Міністерство освіти і науки України
Харківський національний університет
радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження штучних нейронних мереж в задачі діагностики захворювань за рентгенівськими зображеннями
(тема)

Виконав:
студент 2 курсу, групи СШІм-19-1
Легенька А. В.
(прізвище, ініціали)

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

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник к.т.н., доц. Магдаліна І. В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

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 Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«16» грудня 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Легенькій Анастасії Вікторівні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження штучних нейронних мереж в задачі діагностики захворювань за рентгенівськими зображеннями

затверджена наказом університету від ___ _____ 20 ___ р. № _____
2. Термін подання студентом роботи до екзаменаційної комісії _____ _____ 20 ___ р.
3. Вихідні дані до роботи Науково-технічні публікації, Інтернет-ресурси по темі, наукові конференції, публікації в наукових журналах, збірники конференцій, документація мови програмування Python

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної галузі та постановка задачі, теоретичні дослідження, практичні дослідження

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1 – Розподіл зображень за класами у навчальній вибірці, Рисунок 2 – Розподіл зображень за класами у тестовій вибірці, Рисунок 3 – Розподіл зображень за класами у перевірочній вибірці, Рисунок 4 – Приклади зображень, згенерованих за допомогою ImageGenerator, Рисунок 5 – Розроблена топологія нейронної мережі, Рисунок 6 – Параметри моделі, Рисунок 7 – Вивід бібліотеки Keras в процесі навчання мережі, Рисунок 8 – Графік помилки моделі, Рисунок 9 – Графік точності моделі, Рисунок 10 - Значення точності та помилки на тестових даних, Рисунок 11 – Значення мір якості навчання за класами, Рисунок 12 – Розподіл відповідей мережі за класами, Рисунок 13 – Приклади правильно класифікованих зображень, Рисунок 14 – Приклади неправильно класифікованих зображень

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	к. т. н., доц., Магдаліна І. В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на дипломну роботу	01.11.2020	виконано
2	Аналіз предметної галузі і постановка завдання	01.11.2020 – 05.11.2020	виконано
3	Теоретичні дослідження	06.10.2020 – 13.11.2020	виконано
4	Практичні дослідження	14.11.2020 – 24.11.2020	виконано
5	Оформлення пояснювальної записки	25.11.2020 – 28.11.2020	виконано
6	Оформлення графічних матеріалів	29.11.2020 – 01.12.2020	виконано
7	Попередній захист	11.12.2020	виконано
7	Захист перед ЕК	16.12.2020	виконано

Дата видачі завдання 01 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____ к. т. н., доц. Магдаліна І. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 80 с., 30 рис., 4 табл., 2 дод., 26 джерел.

ГЛИБИННЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, КЛАСИФІКАЦІЯ, СУБДИСКТРЕТИЗАЦІЯ, ШТУЧНА НЕЙРОННА МЕРЕЖА, ШТУЧНИЙ ІНТЕЛЕКТ

Об’єкт дослідження – згорткові нейронні мережі та їх адаптація для задачі діагностики захворювань у пацієнтів за їх рентгенівськими зображеннями, що включає в себе класифікацію рентгенівських зображень за наявністю на них певних ознак захворювання.

Предмет дослідження – адаптація згорткової нейронної мережі для класифікації рентгенівських зображень з використанням мови програмування Python та бібліотек Keras і TensorFlow, а саме підбір гіперпараметрів і топології мережі, навчання і тестування мережі.

Мета роботи – дослідження глибинних нейронних мереж для вирішення задачі діагностики захворювань за рентгенівськими зображеннями.

Методи дослідження – аналіз існуючих алгоритмів і систем, аналіз літератури, вирішення практичних завдань і проведення порівняльного аналізу.

РЕФЕРАТ

Пояснительная записка: 80 с., 30 рис., 4 табл., 2 прил., 26 источников.

ГЛУБОКОЕ ОБУЧЕНИЕ, ИСКУССТВЕННАЯ НЕЙРОННАЯ СЕТЬ, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, КЛАССИФИКАЦИЯ, ПОДВЫБОРКА, СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ

Объект исследования – сверточные нейронные сети и их адаптация для задачи диагностики заболеваний у пациентов по их рентгеновским снимкам, что включает в себя классификацию рентгеновских снимков по наличию на них определенных признаков заболевания.

Предмет исследования – адаптация сверточной нейронной сети для классификации рентгеновских снимков с использованием языка программирования Python и библиотек Keras и TensorFlow, а именно подбор гиперпараметров и топологии сети, обучение и тестирование сети.

Цель работы – исследование глубоких нейронных сетей для решения задачи диагностики заболеваний по рентгеновским снимкам.

Методы исследования – анализ существующих алгоритмов и систем, анализ литературы, решение практических задач и проведение сравнительного анализа.

ABSTRACT

Explanatory note: 80 p., 30 fig., 4 tabl., 2 ann., 26 sources.

ARTIFICIAL INTELLIGENCE, ARTIFICIAL NEURAL NETWORK, CLASSIFICATION, CONVOLUTIONAL NEURAL NETWORK, DEEP LEARNING, SUBSAMPLING

The object of study is convolutional neural networks and their adaptation for detecting patients` diseases from their X-ray images task, which includes the classification of X-ray images by presence of certain signs of disease on them.

The subject of study is adaptation of convolutional neural networks for classifying X-ray images using Python programming language and Keras and TensorFlow libraries, namely the selection of hyperparameters and network topology, training and testing of the network.

The purpose of work is to study deep neural networks to solve the problem of detecting diseases from X-ray images.

Research methods – analysis of existing algorithms and systems, literature analysis, solving practical problems and comparative analysis.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 8
1 Аналіз предметної області та постановка задачі 10
1.1 Аналіз предметної області 10
1.2 Аналіз існуючих систем 13
1.3 Проблеми задачі класифікації рентгенівських зображень 15
1.4 Постановка задачі 17
2 Теоретичні дослідження 19
2.1 Штучні нейронні мережі 19
2.2 Навчання штучних нейронних мереж 32
2.3 Згорткові нейронні мережі 37
2.4 Методи підвищення якості навчання ШНМ 43
2.5 Перенос навчання та попередньо навчені ШНМ 46
3 Експериментальне моделювання та навчання моделі 49
3.1 Вибір програмних засобів 49
3.2 Підготовка набору даних для навчання 52
3.3 Розробка топології мережі 58
3.4 Аналіз результатів 63
Висновки 71
Перелік джерел посилання 73
Додаток А Вихідний код програми 75
Додаток Б Відомість атестаційної роботи магістра 80

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КТ – комп’ютерна томографія;

МРТ – магнітно-резонансна томографія;

САПР – система автоматизованого проектування;

УЗД – ультразвукове дослідження;

ШІ – штучний інтелект;

ШНМ – штучна нейронна мережа;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

GPU – Graphics Processing Unit – графічний процесор;

MSE – Mean Squared Error – середньоквадратична помилка;

SGD – Stochastic gradient descent – стохастичний градієнтний спуск.

ВСТУП

У сучасному світі хвороби легень стають все більш поширеними через зміни навколишнього середовища, зміни клімату, способу життя та інших факторів.

Одним з найпоширеніших захворювань легень є пневмонія – хвороба, для якої характерне запалення легень, зазвичай інфекційного походження. Щороку на пневмонію хворіють близько 450 мільйонів людей: 7 відсотків населення планети; вона також стає причиною близько 4 мільйонів смертей.

Питання своєчасної та якісної діагностики пневмонії стало особливо актуальним у 2020 році з поширенням у світі пандемії COVID-19, оскільки вірусна пневмонія є одним з характерних ускладнень цієї хвороби. 11 березня 2020 року поширення COVID-19 було визнано ВООЗ пандемією – надзвичайно сильною епідемією, що поширилась на території цілих країн та континентів. Станом на 26 листопада 2020 року по даним ВОЗ у світі зареєстровано 62.8 мільйонів випадків захворювання на корона-вірус та 1.46 мільйона летальних випадків [1]. Згідно з аналізом даних по 1099 пацієнтам станом на 28 лютого 2020 року, у близько 91.1% пацієнтів з COVID-19 діагностувалася пневмонія [2].

При вірусній пневмонії легені заповнюються рідиною і не в змозі передавати достатньо кисню в кров, що знижує здатність організму приймати кисень та позбавлятися від вуглекислого газу і може призвести до смерті. Лікарі визнають велику небезпеку у тому, що при зараженні COVID-19 людина може довгий час не відчувати нездужання, її самопочуття не змінюється, але зміни у легенях відбуваються. Тому актуальним є питання швидкої та якісної діагностики пневмонії.

Діагностика пневмонії зазвичай базується на сполученні клінічних ознак та даних рентгенографії грудної клітки або КТ (комп’ютерної томографії). Хоч комп’ютерна томографія є більш ефективним способом діагностики вірусної пневмонії, вона також є більш дорогою і складною та

вимагає дуже високотехнологічного обладнання. Рентген є недорогим методом діагностики, рентгенівське обладнання легке в управлінні, і його мобільність дозволяє проводити обстеження пацієнта прямо в лікарняній палаті, не піддаючи персонал радіографічного відділення додатковій загрозі інфікування. Тому рентгенографія залишається дуже поширеним методом діагностики пневмонії у всьому світі, особливо у країнах, що розвиваються.

Розшифруванням рентгенівських зображень займається рентгенолог – лікар, спеціаліст з променевої діагностики. Висококваліфіковані фахівці-рентгенологи коштують дорого, і на них колосальний попит. Вони відчувають неабиякий тиск, буквально грузнучи в потоках даних, які сиплються на них з усіх боків. Якщо вірити статті [3], такий фахівець повинен видавати діагноз кожні 3-4 секунди. А на практиці таке розшифрування може займати від кількох хвилин до кількох днів при великій кількості пацієнтів. Тим більше, у зв’язку з пандемією, безліч медичних установ по всьому світу зараз перевантажені пацієнтами, відчувають недостачу медичного персоналу та обладнання.

Машинне навчання та глибоке навчання можуть зіграти життєво важливу роль для цього. Штучний інтелект (ШІ) і машинне навчання успішно застосовуються в медицині і вирішують широке коло завдань, поступово перетворюючись з допоміжного інструменту в хороших помічників медичного персоналу.

Саме тому зараз як ніколи актуальною є задача діагностики захворювань за рентгенівськими зображеннями методами машинного навчання, що вирішується у даній роботі. Розроблена в ході роботи система допоможе знизити навантаження на медичних працівників, підвищити швидкість, точність і зручність діагностики бактеріальної та вірусної пневмонії за рентгенівськими знімками та суттєво зменшити витрати системи охорони здоров’я на проведення подібних медичних маніпуляцій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Пневмонія – хвороба, для якої характерне запалення легень, яке відбувається перш за все у повітряних міхурцях, що називаються альвеолами. Зазвичай виникає при інфікуванні бактеріями, значно рідше – вірусами та іншими мікроорганізмами, ураженні деякими медичними препаратами, ураженні легень при автоімунних захворюваннях тощо [4]. Зазвичай при захворюванні спостерігають наступні симптоми: кашель, біль у грудях, гарячка, ускладнене дихання [5].

Пневмонія – це розповсюджена хвороба, яка вражає приблизно 450 мільйонів людей по всьому світу щорічно. Це – найчастіша причина смерті в усіх вікових групах – приблизно 4 мільйони смертей (7 % загальних випадків смертей у світі) щорічно. Найвищі показники смертності спостерігаються серед дітей віком до п'яти років та літніх людей старше 75 років. Показники захворюваності у країнах, що розвиваються, приблизно у п'ять разів вище у порівнянні з розвиненими країнами. На вірусну пневмонію припадає приблизно 200 мільйонів випадків. За даними 2009 року пневмонія займає 8-у позицію серед найчастіших причин смерті у Сполучених Штатах [6].

Хоча Вільям Ослер в XIX столітті назвав пневмонію «водієм ешелону смерті», поява антибіотикотерапії та вакцин у XX столітті значно збільшила кількість людей, що одужали. Тим не менш, в країнах що розвиваються, а також серед людей похилого віку, дітей та тих, хто має хронічні захворювання, пневмонія залишається великою загрозою.

Для діагностики пневмонії можна зробити один із таких тестів: рентген грудної клітки, КТ легень, УЗД грудної клітки, голчаста біопсія легені та МРТ грудної клітки. В даний час рентген грудної клітки є одним з найкращих методів виявлення пневмонії. Рентгенологічне зображення іноді

є кращим, ніж КТ-зображення, оскільки КТ-дослідження зазвичай займає значно більше часу, ніж рентгенологічне, а в багатьох слаборозвинених регіонах може бути недостатньо якісних КТ-сканерів. Натомість, рентген є найпоширенішим і широко доступним методом діагностики і відіграє вирішальну роль у клінічній допомозі та епідеміологічних дослідженнях. Є кілька регіонів світу, де медичних працівників та рентгенологів мало, наприклад Південна Азія та Африка, і там прогнозування пневмонії методом рентгенографії має велике значення.

Рентгенографія – дослідження внутрішньої структури об'єктів, які проектуються за допомогою рентгенівських променів на спеціальну плівку або папір. Найбільш часто термін відноситься до медичного не інвазивного дослідження, заснованого на отриманні сумарного проекційного зображення анатомічних структур організму за допомогою проходження через них рентгенівських променів і реєстрації ступеня ослаблення рентгенівського випромінювання [7].

В Україні найбільш поширеним способом запису рентгенівського зображення є фіксація його на рентген-чутливій плівці з подальшим її проявом. У даний час також існують системи, що забезпечують реєстрацію даних у цифровому вигляді. У більшості розвинених країн цей спосіб вже витіснив аналоговий. В країнах, що розвиваються, у зв'язку з високою вартістю і складністю виготовлення, цифровий спосіб за поширеністю поступається аналоговому.

Знімки робляться в різних положеннях – в основному стоячи, але інколи і лежачи, для немобільних пацієнтів.

Будь-яка патологія, виявлена за допомогою рентгену грудної клітини, відображається на знімку у вигляді відповідної зміни в зображенні органів:

– зміні прозорості легеневих полів;
– посиленні або відсутності легеневого малюнка;
– патологічному затіненні або просвітленні певних ділянок;
– зміні контурів серця, зміні висоти стояння купола діафрагми.

За своєю суттю, рентгенівські знімки є негативами, тому більш світлі ділянки на них називаються «затемненням». Наприклад, здорові легені, заповнені повітрям, на рентгені виглядають чорними. Ділянка запалення при пневмонії – це більш світла пляма, яку лікарі назвуть тінню. Різницю між рентгенівським зображення пацієнта з пневмонією та здорового пацієнта відповідно можна спостерігати на рисунках 1.1, 1.2.

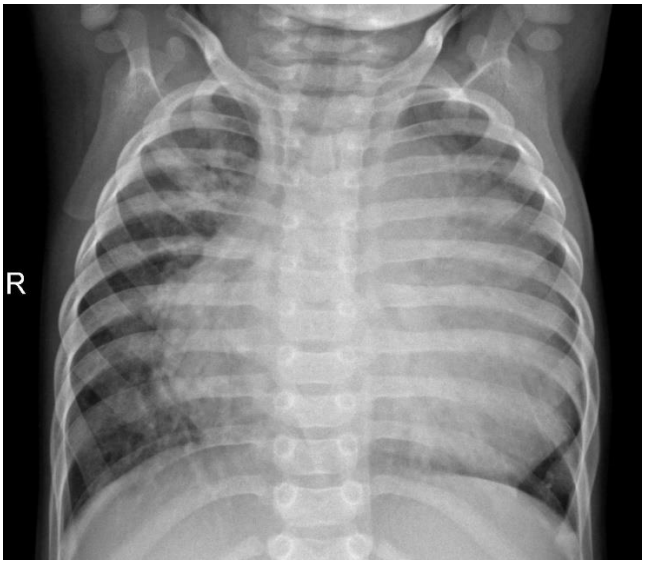

Рисунок 1.1 – Приклад рентгенівського зображення пацієнта з пневмонією

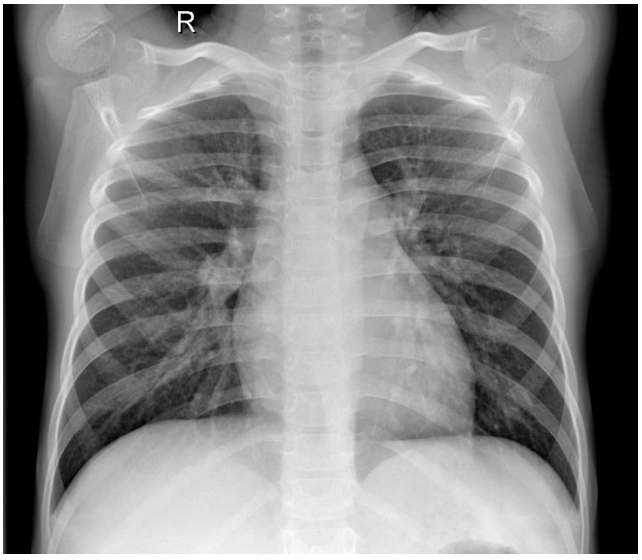

Рисунок 1.2 – Приклад рентгенівського зображення здорового пацієнта

1.2 Аналіз існуючих систем

Незважаючи на запуск першої системи САПР для виявлення вузликів в легенях або уражених клітин легень в кінці 1980-х, цих зусиль було мало. Це пов'язано з тим, що в той час було недостатньо обчислювальних ресурсів для реалізації передових методів обробки зображень. Виявлення захворювань легень з використанням базових методів обробки зображень також вимагає багато часу. Після успішного винаходу GPU і згорткових нейронних мереж (CNN) продуктивність САПР для діагностики захворювань легень значно зросла. Головною перевагою CNN у порівнянні зі своїми попередниками є те, що вони здатні виявляти відповідні особливості без будь-якого людського нагляду.

Плюс перехід радіографії на цифрову візуалізацію означає, що сучасні інструменти аналізу зображень, що раніше використовувалися виключно для більш просунутих методів, тепер можуть застосовуватися і для неї.

На тлі спалаху COVID-19 багато компаній на світовому рівні розробили безліч рішень на основі штучного інтелекту (ШІ) для виявлення COVID-19 при рентгенографічному дослідженні органів грудної клітки. Очевидно, що інструменти глибокого навчання ефективно використовуються для скринінгу легких випадків, сортування нових інфекцій та моніторингу розвитку хвороби. Цей спосіб діагностики може зменшити зростаюче навантаження на рентгенологів, а також витіснити стандартні тести нуклеїнових кислот як основний діагностичний засіб для зараження коронавірусом. Також повідомляється, що тест на мазок потребує ізоляції для проведення процедури тестування, тоді як виявлення на основі рентгенографії грудної клітки може бути легко керованим. Кермані та ін. [8] запропонували модель глибокого навчання на основі рентгенівських зображень для виявлення пневмонії та класифікації інших захворювань з використанням різних наборів медичних даних з точністю тестування 91.80%. В іншому подібному дослідженні Штефан та ін. [9]

проілюстрували ефективний підхід для класифікації пневмонії за допомогою глибокого навчання, використовуючи чотири згорткові шари та два повнозв’язних, а також доповнення даних, і досягли 92.73% точності тестування.

Згодом Нарін та ін. [10] запропонували глибоку згорткову нейромережеву модель автоматичного прогнозування COVID-19 за допомогою попередньо навчених моделей з використанням рентгенівських зображень. У цьому дослідженні автори використовували попередньо навчені моделі ResNet50, InceptionV3 та Inception-ResNetV2, щоб отримати вищу точність прогнозування для підмножини рентгенівських даних.

Згідно дослідження Лондонського коледжу радіології (Royal College of Radiologists), у якому взяли участь 138 лікарів, точність діагностики захворювань легень за рентгенівськими знімками лікарями складає в середньому 82.6% [11]. Можна припустити, що серед звичайних людей, що не є лікарями-рентгенологами і не мають медичної освіти, цей відсоток буде ще нижчим.

Тобто розглянуті системи за своєю точністю навіть перевершують лікарів. Але, не дивлячись на це, навряд чи можна вважати що найближчим часом можна замінити усіх лікарів-рентгенологів штучним інтелектом.

По-перше це пов’язано з рядом регуляторних і сертифікаційних кроків, наприклад необхідністю стандартизувати формат і якість видачі знімків на різноманітному рентгенівському обладнанні.

По-друге, згадаймо теорему Кондорсе (англ. Condorcet Jury Theorem), яка говорить про те, що ймовірність прийняття правильного рішення групою людей вище, ніж кожним з них окремо [12]. Таким чином, якість класифікації спільно лікаря і моделі вище, ніж якість класифікації будь-якого одного з них.

Таким чином, лікар може використовувати модель в ролі порадника. Навіщо? Тому, що у лікаря у самого є свої precision і recall. Нехай, лікарі це так не називають, але у них є помилки. Одні помилки призводять до того,

що частину захворювань пропускають. Таких помилок, напевно, менше, так як лікарі прагнуть мінімізувати помилку першого роду. Інші помилки призводять до того, що людей лікують від пневмонії, якої у них немає, і частина місць в лікарнях зайнята без необхідності.

Таким чином, можна зробити висновок, що побудована в результаті даної роботи модель буде радше використана для визначення пацієнтів, які дійсно потребують невідкладної медичної допомоги, ніж замінить лікаря-рентгенолога. Оскільки, за даними досліджень [13], в 80% випадків у пацієнтів, що звертаються до лікарень, і зокрема роблять рентгенологічне обстеження, насправді немає жодних захворювань. Система штучного інтелекту дозволить швидше виявляти ті самі 20%, яким дійсно необхідна невідкладна допомога.

1.3 Проблеми задачі класифікації рентгенівських зображень

Задача діагностики захворювань за рентгенівськими зображеннями є більш складною, ніж стандартна задача розпізнавання об’єктів на зображеннях. Різниця між об’єктами різних класів в цій задачі є мінімальною і інколи не помітна навіть людському оку. Так, часто звичайна людина не може за рентгенівськими знімками відрізнити випадки пневмонії і норми. Навіть серед лікарів відсоток помилок доволі високий. Як свідчить дослідження Лондонського коледжу радіології (Royal College of Radiologists), у якому взяли участь 138 лікарів, точність діагностики захворювань легень за рентгенівськими знімками лікарями складає в середньому 82.6% [11]. Можна припустити, що серед звичайних людей, що не є лікарями-рентгенологами і не мають медичної освіти, цей відсоток буде ще нижчим.

Таким чином архітектура нейронної мережі для вирішення цієї задачі повинна бути такою, щоб вона могла визначити на зображеннях доволі складні ознаки, інколи не помітні навіть оку звичайної людини. Розробка

такої архітектури мережі потребує комплексних досліджень.

Ще одна із значущих проблем застосування ШІ в медицині – підготовка коректних медичних даних для навчання алгоритмів, так як для цього потрібна велика кількість часу фахівців вузького профілю. Також у багатьох країнах дані з медичних закладів є конфіденційними. Тому у відкритому доступі знаходиться дуже мало наборів даних рентгенівських знімків і більшість з них містять дуже мало даних, враховуючи що для якісного навчання нейронної мережі можуть знадобитись десятки або навіть сотні тисяч зображень. Отже очевидно, що для вирішення задачі діагностики захворювань на рентгенівських зображеннях методами машинного навчання наразі недостатньо навчальних даних і є потреба в зборі даних для цієї задачі. Можливим рішенням бачиться створення об'єднаної платформи зберігання медичних даних, де лікарі зможуть готувати дані для застосування ШІ в своїй спеціальності. Це дозволить в майбутньому підвищити ефективність застосування машинного навчання в медицині завдяки аналізу різнопланових даних з різних джерел.

Таким чином, через недостачу навчальних даних, вирішення задачі діагностики захворювань за рентгенівськими зображеннями потребує використання методів попередньої обробки і доповнення вхідних даних. Ці методи будуть досліджені далі у даній роботі.

Рентгенографія – це технічно простий і найбільш доступний для населення метод попередньої діагностики захворювань легень, в тому числі відстеження захворювання на ранній стадії. Даний тип дослідження широко поширений по всьому світу і, попри наявність інших, більш складних і ефективних способів діагностики, він залишається пріоритетним в медичній практиці навіть в добре оснащених клініках і медичних центрах. Незважаючи на всі переваги даного методу, аналіз і виявлення ознак захворювання по флюорографічних знімках є досить складним завданням, що вимагає участі декількох високо кваліфікованих фахівців. Складність завдання полягає, як правило, в низькій роздільній здатності зображень і

наявності різного роду «шумів» (засвічені зображення, наявність сторонніх предметів і т. д.). Крім шумів, додаткові труднощі вносить наявність на знімках природних анатомічних структур (наприклад, кісток, та інших структур, що перекривають органи), які найчастіше, приховують аномалії, що містяться в легеневій тканині. Тому багатьом спеціалістам для постановки діагнозу потрібно кілька знімків, зазвичай фронтальний і боковий.

Також проблемою може бути наявність в наборі для навчання неякісних, засвічених або навпаки затемнених даних. Тіло пацієнта на знімках може знаходитись в різних положеннях, бути повернутим під різним кутом (особливо при проведенні рентгенографії лежачих та немобільних пацієнтів). Все це ускладнює роботу алгоритмів машинного навчання.

Зважаючи на вище перелічені проблеми можна зробити висновок, що задача діагностики захворювань за рентгенівськими зображеннями є нетривіальною, доволі комплексною і потребує ретельного вивчення і теоретичних та експериментальних досліджень, які і будуть проведені у цій роботі.

1.4 Постановка задачі

Задачею даної роботи є полегшити роботу медичних працівників при діагностиці захворювань за рентгенівськими зображеннями, використовуючи методи машинного навчання, зокрема сконцентруватись на використанні нейронних мереж для вирішення даної задачі.

Можна виділити наступні необхідні кроки при вирішенні завдань машинного навчання за допомогою штучних нейронних мереж:

– визначити завдання, що буде вирішуватись нейронною мережею: класифікація, прогнозування, кластеризація);

– визначити обмеження розв'язуваної задачі (швидкість, точність

відповіді);

– визначити вхідні (тип: зображення, звук, розмір: 100x100, 30x30, формат) і вихідні дані (кількість класів);

– визначити оптимальну архітектуру нейронної мережі для вирішення поставленої задачі.

Розв'язувана даною нейронною мережею задача – класифікація зображень, конкретно класифікація рентгенівських зображень. Накладені обмеження на мережу – це швидкість відповіді – не більше 1 секунди, і точність розпізнавання не менше 90%. Вхідними даними будуть виступати зображення в форматі jpeg, розміру 150x150 пікселів. На виході мають бути два класи, що визначатимуть виявлена пневмонія на представленому знімку, або не виявлена.

В результаті аналізу предметної області, існуючих систем та проблем задачі класифікації рентгенівських зображень був розроблений план реалізації даного завдання. План проектування і розробки даної системи є ітеративним і містить наступні етапи:

– провести теоретичні дослідження і дослідити існуючі методи глибокого машинного навчання в контексті застосування до медичних задач;

– знайти і підготувати набір даних, необхідний для навчання алгоритмів. При цьому дослідити техніки та методи попередньої обробки даних, які можуть підвищити якість навчання алгоритмів;

– проаналізувати і обрати найбільш прийнятні алгоритми для класифікації зображень, а саме сфокусуватись на використанні штучних нейронних мереж для цієї мети;

– провести необхідні експерименти, здійснити експериментальне моделювання та навчання моделі;

– проаналізувати отримані результати;

– провести порівняльний аналіз роботи різних алгоритмів.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Штучні нейронні мережі

Посилений інтерес до штучних нейронних мереж у науковій спільноті виник відносно нещодавно, хоча насправді це один з найстарших алгоритмів машинного навчання.

Термін «нейронна мережа» з'явився в середині XX століття. Перші роботи, в яких були отримані основні результати в даному напрямку, були виконані Мак-Каллоком і Піттсом. У 1943 році ними була розроблена комп'ютерна модель нейронної мережі на основі математичних алгоритмів і теорії діяльності головного мозку. Вони висунули припущення, що нейрони можна спрощено розглядати як пристрої, які оперують двійковими числами, і назвали цю модель «пороговою логікою». Подібно до свого біологічного прототипу нейрони Мак-Каллока-Піттса були здатні навчатися шляхом підстроювання параметрів, що описують синаптичну провідність. Дослідники запропонували конструкцію мережі з електронних нейронів і показали, що подібна мережа може виконувати практично будь-які числові чи логічні операції. Мак-Каллок і Піттс припустили, що така мережа може також навчатися, розпізнавати образи, узагальнювати, тобто має всі риси інтелекту.

Нейронні мережі, описані Мак-Каллоком і Піттсом, мали і порогові значення, і ваги, але не були організовані пошарово, і вчені не задавали ніякого конкретного механізму навчання. Але Мак-Каллок і Піттс показали, що нейронна мережа могла б, в принципі, розрахувати будь-яку функцію, як будь-який цифровий комп'ютер. Результат був більше з області нейробіології, ніж інформатики: потрібно було припустити, що людський мозок можна розглядати як обчислювальний пристрій [14].

Дана модель заклала основи двох різних підходів до досліджень нейронних мереж. Один підхід орієнтований власне на вивчення

біологічних процесів в головному мозку, інший – на застосування нейронних мереж як методу штучного інтелекту для вирішення різних прикладних задач.

У 1949 році канадський фізіолог і психолог Хебб висловив ідеї про характер з'єднання нейронів мозку і їх взаємодії. Він першим припустив, що навчання полягає в першу чергу в змінах сили синаптичних зв'язків, і тим самим запропонував перший алгоритм навчання. У 1954 році в Массачусетському технологічному інституті з використанням комп'ютерів Фарлі і Кларк розробили імітацію мережі Хебба [15].

Перша нейронна мережа, що може навчатися, «перцептрон» (або «персептрон»), була продемонстрована психологом корнельського університету Франком Розенблаттом в 1957 році. Дизайн персептрона був схожий на сучасну нейронну мережу, за винятком того, що у нього був один шар з регульованими вагами і порогами, затиснутий між вхідним і вихідним шарами. У 1958 році Розенблаттом була запропонована модель електронного пристрою, який повинен був імітувати процеси людського мислення, а два роки потому була продемонстрована перша діюча машина, яка могла навчитися розпізнавати деякі з букв, написаних на картках, які підносили до її «очей», що нагадували кінокамери [16].

Персептрони активно досліджувалися в психології і інформатиці до 1959 року, коли Мінський та Паперт опублікували книгу під назвою «Персептрони», яка показала, що проведення цілком звичайних обчислень на персептронах було непрактичним з точки зору витрат часу [17]. Ця робота викликала спад інтересу до дослідження штучних нейронних мереж.

Також на той час комп’ютери не мали достатньої обчислювальної потужності, щоб ефективно обробляти величезний обсяг обчислень, необхідний для глибоких нейронних мереж. Тому інтерес до дослідження штучних нейронних мереж спав і поновився лише коли комп’ютери досягли великих обчислювальних потужностей – з’явилися багатоядрові процесори з великою обчислювальною здатністю.

Також недавній сплеск інтересу до нейронних мереж – революція глибокого навчання – зобов'язаний індустрії комп'ютерних ігор. Складна графічна складова і швидкий темп сучасних відеоігор вимагає апаратного забезпечення, в результаті чого з'явився GPU (графічний процесор) з тисячами відносно простих обробляючих ядр на одному чіпі. Дуже скоро вчені зрозуміли, що архітектура графічного процесора прекрасно підходить обчислень і нейронних мереж.

Таким чином, якщо раніше для навчання нейронної мережі потребувалися дні і навіть місяці, то зараз можна навчити мережу такої самої складності за лічені хвилини.

Сучасні графічні процесори дозволили вибудувати мережі 1960-х років та двох і тришарові мережі 1980-х в букети з 10, 15 і навіть 50 шарів. Ось за що відповідає слово «глибоке» в «глибокому навчанні». В даний час глибоке навчання відповідає за найбільш ефективні системи практично у всіх областях досліджень штучного інтелекту [18].

Інша причина росту популярності нейронних мереж полягає в тому, що зараз відбувся різкий ріст накопленого обсягу даних, і цього обсягу стало достатньо для того, щоб навчити нейронні мережі вирішувати складні задачі. Також велику роль відіграє те, що зараз існує велика кількість безкоштовних і добре працюючих систем глибокого навчання нейронних мереж. Таким чином для практичного використання немає необхідності реалізовувати нейронні мережі і методи їх навчання спочатку, можна взяти готову бібліотеку і швидко застосувати її для вирішення своєї задачі.

Нейронні мережі, з їхньою чудовою здатністю отримувати інформацію із складних або неточних даних, можуть бути використані для вилучення закономірностей та виявлення тенденцій, які є занадто складними, щоб їх могли помітити люди або інші комп'ютерні методи.

Нейронна мережа може наблизити будь-яку функцію з будь-якою заданою точністю.

Нейронні мережі можуть вирішувати наступні задачі:

– класифікація і розпізнавання образів. Як образи можуть виступати різні за своєю природою об'єкти: символи тексту, зображення, зразки звуків і т. д. При навчанні мережі пропонуються різні зразки образів із зазначенням того, до якого класу вони відносяться. Зразок, як правило, представляється як вектор значень ознак. При цьому сукупність усіх ознак повинна однозначно визначати клас, до якого належить зразок. У разі, якщо ознак недостатньо, мережа може співвіднести один і той же зразок з декількома класами, що невірно. Після закінчення навчання мережі їй можна пред'явити невідомі раніше образи і отримати відповідь про належність до певного класу. Топологія такої мережі характеризується тим, що кількість нейронів у вихідному шарі, як правило, дорівнює кількості визначених класів. При цьому встановлюється відповідність між виходом нейронної мережі і класом, який він представляє;

– кластеризація. Під кластеризацією розуміється розбиття множини вхідних сигналів на класи, при тому, що ні кількість, ні ознаки класів заздалегідь не відомі. Після навчання така мережа здатна визначати, до якого класу належить вхідний сигнал. Мережа також може сигналізувати про те, що вхідний сигнал не відноситься ні до одного з виділених класів – це є ознакою нових, відсутніх в навчальній вибірці, даних. Таким чином, подібна мережа може виявляти нові, невідомі раніше класи сигналів;

– прогнозування. Здібності нейронної мережі до прогнозування безпосередньо випливають з її здатності до узагальнення і виділення прихованих залежностей між вхідними та вихідними даними. Після навчання мережа здатна передбачити майбутнє значення якоїсь послідовності на основі декількох попередніх значень і (або) якихось існуючих зараз чинників. Слід зазначити, що прогнозування можливо тільки тоді, коли попередні зміни дійсно в якійсь мірі визначають майбутні.

Нейронні мережі останнім часом використовується активніше, ніж інші методи машинного навчання, тому що в традиційному машинному навчанні необхідно вибрати з великого обсягу наявних даних ті, які

дозволяють вирішити поставлену задачу. Такі дані називаються ознаками. Вибором важливих для різних задач ознак займається людина, і якщо їй вдалося знайти сприятливі ознаки, то задача вирішується, а якщо такі ознаки знайти не вдалося, то метод машинного навчання працюватиме погано. Виграшна особливість нейронних мереж полягає в тому, що вони автоматично відділяють потрібні дані від непотрібних і вміють обирати правильні ознаки. З іншої сторони, так як в нейронній мережі доводиться обробляти набагато більше даних в порівнянні з іншими методами машинного навчання, використання нейронних мереж вимагає великих обчислювальних потужностей [18].

Вирішення певної задачі за допомогою нейронної мережі вимагає виконання наступної послідовності етапів:

– збір даних для навчання;
– підготовка і нормалізація даних;
– вибір топології мережі;
– експериментальний підбір характеристик мережі;
– експериментальний підбір параметрів навчання;
– власне навчання;
– перевірка адекватності навчання;
– коригування параметрів, остаточне навчання;
– вербалізація мережі з метою подальшого використання.

Штучна нейрона мережа (ШНМ) – це математична модель, що представляє собою систему з’єднаних і взаємодіючих між собою простих процесорів (штучних нейронів).

Штучні нейрони використовують біологічну аналогію з нейронами головного мозку людини. Біологічні нейрони влаштовані дуже складно, але спрощено їх основні компоненти можна описати наступним чином. В центрі знаходиться ядро, в якому накопляється електричний заряд. Він поступає від інших нейронів через відростки, які називаються дендритами. Після того, як всередині ядра накопичився певний обсяг заряду, нейрон

спрацьовує і видає електричний сигнал на вихідний відросток, який називається аксоном. Аксон в свою чергу прикріпляється до дендритів інших нейронів через так звані синапси, які можуть змінювати сигнал, що ними передається. Якщо сигнал в синапсі збільшується, то такий синапс називається збуджуючим, а якщо зменшується – гальмуючим [15].

Нейрон – це обчислювальна одиниця, яка отримує інформацію, здійснює над нею прості обчислення і передає далі.

За моделлю Мак-Каллока і Піттса у нейрона є кілька входів (аналог біологічних дендритів), на які подаються вхідні сигнали $x_1,\ x_2, \dots, x_n$ і один вихід a, через який видається вихідний сигнал, по аналогії з аксоном у біологічному нейроні. Кожному входу назначається деяка вага $w_1, w_2, \dots, w_n$ по аналогії з роботою синапсів головного мозку. Велика додатня вага посилює сигнал, як збуджуючий синапс, а від'ємна послаблює вхідний сигнал, як гальмуючий синапс. Вхідні сигнали нейрона, що подаються на кожний вхід, помножуються на відповідну вагу, потім складаються і подаються на вхід деякій нелінійній функції, що називається функцією активації [21]. Функція активації визначає спрацьовує нейрон чи ні. Загальну схему нейрона зображено на рисунку 2.1.

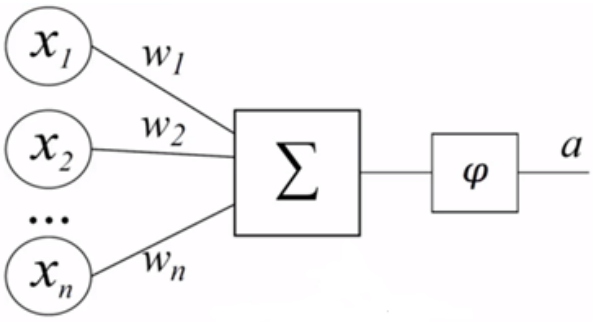

Рисунок 2.1 – Загальна схема нейрона

Вихідний сигнал нейрона обчислюється за формулою 2.1.

$$a = \varphi\left(\sum_{i=1}^{N} w_i\, x_i\right), \tag{2.1}$$

де N – кількість входів нейрона;

x_i – вхідне значення;

w_i – ваговий коефіцієнт на і-му вході;

φ – деяка функція активації.

В моделі Мак-Каллока і Піттса в якості функції активації використовувалась так звана функція Хевісайда, яку ще називають ступінчата або порогова (рисунок 2.2). Згідно з цією функцією, якщо вхідний сигнал менше заданого рівня, то на виході нейрону буде нуль, а після того як сигнал досягнув певного рівня, нейрон видає 1 (формула 2.2).

$$\theta(x) = \begin{cases} 0, x < 0 \\ 1, x > 0 \end{cases}, \tag{2.2}$$

де $\theta(x)$ – вихідний сигнал;

x – вхідний сигнал.

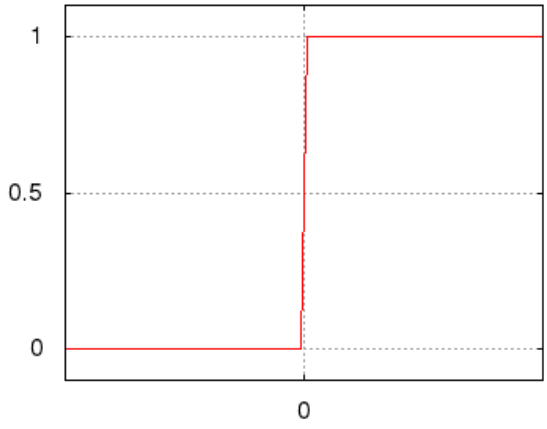

Рисунок 2.2 – Графік порогової функції активації

Також як функції активації застосовують сигмоїдальні функції:

– логістичну функцію (формула 2.3, рисунок 2.3);

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{2.3}$$

де $\sigma(x)$ – вихідний сигнал;

x – вхідний сигнал.

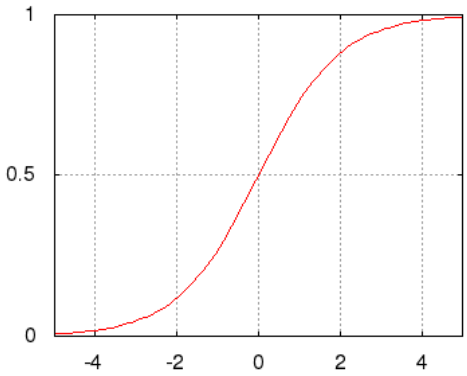

Рисунок 2.3 – Графік логістичної функції активації

– гіперболічний тангенс (формула 2.4, рисунок 2.4);

$$th(x) = \frac{e^{2x}-1}{e^{2x}+1}, \tag{2.4}$$

де $th(x)$ – вихідний сигнал;

x – вхідний сигнал.

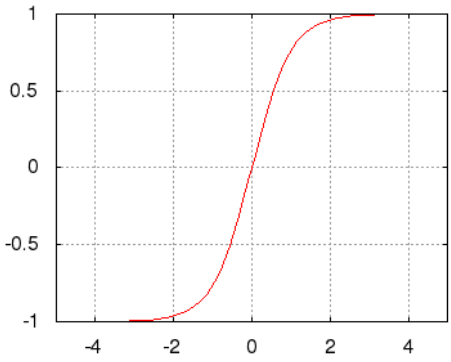

Рисунок 2.4 – Графік функції активації гіперболічний тангенс

Логістична функція звужує діапазон зміни так, що значення виходу лежить між нулем і одиницею, а гіперболічний тангенс дозволяє отримати на виході значення різних знаків (наприклад, від -1 до 1), що може бути корисним для ряду мереж.

Лінійна функція майже ніколи не використовується, за винятком випадків, коли потрібно протестувати нейронну мережу або передати значення без перетворень.

Наразі у нейронних мережах широко використовується функція активації ReLU (rectified linear unit) (рисунок 2.5). ReLU повертає значення x, якщо x додатне, і 0 в протилежному випадку.

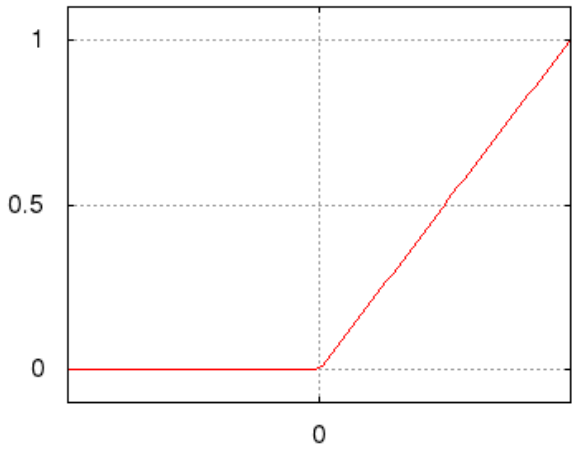

Рисунок 2.5 – Графік функції активації ReLU

Існує кілька її модифікацій, таких як нещільна (Leaky ReLU), параметрична (Parametric ReLU), експоненціально-лінійна (ELU).

ReLU менш вимоглива до обчислювальних ресурсів, ніж гіперболічний тангенс або логістична функція, так як виконує більш прості математичні операції. Тому має сенс використовувати ReLU при створенні глибоких нейронних мереж.

Функція м’якого максимуму (SoftMax), часто використовується в нейронних мережах в якості функції активації при вирішенні задачі

класифікації. Функція SoftMax задається формулою 2.5.

$$\sigma(z)_i = \frac{\mathrm{e}^{z_i}}{\sum_{k=1}^{N} e^{z_k}} \tag{2.5}$$

де z_i – значення на виході з і-го нейрона до активації;

N – загальна кількість нейронів в шарі.

Функцію активації м'якого максимуму зручно застосовувати для задач класифікації, тому що вона дозволяє трактувати вихідні значення нейронів як ймовірність приналежності до даного класу. Це означає, що кожне значення має бути в діапазоні від 0 до 1, і сума всіх значень повинна дорівнювати одиниці. Також функція SoftMax забезпечує те, щоб тільки одне вихідне значення було близьке до одиниці за рахунок застосування експоненти.

Слід зазначити, що немає теоретичної необхідності використовувати саме SoftMax для задач класифікації. При наявності достатнього обсягу даних нейронна мережа може навчитися з іншими функціями активації на останньому шарі. Але з функцією м'якого максимуму навчання відбудеться швидше, тому що дана функція добре підходить саме для класифікації з непересічними класами.

Вибір активаційної функції визначається специфікою поставленого завдання або обмеженнями, що накладаються деякими алгоритмами навчання [19].

Для поєднання штучних нейронів в нейронні мережі вихід одного нейрона подається на вхід іншому нейрону. Нейронна мережа складається з деякої кількості нейронів, об’єднаних в шари (рисунок 2.6). Вхідний шар нейронів отримує сигнали від зовнішнього світу, вихідний – видає сигнали у зовнішній світ, а прихований шар нейронів (таких шарів може бути багато) отримує сигнали від нейронів вхідного шару і видає сигнал на вихідний шар. Прихований шар має таку назву тому що те, що в ньому відбувається, не

видно із зовнішнього середовища.

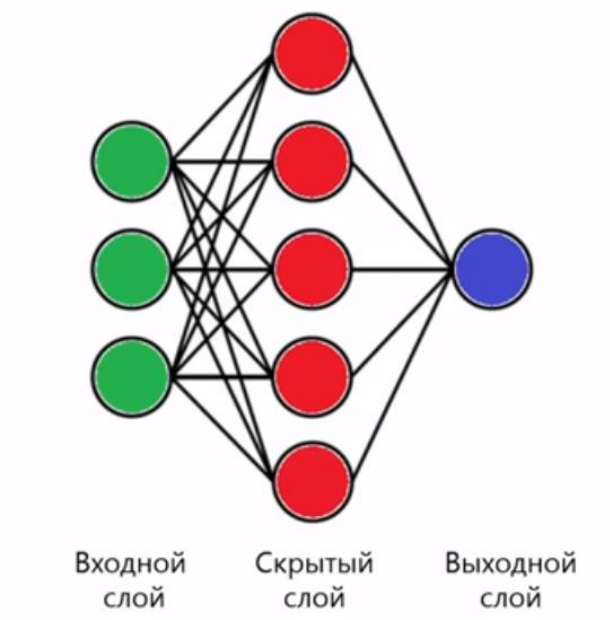

Рисунок 2.6 – Архітектура нейронної мережі

Зміщення – це ваги, додані до прихованих шарів. Вони теж ініціалізуються випадковим чином і оновлюються так само, як і звичайні ваги прихованих шарів. Роль ваги прихованого шару полягає в тому, щоб визначити форму базової функції в даних, в той час як роль зміщенння – змістити знайдену функцію в сторону так, щоб вона частково збіглася з вихідною функцією.

Є два основні типи нейронних мереж: мережі з прямим поширенням сигналу (рисунок 2.7) і рекурентні мережі (рисунок 2.8) [20].

В мережі з прямим поширенням сигналу немає циклів: сигнал, який поступає на вхідний шар передається на прихований шар (їх може бути кілька), далі на вихідний шар і у зовнішнє середовище.

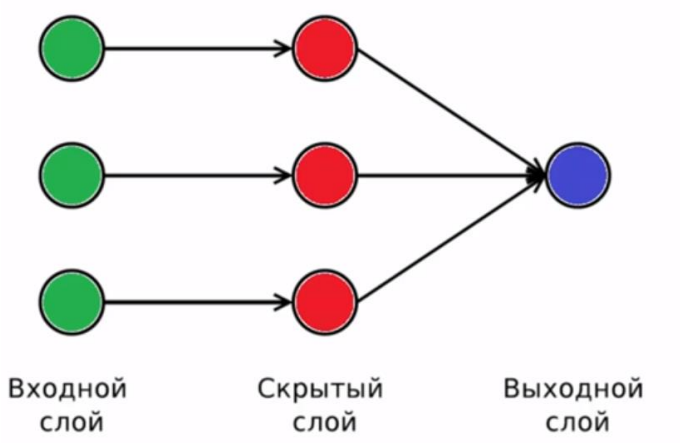

Рисунок 2.7 – Нейронна мережа із прямим поширенням сигналу

В рекурентних мережах можливі цикли. Це означає, що вихідний сигнал від нейрону може поступати на вхід до цього нейрона, інших нейронів цього ж шару або навіть до нейронів попереднього шару.

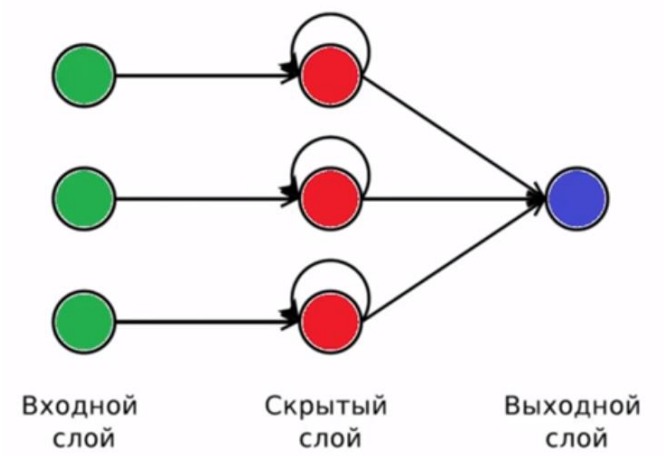

Рисунок 2.8 – Рекурентна нейронна мережа

Нейронні мережі можуть включати в себе кілька прихованих шарів (рисунок 2.9). Виявилося, що такі багатошарові мережі мають більші можливості, ніж одношарові, і в останні роки були розроблені алгоритми для їх навчання. В такому випадку вони називаються глибокими нейронними мережами [18].

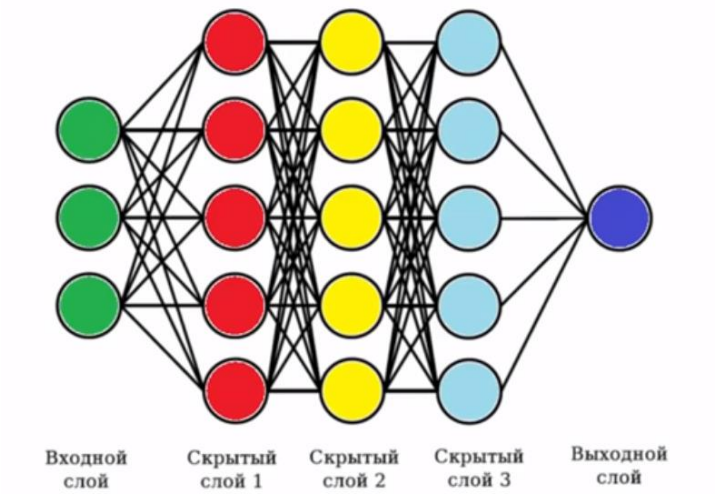

Рисунок 2.9 – Глибока нейронна мережа

Отже, чим більше число прихованих шарів, тим більше можливості навчання мережі. Однак збільшення числа нейронів в одному шарі робить мережу широкою, а не глибокою, і не призводить до глибокого розуміння даних, а призводить до вивчення більшого числа ознак.

Дуже заманливо використовувати глибокі і широкі нейронні мережі для кожного завдання. Але це може бути поганою ідеєю, тому що:

– обидві вимагають значно більшої кількості даних для навчання, щоб досягти мінімальної бажаної точності;

– обидві мають експонентну складність;

– занадто глибока нейронна мережа спробує зламати фундаментальні уявлення, але при цьому вона буде робити хибні припущення і намагатися знайти псевдо-залежності, яких не існує;

– занадто широка нейронна мережа буде намагатися знайти більше ознак, ніж є. Таким чином, подібно до попередньої, вона почне робити неправильні припущення про дані.

Також виділяють повнозв'язні нейронні мережі (коли всі вузли одного шару з'єднані з усіма вузлами суміжних шарів) і неповнозв'язні (коли деякі синаптичні зв'язки відсутні.

2.2 Навчання штучних нейронних мереж

Навчання нейронної мережі – це підбір вагових коефіцієнтів таким чином, щоб мережа вирішувала поставлену задачу. Можливість навчання – одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що в разі успішного навчання мережа зможе повернути вірний результат на основі даних, які були відсутні в навчальній вибірці, а також неповних і/або «зашумлених», частково спотворених даних.

Існують наступні типи навчання нейронних мереж:

– навчання з учителем. Має бути набір даних (сигналів, що подаються на вхід нейронної мережі), для яких заздалегідь відомі правильні відповіді;

– навчання без учителя. Наявні дані, правильні відповіді для яких заздалегідь невідомі. Навчання без учителя ґрунтується на виявленні структурних відмінностей в даних;

– навчання з підкріпленням. Немає заздалегідь підготованих наборів даних, мережа діє як агент у зовнішньому середовищі і отримує сигнали від зовнішнього середовища про те, правильні дії виконуються чи ні.

Першим підхід до навчання нейронних мереж сформулював Хебб у 1949 році. Він використовував біологічні передумови: якщо нейрони головного мозку спрацьовують разом, то їх зв’язки зміцнюються, а якщо вони спрацьовують окремо, то зв’язки між ними послаблюються. Хебб розглядав нейрон, що видає сигнали 0 або 1, і задачу навчання з учителем, де необхідний набір вхідних сигналів, для яких заздалегідь відомі правильні відповіді. Початкові значення вагових коефіцієнтів вхідних сигналів задаються випадковим чином. Вхідні сигнали по черзі подаються на нейрон, перевіряється вихід нейрона і за необхідності змінюються значення вагових

коефіцієнтів. Вагові коефіцієнти змінюються лише якщо нейрон видає неправильний сигнал. Якщо сигнал нейрона неправильний і дорівнює 0, то необхідно збільшити вагу тих входів, на які було подано 1. І навпаки, якщо сигнал нейрона неправильний і дорівнює 1, то вагу тих входів, на які було подано 1, необхідно зменшити [14].

Френк Розенблатт розробив інший підхід до навчання нейронної мережі. Навчання його перспетрона відбувається наступним чином: якщо вихідний сигнал неправильний і дорівнює 0, то значення, які подаються на вхід, додаються до ваги, а якщо перспетрон видає невірний сигнал рівний 1, то значення, які подаються на вхід, віднімаються до ваги [16].

Зараз для навчання нейронних мереж найчастіше використовується метод зворотного поширення помилки [18], [20]. Вихідний сигнал, на відміну від перспетрона Розенблатта та правил Хебба, не бінарне, а дійсне число. Необхідно задати певну міру помилки – на скільки дійсне число, яке видає мережа, відрізняється від правильної відповіді. Можливі різні варіанти міри помилки, найчастіше використовується середньоквадратична.

Середньоквадратична міра помилки (MSE) – це різниця між значенням, виданим мережею, і правильною відповіддю, піднесена до квадрату. Вона повинна бути порахована для всіх об’єктів, що будуть використані для навчання. Середньоквадратична міра помилки розраховується за формулою 2.6.

$$\varepsilon = \frac{1}{2}\sum_{j=1}^{M}\left(a_j - y_j\right)^2, \tag{2.6}$$

де a_j – значення, видане мережею;

y_j – правильна відповідь;

M – кількість об’єктів, використаних для навчання.

Але MSE – це не єдиний варіант функції помилки, для мереж з вихідним шаром Softmax звичайно використовують середню крос-ентропію

по всім об’єктам навчальної вибірки (формула 2.7):

$$E = \frac{1}{k}\sum_{i=1}^{k}(-log(p_i)), \tag{2.7}$$

де k – кількість об’єктів навчальної вибірки;

p_i – видана класифікатором ймовірність приналежності об’єкта до свого класу.

Середня крос-ентропія дозволяє отримати сумарне значення всіх нейронів на виході з шару рівне 1. Це дозволяє трактувати вихід із шару, який використовує таку функцію, як ймовірність.

Навчання полягає у зміні вагових коефіцієнтів таким чином, щоб міра помилки зменшувалася. Для цього використовується метод мінімізації градієнтного спуску. Градієнтний спуск – це спосіб знаходження локального мінімуму або максимуму функції за допомогою руху уздовж градієнта. Необхідно знайти напрямок найшвидшого змінення функції помилки, який називається градієнтом, і рухатись у цьому напрямку. Щоб знайти напрямок градієнта треба взяти частинну похідну по ваговим коефіцієнтам. При зміні вагових коефіцієнтів для нейрона використовується так зване дельта-правило, що для простого лінійного нейрона описується формулою 2.8.

$$w_i = w_i - \dot{\eta}\sum_{j=1}^{M} x_j^i(a_j - y_j), \tag{2.8}$$

де w_i – значення вагового коефіцієнта на попередньому кроці;

$\sum_{j=1}^{M} x_j^i(a_j - y_j)$– частинна похідна помилки відповіді;

η – параметр швидкості навчання.

Якщо використовується нейрон з нелінійною функцією активації, то в дельта-правило додається похідна цієї функції активації.

Параметр швидкості навчання говорить про те, наскільки сильно

змінюються вагові коефіцієнти. Цей параметр має бути достатньо маленьким, щоб не пропустити мінімальне значення функції помилки, але з іншої сторони достатньо великим, щоб навчання проводилось за допустимий час.

Існують три варіанти реалізації навчання:

– повне навчання. Вагові коефіцієнти змінюються після обробки всіх елементів навчальної вибірки. Однак коли даних багато такий підхід є неефективним;

– онлайн навчання (його ще інколи називають стохастичним). Вагові коефіцієнти змінюються після обробки кожного об’єкта;

– міні-вибірки. Вагові коефіцієнти змінюються після обробки 10-100 об’єктів.

Метод зворотного поширення помилки полягає в тому, що по значенню помилки на вихідному шарі розраховується значення помилки на виході прихованого шару, потім на вході наступного прихованого шару і так далі до вхідного шару. Далі використовується дельта-правило для визначення того як потрібно змінити вагові коефіцієнти прихованого шару. Таким чином розраховується помилка від вихідного шару через приховані шари до вхідного шару, і саме тому метод навчання називається зворотне поширення помилки.

Щоб отримати помилку на вході в нейрон, якщо відома помилка на виході з нього, для лінійного нейрона використовується формула 2.9.

$$\varepsilon_i^h = \sum_{m=1}^{M} \varepsilon_i^m w_{hm}\,, \tag{2.9}$$

де ε_i^m– значення помилки на виході з нейрона;

w_{hm} – вага цього виходу.

Вибір даних для навчання мережі та їх обробка є найскладнішим етапом вирішення задачі. Набір даних для навчання повинен задовольняти

декільком критеріям:

– репрезентативність – дані повинні ілюструвати справжній стан речей в предметній області;

– несуперечливість – суперечливі дані в навчальній вибірці призведуть до поганої якості навчання мережі.

Вихідні дані перетворюються до вигляду, в якому їх можна подати на входи мережі. Кожен запис у файлі даних називається навчальною парою або навчальним вектором. Навчальний вектор містить по одному значенню на кожен вхід мережі і, в залежності від типу навчання (з учителем або без), по одному значенню для кожного виходу мережі. Навчання мережі на «сирому» наборі, як правило, не дає якісних результатів. Існує ряд способів поліпшити «сприйняття» мережі. До них відноситься нормування, яке виконується, коли на різні входи подаються дані різної розмірності. Наприклад, на перший вхід мережі подаються величини зі значеннями від нуля до одиниці, а на другий – від ста до тисячі. При відсутності нормування значення на другому вході будуть завжди надавати істотно більший вплив на вихід мережі, ніж значення на першому вході. При нормуванні розмірності всіх вхідних і вихідних даних зводяться воєдино.

В процесі навчання мережа в певному порядку переглядає навчальну вибірку. Мережі, які навчаються з учителем, переглядають вибірку багато разів, при цьому коли нейронна мережа пройшла один навчальний сет, вважається що вона пройшла одну ітерацію. Ітерація – це своєрідний лічильник, який збільшується кожен раз, коли нейронна мережа проходить один тренувальний сет. Іншими словами це загальна кількість тренувальних сетів, пройдених нейронною мережею. Один повний прохід по вибірці називається епохою навчання. При ініціалізації нейронної мережі ця величина встановлюється в 0, а її максимальне значення задається вручну. Чим більше епоха, тим краще натренована мережа, і відповідно краще її результат. Епоха збільшується кожного разу, коли мережа проходить весь набір тренувальних сетів.

При навчанні з учителем набір вихідних даних ділять на дві частини – навчальну вибірку і тестову вибірку; принцип поділу може бути довільним. Навчальні дані подаються мережі для навчання, а тестові використовуються для оцінки якості роботи мережі після завершення навчання (тестові дані ніколи для навчання мережі не застосовуються). Таким чином, якщо на тестових даних помилка зменшується, то мережа дійсно виконує узагальнення. Якщо помилка на навчальних даних продовжує зменшуватися, а помилка на тестових даних збільшується, значить, мережа перестала виконувати узагальнення і просто «запам'ятовує» навчальні дані. Це явище називається перенавчанням мережі або оверфіттінгом. У таких випадках навчання зазвичай припиняють. Перенавчання виникає, коли нейронна мережа адаптується до конкретних особливостей навчальної вибірки, а не до загальних закономірностей в даних. Щоб уникнути перенавчання, не варто довго тренувати ШНМ на одних і тих же або дуже схожих даних. Також, перенавчання може бути викликано неправильним підбором гіперпараметрів або неправильною архітектурою. Гіперпараметри – це значення, які потрібно підбирати вручну. Серед таких значень можна виділити момент і швидкість навчання, кількість прихованих шарів, кількість нейронів у кожному шарі, наявність або відсутність нейронів зміщення, кількість епох навчання.

У процесі навчання можуть проявитися інші проблеми, такі як параліч або потрапляння мережі в локальний мінімум поверхні помилок. Неможливо заздалегідь передбачити прояв тієї чи іншої проблеми, так само як і дати однозначні рекомендації щодо їх вирішення [20].

2.3 Згорткові нейронні мережі

Повнозв’язні нейронні мережі мають недоліки при вирішенні певних задач. Насамперед це велика кількість вагових коефіцієнтів для навчання, особливо при роботі з даними великого розміру. Також при використанні

повнозв’язних нейронних мереж дані представляються у вигляді одномірного вектору або плоского масиву, таким чином втрачається важлива топологічна інформація.

Для того щоб усунути ці недоліки була запропонована інша архітектура нейронних мереж – згорткові нейронні мережі (convolutional neural networks). Вони є глибокими штучними нейронними мережами.

Згорткові нейронні мережі – це прототип зорової кори мозку. Зорова кора має невеликі ділянки клітин, які чутливі до певних областей поля зору. Цю ідею детально розглянули Хьюбел і Візель в 1962 році в своєму експерименті, в якому показали, що окремі мозкові нервові клітини реагували (або активувалися) тільки при візуальному сприйнятті границь певної орієнтації. Наприклад, деякі нейрони активувалися, коли сприймали вертикальні границі, а деякі – горизонтальні або діагональні. Хьюбел і Візель з'ясували, що всі ці нейрони зосереджені в вигляді стрижневої архітектури і разом формують візуальне сприйняття. Цю ідею спеціалізованих компонентів всередині системи, які вирішують конкретні завдання (як клітини зорової кори, які шукають специфічні характеристики) і використовують машини, і ця ідея – основа згорткових нейронних мереж [21].

Згорткові нейронні мережі працюють з даними, що представлені у вигляді двовимірної матриці, тому використовуються в першу чергу для класифікації зображень. В середньому точність таких мереж у задачі розпізнавання зображень перевищує точність звичайних ШНМ на 10-15%. Але є і інші способи застосування згорткових мереж: вони можуть бути застосовані до звуку, який представлений візуально як спектрограма, до аналітики тексту, а також до графічних даних [22].

Згорткові нейронні мережі складаються з шарів згортки (convolution layer) і субдискретизації (subsampling/pooling layer), що чергуються, поєднаних між собою.

Операція згортки полягає в тому, що розглядається деяка

область (фрагмент) вхідних даних, наприклад 3x3, і є деяке ядро згортки – матриця такого ж розміру, як і ділянка даних, що розглядається. Кожен такий фрагмент вхідних даних множиться на матрицю (ядро) згортки поелементно, після цього всі отримані елементи складаються. Кожен нейрон працює зі своїм фрагментом даних, який може частково перетинатися з фрагментом сусіднього нейрона (рисунок 2.10). Далі в нейроні до отриманого значення може застосовуватись певна нелінійна функція активації.

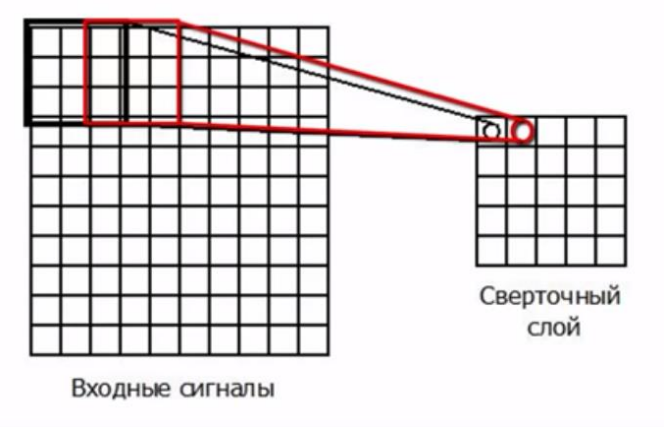

Рисунок 2.10 – Операція згортки

Операція згортки описується формулою 2.10.

$$(f \times g)[m,n] = \sum_{kl} f[m-k, n-l] * g[k,l], \tag{2.10}$$

де f – вхідна матриця зображення;

g – ядро згортки.

Всі нейрони мережі використовують однакові ядра згортки для обробки різних областей вхідних даних (рисунок 2.11). Таким чином певна характеристика шукається на різних ділянках вхідних даних. Якщо деяка шукана характеристика присутня у фрагменті, операція згортки на виході буде видавати число з відносно більшим значенням. Якщо ж характеристика

відсутня, вихідне число буде маленьким. Також за рахунок використання одних і тих же вагових коефіцієнтів у ядрах згортки значно зменшується кількість вагових коефіцієнтів, які необхідно визначити в процесі навчання.

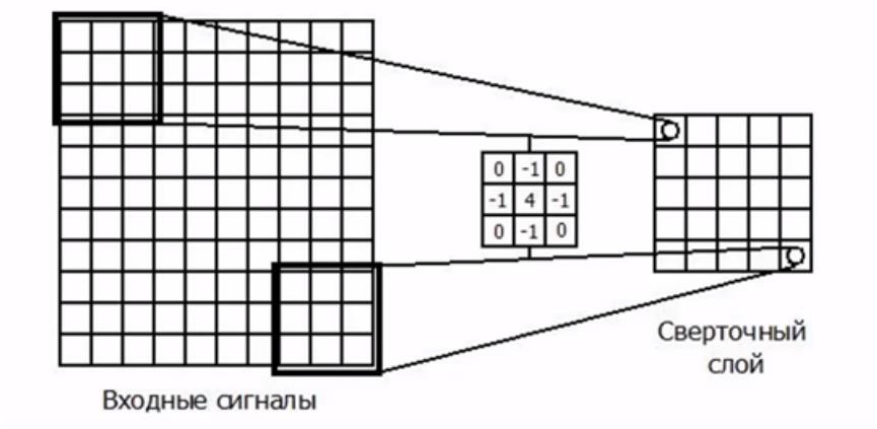

Рисунок 2.11 – Схема роботи шару згортки

Розмір ядра звичайно береться в межах від 3x3 до 7x7. Якщо розмір ядра надто маленький, то воно не зможе виділити які-небудь ознаки у вхідних даних, якщо занадто великий, збільшиться кількість зв’язків між нейронами, що ускладнює навчання [22]. Розмір ядра згортки також залежить від вхідних даних – для даних більшої розмірності використовується більше ядро згортки. Крім того, розмір ядра обирається таким чином, щоб розмір карт згорткового шару був парним, що дозволяє не втрачати інформацію при зменшенні розміру в шарі субдискретизації, робота якого буде описана нижче.

В нейронних мережах ядра згортки визначаються автоматично в процесі навчання – мережа сама визначить потрібні ядра згортки в залежності від того які дані вона обробляє.

Для пошуку різних ознак у вхідних даних може використовуватись декілька ядр. Набір шарів нейронів, кожний з яких використовує різне ядро згортки для пошуку різних ознак на зображенні, називається картою ознак.

Число карт визначається вимогами до задачі. Більш комплексні задачі потребують більшого числа карт ознак. Якщо взяти велику кількість карт – підвищиться якість розпізнавання, але збільшиться обчислювальна складність. Звичайно їх кількість збільшують після кожного шару. Перші шари (з меншою кількістю карт ознак) виділяють прості ознаки (наприклад, межі об'єктів на зображенні), а наступні намагаються виділити більш складні ознаки, базуючись на простих [22], [23].

Розмір у всіх карт згорткового шару однаковий і визначається за формулою 2.11.

$$(w, h) = (mW - kW + 1, mH - kH + 1), \qquad (2.11)$$

де *(w, h)* – розмір згорткової карти;

mW – ширина попередньої карти;

mH – висота попередньої карти;

kW – ширина ядра згортки;

kH – висота ядра згортки.

Далі для зменшення розмірності в згорткових нейронних мережах використовуються шари субдискретизації (subsampling) (рисунок 2.12). Мета шару – зменшення розмірності карт попереднього шару. Якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладні дані вже не потрібні, і вони ущільняються до менш докладних. До того ж фільтрація вже непотрібних деталей допомагає не перенавчатися. Шар субдискретизації, як і згортковий, має карти, але їх кількість співпадає з попереднім (згортковим) шаром. Операція субдискретизації виконує зменшення розмірності сформованих карт ознак. Один нейрон шару субдискретизації підключений до кількох нейронів попереднього шару, як правило до фрагменту 2x2 нейрони, хоча можуть використовуватися і фрагменти інших розмірів. У даній архітектурі мережі вважається, що інформація про факт наявності шуканої ознаки

важливіше точного знання її координат, тому з кількох сусідніх нейронів карти ознак вибирається максимальний і приймається за один нейрон ущільненої карти ознак меншої розмірності. Крім субдискретизації з функцією максимуму можна використовувати і інші функції – наприклад, середнього значення або L2-нормування. Однак практика показала переваги саме субдискретизації з функцією максимуму, яка включається в типові системи [23].

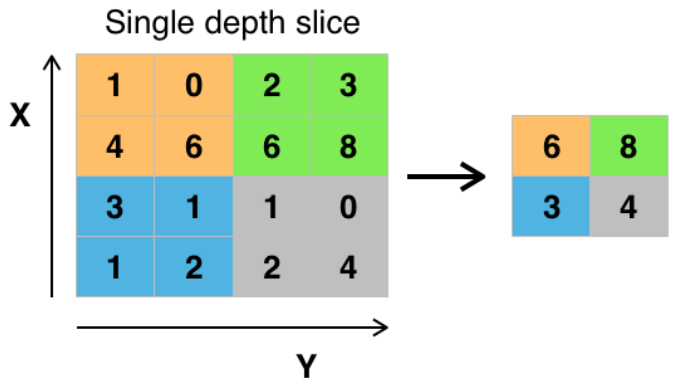

Рисунок 2.12 – Приклад роботи шару субдискретизації

Зазвичай, кожна карта має ядро розміром 2x2, що дозволяє зменшити попередні карти згорткового шару в 2 рази. Великі об’єми вхідних даних можуть виправдовувати використання субдискретизації 4x4, але це може призвести до відкидання великого обсягу інформації. У процесі сканування ядром шару субдискретизації карти попереднього шару, скануюче ядро не перетинається на відміну від згорткового шару.

Після шарів згортки і субдискретизації іде класифікатор, який за ознаками, знайденими згортковою мережею, виконує визначення до якого конкретно класу належить об’єкт. Для початку необхідно перетворити мережу із двовимірного представлення в плоске. В якості класифікатора зазвичай використовується послідовність повнозв’язних шарів. Вихідний шар містить кількість нейронів відповідну кількості класів в задачі, кожний

з яких говорить про приналежність до певного класу.

Робота згорткової нейронної мережі зазвичай інтерпретується як перехід від конкретних особливостей даних до більш абстрактних, і далі до ще більш абстрактних аж до виділення понять високого рівня. При цьому мережа самонастроюється і виробляє сама необхідну ієрархію абстрактних ознак (послідовності карт ознак), фільтруючи незначні деталі і виділяючи істотне. Подібна інтерпретація носить скоріше метафоричний або ілюстративний характер. Фактично «ознаки», що виробляються складною мережею, малозрозумілі і важкі для інтерпретації настільки, що в практичних системах не рекомендується намагатися зрозуміти зміст цих ознак або намагатися їх «підправити», замість цього рекомендується вдосконалити саму структуру і архітектуру мережі, щоб отримати кращі результати. Так, ігнорування системою якихось істотних явищ може говорити про те, що або не вистачає даних для навчання, або структура мережі має недоліки і система не може виробити ефективних ознак для даних явищ [24].

Отже перевагами згорткових ШНМ є те, що враховуються локальні особливості в даних, і їх просте навчання завдяки розділенню вагових коефіцієнтів в ядрах згортки. Для навчання згорткових нейронних мереж застосовується підхід навчання з учителем і алгортим зворотного поширення помилки, але з обмеженням на вагові коефіцієнти, які мають бути однаковими.

2.4 Методи підвищення якості навчання ШНМ

Для запобігання перенавчанню мережі і підвищення якості навчання, окрім правильного підбору гіперпараметрів, хорошою практикою серед розробників вважається використання допоміжних методів, які будуть розглянуті далі.

Для навчання глибоких нейронних мереж необхідний великий набір

розмічених даних – десятки або сотні тисяч елементів. Але, якщо немає в наявності такої великої кількості даних, можна використовувати техніку доповнення або розширення даних (англ. data augmentation), коли на основі тих даних, які є в наявності, генеруються схожі дані.

Доповнення даних при аналізі даних – це метод, що використовується для збільшення обсягу даних шляхом додавання злегка модифікованих копій вже наявних даних або новостворених синтетичних даних із наявних даних, але при цьому не змінюючи мітки класів.

Для доповнення даних можна використовувати геометричні перетворення зображення, перевертання зображення, зміну кольору, обрізку, поворот, збільшення, зменшення, додавання шуму, випадкове стирання, тощо.

Доповнення даних можна виконати за допомогою функції генератора зображень (ImageGenerator), яку надає бібліотека глибокого навчання Keras. При створенні генератора можна вказати як можна змінювати зображення і Keras, при використанні цього генератора, буде випадковим чином обирати кілька варіантів зміни зображення і застосовувати їх.

Доповнення даних може значно збільшити точність навчання нейронної мережі за умови, коли даних для навчання недостатньо, вони незбалансовані або в них нерівномірно представлені різні класи. З огляду на те, що мережа, що навчається, постійно бачить нові, дещо змінені версії вхідних даних, вона може вивчати більш надійні функції.

Під час тестування доповнення даних не застосовується. Навчена мережа оцінюється на основі немодифікованих даних тестування – у більшості випадків можна побачити підвищення точності на тестових даних, можливо, за рахунок незначного зниження точності на навчальних даних.

Також для поліпшення роботи мережі, підвищення її стійкості і запобігання перенавчанню застосовується виключення (Dropout) – метод тренування підмережі з викиданням випадкових одиничних нейронів.

В згорткових нейронних мережах може виникати перенавчання, коли нейрони, що знаходяться поряд, налаштовуються на спільне виділення потрібних ознак. Через це вони налаштовуються на особливості конкретної вибірки, а не на загальні закономірності даних. Техніка Droupout запобігає перенавчанню, що виникає в згортковій нейронній мережі. Для цього в процесі навчання, коли на вхід нейронної мережі подається новий об'єкт, випадковим чином вимикається певна кількість нейронів із заданою ймовірністю. Нейронам, що залишилися, під час навчання доводиться підбирати вагові коефіцієнти таким чином, щоб знаходити важливі ознаки самостійно, без участі сусідніх нейронів [23].

Для прискорення та підвищення стабільності штучних нейронних мереж використовується пакетна нормалізація (англ. batch-normalization). Пакетна нормалізація – це метод, який дозволяє підвищити продуктивність і стабілізувати роботу штучних нейронних мереж. Суть даного методу полягає в тому, що деяким шарам нейронної мережі на вхід подаються дані, попередньо оброблені і мають нульове математичне сподівання і одиничну дисперсію.

Використання пакетної нормалізації має кілька корисних властивостей:

– досягається більш швидка збіжність моделей, незважаючи на виконання додаткових обчислень;

– пакетна нормалізація дозволяє кожному шару мережі навчатися більш незалежно від інших шарів;

– стає можливим використання більш високого темпу навчання, так як пакетна нормалізація гарантує, що виходи вузлів нейронної мережі не матимуть занадто великих або малих значень;

– пакетна нормалізація в якомусь сенсі також є механізмом регуляризації: даний метод привносить в виходи вузлів прихованих шарів деякий шум, аналогічно методу dropout;

– моделі стають менш чутливі до початкової ініціалізації ваг.

2.5 Перенос навчання та попередньо навчені ШНМ

Навчання нейронної мережі може займати довгий час – години або навіть дні. Один з варіантів вирішення цієї проблеми вже було розглянуто – це використання прискорювачів обчислень GPU для скорочення часу навчання. Інший варіант – використання готової нейронної мережі, яка вже була навчена. Такі мережі називаються попередньо навченими. Вони вже були навчені вирішувати поставлену задачу протягом довгого часу. Далі дані про архітектуру такої мережі і вагові коефіцієнти мережі були збережені і зроблені загальнодоступними. Такі дані можна завантажити у власну програму та використовувати для вирішення своїх задач.

В Keras попередньо навчені нейронні мережі містяться в модулі Keras Applications. Модуль Keras Applications містить кілька популярних попередньо навчених нейронних мереж. Більша частина з них використовується для розпізнавання об’єктів на зображеннях.

Як правило, попередньо навчені нейронні мережі створюються компаніями з великими обчислювальними ресурсами (Google, Microsoft і т. д.). Наприклад:

– VGG16 – мережа Visual Geometry Group, для розпізнавання зображень. Це доволі глибока нейронна мережа, містить 16 шарів;

– VGG19 – мережа Visual Geometry Group, для розпізнавання зображень, але більш глибока. Містить 19 шарів;

– Inception v3 – мережа, яку розробила компанія Google для розпізнавання об’єктів на зображеннях. Містить 159 шарів;

– ResNet50 – мережа від компанії Microsoft. В даній мережі використовується залишкове навчання. Містить 50 шарів;

– Xception – модифікація мережі Inception компанії Google від творця Keras Francois Chollet.

Всі вищеперелічені мережі навчалися на наборі даних ImageNet. Це відкритий набор даних, що містить 14 мільйонів зображень об'єктів з

правильними відповідями. Всього – 1000 категорій об'єктів.

Розглянемо, наприклад, архітектуру мережі VGG16, представлену на рисунку 2.13.

Рисунок 2.13 – Архітектура мережі VGG16

Дана мережа складається с двох великих частин – перша частина, що складається з шарів згортки та субдискретизації і відповідає за виділення характерних ознак у зображенні, і друга частина, що складається с трьох повнозв’язних шарів відповідає за класифікацію, тобто визначення який саме об’єкт знаходиться на зображенні, на основі ознак, виділених попередньою згортковою частиною. Згорткова частина складається з п’яти каскадів. Перші два каскади включають в себе два рівня згортки (розмір ядра згортки 3x3) та рівень субдискретизації. Тут використовується вибір максимального значення з квадрату 2x2. Далі йдуть три каскади, в яких використовуються три рівня згортки і рівень субдискретизаціїї. Як і на попередніх шарах, розмір згорткового ядра 3x3, а в якості субдискретизації використовується вибір максимального значення з квадрату 2x2. На останньому шарі 1000 нейронів, які відповідають 1000 класів об’єктів в наборі ImageNet.

Попередньо навчені нейронні мережі можна використовувати для вирішення інших задач, не тільки розпізнавання зображень з набору даних ImageNet. Для цього використовується підхід переносу навчання (transfer learning) – використання попередньо навченої нейронної мережі для

вирішення інших задач. При цьому модифікується архітектура попередньо навченої нейронної мережі. Також можна виконати донавчання (fine tuning) зміненої мережі на основі своїх даних. Такий підхід працює тому, що попередньо навчені нейронні мережі вже навчились розпізнавати велику кількість ознак різних об’єктів. Все, що потрібно зробити, – це навчити таку мережу на основі ознак, які вона вже знає, робити трохи іншу класифікацію. Для цього, як правило, потрібно набагато менше даних і часу, ніж для повного навчання мережі з нуля. Для реалізації переносу навчання береться згорткова частина мережі і до неї додається інша частина, яка відповідає за класифікацію, що підходить для вирішення поставленої задачі. Далі така нейронна мережа навчається на потрібному наборі даних. Таким чином в процесі навчання згорткова частина мережі є «замороженою» – вона не навчається. Навчається лише нова частина мережі, що використовується для класифікації. Перед навчанням вагові коефіцієнти в цій частині ініціалізуються випадковим чином, а далі навчання відбувається методом зворотнього поширення помилки.

Наступний крок, за допомогою якого можна підвищити точність роботи такої складеної мережі – це тонке налаштування (англ. fine tuning), при якому навчається і згорткова частина також, за рахунок чого мережа краще навчається виділяти ознаки тих об’єктів, які входять у потрібний набір даних. Тонке налаштування можна використовувати тільки після того, як класифікатор був навчений на потрібному наборі даних. Можна донавчати згортокову частину повністю або «розморозити» один чи кілька шарів згорткової частини нейронної мережі або навіть декілька блоків (каскадів шарів згортки і зубдискретизації). Кількість шарів чи блоків, які необхідно «розморозити» і донавчити залежить від того, наскільки сильно набор даних задачі відрізняється від набору даних, на якому відбувалося попереднє навчання нейронної мережі.

3 ЕКСПЕРИМЕНТАЛЬНЕ МОДЕЛЮВАННЯ ТА НАВЧАННЯ МОДЕЛІ

3.1 Вибір програмних засобів

Для вирішення поставленої задачі обрано мову програмування Python. Вона широко використовується для машинного навчання, так як писати на цій мові доволі просто і швидко – для реалізації тої ж самої функціональності на C++ або Java знадобилось би набагато більше строк коду. Також Python має простий синтаксис і динамічну типізацію. При використанні нейронних мереж необхідно проводити велику кількість експериментів: визначити архітектуру нейронної мережі, що буде підходити для поставленої задачі, оцінити гіперпараметри навчання і таке інше. Для цього добре підходить мова Python, тому що дозволяє легко модифікувати програму для проведення різноманітних експериментів, і в той же час програма при таких модифікаціях залишається такою, що легко читається і добре підтримується [25].

Та все ж одна з найбільших переваг Python і головна причина чому він використовується для машинного навчання полягає в тому, що у нього є безліч фреймворків і бібліотек, які спрощують процес написання коду і скорочують час на розробку. Основними фреймворками і бібліотеками, які використовуються у машинному навчанні, а також будуть використані у цій роботі є: NumPy, SciPy, Scikit-learn, Matplotlib, Pandas, Tensor Flow та Keras.

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

SciPy – бібліотека для наукових розрахунків, що ґрунтується на NumPy і розширює її можливості. Включає методи лінійної алгебри і методи для роботи з ймовірнісними розподілами, інтегральним обчисленням і

перетвореннями Фур'є.

Scikit-learn – бібліотека, що ґрунтується на NumPy і SciPy. Надає алгоритми для машинного навчання та інтелектуального аналізу даних: кластеризації, регресії і класифікації.

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

Pandas – бібліотека, що надає структури даних і інструменти для аналізу. Підходить для обробки неповних, невпорядкованих і немаркованих даних (як раз такі найчастіше і зустрічаються в житті). Pandas дозволяє замінити досить складні операції з даними на одну-дві команди. Містить багато готових методів групування, фільтрації, об'єднання даних, а також можливість розпізнавання різних типів джерел. У бібліотеці можна об'єднувати таблиці по аналогії з SQL JOIN. При цьому дані беруться прямо з файлів, завдяки чому відпадає необхідність в організації баз даних. Ще одна перевага Pandas – швидкість роботи.

TensorFlow – бібліотека, що використовується для налаштування, тренування і застосування штучних нейронних мереж з численними наборами даних. TensorFlow – бібліотека машинного навчання від компанії Google, яка зараз є одною з провідних компаній, що використовують на практиці машинне навчання. TensorFlow – це бібліотека з відкритими вихідними кодами. Два основних компоненти, які відображені у назві – це робота з багатовимірними матрицями (тензорами), які широко використовуються при навчанні нейронних мереж і розрахунки на графах потоків даних (flow) між якими передаються тензори. Недолік TensorFlow полягає у тому, що замість нейронної мережі необхідно описувати граф потоків даних, який відповідає потрібній нейронній мережі, тобто поверх ефективних обчислень з тензорами самостійно реалізувати нейронну мережу [25].

Тому для розробки програми буде використана ще одна допоміжна бібліотека глибокого навчання – Keras, яка використовує TensorFlow для виконання ефективних обчислень. Особливість бібліотеки Keras полягає в тому, що вона дозволяє на Python описувати нейронну мережу. За допомогою цієї бібліотеки можна вказати з яких шарів буде складатися нейронна мережа, які будуть використовуватись функції активації, метод оптимізації для зменшення помилки та інші параметри важливі для навчання нейронної мережі. Далі бібліотека Keras сама побудує необхідну нейронну мережу і для проведення обчислень буде викликати високоефективні методи з TensorFlow [26].

Традиційні програми на Python працюють повільно, але наявні бібліотеки глибокого навчання використовують різноманітні методи для пришвидшення програм – використовуються оптимізовані математичні бібліотеки, написані на C, деякі бібліотеки на основі коду на Python автоматично генерують код на C і виконується вже цей код, що працює значно швидше. Для ще більшого підвищення продуктивності є можливість використовувати графічні прискорювачі (GPU) за допомогою NVIDIA cuDNN.

Для написання коду було використано інтерактивну оболонку для мови Python – Jupyter Notebook. Вона надає розширену інтроспекцію, додатковий командний синтаксис, доступ до системної оболонки, стикування з pdb відладчиком і Python профайлером, зберігає історію вводу у всіх сеансах, підсвічує і автоматично доповнює код. У Jupyter Notebook можна розробляти, документувати та запускати програми на мові Python, він складається з двох компонентів: веб-додаток, що запускається в браузері, і ноутбуки – файли, в яких можна працювати з вихідним кодом програми, запускати його, вводити і виводити дані і т.д. Веб додаток дозволяє редагувати Python код в браузері, з підсвічуванням синтаксису, автовідступом і автодоповненням, запускати код в браузері, відображати результати обчислень з медіа поданням (схеми, графіки). Ноутбуки – це

файли, в яких зберігаються вихідний код, вхідні і вихідні дані, отримані в рамках сесії. Фактично, він є записом роботи, але при цьому дозволяє заново виконати код, присутній на ньому. Ноутбуки можна експортувати у формати PDF і HTML. Jupyter Notebook входить до складу входить до складу дистрибутиву для мови Python – Anaconda.

Також експериментального моделювання було використано безкоштовну платформу для машинного навчання і нейронних мереж Google Colaboratory. На цій платформі вже встановлено всі необхідні бібліотеки для навчання нейронних мереж, такі як Keras, TensorFlow, PyTorch та інші. На платформі можна використовувати ноутбуки, що дуже схожі на ноутбуки Jupyter. Головний плюс цієї платформи полягає в тому, що навчання нейронних мереж потребує великих обчислювальних ресурсів, а на платформі Google Colaboratory користувачам безкоштовно надається доволі потужний GPU. Так, за допомогою платформи Google Colaboratory вдалося значно збільшити швидкість навчання нейронних мереж.

Для автоматичного підбору гіперпараметрів нейронної мережі біло використано інструмент Keras Tuner. Keras Tuner – це оптимізатор гіперпараметрів, розроблений компанією Google, спеціально для Keras у складі Tensorflow 2.0. Всі гіперпараметри мережі впливають один на одного, і підібрати вручну оптимальну комбінацію гіперпараметрів, при якій використовується максимальна якість роботи нейронної мережі, доволі складно. За допомогою Keras Tuner було підібрано кількість прихованих шарів у мережі, кількість нейронів у кожному шарі, функцію активації та оптимізатор.

3.2 Підготовка набору даних для навчання

Для навчання нейронної мережі було використано набір даних Chest X-ray Images, зібраний з даних пацієнтів лікарями Гуанчжоуського жіночого та дитячого медичного центру, Китай (Guangzhou Women and Children’s

Medical Center, China). Цей набір даних знаходиться у відкритому доступі і містить 5863 файли (загальний розмір: 2Гб). Набір даних містить рентгенівські знімки у форматі JPEG, що розділені на 2 класи (пневмонія / норма). Набір даних складається з 3 каталогів (train, test, val) і містить підкаталоги для кожної категорії зображень (Pneumonia / Normal).

Всі рентгенологічні знімки органів грудної клітки проводились як частина звичайної клінічної допомоги пацієнтам. Авторами набору даних був проведений контроль якості шляхом видалення низькоякісних зображень або таких, що не зчитуються. Також, перш ніж зображення були відібрані для навчання системи ШІ, діагнози для них були оцінені двома досвідченими лікарями. Щоб виключити будь-які помилки в оцінюванні, потім ці дані були також перевірені третім експертом.

Приклад зображення з набору даних Chest X-ray Images наведено на рисунку 3.1. Дане зображення відноситься до класу Пневмонія.

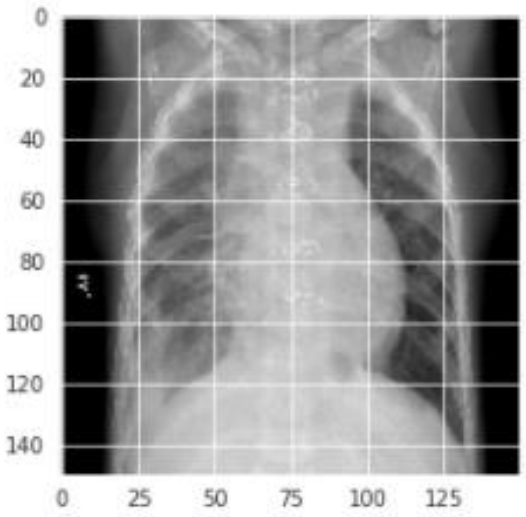

Рисунок 3.1 – Приклад зображення з набору даних Chest X-ray Images

Для навчання нейронної мережі набір даних було розділено на навчальну (training set) та тестову (test set) вибірки у класичному

співвідношенні 80:20. Далі з навчального набору даних було виділено 10% на перевірочну вибірку (validation set) – набір даних, що буде використовуватись в процесі навчання для підбору гіперпараметрів мережі. Таким чином до навчальної вибірки потрапило 4222 зображення, до тестової 1172, а до перевірочної 469 зображень.

Далі набір даних було перемішано, щоб забезпечити рівномірне представлення кожного класу у навчальній, тестовій і перевірочній вибірках.

Видно, що у наборі даних зображення розподілені за класами нерівномірно. Так у навчальну вибірку потрапило 3121 зображення класу Пневмонія та 1001 зображення класу Норма, у тестову вибірку – 804 зображення Пневмонії і 384 Норми і у перевірочну вибірку – 355 і 114 зображень кожного класу відповідно. Таким чином у кожній вибірці співвідношення класів дорівнює приблизно 3:1. Зважаючи на таку нерівномірність даних, а також на їх порівняно невелику кількість, якої є недостатньо для якісного навчання мережі, було застосовано техніку доповнення даних (англ. data augmentation). Розподіл зображень за класами у навчальній, тестовій та перевірочній вибірках наведено на рисунках 3.2, 3.3 та 3.4 відповідно та у таблиці 3.1.

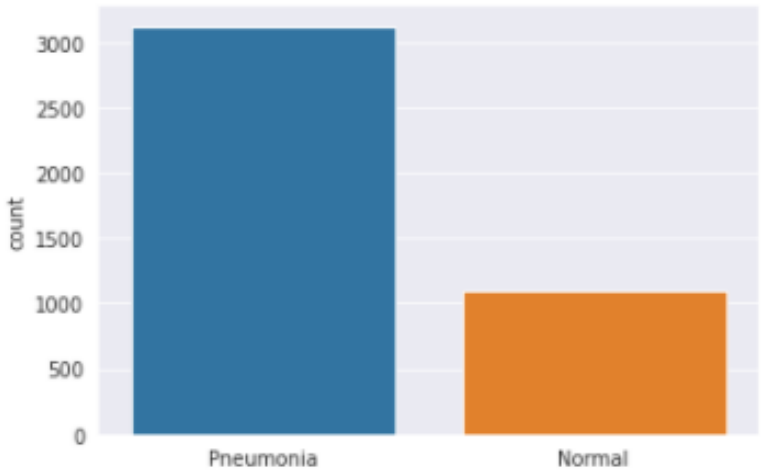

Рисунок 3.2 – Розподіл зображень за класами у навчальній вибірці

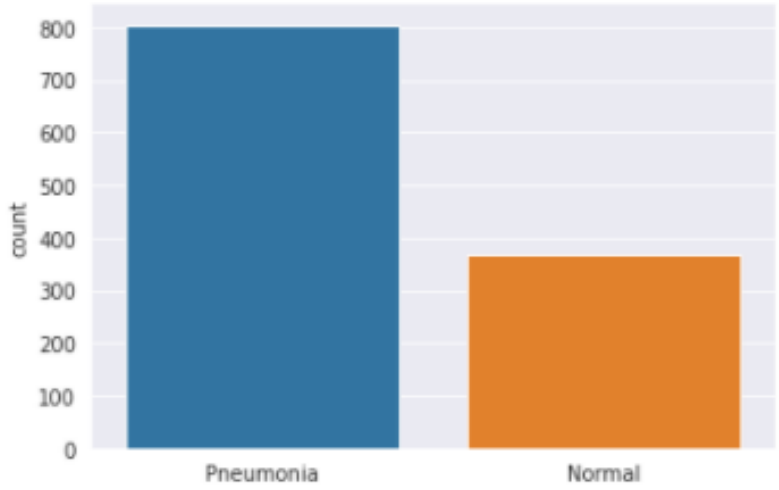

Рисунок 3.3 – Розподіл зображень за класами у тестовій вибірці

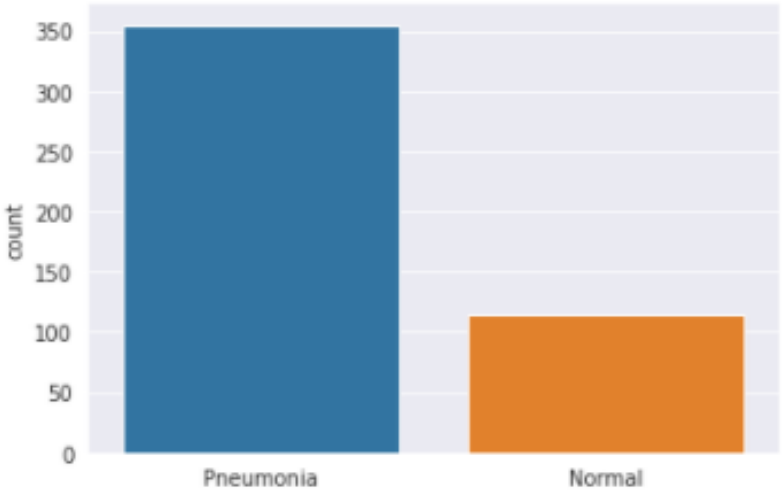

Рисунок 3.4 – Розподіл зображень за класами у перевірочній вибірці

Таблиця 3.1 – Розподіл зображень за класами у вибірках

Категорія	Навчальна вибірка (training set)	Тестова вибірка (test set)	Перевірочна вибірка (validation set)
Норма	1001	384	114
Пневмонія	3121	804	355
Всього	4222	1172	469

Доповнення даних може значно збільшити точність навчання нейронної мережі за умови, коли даних для навчання недостатньо, вони незбалансовані або в них нерівномірно представлені різні класи, що і спостерігається у даному наборі даних. Для доповнення даних було використано функцію генератора зображень (ImageGenerator), яку надає бібліотека глибокого навчання Keras. При налаштуванні генератора було задано наступні можливості:

– поворот у всіх можливих напрямках на кут до 30 градусів;

– зсув по ширині на 10% від загальної ширини зображення або менше;

– зсув по висоті на 10% від загальної висоти зображення або менше;

– збільшення зображення на 20% від його розміру або менше;

– відображення зображення по горизонталі;

– відображення зображення по вертикалі.

Таким чином, при використанні створеного генератора, Keras буде випадковим чином обирати кілька варіантів зміни зображення і застосовувати їх.

Для тестового та перевірочного набору даних доповнення даних не було застосовано, оскільки коректніше оцінювати навчену мережу на основі немодифікованих даних – у більшості випадків можна побачити підвищення точності на тестових даних, можливо, за рахунок незначного зниження точності на навчальних даних.

Приклади зображень, згенерованих за допомогою ImageGenerator на основі зображення з навчальної вибірки (рисунок) наведено на рисунку 3.5.

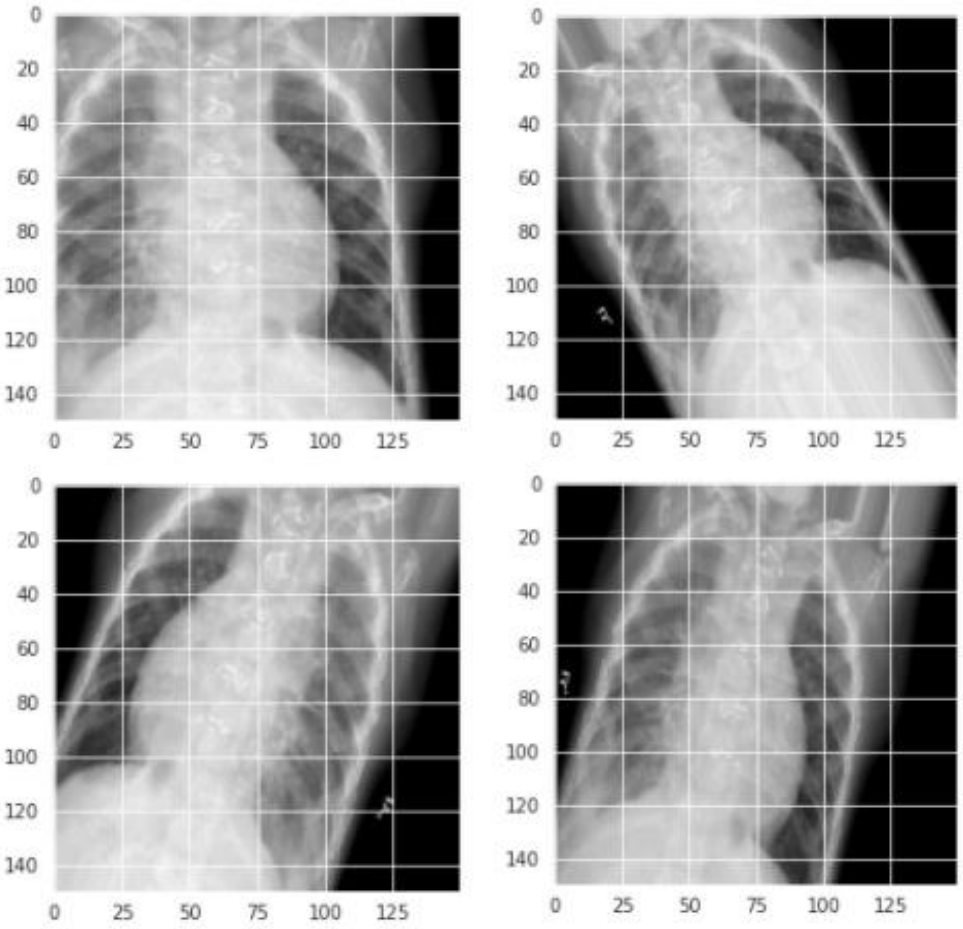

Рисунок 3.5 – Приклади зображень, згенерованих за допомогою ImageGenerator

Перед подачею на вхід нейронної мережі дані необхідно нормалізувати. Оскільки нейронна мережа може приймати на вхід лише вектори однакової розмірності, за допомогою бібліотеки OpenCV всі зображення були приведені до єдиного розміру – 150x150 пікселів. Далі була виконана нормалізація даних про інтенсивність пікселів у зображенні – значення кожного пікселю було розділено на 255, щоб дані на вході у нейронну мережу були в діапазоні від 0 до 1. Це зручно для роботи алгоритмів оптимізації, що використовуються для навчання нейронних мереж.

3.3 Розробка топології мережі

Визначення топології мережі орієнтується на завдання, дані з наукових статей і власний експериментальний досвід.

Стандартна архітектура згорткової нейронної мережі складається з шарів згортки і субдискретизації, що чергуються між собою. Часто зустрічаються архітектури з чергуванням кількох шарів згортки і одного шару субдискретизації. Більша кількість шарів згортки допомагає якісніше виявити важливі ознаки у навчальних даних. Тому для розробленої мережі було обрано архітектуру з чергуванням двох шарів згортки і одного шару субдискретизації. Дослідним шляхом, за допомогою інструмента Keras Tuner, було виявлено, що оптимальним числом таких каскадів з шарів згортки і субдискретизації є чотири. Таким чином згорткова частина запропонованої моделі мережі складається з 8ми шарів згортки і 4х шарів субдискретизації. При такому числі шарів мережа є достатньо глибокою і може виявляти комплексні ознаки в даних, але в той же час її навчання займає менший час.

Для визначення оптимального числа карт ознак у згорткових шарах було використано Keras Tuner. Досліди показали, що з ростом числа карт ознак збільшується точність нейронної мережі, але при великих значеннях це збільшення не таке значне. До того ж чим більше карт ознак – тим більше вагових коефіцієнтів повинна розрахувати нейронна мережа і тим більше часу займає її навчання. Звичайно кількість карт ознак збільшують після кожного шару. Перші шари (з меншою кількістю карт ознак) виділяють прості ознаки (наприклад, межі об’єктів на зображенні), а наступні намагаються виділити більш складні ознаки, базуючись на простих. Тому було обрано таке значення, щоб точність мережі була достатньо високою і були виділені доволі складні ознаки в даних, але в той же час навчання мережі і обробка нею тестових даних займали допустимий час – 32 карти ознак на першому каскаді шарів згортки і субдискретизації, 64 – на другому,

128 і 256 відповідно на третьому і четвертому . Розмір ядра згортки обрано 3x3, оскільки вхідні дані мають середню розмірність.

В якості функції активації обрано ReLU, що традиційно використовується при створенні CNN, оскільки вона менш вимоглива до обчислювальних ресурсів, ніж гіперболічний тангенс або логістична функція, так як виконує більш прості математичні операції.

В якості шару субдискретизації обрано MaxPooling, тобто субдискретизацію з вибором максимального значення і стандартним розміром 2x2, так як вхідні дані невеликого об’єму і використання субдискретизації 4x4 може призвести до відкидання великого обсягу інформації.

Після кожного каскаду шарів згортки і субдискретизації для запобігання перенавчанню додано шар регуляризації – Dropout зі значенням 0.2, що означає, що нейрон буде вимикатися з ймовірністю 20%.

Після чотирьох каскадів згорткових шарів і шарів субдискретизації слідує класифікатор, який за ознаками, знайденими згортковою мережею, визначає до якого конкретно класу належить об’єкт. Для використання класифікатора треба перетворити мережу з двомірного виду в плоский, для цього додано шар Flatten. Далі додано три повнозв'язних шари типу Dense, в першому і другому міститься 512 і 128 нейронів відповідно і використовується функція активації ReLU, у вихідному шарі міститься 1 нейрон, оскільки вирішується задача бінарної класифікації. Оптимальну кількість повнозв’язних шарів та кількість нейронів у них визначено експериментальним шляхом за допомогою інструмента Keras Tuner. На вихідному шарі використовується функція активації Sigmoid, що змінює своє значення від 0 до 1 і тим самим гарно підходить для задачі бінарної класифікації.

Між повнозв’язними шарами також додано шар регуляризації Dropout, який вимикає нейрони з ймовірністю 20%.

Створена мережа є глибокою – вона містить 15 шарів: 8 згорткових

шарів, 4 шари субдискретизації і 3 повнозв’язних шари. Схему розробленої топології подано на рисунку 3.6.

Рисунок 3.6 – Розроблена топологія нейронної мережі

В якості функції помилки використана бінарна крос-ентропія, яка гарно підходить для задачі бінарної класифікації.

В якості метрики при навчанні мережі використовується точність (accuracy) – доля правильних відповідей мережі.

Для вибору оптимального оптимізатора було проведено порівняння точності роботи мережі на тестових даних при використанні трьох найпопулярніших оптимізаторів – SGD, Rmsprop та Adam. Результати експерименту відображено у таблиці 3.2. Видно, що найвищу точність мережа показала при використанні оптимізатора Rmsprop.

Таблиця 3.2 – Точність і помилка мережі на тестових даних в залежності від використаного оптимізатора

Оптимізатор	Точність	Помилка
SGD	80.72%	0.8871
Rmsprop	93.43%	0.1987
Adam	92.66%	0.3123

Розмір міні-вибірок обрано рівним 32, тобто мережа змінює вагові

коефіцієнти після обробки кожних 32 об’єктів.

Дослідним шляхом було визначено оптимальну кількість епох навчання. Було проведено навчання мережі при різній кількості епох – 15, 20, 25, 30, 35. Результати експерименту відображено у таблиці 3.3. Таким чином оптимальна кількість епох навчання склала 30, так як при меншій або більшій кількості епох навчання мережа показувала нижчу точність на тестовій вибірці. При малій кількості епох це зумовлено тим, що мережа не встигла виділити достатню кількість ознак на навчальних даних, а при більшій кількості епох – через початкові ознаки перенавчання.

Таблиця 3.3 – Точність мережі на тестових даних в залежності від кількості епох навчання

Кількість епох навчання	15	20	25	30	35
Точність на тестовій вибірці	88.74%	91.55%	92.24%	93.43%	91.89%

Програмно побудовану за допомогою бібліотеки Keras модель і кількість параметрів на кожному шарі відображено на рисунку 3.7. Таким чином побудована модель містить всього 14 347 681 параметрів.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 150, 150, 32)      896
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 150, 150, 32)      9248
_________________________________________________________________
batch_normalization_4 (Batch (None, 150, 150, 32)      128
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 75, 75, 32)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 75, 75, 64)        18496
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 75, 75, 64)        36928
_________________________________________________________________
batch_normalization_5 (Batch (None, 75, 75, 64)        256
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 38, 38, 64)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 38, 38, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 38, 38, 128)       73856
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 38, 38, 128)       147584
_________________________________________________________________
batch_normalization_6 (Batch (None, 38, 38, 128)       512
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 19, 19, 128)       0
_________________________________________________________________
dropout_8 (Dropout)          (None, 19, 19, 128)       0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 19, 19, 256)       295168
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 19, 19, 256)       590080
_________________________________________________________________
batch_normalization_7 (Batch (None, 19, 19, 256)       1024
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 10, 10, 256)       0
_________________________________________________________________
dropout_9 (Dropout)          (None, 10, 10, 256)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25600)             0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               13107712
_________________________________________________________________
dropout_10 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_11 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 129
=================================================================
Total params: 14,347,681
Trainable params: 14,346,721
Non-trainable params: 960
_________________________________________________________________
```

Рисунок 3.7 – Параметри моделі

3.4 Аналіз результатів

Готова підсистема була реалізована за допомогою мови програмування Python і біблотек машинного навчання TensorFlow та Keras. Вона дозволяє завантажити рентгенівське зображення, що буде класифіковане нейронною мережею і з певною ймовірністю віднесене до одного з двох класів – Пневмонія або Норма.

Оскільки мережа є глибокою (має 15 прихованих шарів), на її повне навчання (впродовж 30 епох) на машині без GPU знадобилося 1136 секунд (майже 20 хвилин) на кожну епоху і більше 9.5 годин на повне навчання мережі. При навчанні на GPU за допомогою платформи Google Colaboratory на кожну епоху було витрачено 26 секунд і 13 хвилин на повне навчання мережі. Такого результату було досягнуто за рахунок невеликого обсягу вхідних даних, а також використання згорткових нейронних мереж, що дозволяють зменшити кількість вагових коефіцієнтів, що мають бути обчислені на кожній епосі і відповідно скоротити час навчання.

В кінці навчання мережі точність на даних, на яких проводилося навчання, склала 95.57%, а на перевірочній вибірці (validation data) – 95.95%. Значення функції втрат (loss) відповідно склали 0.1353 на навчальній вибірці 0.1558 на перевірочній вибірці (validation loss) (рисунок 3.8).

```
Epoch 1/30
132/132 [==============================] - 26s 200ms/step - loss: 1.5610 - accuracy: 0.7752 - val_loss: 4.4258 - val_accuracy: 0.7569
Epoch 2/30
132/132 [==============================] - 26s 193ms/step - loss: 0.3990 - accuracy: 0.8411 - val_loss: 1.5706 - val_accuracy: 0.7569
Epoch 3/30
132/132 [==============================] - 26s 193ms/step - loss: 0.3510 - accuracy: 0.8742 - val_loss: 6.8769 - val_accuracy: 0.7655
Epoch 4/30
132/132 [==============================] - 25s 193ms/step - loss: 0.3385 - accuracy: 0.8868 - val_loss: 0.5616 - val_accuracy: 0.7569
Epoch 5/30
132/132 [==============================] - 26s 195ms/step - loss: 0.2753 - accuracy: 0.9005 - val_loss: 1.0836 - val_accuracy: 0.8316
Epoch 6/30
132/132 [==============================] - 26s 193ms/step - loss: 0.2699 - accuracy: 0.9086 - val_loss: 0.4880 - val_accuracy: 0.7591
Epoch 7/30
132/132 [==============================] - 25s 193ms/step - loss: 0.3041 - accuracy: 0.9088 - val_loss: 0.5063 - val_accuracy: 0.8742
Epoch 8/30
132/132 [==============================] - 26s 194ms/step - loss: 0.2337 - accuracy: 0.9138 - val_loss: 1.7783 - val_accuracy: 0.7868
Epoch 9/30
132/132 [==============================] - ETA: 0s - loss: 0.2684 - accuracy: 0.9105
Epoch 00009: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
132/132 [==============================] - 26s 194ms/step - loss: 0.2684 - accuracy: 0.9105 - val_loss: 0.3961 - val_accuracy: 0.8401
Epoch 10/30
132/132 [==============================] - 26s 196ms/step - loss: 0.1808 - accuracy: 0.9389 - val_loss: 1.5576 - val_accuracy: 0.7569
Epoch 11/30
132/132 [==============================] - ETA: 0s - loss: 0.1801 - accuracy: 0.9441
Epoch 00011: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
132/132 [==============================] - 26s 196ms/step - loss: 0.1801 - accuracy: 0.9441 - val_loss: 76.6414 - val_accuracy: 0.7676
Epoch 12/30
132/132 [==============================] - 26s 196ms/step - loss: 0.1871 - accuracy: 0.9479 - val_loss: 1.5775 - val_accuracy: 0.9275
Epoch 13/30
132/132 [==============================] - 26s 195ms/step - loss: 0.1765 - accuracy: 0.9505 - val_loss: 0.1586 - val_accuracy: 0.9446
Epoch 14/30
132/132 [==============================] - 26s 196ms/step - loss: 0.1473 - accuracy: 0.9517 - val_loss: 0.1013 - val_accuracy: 0.9595
Epoch 15/30
132/132 [==============================] - 26s 195ms/step - loss: 0.1333 - accuracy: 0.9543 - val_loss: 1.3486 - val_accuracy: 0.9147
Epoch 16/30
132/132 [==============================] - ETA: 0s - loss: 0.1313 - accuracy: 0.9538
Epoch 00016: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
132/132 [==============================] - 26s 197ms/step - loss: 0.1313 - accuracy: 0.9538 - val_loss: 1.2234 - val_accuracy: 0.9275
Epoch 17/30
132/132 [==============================] - 26s 195ms/step - loss: 0.1373 - accuracy: 0.9545 - val_loss: 0.1461 - val_accuracy: 0.9467
Epoch 18/30
132/132 [==============================] - ETA: 0s - loss: 0.1353 - accuracy: 0.9505
Epoch 00018: ReduceLROnPlateau reducing learning rate to 8.100000013655517e-06.
132/132 [==============================] - 26s 197ms/step - loss: 0.1353 - accuracy: 0.9505 - val_loss: 0.1365 - val_accuracy: 0.9446
Epoch 19/30
132/132 [==============================] - 26s 197ms/step - loss: 0.1381 - accuracy: 0.9557 - val_loss: 0.1361 - val_accuracy: 0.9467
Epoch 20/30
132/132 [==============================] - ETA: 0s - loss: 0.1249 - accuracy: 0.9512
Epoch 00020: ReduceLROnPlateau reducing learning rate to 2.429999949526973e-06.
132/132 [==============================] - 26s 195ms/step - loss: 0.1249 - accuracy: 0.9512 - val_loss: 0.1707 - val_accuracy: 0.9531
Epoch 21/30
132/132 [==============================] - 26s 196ms/step - loss: 0.1409 - accuracy: 0.9522 - val_loss: 0.1566 - val_accuracy: 0.9488
Epoch 22/30
132/132 [==============================] - ETA: 0s - loss: 0.1396 - accuracy: 0.9564
Epoch 00022: ReduceLROnPlateau reducing learning rate to 1e-06.
132/132 [==============================] - 26s 196ms/step - loss: 0.1396 - accuracy: 0.9564 - val_loss: 0.1872 - val_accuracy: 0.9552
Epoch 23/30
132/132 [==============================] - 26s 196ms/step - loss: 0.1356 - accuracy: 0.9548 - val_loss: 0.1164 - val_accuracy: 0.9616
Epoch 24/30
132/132 [==============================] - 26s 199ms/step - loss: 0.1446 - accuracy: 0.9533 - val_loss: 0.1280 - val_accuracy: 0.9510
Epoch 25/30
132/132 [==============================] - 27s 203ms/step - loss: 0.1491 - accuracy: 0.9557 - val_loss: 0.1648 - val_accuracy: 0.9467
Epoch 26/30
132/132 [==============================] - 26s 199ms/step - loss: 0.1202 - accuracy: 0.9567 - val_loss: 0.1160 - val_accuracy: 0.9574
Epoch 27/30
132/132 [==============================] - 26s 198ms/step - loss: 0.1370 - accuracy: 0.9529 - val_loss: 0.1402 - val_accuracy: 0.9595
Epoch 28/30
132/132 [==============================] - 26s 198ms/step - loss: 0.1467 - accuracy: 0.9552 - val_loss: 0.1399 - val_accuracy: 0.9510
Epoch 29/30
132/132 [==============================] - 26s 196ms/step - loss: 0.1444 - accuracy: 0.9545 - val_loss: 0.1783 - val_accuracy: 0.9382
Epoch 30/30
132/132 [==============================] - 26s 197ms/step - loss: 0.1353 - accuracy: 0.9557 - val_loss: 0.1558 - val_accuracy: 0.9595
```

Рисунок 3.8 – Вивід бібліотеки Keras в процесі навчання мережі

Було побудовано графіки зміни помилки (рисунок 3.9) і точності мережі в процесі навчання (рисунок 3.10). На даних графіках видно, що під час навчання мережі на кожній епосі помилка на наборі даних для навчання і наборі даних для перевірки зменшується, а якість навчання збільшується, тобто перенавчання не виникло. Якщо спочатку, на перших епохах навчання

помилка знижується як на даних для навчання, так і на перевірочних даних, то ближче до кінця навчання значення помилки зменшується лише на даних для навчання, а на наборі даних для перевірки значення помилки майже не змінюється. Також на наборі даних для перевірки майже не збільшується якість навчання. Це свідчить про те, що якщо продовжити навчати мережу, її узагальнююча здатність знизиться – вона буде гірше розпізнавати нові зображення, які не бачила в процесі навчання. Тому навчання мережі було припинене, щоб уникнути перенавчання.

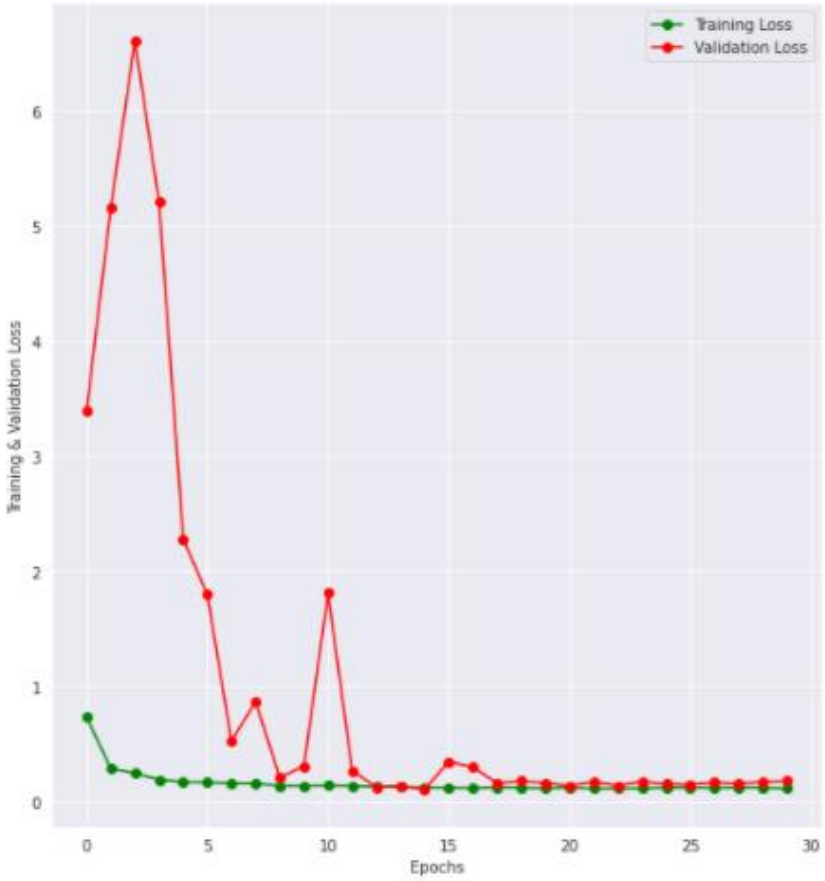

Рисунок 3.9 – Графік помилки моделі

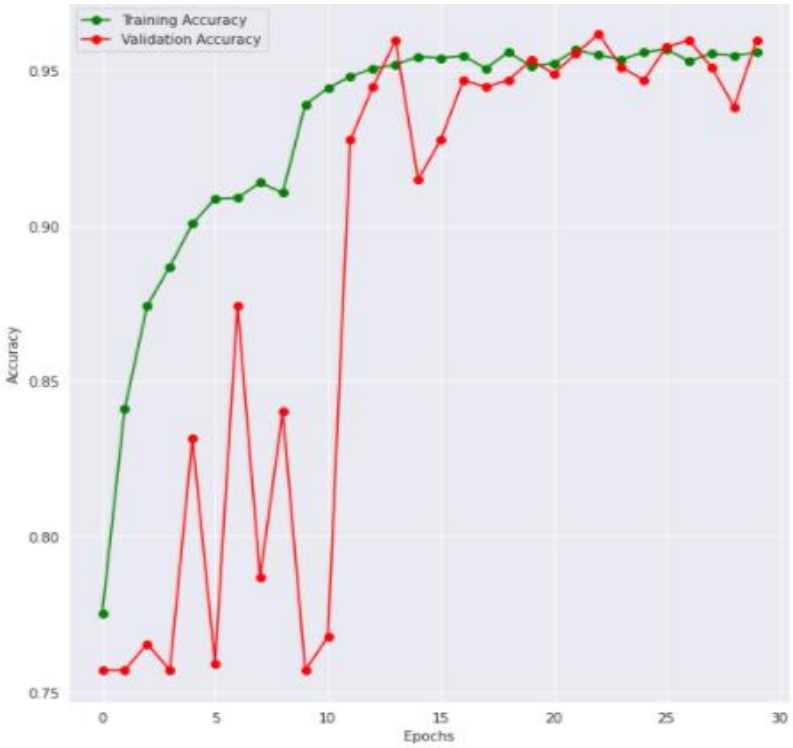

Рисунок 3.10 – Графік точності моделі

Після завершення навчання була перевірена якість роботи мережі на тестових даних, які мережа не бачила в процесі навчання (рисунок 3.11). Ці дані були попередньо відібрані з навчального набору і включають в себе 1172 зображення. Для даної конфігурації мережі і обраних параметрів навчання точність склала 93.43%, що менше ніж на тестовій і навчальній вибірці. Але ця точність є вищою ніж точність аналогічних систем на 0.7% – найточніший з досліджених аналогів показує точність 92.73%. Також вона є вищою, ніж точність розпізнавання пневмонії на рентгенівських знімках звичайною людиною та навіть лікарями-радіологами, що за даними досліджень складає 82.6% [34]. Помилка на тестових даних склала 0.1987, що є не набагато більше ніж на навчальних та перевірочних даних. Отже показані системою показники точності та помилки можна вважати хорошими.

```
37/37 [==============================] - 1s 30ms/step - loss: 0.1987 - accuracy: 0.9343
Loss of the model is -  0.19866324961185455
37/37 [==============================] - 1s 23ms/step - loss: 0.1987 - accuracy: 0.9343
Accuracy of the model is -  93.43003630638123 %
```

Рисунок 3.11 – Значення точності та помилки на тестових даних

Було порівняно якість роботи мережі на тестових даних за умови використання техніки доповнення даних (англ. data augmentation) та без її використання. Доповнення застосовувалось лише для навчання мережі, так як навчену мережу прийнято оцінювати на основі немодифікованих даних тестування. Без використання доповнення даних мережі знадобилось менше часу на навчання – лише 10 епох і 11 секунд на кожну епоху, але точність на тестових склала всього 84.64%, а помилка аж 4.3695. Таким чином було встановлено, що використання доповнення даних для даного набору даних дало виграш у точності у 9.21% і у 22 рази меншу помилку, але навчання зайняло на понад 11 хвилин більше часу.

Також було оцінено інші характеристики якості навчання мережі, зокрема точність (precision), відгук (recall) та f-міру (f-measure). Значення цих мір за класами наведено на рисунку 3.12. Видно, що модель краще розпізнає зображення пневмонії, тому що цього класу було представлено більше у навчальній вибірці. Але в цілому значення цих метрик можна оцінити як високе.

	precision	recall	f1-score	support
Pneumonia (Class 0)	0.94	0.97	0.95	804
Normal (Class 1)	0.92	0.86	0.89	368
accuracy			0.93	1172
macro avg	0.93	0.91	0.92	1172
weighted avg	0.93	0.93	0.93	1172

Рисунок 3.12 – Значення мір якості навчання за класами

Кількість правильно і неправильно класифікованих зображень під час тестування мережі за класами можна побачити на рисунку 3.13. На рисунку 3.14 приведено кілька зображень, які мережа вірно віднесла до класу Норма, а на рисунку 3.15 – зображення класу Норма, які мережа невірно класифікувала як Пневмонію.

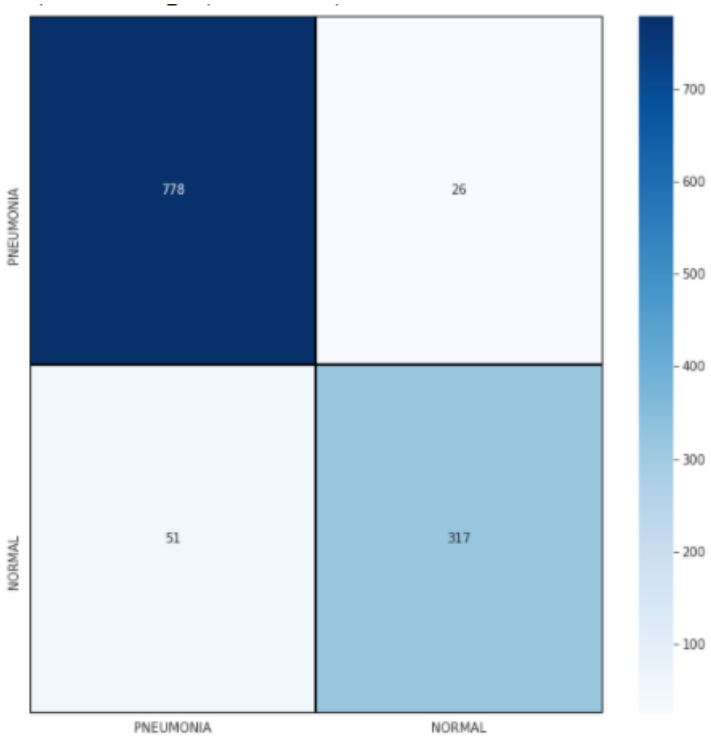

Рисунок 3.13 – Розподіл відповідей мережі за класами

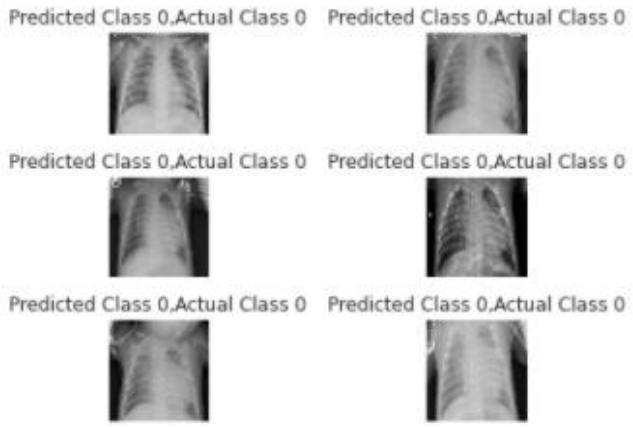

Рисунок 3.14 – Приклади правильно класифікованих зображень

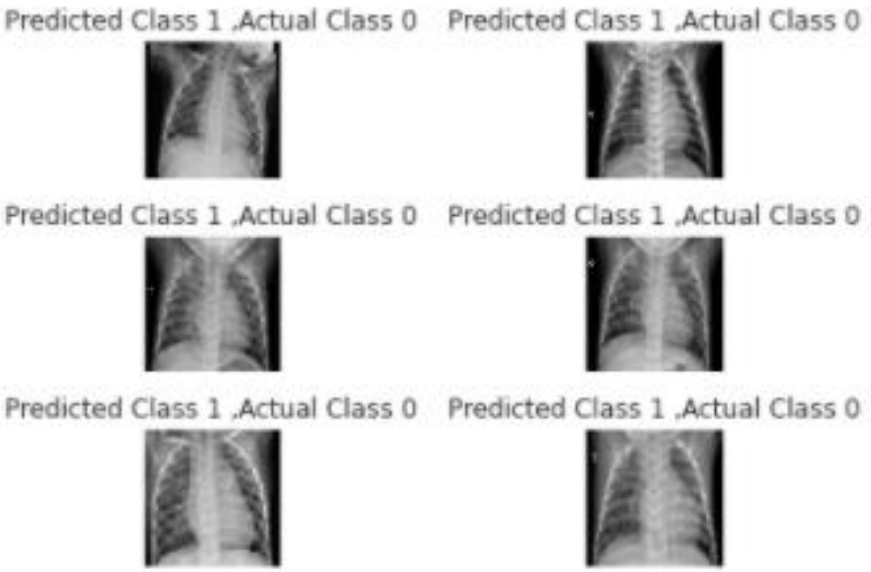

Рисунок 3.15 – Приклади неправильно класифікованих зображень

Також було проведено порівняння точності роботи на тестових даних побудованої моделі з попередньо навченими мережами, щоб встановити доцільність побудови власної моделі. Для цього було обрано наступні найбільш популярні мережі з модуля Keras Applications, що використовуються для розпізнавання об’єктів на зображеннях: VGG16, VGG19 та Inception V3. Ці мережі було донавчено на навчальних даних з набору Chest X-ray Images. Для цього додано класифікатор з трьох шарів – 512 нейронів на першому, 128 на другому і 1 на останньому (так як вирішується задача бінарної класифікації). Навчання проводилось впродовж 15 епох. Функція активації на перших двох шарах класифікатора – ReLU, на останньому – sigmoid. Оптимізатор – Rmsprop, функція втрат – бінарна крос-ентропія. Між повнозв’язними шарами також додано шар регуляризації Dropout, який вимикає нейрони з ймовірністю 20%. Для донавчання цих мереж так само використовувалось доповнення даних.

Порівняння точності роботи і помилки на тестових даних донавчених мереж з авторською CNN, розробленою у цій роботі, наведено у таблиці 3.4.

Таблиця 3.4 – Порівняння точності різних архітектур CNN при вирішенні задачі класифікації рентгенівських зображень

Архітектура мережі	Точність (accuracy), %	Помилка (loss)
VGG16	92.52	0.5670
VGG19	91.13	0.7121
Inception V3	90.19	0.9538
Авторська CNN	93.43	0.1987

Видно, що авторська CNN демонструє вищу точність і меншу помилку. Це може бути зумовлено тим, що зображення з набору даних Chest X-ray Images значно відрізняються від зображень з набору даних ImageNet. Точніше, згорткові частини попередньо навчених мереж виділяють ознаки, які підходять для зображень ImageNet, але не підходять для зображень з набору даних Chest X-ray Images, використаного у даній роботі. Також зі зростанням кількості прихованих шарів у попередньо навчених нейронних мережах знижується їх якість роботи при вирішенні даної задачі. Так найкращу точність і найнижчу помилку показала мережа VGG16 з 16 шарами. Її показники можна порівняти з показниками розробленої у даній роботі моделі CNN. А от мережі VGG19 і Inception V3 з більшою кількістю шарів нейронів (19 і 159 відповідно) показали гірші результати. Це свідчить про те, що для вирішення даної задачі не потрібна велика кількість нейронних шарів у згортковій частині CNN.

Для підвищення точності системи можна збільшити навчальну вибірку, розширити діапазон представлених у ній зображень та додати у створену систему додаткові дані про пацієнтів, наприклад про їх стать і вік.

На класифікацію готовою мережею одного зображення потрібно менше 1 секунди, що дає можливість отримати від неї відповідь фактично миттєво, тобто поставлені вимоги до швидкості відгуку мережі, як і до точності, було виконано.

ВИСНОВКИ

У даній роботі було досліджено використання штучних нейронних мереж у задачі діагностики захворювань за рентгенівськими зображеннями, що передбачає можливість за рентгенівським знімком пацієнта автоматично визначити наявність на ньому захворювання, а саме пневмонії.

В рамках даної роботи були досягнуті наступні результати:

– проведено огляд існуючих методів глибокого навчання і алгоритмів попередньої обробки рентгенівських зображень;

– обрано і підготовано придатний набір даних для навчання;

– для кожного етапу навчання підібрано відповідний алгоритм, обрано найбільш прийнятну архітектуру нейронної мережі;

– проведено експерименти шляхом навчання нейронних мереж на даних з попередньою обробкою та без неї;

– представлено порівняльний аналіз роботи алгоритмів, підтверджено ефективність застосування методу доповнення даних (англ. data augmentation) для задачі розпізнавання зображень при малій кількості навчальних даних.

Для вирішення поставленої задачі було застосовано згорткові нейронні мережі. Практичні дослідження показали, що згорткові нейронні мережі можуть бути успішно застосовані для задачі діагностики захворювань за рентгенівськими зображеннями.

Також гарні результати дало використання техніки доповнення даних. Доповнення даних дало можливість отримати високий результат точності розробленої системи і підвищити його на 9%. Розроблена в ході даної роботи система дозволяє подавати на вхід зображення у форматі jpeg, аналізувати їх і розпізнати у них 2 класи – Пневмонія і Норма.

Точність системи склала 93.43%, що вище ніж точність аналогічних систем на 0.7% – найточніший з досліджених аналогів показує точність 92.73%. Також вона є вищою, ніж точність розпізнавання пневмонії на

рентгенівських знімках звичайною людиною та навіть лікарями-радіологами, що за даними досліджень складає 82.6% [11]. Отже показаний системою показник точності можна вважати хорошим.

Також розроблена система показала кращу точність ніж попередньо навчені мережі при застосуванні підходу transfer learning (донавчання), що підтверджує необхідність розробки власної моделі CNN для вирішення поставленої задачі.

При розробці програми було використано мову програмування Python і бібліотеки TensorFlow та Keras, що дозволяють швидко побудувати і натренувати нейронну мережу. Для тренування нейронної мережі використано безкоштовну платформу для машинного навчання і нейронних мереж Google Colaboratory, де користувачам безкоштовно надається доволі потужний GPU. Було встановлено, що навчання нейронних мереж на GPU дозволяє значно збільшити швидкість навчання, для даної моделі таким чином вдалося скоротити час навчання у майже 44 рази.

Створена програма представляє собою модуль, який може бути використаний у додатках-асистентах для лікарів, медичному обладнанні тощо.

Для підвищення точності системи можна збільшити навчальну вибірку, розширити діапазон представлених у ній зображень та додати у створену систему додаткові дані про пацієнтів, наприклад про їх стать і вік.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Template: COVID-19 pandemic data. URL: https://en.wikipedia.org/wiki/Template:COVID-19_pandemic_data (дата звернення 26.11.2020).

2. Kuldeep D., Sharun K., Ruchi T., Shubhankar S. Coronavirus Disease 2019–COVID-19. *Clin Microbiol Rev*. 2020. October. P. 20–28.

3. Hosny A., Parmar C., Quackenbush J., Schwartz L. H. Artificial intelligence in radiology. *Nat Rev Cancer*. 2018. August. P. 500–510.

4. McLuckie A. Respiratory disease and its management. New York: Springer, 2009. 51 p.

5. Ashby B., Turkington C. The encyclopedia of infectious diseases. 3rd ed. New York: Facts on File, 2007. 242 p.

6. Ruuskanen O., Lahti E., Jennings L., Murdoch D. Viral pneumonia. *Lancet*. 2011. March 22. P. 337–351.

7. Кишковский А. Н., Тютин Л. А., Есиновская Г. Н. Атлас укладок при рентгенологических исследованиях. Ленинград: Медицина, 1987. 520 с.

8. Kermany D., Goldbaum M., Cai W., Valentim C. Identifying medical diagnoses and treatable diseases by deep learning. *Cell*. 2018. P. 1122–1131.

9. Stephen O., Sain M., Maduh U., Jeong D. An efficient deep learning approach to pneumonia classification in healthcare. *J Healthcare Eng*. 2019. March 22. P. 471–480.

10. Narin A., Kaya C., Pamuk Z. Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. New York: Cornell University Press, 2020. 31 p.

11. Satia I., Bashagha S., Bibi A., Ahmed R. Assessing the accuracy and certainty in interpreting chest X-rays in the medical division. *Clin Med (Lond).* 2013. August 13. P. 825–851.

12. Grofman B., Owen G., Feld S. L. Thirteen theorems in search of the truth. *Theory & Decision*. 1983. P. 261–278.

13. McGrew, Roderick E. Encyclopedia of Medical History. London: Palgrave Macmillan UK. 400 p.

14. Bishop J. M. History and Philosophy of Neural Networks. London: University of London. 2015. 73 p.

15. Ніколенко С. І., Кадурін А. А., Архангельска Е. О. Глибоке навчання. Занурення у світ нейронних мереж. Пітер, 2016. 480 с.

16. Ф. Розенблатт. Принципи нейродинаміки. Перцептрони і теорія механізмів мозку. Москва: Мир, 1965. 478 с.

17. Minsky M., Papert A. Perceptrons - Expanded Edition. Cambridge, MA: The MIT Press, 1988. 308 p.

18. Aggarwal C. Neural Networks and Deep Learning. London: Springer International Publishing, 2018. 405 p.

19. Agostinelli F., Hoffman F., Sadowski P. Learning Activation Functions to Improve Deep Neural Networks. *Workshop paper contribution at the International Conference on Learning Representations (ICLR)*. 2015. P. 555-680.

20. Хайкін С. Нейронні мережі: повний курс. Київ: Діалектика-Вільямс, 2018. 2-е вид. 1104 с.

21. Khan S., Rahmani K. Shah S. A Guide to Convolutional Neural Networks for Computer Vision. Deli: Morgan & Claypool, 2018. 207 p.

22. Millstein F. Convolutional Neural Networks In Python: Beginner's Guide To Convolutional Neural Networks In Python. Berlin: CreateSpace Independent Publishing Platform, 2018. 121 p.

23. Ranjith M. Hunting Convolutional Neural Networks. New York: Creative Commons, 2019. 63 p.

24. Zafar I., Tzanidou G. Hands-On Convolutional Neural Networks with TensorFlow: Solve computer vision problems with modeling in TensorFlow and Python. Boston: Packt Publishing, 2018. 272 p.

25. Лутц М. Изучаем Python. Москва: Символ-Плюс, 2011. 355 с.

26. Gulli A., Sujit P. Deep Learning with Keras. Birmingham, UK: Packt Publishing. 2017. 318 p.