sore throat, dry cough, sneezing, high fever. In addition, some patients show no symptoms, so carrier activity is a concern for health care organizations. The World Health Organization has recommended more tests to screen patients with COVID-19 to contain the spread of the virus.

A critical and important step in the fight against COVID-19 is the effective screening of infected patients so that positive patients can be isolated and treated. Currently, the main screening method used to detect COVID-19 is real-time reverse transcription-polymerase chain reaction. The test is performed on a patient's respiratory samples, and results can be available within a few hours to 2 days. Various research articles published in the journal Radiology [1,2] indicate that chest scans may be useful in detecting COVID-19. Researchers have found that the lungs of patients with symptoms of COVID-19 have certain visual features, such as matte opacities — hazy dark spots that can distinguish patients infected with COVID-19 from patients who are not infected [3,4]. The researchers

believe that a chest radiology-based system could be an effective tool for identifying, quantifying and following up on COVID-19 cases.

A detection system based on chest x-ray imaging can have many advantages over the conventional method. It can be fast, analyze multiple cases at the same time, have greater availability and, more importantly, such a system can be very useful in hospitals where there are no or limited number of test kits and resources. In addition, given the importance of radiography in the modern healthcare system, radiological imaging systems are available in every hospital, making the radiography-based approach more convenient and affordable.

The purpose of the work is to create an intelligent diagnostic system based on artificial neural networks for diagnosing COVID-19 using machine learning methods.

The object of the study is CX-rays of patients with COVID-19. The subject of research is machine learning methods for the analysis of CX-ray images for the diagnosis of COVID-19.

X-ray imaging is used to detect most chest infections, such as pneumonia, bronchitis, and bronchiolitis. Although the use of X-rays is considered non-specific for radiological findings, it will help in further management of the disease. Reverse transcription polymerase chain reaction kits are used primarily for testing patients with COVD-19. Test kits are expensive and also limited in supply. In addition, the test takes an average of 24 hours, which significantly slows down the testing process. Because test kits are limited, using X-rays may be a viable option, especially in remote and rural areas. This pandemic is causing health systems to be overwhelmed by the large number of patients. Chest X-rays can be used in conjunction with appropriate tests to quickly rule out patients who do not have COVID, ultimately reducing the burden on health care systems. Like any other pneumonia, COVID-19 shows clear markers on chest X-rays. Detection of the disease using X-ray images is an additional method of detecting covid pathologies, not a replacement for a full-fledged PCR test.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ГРДС – гострий респіраторний дистрес-синдром;

ЕКГ – електрокардіограма;

МАНК – методи ампліфікації нуклеїнових кислот;

ПЛР – полімеразна ланцюгова реакція;

РНК – рибонуклеїнова кислота;

ШІ – штучний інтелект;

CNN – Convolutional Neural Networks (згорткові нейронні мережі);

COVID-19 – назва хвороби, спричиненою коронавірусом SARSCoV-2;

CXR – Chest X-ray (рентген грудної клітки);

GPU – Graphics Processing Unit (графічний процесор);

VGG – Visual Geometry Group (групи візуальної геометрії).

ВСТУП

В останні роки найбільш обговорюваними темами є коронавірус (у медичній спільноті) і глибоке навчання (у спільноті комп'ютерного зору та машинного навчання). Коронавірус (також відомий як SARS-CoV-2), що належить до сімейства Corona, є вірусом, який викликає легеневі інфекції. Вірус дуже заразний, про що свідчить експоненціальне зростання позитивних випадків у всьому світі за короткий період часу з обмеженим тестуванням. Інфекція викликає серйозне ураження легенів, викликаючи пневмонію з супутніми симптомами: першіння в горлі, сухий кашель, чхання, висока температура. На додаток, деякі пацієнти не виявляють ознак, тому діяльність у ролі носія викликає занепокоєння для організацій охорони здоров'я. Всесвітня організація охорони здоров'я рекомендувала проводити більше тестів для скринінгу пацієнтів з COVID-19, щоб стримати поширення вірусу.

Критичним і важливим кроком у боротьбі з COVID-19 є ефективний скринінг інфікованих пацієнтів, щоб можна було ізолювати та лікувати позитивних пацієнтів. На цей момент основним методом скринінгу, який використовується для виявлення COVID-19, є полімеразна ланцюгова реакція зі зворотною транскрипцією в режимі реального часу. Тест проводиться на респіраторних зразках пацієнта, і результати можуть бути доступні протягом кількох годин до 2 днів. Різні дослідницькі статті, опубліковані в журналі Radiology [1,2] вказують на те, що сканування грудної клітини може бути корисним для виявлення COVID-19. Дослідники виявили, що на легенях пацієнтів із симптомами COVID-19 є певні візуальні ознаки, як-от матове помутніння — туманні темні плями, за якими можна відрізнити пацієнтів, інфікованих COVID-19, від пацієнтів, які не інфіковані [3,4]. Дослідники вважають, що система, заснована на радіології

органів грудної клітки, може бути ефективним інструментом для виявлення, кількісної оцінки та подальшого спостереження за випадками COVID-19.

Система виявлення на основі рентгенологічного зображення грудної клітини може мати багато переваг перед звичайним методом. Вона може бути швидкою, аналізувати кілька випадків одночасно, мати більшу доступність і, що більш важливо, така система може бути дуже корисною в лікарнях, де немає або обмежена кількість наборів для тестування та ресурсів. Крім того, враховуючи важливість рентгенографії в сучасній системі охорони здоров'я, системи радіологічної візуалізації доступні в кожній лікарні, що робить підхід, заснований на рентгенографії, більш зручним і доступним.

Метою роботи є створення інтелектуальної системи діагностування на основі штучних нейронних мереж для аналізу COVID-19 використовуючи методи машинного навчання.

Об'єктом дослідження є рентгенографії хворих COVID-19. Предметом дослідження є методи машинного навчання для аналізу рентгенівських зображень для діагностики COVID-19.

Задачами наукової дисертації є:

1. Аналітичний огляд важливих ознак COVID-19 при рентгені грудної клітки;
2. Аналіз методів машинного навчання для автоматизації аналізу рентгенівських знімків;
3. Дослідження архітектур глибоких нейронних мереж, щодо класифікації рентгенівських зображень грудної клітини при COVID-19;
4. Створення та дослідження класифікатора зображень методом передавального навчання;

У першому розділі надані теоритичні відомості про інфекційне захворювання COVID-19, клінічна картина хвороби, основні симптоми, принципи моніторингу та обумовлена рентгенографія грудної клітини при

захворюванні COVID-19. Рентгенівське зображення використовується для виявлення більшості інфекцій органів грудної клітки, таких як пневмонія, бронхіт і бронхіоліт. Хоча використання рентгенівських променів вважається неспецифічним для радіологічних знахідок, воно допоможе в подальшому лікуванні захворювання. Пандемія призводить до того, що системи охорони здоров’я перевантажені великою кількістю пацієнтів. Рентген грудної клітини можна використовувати разом із відповідними тестами, щоб швидко виключити пацієнтів, які не хворіють на COVID, що зрештою зменшить навантаження на системи охорони здоров’я.

У другому розділі проведено аналіз методів машинного навчання, оглянуті CNN, архітектура глибинних нейронних мереж, оглянуті існуючі мережі глибокого навчання, оглянуто метод передавального навчання (transfer learning) та поставлена задача побудови класифікатора рентгенівських знімків. Автоматичний аналіз рентгенівських знімків грудної клітки можна виконати за допомогою методів на основі глибокого навчання, що може прискорити час аналізу. Методи машинного навчання можуть тренувати ваги мереж на великих наборах даних, а також точно налаштовувати ваги попередньо навчених мереж на малих наборах даних. Розв'язувана даною нейронною мережею задача – класифікація зображень, а саме: класифікація рентгенівських зображень COVID-19, вірусної пневмонії, помутніння легень та зображення здорових легень

У третьому розділі описано програмні засоби для виконання завдання, описано набір медичних даних які будуть використані в роботі, за допомогою метода передавального навчання було перетреновано три моделі для класифікації рентгенівських знімків. Однією з важливих проблем у автоматичному виявленні COVID-19 на рентгенівських зображеннях є вибір відповідної архітектури глибокого навчання ШІ. Таким чином, у цьому документі оцінюється ефективність шести високоефективних і точних моделей навчання передачі (VGG-16, ResNet-50, Inception3) для раннього виявлення COVID-19 на рентгенівському знімку легень.

1 КЛІНІЧНІ ОЗНАКИ COVID-19 ДО АНАЛІЗУ РЕНТГЕНІВСЬКИХ ЗОБРАЖЕНЬ

1.1 Клінічна картина хвороби

Коронавірусна інфекція (COVID-19) – інфекційне захворювання, що викликається вірусом SARS-CoV-2. У більшості хворих інфекція протікає в легкій формі або безсимптомно [5]. У важкій формі, з необхідністю застосування кисневої терапії, протікає приблизно в 15% випадків захворювання, в 5% стан хворих критичний.

Найпоширенішими симптомами є:

– лихоманка;

– кашель;

– задишка;

– змінене відчуття смаку / запаху.

До менш поширених симптомів належать:

– втома;

– вироблення мокротиння;

– ущільнення в грудях;

– шлунково-кишкові симптоми;

– біль у горлі;

– головний біль;

– біль в грудях;

– кровохаркання.

Особливістю хвороби є двосторонні зміни по типу «матового скла», що зачіпають в основному нижні відділи легень [6]. У третини пацієнтів розвивається гострий респіраторний дистрес-синдром. Під час дистрес-синдрому може проявлятися тахікардія, збільшення частоти дихання, або

ціаноз, який супроводжується гіпоксією. Також можлива дихальна недостатність, сепсис і септичний шок.

1.1.1 Гіперзапальний синдром при COVID-19

При COVID-19 визначалися підвищені рівні деяких цитокінів. Це стосується рівня прозапальних цитокінів, що є одним з основних маркерів наявності цитокинового шторму.

Гіперцитокінемія або цитокіновий шторм — це явище вироблення організмом надмірної кількості імунних клітин, які виділяють цитокіни. Надлишок цитокінів може призвести до запалення в легенях, яке у важких випадках може призвести до летального результату. Явище полягає в неконтрольованій активації цитокінами імунних клітин у вогнищі запалення і вивільненні останніми нової порції цитокінів, внаслідок наявності прямого зв'язку між цими процесами. Це викликає руйнування тканин вогнища запалення, одночасно реакція поширюється на сусідні тканини й по мірі розвитку набуває системного характеру, охоплюючи весь організм в цілому.

Цитокіни є частиною здорової імунної системи. Ці маленькі білки допомагають контролювати ріст і активність клітин крові та імунних клітин. Цитокіни змушують вашу систему виконувати свою роботу. Але коли вивільняється занадто багато цитокінів, це може спричинити перенапруження імунної системи, що призведе до цитокінового шторму.

В подальшому синдром цитокінового шторму в важких випадках COVID-19 отримав назву гіперзапального синдрому, пов'язаного з COVID-19. Одним з можливих підходів для раннього діагностування цитокинового шторму серед пацієнтів з COVID-19 є виявлення фібрильних пацієнтів з гіперферитинемією. Однак через низький рівень цитокінів у порівнянні з

іншими синдромами цитокинового шторму, але схожими рівнями деяких нецитокінових біомаркерів, системне запалення явно відрізняється від інших синдромів цитокинового шторму і розглядання запального процесу як результату цитокинового шторму може виявитися невірним [7].

Вплив цитокінового шторму на легені показано на рисунку 1.1.Легені є основною мішенню цитокінового шторму, який може бути спровокований важким гострим респіраторним синдромом коронавірусу. У деяких пацієнтів SARS-CoV-2 сприяє дисфункціональній імунній відповіді, яка порушує регуляцію секреції цитокінів. Гіперцитокінемія лежить в основі гіперзапального стану, що призводить до пошкодження епітеліальних клітин альвеол і ендотеліальних клітин судин, а також до легеневої інфільтрації, що підтримується нейтрофілами та макрофагами. У такому патогенному контексті інтерлейкін-6 та інші цитокіни відіграють ключову прозапальну роль.[8]

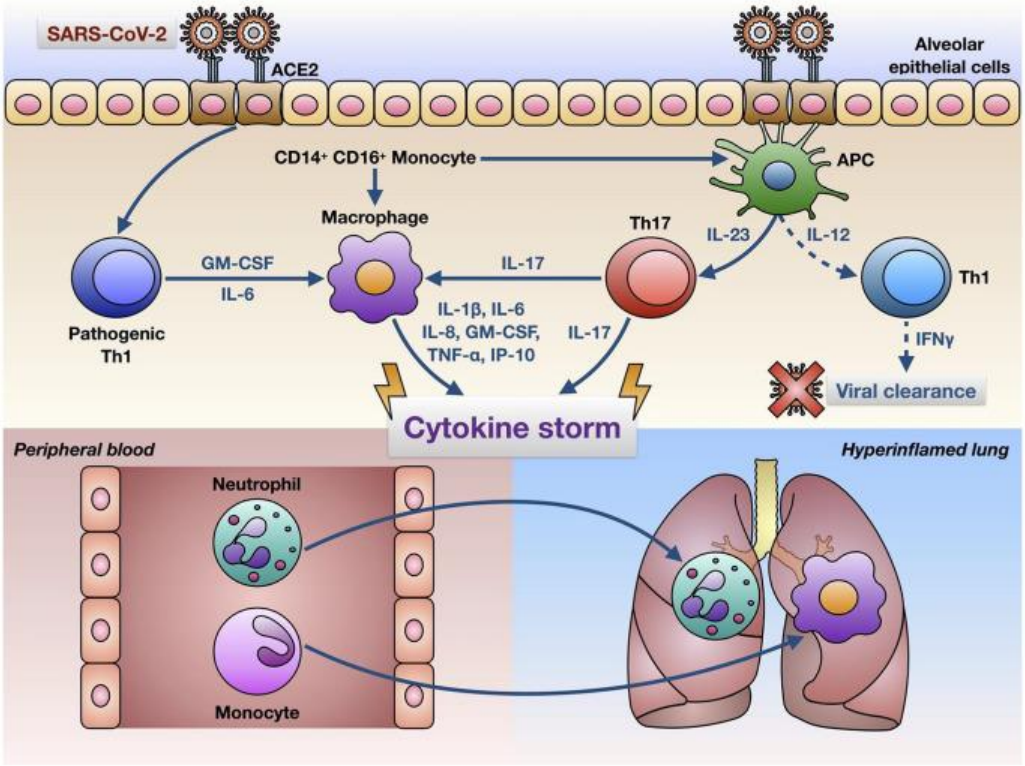

Рисунок 1.1 - Гіпотетичні механізми, що лежать в основі цитокінового шторму, спричиненого SARS-CoV-2 в інфікованих легенях. SARS-CoV-2 проникає в клітини-мішені (наприклад, клітини альвеолярного епітелію)

через взаємодію з рецептором, таким чином запускаючи складну імунну відповідь, що характеризується активацією патогенних клітин, моноцитів, альвеолярних макрофагів і лімфоцитів

Специфічні симптоми можуть відрізнятися залежно від того, на які системи організму впливає цитокіновий шторм. Вплив на серце - підвищення частоти серцевих скорочень, зниження функції серця або нерегулярне серцебиття. Вплив на легені - розвиток кашлю або задишка. Цитокіновий шторм може впливати на кілька систем органів одночасно, що може призвести до смерті.

1.1.2 Вплив на органи дихання

Через патогенність вірусу, яка у важких випадках дуже схожа на гостру респіраторну вірусну інфекцію, коронавірус має сильний негативний вплив на дихальну систему. Більшість людей з COVID-19 мають лише легкі побічні ефекти або взагалі їх не мають, у середньому 14% важких захворювань вимагають госпіталізації та кисневої підтримки, а 5% потребують госпіталізації [9]. У важких випадках COVID-19 може бути пов’язаний із гострою респіраторний дистрес-синдром, сепсис і септичний шок, гостра ниркова недостатність і серцева недостатність [10]. Серед людей похилого віку кількість людей, які потребують госпіталізації та респіраторної підтримки, різко зросла, а рівень смертності серед людей старше 80 років, як повідомляється, становить приблизно 21 через важку пневмонію [11]. У більшості випадків, захворювання протікає як лихоманка або без будь-яких респіраторних симптомів; однак через пошкодження легеневої тканини у всіх

пацієнтів пізніше з’являються легеневі аномалії різного ступеня. Ці випадки спостерігалися при скануванні легенів [12].

COVID-19 може спричинити легеневі ускладнення, такі як пневмонія та, у найважчих випадках, гострий респіраторний дистрес-синдром. Сепсис, ще одне можливе ускладнення COVID-19, також може завдати тривалої шкоди легеням та іншим органам. Новіші варіанти коронавірусу також можуть спричинити більше захворювань дихальних шляхів, таких як бронхіт, які можуть бути досить серйозними, щоб вимагати госпіталізації.

1.1.2.1 Пневмонія при COVID-19

При пневмонії легені наповнюються рідиною і запалюються, що призводить до утрудненого дихання. У деяких людей проблеми з диханням можуть стати настільки серйозними, що потребують лікування в лікарні за допомогою кисню або навіть апарату штучної вентиляції легень.

Пневмонія, яку викликає COVID-19, має тенденцію охоплювати обидві легені. Повітряні мішки в легенях наповнюються рідиною, обмежуючи їх здатність приймати кисень і викликаючи задишку, кашель та інші симптоми.

Хоча більшість людей одужують від пневмонії без тривалого ураження легенів, пневмонія, пов’язана з COVID-19, може бути важкою. Навіть після того, як хвороба пройшла, пошкодження легенів може призвести до утрудненого дихання, для покращення якого можуть знадобитися місяці.

1.1.2.2 COVID-бронхіт

Бронхіт - це загальний термін для запалення клітин, що вистилають трубки, які несуть повітря в легені та з них. Основним симптомом є сильний кашель, який швидко. Це може початися як сухий надривний кашель, але зазвичай ви починаєте помічати слиз.

Наявність бронхіту не підвищує ризик зараження новим коронавірусом. Але через запалення легенів, спричинене бронхітом, цей стан — особливо хронічний бронхіт — може підвищити ризик виникнення більш серйозних ускладнень, при COVID-19.

При бронхіті запалюється епітеліальна оболонка бронхів. Ці трубки транспортують повітря до легенів і з них. Запалення та пошкодження цієї оболонки можуть послабити клітинний бар'єр, який допомагає захистити легені.

Вироблення слизу в набряклих дихальних шляхах також може блокувати волосоподібні виступи у легенях від вимітання мікробів і сміття з дихальних шляхів.

Пацієнти можуть відчувати кашель, який залишається з ними протягом кількох місяців після первинної інфекції. Цей частий кашель і постійна закладеність грудей можуть вплинути на якість життя [13].

1.1.2.3 Гострий респіраторний дистрес-синдром

Якщо пневмонія COVID-19 прогресує, більше повітряних мішків може наповнитися рідиною, що витікає з крихітних кровоносних судин у легенях. Згодом з'являється задишка, що може призвести до гострого респіраторного дистрес-синдрому, форми легеневої недостатності. Пацієнти часто не можуть дихати самостійно, і їм може знадобитися штучна вентиляція легенів, щоб допомогти циркулювати кисень в організмі.

Незалежно від того, чи відбувається це вдома чи в лікарні, ГРДС може бути смертельним. Люди, які перенесли ГРДС і одужали від COVID-19, можуть мати тривалі легеневі рубці.

1.1.2.4 Сепсис

Сепсис є серйозним потенційно смертельним клінічним синдромом і є однією з основних причин смерті, пов'язаної з інфекцією.

Незважаючи на те, що патогенез COVID-19 повністю не пояснений, дані, отримані на даний момент у госпіталізованих пацієнтів, показали, що рівні цитокінів і хемокінів у сироватці крові у важких пацієнтів із COVID-19 високі, подібні до тих, які виявляються при сепсисі. COVID-19 може вражати кілька систем органів. Окрім легенів, вірус виділено з крові, сечі, фекалій, печінки та жовчного міхура. Результати серії розтинів у пацієнтів з COVID-19 продемонстрували широкий спектр знахідок, включаючи ураження судин, застій, та крововилив, а також дифузне альвеолярне пошкодження легеневої тканини, що відповідає гострому респіраторному дистрес-синдрому. Вірусний сепсис має деякі подібності, але також і деякі відмінності порівняно з бактеріальним сепсисом.

При бактеріальному сепсисі системне запалення, що вражає кілька органів, є більш домінуючим, ніж при сепсисі COVID-19. У той час як бактеріальний сепсис викликає раннє та раптове клінічне погіршення, вірусні захворювання можуть виявляти відносно пізній початок і хронічний перебіг. Розгляд важкої хвороби COVID-19 як синдрому сепсису є актуальним і може допомогти у визначенні лікування, яке буде модулювати імунну відповідь, обмежувати внутрішні пошкодження тканин і органів і потенційно покращувати результат. Сепсис виникає, коли інфекція досягає і поширюється через кровотік, викликаючи пошкодження тканин [14].

Легені, серце та інші системи організму працюють разом, як інструменти в оркестрі. При сепсисі взаємодія між органами розпадається. Цілі системи органів можуть почати відключатися одна за одною, включаючи легені та серце. Сепсис може призвести до тривалого ураження легень та інших органів.

1.2 Діагностичні біомаркери і показники

Зазвичай легені є органами, на які переважно впливає SARS-CoV-2, через їхню велику та сильно васкуляризовану площу поверхні [15]. Патогенез COVID-19 у легенях включає початкову фазу локального запалення, пошкодження ендотеліальних клітин і антифібринолітичну активацію у верхніх і нижніх дихальних шляхах з наступними механізмами відновлення, які можуть викликати відновлення нормальної архітектури легень. Запалення супроводжується залученням тромбоцитів із дегрануляцією, утворенням тромбу, зміною проникності судин і накопиченням лейкоцитів у місці пошкодження, що призводить до залучення інших запальних клітин із залученням специфічних цитокінів, які також відповідають за профіброзну активність [16].

Легенева інфекція SARS-CoV-2 викликає широкий спектр клінічних проявів і симптомів, від безсимптомного, легкого та помірного перебігу захворювання до важкого COVID-19. Важкі та критичні захворювання становлять до 14% і 5% випадків відповідно, ГРДС зустрічається у 10-20% пацієнтів; може наступити поліорганна недостатність і смерть [17,18]. За допомогою комп’ютерної томографії [19,20] було виявлено різні фенотипи, включаючи фенотип L або 1, який характеризується низькою податливістю, зміненою вентиляцією та перфузією та шунтуванням з вогнищевими

гіпо/гіперперфузійними матовими помутніннями. Прогресуюча еволюція COVID-19 [21] може призвести до фенотипу F, спричиненого механічним розтягуванням легеневих епітеліальних клітин і патологічною фіброзною проліферацією та ремоделюванням позаклітинного матриксу, з підвищеною експресією профіброзних маркерів, як це в основному характерно для важких форми захворювання легень [22].

Незважаючи на те, що вони не є специфічними для легеневих захворювань, було виявлено декілька біомаркерів різних стадій ураження легенів при COVID-19, які пов’язані з легеневим і системним гіперзапаленням і фіброзним пошкодженням. На ранніх стадіях захворювання нейрон-специфічну енолазу можна використовувати для диференціації пацієнтів, у яких може розвинутися задишка [23]. Під час госпіталізації вища кількість лімфоцитів і тромбоцитів і нижчий рівень феритину, D-димеру, лактатдегідрогенази і аспартат-трансамінази асоціювалися з нижчим ризиком смертності у пацієнтів з COVID-19, які зрештою потребували інтубації. Рівні поверхнево-активного протеїну D, ангіопоетину-2, тригерного рецептора, що експресується на мієлоїдних клітинах (TREM)-1, і TREM-2 вищі при легкій/помірній та важкій/критичній пневмонії COVID-19, ніж у безсимптомних і неускладнених випадках. Крім того, ці біомаркери добре корелювали з клінічною тяжкістю [24,25]. У важких випадках COVID-19 загальний тіол, феритин були ідентифіковані як прогностичні біомаркери розвитку ГРДС [26]. Під час екстубації ті, хто пережив COVID-19, мали вищу кількість тромбоцитів і співвідношення нейтрофілів до лімфоцитів і нижчі С-реактивний білок [27].

1.3 Вимоги до діагностики пацієнта при захворюванні COVID-19

Загалом діагностика COVID-19 проводиться за допомогою принаймні одного з трьох тестів.

Оцінка на основі комп'ютерної томографії полягає в аналізі 3D-рентгенографічних зображень під різними кутами. Обладнання, необхідне для цієї оцінки, недоступне в більшості лікарень, і вона займає більше 15 хвилин на пацієнта.

Тест полімеразної ланцюгової реакції зворотної транскрипції: він виявляє РНК вірусу з мокротиння або мазка з носоглотки. Для цього потрібні спеціальні матеріали та обладнання, які є важкодоступними, і це займає щонайменше 12 годин, що небажано, оскільки позитивні пацієнти з COVID-19 повинні бути ідентифіковані та відстежені якнайшвидше. У деяких дослідженнях було виявлено, що результати кількох тестів у різних точках у одних і тих самих пацієнтів були різними протягом хвороби, спричиняючи високий рівень хибнонегативних результатів.

Рентген грудної клітки: Обладнання, необхідне для цієї оцінки, є менш громіздким і може бути легким і транспортабельним. Загалом цей тип ресурсів більш доступний, ніж необхідний для тестів ПЛР і КТ-сканування. Крім того, CXR-тест займає близько 15 секунд для кожного пацієнта, що робить CXR одним із найбільш ефективних за часом і витратами інструментів оцінки.

1.3.1 Точність у діагностиці COVID-19 на основі КТ, ПЛР та CXR

Дослідження на групі з 51 пацієнта з КТ грудної клітки та ПЛР, виконане протягом 3 днів, показало точність КТ 98% порівняно з ПЛР точністю 71% [31]. Інше дослідження за участю 64 пацієнтів повідомило про точність 69% для CXR порівняно з 91% для початкової ПЛР [32]. Згідно з аналізом 636 амбулаторних пацієнтів [33], більшість пацієнтів, які

звертаються до центрів невідкладної допомоги з підтвердженою коронавірусною хворобою 2019, мають нормальні або помірно відхилені дані рентгенографії. Лише 58,3% цих пацієнтів отримують правильний діагноз.

На практиці пацієнт, класифікований експертом-радіологом як нормальний, може мати позитивний ПЛР. На цій оцінці базується експертна анотація, прийнята в даній роботі.

Автоматизований аналіз зображень за допомогою моделей глибокого навчання має великий потенціал для оптимізації ролі CXR-зображень для швидкої діагностики COVID-19. Надійна та точна модель може служити методом сортування та підтримкою для прийняття медичних рішень. Все більша кількість останніх робіт стверджує, що досягається висока точність >95%, що значно перевищує точність експертів-радіологів. Отримані моделі також не матимуть клінічної цінності, оскільки вони не зможуть виявити пацієнтів із низьким та середнім ступенем тяжкості, які є ціллю системи клінічного сортування. Зважаючи на цю ситуацію, все ще існує величезна потреба у високоякісних наборах даних, створених в тісній співпраці з експертами-рентгенологами.

1.4 Вигляд інфікування COVID-19 на знімку грудної клітини

Розглянемо один підтверджений випадок зараження COVID-19 з Китаю.

Жінка з провінції Сичуань звернулась до лікарні з гарячкою та ознобом. У її родині не було контактів з хворими, але вона наголосила про переліт на літаку за 5 днів до появи симптомів з Лондона, Англія, до Ченду, Китай. Рентгенограма грудної клітки (рис. 1.2) і комп’ютерна томографія грудної клітки (рис. 1.3, 1.4) під час пред’явлення показали плямисті матові

помутніння правої нижньої частки. Контрольні КТ-зображення грудної клітини (рис. 1.5, 1.6), отримані через 2 дні, показали збільшення помутніння матового скла з розвитком субплевральних криволінійних ліній [28].

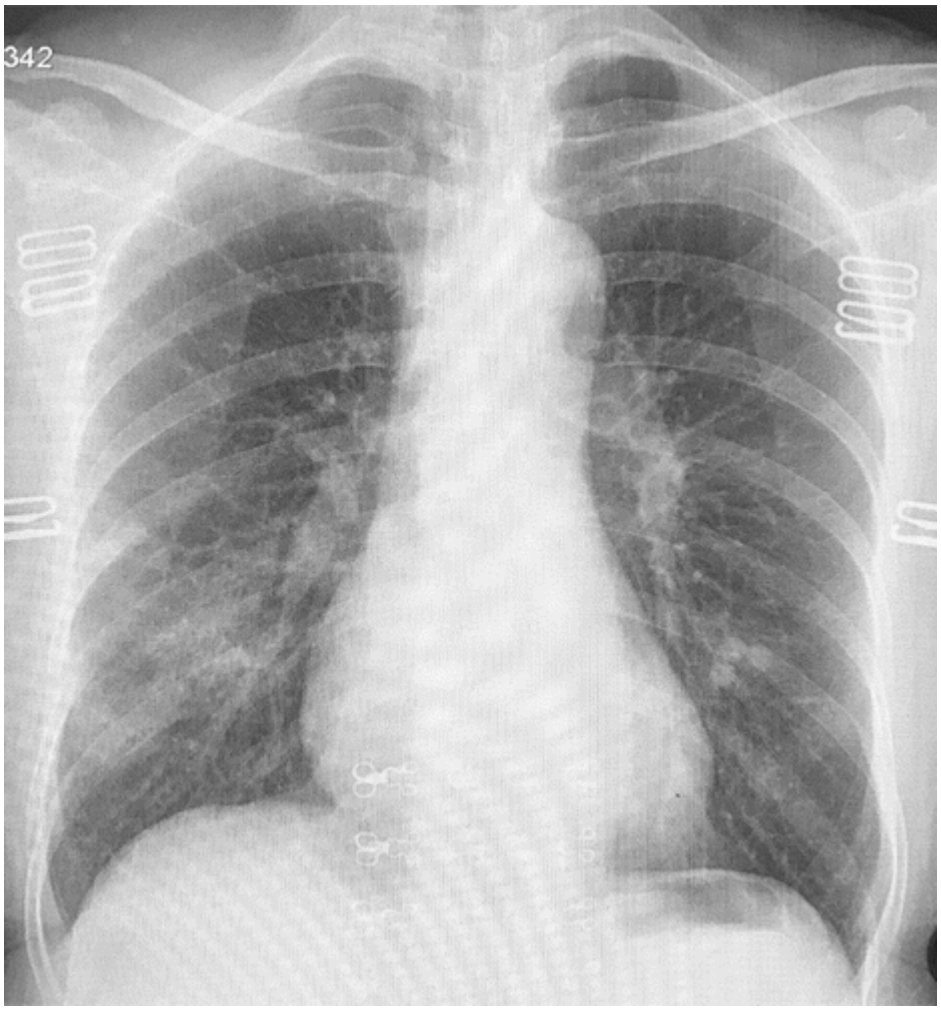

Рисунок 1.2 - Рентгенограма грудної клітки у пацієнта з інфекцією COVID-19 демонструє помутніння правого підгорбкового простору

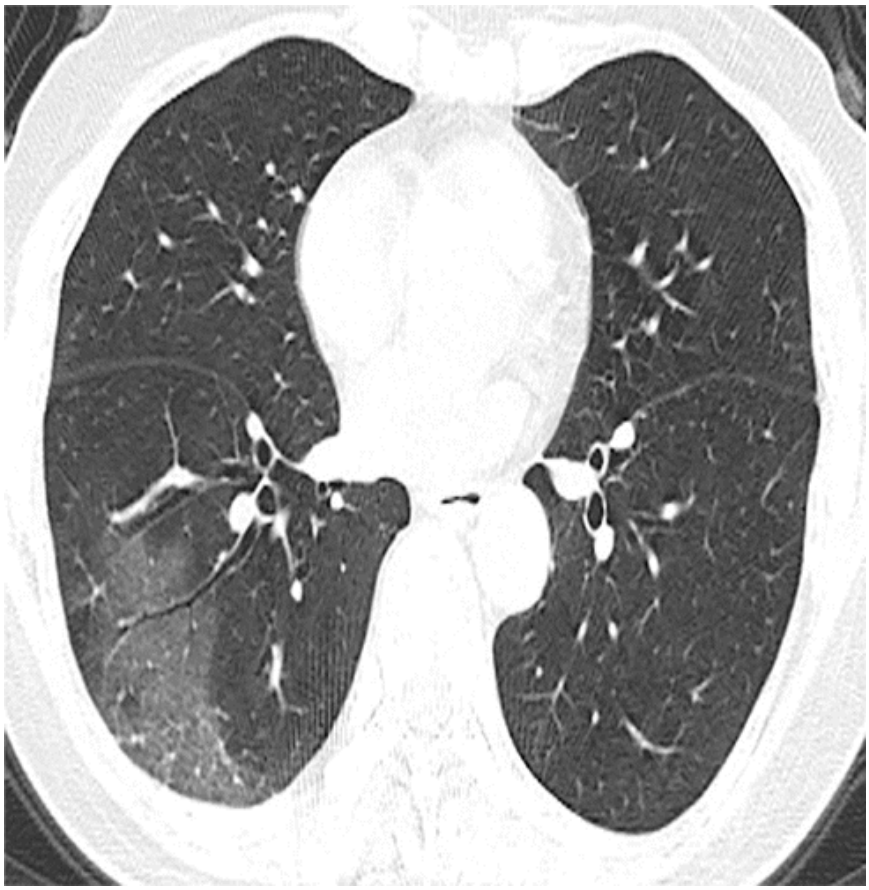

Рисунок 1.3 - Аксіальна КТ грудної клітки демонструє помутніння периферичної правої нижньої частки

Рисунок 1.4 - Сагітальне КТ грудної клітки демонструє помутніння периферичної правої нижньої частки

Рисунок 1.5 - Контрольне аксіальне КТ грудної клітки через 2 дні що показує збільшення ступеня помутніння матового скла з більшою кількістю субплевральних криволінійних ліній (стрілки)

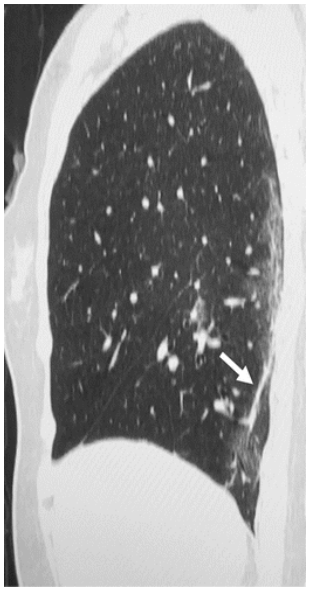

Рисунок 1.6 - Контрольне сагітальне КТ грудної клітки через 2 дні показують збільшення ступеня помутніння матового скла з більшою кількістю субплевральних криволінійних ліній (стрілки)

Найбільша серія випадків візуалізації грудної клітки у 21 пацієнта описувала багаточасткове залучення та округлі, а також периферичні помутніння повітряного простору [29]. Найбільш поширені описувані непрозорості мають матове скло (57%) і змішане розрідження (29%), подібно до представленого випадку. Враховуючи переважання матових помутнінь, КТ грудної клітки є більш чутливою, і знахідки можуть бути пропущені на рентгенограмах. Примітно, що підтверджені випадки також можуть мати нормальну КТ грудної клітки [29].

1.5 Висновки до розділу 1

Коронавіруси отримали свою назву за свій коронoподібний вигляд через поверхневі шипи та поділяються на чотири основні групи (альфа, бета, гамма та дельта). Більшість коронавірусів вражають тварин, хоча вони є зоонозними і можуть передаватися між тваринами та людьми. На сьогодні відомо сім типів коронавірусу людини. Зазвичай вони призводять до легкого захворювання, але за останні 2 десятиліття відомо, що два коронавіруси людини викликають важкі респіраторні захворювання та навіть смерть. Ці віруси включали коронавірус важкого гострого респіраторного синдрому, який вперше було виявлено в Китаї в 2002 році, і коронавірус близькосхідного респіраторного синдрому, про який вперше було повідомлено в 2012 році в Саудівській Аравії. COVID-19 є останнім доповненням до цієї групи.

Рентгенівське зображення використовується для виявлення більшості інфекцій органів грудної клітки, таких як пневмонія, бронхіт і бронхіоліт. Хоча використання рентгенівських променів вважається неспецифічним для радіологічних знахідок, воно допоможе в подальшому лікуванні захворювання. Набори для полімеразної ланцюгової реакції зворотної

транскрипції використовуються в основному для тестування пацієнтів із COVD-19. Тестові набори дорогі, а також обмежені в постачанні. Крім того, час виконання тесту в середньому становить 24 години, що значно сповільнює процес тестування. Оскільки набори тестів обмежені, використання рентгенівського випромінювання може бути життєздатним варіантом, особливо у віддалених і сільських районах. Ця пандемія призводить до того, що системи охорони здоров'я перевантажені великою кількістю пацієнтів. Рентген грудної клітини можна використовувати разом із відповідними тестами, щоб швидко виключити пацієнтів, які не хворіють на COVID, що зрештою зменшить навантаження на системи охорони здоров'я. Як і будь-яка інша пневмонія, COVID-19 демонструє чіткі маркери на рентгенівських знімках грудної клітки. Виявлення захворювання за допомогою рентгенівських зображень являється додатковим методом виявлення патологій ковід, не заміна повноцінному ПЛР тесту.

2 МЕТОДИ МАШИННОГО НАВЧАННЯ АНАЛІЗУ РЕНТГЕНІВСЬКИХ ЗОБРАЖЕНЬ

2.1 Загальний підхід побудови діагностичного класифікатора на основі технологій машинного навчання

Клінічні симптоми, пов’язані з COVID-19, включають лихоманку, сухий кашель, задишку та пневмонію. Медична візуалізація легенів може бути відповідним допоміжним методом діагностики COVID-19, оскільки вона використовує доступні медичні технології та клінічні обстеження. Рентгенографія грудної клітки і комп’ютерна томографія грудної клітки є найпоширенішими методами візуалізації легенів, які доступні в більшості лікарень у всьому світі. Різні тканини тіла різною мірою поглинають рентгенівське випромінювання, що призводить до отримання зображень у градаціях сірого, які дозволяють виявляти аномалії на основі контрасту зображень. КТ відрізняється від звичайної рентгенографії тим, що має чудовий контраст тканини з різними відтінками сірого. КТ-зображення проходять цифрову обробку для створення тривимірного зображення тіла. Однак КТ обстеження є дорожчим, ніж CXR обстеження. Недавні дослідження показали, що використання зображень CXR та КТ призвело до покращення діагностичної чутливості для виявлення COVID-19 [30]. Інтерпретація медичних зображень займає багато часу, трудомістка і часто суб'єктивна. Медичні зображення спочатку коментуються експертами для створення звіту про результати рентгенографії. Згодом результати рентгенографії аналізуються та клінічні фактори розглядаються для встановлення діагнозу. Однак під час поточної пандемії лікарі-експерти стикаються з величезним навантаженням і браком часу, що збільшує фізичне та психологічне навантаження на персонал і може негативно вплинути на ефективність діагностики.

Оскільки сучасні лікарні мають передові цифрові технології зображення, методи обробки медичних зображень можуть мати потенціал для швидкої та точної діагностики COVID-19, щоб зменшити навантаження на експертів.

Виходячи з цього, доцільно – побудувати класифікатор, що може відрізнити рентгенівське зображення здорової людини від хворої COVID-19.

Задача класифікації — формалізована задача, яка містить множину об'єктів, розділених певним чином на класи. Ця множина називається вибіркою. Класифікувати об’єкт — означає, вказати номер чи назву класу, якому належить цей об’єкт. Класифікація об'єкта — номер або найменування класу, що видається алгоритмом класифікації в результаті його застосування до цього об'єкта.

На меті маємо задачу класифікації рентгенівських зображень, класами будуть виступати зображення хворих на COVID-19 та зображення здорових людей. Для покращення якості класифікації та додаткової діагностичної цінності, збільшимо вибірку – додамо ще два класи зображень хворих: вірусна пневмонія та помутніння легень.

Методи глибокого навчання, особливо згорткові нейронні мережі, є ефективними підходами до навчання з використанням багатошарових нейронних мереж і забезпечили чудові рішення для багатьох проблем у класифікації зображень, виявленні об’єктів, а також оброці мови. Глибока залишкова мережа — це тип архітектури CNN, який використовує стратегію пропуску з’єднань, щоб уникнути деградації моделей. Однак застосування глибокого навчання для клінічних діагнозів залишається обмеженим через відсутність інтерпретації моделі глибокого навчання та мультимодальних властивостей клінічних даних. Деякі дослідження продемонстрували чудову ефективність методів глибокого навчання для виявлення раку легенів за допомогою КТ-зображень [34], пневмонії за допомогою CXR-зображень [35] та діабетичної ретинопатії за допомогою фотографій очного дна сітківки [36].

2.2 Згорткова нейронна мережа

Згорткова нейронна мережа — це алгоритм глибокого навчання, який може сприймати вхідне зображення, призначати важливість (вагові значення та зміщення) різним аспектам чи об’єктам зображення та мати можливість відрізнити один від іншого. Попередня обробка, необхідна в CNN, набагато менша порівняно з іншими алгоритмами класифікації. У той час як у примітивних методах фільтри розробляються вручну, після достатнього навчання, CNN мають можливість вивчати ці фільтри/характеристики.

Згортка полягає в тому, що розглядається область вхідних даних, наприклад 3x3, і є деяке ядро згортки – матриця такого ж розміру, як і ділянка даних, що розглядається. Кожен такий фрагмент вхідних даних множиться на матрицю згортки поелементно, після цього всі отримані елементи складаються. Кожен нейрон працює зі своїм фрагментом даних, який може частково перетинатися з фрагментом сусіднього нейрона. Потім в нейроні до отриманого значення може застосовуватись певна нелінійна функція активації.

Моделі згорткових нейронних мереж показали кращу продуктивність у різних сферах застосування, таких як сільське господарство, промисловість і діагностика медичних захворювань. Архітектура типової CNN показана на рис. 2.1. Вона складається з трьох рівнів, де перший рівень називається згортковим, другий — об’єднаним, а останній — так званим повністю підключеним. Рівні згортки та об’єднання відповідають за вивчення моделі, а повністю зв’язаний рівень виконує класифікацію [37].

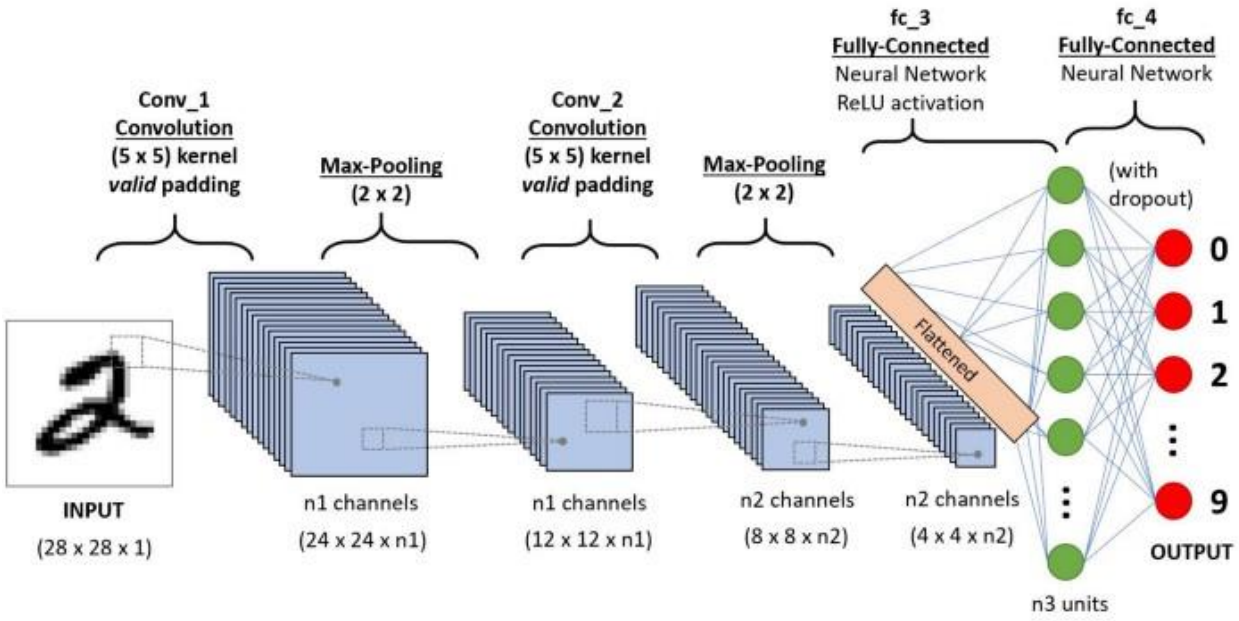

Рисунок 2.1 - Архітектура згорткових нейронних мереж

Архітектура CNN аналогічна структурі підключення нейронів у людському мозку. Окремі нейрони реагують на стимули лише в обмеженій області поля зору, відомої як рецептивне поле. Набір таких полів перекривається, щоб охопити всю візуальну область.

Класифікатори у машинному навчанні — це алгоритми, які автоматично впорядковують або класифікують дані в один або декілька з наборів класів. Одним із найпоширеніших прикладів є класифікатор електронної пошти, який сканує електронні листи, фільтруючи їх за міткою класу: спам чи не спам.

Алгоритми машинного навчання допомагають автоматизувати завдання, які раніше потрібно було виконувати вручну. Вони можуть заощадити величезну кількість часу та грошей і зробити бізнес ефективнішим.

Класифікатор — це сам алгоритм який використовуються машинами для класифікації даних. Модель класифікації, з іншого боку, є кінцевим результатом машинного навчання класифікатора. Модель навчається за допомогою класифікатора, щоб модель, зрештою, класифікувала дані.

Існують керовані і некеровані класифікатори. Некеровані класифікатори машинного навчання передають лише немарковані набори даних, які вони класифікують відповідно до розпізнавання шаблонів або структур і аномалій у даних. Керовані та напівкеровані класифікатори отримують навчальні набори даних, з яких вони вчаться класифікувати дані відповідно до заздалегідь визначених категорій.

Аналіз настроїв є прикладом керованого машинного навчання, де класифікатори навчаються аналізувати текст на полярність думок і виводити текст у клас: позитивний, нейтральний або негативний.

Класифікатори машинного навчання використовуються для автоматичного аналізу коментарів клієнтів із соціальних мереж, електронних листів, онлайн-оглядів тощо, щоб дізнатися, що клієнти говорять про бренд.

Класифікатори машинного навчання виходять за рамки простого відображення даних, дозволяючи користувачам постійно оновлювати моделі новими навчальними даними та адаптувати їх до мінливих потреб. Наприклад, безпілотні автомобілі використовують алгоритми класифікації для введення даних зображення в категорію; будь то знак «стоп», пішохід чи інший автомобіль, постійно навчаючись і вдосконалюючись з часом.

Алгоритми класифікації дозволяють автоматизувати завдання машинного навчання. Вони дозволяють навчити моделі штучного інтелекту відповідно до потреб, мови та критеріїв, працюючи набагато швидше та з більшим рівнем точності, ніж люди.

2.3 Архітектура глибинних нейронних мереж

Розглянемо традиційні мережі глибокого навчання, щоб спостерігати за їхніми можливостями впоратися з хворобою. COVNet [38] використовує структуру 3D глибокого навчання для вилучення 2D локальних і 3D функцій

для виявлення COVID-19. ResNet [39] використовується як основа для вилучення характеристик із вхідних зрізів КТ. Попередні дослідження показали успішне застосування методології глибокого навчання до рентгенівських знімків грудної клітки для діагностики бактеріальних і вірусних інфекцій [40,41].

Для виявлення та локалізації уражень, що викликають COVID-19, була запропонована система діагностики КТ на основі глибокого навчання, яка називається DeepPneumonia [42]. Спочатку область легенів виділяється на кожному КТ-зображенні, а потім передається в нейронну мережу вилучення деталей для створення K найвищих деталей у КТ-сканері за допомогою попередньо навченого ResNet із мережею піраміди характеристик. Модуль агрегації агрегує прогнози для прогнозування діагнозу на рівні пацієнта.

Глибоке навчання вимагає значної кількості анотованих даних для навчання моделі. Тривимірна глибока згорточна нейронна мережа під назвою DeCoVNet [43] використовується для введення об’ємів КТ і тривимірних легеневих масок для виведення ймовірності COVID-19 чи відсутності COVID-19. Попередньо підготовлена модель використовується для створення 3D-маски легенів. Перший етап архітектури складається з 3D-згорткової бази, за якою йде пакетна нормалізація та максимальне об’єднання для створення 3D-карти функцій. На другому етапі тривимірна карта ознак пропускається через два тривимірні залишкові блоки з пакетною нормою. На останньому етапі прогресивний класифікатор поступово абстрагує інформацію в 3D-томах і класифікує її за допомогою функції softmax, щоб вивести ймовірність COVID-19 або не COVID-19.

Рентгенографія органів грудної клітки широко використовується для діагностики інфекцій через її нижчу вартість і більш широку доступність.

2.4 Огляд існуючих мереж глибокого навчання

Оглянемо основні блоки поточних найсучасніших архітектур глибокого навчання для завдань класифікації зображень. Згорткові нейронні мережі зазвичай використовуються для класифікації зображень і можуть визначати потужні загальні характеристики із зображення, застосовуючи фільтри згортки. У CNN параметри фільтра навчаються за допомогою зворотного поширення, де функції низького рівня, такі як ребра, визначаються на нижніх рівнях архітектури, а функції високого рівня, такі як форми, виявляються на більш глибоких рівнях мережі. ImageNet Large-Scale Visual Recognition Challenge [44] відіграв значну роль у виборі структури дизайну сучасної архітектури CNN. Нові архітектури повинні мати можливість наскрізного навчання та вивчати багатомасштабні функції з меншою кількістю параметрів і меншими розмірами моделі. Інші варіанти дизайну включають відключення, нормалізацію пакетів, оптимізацію та функції втрат.

ImageNet - набір даних, що складається з більш ніж 15 мільйонів розмічених високоякісних зображень, розділених на 22000 категорій. Зображення були взяті з інтернету та розмічені вручну людьми-розмітниками за допомогою краудсорсингового майданчика Mechanical Turk від Amazon.

У 2010 році, як частина Pascal Visual Object Challenge, розпочалося щорічне змагання - ImageNet Large-Scale Visual Recognition Challenge. У змаганні використовується підвибірка з ImageNet розміром 1000 зображень у кожній із 1000 категорій. Таким чином, тренувальний сет складався приблизно з 1.2 мільйонів зображень, перевірочний — 50000 зображень, тестовий — 150000 зображень. Так як ImageNet складається з зображень різного розміру, їх необхідно було привести до єдиного розміру 256x256. Якщо зображення є прямокутником, то воно масштабується і з нього вирізається центральна частина розміром 256x256.

2.4.1 Прості мережі

AlexNet [45] є першою архітектурою, яка викликала дослідницький інтерес до глибинного навчання, коли вона виграла виклик ImageNet зі значним відривом. Архітектура складається з восьми рівнів, у тому числі п’яти згорткових, функції активації та трьох пов’язаних рівнів. AlexNet вперше використав мультиграфічні процесори для навчання більших моделей і скорочення часу навчання. Приклад архітектури простих мереж показаний на рисунку 2.2.

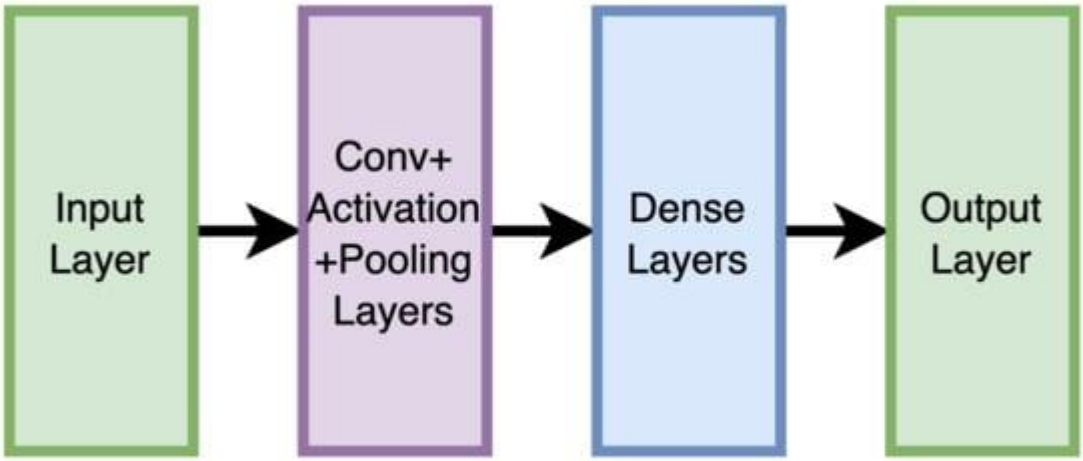

Рисунок 2.2 - Приклад архітектури простих мереж

Згодом було запропоновано групи візуальної геометрії (VGG) [46] у різних варіантах, таких як VGG16 і VGG19, які є найпоширенішими архітектурами з 16 і 19 шарами відповідно. Типовий шаблон серед цих архітектур використовує лише фільтри 3 × 3. На початкових рівнях використовується кілька фільтрів, але їхня кількість збільшується зі збільшенням глибини мережі, що є зразком, який також можна побачити в інших архітектурах. На попередніх рівнях VGG або інших простих архітектурах вони вивчають більше просторової інформації для фільтрів, тоді

як пізніші рівні використовують більше фільтрів, щоб збалансувати наявність меншої кількості просторової інформації. Спочатку архітектуру VGG було важко навчити за допомогою випадкової ініціалізації вагових коефіцієнтів. Однак навчання стало більш доступним із впровадженням інтелектуальних методів ініціалізації. VGG19 [46].

Модель досягає точності 92.7% - топ-5, при тестуванні на ImageNet у задачі розпізнавання об'єктів на зображенні. Мережа VGG має дві серйозні недоліки:

– дуже повільна швидкість навчання;

– сама архітектура мережі важить занадто багато (з'являються проблеми з диском та пропускною спроможністю)

Через глибину та кількість пов'язаних вузлів, VGG16 важить більше 533 МБ. Це робить процес розгортання VGG тривалим завданням. Хоча VGG16 і використовується для вирішення багатьох проблем класифікації за допомогою нейронних мереж, менші архітектури кращі (SqueezeNet, GoogLeNet та інші). Незважаючи на недоліки, дана архітектура є чудовим будівельним блоком для навчання, тому що її легко реалізувати.

2.4.2 мережі GoogleNet Inception

На відміну від простих мереж, наступні архітектури мають спільну властивість (тобто використання коротких шляхів від попередніх рівнів до останніх), яка вирішує проблему зникнення градієнта [47] при навчанні глибоких нейронних мереж.

Продуктивність GoogleNet була трохи кращою, ніж VGG. Однак модель GoogleNet була значно меншою за розміром, лише 28,12 МБ порівняно з 574 МБ для моделі VGG. Основний блок GoogleNet називається

«модулем Inception», який доступний у різних варіантах, що робить його більш точним, ніж початкова реалізація GoogleNet Inception. Ідея Inception полягає в тому, щоб одночасно використовувати фільтри різних розмірів, а потім під час оптимізації мережа вирішує, які ваги є важливими. Таким чином мережа ефективно вивчає багатомасштабні функції. Згортка 1 × 1 була використана для зменшення розміру карти функцій за обсягом перед застосуванням будь-якого іншого фільтра, таким чином значно зменшивши розмір моделі. Модуль Inception, як показано на рисунку 2.3, використовує згортку 1 × 1 у першому шарі та максимальне об'єднання, за яким слідує будь-який інший фільтр. Вихід усіх фільтрів об'єднується за обсягом перед тим, як перейти на наступний рівень мережі.

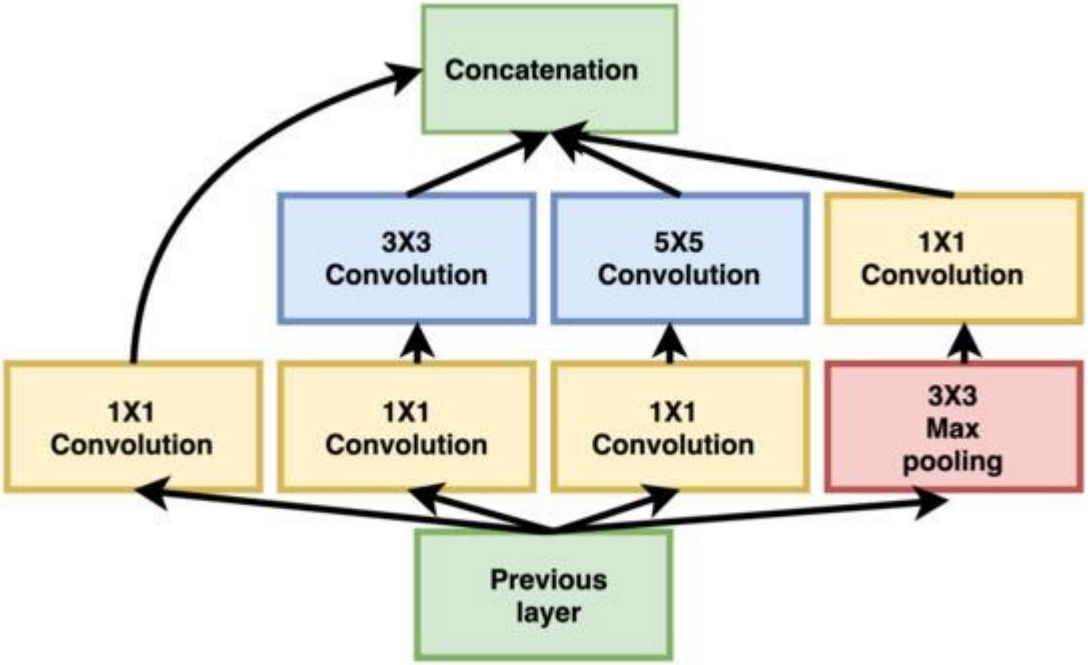

Рисунок 2.3 - Базовий блок GoogleNet

2.4.3 Залишкові мережі ResNet

Традиційна мережа страждає від проблеми зникнення градієнта [47] під час зворотного поширення. Градієнт стає значно меншим і не може оновити початкові параметри шарів, що призводить до подовження навчання мережі. Залишкова мережа (ResNet) дозволила навчити більш глибокі мережі [39]. Базовий модуль ResNet називається залишковим блоком, як показано на рисунку 1.3., який починається на вході модуля з двома гілками. Одна з гілок приймає вхідні дані через серію згорток, активацій і пакетної нормалізації, тоді як інша гілка є ярликом, який пропускає всі операції та додається до виводу іншої гілки, відомого як відображення ідентичності. Залишковий рівень починає навчання з функції ідентифікації та вивчає більш складні та надійні функції щодо глибини архітектури. В останній версії ResNet порядок операцій у першій гілці змінено зі згортання, активації та пакетної нормалізації (Conv-ReLU-BN) на пакетну нормалізацію, активацію та згортання (BN-ReLU-Conv). Цей метод називається попередньою активацією.

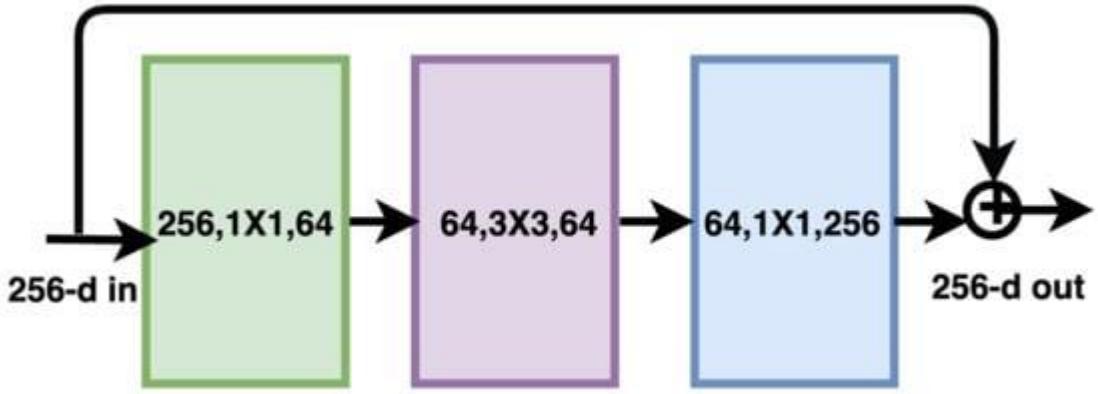

Рисунок 2.4 - Основний блок залишкової мережі

ResNet є особливим типом згорткової нейронної мережі, представленої в статті «Глибоке залишкове навчання для розпізнавання зображень». CNN зазвичай використовуються для роботи програм комп’ютерного зору. ResNet-50 — це 50-шарова згорточна нейронна мережа (48 згорткових шарів, один шар MaxPool і один шар середнього пулу). Залишкові нейронні мережі — це

тип штучної нейронної мережі, яка формує мережі шляхом укладання залишкових блоків.

Оригінальною архітектурою ResNet була ResNet-34, яка складалася з 34 зважених рівнів. Це забезпечило новий спосіб додати більше згорткових шарів до CNN, не стикаючись із проблемою зникнення градієнта, використовуючи концепцію скорочених з'єднань. Підключення швидкого доступу «пропускає» деякі рівні, перетворюючи звичайну мережу на залишкову.

Звичайна мережа була заснована на нейронних мережах VGG (VGG-16 і VGG-19) — кожна згорточна мережа мала фільтр 3×3. Однак ResNet має менше фільтрів і менш складна, ніж VGGNet. Структура мережі з різною кількістю шарів показана на рисунку 2.5 [48].

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64]×2	[3×3, 64; 3×3, 64]×3	[1×1, 64; 3×3, 64; 1×1, 256]×3	[1×1, 64; 3×3, 64; 1×1, 256]×3	[1×1, 64; 3×3, 64; 1×1, 256]×3
conv3_x	28×28	[3×3, 128; 3×3, 128]×2	[3×3, 128; 3×3, 128]×4	[1×1, 128; 3×3, 128; 1×1, 512]×4	[1×1, 128; 3×3, 128; 1×1, 512]×4	[1×1, 128; 3×3, 128; 1×1, 512]×8
conv4_x	14×14	[3×3, 256; 3×3, 256]×2	[3×3, 256; 3×3, 256]×6	[1×1, 256; 3×3, 256; 1×1, 1024]×6	[1×1, 256; 3×3, 256; 1×1, 1024]×23	[1×1, 256; 3×3, 256; 1×1, 1024]×36
conv5_x	7×7	[3×3, 512; 3×3, 512]×2	[3×3, 512; 3×3, 512]×3	[1×1, 512; 3×3, 512; 1×1, 2048]×3	[1×1, 512; 3×3, 512; 1×1, 2048]×3	[1×1, 512; 3×3, 512; 1×1, 2048]×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Рисунок 2.5 - Структура ResNet

Архітектура ResNet дотримується двох основних правил проектування. По-перше, кількість фільтрів у кожному шарі є однаковою залежно від розміру вихідної карти функцій. По-друге, якщо розмір карти функцій зменшується вдвічі, вона матиме подвоєну кількість фільтрів, щоб підтримувати часову складність кожного шару.

2.4.4 Щільні мережі DenseNet

У DenseNet [49] кожен шар об’єднує карти функцій з усіх попередніх шарів, використовуючи колективні знання в обчисленні поточної карти функцій. Поточний рівень передає свою карту функцій усім наступним шарам, забезпечуючи максимальний потік інформації та градієнти між шарами мережі. Навпаки, ResNet додає функції вхідного модуля до вихідного рівня. На рисунку 2.6 показаний модуль DenseNet. Композиційний шар використовує попередню активацію на всіх попередніх шарах перед об’єднанням із поточним шаром. DenseNet має менше параметрів і може вивчати більш складні функції.

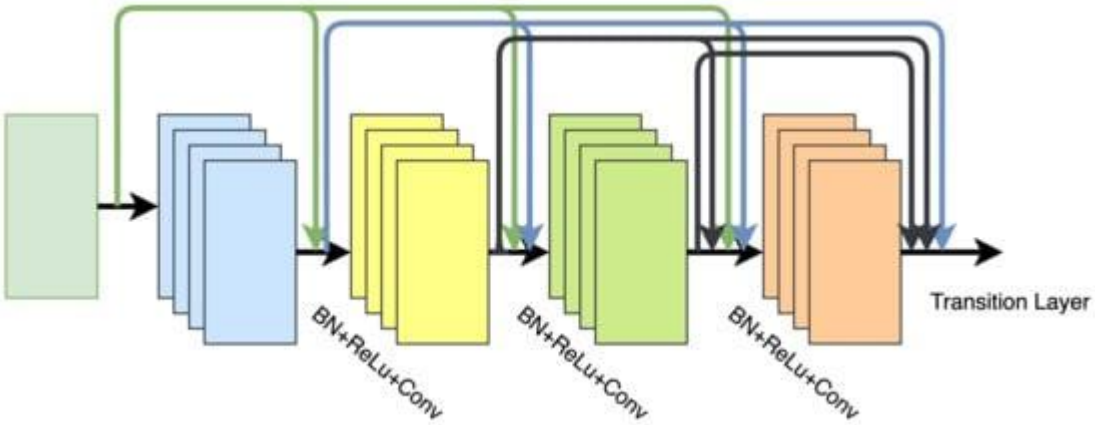

Рисунок 2.6 - Основний модуль DenseNet, що показує конкатенацію функцій з попередніх рівнів

2.4.5 Ефективні мережі EfficientNets

Як правило, глибші мережі ConvNet мають тенденцію до досягнення кращої точності у таких складних завданнях, як виявлення та класифікація

ImageNet. Однак навчені моделі занадто параметризовані, їх важко навчити та розгорнути на доступних апаратних ресурсах. Завжди існував компроміс між точністю та ефективністю вибору моделі для конкретного застосування. Традиційно моделі досягали кращої точності шляхом збільшення глибини архітектури (використовуючи більше шарів), ширини архітектури (через більше каналів) або збільшення роздільної здатності вхідного зображення. Було запропоновано [50] комплексне масштабування для збільшення всіх трьох критичних параметрів (ширина, глибина та роздільна здатність вхідного зображення) для покращення продуктивності моделі. Запропонована мережа називається EfficientNets, яка є сімейством високомасштабованих та ефективних архітектур нейронних мереж, які використовують складне масштабування для вибору моделей, таких як EfficientNet-B0–B7, з урахуванням вимог до ресурсів.

2.4.6 Squeeze Networks

Більшість архітектур CNN вимагають ресурсів для досягнення високої точності для певного набору даних. Менші архітектури з еквівалентною точністю вимагають меншої пропускної здатності, легко розгортаються на обладнанні з обмеженою ємністю та пропонують переваги в розподіленому навчанні. Вага моделі для більшості архітектур CNN коливається від 100 МБ (наприклад, ResNet) до 553 МБ (наприклад, VGG). Вага для AlexNet знаходиться посередині – 249 МБ. Нещодавно в SqueezNet [51] автор запропонував архітектуру з точністю, порівнянною з AlexNet, але з меншою кількістю параметрів і з неймовірно меншою вагою 4,9 МБ. Обсяг було додатково зменшено до 0,5 МБ шляхом застосування методів стиснення, таких як скорочення ваги та розрідження моделі.

Базовий модуль Fire Module використовується в архітектурі SqueezNet з розумною комбінацією фільтрів 1 × 1 і 3 × 3. Модуль складається з двофазної операції «стиснення» та «розгортання». Під час фази розширення використовується комбінація фільтрів 1 × 1 і 3 × 3 для захоплення просторового відношення та вилучення більш складних функцій, як показано на рисунку 2.7.

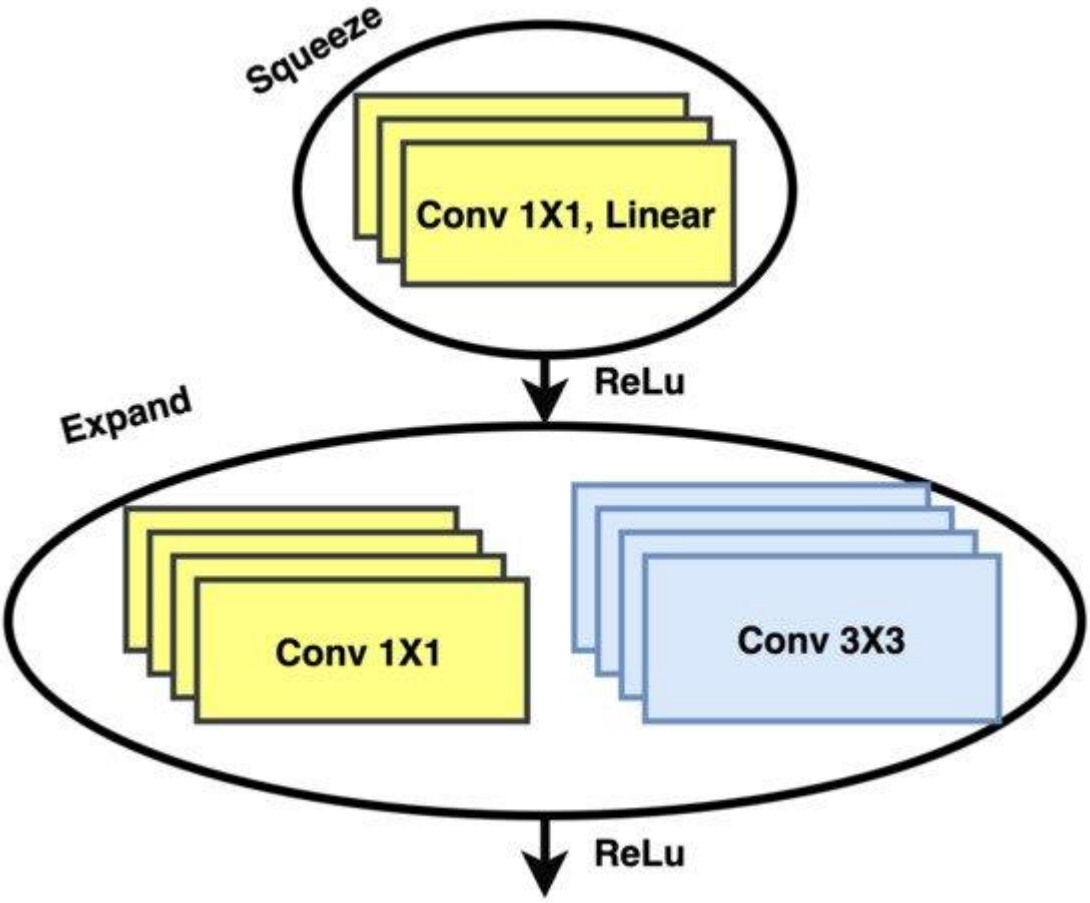

Рисунок 2.7 - Модуль Fire: магістральний модуль мережі стискання

Фаза стиснення застосовує меншу кількість фільтрів 1 × 1, ніж вхідний об’єм, таким чином зменшуючи розміри вихідної карти функцій. Перед подачею до фази розширення активація ReLU застосовується до виходу фази стиснення.

2.4.7 Перетасовуючі мережі ShuffleNet

ShuffleNet [52] — ще одна легка архітектурна конструкція, що належить до сімейства архітектур, таких як MobileNet [53], CondenseNet [54], MobileNetV2 [55] і Xception [56]. Ці архітектури використовують групову та глибинну згортку та підходять для недорогих пристроїв. Автори ShuffleNet запропонували рекомендації щодо ефективного проектування архітектури мережі. Традиційно широко прийнята непряма метрика під назвою FLOPS використовується як єдина міра обчислювальної складності (оцінка фактичного часу виконання). Однак пряма метрика, така як швидкість або затримка, є більш актуальною, якщо розглядати групові та глибинні згортки на пристроях нижчого класу. Вартість доступу до пам’яті цих операцій слід враховувати при розробці нейронної архітектури для пристроїв низького класу. Другим важливим фактором, який слід враховувати, є ступінь паралелізму. Наприклад, за тих самих FLOP модель з високим ступенем паралелізму може працювати краще, ніж аналог з низьким ступенем.

2.4.8 Мобільні мережі

MobileNetV2 [55] розроблено для використання в програмах комп’ютерного зору, розроблених для пристроїв з обмеженими ресурсами. Модель використовує менше операцій і доступу до пам’яті, щоб забезпечити точність, порівнянну з AlexNet. На відміну від стандартної згортки, модель MobileNetV1 пропонує використання розділених по глибині згорток, що означає, що за згорткою по глибині слідує згортка 1 × 1. Один фільтр застосовується для кожного вхідного каналу в глибинній згортці, а поточкова операція використовується для об’єднання вихідних даних глибинної згортки. Моделі є легкими завдяки зменшеному множенню, що зменшує обчислювальну складність. У MobileNetV2 автори представили інвертований

залишок із лінійним шаром вузького місця. У залишковому блоці з одним кроком перший шар приймає згортку 1 × 1, за якою слідує згортка по глибині в другому шарі. Третій шар використовує згортку 1 × 1 без функції активації, як показано на рисунку 2.8.

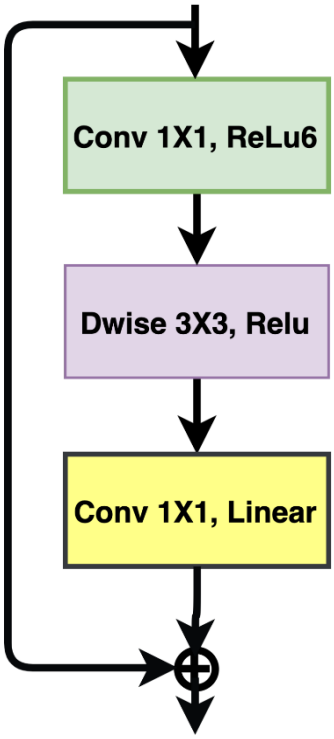

Рисунок 2.8 - Мобільна мережа з блоком MBConvolution, що складається з трьох згорткових шарів і лінійного шару

2.4.9 Пошукова нейронна архітектура NASNet

Пошукова нейронна архітектура NASNet належить до сімейства методів глибокого навчання, які називаються метанавчанням. Ці алгоритми використовують допоміжні методи пошуку, такі як випадкові, еволюційні та рекурентні нейронні мережі та глибоке підкріплення навчання для розробки характеристик різних мережевих архітектур. Ці характеристики включають швидкість навчання, кількість фільтрів і карти фільтрів. При пошуку за нейронною архітектурою ці характеристики вивчаються іншою нейронною

мережею для пошуку в дискретному просторі. У пошуках нейронної архітектури використовуються два типи згорткових шарів, які називаються нормальною коміркою та редукційною коміркою. Типова клітинка повертає карту ознак тих самих розмірів, що й у вхідних даних, тоді як клітинка зменшення зменшує розмірність на два.

Основна ідея полягає в тому, щоб спроектувати одну комірку замість цілої мережі. Алгоритм пошуку шукатиме оптимальні параметри з набору параметрів, а потім створюватиме повну архітектуру шляхом укладання звичайних і редукційних комірок. Як правило, дослідження архітектури виконується на менших наборах даних, а вивчені шари передаються в архітектуру пошуку для великих наборів даних. Однак ці підходи не дозволяють різноманітність рівнів, необхідну для високої точності та меншої затримки в мобільних додатках. У MNASNets [57] пропонується багатоцільовий пошук з використанням глибокого підкріплення для пошуку моделей CNN з підвищеною точністю та низькою затримкою висновку, які підходять для мобільних пристроїв.

2.4.10 Wide-ResNets

Коли ResNet масштабується до тисячі шарів, часткове підвищення точності вимагає подвоєння шарів і експоненціального збільшення часу навчання. У Wide-ResNet [58] автори запропонували архітектуру зі зменшеною глибиною та збільшеною шириною порівняно з ResNets, щоб отримати хорошу точність порівняно з тонким і глибоким аналогом.

У типовій архітектурі ResNet є два блоки: основний (залишковий) і вузьке місце. Блок «вузького місця» використовувався, щоб зробити мережу тоншою, менш дорогою з точки зору обчислень і придатною для

проектування більш глибоких мереж. Тим не менш, автори використовували лише залишковий блок із збільшенням згорткових шарів, карт функцій і розмірів фільтрів для кращої продуктивності. Автори врахували два фактори: (i) фактор поглиблення l і (ii) фактор розширення k, де l представляє кількість згортань у блоці, а k — кількість карт ознак у згорткових шарах. Кількість параметрів зростає лінійно з l і квадратично з k. Пам'ятаючи, що графічні процесори більш ефективні в паралельних обчисленнях, а отже, розширення архітектури ефективніше. Продуктивність можна додатково підвищити, якщо використовувати відключення в залишковому блоці. На рисунку 2.9 показаний блок Wide-ResNet [58] з випаданням і без нього.

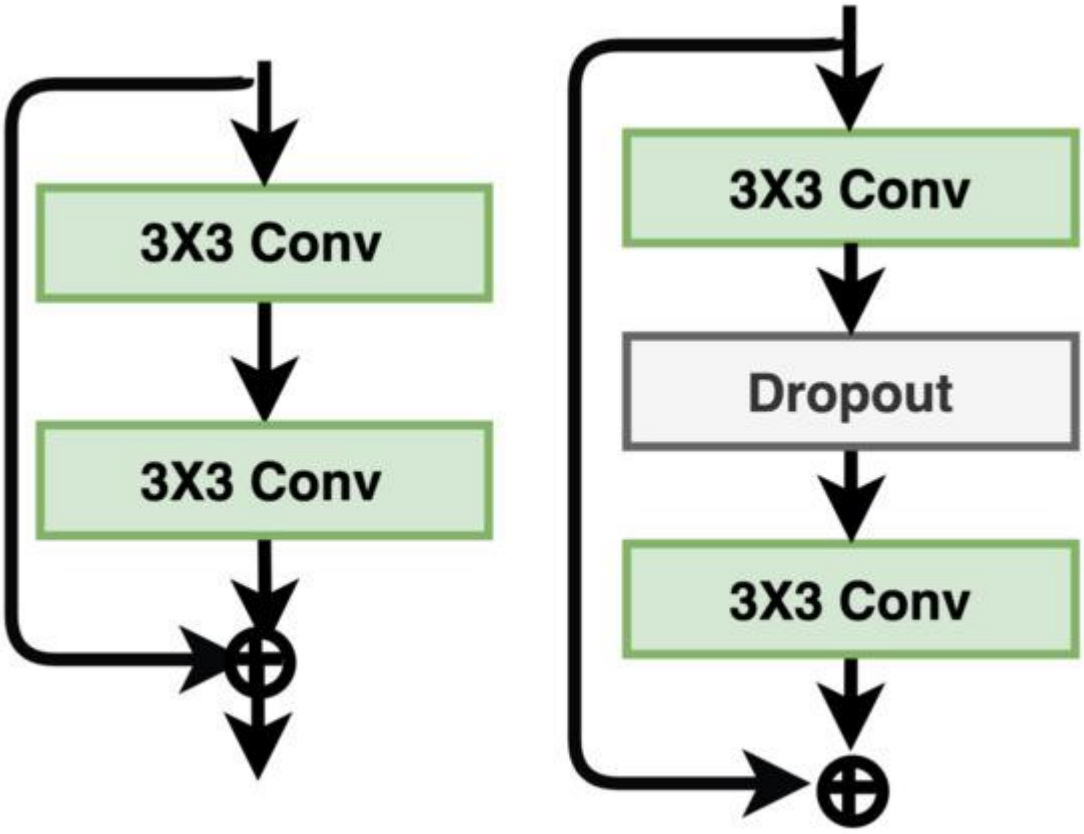

Рисунок 2.9 - Основна схема блоку Wide-ResNet із шаром вилучення та без нього

2.4.11 ResNeXt

ResNeXt [59] бере приклад з пропускних з'єднань ResNet, стекування шарів VGG і стратегії розділення-перетворення-злиття. Модуль ResNeXt дуже схожий на стратегію розділеного перетворення, за винятком того, що вихідні шляхи об'єднуються шляхом додавання замість конкатенації по глибині, як показано на рисунку 2.10. Ще одна ключова відмінність від Inception полягає в тому, що всі розділені шляхи мають однакову топологію. ResNeXt запропонував гіперпараметр під назвою потужність, який відноситься до незалежних шляхів для налаштування ємності моделі.

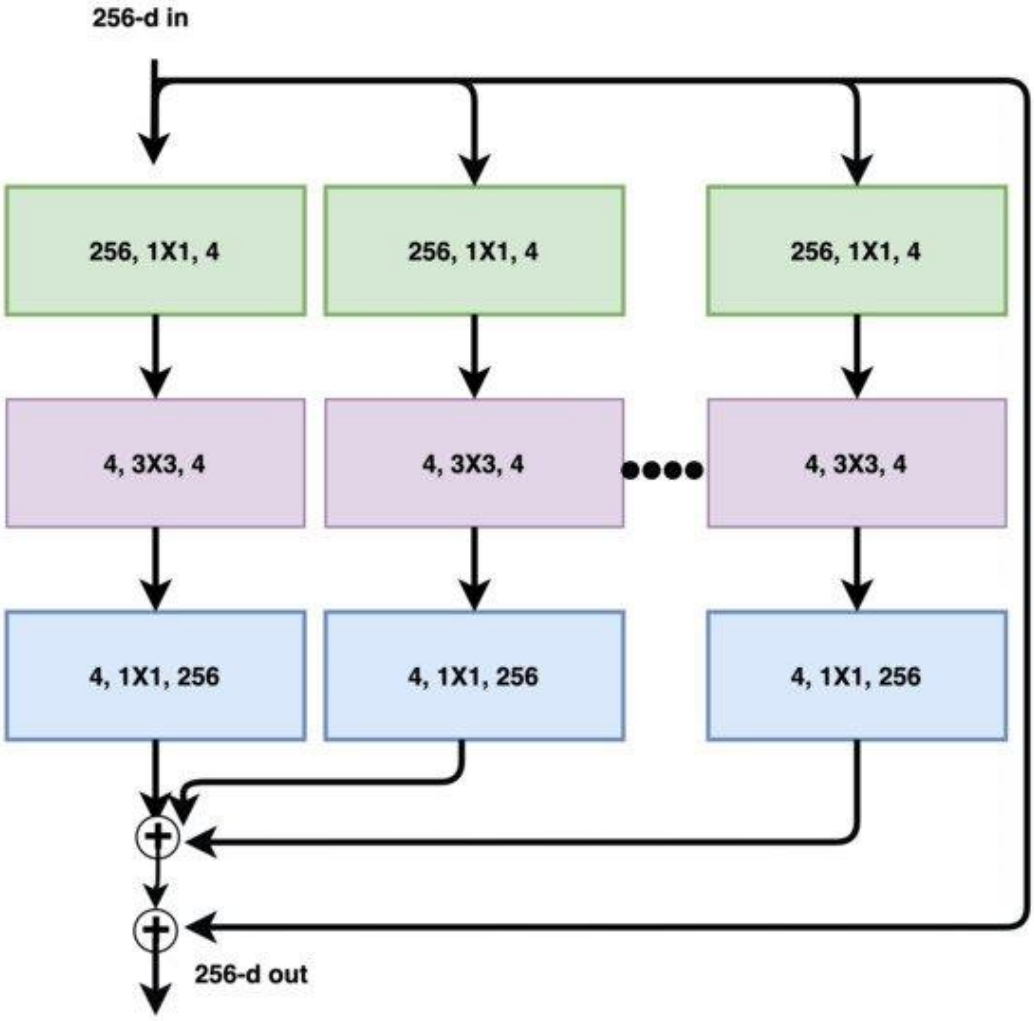

Рисунок 2.10 - Основний модуль ResNeXt, що показує різні шляхи вивчення функцій

Модель досягла кращих результатів із збільшенням потужності, а не заглибленням і ширше. Модель також легше адаптувати до нових наборів даних, оскільки існує лише один параметр для коригування.

2.5 Порівняння згорткових нейронних мереж

Шість популярних попередньо навчених моделей CNN із глибоким навчанням використовувалися для виявлення COVID-19 за допомогою зображень ЕКГ. Це ResNet18, ResNet50, ResNet101, DenseNet201, InceptionV3 і MobileNetV2, які спочатку навчаються на базі даних ImageNet. Залишкова мережа була створена для подолання проблеми зникнення градієнта та деградації. ResNet має різні варіанти залежно від кількості шарів у залишковій мережі: ResNet18, ResNet50, ResNet101 і ResNet152. ResNet широко використовується для передавального навчання в класифікації біомедичних зображень. Під час навчання шари глибокої нейронної мережі зазвичай вивчають функції низького або високого рівня, тоді як ResNet вивчає залишки, а не особливості.

Щільна згорточна мережа (DenseNet) вимагає менше параметрів, ніж традиційна CNN, оскільки не потребує навчання надлишковим картам функцій. DenseNet має дуже вузькі шари, тому додає лише невелику кількість нових карт функцій. DenseNet має чотири різні відомі варіанти: DenseNet264, DenseNet169, DenseNet121 і DenseNet20. DenseNet забезпечує прямий доступ до оригінального вхідного зображення, а також до градієнтів від функції втрати в кожному шарі. У результаті обчислювальна вартість DenseNet була значно знижена, що зробило його кращим вибором для класифікації зображень.

Крім того, MobileNetv2 не можна порівняти з іншими мережами по глибині, скоріше це компактна мережа. За винятком першого шару, який є

повною згорткою, решта шарів не згортаються. За винятком останнього повністю підключеного рівня, який не має нелінійності та подається на рівень Softmax для класифікації, структура MobileNet побудована на згортинах, які можна розділити по глибині. Архітектура мережі наведена на рисунку 2.11.

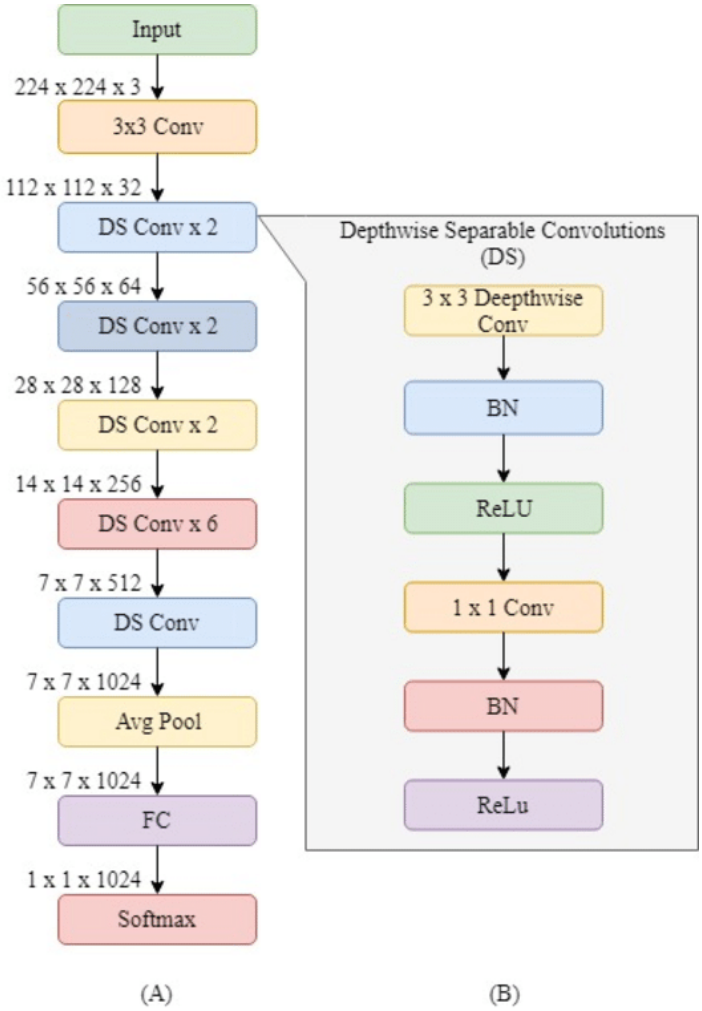

Рисунок 2.11 - Ілюстрація архітектури MobileNet. (A) Загальна архітектура MobileNet і (B) поглиблене пояснення рівня DS.

Пакетна нормалізація та нелінійність Rectified Linear Units (ReLU) застосовуються до всіх шарів. Перед повністю з’єднаним шаром кінцеве середнє об’єднання зменшує просторову роздільну здатність до 1. MobileNet має 28 рівнів, коли глибинні та поточкові згортки підраховуються окремо.

2.5 Передавальне навчання на основі глибоких згорткових нейронних мереж

Навчання нейронної мережі може займати довгий час – години або навіть дні. Моделі глибокого навчання потребують більшого набору даних для навчання моделі. Для менших наборів даних моделі глибокого навчання страждають від проблеми переобладнання. Усі ці проблеми можна вирішити за допомогою передавального навчання. У передавальному навчанні використовуються попередньо підготовлені моделі. Ці попередньо підготовлені моделі були навчені на різних наборах даних із великим об’ємом зображень. Передавальне навчання вирішує проблему навчання даних шляхом передачі наявних знань у цільовій галузі, де доступних вибіркових даних дуже мало або зовсім немає. Використання передавального навчання також полегшує навчання даних з меншими витратами на створення моделі. Також можна виконати донавчання зміненої мережі на основі своїх даних.

Цей підхід працює оскільки попередньо навчені нейронні мережі вже навчились розпізнавати велику кількість ознак різних об’єктів. Все, що потрібно зробити, – це довчити таку мережу на основі ознак, які вона вже знає, робити іншу класифікацію. Для цього потрібно набагато менше даних і часу, ніж для повного навчання мережі з самого початку.

Передавальне навчання зосереджується на зберіганні знань, отриманих під час вирішення однієї проблеми, і застосуванні їх до іншої, але пов’язаної проблеми. Наприклад, знання, отримані під час навчання розпізнавання автомобілів, можуть бути використані при спробі розпізнати вантажівки. Ця область досліджень має певний зв’язок із довгою історією психологічної літератури про передачу навчання, хоча практичні зв’язки між двома галузями обмежені. З практичної точки зору, повторне використання або

передача інформації з раніше вивчених завдань для вивчення нових завдань має потенціал для значного підвищення ефективності вибірки навчання з підкріпленням. Блок схема передавального навчання наведена на рисунку 2.12.

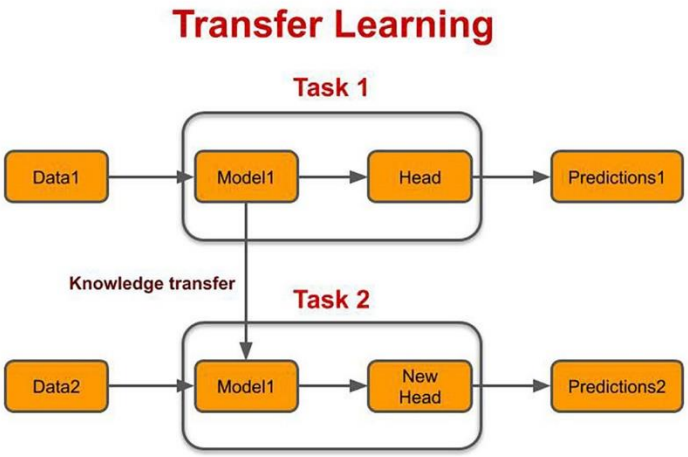

Рисунок 2.12 - Блок схема передавального навчання.

Для реалізації передавального навчання береться згорткова частина мережі і до неї додається інша частина, яка відповідає за класифікацію, що підходить для вирішення поставленої задачі. Потім така нейронна мережа донавчається на потрібному наборі даних. Під час навчання частина мережі є «замороженою» – вона не навчається. Навчається лише нова частина мережі, що використовується для класифікації. Перед навчанням вагові коефіцієнти в цій частині ініціалізуються випадковим чином, а далі навчання відбувається методом зворотнього поширення помилки.

2.6 Висновки до розділу 2

Автоматичний аналіз рентгенівських знімків грудної клітки можна виконати за допомогою методів на основі глибокого навчання, що може

прискорити час аналізу. Методи машинного навчання можуть тренувати ваги мереж на великих наборах даних, а також точно налаштовувати ваги попередньо навчених мереж на малих наборах даних.

Отже, задачею роботи є полегшити роботу медичних працівників при діагностиці захворювання COVID-19 за рентгенівськими зображеннями, використовуючи методи машинного навчання. Для вирішення задачі було запропоновано використання методу передавального навчання. Через його низьку ресурсозатратність та гнучкість самого методу. Оберемо кілька популярних моделей для порівняння результатів, а саме Resnet50, VGG16 та Inception3.

Розв'язувана даною нейронною мережею задача – класифікація зображень, а саме: класифікація рентгенівських зображень COVID-19, вірусної пневмонії, помутніння легень та зображення здорових легень. Вхідними даними буде виступати набір рентгенівських зображень. На виході мають бути класи, що визначатимуть виявлену патологію на представленому знімку, або норму.

3 РОЗРОБКА СИСТЕМИ ДІАГНОСТУВАННЯ COVID-19 ТА АНАЛІЗУ РЕНТГЕНІВСЬКИХ ЗОБРАЖЕНЬ ЗА ДОПОМОГОЮ ГЛИБИННИХ НЕЙРОННИХ МЕРЕЖ

3.1 Програмні засоби

Для вирішення задачі обрано мову програмування Python. Вона широко використовується для машинного навчання, має простий синтаксис і динамічну типізацію, дозволяє легко модифікувати програму для проведення різноманітних експериментів, при цьому, програма при таких модифікаціях залишається такою, що легко читається і добре підтримується. Python має безліч фреймворків і бібліотек, які спрощують процес написання коду і скорочують час на розробку. Основними фреймворками і бібліотеками, які використовуються у машинному навчанні, а також будуть використані у цій роботі є: NumPy, PyTorch, Scikit-learn, Matplotlib, Seaborn, Python Imaging Library.

NumPy – основна бібліотека Python, яка спрощує роботу з векторами і матрицями. Містить готові методи для різних операцій: від створення, зміни форми, множення і розрахунку детермінанта матриць до вирішення лінійних рівнянь і сингулярного розкладання.

Matplotlib – бібліотека для створення двовимірних діаграм і графіків. Важливою задачею машинного навчання є візуалізація даних. За допомогою бібліотеки Matplotlib можна побудувати будь-який графік.

Seaborn — це бібліотека візуалізації даних Python на основі matplotlib. Він забезпечує високорівневий інтерфейс для малювання привабливої та інформативної статистичної графіки.

Python Imaging Library додає можливості обробки зображень. Ця бібліотека забезпечує розширену підтримку форматів файлів, ефективне внутрішнє представлення та досить потужні можливості обробки зображень.

Основна бібліотека зображень розроблена для швидкого доступу до даних, що зберігаються в кількох основних форматах.

PyTorch — це оптимізована бібліотека тензорів Deep Learning на основі Python і Torch, яка в основному використовується для додатків, які використовують графічні та центральні процесори. PyTorch користується перевагою перед іншими фреймворками Deep Learning, такими як TensorFlow і Keras, оскільки він використовує динамічні обчислювальні графіки та є повністю Pythonic. Це дозволяє вченим, розробникам і налагоджувачам нейронних мереж запускати та тестувати частини коду в режимі реального часу. Таким чином, користувачам не потрібно чекати, поки буде реалізовано весь код, щоб перевірити, чи працює частина коду чи ні. Дві основні функції PyTorch - Tensor Computation із сильною підтримкою прискорення GPU та автоматична диференціація для створення та навчання глибоких нейронних мереж.

Для написання коду було використано інтерактивну оболонку для мови Python – Jupyter Notebook. Jupyter Notebook — додаток із відкритим вихідним кодом, у якому можна одразу побачити результат виконання коду. Основна відмінність від традиційних інструментів розробки — можливість розбити код на частини та виконати їх окремо. Наприклад, ви можете написати одну функцію і відразу перевірити, як вона працює, чи не запускаються інші фрагменти коду. Також можна змінити порядок виконання коду. Завдяки цьому ноутбук Jupyter дуже популярний в аналітиці даних і Data Science.

Також для експериментального моделювання було використано безкоштовну платформу для машинного навчання і нейронних мереж Google Colaboratory. На платформі можна використовувати ноутбуки, що дуже схожі на ноутбуки Jupyter. Головний плюс цієї платформи полягає в тому, що навчання нейронних мереж потребує великих обчислювальних ресурсів, а на платформі Google Colaboratory користувачам безкоштовно надається доволі потужний GPU.

3.2 Набір даних для навчання

Для навчання нейронної мережі було використано набір даних Chest X-ray [60]. Команда дослідників з університету Катару і університету Дакки разом зі своїми співробітниками з Пакистану та Малайзії у співпраці з лікарями створили базу даних рентгенівських зображень грудної клітки для позитивних випадків COVID-19 разом із зображеннями нормальної та вірусної пневмонії. Всього в наборі даних 3616 зображення COVID19, 6012 зображень помутніння легень, 10192 зображень нормальних легень, 1345 зображень вірусної пневмонії.

Дані про COVID збираються з різних загальнодоступних наборів даних, онлайн-джерел і опублікованих документів.

-2473 зображення CXR зібрано з набору даних padchest[61];

-183 CXR зображення з німецької медичної школи[62];

-559 CXR зображення з SIRM, Github, Kaggle & Tweeter[63];

-400 зображень CXR з іншого джерела Github[64].

Звичайні зображення легень збираються з різних наборів даних

-8851 Радіологічноге товариство Північної Америки [65];

-1341 Kaggle [66].

Зображення легень з помутніннями:

-6012 CXR-зображення непрозорості легень зібрано з набору даних CXR Радіологічного товариства Північної Америки [65].

Зображення вірусної пневмонії:

-1345 Дані про вірусну пневмонію збираються з бази даних рентгенівських зображень грудної клітини [66].

Приклади зображення з набору даних Chest X-ray Images наведено на рисунку 3.1.

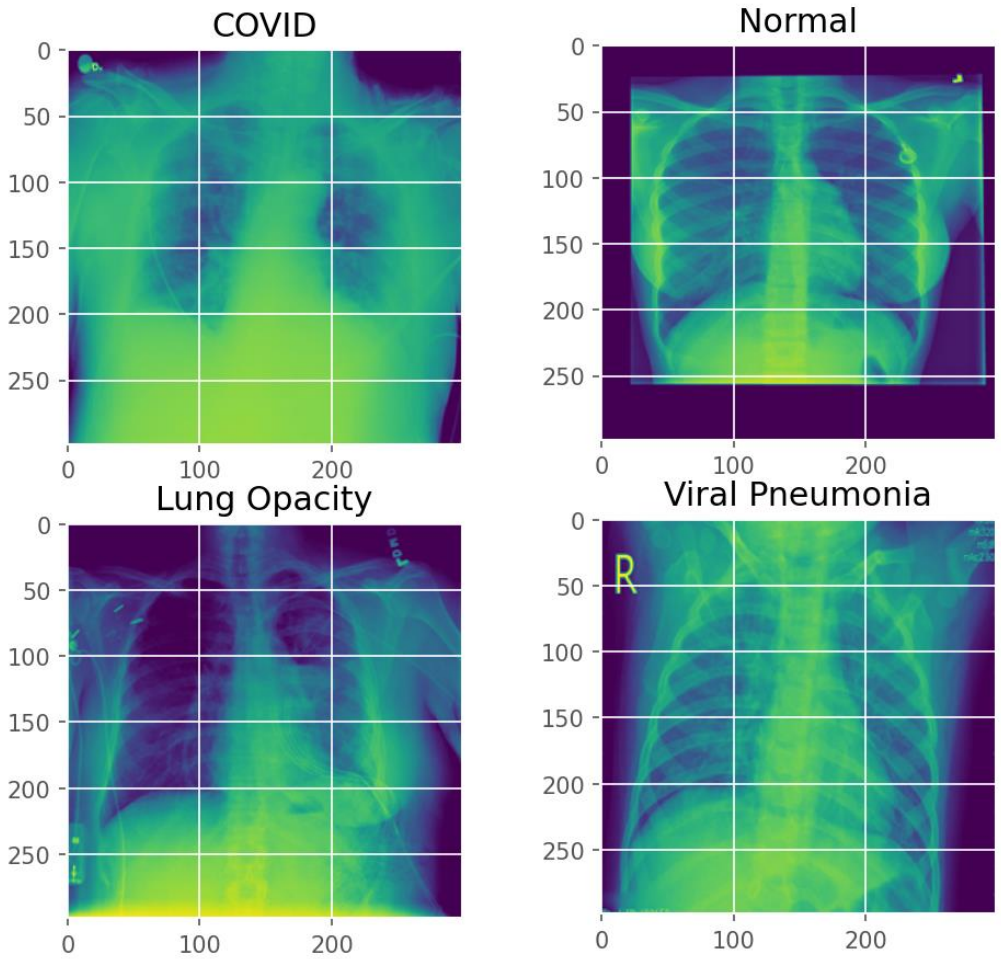

Рисунок 3.1 – Приклади зображення з набору даних Chest X-ray Images

Для передавального навчання набір даних було розділено на навчальну (training set), тестову (test set) та перевірочну (validation set) вибірки у співвідношенні 80:10:10. Таким чином до навчальної вибірки потрапило 16932 зображення, до тестової та перевірочної по 2116 зображень.

Доповнення даних може значно збільшити точність навчання нейронної мережі за умови, коли даних для навчання недостатньо, вони незбалансовані або в них нерівномірно представлені різні класи. Для доповнення даних було використано функції PyTorch для модифікації тренувальних даних:

-обертання зображення на випадковий кут у діапазоні від -15 до 15 градусів;

-випадкове перевернення зображення горизонтально з імовірністю за замовчуванням 50%.

Приклади зображень після перетворень наведені на рисунку 3.2.

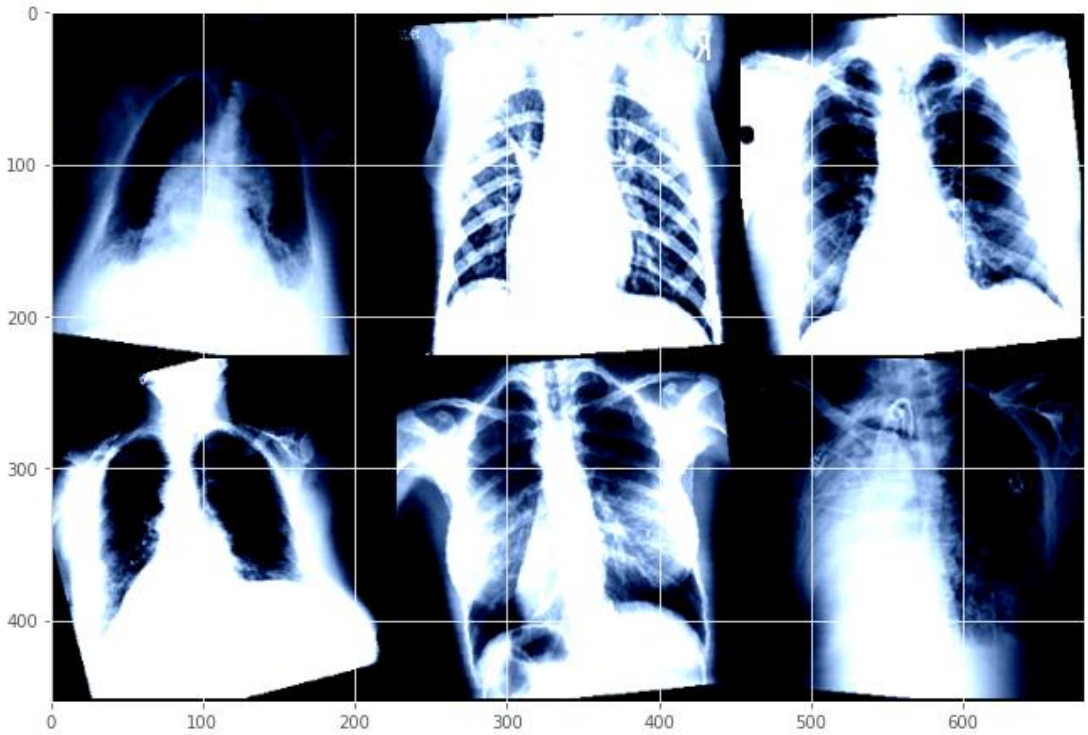

Рисунок 3.2 – Приклади згенерованих зображень.

Попередня обробка зображення є важливою частиною системи і може впливати на максимальну точність, яку досягає модель під час навчання. Перед подачею на вхід нейронної мережі дані необхідно нормалізувати. Оскільки нейронна мережа може приймати на вхід лише вектори однакової розмірності, всі зображення були приведені до єдиного розміру – 224x224 пікселів. Для Inception зображення повинні мати розмір 299x299 пікселів.

Зрештою, зображення перетворюється на тензор і нормалізується на середнє значення та стандартне відхилення всіх зображень у ImageNet.

Потім ми завантажуємо зображення за допомогою DataLoader. Трансформації зображень, застосовуються до даних під час їх завантаження за допомогою DataLoader. Порядок даних також перемішується. Пакет torchvision.transforms і DataLoader є дуже важливими функціями PyTorch, які дуже спрощують процеси розширення та завантаження даних.

3.4 Навчання моделі

Зібрати зображення, і навчити класифікатор з самого початку важко та займає багато часу. Отже, ми використовуємо попередньо навчену модель як нашу основу та змінюємо останні кілька шарів, щоб ми могли класифікувати зображення відповідно до бажаних класів. Це допомагає отримувати результати навіть із невеликим набором даних, оскільки основні функції зображення вже вивчено в попередньо навченій моделі зі значно більшого набору даних, наприклад ImageNet.

Внутрішні шари зберігаються такими ж, як і попередньо навчена модель, і лише останні шари змінюються відповідно до нашої кількості класів. У цій роботі ми використовуємо попередньо навчені моделі ResNet50, VGG16 та Inception3, самі моделі були взяті з бібліотеки PyTorch.

Модель ResNet50 — має хороший компроміс між точністю та часом висновку. VGG16 – одна з найзнаменитіших моделей, відправлених на змагання. Вона є покращеною версією AlexNet, в якій замінені великі фільтри (розміру 11 і 5 в першому і другому шарі верстата, відповідно) на кілька фільтрів розміру 3x3, наступних один за одним. Inception-v3 — це архітектура згорткової нейронної мережі з сімейства Inception, яка вносить кілька покращень, включаючи використання згладжування міток,

факторизованих згорток 7 x 7 і використання допоміжного класифікатора для поширення інформації про мітки нижче в мережі (поряд із використанням пакетного нормалізація для шарів у боковій частині).

Коли модель завантажується в PyTorch, для всіх її параметрів у полі «requires_grad» встановлено значення true за замовчуванням. Це означає, що кожна зміна значень параметрів буде збережена для використання в графіку зворотного поширення, який використовується для навчання. Це збільшує вимоги до пам'яті. Оскільки більшість параметрів у наших попередньо навчених моделях вже навчені, ми скидаємо для поля requires_grad значення false.

Далі замінюється останній шар моделей невеликим набором послідовних шарів. Вхідні дані останнього повністю підключеного рівня подаються на лінійний рівень. Оскільки ми будемо проводити навчання на GPU, ми готуємо модель для GPU.

Потім визначається функція втрат і оптимізатор, який буде використовуватися для навчання. PyTorch надає різноманітні функції втрат. Оберемо функцію перехресної втрати ентропії (CrossEntropyLoss). Перехресна втрата ентропії – це показник, який використовується для вимірювання ефективності моделі класифікації в машинному навчанні. Втрата (або помилка) вимірюється як число від 0 до 1, де 0 є ідеальною моделлю. PyTorch також підтримує кілька оптимізаторів. Ми використовуємо оптимізатор Adam. Adam є одним з найпопулярніших оптимізаторів, оскільки він може адаптувати швидкість навчання для кожного параметра окремо.

3.4.1 Тренування моделей

Навчання виконується для фіксованого набору епох, в нашому випадку це – 15 для VGG-16, Inception3 та 30 епох для ResNet50. Швидкість навчання

– 0.0001. Кожне зображення обробляється один раз в одній епосі. Завантажувач навчальних даних завантажує дані пакетами. У нашому випадку ми задали розмір пакету 64. Це означає, що кожен пакет може містити максимум 64 зображення.

Градієнти втрат щодо параметрів, які можна навчити, обчислюються за допомогою зворотної функції. Перевагою цього підходу є те, що тепер нам потрібно навчити лише приблизно десяту частину загальної кількості параметрів моделі.

Обчислення градієнта виконується за допомогою автоградації та зворотного поширення. PyTorch накопичує всі градієнти під час проходу назад. Тому важливо обнулити їх на початку тренувального циклу. Це досягається за допомогою функції zero_grad оптимізатора. Після обчислення градієнтів у зворотному проході параметри оновлюються за допомогою покрокової функції оптимізатора.

Загальні втрати та точність обчислюються для всієї партії, які потім усереднюються для всіх партій, щоб отримати значення втрат і точності для всієї епохи.

3.4.2 Перевірка моделей

Оскільки навчання виконується протягом більшої кількості епох, модель має тенденцію переналаштовувати дані, що призводить до її поганої продуктивності на нових тестових даних. Важливо підтримувати окремий набір перевірки, щоб зупинити навчання в потрібній точці та запобігти перенавчанню. Перевірка виконується в кожній епосі відразу після циклу навчання. Оскільки нам не потрібні будь-які обчислення градієнта в процесі перевірки, це робиться в блоці torch.no_grad().

Для кожного пакету перевірки вхідні дані та мітки передаються до графічного процесора (якщо доступна Cuda, інакше вони передаються до центрального процесора). Вхідні дані проходять через прямий прохід, за яким слідують обчислення втрат і точності в кінці циклу для всієї епохи. Нижче, на рисунках 3.3, 3.4, 3.5, наведені графіки точності та втрат.

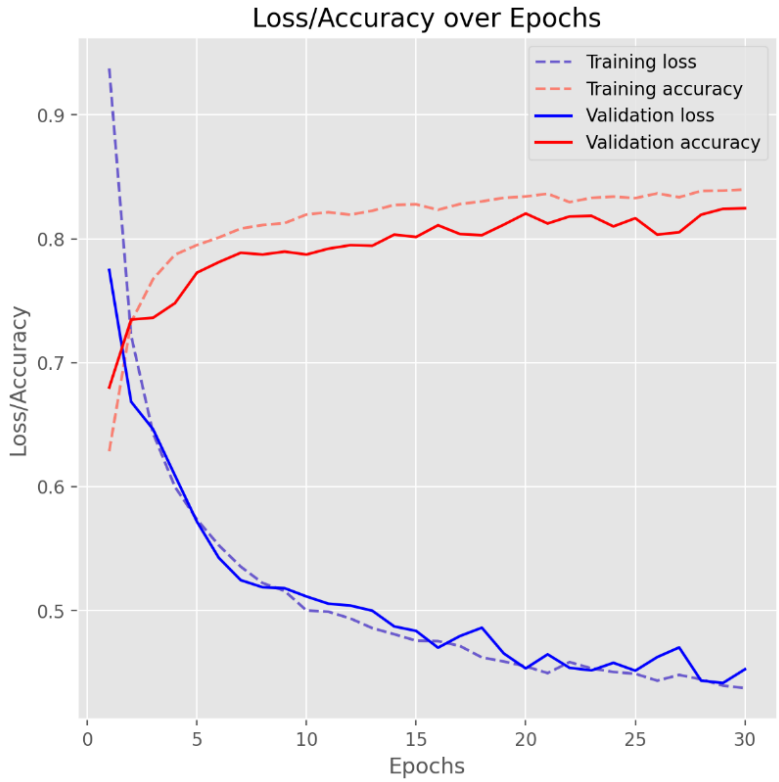

Рисунок 3.3 - Криві точності та втрат для навчання та перевірки моделі ResNet50

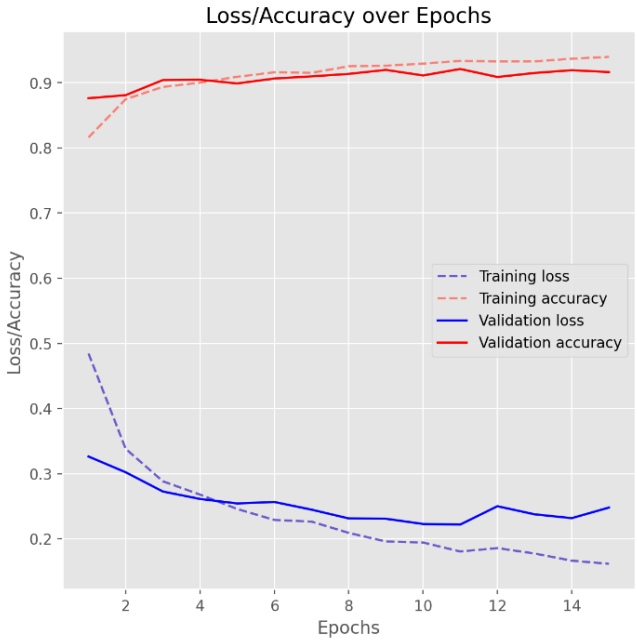

Рисунок 3.4 - Криві точності та втрат для навчання та перевірки моделі VGG16

Рисунок 3.5 - Криві точності та втрат для навчання та перевірки моделі Inception3

Втрати як під час перевірки, так і під час навчання, для цього набору даних, зменшуються досить швидко. Точність також дуже швидко зростає до діапазону 0,9. Зі збільшенням кількості епох втрати від навчання ще більше

зменшуються, що призводить до перенавчання, але результати перевірки не покращуються. Вихідний код програми наведений в додатку А. Процес тренування показано в додатку Б.

3.5 Показники оцінки ефективності

Для оцінки кожного алгоритму було взято до уваги наступні чотири показники: точність, відклик, точність класу, F1 міра.

Точність (accuracy) – це відношення правильно прогнозованих позитивних пацієнтів із COVID-19 до загальної кількості позитивних прогнозів (тобто справжні позитивні та хибні позитивні результати). Цей показник дає алгоритму можливість визначати рівень помилкових спрацьовувань:

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

Де істинно позитивний результат (TP);

Істинно негативний (TN);

Хибнопозитивний (FP);

Хибнонегативний результат (FN).

Істинно-позитивні і істинно-негативні прогнози є точними, тоді як хибно-позитивні і хибно-негативні є неправильними. Точність позначається числом від 0 до 1 і часто виражається у відсотках.

Відклик (recall), або чутливість алгоритму - це співвідношення правильно передбачених позитивних результатів (тобто справжніх позитивних) до фактичних спостережень класу (тобто справжніх позитивних і хибнонегативних):

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Точність класу (precision) є найбільш використовуваною та інтуїтивно зрозумілою мірою класифікації. Точність визначається як відношення правильних прогнозів до загальної кількості зразків:

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

F1 міра оцінює хибно-позитивні та хибно-негативні результати, взявши середнє зважене значення вищезгаданих показників. Оцінка F1 корисна у випадках, коли розподіл класів є нерівномірним.

Оцінка F1 – це в основному гармонійне середнє між чутливістю та точністю. Вона використовується для вимірювання точності тестів і є прямим показником продуктивності моделі. Його оцінюють на основі наступного рівняння:

$$\text{F1 score} = \frac{2 \times recall \times precision}{recall + precision}$$

Діапазон оцінок F1 становить від 0 до 1, мета полягає в тому, щоб отримати якомога ближче до 1. Потрібна модель, яка має як хорошу точність, так і хорошу чутливість. Оцінка F1 є показником балансу між точністю та запам’ятовуванням, досягнутим запропонованими моделями. Загалом, показник F1 є низьким, коли чутливість або точність низькі, а коли обидві змінні високі, він наближається до одиниці.

3.6 Аналіз результатів

Після завершення навчання була перевірена якість роботи мережі на тестових даних, які не були застосовані в процесі навчання. Перед тустуванням було обрано моделі, які мали найвищу точнісь під час тренування:

- ResNet50 – 81.75% – тридцята епоха навчання;

- VGG16 – 93.96% – п’ятнадцята епоха навчання;
- Inception3 – 98.75% - тринадцята епоха навчання.

Також було оцінено інші характеристики якості навчання мережі, зокрема точність (precision), відгук (recall) та F1-міру. Значення цих мір за класами наведено в таблиці 3.1.

Таблиця 3.1 – Характеристики моделей

Model name	ResNet50	VGG16	Inception3
All prediction	2117	2117	2117
True prediction	1718	1912	2039
False prediction	399	205	78
Accuracy	0.811	0.903	0.963
	Covid prediction	Covid prediction	Covid prediction
Precision	0.524	0.889	0.986
Recall	0.872	0.981	0.994
F1 Score	0.655	0.933	0.99
	Lung opacity prediction	Lung opacity prediction	Lung opacity prediction
Precision	0.797	0.768	0.935
Recall	0.781	0.927	0.949
F1 Score	0.789	0.84	0.942
	Normal prediction	Normal prediction	Normal prediction
Precision	0.909	0.974	0.973
Recall	0.802	0.865	0.956
F1 Score	0.852	0.916	0.964
	Viral pneumonia	Viral pneumonia	Viral pneumonia
Precision	0.864	0.971	0.948
Recall	0.93	0.958	0.992
F1 Score	0.896	0.964	0.97

Прогнозовані результати були порівняні з фактичними результатами та були побудовані матриці помилок. Матриці помилок наведені нижче на рисунках 3.6, 3.7, 3.8.

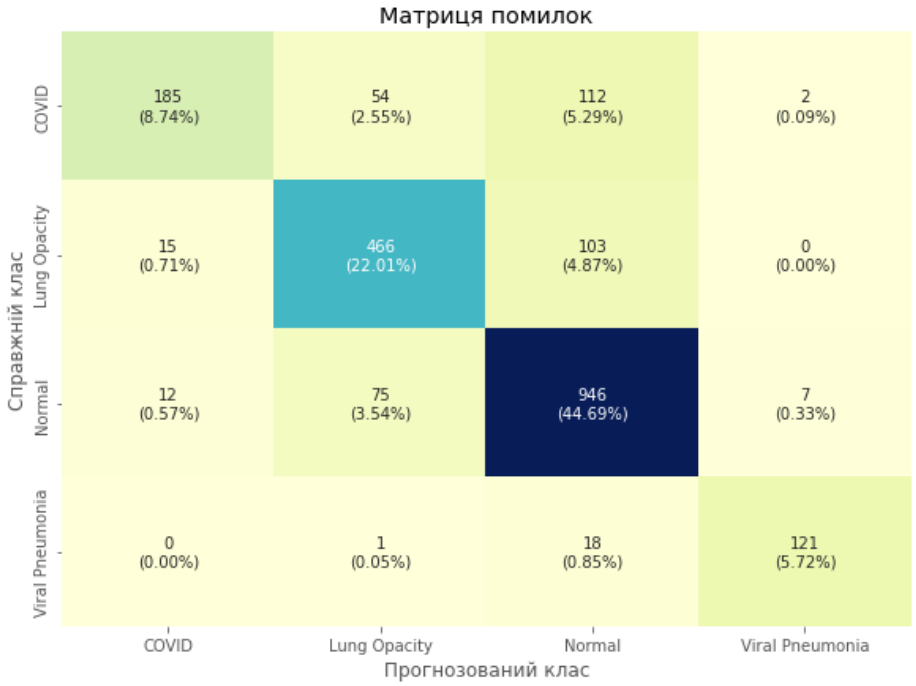

Рисунок 3.6 – Матриця помилок моделі ResNet50

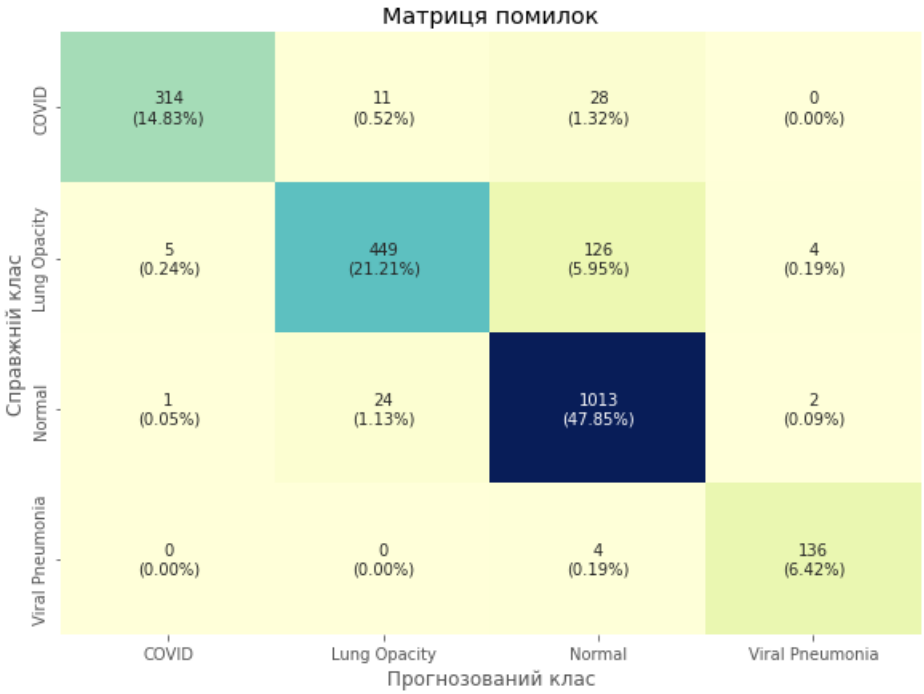

Рисунок 3.7 – Матриця помилок моделі VGG16

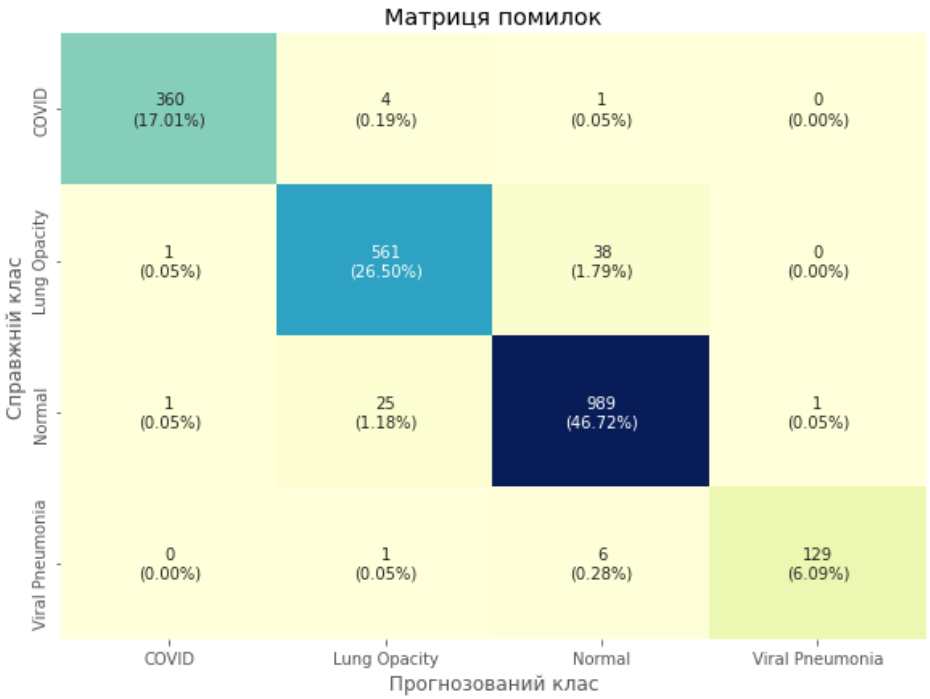

Рисунок 3.8 – Матриця помилок моделі inception3

Висока точність може бути зумовлено тим, що зображення з набору даних Chest X-ray Images значно відрізняються від зображень з набору даних ImageNet. Згорткові частини попередньо навчених мереж виділяють ознаки, які підходять для зображень ImageNet, але не підходять для зображень з набору даних Chest X-ray Images, використаного у даній роботі.

Згідно з результатами, модель Inception v3 досягла найвищої ефективності в чотирикласовій класифікації. Загальна точність розпізнавання 96%. Час, необхідний для навчання моделі Inception v3 на платформі Google Colab із графічним процесором, становило 100 хвилин. Для навчання використовувалось 25.120.460 параметра.

Модель VGG-16 використовувала 119.562.244 параметри для досягнення загальної точності розпізнавання 90%. Час, необхідний для навчання моделі VGG-16 на платформі Google Colab із графічним процесором, становив 70 хвилин. Основним недоліком мереж VGG є те, що

вони повільно навчаються через свою глибину, вимагають дуже великої кількості параметрів і тому повільні та часто створюють дуже великі моделі.

ResNet50 показала найменшу точність - 81.1% за 30 епох. Якщо запустити навчання протягом більшої кількості епох точність з часом покращиться. Час, необхідний для навчання моделі становив 109 хвилин.

3.7 Висновки до розділу 3

Однією з важливих проблем у автоматичному виявленні COVID-19 на рентгенівських зображеннях є вибір відповідної архітектури глибокого навчання ШІ. Таким чином, у цьому документі оцінюється ефективність шести високоефективних і точних моделей навчання передачі (VGG-16, ResNet-50, Inception3) для раннього виявлення COVID-19 на рентгенівському знімку легень. зображення. У цьому дослідженні розглядалася багатокласова класифікація рентгенограм на COVID-19, не-COVID-пневмонію, помутніння легень та нормальний стан. При розмірі вибірки 21165 рентгенограм легень лише 80% було використано для навчального процесу. Ефективність моделей навчання переносу оцінювалася з точки зору точності, чутливості, прецизійності та оцінки F1.

Класифікатори були реалізовані за допомогою мови програмування Python і біблотек машинного навчання та існуючих моделей для розпізнавання зображень. Вони дозволяють завантажити рентгенівське зображення, що буде класифіковане нейронною мережею і з певною ймовірністю віднесене до одного з чотирьох класів – COVID19, помутніння легень, норма, вірусна пневмонія. При навчанні на GPU за допомогою платформи Google Colaboratory на дотренування кожної моделі було витрачено приблизно по годині.

З трьох тренованих моделей Inception v3 показала високу точність 96%. Найнижча точність була отримана з мережею ResNet50 (81,1%). Зображення COVID19 найкраще класифікувала модель Inception v3 – 98,6% .

Для підвищення точності системи можна розширити діапазон представлених у вибірці зображень та додати у створену систему додаткові дані про пацієнтів, наприклад про їх стать і вік. На класифікацію мережею одного зображення, за допомогою графічного перетворювача, потрібно менше 1 секунди, що дає можливість отримати від неї відповідь фактично миттєво.

ВИСНОВКИ

У роботі розглянуто клінічну картину захворювання COVID-19, діагностичні показники, загальні принципи ренгенографії. Проведено аналіз клінічно важливих ознак при моніторингу COVID-19 за допомогою рентгену.

Розглянуто архітектуру глибоких нейронних мереж, методи машинного навчання та розглянуто існуючі моделі для розпізнавання зображень. Виходячи з картини захворювання, поставлено задачу розробки автоматичного класифікатора рентгенівських зображень на основі методів машинного навчання. Досліджено використання штучних нейронних мереж у задачі діагностики захворювань за рентгенівськими зображеннями, що передбачає можливість за рентгенівським знімком пацієнта автоматично визначити наявність на ньому захворювання, а саме COVID, помутніння легень, норма, вірусна пневмонія.

Оглянуто існуючі методи глибокого навчання і алгоритми попередньої обробки рентгенівських зображень. Підготовано придатний набір даних для навчання. Проведено експерименти шляхом донавчання нейронних мереж на даних з попередньою обробкою. Для вирішення поставленої задачі було застосовано згорткові нейронні мережі та метод передавального навчання, у якості дотренованих моделей виходять моделі - ResNet50, VGG16 та Inception3.

Згорткові нейронні мережі можуть бути успішно застосовані для задачі діагностики захворювань за рентгенівськими зображеннями. Розроблена в ході даної роботи система дозволяє подавати на вхід зображення, аналізувати їх і розпізнати у них чотири класи – COVID, помутніння легень, нормальний стан та вірусна пневмонія.

Точність систем під час тестування склала:

- ResNet – 81.1%;

- VGG16 – 90.3%;
- InceptionV3 – 96.3%.

З поставленою задачою класифікації зображень хворих COVID найкраще справилась модель InceptionV3 - 98.6%. В порівнянні з подібними експериментами [67,68] дана точність класифікації рентгенівських зображень COVID, для моделі InceptionV3, являється найвищою. Але такі системи класифікації не можна використовувати як основне джерело інформації, через специфіку захворювань, при діагностиці. Класифікатори на основі штучного інтлекту варто використовувати як додатковий метод для визначення патологій при захворюваннях.

При розробці програми було використано мову програмування Python і бібліотеки PyTorch. Для тренування нейронної мережі використано безкоштовну платформу для машинного навчання і нейронних мереж Google Colaboratory, де користувачам безкоштовно надається доволі потужний GPU. Навчання нейронних мереж на GPU дозволяє значно збільшити швидкість навчання, для даних моделей. Створена програма являє собою модуль, який може бути використаний у додатках-асистентах для лікарів, медичному обладнанні тощо. Для підвищення точності системи можна збільшити навчальну вибірку, розширити діапазон представлених у ній зображень та додати у створену систему додаткові дані про пацієнтів. Надалі можна буде оптимізувати кількість шарів і вузлів, а також кількість параметрів у кожному шарі. Вибір швидкості навчання, кількості епох, інтенсивності регуляризації, архітектури мережевих рівнів і оптимізації вузлів вимагають додаткових знань і навичок. Використовуючи інформацію з різних джерел, у майбутньому можна створити надійніші моделі.

ПЕРЕЛІК ПОСИЛАНЬ

[1] – A. Bernheim, X. Mei, M. Huang, Y. Yang, Z.A. Fayad, N. Zhang, K. Diao, B. Lin, X. Zhu, K. Li, S. Li, Chest CT findings in coronavirus disease-19 (COVID-19): relationship to duration of infection. Radiology. 2020 Feb 20:200463.

[2] - X. Xie, Z. Zhong, W. Zhao, C. Zheng, F. Wang, J. Liu, Chest CT for typical 2019-nCoV pneumonia: relationship to negative RT-PCR testing. Radiology. 2020 Feb 12:200343.

[3] - Y. Fang, H. Zhang, J. Xie, M. Lin, L. Ying, P. Pang, W. Ji, Sensitivity of Chest CT for COVID-19: comparison to RT-PCR. Radiology2020 Feb. 19:200432. [Epub ahead of print]. Режим доступу до ресурсу: https://pubs.rsna.org/doi/full/10.1148/radiol.2020200432.

[4] - X. Xie, Z. Zhong, W. Zhao, C. Zheng, F. Wang, J. Liu, Chest CT for typical 2019-nCoV pneumonia: relationship to negative RT-PCR testing. Radiology 2020 Feb. 7 (Epub ahead of print). Режим доступу до ресурсу: https://pubs.rsna.org/doi/10.1148/radiol.2020200343.

[5] - Клиническая характеристика 1007 больных тяжелой SARS-CoV-2 пневмонией, нуждавшихся в респираторной поддержке П.В. Глыбочко, В.В. Фомин, С.Н. Авдеев Режим доступу до ресурсу: https://clinpharm-journal.ru/files/articles/klinicheskaya-harakteristika-1007-bolnyh-tyazheloj-sars-cov-2-pnevmoniej-nuzhdavshihsya-v-respiratornoj-podderzhke.pdf.

[6] - Americanl Journal of Roentgenology: Coronavirus Disease 2019 (COVID-19): A Systematic Review of Imaging Findings in 919 Patients. Режим доступу до ресурсу: https://www.ajronline.org/doi/full/10.2214/AJR.20.23034.

[7] - THE LANCET Respiratory Medicine :Cytokine elevation in severe and critical COVID-19: a rapid systematic review, meta-analysis, and comparison with other inflammatory syndromes. Режим доступу до ресурсу:

https://www.thelancet.com/journals/lanres/article/PIIS2213-2600(20)30404-5/fulltext.

[8] - Lung under attack by COVID-19-induced cytokine storm: pathogenic mechanisms and therapeutic implications. Режим доступу до ресурсу: https://pubmed.ncbi.nlm.nih.gov/32539627/

[9] - Surveillances V. The epidemiological characteristics of an outbreak of 2019 novel coronavirus diseases (COVID-19)—China, 2020. China CDC Wkly. 2020;2:113–22.

[10] - Wu F, Zhao S, Yu B, Chen Y-M, Wang W, Song Z-G, et al. A new coronavirus associated with human respiratory disease in China. Nature. 2020;579:265–9.

[11] - Huang C, Wang Y, Li X, Ren L, Zhao J, Hu Y, et al. Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China. Lancet. 2020;395:497–506.

[12] - Organization H. Report of the WHO-China Joint Mission on Coronavirus Disease 2019 (COVID-19) 2020.

[13] - What to Know About COVID-19 and Pneumonia. Режим доступу до ресурсу: https://www.healthline.com/health/coronavirus-pneumonia.

[14] - COVID-19 and Sepsis. Режим доступу до ресурсу: https://jamanetwork.com/journals/jamainternalmedicine/fullarticle/2767939.

[15] - Lopes-Pacheco M, Silva PL, Cruz FF, Battaglini D, Robba C, Pelosi P, et al.. Pathogenesis of Multiple Organ Injury in COVID-19 and Potential Therapeutic Strategies. Front Physiol (2021).

[16] - Vianello A, Guarnieri G, Braccioni F, Lococo S, Molena B, Cecchetto A, et al.. The Pathogenesis, Epidemiology and Biomarkers of Susceptibility of Pulmonary Fibrosis in COVID-19 Survivors. Clin Chem Lab Med.

[17] – Chen N, Zhou M, Dong X, Qu J, Gong F, Han Y, et al.. Epidemiological and Clinical Characteristics of 99 Cases of 2019 Novel Coronavirus Pneumonia in Wuhan, China: A Descriptive Study. Lancet (2020).

[18] - Zhou M, Zhang X, Qu J. Coronavirus Disease 2019 (COVID-19): A Clinical Update. Front Med (2020).

[19] - Orlandi D, Battaglini D, Robba C, Viganò M, Bergamaschi G, Mignatti T, et al.. COVID-19 Phenotypes, Lung Ultrasound, Chest Computed Tomography, and Clinical Features in Critically Ill Mechanically Ventilated Patients. Ultrasound Med Biol (2021).

[20] - Pelosi P, Ball L, Barbas CSV, Bellomo R, Burns KEA, Einav S, et al.. Personalized Mechanical Ventilation in Acute Respiratory Distress Syndrome. Crit Care (2021).

[21] - Tonelli R, Marchioni A, Tabbì L, Fantini R, Busani S, Castaniere I, et al.. Spontaneous Breathing and Evolving Phenotypes of Lung Damage in Patients With COVID-19: Review of Current Evidence and Forecast of a New Scenario. J Clin Med (2021).

[22] - George PM, Wells AU, Jenkins RG. Pulmonary Fibrosis and COVID-19: The Potential Role for Antifibrotic Therapy. Lancet Respir Med (2020).

[23] - Cione E, Siniscalchi A, Gangemi P, Cosco L, Colosimo M, Longhini F, et al.. Neuron-Specific Enolase Serum Levels in COVID-19 Are Related to the Severity of Lung Injury. PLoS One (2021).

[24] - Alay H, Laloglu E. The Role of Angiopoietin-2 and Surfactant Protein-D Levels in SARS-CoV-2-Related Lung Injury: A Prospective, Observational, Cohort Study. J Med Virol (2021).

[25] - Kerget F, Kerget B, İba Yılmaz S, Kızıltunç A. Evaluation of the Relationship Between TREM-1/TREM-2 Ratio and Clinical Course in COVID-19 Pneumonia. Int J Clin Pract (2021).

[26] - Martinez Mesa A, Cabrera César E, Martín-Montañez E, Sanchez Alvarez E, Lopez PM, Romero-Zerbo Y, et al.. Acute Lung Injury Biomarkers in the Prediction of COVID-19 Severity: Total Thiol, Ferritin and Lactate Dehydrogenase. Antioxidants (2021).

[27] - Topp G, Bouyea M, Cochran-Caggiano N, Ata A, Torres P, Jacob J, et al.. Biomarkers Predictive of Extubation and Survival of COVID-19 Patients. Cureus (2021).

[28] - Chest Imaging Appearance of COVID-19 Infection. Режим доступу до ресурсу: https://pubs.rsna.org/doi/10.1148/ryct.2020200028#fig1b.

[29] - Lei J, Li J, Li X, Qi X. CT Imaging of the 2019 Novel Coronavirus. Режим доступу до ресурсу: (2019-nCoV) Pneumonia. Radiology 2020 https://www.sciencedirect.com/science/article/pii/S0140673620302117.

[30] - Udugama, B. et al. Diagnosing covid-19: The disease and tools for detection. ACS Nano 14, 3822–3835. Режим доступу до ресурсу: https://webmed.irkutsk.ru/doc/pdf/covid19what.pdf.

[31] - Inui AF, Jitsu M, Kunishima N, Watanabe S, Suzuki Y, Umeda S, et al. Chest CT findings in cases from the cruise ship “Diamond Princess” with coronavirus disease 2019 (COVID-19) Radiology. Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7233452/.

[32] - Zhu W, Xie K, Lu H, Xu L, Zhou S, Fang S. Initial clinical features of suspected Coronavirus Disease 2019 in two emergency departments outside of Hubei, China. J Med Virol. Режим доступу до ресурсу: https://onlinelibrary.wiley.com/doi/full/10.1002/jmv.25763.

[33] - Chest x-ray findings in 636 ambulatory patients with COVID-19 presenting to an urgent care center: a normal chest x-ray is no guarantee. Journal of Urgent Care Medicine. Режим доступу до ресурсу https://www.jucm.com/wp-content/uploads/2021/02/2020-14813-18-Original-Research.pdf.

[34] - Kuruvilla, J. & Gunavathi, K. Lung cancer classification using neural networks for CT images. Comput. Meth. Prog. Biomed. 113, 202–209 (2014).. Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0169260713003532.

[35] - Rajpurkar, P. et al. Chexnet: radiologist-level pneumonia detection on chest x-rays with deep learning. Режим доступу до ресурсу: https://arxiv.org/abs/1711.05225.

[36] - Gulshan, V. et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA 316, 2402–2410 (2016). Режим доступу до ресурсу: https://jamanetwork.com/journals/jama/fullarticle/2588763/.

[37] - A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Режим доступу до ресурсу: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[38] - Artificial intelligence distinguishes COVID-19 from community acquired pneumonia on chest CT. Radiology 2020. Режим доступу до ресурсу: https://covid-19.conacyt.mx/jspui/handle/1000/5328.

[39] - Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Режим доступу до ресурсу: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

[40] - Rajaraman, S.; Candemir, S.; Kim, I.; Thoma, G.; Antani, S. Visualization and interpretation of convolutional neural network predictions in detecting pneumonia in pediatric chest radiographs. Режим доступу до ресурсу: https://www.mdpi.com/2076-3417/8/10/1715.

[41] - Kermany, D.S.; Goldbaum, M.; Cai, W.; Valentim, C.C.; Liang, H.; Baxter, S.L.; McKeown, A.; Yang, G.; Wu, X.; Yan, F.; et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/9376253.

[42] - Song, Y.; Zheng, S.; Li, L.; Zhang, X.; Zhang, X.; Huang, Z.; Chen, J.; Zhao, H.; Jie, Y.; Wang, R.; et al. Deep learning enables accurate diagnosis of novel coronavirus (COVID-19) with CT images. Режим доступу до ресурсу: https://www.st.com/en/microcontrollers-microprocessors/stm32-wireless-mcus.html

[43] - Zheng, C.; Deng, X.; Fu, Q.; Zhou, Q.; Feng, J.; Ma, H.; Liu, W.; Wang, X. Deep learning-based detection for COVID-19 from chest CT using weak label. medRxiv 2020. Режим доступу до ресурсу: https://www.medrxiv.org/content/10.1101/2020.03.12.20027185v2.

[44] - Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/5206848.

[45] - Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems.

[46] - Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations. Режим доступу до ресурсу: https://arxiv.org/abs/1409.1556.

[47] - Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int. J. Uncertain. Fuzziness-Knowl. Syst. 1998. Режим доступу до ресурсу: https://www.worldscientific.com/doi/abs/10.1142/s0218488598000094.

[48] - ResNet-50: The Basics and a Quick Tutorial. Режим доступу до ресурсу: https://datagen.tech/guides/computer-vision/resnet-50/.

[49] - Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.

[50] - Tan, M.; Le, Q.V. Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv 2019. Режим доступу до ресурсу: http://proceedings.mlr.press/v97/tan19a.html.

[51] - Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. Режим доступу до ресурсу: https://arxiv.org/abs/1602.07360.

[52] - Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City.

[53] - Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. Режим доступу до ресурсу: https://arxiv.org/abs/1704.04861.

[54] - Huang, G.; Liu, S.; Van der Maaten, L.; Weinberger, K.Q. Condensenet: An efficient densenet using learned group convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[55] - Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Режим доступу до ресурсу.

[56] - Chollet, F. Xception: Deep learning with depthwise separable convolutions.. Режим доступу до ресурсу: https://openaccess.thecvf.com/content_cvpr_2017/html/Chollet_Xception_Deep_Learning_CVPR_2017_paper.html.

[57] - Platform-aware neural architecture search for mobile. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Режим доступу до ресурсу: https://openaccess.thecvf.com/content_CVPR_2019/html/Tan_MnasNet_Platform-Aware_Neural_Architecture_Search_for_Mobile_CVPR_2019_paper.

[58] - Zagoruyko, S.; Komodakis, N. Wide residual networks. Режим доступу до ресурсу: https://arxiv.org/abs/1605.07146.

[59] - Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[60] - COVID-19 Radiography Database. Режим доступу до ресурсу: https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database.

[61] - BIMCV-COVID19+: a large annotated dataset of RX and CT images of COVID19 patients. Режим доступу до ресурсу: https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/#1590858128006-9e640421-6711.

[62] - Covid-19-image-repository. Режим доступу до ресурсу: https://github.com/ml-workgroup/covid-19-image-repository/tree/master/png.

[63] - Covid-19-images. Режим доступу до ресурсу: https://sirm.org/category/senza-categoria/covid-19/.

[64] - COVID-CXNet. Режим доступу до ресурсу: https://github.com/armiro/COVID-CXNet.

[65] - RSNA Pneumonia Detection Challenge. Режим доступу до ресурсу: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data.

[66] - Chest X-Ray Images (Pneumonia). Режим доступу до ресурсу: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia.

[67] - Deep transfer learning for COVID-19 detection and infection localization with superpixel based segmentation. Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S221067072100528X#b46.

[68] - Automated Deep Transfer Learning-Based Approach for Detection of COVID-19 Infection in Chest X-rays. Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S1959031820301172.

Додаток А

Вихідний код програми

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


import warnings
warnings.filterwarnings('ignore')
!pip3 install torch pip3 install torch==1.3.1+cpu torchvision==0.4.2+cpu -
f
import torch
import torch.nn as nn
from torchvision import transforms
from torch.utils.data import Dataset, DataLoader
from torch.utils.data import Subset, WeightedRandomSampler
from torchvision.datasets import ImageFolder
from torchvision import models
import torch.optim as optim
import time
from tqdm import tqdm
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import seaborn as sns
import numpy as np
import os
import PIL
import gc
import torchvision
from torchvision.io import read_image
from torchvision.utils import make_grid
```

DATA EXTRACTION FROM RAR==========

```
!pip3 install patool
import patoolib
!pip3 install pyunpack
from pyunpack import Archive

Archive('/content/drive/MyDrive/Colab Notebooks/Dataset.rar').extractall('
/content/drive/MyDrive/Colab Notebooks/Dataset')
```

SAMPLES VISUALISATION==========

```
covid_example = '/content/drive/MyDrive/Colab Notebooks/Dataset/COVID-
19_Radiography_Dataset/COVID/images/COVID-1.png'
```

```
normal_example = '/content/drive/MyDrive/Colab Notebooks/Dataset/COVID-
19_Radiography_Dataset/Normal/images/Normal-1.png'
opacity_example = '/content/drive/MyDrive/Colab Notebooks/Dataset/COVID-
19_Radiography_Dataset/Lung_Opacity/images/Lung_Opacity-1.png'
pneumonia_example = '/content/drive/MyDrive/Colab Notebooks/Dataset/COVID-
19_Radiography_Dataset/Viral Pneumonia/images/Viral Pneumonia-1.png'

covid_example = PIL.Image.open(covid_example)
normal_example = PIL.Image.open(normal_example)
opacity_example = PIL.Image.open(opacity_example)
pneumonia_example = PIL.Image.open(pneumonia_example)

fig, axs = plt.subplots(2, 2, figsize=(15,7), dpi=150)

axs[0, 0].imshow(covid_example)
axs[0, 0].set_title('COVID')

axs[0, 1].imshow(normal_example)
axs[0, 1].set_title('Normal')

axs[1, 0].imshow(opacity_example)
axs[1, 0].set_title('Lung Opacity')

axs[1, 1].imshow(pneumonia_example)
axs[1, 1].set_title('Viral Pneumonia')
plt.show()

print(covid_example.size)
print(covid_example.getextrema())
covid_example
```

DATA PREPARATION
==========

```
class CovidDataset(Dataset):
    def __init__(self, dataset, transform):
        self.dataset = dataset
        self.transform = transform

    def __getitem__(self, idx):
        data = dataset[idx][0]
        label = dataset[idx][1]

        data = self.transform(data)
        label = torch.tensor(label)

        return data, label

    def __len__(self):
        return len(self.dataset)
```

```
#TRANSFORMATIONS FOR RESNET50 AND VGG16
train_transform = transforms.Compose([transforms.ToTensor(),
                                      transforms.RandomRotation(degrees=15
),
                                      transforms.RandomHorizontalFlip(),
                                      transforms.Resize(224),
                                      transforms.Normalize([0.485, 0.456,
0.406], [0.229, 0.224, 0.225])
                                      ])

eval_transform = transforms.Compose([transforms.ToTensor(),
                                     transforms.Resize(224),
                                     transforms.Normalize([0.485, 0.456, 0
.406], [0.229, 0.224, 0.225])
                                     ])


#TRANSFORMATIONS FOR INCEPTION_V3
train_transform = transforms.Compose([transforms.ToTensor(),
                                      transforms.RandomRotation(degrees=15
),
                                      transforms.RandomHorizontalFlip(),
                                      transforms.Resize(299),
                                      transforms.Normalize([0.485, 0.456,
0.406], [0.229, 0.224, 0.225])
                                      ])

eval_transform = transforms.Compose([transforms.ToTensor(),
                                     transforms.Resize(299),
                                     transforms.Normalize([0.485, 0.456, 0
.406], [0.229, 0.224, 0.225])
                                     ])

#Creating a dataset with all images:
dataset = ImageFolder('/content/drive/MyDrive/Colab Notebooks/Dataset/COVI
D-19_Radiography_Dataset') #full dataset
#dataset = ImageFolder('/content/drive/MyDrive/Colab Notebooks/COVID-
19_Radiography_Dataset1') # small test dataset
dataset


print(dataset.class_to_idx)
np.unique(dataset.targets, return_counts=True)
class_sample_counts = np.unique(dataset.targets, return_counts=True)[1]
class_sample_counts

weights = 1. / torch.tensor(class_sample_counts, dtype=torch.float)
print(weights)
```

```
print(weights.round())

samples_weights = weights[dataset.targets]
print(samples_weights)
print(samples_weights.size())

sampler = WeightedRandomSampler(weights=samples_weights, num_samples=len(s
amples_weights), replacement=True)

train_dataset = CovidDataset(dataset, train_transform)
val_dataset = CovidDataset(dataset, eval_transform)
test_dataset = CovidDataset(dataset, eval_transform)

train_size = 0.8

num_train = len(dataset)
indices = list(range(num_train))
split = int(np.floor(train_size * num_train))
split2 = int(np.floor((train_size+(1-train_size)/2) * num_train))
np.random.shuffle(indices)
train_idx, valid_idx, test_idx = indices[:split], indices[split:split2], i
ndices[split2:]

train_data = Subset(train_dataset, indices=train_idx)
val_data = Subset(val_dataset, indices=valid_idx)
test_data = Subset(test_dataset, indices=test_idx)

print(len(train_data))
print(len(val_data))
print(len(test_data))

#SAVING PREPARED DATA
save_path_train_data = '/content/drive/MyDrive/Colab Notebooks/train_data1
_FI.pt'
save_path_val_data = '/content/drive/MyDrive/Colab Notebooks/val_data1_FI.
pt'
save_path_test_data = '/content/drive/MyDrive/Colab Notebooks/test_data1_F
I.pt'

torch.save(train_data, save_path_train_data)
torch.save(val_data, save_path_val_data)
torch.save(test_data, save_path_test_data)

#LOADING PREPARED DATA
save_path_train_data = '/content/drive/MyDrive/Colab Notebooks/train_data1
.pt'
save_path_val_data = '/content/drive/MyDrive/Colab Notebooks/val_data1.pt'
save_path_test_data = '/content/drive/MyDrive/Colab Notebooks/test_data1.p
t'
```

```
train_data = torch.load(save_path_train_data)
val_data = torch.load(save_path_val_data)
test_data = torch.load(save_path_test_data)
train_dataloader = DataLoader(train_data, shuffle=True, batch_size=64, num
_workers=8)
val_dataloader = DataLoader(val_data, shuffle=True, batch_size=64, num_wor
kers=4)
test_dataloader = DataLoader(test_data, shuffle=False, batch_size=64, num_
workers=2)

dataloaders = {'train':train_dataloader, 'val':val_dataloader}

# VISUALISATION OF SOME PREPROCESSED DATA
def show_transform_images(dataset):
    loader = torch.utils.data.DataLoader(dataset, batch_size = 6, shuffle
= True)
    batch = next(iter(loader))
    images, labels = batch

    grid = torchvision.utils.make_grid(images, nrow=3)
    plt.figure( figsize = (11, 11) )
    plt.imshow( np.transpose(grid, (1,2,0)))
    print('labels: ', labels)

show_transform_images(train_data)
```

TRAINING
==========

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(f"Computation device: {device}\n")

#PREPARATION FOR RESNET50 TRAINING
model = models.resnet50(pretrained=True)

for param in model.parameters(): #?
    param.requires_grad = False

num_ftrs = model.fc.in_features # gets last layer's features
model.fc = nn.Sequential(nn.Linear(num_ftrs, 4)) # changes output to 4

model = model.to(device)  # pass model to GPU

#PREPARATION FOR VGG16 TRAINING
model = models.vgg16(pretrained=True)
# Here we are freezing the feature parameters and using same as in vgg16.
These parameters are very good as this model is trained on 14 million imag
es for 2 weeks with many GPUs
for param in model.features.parameters():
```

```
  param.requires_grad = False # the feature layers do not require any grad
ient.
  #We take the last fully connected layer from vgg16 and replace it with o
ur layer as we need to classify only 2 different categories

n_inputs = model.classifier[6].in_features # the i/p features of our class
ifier model
last_layer = nn.Linear(n_inputs, 4)  # new layer that we want to put in th
ere.
model.classifier[6] = last_layer  # replacing last layer of vgg16 with our
 new layer
model.to(device)  # Put the model in device for higher processing power
print(model.classifier[6].out_features)

#PREPARATION FOR INCEPTION_V3 TRAINING
model = models.inception_v3(pretrained=True)

num_ftrs = model.fc.in_features # gets last layer's features
model.fc = nn.Sequential(nn.Linear(num_ftrs, 4)) # changes output to 4

model = model.to(device)  # pass model to GPU

model.aux_logits=False
torch.cuda.empty_cache()
gc.collect()
torch.cuda.memory_summary(device=None, abbreviated=False)

# hyperparameters
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)

#test
#save_path = '/content/drive/MyDrive/Colab Notebooks/test.pt'
#history_path = '/content/drive/MyDrive/Colab Notebooks/loss_acc_history/t
estHistory.pt'

#resnet50
save_path = '/content/drive/MyDrive/Colab Notebooks/model1_2.pt'
history_path = '/content/drive/MyDrive/Colab Notebooks/loss_acc_history/re
snet50_history2.pt'

#vgg16
#save_path = '/content/drive/MyDrive/Colab Notebooks/model2_1.pt'
#history_path = '/content/drive/MyDrive/Colab Notebooks/loss_acc_history/v
gg16_history1.pt'

#inception_v3
#save_path = '/content/drive/MyDrive/Colab Notebooks/model3_1.pt'
#history_path = '/content/drive/MyDrive/Colab Notebooks/loss_acc_history/i
nception_v3_history1.pt'
```

```
epochs = 30
# training loop
history = {'train_loss': [], 'train_acc': [], 'val_loss': [], 'val_acc': [
]}

since = time.time()

best_acc = 0

for epoch in range(epochs):
     # Each epoch has a training and validation phase
      for phase in ['train', 'val']:
        if phase == 'train':
            model.train()  # Set model to training mode

        else:
            model.eval()  # Set model to evaluate mode


        running_loss = 0.0
        running_corrects = 0

        # Iterate over data
        loop = tqdm(dataloaders[phase], total=len(dataloaders[phase]), pos
ition=0, leave=True)
        for inputs, labels in (loop):
            try:
                inputs = inputs.to(device)
            except PIL.UnidentifiedImageError as e:
                print(e)

            labels = labels.to(device)

            # zero the parameter gradients
            optimizer.zero_grad()

            # forward
            # track history only in train
            with torch.set_grad_enabled(phase == 'train'):
                outputs = model(inputs)
                _, preds = torch.max(outputs, 1)
                loss = criterion(outputs, labels)

                 # backward + optimize only if in training phase
                if phase == 'train':
                    loss.backward()
                    optimizer.step()

            # statistics
            running_loss += loss.item() * inputs.size(0)
```

```
            running_corrects += torch.sum(preds == labels.data)

            # update tqdm
            loop.set_description(f'Epoch {epoch+1}/{epochs} [{phase}]')

        epoch_loss = running_loss / len(dataloaders[phase].dataset)
        epoch_acc = running_corrects.double() / len(dataloaders[phase].dat
aset)

        if phase == 'train':
            history['train_loss'].append(epoch_loss)
            history['train_acc'].append(epoch_acc)

        elif phase == 'val':
            history['val_loss'].append(epoch_loss)
            history['val_acc'].append(epoch_acc)

        print(' {} Loss: {:.4f} Acc: {:.4f}'.format(phase, epoch_loss, epo
ch_acc))

        if epoch_acc > best_acc:
            best_acc = epoch_acc

            #Resnet Saving best model
            torch.save(model.state_dict(), os.path.join('/content/drive/My
Drive/Colab Notebooks/best_model', 'model_resnet50_attempt2_' + 'epoch-
{}.pt'.format(epoch)))

            #VGG16 Saving best model
            #torch.save(model.state_dict(), os.path.join('/content/drive/M
yDrive/Colab Notebooks/best_model', 'model_vgg16_' + 'epoch-
{}.pt'.format(epoch)))

            #Inception Saving best model
            #torch.save(model.state_dict(), os.path.join('/content/drive/M
yDrive/Colab Notebooks/best_model', 'model_inception_' + 'epoch-
{}.pt'.format(epoch)))

    print()

time_elapsed = time.time() - since
print('\n Training complete in {:.0f}m {:.0f}s \n'.format(time_elapsed //
60, time_elapsed % 60))

torch.save(model.state_dict(), save_path)
torch.save(history, history_path)
```

RESULT VISUALISATION
==========

```
#history_path = '/content/drive/MyDrive/Colab Notebooks/loss_acc_history/t
estHistory.pt'
history_load = '/content/drive/MyDrive/Colab Notebooks/loss_acc_history/re
snet50_history2.pt'
#history_load = '/content/drive/MyDrive/Colab Notebooks/loss_acc_history/v
gg16_history1.pt'
#history_load = '/content/drive/MyDrive/Colab Notebooks/loss_acc_history/i
nception_v3_history1.pt'

history1 = torch.load(history_load, map_location=torch.device('cpu'))
#history1

plt.style.use('ggplot')

# loss and accuracy
epochs = list(range(1, len(history1['train_loss'])+1))

plt.figure(figsize=(7,7), dpi=200)
plt.plot(epochs, history1['train_loss'], c='slateblue', label='Training lo
ss', linestyle='--')
plt.plot(epochs, history1['train_acc'], c='salmon', label='Training accura
cy', linestyle='--')

plt.plot(epochs, history1['val_loss'], c='blue', label='Validation loss')
plt.plot(epochs, history1['val_acc'], c='red', label='Validation accuracy'
)

plt.xlabel('Epochs')
plt.ylabel('Loss/Accuracy')

plt.legend()
plt.title('Loss/Accuracy over Epochs')

plt.savefig('/content/drive/MyDrive/Colab Notebooks/resnet50_loss_acc2.pdf
')
#plt.savefig('/content/drive/MyDrive/Colab Notebooks/vgg16_loss_acc.pdf')
#plt.savefig('/content/drive/MyDrive/Colab Notebooks/inception_v3_loss_acc
.pdf')
```

TEST
==========

```
def load_model(model_path):
    # instantiate the model

    model = models.resnet50(pretrained=True)
    #model = models.vgg16(pretrained=True)
    #model = models.inception_v3(pretrained=True)

        # Features for resnet50 & inception_v3
```

```
    num_ftrs = model.fc.in_features
    model.fc = nn.Sequential(nn.Linear(num_ftrs, 4))
    model = model.to(device)

        # Features for VGG16
    #n_inputs = model.classifier[6].in_features # the i/p features of our
classifier model
    #last_layer = nn.Linear(n_inputs, 4)  # new layer that we want to put
in there.
    #model.classifier[6] = last_layer  # replacing last layer of vgg16 wit
h our new layer
    #model.to(device)  # Put the model in device for higher processing pow
er

    # load the trained model weights
    model.load_state_dict(torch.load(model_path, map_location=torch.device
(device)))
    print('model loaded successfully!')

    return model

#save_path = '/content/drive/MyDrive/Colab Notebooks/test.pt'
save_path = '/content/drive/MyDrive/Colab Notebooks/best_model/model_resne
t50_attempt2_epoch-29.pt' #Resnet50
#save_path = '/content/drive/MyDrive/Colab Notebooks/best_model/model_vgg1
6_epoch-14.pt' #VGG_16
#save_path = '/content/drive/MyDrive/Colab Notebooks/best_model/model_ince
ption_epoch-12.pt' #Inception_V3

model = load_model(save_path)

model.eval()

true_labels = []
predictions = []

with torch.no_grad():
    loop = tqdm(test_dataloader, total=len(test_dataloader), position=0, l
eave=True)
    for inputs, labels in loop:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        preds = preds.cpu().numpy()
        labels = labels.cpu().numpy()

        true_labels.append(labels)
        predictions.append(preds)
```

```
    true_labels = np.concatenate(true_labels)
    predictions = np.concatenate(predictions)
```

CONFUSION MATRIX==========

```
import pandas as pd

cm_path = '/content/drive/MyDrive/Colab Notebooks/test_cm/resnet50_loss_ac
c2.pdf'
#cm_path = '/content/drive/MyDrive/Colab Notebooks/test_cm/vgg16_loss_acc.
pdf'
#cm_path = '/content/drive/MyDrive/Colab Notebooks/test_cm/inception_v3_lo
ss_acc.pdf'

cf_matrix = confusion_matrix(true_labels, predictions)

class_names = ('COVID', 'Lung Opacity', 'Normal', 'Viral Pneumonia')
dataframe = pd.DataFrame(cf_matrix, index=class_names, columns=class_names
)
print(dataframe)

group_counts = ['{0:0.0f}'.format(value) for value in cf_matrix.flatten()]
group_percentages = ['({0:.2%})'.format(value) for value in cf_matrix.flat
ten()/np.sum(cf_matrix)]
labels = [f'{v1}\n{v2}' for v1, v2 in zip(group_counts, group_percentages)
]
labels = np.asarray(labels).reshape(4,4)

plt.figure(figsize=(8, 6))

sns.heatmap(dataframe, annot=labels, cbar=None,cmap="YlGnBu",fmt="")

plt.title("Матриця помилок"), plt.tight_layout()
plt.ylabel("Справжній клас"),
plt.xlabel("Прогнозований клас")
plt.show()
plt.savefig(cm_path)
```

POSITIVE AND NEGATIVE RESULT CALCULATION==========

```
Covid_TP = dataframe.iloc[0, 0]
Covid_FP = dataframe.iloc[0, 1] + dataframe.iloc[0, 2] + dataframe.iloc[0,
 3]
Covid_TN = dataframe.iloc[1, 1] + dataframe.iloc[1, 2] + dataframe.iloc[1,
 3] + dataframe.iloc[2, 1] + dataframe.iloc[2, 2] + dataframe.iloc[2, 3] +
 dataframe.iloc[3, 1] + dataframe.iloc[3, 2] + dataframe.iloc[3, 3]
```

```
Covid_FN = dataframe.iloc[1, 0] + dataframe.iloc[2, 0] + dataframe.iloc[3,
 0]

Lung_Opacity_TP = dataframe.iloc[1, 1]
Lung_Opacity_FP = dataframe.iloc[1, 0] + dataframe.iloc[1, 2] + dataframe.
iloc[1, 3]
Lung_Opacity_TN = dataframe.iloc[0, 0] + dataframe.iloc[0, 2] + dataframe.
iloc[0, 3] + dataframe.iloc[2, 0] + dataframe.iloc[2, 2] + dataframe.iloc[
2, 3] + dataframe.iloc[3, 0] + dataframe.iloc[3, 2] + dataframe.iloc[3, 3]

Lung_Opacity_FN = dataframe.iloc[0, 1] + dataframe.iloc[2, 1] + dataframe.
iloc[3, 1]

Normal_TP = dataframe.iloc[2, 2]
Normal_FP = dataframe.iloc[2, 0] + dataframe.iloc[2, 1] + dataframe.iloc[2
, 3]
Normal_TN = dataframe.iloc[0, 0] + dataframe.iloc[0, 1] + dataframe.iloc[0
, 3] + dataframe.iloc[1, 0] + dataframe.iloc[1, 1] + dataframe.iloc[1, 3]
+ dataframe.iloc[3, 0] + dataframe.iloc[3, 1] + dataframe.iloc[3, 3]
Normal_FN = dataframe.iloc[0, 2] + dataframe.iloc[1, 2] + dataframe.iloc[3
, 2]

Viral_Pneumonia_TP = dataframe.iloc[3, 3]
Viral_Pneumonia_FP = dataframe.iloc[3, 0] + dataframe.iloc[3, 1] + datafra
me.iloc[3, 2]
Viral_Pneumonia_TN = dataframe.iloc[0, 0] + dataframe.iloc[0, 1] + datafra
me.iloc[0, 2] + dataframe.iloc[1, 0] + dataframe.iloc[1, 1] + dataframe.il
oc[1, 2] + dataframe.iloc[2, 0] + dataframe.iloc[2, 1] + dataframe.iloc[2,
 2]
Viral_Pneumonia_FN = dataframe.iloc[0, 3] + dataframe.iloc[1, 3] + datafra
me.iloc[2, 3]
```

METRICS CALCULATION
==========

```
def compare(arr1, arr2):
  count = 0

  for x, y in zip(arr1, arr2):
    if x != y:
      count += 1

  return count

amount_of_prediction = len(true_labels)
false_prediction = compare(true_labels, predictions)
true_prediction = len(true_labels) - false_prediction
accuracy = true_prediction / (true_prediction + false_prediction)

#precision = TP / (TP + FP)
```

```
#recall = TP / (TP + FN)
#f1score = (2 * precision * recall) / (precision + recall)

print("All prediction:", amount_of_prediction)
print("True prediction:", true_prediction)
print("False prediction:", false_prediction)
print("Accuracy:\t", accuracy)

print("Covid prediction")
print("Precision:\t", Covid_TP / (Covid_TP + Covid_FP))
print("Recall:\t\t", Covid_TP / (Covid_TP + Covid_FN))
print("F1 Score:\t", (2 * (Covid_TP / (Covid_TP + Covid_FP)) * (Covid_TP /
 (Covid_TP + Covid_FN))) / ((Covid_TP / (Covid_TP + Covid_FP)) + (Covid_TP
 / (Covid_TP + Covid_FN))))

print("Lung opacity prediction")
print("Precision:\t", Lung_Opacity_TP / (Lung_Opacity_TP + Lung_Opacity_FP
))
print("Recall:\t\t", Lung_Opacity_TP / (Lung_Opacity_TP + Lung_Opacity_FN)
)
print("F1 Score:\t", (2 * (Lung_Opacity_TP / (Lung_Opacity_TP + Lung_Opaci
ty_FP)) * (Lung_Opacity_TP / (Lung_Opacity_TP + Lung_Opacity_FN))) / ((Lun
g_Opacity_TP / (Lung_Opacity_TP + Lung_Opacity_FP)) + (Lung_Opacity_TP / (
Lung_Opacity_TP + Lung_Opacity_FN))))

print("Normal prediction")
print("Precision:\t", Normal_TP / (Normal_TP + Normal_FP))
print("Recall:\t\t", Normal_TP / (Normal_TP + Normal_FN))
print("F1 Score:\t", (2 * (Normal_TP / (Normal_TP + Normal_FP)) * (Normal_
TP / (Normal_TP + Normal_FN))) / ((Normal_TP / (Normal_TP + Normal_FP)) +
(Normal_TP / (Normal_TP + Normal_FN))))

print("Viral pneumonia prediction")
print("Precision:\t", Viral_Pneumonia_TP / (Viral_Pneumonia_TP + Viral_Pne
umonia_FP))
print("Recall:\t\t", Viral_Pneumonia_TP / (Viral_Pneumonia_TP + Viral_Pneu
monia_FN))
print("F1 Score:\t", (2 * (Viral_Pneumonia_TP / (Viral_Pneumonia_TP + Vira
l_Pneumonia_FP)) * (Viral_Pneumonia_TP / (Viral_Pneumonia_TP + Viral_Pneum
onia_FN))) / ((Viral_Pneumonia_TP / (Viral_Pneumonia_TP + Viral_Pneumonia_
FP)) + (Viral_Pneumonia_TP / (Viral_Pneumonia_TP + Viral_Pneumonia_FN))))
```

Додаток Б

Процес тренування моделей

RESNET50:

```
Epoch 1/30 [train]: 100%|■■■■■■■■■■| 265/265 [11:36<00:00,  2.63s/it]
 train Loss: 0.9375 Acc: 0.6287
Epoch 1/30 [val]: 100%|■■■■■■■■■■| 34/34 [02:58<00:00,  5.26s/it]
 val Loss: 0.7749 Acc: 0.6801

Epoch 2/30 [train]: 100%|■■■■■■■■■■| 265/265 [03:01<00:00,  1.46it/s]
 train Loss: 0.7219 Acc: 0.7329
Epoch 2/30 [val]: 100%|■■■■■■■■■■| 34/34 [00:14<00:00,  2.42it/s]
 val Loss: 0.6685 Acc: 0.7349

Epoch 3/30 [train]: 100%|■■■■■■■■■■| 265/265 [02:58<00:00,  1.48it/s]
 train Loss: 0.6431 Acc: 0.7675
Epoch 3/30 [val]: 100%|■■■■■■■■■■| 34/34 [00:15<00:00,  2.26it/s]
 val Loss: 0.6464 Acc: 0.7363

Epoch 4/30 [train]: 100%|■■■■■■■■■■| 265/265 [03:00<00:00,  1.47it/s]
 train Loss: 0.5998 Acc: 0.7872
Epoch 4/30 [val]: 100%|■■■■■■■■■■| 34/34 [00:14<00:00,  2.41it/s]
 val Loss: 0.6090 Acc: 0.7481

...

Epoch 29/30 [train]: 100%|■■■■■■■■■■| 265/265 [02:59<00:00,  1.48it/s]
 train Loss: 0.4395 Acc: 0.8389
Epoch 29/30 [val]: 100%|■■■■■■■■■■| 34/34 [00:15<00:00,  2.15it/s]
 val Loss: 0.4416 Acc: 0.8242

Epoch 30/30 [train]: 100%|■■■■■■■■■■| 265/265 [02:57<00:00,  1.50it/s]
 train Loss: 0.4375 Acc: 0.8398
Epoch 30/30 [val]: 100%|■■■■■■■■■■| 34/34 [00:14<00:00,  2.27it/s]
 val Loss: 0.4527 Acc: 0.8247

 Training complete in 109m 16s
```

VGG16:

```
Epoch 1/15 [train]: 100%|■■■■■■■■■■| 265/265 [11:38<00:00,  2.63s/it]
 train Loss: 0.4845 Acc: 0.8160
Epoch 1/15 [val]: 100%|■■■■■■■■■■| 34/34 [03:05<00:00,  5.47s/it]
 val Loss: 0.3264 Acc: 0.8762

Epoch 2/15 [train]: 100%|■■■■■■■■■■| 265/265 [03:34<00:00,  1.24it/s]
 train Loss: 0.3382 Acc: 0.8747
Epoch 2/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:15<00:00,  2.13it/s]
 val Loss: 0.3022 Acc: 0.8809

Epoch 3/15 [train]: 100%|■■■■■■■■■■| 265/265 [03:21<00:00,  1.31it/s]
```

train Loss: 0.2884 Acc: 0.8936
Epoch 3/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:21<00:00, 1.57it/s]
val Loss: 0.2727 Acc: 0.9041

Epoch 4/15 [train]: 100%|■■■■■■■■■■| 265/265 [03:35<00:00, 1.23it/s]
train Loss: 0.2680 Acc: 0.9001
Epoch 4/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:19<00:00, 1.74it/s]
val Loss: 0.2613 Acc: 0.9045

...

Epoch 13/15 [train]: 100%|■■■■■■■■■■| 265/265 [03:22<00:00, 1.31it/s]
train Loss: 0.1776 Acc: 0.9327
Epoch 13/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:17<00:00, 1.94it/s]
val Loss: 0.2377 Acc: 0.9149

Epoch 14/15 [train]: 100%|■■■■■■■■■■| 265/265 [03:17<00:00, 1.34it/s]
train Loss: 0.1666 Acc: 0.9369
Epoch 14/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:21<00:00, 1.59it/s]
val Loss: 0.2318 Acc: 0.9192

Epoch 15/15 [train]: 100%|■■■■■■■■■■| 265/265 [03:45<00:00, 1.18it/s]
train Loss: 0.1618 Acc: 0.9396
Epoch 15/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:23<00:00, 1.42it/s]
val Loss: 0.2480 Acc: 0.9164

Training complete in 70m 52s

INCEPTION V3:

Epoch 1/15 [train]: 100%|■■■■■■■■■■| 265/265 [26:04<00:00, 5.90s/it]
train Loss: 0.3074 Acc: 0.8895
Epoch 1/15 [val]: 100%|■■■■■■■■■■| 34/34 [06:38<00:00, 11.72s/it]
val Loss: 0.2070 Acc: 0.9319

Epoch 2/15 [train]: 100%|■■■■■■■■■■| 265/265 [04:27<00:00, 1.01s/it]
train Loss: 0.1611 Acc: 0.9458
Epoch 2/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:17<00:00, 1.91it/s]
val Loss: 0.1432 Acc: 0.9513

Epoch 3/15 [train]: 100%|■■■■■■■■■■| 265/265 [04:29<00:00, 1.02s/it]
train Loss: 0.1263 Acc: 0.9577
Epoch 3/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:16<00:00, 2.07it/s]
val Loss: 0.1940 Acc: 0.9405

...

Epoch 14/15 [train]: 100%|■■■■■■■■■■| 265/265 [04:29<00:00, 1.02s/it]
train Loss: 0.0403 Acc: 0.9858
Epoch 14/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:17<00:00, 1.93it/s]
val Loss: 0.1659 Acc: 0.9518

Epoch 15/15 [train]: 100%|■■■■■■■■■■| 265/265 [04:30<00:00, 1.02s/it]
train Loss: 0.0362 Acc: 0.9872
Epoch 15/15 [val]: 100%|■■■■■■■■■■| 34/34 [00:17<00:00, 1.92it/s]
val Loss: 0.1840 Acc: 0.9584

Training complete in 99m 52s