Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

ГЮИК.502840.006 ПЗ
(позначення документа)

Дослідження інтелектуальних методів аналізу рентгенівських знімків
(тема)

Виконав:
студент 2 курсу, групи СПРм-20-1

Каменєва К.С.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне проектування
(повна назва освітньої програми)

Керівник доц. Петрова Р.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри системотехніки _____ Гребеннік І. В.
(підпис) (прізвище, ініціали)

2021 р.

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Керівник кваліфікаційної роботи доц. Петрова Р. В.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Керівник кваліфікаційної роботи доц. Петрова Р. В.

Попередній захист проведений «14» грудня 2021 р.

Керівник кваліфікаційної роботи доц. Петрова Р. В.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*

Кафедра *Системотехніки*

Рівень вищої освіти *другий (магістерський)*

Спеціальність *122 — Комп'ютерні науки*
(код і повна назва)

Тип програми *Освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри *СТ*
проф. Гребеннік І.В.
«___ » *2021 р.*

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Каменєвій Ксенії Сергіївні*
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження інтелектуальних методів аналізу рентгенівських знімків* затверджена наказом по університету від *08.11.2021 р. № 1664 Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *15 грудня 2021 р.*
3. Вихідні дані до роботи *Тема дослідження, дані Інтернет-джерел та відомих проектів з тематики дослідження. Перелік використаних програмних засобів: мова програмування Python 3.8, середа розробки Jupyter Notebook 6.3.0, основна бібліотека для роботи Pytorch*
4. Перелік питань, що потрібно опрацювати в роботі: *4.1 вступ; 4.2 аналіз предметної галузі та постановка задачі; 4.3 згорткові нейронні мережі ; 4.4 модель глибинного навчання; 4.5 розробка архітектури нейронної мережі; 4.6 висновки; 4.7 перелік посилань; 4.8 текст програми; 4.9 графічні матеріали.*
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *5.1 відображення зображень рентгенівських знімків легенів з навчального набору, 5.2 структура штучної нейронної мережі, 5.3 функції активації та їх графіки, 5.4 архітектура нейронної мережі efficientnet, 5.5 продуктивність efficientnet для набору даних imagenet в порівнянні з іншими архітектурами мереж.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання завдання на кваліфікаційну роботу*	*08.11.2021*	*Виконано*
2	*Аналіз завдання, літератури та аналогів з теми кваліфікаційної роботи*	*10.11.2021-17.11.2021*	*Виконано*
2	*Дослідження існуючих методів розв'язання задачі*	*17.11.2021-21.11.2021*	*Виконано*
3	*Дослідження існуючих архітектур згроткових нейронних мереж*	*21.11.2021-24.11.2021*	*Виконано*
4	*Дослідження методів оптимізації моделі навчання*	*24.11.2021-28.11.2021*	*Виконано*
5	*Формування наборів даних*	*28.11.2021*	*Виконано*
6	*Розробка та налаштування моделі*	*29.11.2021*	*Виконано*
7	*Проведення експериментів*	*29.11.2021-05.12.2021*	*Виконано*
8	*Оформлення пояснювальної записки*	*06.12.2021*	*Виконано*
9	*Представлення на рецензування*	*09.12.2021*	*Виконано*
10	*Попередній захист*	*14.12.2021*	*Виконано*
11	*Подання роботи до екзаменаційної комісії*	*15.12.2021*	*Виконано*

Дата видачі завдання *08.11.2021 р.*

Студент _____ Каменєва К.С.
(підпис)

Керівник роботи _____ *доц. Петрова Р.В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 98 с., 30 рис., 2 дод., 26 джерел.

КЛАСИФІКАЦІЯ, COVID-19, НЕЙРОННІ МЕРЕЖІ, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, OBJECTS RECOGNITION

Об’єкт дослідження – методи аналізу та попереднього діагностування за рентгенівськими знімками.

Предмет дослідження – архітектура та параметри нейронних мереж що призначені для розпізнавання образів.

Мета роботи – дослідження параметрів нейронних мереж для покращення точності аналізу рентгенівських знімків.

Методи дослідження – використання глибоких згорткових нейронних мереж для класифікації об’єктів на зображеннях.

ABSTRACT

Explanatory note: 98 p., 30 fig., 2 ann., 26 sources.

CLASSIFICATION, COVID-19, NEURAL NETWORKS, IMAGE RECOGNITION, OBJECTS RECOGNITION

The object of study is the methods of analysis and preliminary diagnosis by X-rays.

The subject of the study is the architecture and parameters of neural networks designed for pattern recognition.

The purpose of the work is to study the parameters of neural networks to improve the accuracy of X-ray analysis.

Research methods – the use of deep convolutional neural networks to classify objects in images.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ШІ – Штучний Інтелект;

ІТ – Інформаційні технології;

КТ – Комп’ютерна томографія;

CNN – Convolutional neural network;

RNN – Recurrent neural network;

PCA – Principal Component Analysis;

FLOPS – кількість операцій з плаваючою комою в секунду;

ImageNet – набір даних від Google;

Transfer learning – підрозділ машинного навчання, метою якого є застосування знань, отриманих з одного завдання, до іншого цільового завдання;

Набір даних – колекція однотипних даних, що застосовується в задачах машинної обробки даних;

Машинне навчання – це підгалузь штучного інтелекту, яка часто застосовує статистичні прийоми для надання комп'ютерам здатності «навчатися»;

ВСТУП

Інформаційні технології (ІТ) відіграють важливу роль у сучасному особистому, комерційному та некомерційному використанні. Важливість ІТ іноді сприймається як належне, оскільки його застосування дуже поширене в усьому світі, наприклад, для спілкування, банківської справи, соціальних медіа, інвестицій, досліджень та багато іншого. Організації не можуть існувати сьогодні і розраховувати на те, щоб залишатися конкурентоспроможними без відповідних систем інформаційних технологій. Провідними тенденціями, що розвиваються сьогодні, є штучний інтелект, хмарні обчислення, блокчейн безпеки для дому та бізнесу, дрони та електрона комерція.

Впровадження систем штучного інтелекту (ШІ) у медицині – це один із найважливіших сучасних трендів світової охорони здоров'я. Технології штучного інтелекту докорінно змінюють світову систему охорони здоров'я, дозволяючи кардинально переробити систему медичної діагностики, розробку нових лікарських засобів, а також загалом підвищити якість послуг охорони здоров'я при одночасному зниженні витрат для медичних клінік.

Сучасні інструменти машинного навчання, що використовують штучні нейронні мережі для вивчення надзвичайно складних взаємозв'язків або технології глибокого навчання, часто перевершують людські можливості під час виконання медичних завдань. Технології ШІ допомагають медичним установам, керівникам та дослідникам задіяти мільйони медичних звітів, записів пацієнтів, клінічних досліджень та медичних журналів з метою отримання цінної інформації.

Розвиток алгоритмів машинного навчання надає широкі можливості у галузі автоматизації вирішення біомедичних завдань. Комп'ютерна обробка біомедичних зображень підвищує точність аналізу зображень, знижує роль

людського фактора при прийнятті рішень, дозволяє оцінити ефективність застосування терапії і в цілому покращує якість життя людей. Активно розвиваються біомедичні дослідження в галузі аналізу та розпізнавання зображень, отриманих при функціональній діагностиці.

Раптовий спалах та поширення вірусу COVID-19 у 2020 послужили поштовхом для вирішення низки біомедичних завдань, у тому числі проблеми розпізнавання знімків функціональної діагностики. Оскільки задача діагностування хворих на COVID-19 за допомогою нейронних мереж є досі новою, було вирішено дослідити існуючі методи вирішення задачі розпізнавання зображень рентгенівських знімків [1]. Після чого виділити однин із методів та вдосконалити його

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність роботи

Інтеграція медичних інформаційних технологій (ІТ) у первинну медико-санітарну допомогу включає різноманітні електронні методи, які використовуються для управління інформацією про здоров’я людей та медичну допомогу як для окремих пацієнтів, так і для груп пацієнтів. Застосування медичних ІТ може покращити якість медичної допомоги, навіть робить медичну допомогу більш економічно ефективною.

Серед важливих медичних технологій, що знайшли місце застування слід зазначити: моніторинг здоров'я, блокчейн, імунотерапія раку, 3D-друк, роботизована хірургія [2].

Штучний інтелект змінює охорону здоров’я, і його використання стає реальністю в багатьох медичних галузях і спеціальностях. ШІ, машинне навчання (ML), обробка природної мови (NLP) і глибоке навчання (DL) дають змогу зацікавленим сторонам у сфері охорони здоров’я та медичним працівникам швидше з більшою точністю визначати потреби та рішення в галузі охорони здоров’я, використовуючи шаблони даних для швидкого прийняття обґрунтованих медичних або бізнес-рішень.

Штучний інтелект здатний аналізувати великі обсяги даних, які зберігаються медичними організаціями, у вигляді зображень, клінічних досліджень та медичних заяв, а також може виявляти закономірності та ідеї, які часто не можна виявити ручними наборами навичок людини. Алгоритми штучний «навчають» ідентифікувати та позначати шаблони даних, тоді як NLP

дозволяє цим алгоритмам ізолювати відповідні дані. DL аналізує дані та інтерпретуються комп’ютерами за допомогою розширених знань.

Використання штучного інтелекту підтримує багато зацікавлених сторін у сфері охорони здоров’я. Наприклад, команди лікарів, дослідників або менеджерів даних, які беруть участь у клінічних випробуваннях, можуть прискорити процес пошуку та підтвердження медичного кодування, що має вирішальне значення для проведення та завершення клінічних досліджень. Також платники медичних послуг можуть персоналізувати свої плани охорони здоров’я, підключивши віртуального агента за допомогою розмовного штучного інтелекту з учасниками, зацікавленими в індивідуальних рішеннях для охорони здоров’я. В свою чергу, лікарі можуть покращити та налаштувати лікування пацієнтам, «перебираючи» медичні дані, щоб швидше передбачити або діагностувати захворювання.

Однією з технологій штучного інтелекту, що набула значного місця в медицині, стала технологія «комп’ютерний зір». Це область штучного інтелекту, яка дозволяє комп'ютерам та системам отримувати значущу інформацію з цифрових зображень, відео та інших візуальних входів - і вживати заходів або давати рекомендації на основі цієї інформації [3].

Комп'ютерний зір охоплює всі завдання, що виконуються системами біологічного зору, включаючи "бачення" або відчуття зорового стимулу, розуміння того, що бачиться, і можливість вилучення складної інформації у представлення, яке можна використовувати в інших процесах. Міждисциплінарне поле моделює та автоматизує елементи систем зору людини за допомогою датчиків, комп’ютерів та алгоритмів машинного навчання. Комп'ютерний зір - це теорія, що лежить в основі здатності систем штучного інтелекту бачити і розуміти навколишнє їх оточення.

Системи комп'ютерного зору здійснюють виявлення об'єктів так само, як це роблять люди, починаючи з найнижчих рівнів обробки, працюють на вищих рівнях, поєднуючи функції разом. Виконуючи цифрову обробку для розпізнавання об'єктів, комп'ютер виконує послідовно комплексні процеси, аналогічні тому, як люди розпізнають об'єкти. Тобто існують наступні етапи виявлення об'єктів: розпізнавання образів, видобуток функції, класифікація.

Під час розпізнавання шаблонів мережа аналізує все зображення і знаходить образи візерунків на зображенні. Після розпізнавання шаблону мережа здійснює вилучення функцій. Мережа приймає знайдені шаблони і розбиває ці шаблони на чіткі риси, вибираючи шаблони, які вона вважає важливими, ігноруючи інші частини зображення. Наступним етапом буде класифікація. Припустимо, що об'єктом інтересу в зображенні є автомобіль. Під час класифікації відповідні ознаки об’єднуються в уявлення об’єкта. Потім це представлення порівнюється з тим, що мережа знає про об’єкти, і різні кластери форм і країв використовуються для розміщення мітки на об'єкті [4].

Комп'ютерний зір зосереджений на розумінні зображення та відео. Він передбачає такі завдання, як виявлення об’єктів, класифікація зображення та сегментація. Медичні візуалізації можуть мати значну користь від останніх досягнень у класифікації зображень та виявленні об’єктів. Кілька досліджень продемонстрували багатообіцяючі результати у складних задачах медичної діагностики, що охоплюють дерматологію, радіологію або патологію. Системи поглибленого навчання можуть допомогти лікарям, запропонувавши другу думку та позначивши її щодо областей на зображеннях.

Комп'ютерний зір використовується в різних додатках охорони здоров'я, щоб допомогти медичним працівникам у прийнятті кращих рішень щодо лікування пацієнтів. Медична візуалізація або аналіз медичних зображень - це один із таких методів, який створює візуалізацію окремих органів і тканин, що

дозволяє поставити більш точний діагноз. Завдяки аналізу медичних зображень лікарям та хірургам стає легше побачити внутрішні органи пацієнта, щоб виявити будь-які проблеми чи аномалії.

Через непередбачуваність та агресивність вірусу COVID-19 вчені не припиняють пошук нових методів діагностики небезпечного захворювання. І якщо на початку пандемії вважалося, що рентгенографія не може показати зараження коронавірусом, то сьогодні думка фахівців докорінно змінилася [5].

Встановлено, що на знімках видно уражені тканини у 9 пацієнтів із 10. Суть променевих методів, до яких відноситься рентген, полягає у здатності променів змінювати інтенсивність при проходженні через тканини організму з різною щільністю. Відбиваючись на світлочутливій плівці, вони переносять на неї зображення органів. Темні області позначають ущільнення, а світлі порожнини, наповнені повітрям або рідиною. Щоб краще побачити стан дихальних шляхів, рентгенограму роблять у двох та більше проекціях.

Не секрет, що рентген-випромінювання шкідливе для людини, проте в ході 1-2 секундного обстеження він отримує дуже малу дозу. Отже, користь вочевидь перевищує ризик. Дорослим можна робити рентген до кількох разів на тиждень, якщо це необхідно з метою діагностики та моніторингу ефективності лікування пневмонії. Інформативність рентгенографії щодо коронавірусного ураження легень була доведена ще на початку пандемії китайськими вченими. Вона дає змогу виявити зміни, характерні для нової інфекції. Це простий та доступний спосіб, за допомогою якого можна обстежити практично всіх пацієнтів, які мають відповідні показання. Рентген-апарати є в кожній поліклініці, серед них є мобільні установки, які зручно використовувати навіть у лікарняній палаті.

Варто зазначити, що інформативним інструментальним дослідженням, яке може визначити коронавірус навіть на ранніх стадіях, є КТ - комп'ютерна

томографія. Це теж променевий метод, але знімок називається томограмою та видається у форматі 3D. Однак КТ – це більш дорогий та складний спосіб. Обстежитись платно не кожен може собі дозволити, а безплатних установок у країні катастрофічно не вистачає, а також у комп'ютерних томографах використовується сильніше випромінювання. Саме тому аналіз рентгенограм на наявність вірусу COVID-19 є актуальним і сьогодні, враховуючи вище зазначені недоліки КТ.

1.2 Огляд існуючих систем

У 2020 році пандемія COVID-19 значною мірою перевернула звичний ритм нашого життя. Зміни відбулись і у технологічному напрямі. Коронавірус змусив уряди різних країн та технологічні компанії діяти набагато швидше. Насамперед це стосується медичної сфери, де кожен втрачений день – це чиєсь життя. Попри те, що всі надії на перемогу над пандемією люди покладають на лікарів та фармацевтичні компанії, до боротьби з коронавірусом долучились також технологічні фірми. Ситуація прискорила розробку алгоритмів штучного інтелекту та розширила їх використання в медицині.

У вересні 2019 року [6] провідна британська лікарня вирішила випробувати алгоритм штучного інтелекту для аналізу рентгенівських променів. Така вимога пояснюється тим, що пацієнтам іноді доводиться чекати до 6 годин, поки лікар не встигне переглянути флюорографії. Тому в лікарні вирішили, що якщо перший висновок на основі зображення дає алгоритм, то лікар лише перевіряє та уточнює діагноз, це в свою чергу значно знизить навантаження на медперсонал, тим самим скоротить час очікування пацієнтів. Провідний рентгенолог із Royal Bolton Hospital проаналізував наявні технічні

рекомендації та вибрав систему аналізу рентгенівського знімка грудної клітки qXR від індійської компанії Qure.ai. По-перше, планується провести піврічний тест системи, під час якого алгоритм дасть альтернативні висновки щодо іміджевої діагностики лікарень-інтернів. Спочатку планували, що якщо вердикт алгоритму постійно збігатиметься з думкою лікаря, то алгоритм згодом зможе замінити лікаря у питанні контролю діагнозів, які поставили інтерни. Врешті-решт, проект був схвалений, але ще до тесту Велика Британія зіткнулася з COVID-19. Було зрозуміло, що нова система може відігравати важливішу роль, ніж планувалося. У разі відсутності ПЛР-тестів або результатів доводиться чекати занадто довго, рентгенівський аналіз став найшвидшим і найекономнішим способом виявлення пацієнтів, які потребують невідкладної допомоги. Тому протягом кількох тижнів Qure.ai налаштував систему виявлення ознак нового вірусу, і лікарня вирішила змінити план тестування, щоб дозволити алгоритму виконувати аналіз необроблених зображень замість того, щоб просто стежити за діагнозом стажера.

Дві китайські компанії розробили програмне забезпечення для діагностики коронавірусу на основі штучного інтелекту. Стартап Infervision, розташований у Пекіні, навчив своє програмне забезпечення виявленню проблем з легкими, використовуючи результати комп'ютерної томографії. Програмне забезпечення, що використовується для діагностики раку легень, може також виявляти пневмонію, пов'язану з респіраторними захворюваннями, такими як коронавірус. Щонайменше 34 китайські лікарні, як повідомляється, використали цю технологію, щоб допомогти їм обстежити 32 000 можливих випадків [7].

Академія Alibaba DAMO, дослідницький підрозділ китайської компанії Alibaba, також підготувала систему штучного інтелекту для розпізнавання

коронавірусів з точністю до 96%. За даними компанії, система може обробити 300-400 сканів, необхідних для діагностики коронавірусу, за 20-30 секунд, тоді як на ту ж операцію у досвідченого лікаря зазвичай йде 10-15 хвилин. Кажуть, що система допомогла щонайменше 26 китайським лікарням розглянути понад 30 000 випадків [8].

Повідомляється, що в Південній Кореї штучний інтелект допоміг скоротити час, необхідний для розробки тестових наборів на основі генетичного складу вірусу, до кількох тижнів, коли зазвичай займає від двох до трьох місяців. Біотехнологічна компанія Seegene використала свою автоматизовану систему розробки тестів для розробки тестового набору та широко його поширила. Великомасштабне тестування справді має вирішальне значення для подолання заходів ізоляції, і ця політика тестування, схоже, сприяла відносному контролю за пандемією в цій країні, яка оснастила цим пристроєм 118 медичних закладів та протестувала понад 230 000 осіб [9].

1.3 Нейронні мережі

Проведемо огляд важливих типів нейронних мереж, які становлять основу для більшості попередньо навчених моделей глибинного навчання: штучні нейронні мережі (ANN або ШНМ), згорткові нейронні мережі (CNN або ЗНМ), рекурентні нейронні мережі (RNN або РНМ).

Один персептрон (або нейрон) можна уявити як логістичну регресію. Штучна нейронна мережа, або ШНМ, — це група з кількох персептронів/нейронів на кожному шарі. ШНМ також відома як нейронна мережа з прямим зв'язком, оскільки вхідні дані обробляються тільки в прямому напрямку.

Як можна побачити на рисунку 1, ШНМ складається з 3 шарів – вхідного, прихованого та вихідного. Вхідний рівень приймає вхідні дані, прихований рівень обробляє вхідні дані, а вихідний рівень видає результат. ШНМ можна використовувати для вирішення проблем, пов'язаних із таблицями, текстовими даними тощо.

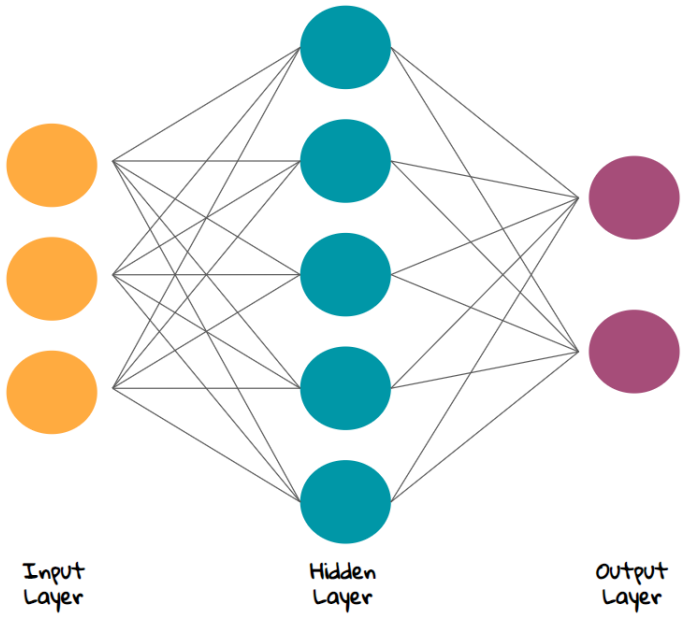


Рисунок 1 – Структура штучної нейронної мережі.

Серед переваг можна відокремити те, що штучна нейронна мережа здатна вивчати будь-яку нелінійну функцію. Отже, ці мережі широко відомі як апроксиматори універсальних функцій. ШНМ мають здатність вивчати вагові показники, які відображають будь-які вхідні та вихідні дані. Однією з головних причин універсальної апроксимації є функція активації. Функції активації вводять у мережу нелінійні властивості. Це допомагає мережі засвоїти будь-які складні відносини між входом і виходом.

Під час розв'язування задачі класифікації зображень за допомогою ШНМ першим кроком є перетворення 2-вимірного зображення в 1-вимірний вектор

перед навчанням моделі, звідки випливають два недоліки. По перше, кількість параметрів, які можна тренувати, різко збільшується зі збільшенням розміру зображення. По друге, ШНМ втрачає просторові особливості зображення. Просторові характеристики відносяться до розташування пікселів на зображенні.

Рекурентна нейронна мережа - це тип вдосконаленої штучної нейронної мережі, що включає спрямовані цикли в пам'яті. Одним з аспектів періодичних нейронних мереж є можливість побудови на більш ранніх типах мереж з вхідними і вихідними векторами фіксованого розміру. Ми можемо використовувати рекурентні нейронні мережі для вирішення проблем, пов’язаних із даними часових рядів, текстовими даними, звуковими даними.

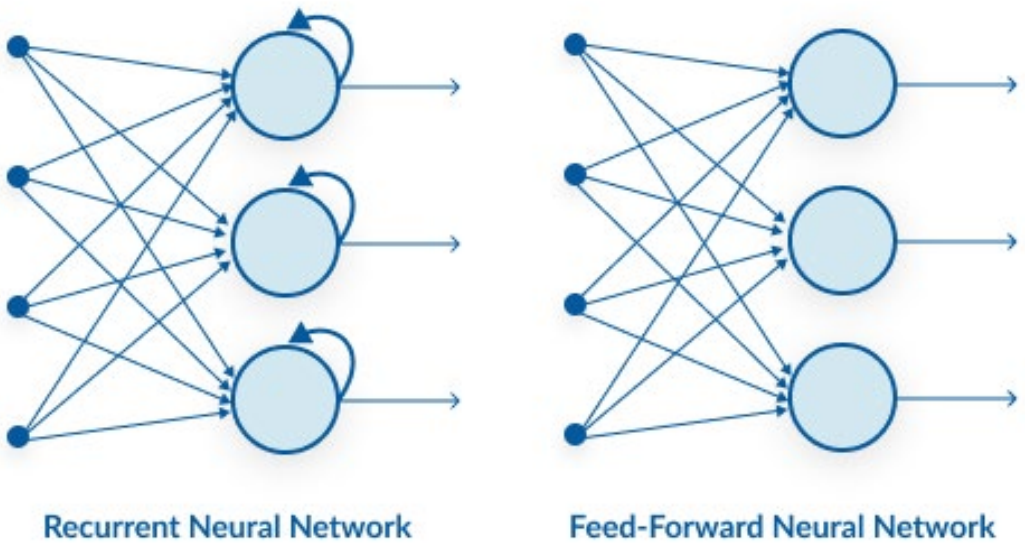


Рисунок 2 – Різниця між рекурентною та штучною нейронними мережами.

Різницю між РНМ і ШНМ з точки зору архітектури зображено на рисунку 2. РНМ має періодичне підключення в прихованому стані. Це обмеження циклу гарантує, що послідовна інформація фіксується у вхідних даних.

Серед переваг відзначають, що РНМ обмінюються параметрами на різних ітераціях, це явище називають спільний доступ до параметрів. Це призводить до зменшення кількості параметрів для навчання та зниження витрат на обчислення. Але глибокі РНМ (РНМ з великою кількістю кроків у часі) також страждають від проблеми зникаючого та вибухаючого градієнта, яка є загальною проблемою в різних типах нейронних мереж.

Згорткова нейронна мережа (CNN) – це клас мереж глибинного навчання, які спеціально створені для роботи із зображень. CNN складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари нейронної мережі складаються зі згорткових шарів, агрегувальних шарів, повноз’єднаних шарів і шарів нормалізації.

На відміну від попередньо зазначених мереж, CNN вивчає фільтри автоматично, не згадуючи про це явно. Ці фільтри допомагають витягти правильні та релевантні функції з вхідних даних. Також CNN фіксує просторові особливості із зображення. Просторові особливості відносяться до розташування пікселів і співвідношення між ними на зображенні. Вони допомагають точно визначити об’єкт, розташування об’єкта, а також його співвідношення з іншими об’єктами на зображенні. З цього випливають переваги згорткової нейронної мережі: вона містить менше параметрів, ніж звичайна повнозв'язна мережа; стійка до переміщення зображення, що потребує розпізнавання. Недоліки згорткової нейронної мережі: тривалий час навчання, необхідність великої кількості даних.

Загалом, CNN, як правило, є більш потужним і точним способом вирішення проблем класифікації. Через здатність CNN розглядати зображення як дані, це найпоширеніший спосіб вирішення проблем комп’ютерного зору та машинного навчання, що залежать від зображення [10].

1.4 Мета роботи

Незважаючи на те, що в даний час широко розвиваються такі методи променевої діагностики, як комп'ютерна томографія, магніторезонансна томографія (МРТ) та позитронно-емісійна томографія (ПЕТ), що мають високу діагностичну інформативність, обстеження цими методами мають певні недоліки: коштовне обладнання та спеціалізоване програмне забезпечення. Все це підвищує вартість обстеження, тому нині дані методи візуалізації використовуються тільки після рентгенографії при підозрі на захворювання, які потребують додаткового обстеження. Актуальність розробки нових методів та моделей аналізу рентгенівських знімків , визначається, насамперед, збільшеними вимогами до якості та надійності розроблюваних систем та медичних пристроїв, створенням перспективних інформаційних технологій з використанням нейронних мереж.

Лікарю-рентгенологу, як правило, доводиться переглядати велику кількість знімків, якість яких не завжди є задовільною, а розмір порівняно малий. Тому якісний аналіз знімка є велике мистецтво і втілює в собі вміння розпізнавати навіть найменші зміни яскравості точок рентгенівських знімків, а також здатність виявляти аномальні структури, особливо за низьку роздільну здатність знімка. Це вимагає високої кваліфікації та доступно небагатьом. Через особливості суб'єктивного зорового сприйняття значно втрачається інформативність рентгенівського знімка. При масових обстеженнях, наприклад, за допомогою флюорографії, ця робота дуже втомлива і може призвести до помилок. Отже, підвищення якості діагностики захворювань легень за допомогою діагностичних інтелектуальних систем аналізу растрових зображень рентгенограм грудної клітини є важливим технічним завданням.

Завданням даної роботи є вивчення архітектури саме згорткової

нейронної мережі, яка дозволить усунути недоліки, описані вище, а також, по можливості, поліпшити точність розпізнавання і швидкість навчання мережі.

1.5 Постановка задачі

У результаті проведеного аналізу актуальних проблем у використанні допоміжних ІТ у сфері охорони здоров'я виникли передумови дослідження можливих методів вирішення задачі розпізнавання на рентгеновському зображенні наявності вірусу COVID-19 за допомогою нейронних мереж та вдосконалення обраного методу.

Основною метою є дослідження та порівняльний аналіз основних моделей глибоких нейронних мереж та алгоритмів розпізнавання образів, які застосовуються в інтелектуальних системах [11]. Згорткові нейронні мережі мають перевагу серед інших в контексті роботи із зображеннями. Мережа поетапно обробляє зображення за допомогою карт згортки і відмінно підходить для класифікації [12].

Для досягнення поставлених цілей необхідно вирішити наступні задачі:

- дослідити існуючі види згорткових нейронних мереж;
- обрати оптимальну модель нейронної мережі для вирішення задачі розпізнавання зображень, враховуючи вимоги до інтелектуальної системи та обмеження;
- дослідити методи підвищення точності моделі розпізнавання;
- здійснити програмну реалізацію оптимізованої моделі навчання.

Інтелектуальний метод аналізу зображень, що розробляється, має бути орієнтованим на інтеграцію у комплексну систему для щоденного використання медичними працівниками, які швидко зможуть отримати

попередній діагноз, оскільки вона допоможе визначити коронавірусну пневмонію, за менший час, що бесперечно має значення в умовах сьогоднішньої пандемії з кількістю захворень. В рамках роботи, потрібно продемонструвати роботу розробленого алгоритму.

2 ЗГОРТКОВІ НЕЙРОННОЇ МЕРЕЖІ

2.1 Складові згорткової нейронної мережі

Відомо, що нейронні мережі хороші у розпізнаванні зображень. Причому хороша точність досягається і звичайними мережами прямого поширення, проте, коли мова заходить про обробку зображень з великим числом пікселів, то параметри для нейронної мережі багаторазово збільшується. Причому настільки, що час, що витрачається на їхнє навчання, стає неймовірно більшим.

Головною особливістю згорткових мереж є те, що вони працюють саме із зображеннями, а тому можна виділити особливості, властиві саме їм. Багатошарові персептрони працюють з векторами, а тому для них немає жодної різниці, чи знаходяться якісь точки поруч або на протилежних кінцях, тому що всі точки рівнозначні і рахуються однаково. Зображення ж мають локальну зв'язковість. Наприклад, якщо йдеться про зображення людських осіб, то цілком логічно очікувати, що точки основних частин обличчя будуть поруч, а не розрізнено розташовуватися на зображенні. Тому потрібно було знайти ефективніші алгоритми для роботи із зображеннями і ними виявилися згорткові мережі.

На відміну від мереж прямого поширення, які працюють із даними у вигляді векторів, згорткові мережі працюють із зображеннями у вигляді тензорів. Тензори – це 3D масиви чисел, або, простіше кажучи, масиви матриць чисел (Рисунок 3).

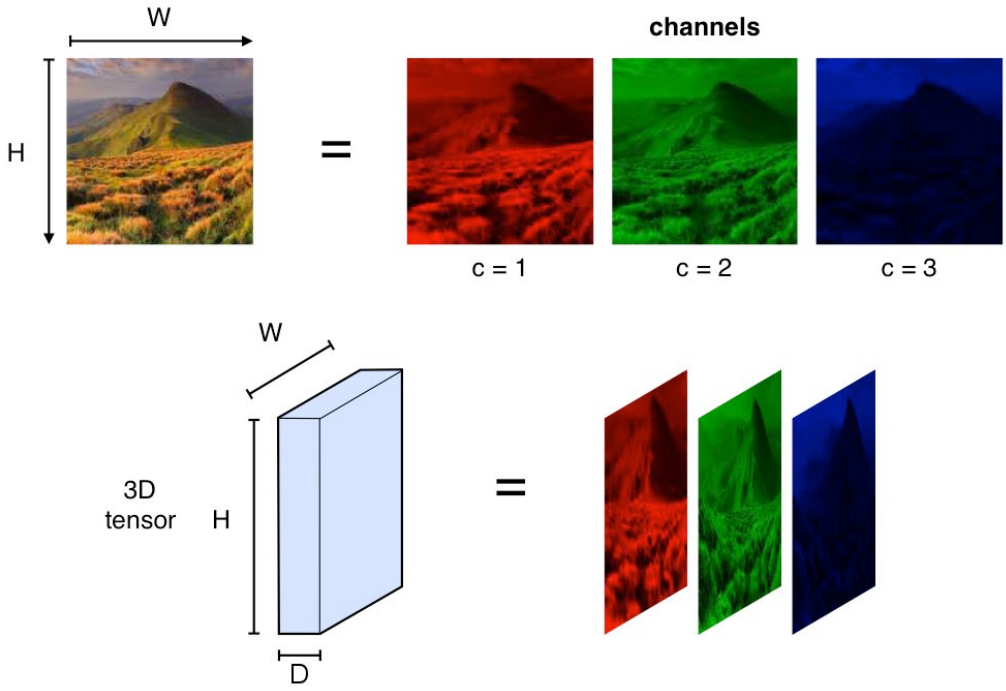


Рисунок 3 – Подання зображення у тензорі.

Зображення в комп'ютері відображаються у вигляді пікселів, а кожен піксель – це значення інтенсивності відповідних каналів. При цьому інтенсивність кожного з каналів описується цілим числом від 0 до 255. Найчастіше використовуються кольорові зображення, які складаються з RGB пікселів – пікселів, що містять яскравості трьох каналів: червоного, зеленого і синього. Різні комбінації цих кольорів дозволяють створити будь-який із кольорів всього спектра. Саме тому цілком логічно використовувати саме тензор для представлення зображень: кожна матриця тензора відповідає за інтенсивність свого каналу, а сукупність всіх матриць описує все зображення.

Згорткові нейронні мережі складаються з базових блоків, завдяки чому їх можна збирати як конструктор, додаючи шар за шаром та отримуючи все потужніші архітектури. Основними блоками згорткових нейронних мереж є згорткові шари, шари підвибірки (пулінгу), шари активації та повнозв'язкові

шари [13]. Розглянемо докладніше кожен із шарів, що формує структуру згорткової мережі.

Шар згортки, як можна здогадатися за назвою типу нейронної мережі, є найголовнішим шаром мережі. Його основне призначення – виділити ознаки на вхідному зображенні та сформувати карту ознак. Карта ознак – це лише черговий тензор (масив матриць), у якому кожен канал відповідає за якусь із виділених ознак.

Для того щоб шар міг виділяти ознаки, в ньому є так звані фільтри (або ядра). Ядра — це в даному випадку набір тензорів. Ці тензори мають зазвичай розмір ядра в межах від 3x3 до 7x7, а їх кількість визначає глибину вихідного 3D масиву. При цьому глибина фільтрів збігається з кількістю каналів вхідного зображення. Так, якщо на вхід згорткового шару подається RGB зображення і потрібна карта ознак, що складається з 32 каналів, то згортковий шар буде містити в собі 32 фільтра глибиною 3, це схематично зображено на рисунку 4.

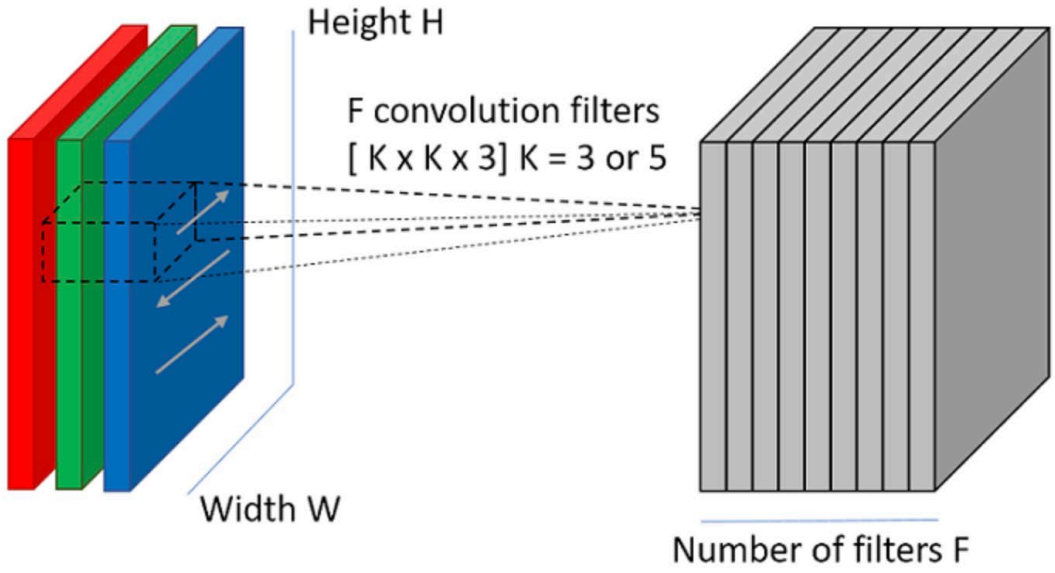


Рисунок 4 – Схематичне зображення згорткового шару.

Для того, щоб сформувати карту ознак із вхідного зображення, здійснюється операція згортки вхідного тензора з кожним із фільтрів. Згортка – це операція обчислення нового значення вибраного пікселя, що враховує значення навколишніх пікселів. Алгоритм отримання результату згортки можна описати так: фільтр накладається на ліву верхню частину зображення та проводиться покомпонентне множення значень фільтра та значень зображення, після чого фільтр переміщається далі за зображенням, доки аналогічним чином не будуть оброблені всі його ділянки (Рисунок 5).

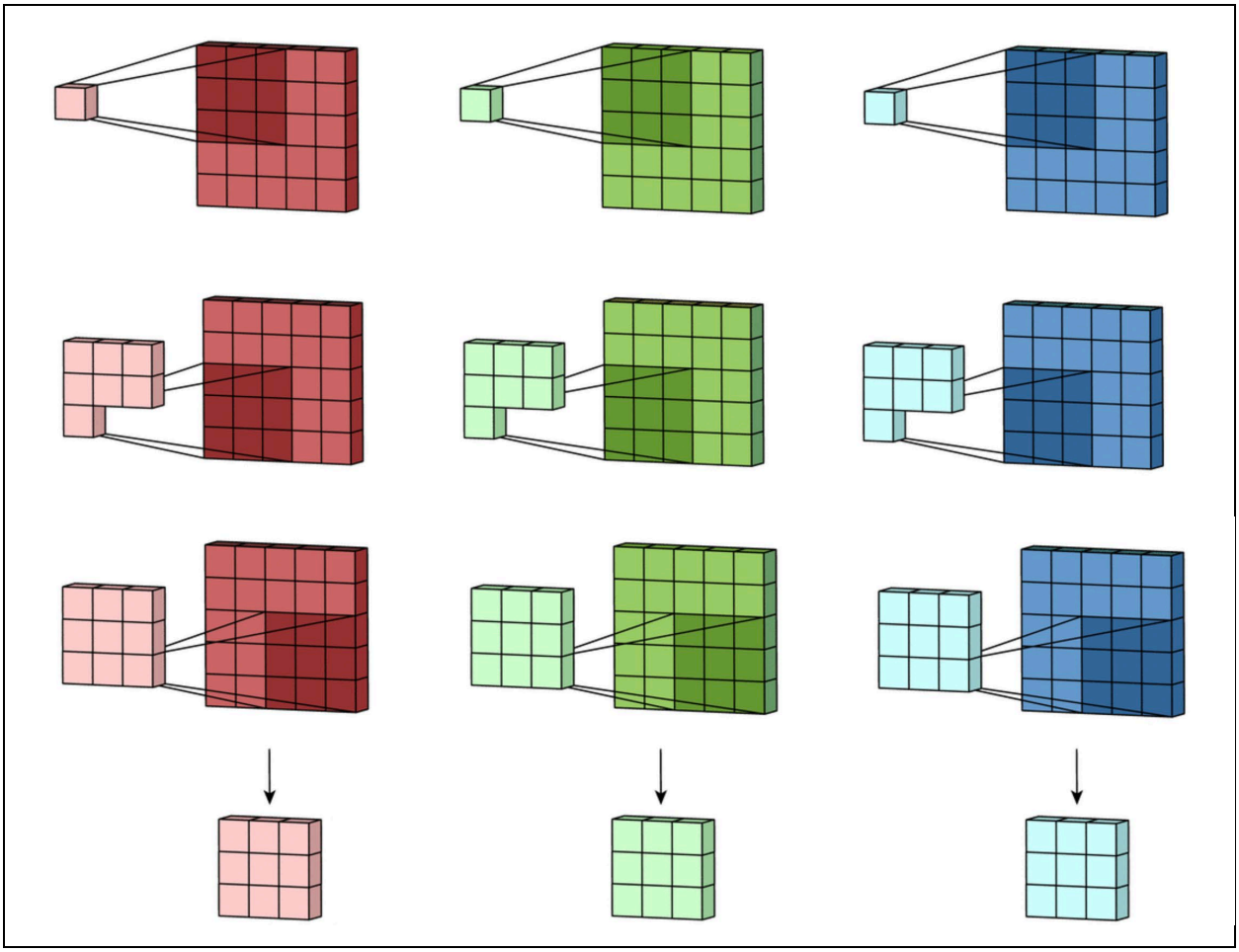


Рисунок 5 – Процес згортки.

Потім числа отриманих матриць підсумовуються в єдину матрицю результату застосування фільтра (Рисунок 6).

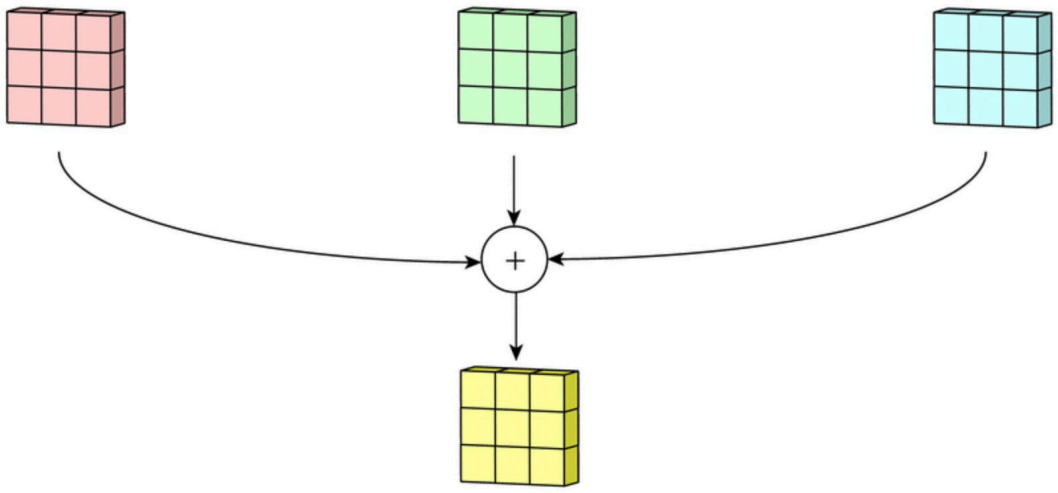


Рисунок 6 – Результат застосування фільтру.

Після цього до кожного значення матриці додається однакове число - значення зміщення фільтра. Отримана матриця становить один канал вихідної карти ознак. Після того, як будуть отримані канали для кожного з фільтрів, матриці об'єднуються в єдиний тензор, завдяки чому на виході знову виходить зображення з іншим числом каналів і, можливо, іншим розміром.

Також обов’язковим елементом згорткової мережі є шар субдискретизації (підвиборки). Цей шар дозволяє зменшити простір ознак, зберігаючи найважливішу інформацію. Існує кілька різних версій шару пулінгу, серед яких максимальний пулінг, середній пулінг та пулінг суми. Найчастіше використовується саме шар макспулінгу. Схематично це можна побачити на рисунку 7. Шару підвиборки потрібен лише один гіперпараметр — крок пулінгу, тобто кількість разів, коли потрібно скоротити просторові розмірності. Найчастіше використовується шар макспулінгу із зменшенням

розміру вхідного тензора вдвічі. Деякі бібліотеки дають змогу задавати окремі параметри зменшення за висотою та шириною, однак найчастіше ці параметри збігаються.

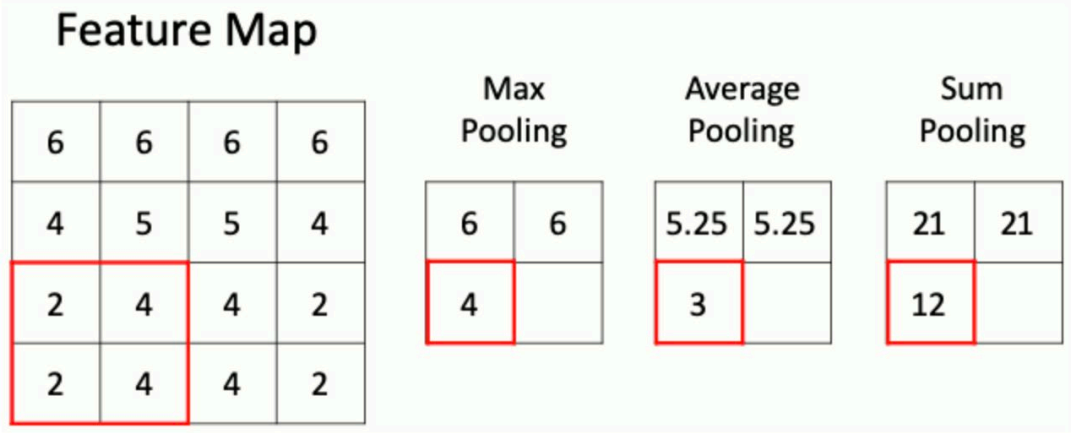


Рисунок 7 – Шар підвиборки.

Ще одним із шарів згорткової мережі є шар активації. Цей шар є деякою функцією, яка застосовується до кожного числа вхідного зображення. Найчастіше використовуються такі функції активації, як ReLU, Sigmoid, Tanh, LeakyReLU (Рисунок 8). Зазвичай активаційний шар ставиться відразу після шару згортки, через що деякі бібліотеки навіть вбудовують функцію ReLU прямо в згортковий шар.

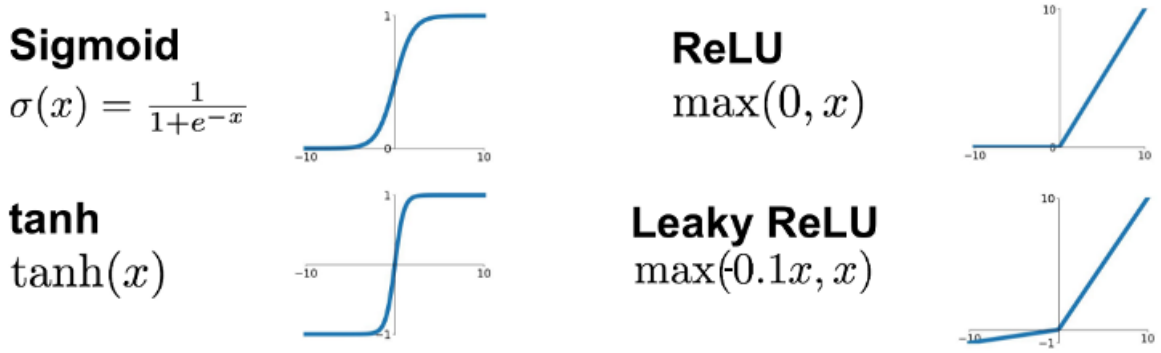


Рисунок 8 – Функції активації.

Останній тип шару, з якого складається згоркова мережа, це повнозв’язний. Даний шар містить матрицю вагових коефіцієнтів і вектор зміщень і нічим не відрізняється від такого ж шару звичайної повнозв'язної мережі (Рисунок 9). Єдиним гіперпараметром шару є кількість вихідних значень. При цьому результатом застосування шару є вектор або тензор, який має матриці в кожному каналі розміром 1x1.

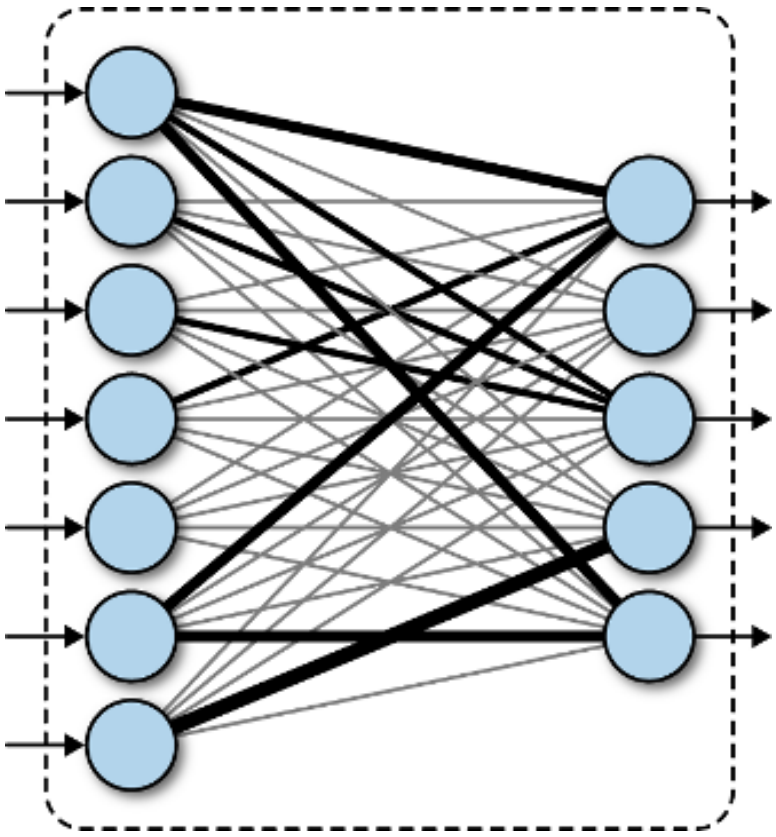


Рисунок 9 – Повнозв’язний шар.

Зі зростанням обчислювальної потужності персональних комп'ютерів, а також появою баз зображень стало можливим навчати глибокі нейронні мережі (Deep Neural Networks).

2.2 Глибинне навчання нейронних мереж

Глибинне навчання — галузь машинного навчання (machine learning), що використовує багатошарові штучні нейронні мережі для розв'язання різних задач - від класифікації до машинного перекладу.

Системи глибинного навчання використовують нейронні мережі, які є обчислювальною системою, змодельованою за моделлю людського мозку. Нейронні мережі мають три різні компоненти: вхідний шар, прихований шар або середній шар і вихідний шар. Вхідний шар – це місце, де обробляються дані, які надсилаються в нейронну мережу, тоді як середні шари/приховані шари складаються зі структури, яка називається вузлом або нейроном. Ці вузли є математичними функціями, які певним чином змінюють вхідну інформацію та передають змінені дані на кінцевий шар або вихідний шар. Прості нейронні мережі можуть розрізняти прості закономірності у вхідних даних, коригуючи припущення або вагові показники щодо того, як точки даних пов’язані одна з одною.

Глибока нейронна мережа отримала свою назву через те, що вона складається з багатьох звичайних нейронних мереж, об’єднаних разом. Чим більше нейронних мереж пов’язано між собою, тим складніші шаблони може розрізнити глибока нейронна мережа і тим більше застосувань вона має. Мережі глибинного навчання відрізняються від більш звичайних одношарових прихованих нейронних мереж своєю глибиною, тобто кількість шарів вузлів, через які дані повинні проходити в багатоетапному процесі розпізнавання образів. Попередні версії нейронних мереж, такі як перші персептрони, були неглибокими, складалися з одного вхідного та вихідного шарів і щонайбільше одного прихованого шару між ними. Більше трьох рівнів (включаючи вхідний

і вихідний) кваліфікується як «глибоке» навчання. Так глибокий — це термін, який означає більше ніж один прихований шар.

У мережах глибинного навчання кожен рівень вузлів тренується на певному наборі функцій на основі результатів попереднього рівня. Чим більше цих рівней, тим складніші функції можуть розпізнавати вузли, оскільки вони об’єднують і знову комбінують функції з попереднього шару. Це відоме як ієрархія функцій, і це ієрархія зростаючої складності та абстракції. Це робить мережі глибинного навчання здатними обробляти дуже великі масиви даних з мільярдами параметрів, які проходять через нелінійні функції . Перш за все, ці нейронні мережі здатні виявляти приховані структури в немаркованих, неструктурованих даних , які є переважною більшістю даних у світі. Іншими словами для неструктурованих даних є необроблені медіа ; тобто зображення, тексти, відео та аудіозаписи. Таким чином, одна з проблем, які глибинне навчання вирішує найкраще, — це обробка та кластеризація сирих, немаркованих середовищ світу, виявлення подібності та аномалій у даних, які жодна людина не впорядковувала в реляційну базу даних і не називала їм [14].

На відміну від більшості традиційних алгоритмів машинного навчання, мережі глибинного навчання виконують автоматичне вилучення функцій без втручання людини. З огляду на те, що виділення функцій — це завдання, на виконання якого командам науковців з даних можуть знадобитися роки, глибинне навчання — це спосіб обійти «слабке місце» для експертів. Це збільшує повноваження невеликих команд науки про дані, які за своєю природою не масштабуються.

Під час навчання на немаркованих даних кожен вузловий шар у глибокій мережі автоматично вивчає особливості, багаторазово намагаючись відновити вхідні дані, з яких він бере свої вибірки, намагаючись мінімізувати різницю

між припущеннями мережі та розподілом ймовірностей самих вхідних даних. Обмежені машини Больцмана, наприклад, створюють так звані реконструкції таким чином.

У процесі ці нейронні мережі вчаться розпізнавати кореляції між певними релевантними характеристиками та оптимальними результатами – вони створюють зв'язки між сигналами ознак і тим, що ці функції представляють, чи то повна реконструкція, чи з позначеними даними [15].

Мережу глибинного навчання, навчену на позначених даних, можна потім застосувати до неструктурованих даних, надаючи їй доступ до набагато більшого вхідного матеріалу, ніж мережі машинного навчання. Це рецепт вищої продуктивності: чим більше даних може тренуватися мережа, тим точнішою вона буде. Погані алгоритми, навчені на великій кількості даних, можуть перевершити хороші алгоритми, навчені на дуже малій кількості. Здатність глибинного навчання обробляти та вчитися на величезній кількості немаркованих даних дає йому явну перевагу перед попередніми алгоритмами.

В задачі розпізнавання зображень найчастіше застосовуються саме глибокі згорткові нейронні мережі [16]. У зв'язку з чим, в рамках роботи було проведено огляд існуючих різновидів CNN.

2.2.1 Класичні архітектури CNN

Найбільш відомі класичні архітектури мереж: LeNet-5, AlexNet, VGG 16.

Модель LeNet-5 Яна Лекуна була розроблена в 1998 році для ідентифікації рукописних цифр для розпізнавання поштових індексів у поштовій службі. Ця новаторська модель значною мірою представила згорткову нейронну мережу, якою ми її знаємо сьогодні [17].

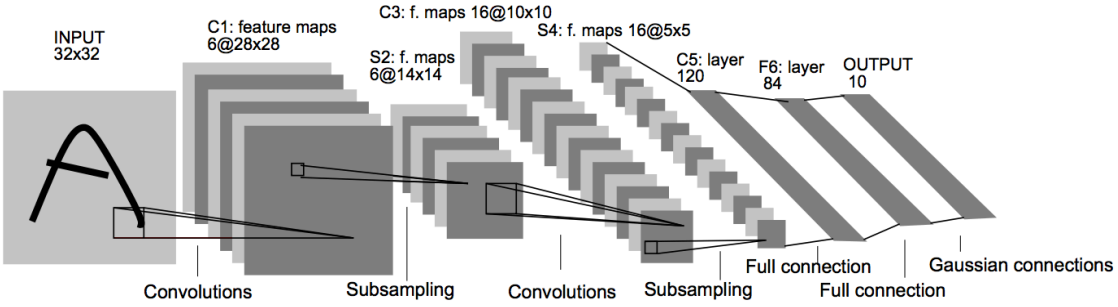


Рисунок 10 – Архітектура LeNet-5.

Згорткові шари використовують підмножину каналів попереднього шару для кожного фільтра, щоб зменшити обчислення та примусово порушити симетрію в мережі. Шари підвибірки використовують форму середнього об’єднання.

Архітектура (Рисунок 10) була розроблена для ідентифікації рукописних цифр у наборі даних MNIST. Архітектура досить проста і зрозуміла. Вхідні зображення мали відтінки сірого та мали розмір 32*32*1, за якими слідували дві пари шару Convolution з кроком 2 і шаром Average pooling з кроком 1. Нарешті, повністю з'єднані шари з активацією Softmax у вихідному шарі. Традиційно ця мережа мала 60000 параметрів.

AlexNet розробили Алекс Крижевський та інші у 2012 році для участі в конкурсі ImageNet. Загальна архітектура дуже схожа на LeNet-5, хоча ця модель значно більша. Успіх цієї моделі (яка зайняла перше місце на конкурсі ImageNet 2012 року) переконав велику частину спільноти комп’ютерного зору серйозно поглянути на глибинне навчання для завдань комп’ютерного зору (Рисунок 11).

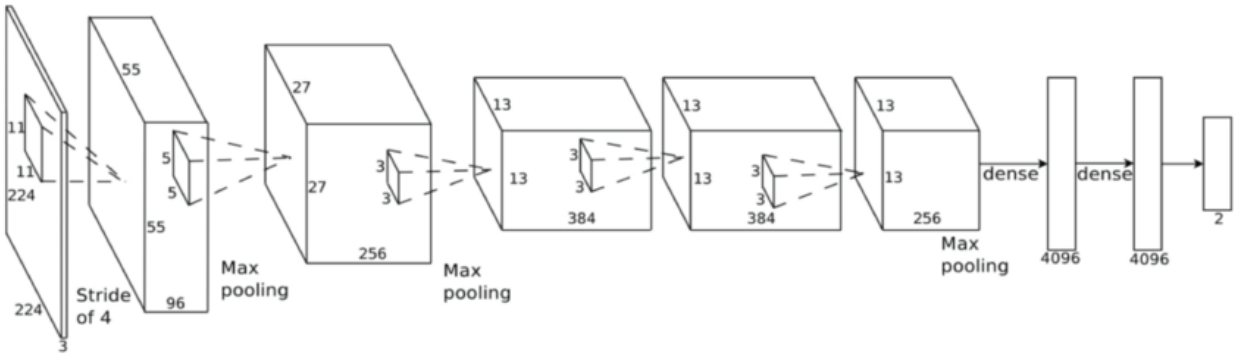


Рисунок 11 – Архітектура AlexNet.

Ця мережа була дуже схожа на LeNet-5, але була глибшою, з 8 шарами, з більшою кількістю фільтрів, складеними згортковими шарами, максимальним об’єднанням у пул, випаданням, розширенням даних, ReLU та SGD. AlexNet став переможцем конкурсу ImageNet ILSVRC-2012. Він навчався на двох графічних процесорах Nvidia Geforce GTX 580, тому мережа була розділена на два конвеєри. AlexNet має 5 шарів згортки і 3 повністю підключених шари. AlexNet складається з приблизно 60M параметрів. Основним недоліком цієї мережі було те, що вона містить занадто багато гіперпараметрів.

Мережа VGG, представлена в 2014 році, пропонує більш глибокий, але простіший варіант згорткових структур, розглянутих вище. На момент свого впровадження ця модель вважалася дуже глибокою. Основний недолік занадто великої кількості гіперпараметрів AlexNet був вирішений VGG Net, замінивши великі фільтри розміром з ядро (11 і 5 у першому і другому шарі згортки відповідно) на кілька фільтрів розміру ядра 3×3 один за одним.

Архітектура, розроблена Сімоньяном і Зісерманом, (Рисунок 12) посіла 1 місце в конкурсі Visual Recognition Challenge 2014 року.

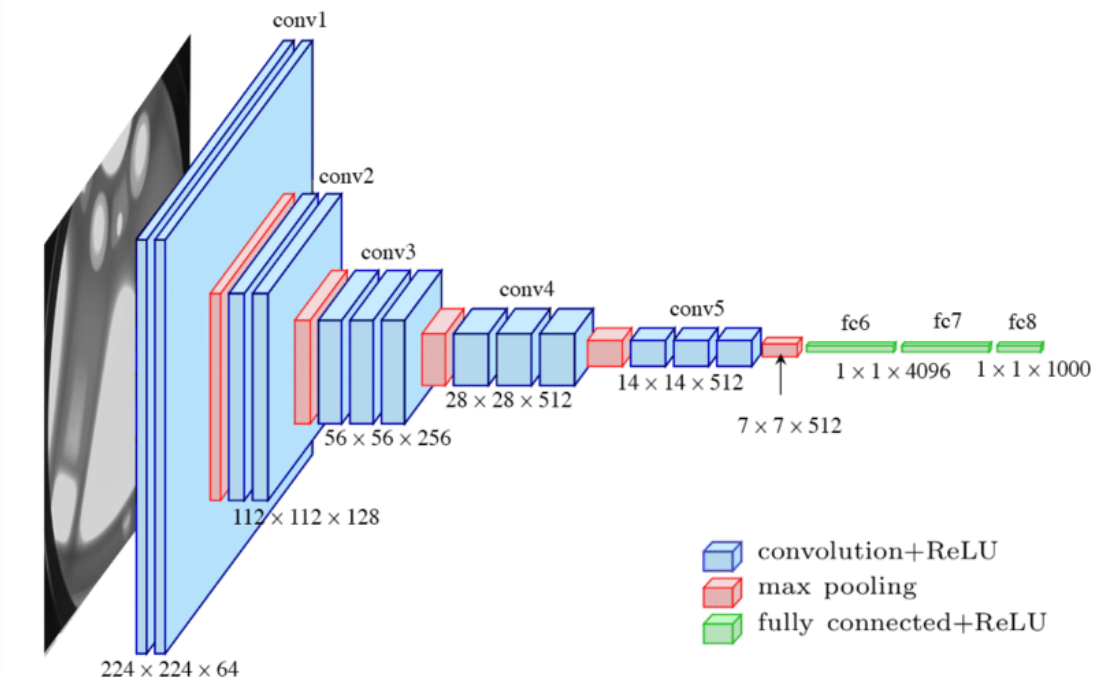


Рисунок 12 – Архітектура VGG.

Архітектура складається з 3*3 згорткових фільтрів, 2*2 шару Max Pooling з кроком 1, зберігаючи заповнення, щоб зберегти розмірність. Всього в мережі є 16 шарів, де вхідне зображення має формат RGB з розмірністю 224*224*3, за яким слідують 5 пар Convolution (фільтри: 64, 128, 256,512,512) і Max Pooling. Вихід цих шарів подається на три повністю з'єднаних шари і функцію softmax у вихідному шарі. Всього в VGG Net є 138 мільйонів параметрів [18].

2.2.2 Сучасні архітектури CNN

Серед сучасних архітектур відокремимо Inception, ResNet, DenseNet, EfficientNet.

Глибокі залишкові мережі (ResNet) були проривною ідеєю, яка дозволила розробити набагато глибші мережі, сотні шарів на відміну від десятків шарів.

Загальновизнаним принципом є те, що більш глибокі мережі здатні вивчати складніші функції та уявлення вхідних даних, що має призвести до кращої продуктивності. ResNet подібні до мереж VGG, але при послідовному підході вони також використовують «пропуск з'єднань» і «пакетну нормалізацію», що допомагає тренувати глибокі шари, не заважаючи продуктивності. Після VGG Nets, оскільки CNN заглибилися, їх стало важко навчати через проблему зникаючих градієнтів, що робить похідну нескінченно малою. Тому загальна продуктивність насичується або навіть погіршується.

Хоча оригінальний ResNet був зосереджений на створенні архітектури мережі для забезпечення більш глибоких структур шляхом пом'якшення проблеми деградації, інші дослідники відтоді вказали, що збільшення ширини мережі (глибини каналу) може бути більш ефективним способом розширення загальної ємності мережі. Широкі залишкові мережі мають архітектуру, схематично зображену на рисунку 13.

Рисунок 13 – Архітектура широких залишкових мережі.

Кожен кольоровий блок шарів представляє серію звивин одного розміру. Відображення ознак періодично зменшується шляхом стрибкоподібної згортки, що супроводжується збільшенням глибини каналу для збереження часової складності на шар. Пунктирні лінії позначають залишкові зв’язки, в яких ми проектуємо вхід через згортку 1x1, щоб відповідати розмірам нового блоку.

Архітектура ResNeXt є розширенням глибокої залишкової мережі, яка замінює стандартний залишковий блок блоком, який використовує стратегію «розщеплення-перетворення-злиття» (тобто розгалужені шляхи всередині комірки), що використовується в початкових моделях. Просто, замість того, щоб виконувати згортки над повною картою вхідних ознак, вхідні дані блоку проектуються в серію нижчих (канальних) розмірних представлень, до яких ми окремо застосовуємо кілька згорткових фільтрів перед об’єднанням результатів. Архітектура ResNet (Рисунок 14) просто імітує моделі ResNet, замінюючи блоки ResNet на блок ResNeXt.

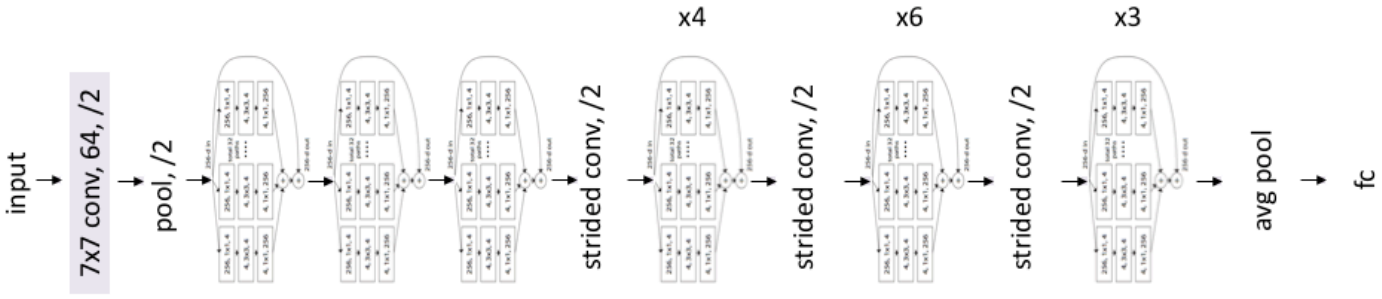


Рисунок 14 – Архітектура ResNeXt.

Ідея щільних згорткових мереж (DenseNet) проста: посилатися на карти об’єктів з попередньої мережі. Таким чином, карта об’єктів кожного шару

з’єднується з входом кожного наступного шару в межах щільного блоку. Це дозволяє пізнішим рівням мережі безпосередньо використовувати функції попередніх шарів, заохочуючи повторне використання функцій у мережі. Автори стверджують, що «об’єднання карт об’єктів, засвоєних різними шарами, збільшує різницю у введенні наступних шарів і покращує ефективність». Однак, оскільки мережа здатна безпосередньо використовувати будь-яку попередню карту ознак, автори виявили, що вони можуть працювати з дуже малою глибиною вихідного каналу (тобто 12 фільтрів на шар), що значно зменшує загальну кількість необхідних параметрів. Загальна архітектура зображена на рисунку 15.

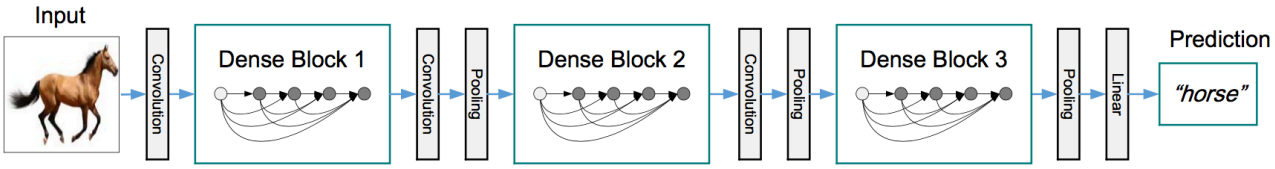


Рисунок 15 – Архітектура DenseNet.

У 2014 році дослідники з Google представили мережу Inception, яка зайняла перше місце в конкурсі ImageNet 2014 року з питань класифікації та виявлення. Модель складається з основного блоку, який називається «початковою коміркою», в якому ми виконуємо серію згорток у різних масштабах і згодом об’єднуємо результати. Щоб заощадити обчислення, для зменшення глибини вхідного каналу використовуються згортки 1x1. Для кожної клітинки ми вивчаємо набір фільтрів 1x1, 3x3 і 5x5, які можуть навчитися вилучати об’єкти в різних масштабах з вхідних даних.

Основна ідея модулів полягає в тому, що замість того, щоб реалізовувати згорткові шари різних гіперпараметрів у різних шарах, виконуються всі

згортки разом, щоб вивести результат, що містить матриці з усіх операцій фільтра разом.

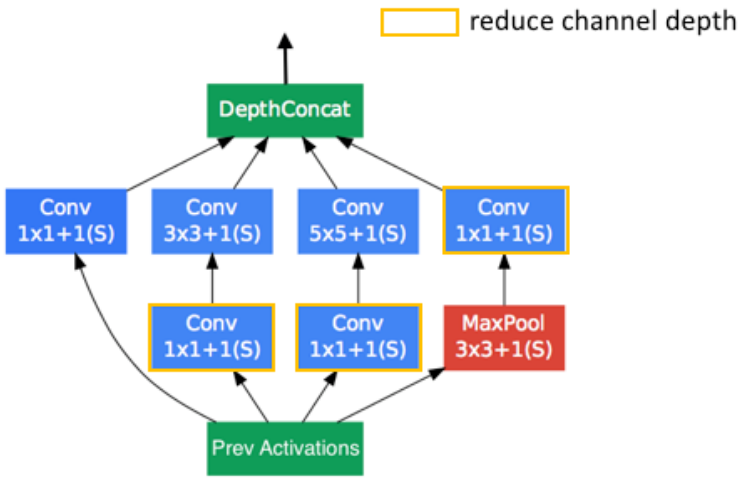


Рисунок 16 – Початковий модуль мережі з архітектурою Inception.

На рисунку 16 схематичне зображення простого початкового модуля з різними згортковими шарами, реалізованими разом.

Автори EfficientNet запропонували масштабувати моделі CNN, щоб отримати кращу точність і ефективність набагато більш морально.

EfficientNet використовує техніку, яка називається складеним коефіцієнтом, для простого, але ефективного масштабування моделей. Замість випадкового збільшення ширини, глибини або роздільної здатності, складне масштабування рівномірно масштабує кожен вимір за допомогою певного фіксованого набору коефіцієнтів масштабування. Використовуючи метод масштабування та AutoML, автори цієї роботи розробили сім моделей різної розмірності, які перевершили сучасну точність більшості згорткових нейронних мереж і з набагато кращою ефективністю [19].

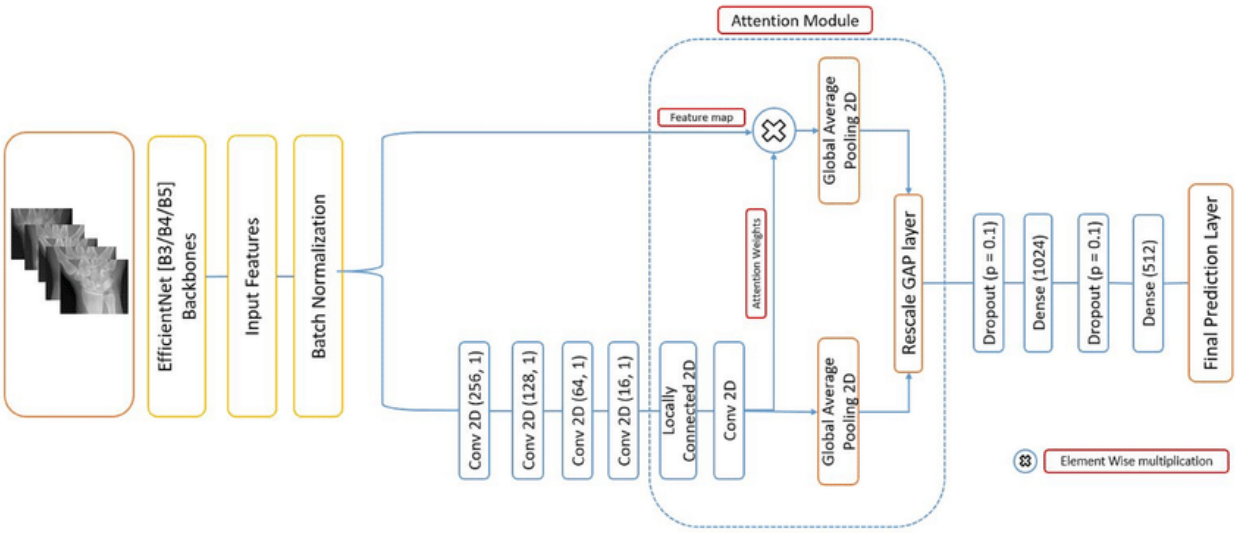


Рисунок 17 – Архітектура EfficientNet.

EfficientNet базується на базовій мережі, розробленій пошуком нейронної архітектури з використанням платформи AutoML MNAS. Мережа точно налаштована для отримання максимальної точності, але також штрафується, якщо мережа дуже важка з точки зору обчислень. Він також штрафується за повільний час висновку, коли мережа займає багато часу для прогнозування. Архітектура (Рисунок 17) використовує мобільну перевернуту згортку вузького місця, подібну до MobileNet V2, але набагато більшу через збільшення FLOPS(операцій з плаваючою комою в секунду). Ця базова модель масштабується, щоб отримати сімейство EfficientNets.

На рисунку 18 показано графік продуктивності EfficientNet порівняно з іншими мережевими архітектурами. Найбільша модель EfficientNet EfficientNet B7 отримала найсучаснішу продуктивність у наборах даних ImageNet і CIFAR-100, приблизно 84,4% [20].

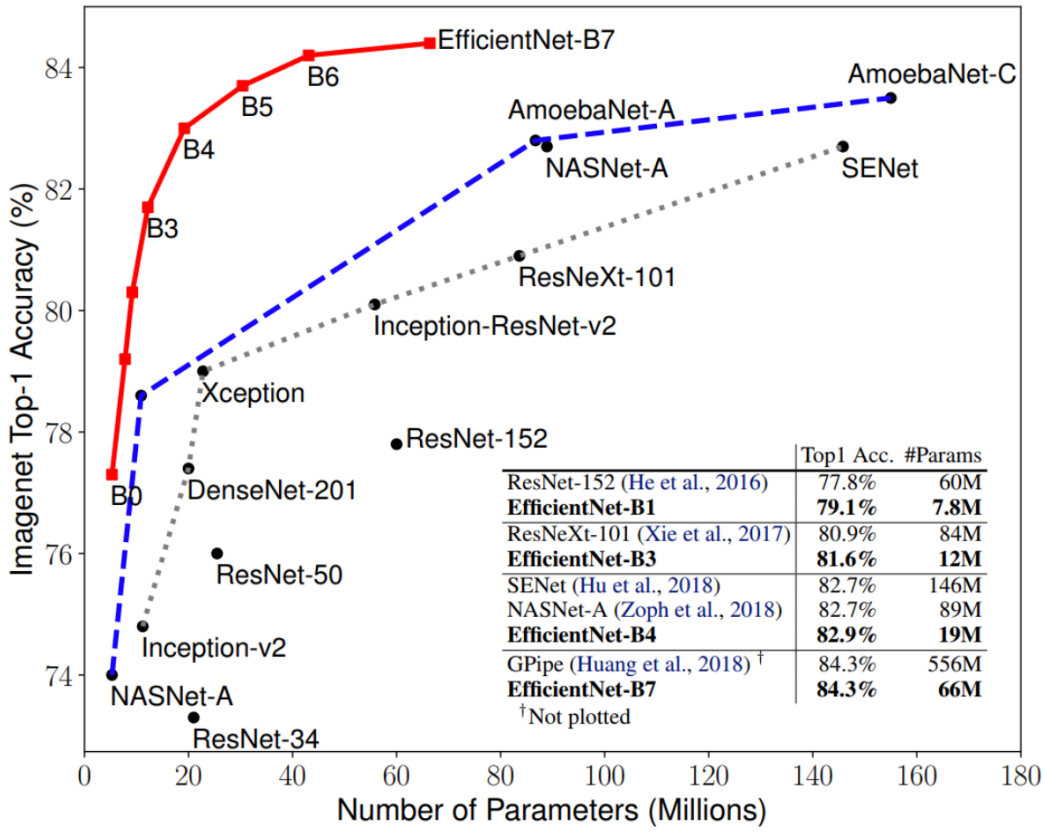


Рисунок 18 – Продуктивність EfficientNet для набору даних ImageNet в порівнянні з іншими архітектурами мереж.

Оскільки ця архітектура має гарні показники ефективності, дослідивши її детальніше можна поліпшити точність розпізнавання в ході розробки моделі навчання. Для виконання поставленої задачі було обрано Efficientnet b3. Це найбільша модель у сімействі Efficientnet, яка задовільняє апаратні обмеження, параметр пам’яті графічного процесору, що використовується під час проектування та реалізації системи.

3 МОДЕЛЬ ГЛИБИННОГО НАВЧАННЯ

3.1 Етапи навчання для розпізнавання зображень

Для розробки методу, що буде аналізувати рентгенівські зображення, слід дотримуватися загальних рекомендацій машинного навчання щодо етапів роботи із моделлю для навчання. Далі наведено загальний план навчання для розпізнавання об’єктів:

1) Збір даних

Як відомо, машини спочатку навчаються на даних, які їм надаються. Надзвичайно важливо збирати надійні дані, щоб модель машинного навчання могла знайти правильні закономірності. Якість даних, які подаються на машину, визначатиме, наскільки точною є модель. Якщо є неправильні або застарілі дані, це може призвести до неправильних результатів або прогнозів, які не мають значення. Необхідно переконатись, що використовуються дані з надійного джерела, оскільки це безпосередньо вплине на результат моделі. Хороші дані є релевантними, містять дуже мало відсутніх і повторюваних значень і добре представляють різні наявні підкатегорії/класи.

2) Підготовка даних

Для того щоб отримати дані, потрібно їх підготувати. Зробити це можна за допомогою деяких дій. Наприклад, об’єднання всіх наявних даних і рандомізації їх. Це допомагає переконатися, що дані розподіляються рівномірно, а впорядкування не впливає на процес навчання. Також однією з операцій підготовки даних є очищення для видалення небажаних даних, відсутніх значень, рядків і стовпців, повторюваних значень, перетворення типів даних тощо. Можливо, навіть доведеться реструктурувати набір даних і

змінити рядки і стовпці або індекс рядків і стовпців. Також можна спробувати візуалізацію даних, щоб зрозуміти, як вони структуровані, і зрозуміти зв’язок між різними змінними та наявними класами. Після чого, необхідно зробити розбиття очищених даних на два набори - навчальний набір і набір для тестування. Навчальний набір – це набір, з якого навчається наша модель. Набір для тестування використовується для перевірки точності нашої моделі після навчання.

3) Вибір моделі

Модель машинного навчання визначає результат, який отримується після запуску алгоритму машинного навчання на зібраних даних. Важливо вибрати модель, яка відповідає поставленій задачі. Для цього слід провести аналіз архітектури нейронної мережі безпосередньо для класу задач, де буде використовуватись розроблена модель. Крім цього, також потрібно перевірити, чи підходить модель для числових чи категорійних даних, і вибрати відповідний вибір.

4) Навчання моделі

Тренування є найважливішим кроком у машинному навчанні**.** Під час навчання передаються підготовлені дані моделі машинного навчання, щоб знаходити закономірності та робити прогнози. З часом, з навчанням, модель стає краще передбачати.

5) Оцінка моделі

Після навчання моделі необхідно перевірити, як вона працює. Це робиться шляхом тестування продуктивності моделі на раніше небачених даних. Використовуються невидимі дані — це тестовий набір, на який раніше розділили дані. Якщо тестування проводити на тих самих даних, які використовуються для навчання, не зможемо отримати точного вимірювання, оскільки модель вже звикла до даних і знаходить в них ті самі закономірності,

що й раніше. Це дасть непропорційно високу точність. Використовуючи дані тестування, можна отримати точне вимірювання ефективності моделі та її швидкості.

6) Налаштування параметрів

Після того як буде проведено оцінку моделі, треба перевірити, чи можна якось підвищити її точність. Це робиться шляхом налаштування параметрів, наявних у моделі. Параметри — це змінні в моделі, які зазвичай вирішує програміст. При певному значенні параметра точність буде максимальною. Налаштування параметрів відноситься до пошуку цих значень.

7) Складання передбачень

Зрештою, можливо використовувати модель на невидимих даних, щоб робити точні прогнози.

3.2 Методи покращення точності моделі навчання

Продуктивність є ключовим фактором, коли мова йде про моделі глибинного навчання, і це стає важким завданням, коли є обмеження в апаратних ресурсах. Одним із важливих параметрів для вимірювання продуктивності є точність. В цьому розділі буде розглянуто методи для покращення точності, пов'язані із даними, налаштуваннями алгоритму та оптимізацією моделі [21].

3.2.1 Гіперпараметри навчальної моделі

Оптимізація моделей - одне з найскладніших завдань у впровадженні рішень машинного навчання. Як правило, ми розглядаємо оптимізацію моделі

як процес регулярної зміни коду моделі з метою мінімізації помилки тестування. Однак глибока оптимізація навчання часто тягне за собою елементи тонкого налаштування, які існують поза моделлю, але можуть сильно впливати на її поведінку. Глибинне навчання часто називає ці приховані елементи гіперпараметрами, оскільки вони є одним із найважливіших компонентів будь-якої програми машинного навчання.

Гіперпараметри – це налаштування, які можна налаштувати для керування поведінкою алгоритму машинного навчання. Критерії визначення гіперпараметру неймовірно абстрактні та гнучкі. Звичайно, існують добре встановлені гіперпараметри, такі як кількість прихованих одиниць або швидкість навчання моделі, але існує довільна кількість налаштувань, які можуть відігравати роль гіперпараметрів для конкретних моделей. Загалом гіперпараметри дуже специфічні для типу моделі машинного навчання, яка потребує оптимізації. Іноді параметр моделюється як гіперпараметр, тому що його не можна вилучити з навчального набору. Класичним прикладом є налаштування, що контролюють ємність моделі (спектр функцій, які модель може представляти). Якщо алгоритм глибинного навчання вивчає ці параметри безпосередньо з навчального набору, він, ймовірно, спробує оптимізувати цей набір даних, що призведе до перенавчання моделі (погане узагальнення).

Кількість та різноманітність гіперпараметрів в алгоритмах машинного навчання дуже специфічна для кожної моделі. Тим не менш, є деякі класичні гіперпараметри, на які ми завжди повинні дивитися і які повинні допомогти вам подумати про цей аспект машинного навчання: швидкість навчання, кількість прихованих юнітів, ширина ядра згортки. Вважається, що швидкість навчання є основою всіх гіперпараметрів, вона кількісно оцінює прогрес навчання моделі в такий спосіб, який можна використовуватиме оптимізації її

можливостей. Кількість прихованих юнітів – це класичний гіперпараметр в алгоритмах глибинного навчання, кількість прихованих одиниць є ключовим для регулювання репрезентативної здатності моделі. У згорткових нейронних мережах (CNN) ширина ядра впливає на кількість параметрів в моделі, що, своєю чергою, впливає на її пропускну спроможність.

Процес вибору гіперпараметрів є ключовим аспектом рішення для глибинного навчання. Більшість алгоритмів глибинного навчання визначають конкретні гіперпараметри, які керують різними аспектами, такими як пам'ять або вартість виконання. Однак додаткові гіперпараметри можуть бути визначені для адаптації алгоритму до конкретного сценарію. Фахівці з науки про дані зазвичай витрачають чимало часу на налаштування гіперпараметрів для досягнення найкращої продуктивності для конкретної моделі.

Коли справа доходить до вибору та оптимізації гіперпараметрів, є два основні підходи: ручний та автоматичний вибір. Обидва підходи технічно життєздатні, і рішення, як правило, є компромісом між глибоким розумінням моделі машинного навчання, необхідної для вибору гіперпараметрів вручну, і високою обчислювальною вартістю, необхідною алгоритмам автоматичного вибору.

Основна мета ручного вибору гіперпараметра – налаштувати ефективну ємність моделі відповідно до складності цільової задачі. Іншими словами, мета навчального процесу полягає в тому, щоб допомогти вам максимізувати вашу ефективну здатність виконувати поставлене завдання. Оскільки алгоритми, цілі, типи даних та набори даних значно змінюються від проекту до проекту, не існує оптимального вибору гіперпараметрів, що підходять для всіх моделей та всіх завдань. Натомість гіперпараметри мають бути оптимізовані в контексті кожного проекту машинного навчання.

Моделі машинного навчання мають поняття ефективної здібності. У цьому контексті ефективна потужність алгоритму машинного навчання визначається трьома основними факторами:

1) Репрезентативна здатність алгоритму чи набору гіпотез, які задовольняють навчальному набору даних.

2) Ефективність алгоритму мінімізації його функції вартості.

3) Ступінь, у якій функція витрат та процес навчання зводять до мінімуму помилку тесту.

Налаштування різних гіперпараметрів є ключем до пошуку оптимального балансу між цими факторами.

Оптимізація гіперпараметра вручну може бути стомлюючим заняттям. Для вирішення цієї проблеми використовують алгоритми, які автоматично визначають потенційний набір гіперпараметрів та намагаються оптимізувати їх, приховуючи цю складність від розробника. Алгоритми, такі як Grid Search і Random Search, починають бути корисними, коли йдеться про виведення та оптимізацію гіперпараметрів.

3.2.2 Трансферне навчання

Одним із найпоширеніших засобів оптимізації часу розробки є використання для глибинного навчання потужних архітектур. Для цього використовують технології трансферного навчання (transfer learning) та попередньо навчені моделі.

Ідея трансферного навчання полягає в тому, щоб взяти модель, навчену одному завданню, і застосувати до другого, подібного завдання. Той факт, що в моделі вже були навчені деякі або всі вагові показники для другого завдання, означає, що модель можна реалізувати набагато швидше. Це дозволяє швидко

оцінити продуктивність і налаштувати модель, що дає змогу пришвидшити розгортання в цілому. Трансферне навчання стає все більш популярним у сфері глибинного навчання завдяки величезній кількості обчислювальних ресурсів і часу, необхідного для навчання моделей глибинного навчання, а також великих, складних наборів даних.

Основним обмеженням трансферного навчання є те, що характеристики моделі, які вивчаються під час першого завдання, є загальними, а не специфічними для першого завдання. На практиці це означає, що моделі, навчені розпізнаванню певних типів зображень, можна повторно використовувати для розпізнавання інших зображень, якщо загальні ознаки зображень подібні.

Використання трансферного навчання має кілька важливих концепцій. Щоб зрозуміти реалізацію трансферного навчання, нам потрібно розглянути, як виглядає попередньо навчена модель і як цю модель можна точно налаштувати для ваших потреб.

Є два способи вибрати модель для трансферного навчання. Можна створити модель з нуля для власних потреб, зберегти параметри та структуру моделі, а потім використовувати її повторно.

Другий спосіб реалізувати навчання з перенесенням — просто взяти вже існуючу модель і повторно використати її, настроюючи її параметри та гіперпараметри під час цього. У цьому випадку ми будемо використовувати попередньо навчену модель і модифікувати її.

Існує велика різноманітність попередньо навчених моделей, які можна використовувати в PyTorch. Прикладом з попередньо підготовлених CNN можуть бути AlexNet, CaffeResNet, Inception, Серія ResNet, Серія VGG. Ці попередньо підготовлені моделі доступні через API PyTorch, і за вказівками PyTorch завантажить їх специфікації на вашу машину. Як пояснює

документація PyTorch [22] щодо навчання з перенесенням, існує два основних способи використання передачі навчання: тонка настройка CNN або використання CNN як екстрактора фіксованих функцій.

Під час точного налаштування CNN використовуються ваги, які має попередньо навчена мережа, замість того, щоб випадково ініціалізувати їх, а потім тренуються як зазвичай. На відміну від цього, підхід вилучення функцій означає, що ви будете підтримувати всі ваги CNN, за винятком кількох останніх шарів, які будуть ініціалізовані випадковим чином і навчені як зазвичай.

Тонка настройка моделі важлива, оскільки, хоча модель була попередньо навчена, вона була навчена іншим хоча і подібним завданням. Оскільки перші кілька шарів мережі — це лише шари вилучення ознак, і вони будуть працювати подібним чином на подібних зображеннях, їх можна залишити як є. Тому, якщо набір даних невеликий і подібний, єдине навчання, яке потрібно зробити, - це навчання кількох останніх шарів. Чим більшим і складнішим стає набір даних, тим більше потрібно буде перенавчати модель. Слід зазначити, що навчання з перенесенням працює найкраще, коли набір даних, який використовується, менший, ніж початкова попередньо навчена модель, і подібний до зображень, поданих до попередньо навченої моделі.

Робота з моделями навчання передачі означає вибір, які шари заморозити, а які розморозити . Заморозити модель означає наказати зберегти параметри (ваги) у вказаних вами шарах. Розморожування моделі означає повідомлення, що ви хочете, щоб шари, які ви вказали, були доступні для навчання, щоб їх ваги можна було навчати.

Простіше кажучи, попередньо навчена модель — це модель, створена кимось іншим для вирішення подібної проблеми. Замість того, щоб створювати модель з нуля для вирішення подібної проблеми, ви

використовуєте модель, навчену іншій задачі, як відправну точку. Попередньо навчена модель може не бути на 100% точною для кожної окремої задачі у застосуванні, але вона економить величезні зусилля.

3.2.3 Попередня обробка даних

Також методом покращення точності є поліпшення продуктивності за рахунок обробки даних. Підготовка даних є важливим кроком, потрібно поекспериментувати з етапами попередньої обробки даних, які підходять для даних, щоб побачити, чи можна отримати бажане підвищення точності моделі.

Існує три типи попередньої обробки: додавання атрибутів до даних, видалення атрибутів з даних, перетворення атрибутів у дані.

Розширені моделі можуть мати відносини зі складних атрибутів, хоча деякі моделі вимагають, щоб ці відносини були чітко прописані. Отримання нових атрибутів з навчальних даних для включення в процес моделювання може підвищити продуктивність моделі.

Атрибути категорій можуть бути перетворені на n-двійкові атрибути, де n – це кількість категорій (або рівнів), які має атрибут. Ці денормалізовані або розкладені атрибути відомі як фіктивні атрибути або змінні фіктивні.

Перетворений варіант атрибуту може бути доданий до набору даних, щоб дозволити лінійному методу використовувати можливі лінійні та нелінійні відносини між атрибутами. Можна використовувати прості перетворення, такі як log, square та square root. Атрибути з відсутніми даними можуть мати ці відсутні дані, поставлені за допомогою надійного методу, такого як k-найближчі сусіди.

Деякі методи погано працюють з надлишковими атрибутами, або атрибутами що дублюються, тому можна підвищити точність моделі, видаляючи їх з ваших даних. Дані навчання можуть проектуватися в простори меншого розміру, але все ж таки характеризують внутрішні зв'язки в даних. Популярним підходом є аналіз основних компонентів (PCA), де основні компоненти, знайдені методом, можуть бути прийняті як скорочений набір вхідних атрибутів.

Просторова проекція символу перетворює дані на поверхню багатовимірної сфери. Результати можуть бути використані для підкреслення існування викидів, які можуть бути змінені або видалені з даних.

Робота деяких алгоритмів погіршуються із існуванням висококорельованих атрибутів. Парні атрибути з високою кореляцією можуть бути ідентифіковані, і найбільш корельовані атрибути можуть бути видалені з даних.

Перетворення навчальних даних може зменшити асиметрію даних, а також помітність викидів даних. Багато моделей очікують перетворення даних до того, як застосується алгоритм. Ще одним загальним способом є стандартизація даних. Перетворення даних так, щоб вони мали середнє значення, що дорівнює нулю, і стандартне відхилення, що дорівнює одиниці.

Стандартне перетворення масштабування - відображення даних у форматі від нуля до одиниці. Зазвичай це називається нормалізацією даних.

Перекошені дані - це дані, розподіл яких зсувається в один чи інший бік (великі чи менші значення), а не розподіляється нормально. Деякі методи передбачають нормально розподілені дані і можуть працювати краще, якщо усунено перекіс. Можна спробувати замінити атрибут логарифмом, квадратним коренем чи зворотним значенням.

Перетворення Боксу-Коксу або сімейство перетворень можна використовувати для надійного коригування даних для усунення перекосу.

Числові дані можна зробити дискретними, згрупувавши значення в комірки. Це називається дискретизацією даних. Цей процес можна виконати вручну, хоча він надійніший, якщо виконується систематично і автоматично з використанням евристики, яка має сенс у цій галузі.

4 РОЗРОБКА АРХІТЕКТУРИ НЕЙРОННОЇ МЕРЕЖІ

4.1 Обґрунтування вибору інструментарію

Для досягнення мети кваліфікаційної роботи та вирішення задачі, поставленої в першому розділі, в якості мови програмування для розробки алгоритму було вирішено використовувати Python. Оскільки ця мова перш за все добре збирає, систематизує та аналізує дані, а також має відмінну продуктивність при обробці даних. Далі трохи детальніше про переваги використання цієї мови для задач комп’ютерного зору.

По-перше, Python пропонує можливість вибрати або використовувати ООП або скрипти. Також немає необхідності перекомпільовувати вихідний код, розробники можуть впровадити будь-які зміни та швидко побачити результати. Програмісти можуть комбінувати Python та інші мови для досягнення своїх цілей. Більше того, гнучкість дозволяє розробникам вибирати стилі програмування, які їм цілком зручні, або навіть комбінувати ці стилі для вирішення різних типів проблем найефективнішим способом.

По-друге, Python не тільки зручна у використанні та легка у освоєнні, але й дуже універсальна мова. Python для розробки машинного навчання може працювати на будь-якій платформі, включаючи Windows, MacOS, Linux, Unix та ще двадцять одну платформу. Щоб перенести процес з однієї платформи на іншу, розробникам необхідно внести кілька невеликих змін, щоб створити виконувану форму коду для вибраної платформи.

Великий вибір бібліотек є однією з головних причин, чому Python є найпопулярнішою мовою програмування, що використовується для ШІ. Бібліотека — це модуль або група модулів, опублікованих різними джерелами,

такими як PyPi, які містять попередньо написаний фрагмент коду, який дозволяє користувачам отримувати певну функціональність або виконувати різні дії. Бібліотеки Python забезпечують елементи базового рівня, тому розробникам не доводиться кожен раз кодувати їх із самого початку. З цього випливає те, що Python відмінно підходить для поставленої задачі.

Отже, наступним кроком буде огляд, порівняння та вибір бібліотеки, що найкраще підходить саме для роботи із зображеннями для подальшого аналізу рентгенівських знімків. Серед найбільш відомих бібліотек відокремлюють наступні:

1) TensorFlow

Платформа Google з відкритим кодом TensorFlow є, мабуть, найпопулярнішим інструментом для машинного навчання та глибинного навчання. TensorFlow заснована на JavaScript і оснащена широким набором інструментів і ресурсів спільноти, які полегшують навчання та розгортання моделей ML/DL. Хоча основний інструмент дозволяє створювати та розгортати моделі у браузерах, можна використовувати TensorFlow Lite для розгортання моделей на мобільних або вбудованих пристроях. Крім того, якщо ви хочете навчати, створювати та розгортати моделі ML/DL у великих виробничих середовищах, TensorFlow Extended служить цій меті. Переваги TensorFlow: найкраще підходить для розробки моделей DL та експериментів з архітектурами глибинного навчання. Вона використовується для функцій інтеграції даних, включаючи введення разом графіків, таблиць SQL та зображень [23].

2) Keras

Ще одна платформа глибинного навчання з відкритим – Keras. Цей інструмент може працювати поверх TensorFlow, Theano, Microsoft Cognitive Toolkit і PlaidML. Головна перевага Keras — це її швидкість — вона має

вбудовану підтримку паралелізму даних, а отже, може обробляти величезні обсяги даних, прискорюючи час навчання моделей. Як написано на Python, вона неймовірно проста у використанні та розширюваний [24]. Переваги Keras: відмінно підходить для новачків, які тільки почали свій шлях у цій галузі. Це дозволяє легко навчатися та створювати прототипи простих концепцій, а також сприяє швидкому експериментуванню з глибокими нейронними мережами. Основне використання Keras — це класифікація, генерація тексту та узагальнення, тегування, переклад разом із розпізнаванням мовлення тощо.

3) PyTorch

PyTorch — це фреймворк глибинного навчання з відкритим кодом, розроблений Facebook. Він заснований на бібліотеці Torch і був розроблений з однією основною метою – прискорити весь процес від дослідницького прототипу до розгортання у виробництві. Він відмінно підходить для навчання, створення, розгортання невеликих проектів і прототипів. Цей фреймворк широко використовується для програм глибинного навчання, таких як обробка природної мови та комп’ютерний зір. PyTorch не має жодного інструменту візуалізації, такого як TensorBoard, але завжди можна використовувати бібліотеку, як-от matplotlib. У порівнянні з TensorFlow, PyTorch є більш інтуїтивно зрозумілим [25]. Останніми роками PyTorch отримав високий рівень поширення серед спільноти фреймворків глибинного навчання і вважається серйозним конкурентом TensorFlow.

Дивлячись на переважне застосування та переваги трьох найбільш відомих бібліотек для машинного навчання, було вирішено обрати бібліотеку PyTorch, оскільки вона спеціалізується на можливостях, необхідних для вирішення задачі комп’ютерного зору.

4.2 Навчальна вибірка

Як було зазначено у попередньому розділі, одним з етапів навчання моделі є формування набору даних. За основу було взято набір даних відкритого доступу [26].

Набір навчальних даних складається з 15264 зображень (512x512 пікселів), які були класифіковані експертом-рентгенологом. Тестовий набір даних складається з 400 таких зображень, які надходять з одного дистрибутива.

В ході дослідження набору, було виявлено, що навчальний набір даних значно незбалансований, і близько 90% зображень належать до класу, що не є Covid. Проблема зі значно незбалансованими наборами полягає в тому, що навіть наївний алгоритм навчання, який просто виводить як вихідний клас класу більшості, досягне високої точності. Іншими словами, він позначатиме всіх як людей без Covid і досягне «точності» 90%, навіть якщо близько 10% людей насправді мають Covid. Спочатку я почала з незбалансованого набору даних для навчання, як він є. З огляду на ретроспективу, це виявилося не найкращою відправною точкою, але треба починати, і в подальшому коригувати процес розробки.

4.3 Розробка алгоритму та проведення експериментів

Як перший крок, було імпортовано бібліотеки numpy, pandas, os, torch тощо та встановлено базовий шлях, де розташована папка даних. Потім створення фреймів даних за допомогою pandas, щоб містити назви зображень навчальних наборів даних, їх мітки та назви зображень тестових наборів даних відповідно.

Щоб перевірити, чи правильно завантажені кадри (фрейми) даних, добре перевірити перші кілька рядків train_dataset і test_dataset. На рисунку 19 код і отриманий результат із записника Jupyter.

```
In [4]: train_dataset.head(3)
```

Out[4]:

	File	Label
0	1329638562-58609.jpg	0
1	956062378-32202.jpg	0
2	118943055-21826.jpg	1

```
In [5]: test_dataset.head(3)
```

Out[5]:

	File
0	1002536285-58583.jpg
1	100738077-54097.jpg
2	100945025-10622.jpg

Рисунок 19 – Перші 3 рядки фреймів навчального набору даних та перші 3 назви зображень із тестового набору даних.

Як можна побачити, назви зображень знаходяться в стовпці «File», а мітки — у стовпці «Label» для навчальних зображень. Для тестових зображень у нас є імена в стовпці «File».

В ході навчання мережа по суті класифікує, чи належіть зображення одному із 2 класів, так в рамках обчислювань клас «0» – рентгенівський знімок нормальний, а «1» – знімок з великою вирогідністю людини, що хвора на COVID-19.

Оскільки одним із методів оптимізації часу розробки було рішення про використання трансферного навчання, тобто за основу нейронної мережі буде попередньо навчена модель, необхідно завантажити її, як зображено на рисунку 20.

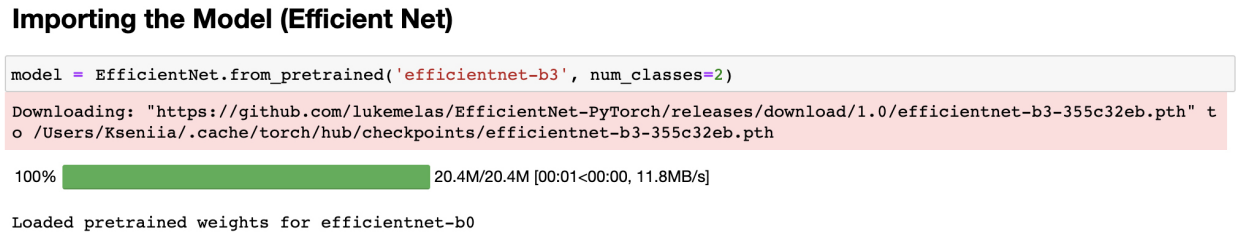


Рисунок 20 – Завантаження попередньо навченої моделі.

Наступним кроком є створення навчального набору даних із зображень і міток у формі, яку може обробляти Pytorch. Dataloader — це функція в Pytorch, яка дає нам змогу зручно об’єднувати зображення та їх мітки для створення матриці/тензора, який потім може використовуватися PyTorch як вхідні дані для алгоритму навчання.

Далі я підготувала навчальний набір за допомогою класу Dataloader набору даних, який я створила раніше. Я назвала його training_set_untransformed, оскільки я ще не застосував перетворення даних, як-от обертання або трансляція, до зображень.

Наступним етапом є нормалізація даних. Оскільки під час перевірки даних можна помітити, що деякі функції представлені в різних масштабах, це може вплинути на продуктивність нейронної мережі, оскільки конвергенція відбувається повільніше. Нормалізація даних означає перетворення всіх їх до однієї шкали в межах [0–1]. Було створено метод transforms.Compose для зміни розміру зображень, випадкового застосування обертання та горизонтального

перекладу та перетворення отриманої матриці в тензор. Таке перетворення корисно для збільшення даних. Функція transforms.ToTensor() перетворює наше зображення у числа. Він поділяє три кольори, з яких складається кожен піксель нашого зображення: червоний, зелений та синій. По суті, це перетворює одне зображення на три зображення (одне пофарбоване в червоний колір, одне зелене та одне синє). Потім він перетворює пікселі кожного забарвленого зображення в яскравість їхнього кольору від 0 до 255. Ці значення діляться на 255, тому вони можуть знаходитися в діапазоні від 0 до 1. Наше зображення тепер є Тензорним факелом – структурою даних, що зберігає багато чисел (Лістинг 4.1).

Лістинг 4.1 Реалізація функції редагування даних

```
transform_train =
transforms.Compose([transforms.Resize((224,224)),transforms.R
andomApply([torchvision.transforms.RandomRotation(10),transfo
rms.RandomHorizontalFlip()],0.7),transforms.ToTensor()])
```

Наступним кроком є використання методу transforms для навчального набору даних. Для зображень у класі меншості (випадки covid) було збільшено вибірку, щоб після застосування трансформації кількість зображень в обох класах була однаковою.

Потім було розділено ці щойно створені зображення на набори для навчання та перевірки у співвідношенні 80:20 (Лістинг 4.2).

Лістинг 4.2 Реалізація розподілу даних у зазначеному співвідношенні

```
train_size = int(0.8 * len(new_created_images))
validation_size = len(new_created_images) - train_size
```

```
train_dataset, validation_dataset =
torch.utils.data.random_split(new_created_images,
[train_size,validation_size])
```

Після чого поступово дійшли до стадії, коли необхідно згадати основні гіпермараметри для навчання, найкращі з яких було виведено шляхом експериментів. Почнемо з гіперпараметру швидкість навчання. Він відноситься до кроку зворотного поширення, коли параметри оновлюються відповідно до функції оптимізації. По суті, це показує, наскільки важливою є зміна ваги після повторного калібрування.

Так наприклад нижча швидкість навчання призводить до кращої конвергенції, але не обов'язково найкращої точності тесту. Проблема може полягати в тому, що алгоритм ще не вивчив набір ваг, які призводять до найменшої помилки для тестового набору. Вища швидкість навчання, з іншого боку, може призвести до того, що алгоритм перевищить оптимальний набір ваг. Були проведені експерименти зі швидкістю навчання 10e-3, 10e-4 і 10e-5 і було виявлено, що 10e-4 призвело до найнижчого рівня помилок для тестового набору даних.

Щодо розміру пакету: коли ви стикаєтеся з мільярдами даних, це може призвести до неефективного, а також контрпродуктивного живлення вашої нейронної мережі. Хорошою практикою є наповнення його меншими вибірками даних, які називаються пакетами: при цьому кожен раз, коли алгоритм тренується сам, він навчатиметься на вибірці такого ж розміру пакету. Типовий розмір — 32 або більше, однак потрібно пам'ятати, що якщо розмір занадто великий, ризик полягає в надто узагальненій моделі, яка погано вписується в нові дані. Під час його збільшення до 256 або навіть до 64 з'явилися проблеми пам'яті, і тому довелося вибрати менший розмір пакету.

Було проведено експеримент із 32 і 16. З цих двох розмірів пакетів розмір пакету 32 привів до найнижчого рівня помилок із швидкістю навчання 10e-4.

Використовуючи функцію Dataloader у Pytorch, було створено пакети по 32 кожного з навчального набору даних і перемішано їх, щоб забезпечити приблизно однакове представлення обох класів у всіх пакетах.

Бажано періодично зберігати ваги, створені тренувальним процесом. У випадку, якщо модель виходить з ладу з якоїсь причини, можна викликати збережені ваги та відновити тренування замість того, щоб знову починати з нуля.

Я використовувала втрату перехресної ентропії як критерій для розрахунку втрат, оскільки ми маємо справу з проблемою класифікації. Нейронні мережі навчаються за допомогою процесу оптимізації, який вимагає функції втрат для обчислення помилки моделі. Як правило, з нейронними мережами ми прагнемо мінімізувати помилку. Таким чином, цільову функцію часто називають функцією вартості або функцією втрат, а значення, обчислене функцією втрат, називають просто «втратою».

Перехресна ентропія та середньоквадратична помилка — це два основні типи функцій втрат, які використовуються під час навчання моделей нейронної мережі. У контексті алгоритму оптимізації функція, яка використовується для оцінки рішення-кандидата, тобто набору ваг, називається цільовою функцією. У рамках максимальної правдоподібності похибка між двома розподілами ймовірностей вимірюється за допомогою крос-ентропії. При моделюванні задачі класифікації, де цікавить відображення вхідних змінних до мітки класу, ми можемо моделювати проблему як прогнозування ймовірності того, що приклад належить кожному класу. У задачі бінарної класифікації буде два класи, тому ми можемо передбачити ймовірність належності прикладу до першого класу. У навчальному наборі

даних ймовірність належності прикладу до даного класу буде 1 або 0, оскільки кожен зразок у наборі навчальних даних є відомим прикладом із домену, відповідь буде відома. Отже, для класифікації переважно використовують перехресну ентропію, тоді як середньоквадратична помилка є одним з кращих варіантів для регресії.

У процесі навчання ми намагаємося мінімізувати втрати функції та оновлювати параметри підвищення точності. Параметри нейронної мережі – це зазвичай ваги зв'язків. І тут параметри вивчаються на етапі навчання. Отже, сам алгоритм і вхідні дані налаштовують ці параметри.

Таким чином, оптимізатор – це метод досягнення найкращих результатів, допомога у прискоренні навчання. Іншими словами, це алгоритм, який використовується для незначної зміни параметрів, таких як ваги та швидкість навчання, щоб модель працювала правильно та швидко.

Також було застосовано оптимізатор Адама, який є добре відомим методом оновлення ваги для глибокого навчання. Адам – це алгоритм оптимізації, який можна використовувати замість класичної процедури стохастичного градієнтного спуску для ітеративного оновлення ваги мережі на основі навчальних даних. Серед його переваг його використання відзначають, що він обчислювально ефективний, має невеликі вимоги до пам'яті, добре підходить для завдань, які є великими з точки зору даних та/або параметрів, а також гіперпараметри мають інтуїтивно зрозумілу інтерпретацію і зазвичай потребують невеликої настройки.

Кажучи про параметр кількості епох, цей гіперпараметр показує, скільки часу ви хочете, щоб ваш алгоритм навчався для всього вашого набору даних слід зазначити, що епохи відрізняються від ітерацій: останні є кількістю пакетів, необхідних для завершення однієї епохи. Знову ж таки, кількість епох залежить від типу даних і завдання, яке стоїть. Ідея може полягати в тому, щоб

накласти умову, при якій епохи зупиняються, коли помилка близька до нуля. Або, що простіше, можна почати з відносно невеликої кількості епох, а потім поступово збільшувати її, відстежуючи деякі показники оцінки (наприклад, точність). Саме так було встановлено показник цього на 11.

Зниження швидкості навчання – це параметр, який використовується для оновлення швидкості навчання після кожної епохи, і я встановила консервативну швидкість загасання 0.99, оскільки швидкість навчання вже низька. Частина коду для тренування моделі зазначена в лістингу 4.3.

Лістинг 4.3 Реалізація коду з навчання моделі після налаштування параметрів.

```
for epoch in range(epochs):
    running_loss = 0.0
    correct=0
    total=0
    class_correct = list(0. for _ in classes)
    class_total = list(0. for _ in classes)

    for i, data in enumerate(training_generator, 0):
        inputs, labels = data
        t0 = time()
        inputs, labels = inputs.to(device), labels.to(device)
        labels = eye[labels]
        optimizer.zero_grad()
        #torch.cuda.empty_cache()
        outputs = model(inputs)
        loss = criterion(outputs, torch.max(labels, 1)[1])
        _, predicted = torch.max(outputs, 1)
        _, labels = torch.max(labels, 1)
        c = (predicted == labels.data).squeeze()
        correct += (predicted == labels).sum().item()
        total += labels.size(0)
        accuracy = float(correct) / float(total)
        history_accuracy.append(accuracy)
        history_loss.append(loss)
        loss.backward()
        optimizer.step()
```

```
        for j in range(labels.size(0)):
            label = labels[j]
            class_correct[label] += c[j].item()
            class_total[label] += 1
        running_loss += loss.item()

        print( "Epoch : ",epoch+1," Batch : ", i+1," Loss :
",running_loss/(i+1)," Accuracy : ",accuracy,"Time
",round(time()-t0, 2),"s" )

    for k in range(len(classes)):
        if(class_total[k]!=0):
            print('Accuracy of %5s : %2d %%' % (classes[k],
100 * class_correct[k] / class_total[k]))

            print('[%d epoch] Accuracy of the network on the
Training images: %d %%' % (epoch+1, 100 * correct / total))

    if epoch%10==0 or epoch==0:
        torch.save(model.state_dict(),
os.path.join(PATH_SAVE,str(epoch+1)+'_'+str(accuracy)+'.pth')
)
torch.save(model.state_dict(),
os.path.join(PATH_SAVE,'Last_epoch'+str(accuracy)+'.pth'))
```

На початку експерименту з відстежування впливу зміни гіперпараметрів на точність передбачення нейронною мережею, процес навчання виглядав, як зображено на рисунку 21.

На цьому етапі не коригувалися розмір пакету, швидкість навчання, а також не проводилась нормалізація даних, саме тому точність класифікації випадків із захворюванням лише 93% після 11 епох навчання, як зазначалось вище до нормалізації даних зразків рентгенівських знімків із захворюванням на COVID-19 було значно менше, аніж здорових.

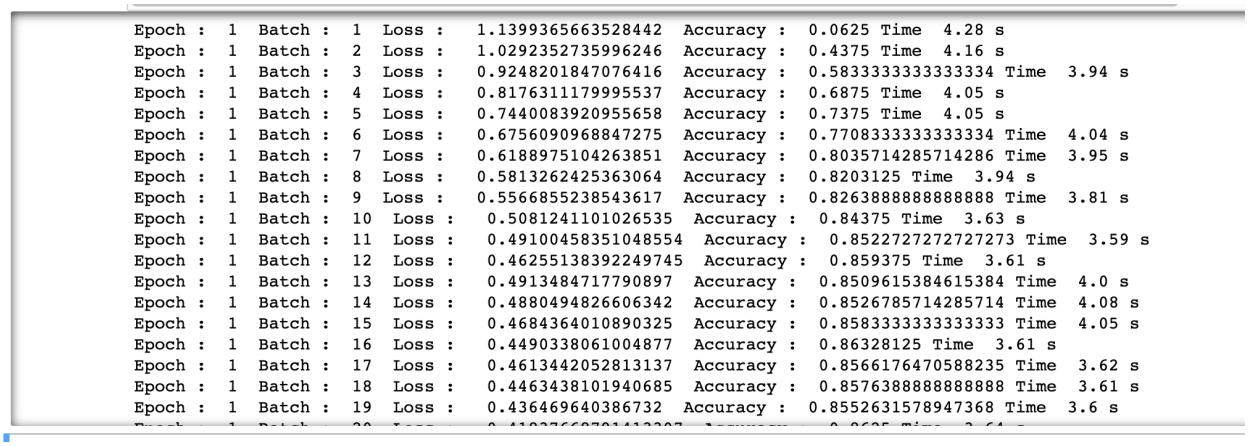


Рисунок 21 – Перша епоха в навчанні моделі без зміни параметрів за замовчуванням.

На рисунку 22 також можна побачити час, за який проходить навчання один пакет навчальних зразків становить від 4 до 6 секунд в середньому, та кількість пакетів становить 954. Але слід зазначити, що на час навчання на одному пакеті не є характеристикою, гарний показник якої впливає на точність аналізу зображення.

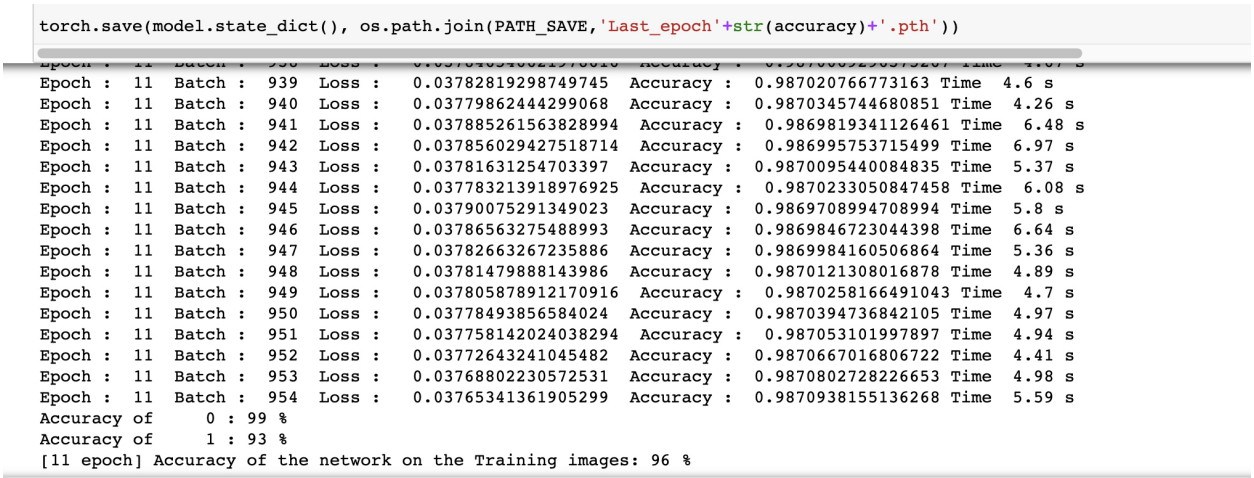


Рисунок 22 – Результат навчання моделі без належних заходів щодо покращення точності.

На рисунку 23 зображено графік точності навчання та втрат крос-ентропії за ітерації, де видно, що рівень

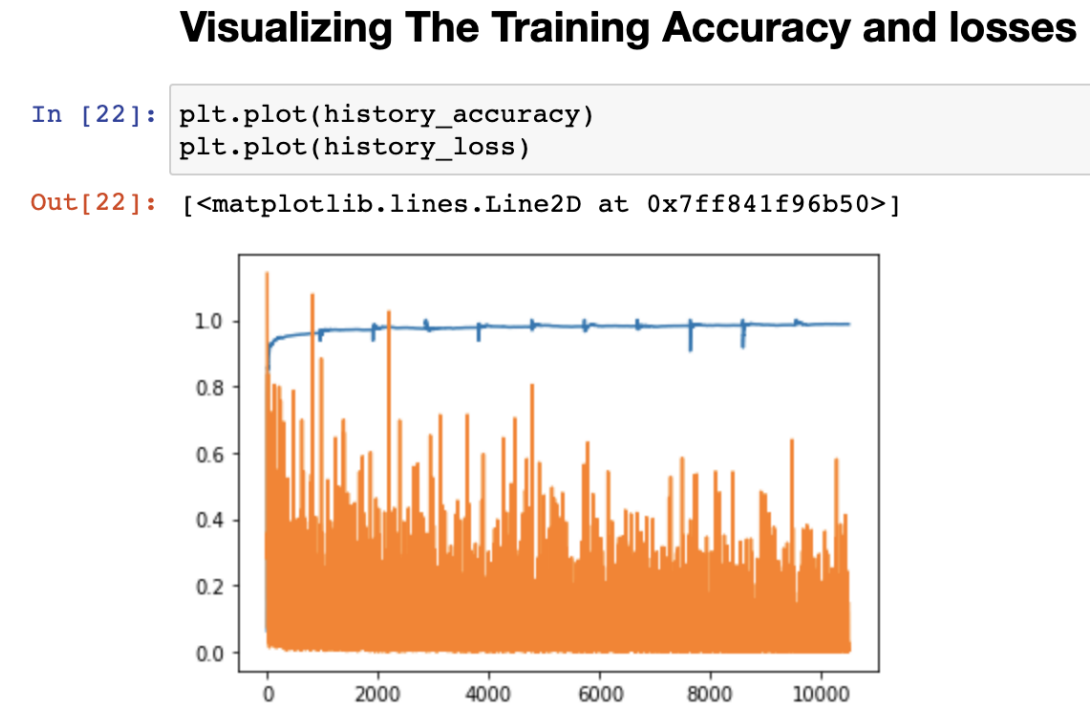


Рисунок 23 – Графік точності навчання та втрат крос-ентропії за ітерації.

Для того, щоб виявити, як зміна показників гіперпараметрів впливає на точність моделі, а також взагалі на процес її навчання, було зроблено наступний експеримент: тестовий набір даних було нормалізовано та було збільшено вибірку даних із рентгенами хворих, щоб після застосування трансформації кількість зображень в обох класах була однаковою; розмір пакету було встановлено 32, що зменшило кількість пакетів за 1 епоху, але щоб досягнути точність близько 98% було збільшено кількість епох до 14, як наслідок час на навчання на одному пакеті даних збільшилось в 3-4 рази. Вище зазначені зміни можна побачити на рисунку 24, а графік точності навчання та втрат зображено на рисунку 25.

```
Epoch :  14  Batch :  462  Loss :   0.008905883685373112  Accuracy :  0.997362012987013 Time  18.49 s
Epoch :  14  Batch :  463  Loss :   0.008921122460469043  Accuracy :  0.9973677105831533 Time  22.86 s
Epoch :  14  Batch :  464  Loss :   0.008910336598358394  Accuracy :  0.9973733836206896 Time  14.47 s
Epoch :  14  Batch :  465  Loss :   0.008891401289551024  Accuracy :  0.9973790322580646 Time  22.08 s
Epoch :  14  Batch :  466  Loss :   0.00892774242885716  Accuracy :  0.9973175965665236 Time  17.17 s
Epoch :  14  Batch :  467  Loss :   0.008912599262266297  Accuracy :  0.9973233404710921 Time  14.58 s
Epoch :  14  Batch :  468  Loss :   0.008906687036341104  Accuracy :  0.9973290598290598 Time  28.53 s
Epoch :  14  Batch :  469  Loss :   0.008892141390394648  Accuracy :  0.9973347547974414 Time  14.53 s
Epoch :  14  Batch :  470  Loss :   0.008876719238453897  Accuracy :  0.9973404255319149 Time  14.07 s
Epoch :  14  Batch :  471  Loss :   0.00886803129654229  Accuracy :  0.9973460721868365 Time  26.45 s
Epoch :  14  Batch :  472  Loss :   0.008866756110534372  Accuracy :  0.9973516949152542 Time  14.56 s
Epoch :  14  Batch :  473  Loss :   0.008848355644555191  Accuracy :  0.9973572938689218 Time  14.53 s
Epoch :  14  Batch :  474  Loss :   0.008831933724903575  Accuracy :  0.9973628691983122 Time  27.28 s
Epoch :  14  Batch :  475  Loss :   0.008840960641253715  Accuracy :  0.9973684210526316 Time  14.49 s
Epoch :  14  Batch :  476  Loss :   0.00882542083474175  Accuracy :  0.9973739495798319 Time  14.49 s
Epoch :  14  Batch :  477  Loss :   0.008807979166454394  Accuracy :  0.9973794549266247 Time  26.77 s
Accuracy of     0 : 99 %
Accuracy of     1 : 98 %
[14 epoch] Accuracy of the network on the Training images: 98 %
```

Рисунок 24 – Результат навчання моделі зі змінення параметрів.

```
plt.plot(history_accuracy)
plt.plot(history_loss)
```

```
[<matplotlib.lines.Line2D at 0x7fd6248c2df0>]
```

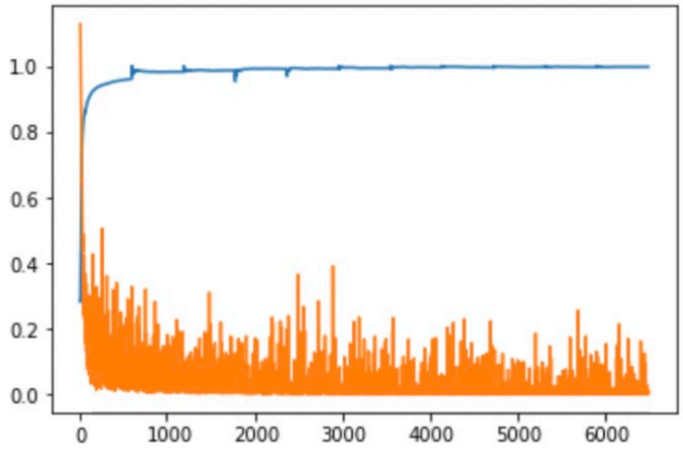
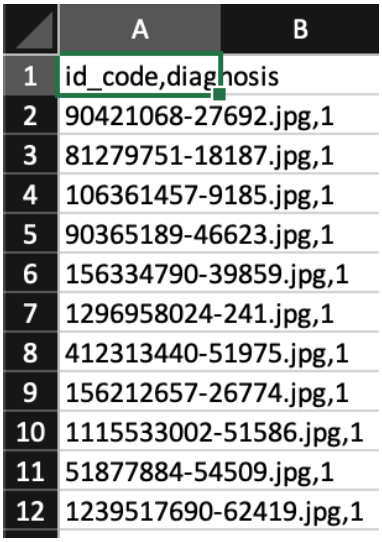


Рисунок 25 – Графік точності навчання та втрат крос-ентропії за ітерації після змін.

Як не дивно, спочатку модель класифікувала майже всі тести як «1», всупереч очікуванням «0». В ході аналізу було виявлено, що проблема полягала в тому, що тестові зображення не трансформувалися, як навчальні, а нетрансформовані тестові зображення чомусь нагадували трансформовані

навчальні зображення мітки «1» (Рисунок 26). Оскільки, перетворюючи навчальні зображення за допомогою обертань і перекладів, було змінено розподіл. Однак, залишивши тестові зображення як є, ненавмисно було застосовано алгоритм до «іншого» розподілу. Тому після цього спостерігання було застосовано ідентичні трансформації до тестових зображень, після чого точність досягла 0,975. Отже, так само як із навчальним набором, для тестового набору даних необхідно провести нормалізацію даних, щоб він відповідав тому самому розподілу, що й навчальний набір даних (Лістинг 4.4).



	A	B
1	id_code,diagnosis	
2	90421068-27692.jpg,1	
3	81279751-18187.jpg,1	
4	106361457-9185.jpg,1	
5	90365189-46623.jpg,1	
6	156334790-39859.jpg,1	
7	1296958024-241.jpg,1	
8	412313440-51975.jpg,1	
9	156212657-26774.jpg,1	
10	1115533002-51586.jpg,1	
11	51877884-54509.jpg,1	
12	1239517690-62419.jpg,1	

Рисунок 26 – Експеримент перевірки моделі на ненормалізованому тестовому наборі.

Лістинг 4.4 Реалізація функції нормалізації даних.

```
test_transforms = transforms.Compose([transforms.Resize((224,
224)),transforms.RandomApply([torchvision.transforms.RandomRo
tation(10),transforms.RandomHorizontalFlip()],0.7),
transforms.ToTensor(),])
```

Щоб передбачити клас тестових зображень, нам потрібно визначити функцію predict_image (Лістинг 4.5):

Лістинг 4.5 Реалізація функції предбачення.

```
def predict_image(image):
    image_tensor = test_transforms(image)
    image_tensor = image_tensor.unsqueeze_(0)
    input = Variable(image_tensor)
    input = input.to(device)
    output = model(input)
    index = output.data.cpu().numpy().argmax()
    return index
```

Наступний фрагмент коду перевіряє, наскільки добре модель працює з набором даних перевірки (Лістинг 4.6).

Лістинг 4.6 Реалізація функції перевірки вхідних тестових зображень.

```
correct_counter=0

for i in range(len(validation_dataset)):
    image_tensor = validation_dataset[i][0].unsqueeze_(0)
    input = Variable(image_tensor)
    input = input.to(device)
    output = model(input)
    index = output.data.cpu().numpy().argmax()
    if index == validation_dataset[i][1]:
        correct_counter+=1
print("Accuracy=",correct_counter/len(validation_dataset))
```

Без цієї перевірки можна натрапити на помилку такого виду, як на рисунку 27. Ця помилка говорить про те, що вхідне зображення має лише 1 канал, коли повинно бути 3 канали. З цього випливає, що вхідне зображення не було нормалізоване для подальшої обробки функціями моделі та бібліотеки Pytorch для подальшого аналізу зображення.

За допомогою наступних рядків можна передбачити класи зображень тестового набору та зберегти їх у файлі csv (Лістинг 4.7).

```
~/opt/anaconda3/lib/python3.8/site-packages/torch/nn/modules/module.py in _call_impl(self, *input, **kwargs)
   1049         if not (self._backward_hooks or self._forward_hooks or self._forward_pre_hooks or _global_backward_ho
oks
   1050                 or _global_forward_hooks or _global_forward_pre_hooks):
-> 1051             return forward_call(*input, **kwargs)
   1052         # Do not call functions when jit is used
   1053         full_backward_hooks, non_full_backward_hooks = [], []

~/opt/anaconda3/lib/python3.8/site-packages/efficientnet_pytorch/utils.py in forward(self, x)
    273     def forward(self, x):
    274         x = self.static_padding(x)
--> 275         x = F.conv2d(x, self.weight, self.bias, self.stride, self.padding, self.dilation, self.groups)
    276         return x
    277

RuntimeError: Given groups=1, weight of size [40, 3, 3, 3], expected input[1, 1, 513, 722] to have 3 channels, but go
t 1 channels instead
```

Рисунок 27 – Повідомлення про помилку при зчитуванні неперевіреного зображення

Лістинг 4.7 Реалізація функції перевірки зображень та зберігання результатів.

```
submission=pd.read_csv(BASE_PATH+'sample_submission.csv')

IMG_TEST_PATH=os.path.join(BASE_PATH,'test/')
for i in range(len(submission)):
    img=Image.open(IMG_TEST_PATH+submission.iloc[i][0])
    prediction=predict_image(img)
```

```
    submission_csv=submission_csv.append({'File':
submission.iloc[i][0],'Label': prediction},ignore_index=True)
    if(i%10==0 or i==len(submission)-1):
        print('[',32*'=','>]
',round((i+1)*100/len(submission),2),' % Complete')

submission_csv.to_csv('submission_epoch11_2.csv',index=False)
```

В кінці навчання моделі вже після застосування усіх необхідних заходів для покращення точності в ході проведення дослідження, на тренувальному наборі даних, точність нейронної мережі складає 99%, результат навчання наведено на рисунку 28.

```
Epoch :  11  Batch :  641  Loss :   0.00594567382477315  Accuracy :  0.9980499219968799 Time 14.13 s
Epoch :  11  Batch :  642  Loss :   0.005936753549560033  Accuracy :  0.9980529595015576 Time 14.33 s
Epoch :  11  Batch :  643  Loss :   0.005927524898009113  Accuracy :  0.9980559875583204 Time 14.9 s
Epoch :  11  Batch :  644  Loss :   0.005925419287726239  Accuracy :  0.9980590062111802 Time 14.14 s
Epoch :  11  Batch :  645  Loss :   0.005917660177357353  Accuracy :  0.998062015503876 Time 15.33 s
Epoch :  11  Batch :  646  Loss :   0.005908548086209301  Accuracy :  0.9980650154798761 Time 14.89 s
Epoch :  11  Batch :  647  Loss :   0.005899633814067241  Accuracy :  0.9980680061823802 Time 14.58 s
Epoch :  11  Batch :  648  Loss :   0.005891362798900762  Accuracy :  0.998070987654321 Time 17.09 s
Epoch :  11  Batch :  649  Loss :   0.005885811755253251  Accuracy :  0.9980739599383667 Time 14.31 s
Epoch :  11  Batch :  650  Loss :   0.005890584583930481  Accuracy :  0.9980769230769231 Time 19.27 s
Epoch :  11  Batch :  651  Loss :   0.006105068317304438  Accuracy :  0.9980318740399385 Time 22.3 s
Epoch :  11  Batch :  652  Loss :   0.006096434002588285  Accuracy :  0.9980348926380368 Time 20.59 s
Epoch :  11  Batch :  653  Loss :   0.006088523375768607  Accuracy :  0.9980379019908117 Time 21.22 s
Epoch :  11  Batch :  654  Loss :   0.00608013110512561  Accuracy :  0.9980409021406728 Time 14.13 s
Epoch :  11  Batch :  655  Loss :   0.006070956866454024  Accuracy :  0.998043893129771 Time 15.33 s
Epoch :  11  Batch :  656  Loss :   0.006069539576211845  Accuracy :  0.998046875 Time 14.56 s
Epoch :  11  Batch :  657  Loss :   0.006060434057112402  Accuracy :  0.9980478979193449 Time 17.14 s
Accuracy of     0 : 99 %
Accuracy of     1 : 99 %
[11 epoch] Accuracy of the network on the Training images: 99 %
```

Рисунок 28 – Точність моделі нейронної мережі після навчання.

Нижче, на рисунку 29, представлено графік точності навчання та втрат крос-ентропії за ітерації, а на рисунку 30 результати аналізу перших 10 із 400 зображень із тестового набору даних. Слід зазначити, що аналіз одного такого зображення нейронною мережею займає близько 0,3 секунд, що безперечно свідчить про те, що даний метод аналізу рентгенівських знімків значною

мірою зможе оптимізувати час постановки діагнозу лікарем, який буде мати попередній діагноз.

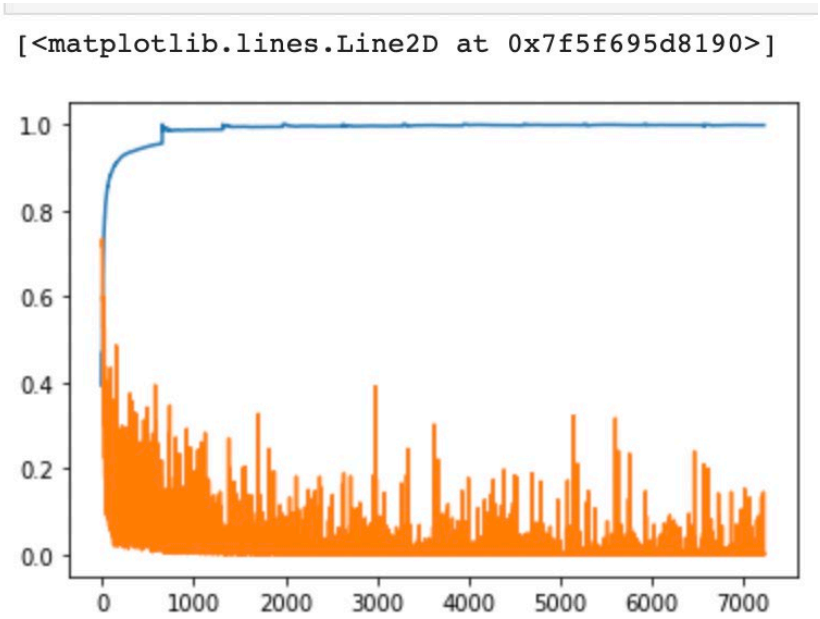


Рисунок 29 – Графік точності навчання та втрат крос-ентропії за ітерації.

	A	B
1	id_code,diagnosis	
2	90421068-27692.jpg,0	
3	81279751-18187.jpg,0	
4	106361457-9185.jpg,1	
5	90365189-46623.jpg,0	
6	156334790-39859.jpg,1	
7	1296958024-241.jpg,1	
8	412313440-51975.jpg,1	
9	156212657-26774.jpg,0	
10	1115533002-51586.jpg,0	

Рисунок 30 – Результат аналізу тестових зображень.

ВИСНОВКИ

В ході виконання роботи було ретельно розглянуто предметну область, а саме інтегрування та застосування різноманітних інформаційних технологій у медичній сфері. Відзначивши актуальність проблеми допоміжних для лікарей засобів діагностування вірусу COVID-19, було чітко визначені завдання та напрямок дослідження. Мною було досліджено застосування нейронних мереж саме для задачі розпізнавання образів на зображенні, в той час як вони використовуються ще й в задачах класифікації, ідентифікації, виявлення, оцінки руху та інших. Для цього було проведено огляд згорткових нейронних мереж, їх характеристик та методів, а також розглянуто варіанти їх реалізації та особливості їх застосування до розпізнавання та класифікації зображень.

В рамках аналізу і дослідження технологій навчання глибоких мереж було розглянуто найпоширеніші і найбільш підходящі архітектури згорткових нейронних мереж для глибинного навчання в рамках даного завдання. Слід враховувати скільки даних вимага та чи інша архітектура для коректного навчання, а також пам’ятати про апаратні обмеження, що можуть впливати на час розробки подібних інтелектуальних методів.

Не дивлячись на те, що вже існують загальні моделі навчання для розпізнавання образів, слід досліджувати існуючи методи та намагатися вдосконалити їх, оскілки через те, що відсутня єдина методологія вирішення прикладних завдань за допомогою нейромереж, доцільно стосовно кожної конкретної задачі вибирати не тільки їх архітектуру, а й метод навчання нейронної мережі. Саме тому було проведено аналіз можливих методів покращення точності роботи моделі глибинного навчання, серед яких слід безумовно відзначити технологію трансферного навчання, що зменшує час розробки нейронної мережі, а також має зазвичай велику документацію та

дослідження на різних прикладах, з яких можна вилучити стратегію поліпшення точності в рамках власної задачі. Ще одним важливим етапом в машинному навчанні є поліпшення показнику точності за рахунок ретельної підготовки даних та експерименти над ними, оскільки правильний формат даних для навчання та тесту поліпшує результат аналізу, тут слід згадати також про корегуванню гіперпараметрів та спостерігання їх впливу на показник точності.

ПЕРЕЛІК ПОСИЛАНЬ

1. Coronavirus. (COVID-19) Cases Statistics and Research Our World in Data. [Електронний ресурс]. Режим доступу: https://ourworldindata.org/covidcases

2. C. H. Chen. Computer Vision In Medical Imaging, 2014. p.394

3. Рамсундар Бхарат, Истман Питер, Уолтерс Патрик, Панде Виджай. Глубокое обучение в биологии и медицине , 2020. 202 с.

4. Осовский С. Нейронні мережі для обробки інформації. М .: Фінанси і статистика, 2012. 344 с.

5. Information Technology Based on Qualitative Methods in Cyber-Physical Systems of Situational Disaster Risk Management / Grebennik I., Hutsa O., Petrova R., Yelchaninov D., Morozova A. // In: Murayama Y., Velev D., Zlateva P. (eds) Information Technology in Disaster Risk Reduction. ITDRR 2020. IFIP Advances in Information and Communication Technology, vol 622. Springer, Cham. https://doi.org/10.1007/978-3-030-81469-4_11, pp 132-143

6. Doctors are using AI to triage covid-19 patients. [Електронний ресурс]. Режим доступу: https://www.technologyreview.com/2020/04/23/1000410/ai-triage-covid-19-patients-health-care/

7. Chinese Hospitals Deploy AI to Help Diagnose Covid-19. [Електронний ресурс]. Режим доступу: https://www.wired.com/story/chinese-hospitals-deploy-ai-help-diagnose-covid-19/

8. How DAMO Academy’s AI System Detects Coronavirus Cases. [Електронний ресурс]. Режим доступу: https://www.alizila.com/how-damo-academys-ai-system-detects-coronavirus-cases/

9. How this South Korean company created coronavirus test kits in three weeks. [Електронний ресурс]. Режим доступу:

https://edition.cnn.com/2020/03/12/asia/coronavirus-south-korea-testing-intl-hnk/index.html

10. Введение в теорию нейронных сетей // Основные понятия нейросетей. [Електронний ресурс]. Режим доступу: http://www.orc.ru

11. Asuncion A., Newman D. J. Machine Learning Repository. Irvine: University of California, School of Information and Computer Sciences, 2007. p.117

12. Переваги згорткових нейронних мереж для пошуку образу на зображенні / Петрова Р.В., Каменєва К.С., Морозова А.І. // Міжнародна науково-практична конференція "Математичне моделювання процесів в економіці та управлінні проектами і програмами" (ММП-2021), Коблево, 13-17 вересня 2021 р. Збірник праць – Харків : ХНУРЕ, 2021. – С.114-117.

13. Логика мышления. Часть 3. Персептрон, сверточные сети [Електронний ресурс]. Режим доступу до ресурсу: https://habr.com/post/214317/.

14. Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep Learning (Adaptive Computation and Machine Learning series), 2016. p.800

15. Chollet François. Deep Learning with Python, 2017. p.384

16. A. Rosebrock. Histogram of Oriented Gradients and Object Detection. 2015. p.85 [Електронний ресурс]. Режим доступу: http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/

17. Петер Флах, Машинне навчання. Наука і мистецтво побудови алгоритмів, які витягують знання з даних. Видавництво: ДМК Пресс, 2015. - 400 с.

18. Осовский С. Нейронні мережі для обробки інформації. М .: Фінанси і статистика, 2012. 344 с.

19. I.Goodfellow, Y.Bengio, A.Courville. Deep Learning. MIT Press, 2016.

[Електронний ресурс]. Режим доступу: http://www.deeplearningbook.org/

20. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling [Електронний ресурс]. Режим доступу: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

21. Meriam Dhouibi, Ahmed Karim Ben Salem, Slim Ben Saoud. Optimization of CNN model for image classification [Електронний ресурс]. Режим доступу: https://www.researchgate.net/publication/352284493

22. Finetuning torchvision models [Електронний ресурс]. Режим доступу: https://pytorch-org.translate.goog/tutorials/beginner/finetuning

23. Why TensorFlow [Електронний ресурс]. Режим доступу: https://www.tensorflow.org/about

24. Simple. Flexible. Powerful. [Електронний ресурс]. Режим доступу: https://keras.io/

25. End-to-end machine learning framework [Електронний ресурс]. Режим доступу: https://pytorch.org/features/

26. Chest X-ray (Covid-19 & Pneumonia) [Електронний ресурс]. Режим доступу: https://www.kaggle.com/prashant268/chest-xray-covid19-pneumonia