Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження штучних нейронних мереж у задачі діагностики респіраторних захворювань за допомогою аналізу рентгенівських знімків
(тема)

Виконав:
студент 2 курсу, групи СШІм-19-2
Горішній Д.О
(прізвище, ініціали)

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

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник д. т. н., проф. Терзіян В.Я
(посада, прізвище, ініціали)

Допускається до захисту

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

2021 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-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Горішньому Данилу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження штучних нейронних мереж у задачі діагностики респіраторних захворювань за допомогою аналізу рентгенівських знімків

затверджена наказом університету від 29 квітня 2021 р. № 309Ст
2. Термін подання студентом роботи до екзаменаційної комісії 20 травня 2021 р.
3. Вихідні дані до роботи Науково-технічні публікації, Інтернет-ресурси з обраної тематики, наукові конференції, публікації в наукових журналах, збірники конференцій, документація мови програмування Python, документація бібліотеки Keras, документація бібліотеки Pandas

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної галузі та постановка задачі, теоретичні дослідження, експериментальне моделювання і навчання моделі

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1 – Приклад рентгенівського зображення здорового пацієнта, Рисунок 2 – Приклад рентгенівського зображення з бактеріальною пневмонією, Рисунок 3 – Приклад рентгенівського зображення з вірусною пневмонією, Рисунок 4 – Схеми архітектури найбільш відомих типів ШНМ, Рисунок 5 – Схема основних шарів типової ШНМ, Рисунок 6 – Схема зворотного поширення помилки, Рисунок 7 – Найбільш поширені функції активації, Рисунок 8 – Приклад замалої швидкості навчання, Рисунок 9 – Приклад завеликої швидкості навчання, Рисунок 10 – Порівняння випадків недонавчання, добре навченої моделі і перенавчання на прикладі логістичної регресії, Рисунок 11 – Схематичне зображення операції згортки, Рисунок 12 – Додавання відступу на границях матриці при операції згортки, Рисунок 13 – Робота шару субдискретизації з функцією максимуму, Рисунок 14 – Попередньо треновані ШНМ у модулі Keras Applications

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на дипломну роботу	29.03.2021	виконано
2	Аналіз предметної галузі і постановка завдання	30.03.2021-31.03.2021	виконано
3	Теоретичні дослідження	31.03.2021-05.04.2021	виконано
4	Експериментальне моделювання та навчання моделі	05.04.2021-10.04.2021	виконано
5	Оформлення пояснювальної записки	10.04.2021-18.04.2021	виконано
6	Оформлення графічних матеріалів	18.04.2021-20.04.2021	виконано
7	Попередній захист	14.05.2021	виконано
8	Захист перед ЕК	20.05.2021	виконано

Дата видачі завдання 29 березня 2021 р.

Студент _____
(підпис)

Керівник роботи _____ д. т. н., проф. Терзіян Я. В
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 77 с., 22 рис., 2 табл., 2 дод., 29 джерел.

ГЛИБИННЕ НАВЧАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, КЛАСИФІКАЦІЯ, РЕНТГЕН, СУБДИСКТРЕТИЗАЦІЯ, ШТУЧНА НЕЙРОННА МЕРЕЖА.

Об’єкт дослідження – згорткові нейронні мережі та можливості їх використання для вирішення задачі діагностики респіраторних захворювань за допомогою аналізу рентгенівських зображень грудної клітки шляхом класифікації рентгенівських зображень за ознаками присутності на них певних ознак захворювань.

Предмет дослідження – адаптація згорткової нейронної мережі для вирішення задачі класифікації рентгенівських зображень за допомогою мови програмування Python та додаткових бібліотек: Scikit, Keras, Pandas, NumPy і TensorFlow, шляхом підбору гіперпараметрів і вибору топології штучної згорткової нейронної мережі, її навчання і тестування.

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

Методи дослідження – аналіз вже існуючих алгоритмів і систем, аналіз релевантної літератури, наукових публікацій і інтернет-джерел, вирішення практичних завдань і проведення експериментів та порівняльного аналізу.

РЕФЕРАТ

Пояснительная записка: 77 с., 22 рис., 2 табл., 2 прил., 29 источников.

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

Объект исследования – сверточные нейронные сети и возможности их использования для решения задачи диагностики респираторных заболеваний с помощью анализа рентгеновских изображений грудной клетки путем классификацию рентгеновских изображений по наличию на них определенных признаков заболеваний.

Предмет исследования – адаптация сверточной нейронной сети для решения задачи классификации рентгеновских изображений с помощью языка программирования Python и дополнительных библиотек: Scikit, Keras, Pandas, NumPy и TensorFlow путем подбора гиперпараметров и выбора топологии сети, ее обучения и тестирования.

Цель работы – исследование глубинных сверточных нейронных сетей для решения задачи диагностики респираторных заболеваний с рентгеновскими изображениями.

Методы исследования – анализ существующих алгоритмов и систем, анализ релевантной литературы, научных публикаций и интернет-источников, в решении практических задач и проведение экспериментов и сравнительного анализа.

ABSTRACT

Explanatory note: 77 p., 22 fig., 2 tabl., 2 ann., 29 sources.

ARTIFICIAL INTELLIGENCE, ARTIFICIAL NEURAL NETWORK, CLASSIFICATION, CONVOLUTIONAL NEURAL NETWORK, DEEP LEARNING, SUBSAMPLING, X-RAY.

The object of research – convolutional neural networks and the possibility of their use to solve the problem of diagnosis of respiratory diseases by analyzing X-ray images of the chest, the classification of X-ray images in the presence of certain signs of disease.

The subject of research is the adaptation of a convolutional neural network to solve the problem of X-ray image classification using Python programming language and additional libraries: Scikit, Keras, Pandas, NumPy and TensorFlow by selecting hyperparameters and selecting network topology, its training and testing.

The aim of the work is to learn deep convolutional neural networks to solve the problem of diagnosing respiratory diseases by X-ray images.

Research methods – analysis of existing algorithms and systems, analysis of relevant literature, scientific publications and Internet sources, solving practical problems and conducting experiments and comparative analysis.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ВООЗ – Всесвітня організація охорони здоров'я;

КТ – комп’ютерна томографія;

ШІ – штучний інтелект;

ШНМ – штучна нейронна мережа;

ASIC – Application-Specific Integrated Circuit – інтегральна схема спеціального призначення;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

GPU – Graphics Processing Unit – графічний процесор;

SGD – Stochastic gradient descent – стохастичний градієнтний спуск;

TPU – Tensor Processing Unit – тензорний процесор.

ВСТУП

Машинне навчання набирає обертів в останні роки і ми вже можемо побачити дуже багато прикладів цього у повсякденному житті: автомобілі з автоматичним керуванням, ефективний пошук в Інтернеті, розпізнавання голосу та зображень.

Однією з задач, яку у наш час успішно вирішують методи машинного навчання, є аналіз та класифікація зображень. Класифікація об'єктів є простим завданням для людини, але досить складним для комп’ютера. Вона може включати попередню обробку зображення, виявлення, сегментацію та класифікацію об'єктів.

Початком буму у сфері використання глибокого навчання і згорткових нейронних мереж для класифікації зображень став 2012 рік, причиною цього стала перемога штучної мережі SuperVision (зараз більш відома як AlexNet) на досить престижному змаганні з машинного навчання ImageNet. Ця згорткова мережа досягла приголомшливих для 2012 року 15,3% похибок, коли її найближчі конкуренти мали більш 26% [1].

Після успіху AlexNet, на тлі зростання обчислювальних потужностей сучасних GPU і популяризації CNN процент помилок при задачі класифікації на прикладі ImageNet став швидко падати, якщо у 2012 році результат переможця змагання AlexNet складав 15,3%, то у 2013 цей показник склав вже 12%, у 2014 7%, для 2015-2016 року близько 3%, у 2017 був досягнений результат у 2.3% похибок.

Станом на 2021 рік класифікація зображень за допомогою CNN стала досить розповсюдженою і точною. З огляду на складність завдання класифікації зображень, високу продуктивність автоматизованих систем (і як наслідок дешевизну їх використання) штучні згорткові мережі в якійсь мірі навчилися вирішувати її краще за багатьох людей.

У той же час, останній 2020 і поточний 2021 рік проходять на тлі епідемії короновірусної інфекції COVID-19. 11 березня 2020 року

поширення COVID-19 було визнано ВООЗ пандемією – надзвичайно сильною епідемією що поширилась на весь світ.

На 15.04.2021 у світі, згідно до даних ВООЗ було більше ніж 126 мільйонів випадків захворювання, при цьому, з них близько 3 млн призвели до смерті. У той же час, в Україні нараховується 1 млн 900 тис діагностованих випадків і більш ніж 38 тисяч загиблих [2].

Станом на 28 лютого 2020 року, у 91% пацієнтів з COVID-19 діагностувалася пневмонія [3].

Зазвичай, діагностика пневмонії використовує сполучення сукупності клінічних ознак та даних рентгенографії грудної клітки або КТ. І хоча комп'ютерна томографія є більш точна і ефективна, у той же час вона також є більш дорогою і складною – саме тому на практиці найбільш широко для діагностики респіраторних захворювань використовують дані рентгенівських знімків.

Зараз, у зв'язку з пандемією, медичні установи перевантажені пацієнтами і відчувають недостачу медичного персоналу та обладнання, тому зараз як ніколи актуальною є задача діагностики респіраторних захворювань за допомогою аналізу рентгенівських знімків методами машинного навчання, яка і була обрана у якості предметної області цієї кваліфікаційної роботи.

Така система, базована на автоматичній класифікації випадків бактеріальної та вірусної пневмонії може допомогти знизити навантаження деяких медичних працівників шляхом підвищення швидкості, точності і зручності діагностики бактеріальної та вірусної пневмонії за рентгенівськими знімками.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Пневмонія – це запальний процес одного або двох легень, який характеризується підняттям температури і викидом рідини в альвеоли. Альвеоли – це невеликі повітряні осередки в легкому, стінки яких відповідають за газообмін і є складовими частинами легеневих тканин.

До ризику виникнення і розвитку пневмонії найчастіше піддаються діти і люди похилого віку, які мають ослаблений імунітет, хронічні хвороби, цукровий діабет, проблеми з серцево-судинною системою. Також в групі ризику знаходяться пацієнти, які зловживають курінням і вживанням алкоголю, люди з вродженими проблемами дихання і легеневої системи.

Зазвичай вона виникає при інфікуванні бактеріями, трохи рідше вірусами та іншими мікроорганізмами, ураженні деякими медичними препаратами, як наслідок автоімунних захворювань, тощо [4]. У більшості випадків при захворюванні спостерігають наступні симптоми: кашель, біль у грудях, гарячка, ускладнене дихання [5].

Навіть якщо не враховувати пандемію COVID-19 – це дуже широко розповсюджена хвороба, що вражає приблизно 500 мільйонів осіб по всьому світу кожного року. Це одна з найпоширеніших з причин смерті в усіх вікових групах – приблизно 5 мільйонів смертей (близько 7 % від загального числа смертей у світі) у річному виразі. З загальної множини у 500 млн випадків на вірусну пневмонію припадає приблизно 250 мільйонів випадків.

Найбільш вразливі групи населення для яких спостерігаються найвищі показники летальності – це люди що страждають на хронічні

захворювання, малі діти віком до п'яти років і літні люди старші за 70 років.

Окрім цього можна побачити диференціацію в залежності від рівня життя у конкретній країні – наприклад показники захворюваності у країнах, що розвиваються, приблизно у п'ять разів вище у порівнянні до розвинених країнах.

Історично, у XIX столітті і раніше пневмонія мала значно більшу летальність, але поява антибіотиків та вакцин у XX столітті значно збільшила відсоток людей що одужали. Але, незважаючи на це, в нерозвинутих країнах, а також серед перерахованих вище особливо вразливих груп населення пневмонія все ще залишається великою загрозою.

У процесі діагностування пневмонії використовують різні методи, такі як: рентген грудної клітини, комп’ютерна томографія легень, ультразвукове дослідження грудної клітки, голчаста біопсія легені та магнітно-резонансну томографію грудної клітки. З усіх перерахованих вище методів рентген грудної клітки є одним з найкращих методів за сукупністю ключових характеристик – і тому є найрозповсюдженішим методом діагностування пневмонії.

Наприклад у порівнянні з КТ дослідженням, рентген зазвичай займає значно більше часу, ніж рентгенологічне, а в багатьох слаборозвинених регіонах у наш час спостерігається істотна нестача якісних КТ-сканерів. У свою чергу саме рентгенівське дослідження ще залишається у якості найпоширенішого і найпопулярнішим методом діагностики респіраторних захворювань легень і займає одну з основних ролей у клінічній допомозі та у різноманітних епідеміологічних дослідженнях.

Рентгенографія – це метод дослідження внутрішньої структури об’єктів, шляхом її проектування за допомогою рентгенівських променів на спеціальну плівку або папір. У більшості випадків цей термін використовується як визначення медичного, не інвазійного методу

дослідження, який базується на отриманні проекції анатомічної структури організму шляхом проходження через нього рентгенівського випромінювання і реєстрації ступені ослаблення цього випромінювання на певному чутливому до рентгенівського випромінювання носії [6].

В Україні основним і найпоширенішим способом отримання рентгенівських знімків є їх фіксація на спеціальній чутливій до рентгенівських променів плівці і її прояві у подальшому. Але також існують системи, що забезпечують отримання подібних даних не у аналоговому, а у цифровому вигляді, який у більшості розвинених країн вже витіснив аналоговий. У менш розвинених країнах що розвиваються, у зв'язку з вартістю і складністю виготовлення, цифровий спосіб все ще значно поступається більш архаїчному аналоговому.

Рентгенівські знімки з метою діагностики респіраторних захворювань можуть виконуватись у досить різних положеннях – у більшості випадків стоячи, але інколи і в профіль, та навіть лежачи, у випадку якщо пацієнт є немобільним.

Патології, що можуть бути діагностуванні за допомогою аналізу рентгенівських знімків спричиняють наступні, можливі зміни у вигляді в зображенні органів на знімку:

- патологічне затінення або навпаки, просвітлення деяких ділянок;
- зміна ступеня прозорості легеневих полів;
- посилення або відсутність легеневого малюнка;
- зміна контурів серця, зміна висоти стояння купола діафрагми.

У зв'язку з тим фактом що рентгенівські знімки є негативами більш світлі ділянки на них лікарі називають «затемненням». Тому наприклад, здорові легені що добре заповнені повітрям на рентгені приймають вигляд чорних ділянок, а у свою чергу ділянки запалення при пневмонії приймають вигляд світлих плям, яку лікарі називають тінню. Приклади зображень рентгену здорового пацієнта, пацієнта з бактеріальною і вірусною пневмонією наведено на рисунках 1.1, 1.2, 1.3 відповідно.

Рисунок 1.1 – Приклад рентгенівського зображення здорового пацієнта

Рисунок 1.2 – Приклад рентгенівського зображення пацієнта з бактеріальною пневмонією

Рисунок 1.3 – Приклад рентгенівського зображення пацієнта з вірусною пневмонією

1.2 Аналіз існуючих систем

Перші системи націлені на автоматизований вияв вузликів в легенях почали з’являтися ще у кінці 20 століття – але ці системи не мали значного успіху. Головна причина – нестача обчислювальних ресурсів для реалізації автоматизованих методів обробки зображень. Да і у будь-якому разі виявлення захворювань легень з використанням базових методів обробки зображень вимагала багато часу.

Після появи GPU і згорткових нейронних мереж (CNN) продуктивність подібних систем для діагностики захворювань легень досить значно покращилась. Головною перевагою використання згорткових нейронних мереж у порівнянні зі їх попередниками стала можливість виконання подібних задач аналізу зображень рентгенівських знімків без прямої участі людини.

Після спалаху COVID-19 велика кількість різноманітних компаній на світовому рівні розробили велику множину рішень на основі штучного інтелекту для діагностування COVID-19 при рентгенографічному дослідженні органів грудної клітки.

Авжеж такі інструменти з використанням методів глибокого навчання штучних нейронних мереж можуть досить гарно використовуються з метою класифікації і розподілу випадків захворювання за конкретною інфекцією або відсотком ураження легень, або спостереженням за процесом розвитку хвороби.

Такий спосіб діагностики з використанням методів штучного інтелекту може допомогли зменшити ступінь завантаження лікарів-рентгенологів, і частково зняти необхідність використання стандартних тестів за допомогою нуклеїнових кислот у якості головного діагностичного способу для виявлення випадків зараження коронавірусом.

Наприклад у роботі за назвою «Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning» була створена модель глибокого навчання для аналізу рентгенівських зображень з ціллю виявлення пневмонії та інших захворювань і ця модель досягла точності тестування майже у 92% точності. [8].

В схожому дослідженні був проілюстрований досить якісний підхід для здійснення класифікації пневмонії за допомогою методів глибокого навчання на основі моделі побудованої на основі згорткової нейронної мережі, структура якої нараховувала чотири згорткових шара у комбінації з двома повнозв’язними, яка у сукупності з використанням додаткових методологій покращення навчання, таких як механізм доповнення даних, досягла значення у 93.73% точності [9].

Інший приклад – глибока згорткова нейромережа створена колективом авторів з Корнельського університету, що призначена для автоматичного діагностування COVID-19 з використанням попередньо навчених моделей що базовані на аналізі рентгенівських зображень. В

рамках цього дослідження автори використовували попередньо навчені моделі ResNet50, InceptionV3 та Inception-ResNetV2 і техніку transfer learning, щоб отримати вищу точність прогнозування на аналізі рентгенівських знімків і досягли приголомшливих 96% точності [10].

Таким чином можна побачити що сучасні глибокі згорткові нейроні мережі без особливих зусиль досягають більш ніж 90% точності у задачі класифікації захворювань легень шляхом аналізу рентгенівських знімків.

Для порівняння, згідно дослідження Лондонського коледжу радіології, в ході якого прийняли участь 138 кваліфікованих лікарів, відсоток вірно класифікованих випадків респіраторних захворювань за рентгенівськими знімками цими висококваліфікованими лікарями склав у середньому близько 82.6% [11].

Тобто, вже зараз, подібні автоматизовані системи засновані на використанні глибоких нейронних сітей за своєю точністю навіть перевершують лікарів. Незважаючи на ці факти і різницю у продуктивності і точності класифікації все ще зарано сподіватись на швидку заміну всіх лікарів-рентгенологів на використання моделей штучного інтелекту у рамках подібних задач. Основна перепона для цього – це велика кількість необхідних регуляторних і сертифікаційних кроків, крім того для впровадження подібних систем є стандартизувати формат і якість видачі знімків для досить широкого спектру номенклатури рентгенівського обладнання.

Крім того треба приймати до уваги той факт що ймовірність прийняття правильного рішення групою фахівців вище, ніж кожним з них окремо [12]. Тому точність спільної класифікації лікаря і моделі вище, ніж точність подібної класифікації їх поодинці. Враховуючи ці фактори можна дійти висновку що станом на сьогодні найбільш доцільно використовувати подібні моделі не в якості повністю автоматизованих систем, а у складі систем підтримки прийняття рішень для видачі порад для лікарів.

Таким чином, лікар може використовувати модель в ролі порадника і в такому разі її головна ціль зменшити кількість помилок. Подібні помилки класифікації призводять до того що частину захворювань пропускають. Подібних похибок на практиці зустрічається дещо менше, у зв'язку з тим фактом що лікарі прагнуть до мінімізації кількості таких помилок першого роду. Інші помилки мають досить неприємні наслідки, наприклад частина людей може проходити лікування від пневмонії, яка насправді у них відсутня і це може призвести до того що частина місць в лікарнях буде зайнята без гострої необхідності і здорові пацієнти вживають непотрібні їм ліки, що можуть нанести шкоду здоров'ю таких пацієнтів і призводить до небажаних витрат на придбання цих ліків.

Згідно до даних досліджень [13], в більшості випадків у пацієнтів що звертаються до лікарень і зокрема здійснюють рентгенологічне обстеження насправді не спостерігається жодних патологій, тому подібна система автоматизованої класифікації на основі використання методів штучного інтелекту допоможе відсіяти такі випадки і більш точно і швидко виділити саме тих пацієнтів що потребують лікування або навіть госпіталізації.

1.3 Проблеми задачі класифікації рентгенівських зображень

Взагалі, задача діагностики захворювань легень за рентгенівськими зображеннями є дещо більш складною ніж стандартна задача розпізнавання або класифікації об'єктів на зображеннях, тому що різниця між об'єктами різних класів в цій задачі є відносно невеликою і у деяких випадках вона не помітна навіть людському оку – тому часто звичайна людина не може за рентгенівськими знімками відрізнити випадки пневмонії і норми.

Навіть професійні лікарі можуть мати досить високий відсоток помилок. Згідно до деяких досліджень, вже наведених вище – в

середньому ця точність може складати близько 80% [11]. Авжеж серед звичайних людей, що не є професійними лікарями-рентгенологами відсоток коректних розпізнавань буде ще нижчим.

Таким чином, у зв’язку с особливостями обраної предметної області архітектура побудованої нейронної мережі для вирішення поставленої задачі повинна бути такою, щоб вона могла визначити на вхідних зображеннях досить складні і водночас невеликі ознаки, що інколи не помітні навіть оку людини. Авжеж розробка такої архітектури мережі потребує комплексного дослідження і тестування.

Одна із найбільших проблем застосування ШІ в медицині – підготовка коректних вхідних медичних даних для процесу навчання алгоритмів, так як для створення подібних датасетів необхідний досить значний обсяг часу професійних і висококваліфікованих фахівців досить вузької спеціалізації.

Крім того, у багатьох країнах дані з медичних закладів є конфіденційними і існує велика кількість юридичних перепон для їх отримання і використання. Саме тому, у відкритому доступі знаходиться досить мало наборів даних рентгенівських знімків і більшість з них містять не дуже багато даних – і це враховуючи те, що для дійсно якісного навчання глибокої нейронної мережі може знадобитися величезна кількість подібних зображень.

Тому очевидно, що для вирішення поставленої задачі діагностики захворювань легень шляхом аналізу рентгенівських зображень методами машинного навчання наразі недостатньо навчальних даних і є потреба в зборі даних для цієї задачі.

Можливим централізованим рішенням бачиться потенційне створення об'єднаної міжнародної платформи зберігання медичних даних, де різноманітні спеціалісти зможуть зібрати дані для подальшого їх застосування у сфері ШІ у комбінації з відповідною оптимізацією законодавства, яке станом на сьогоднішній день досить сурово охороняє

персональні дані громадян у сфері охорони здоров’я. Це дозволить в майбутньому підвищити ефективність застосування машинного навчання в медицині завдяки аналізу різнопланових даних з різних джерел.

Таким чином, у зв’язку з досить ймовірною нестачею навчальних даних, вирішення задачі діагностики захворювань легень за рентгенівськими зображеннями потребує від нас використання методів попередньої обробки і доповнення вхідних даних. Ці методи будуть розглянуті далі у даній роботі.

Як неодноразово було зазначено вище, на теперішній час, рентгенографія – це з технічної точки зору простий і у той же час найбільш доступний для населення (особливо для не самих заможних країн) метод попереднього діагностування респіраторних захворювань легень, в тому числі і діагностування подібних захворювань на самих ранніх стадіях їх розвитку.

Цей тип діагностування дуже широко представлений у світі, і навіть незважаючи на існування інших, більш досконалих і з певної точки зору ефективних способів діагностики, саме цей спосіб все ще залишається у наш час пріоритетним у широкій медичній практиці, і все ще використовується навіть у досить сучасних і добре оснащених клініках та медичних центрах.

Додаткову складність нашого завдання полягає в тому що вхідні рентгенівські зображення як правило мають досить низьку роздільну здатність зображень, а у випадку використання або обробці даних з різноманітних джерел можуть виникнути проблеми пов’язанні з їх неоднорідністю і крім того у можливості з наявності на зображені різного роду «шумів» (у якості прикладу можуть слугувати засвічені зображення, наявність додаткових сторонніх предметів та позначок на знімку і т. п.).

В контексті нашої задачі, окрім шумів, додаткові перепони для аналізу може внести наявність на зображеннях деяких природних анатомічних структур (кісток та інших структур), які у деяких випадках і

положеннях можуть приховати наші цільові аномалії, що знаходяться в легеневій тканині. Саме по цій причині багатьом спеціалістам для постановки більш точного діагнозу необхідна серія з декількох знімків, наприклад фронтальний і боковий.

Додатковою проблемою може бути присутність в датасеті для навчання деяких неякісних, пересвічених або навпроти, затемнених даних. Слід пам’ятати що тіло пацієнта на рентгенівських знімках може знаходитись в досить різних положеннях, наприклад бути повернутим під кутом (наприклад при проведенні дослідження лежачих або немобільних пацієнтів). Такі фактори в деякій мірі можуть ускладнювати використання методів машинного навчання.

Тому можна дійти висновку, що наша задача діагностики захворювань легень за рентгенівськими зображеннями є досить нетривіальною, комплексною і тому потребує досить ретельного вивчення і комбінування теоретичних та експериментальних досліджень, які і будуть проведені нижче у цій роботі.

1.4 Постановка задачі

Задачею даної кваліфікаційної роботи є полегшення роботи медичних працівників при діагностиці респіраторних захворювань легень за допомогою аналізу рентгенівських зображень використовуючи методи машинного навчання.

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

– визначення завдання, яке буде вирішуватись штучною нейронною мережею: класифікація, прогнозування, кластеризація, інше;

– визначення обмежень розв'язуваної задачі (швидкість, точність відповіді);

– визначення основних параметрів вхідних даних (тип зображення, його формат і розмірність) і вихідних даних (вихідні класи);

– визначення оптимальної структури нейронної мережі для вирішення поставленої задачі.

Задача створюваної штучної нейронної мережі це класифікація зображень, конкретно класифікація рентгенівських зображень. Накладаються наступні обмеження на мережу, перше, це швидкість відповіді що повинна становити не більше 1 секунди, і друге, це загальна точність розпізнавання не менше 90%. Вхідними даними будуть виступати зображення в форматі .jpeg, розміру 224x224 пікселів. На виході мають бути три класи, що визначатимуть те що на рентгенівському знімку була виявлена вірусна або бактеріальна пневмонія, або пневмонія взагалі не виявлена.

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

– проведення теоретичних досліджень і розгляд існуючих методів машинного навчання у контексті застосування їх до обраної проблематики;

– пошук і підготовка вхідного набору даних що буде використаний для навчання розробленої штучної нейронної мережі;

– проведення аналізу і вибір за його результатами найбільш прийнятних алгоритмів і методів для вирішення задачі класифікації зображень, а саме сфокусуватись на використанні штучних згорткових нейронних мереж для цієї мети;

– проведення необхідних експериментів, здійснення експериментального моделювання та навчання моделі, її тестування;

– проведення порівняльного аналізу роботи різноманітних алгоритмів;

– проведення аналізу отриманих результатів.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Штучні нейронні мережі

Дослідження в області ШНМ пережили три періоди активізації [14]. Перший пік в 40-х роках обумовлений піонерською роботою Маккаллок і Питтса [15].

Другий виник в 60-х завдяки теоремі збіжності перцептрона Розенблатта [16] і роботі Мінського і Пейперта [17], яка вказала обмежені можливості найпростішого перцептрону.

Результати Мінського і Пейперта погасили ентузіазм більшості дослідників, особливо тих, хто працював в області обчислювальних наук. Після цього, в дослідженнях по нейронних мереж затишшя тривало майже 20 років.

З початку 80-х років ШНМ знову привернули інтерес дослідників, що пов'язано з енергетичним підходом Хопфілда [18] і алгоритмом зворотного поширення для навчання багатошарового перцептрона (багатошарові мережі прямого поширення), вперше запропонованого Вербосом [19] і незалежно розробленого рядом інших авторів. Алгоритм здобув популярність завдяки Румельхарту [20] – в 1986 році Андерсон і Розенфельд [21] підготували докладну історичну довідку про розвиток ШНМ прямого поширення), що були вперше запропоновані Вербосом.

На сьогоднішній день штучні нейроні мережі успішно використовуються у широкому спектрі різноманітних задач, які можна поділити на наступні групи:

– розпізнавання образів і класифікація – з метою вирішенняя таких задач у якості образів можуть бути представленні досить різні об'єкти, наприклад текстові символи, зображення, звуки і т.п. Цей навчальний датасет повинен бути розмічений – мережа повинна мати чітку інформацію щодо належності об'єкту до одного з вихідних класів. Зазвичай вхідні дані

трансформують певним чином до форми вектора ознак. Необхідно забезпечити той факт, що загальна сукупність ознак об'єкту чітко відносила його до певного вихідного класу з метою уникнення ситуацій коли мережа буде класифікувати один і той же зразок до декількох вихідних. В результаті навчання така модель може приймати на свій вхід образи, клас яких невідомий і на її виході отримувати інформацію щодо належності її до певного вихідного класу. Архітектура таких мереж має характерну рису – кількість нейронів у вихідному шарі точно дорівнює числу вихідних класів, таким чином з’являється відповідність між виходом нейронної мережі і класом, який він представляє. Після того як на вхід мережі подається певний образ, на її вихідному шарі повинен активуватися один з нейронів, що буде сигналізувати до відношення цього вхідного об’єкту до певного вихідного класу, при цьому на інших виходах повинен з’явитися сигнал того що до цього класу вхідний об’єкт не належить – якщо ознака приналежності до класу з’являється на декількох виходах одночасно вважається що мережа «не впевнена» в своїй відповіді;

– прогнозування – вирішення таких задач базуються на можливості нейронної мережі до прогнозування безпосередньо випливають з її здатності до узагальнення і виділення прихованих залежностей між вхідними та вихідними даними. Після завершення навчання мережа стає спроможна передбачити значення певної послідовності на основі ряду її попередніх значень і (або) набору певних, існуючих зараз зовнішніх чинників. Авжеж таке прогнозування можливо тільки у тих випадках коли історія попередніх змін дійсно має вплив на майбутні значення. Наприклад прогнозування ціни акцій на основі цін за минулий місяць дійсно може спрацювати, але у той же час спроба прогнозування результатів лотереї на основі даних про її попередні тиражі не дасть ніяких результатів;

– прийняття рішень і управління – ці завдання близькі до задачі класифікації. Класифікації підлягають ситуації, характеристики яких надходять на вхід нейронної мережі. На виході мережі при цьому повинен

з'явитися ознака рішення, яке вона прийняла. При цьому в якості вхідних сигналів використовуються різні критерії опису стану керованої системи;

– кластеризація, до цього типу задач відносять ті задачі, які пов’язані з поділом множини вхідних сигналів на класи за той умовою що кількість і певні ознаки вихідних класів нам заздалегідь не відомі. Після завершення навчання така модель здатна віднести вхідний об’єкт до певного кластеру (вихідного класу). Також цілком можлива ситуація коли вхідний об’єкт не відноситься ні до одного з виділених кластерів і це може слугувати ознакою нових, відсутніх в навчальному датасету даних. Після завершення навчання відповідність класів що виділила модель і певними класами, предметної області вже встановлюється людиною;

– апроксимація – нейронні мережі можуть апроксимувати безперервні функції. У відповідності до апроксимаційної теореми [22] за допомогою лінійних операцій і каскадного з'єднання можна з довільного нелінійного елемента отримати пристрій, обчислює будь-яку безперервну функцію з деякої наперед заданою точністю. Це означає, що нелінійна характеристика нейрона може бути довільною: від сігмоідальної до довільного хвильового пакета або вейвлета, синуса або многочлена. Від вибору нелінійної функції може залежати складність конкретної мережі, але з будь-якої нелінійністю мережа все ще залишається універсальним аппроксиматором і при правильному виборі структури може досить точно апроксимувати функціонування будь-якого безперервного автомата;

– стиснення даних і асоціативна пам'ять – здатність нейромереж до виявлення взаємозв'язків між різними параметрами дає можливість висловити дані великої розмірності більш щільно, якщо дані тісно взаємопов'язані між собою. Зворотний процес – відновлення вихідного набору даних з частини інформації – називається (авто) асоціативної пам'яттю. Асоціативна пам'ять дозволяє також відновлювати вихідний сигнал, образ з зашумленних або пошкоджених вхідних даних. Рішення

завдання гетероасоціативної пам'яті дозволяє реалізувати пам'ять, що адресується за вмістом.

Як вже зазначалось, сфера використання штучний нейронних мереж досить широка і наведена вище класифікація є досить умовною і нейронні мережі також використовуються і для інших задач, наприклад для аналізу даних, розв'язання оптимізаційних задач, знаходження патернів у великих обсягах даних, орієнтації в просторі та інших різноманітних завдань.

Широта сфер використання штучних нейронних мереж обумовлена великою кількістю переваг у порівнянні з більш традиційними алгоритмами вирішення вищеперерахованих задач, таких як:

– можливість вирішення задач в умовах невизначеності. Дякуючи здатності до навчання нейронні мережі дозволяють виконувати вирішування завдань в умовах невідомих закономірностей і залежностей між вхідними та вихідними даними, що у свою чергу дозволяє ШНМ працювати з неповними даними;

– стійкість до шумів у вхідних даних. Штучна нейронна мережа має здатність до самостійного виявлення неінформативних для подальшого аналізу параметрів і виконувати їх фільтрацію, у зв'язку з чим відпадає необхідність у виконанні попереднього аналізу вхідних даних;

– гнучкість структури нейронних мереж. Компоненти моделей штучних нейронних мереж – нейрони і зв'язки між ними можуть бути комбіновані різноманітними шляхами.

– висока швидкодія. Обробка вхідних даних може виконуватися багатьма нейронами одночасно у паралельному режимі, таким чином штучні нейронні мережі мають здатність вирішення завдання швидше, ніж більшість класичних алгоритмів;

– адаптація до змін навколишнього середовища. Штучні нейронні мережі у ході навчання на початкових даних можуть гарно підлаштовуватися під зміни навколишнього середовища (у якості прикладу може слугувати зміна ситуації на ринку цінних паперів, якщо завдання

нашої нейромережі – прогнозування коливань цін на біржі). Тому у випадках коли необхідно вирішувати певне завдання в умовах нестаціонарного, мінливого середовища, гарним рішенням можуть стати штучні нейронні мережі які здатні обробляти дані у режимі реального часу. Чим вище подібні адаптивні здібності побудованої системи, тим більш стійкою і результативною буде її робота в умовах нестаціонарного і мінливого навколишнього середовища;

– відмовостійкість нейронних мереж. На певну, несприятливу для побудованої штучної нейронної мережі зміну умов ШНМ реагує лише невеликим зниженням своєї результативності. Ця позитивна і корисна особливість ШНМ пояснюється розподіленим характером зберігання інформації в нейронної мережі, тому тільки суттєві і серйозні пошкодження її структури можуть істотно вплинути на результативність штучних нейронних мереж.

Незважаючи на великий вищенаведений список переваг, штучні нейронні мережі мають і великий кількість недоліків:

– відповідь отримана від штучної нейронної мережі завжди носить приблизний характер. Нейронні мережі нажаль не здатні давати точні і однозначні відповіді. Але, у той же час, завдання, в яких виникає необхідність застосовування ШНМ і одночасно отримувати лише точні відповіді, зустрічаються не дуже часто;

– нездатність до прийняття рішень в кілька послідовних етапів. Штучна нейронна мережа не здатна до вирішення завдань, що вимагають послідовного, ітеративного виконання деякої послідовності з декількох кроків, вона здатна вирішувати завдання тільки в рамках однієї ітерації. Саме тому штучна нейронна мережа не може наприклад довести математичну теорему або вирішити певну математичну задачу аналітичним шляхом;

– нездатність до вирішення обчислювальних завдань. У штучну нейрону мережу не можна завантажити, припустимо, математичне

рівняння і отримати його рішення для різних параметрів. Але це і не є призначенням нейронних мереж. Хоча у наш час, базуючись на використанні авторегресійних генеративних мовних моделей є досить успішні спроби вирішення і таких завдань;

– трудомісткість і тривалість навчання. Для того щоб нейронна мережа могла коректно вирішувати поставлені завдання, потрібно провести її навчання на десятках мільйонів наборів вхідних даних. Але вже розроблені різні технології прискореного навчання, сучасні відеокарти дозволяють навчати нейронні мережі в сотні разів швидше, а недавно з'явилися готові, попередньо навчені нейронні мережі, зокрема, що розпізнають образи. На основі таких нейронних мереж можна створювати додатки, не займаючись тривалим навчанням, а займаючись лише донавчанням моделі під свою предметну область. Ця технологія використання попередньо навчених ШНМ носить назву трансферного навчання (transfer learning).

Трансферне навчання – це метод машинного навчання, в якому модель, розроблена для завдання, повторно використовується в якості відправної точки для моделі для другого завдання.

Це доволі популярний підхід в глибокому навчанні, коли попередньо навчені моделі використовуються в якості відправної точки в задачах комп'ютерного зору і обробки природної мови, з огляду на величезні обчислювальні і часові ресурси необхідні для навчання моделей ШІ.

Користуючись цим підходом, навіть в умовах невеликого за розміром навчального датасету і обмежених обчислювальних ресурсів і відведеного часу на навчання ШНМ можна досягти гарних показників результативності моделі, при цьому оптимізувавши трудовитрати і на проектування структури створюваної ШНМ.

Станом на сьогоднішній день, з метою вирішення різноманітних типів задач була розроблена величезна кількість різноманітних архітектур

штучних нейронних мереж – більшість з яких наведена нижче, на рисунку 2.1.

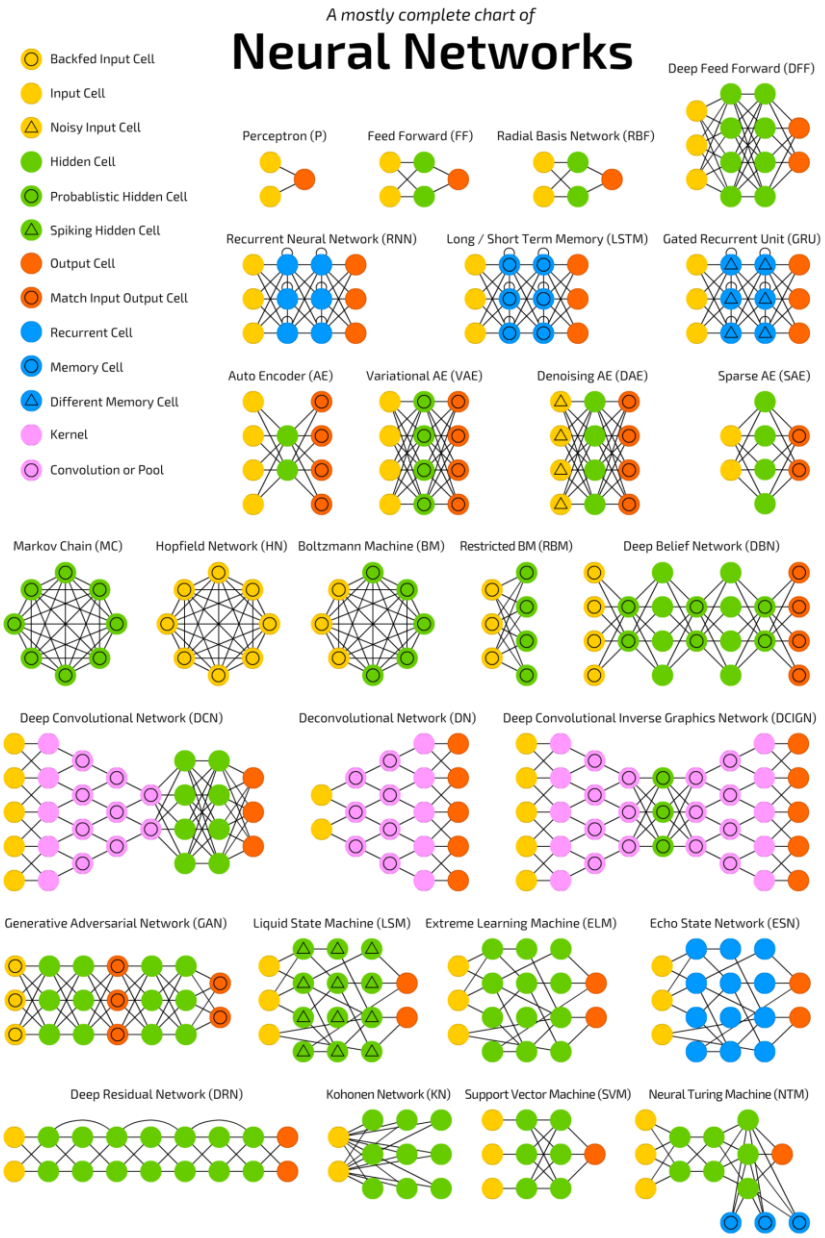

Рисунок 2.1 – Схеми архітектури найбільш відомих типів ШНМ

2.2 Навчання штучних нейронних мереж

Найважливішим властивістю нейронних мереж є їх здатність навчатися на основі даних навколишнього середовища і в результаті навчання підвищувати свою продуктивність. Підвищення продуктивності

відбувається з часом відповідно до певних правил. Навчання нейронної мережі відбувається за допомогою інтерактивного процесу корегування синаптичних ваг і порогів. В ідеальному випадку нейронна мережа отримує знання про навколишнє середовище на кожній ітерації процесу навчання.

Навчання – це процес, в якому вільні параметри нейронної мережі настроюються за допомогою моделювання середовища, в яку ця мережа вбудована. Тип навчання визначається способом підстроювання цих параметрів.

Це визначення процесу навчання нейронної мережі передбачає наступну послідовність подій: спочатку нейронну мережу надходять стимули із зовнішнього середовища, потім, як наслідок змінюються вільні гіперпараметри нейронної мережі і після зміни внутрішньої структури нейронна мережа відповідає на вхідні данні вже іншим чином.

Вищевказаний список чітких правил вирішення проблеми навчання нейронної мережі називається алгоритмом навчання. Нескладно здогадатися, що не існує універсального алгоритму навчання, відповідного для всіх архітектур нейронних мереж. Існує лише набір засобів, представлений безліччю алгоритмів навчання, кожен з яких має свої переваги. Алгоритми навчання відрізняються один від одного способом налаштування синаптичних ваг нейронів. Ще однією відмінною характеристикою є спосіб зв'язку навченою нейронної мережі з зовнішнім світом. У цьому контексті говорять про парадигму навчання, пов'язаної з моделлю навколишнього середовища, в якому функціонує дана нейронна мережа.

Існують два концептуальних підходи до навчання нейронних мереж: навчання з вчителем і навчання без учителя.

Навчання нейронної мережі з учителем передбачає, що для кожного вхідного вектора з навчальної множини існує необхідне значення вихідного вектора, званого цільовим. Ці вектори утворюють навчальну

пару. Ваги мережі змінюють доти, поки для кожного вхідного вектора не буде отриманий прийнятний рівень відхилення вихідного вектора від цільового.

Навчання нейронної мережі без вчителя є набагато більш правдоподібною моделлю навчання з точки зору біологічних коренів штучних нейронних мереж. Навчальна множина складається лише з вхідних векторів. Алгоритм навчання нейронної мережі підлаштовує ваги мережі так, щоб виходили узгоджені вихідні вектори, тобто щоб пред'явлення досить близьких вхідних векторів давало однакові виходи.

У більшості випадків штучна нейронна мережа умовно складається з 3-х основних компонентів:

– вхідний шар;
– приховані (обчислювальні) шари;
– вихідний шар.

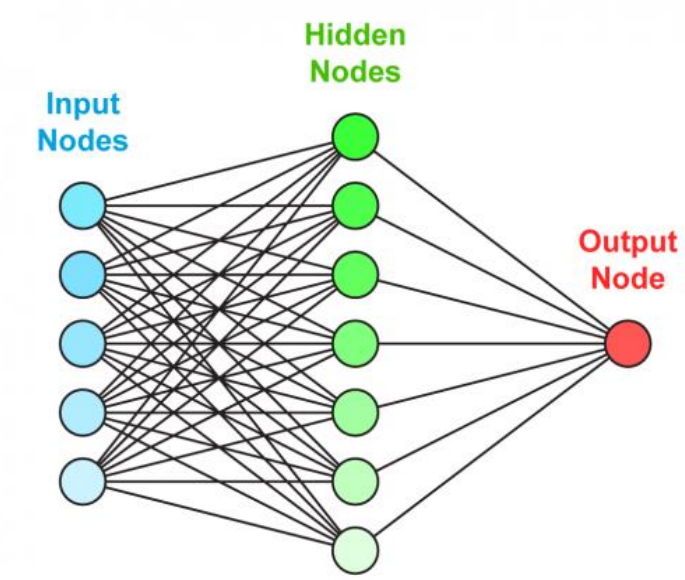

Рисунок 2.2 – Схема основних шарів типової ШНМ

Навчання нейромереж відбувається в два етапи, перший етап це пряме поширення помилки, другий – зворотне поширення помилки. Під час прямого поширення помилки робиться прогноз відповіді. При

зворотному поширенні помилка між фактичною відповіддю і передбаченим результатом мінімізується. Схема етапу прямого поширення помилки наведена на рисунку 2.3.

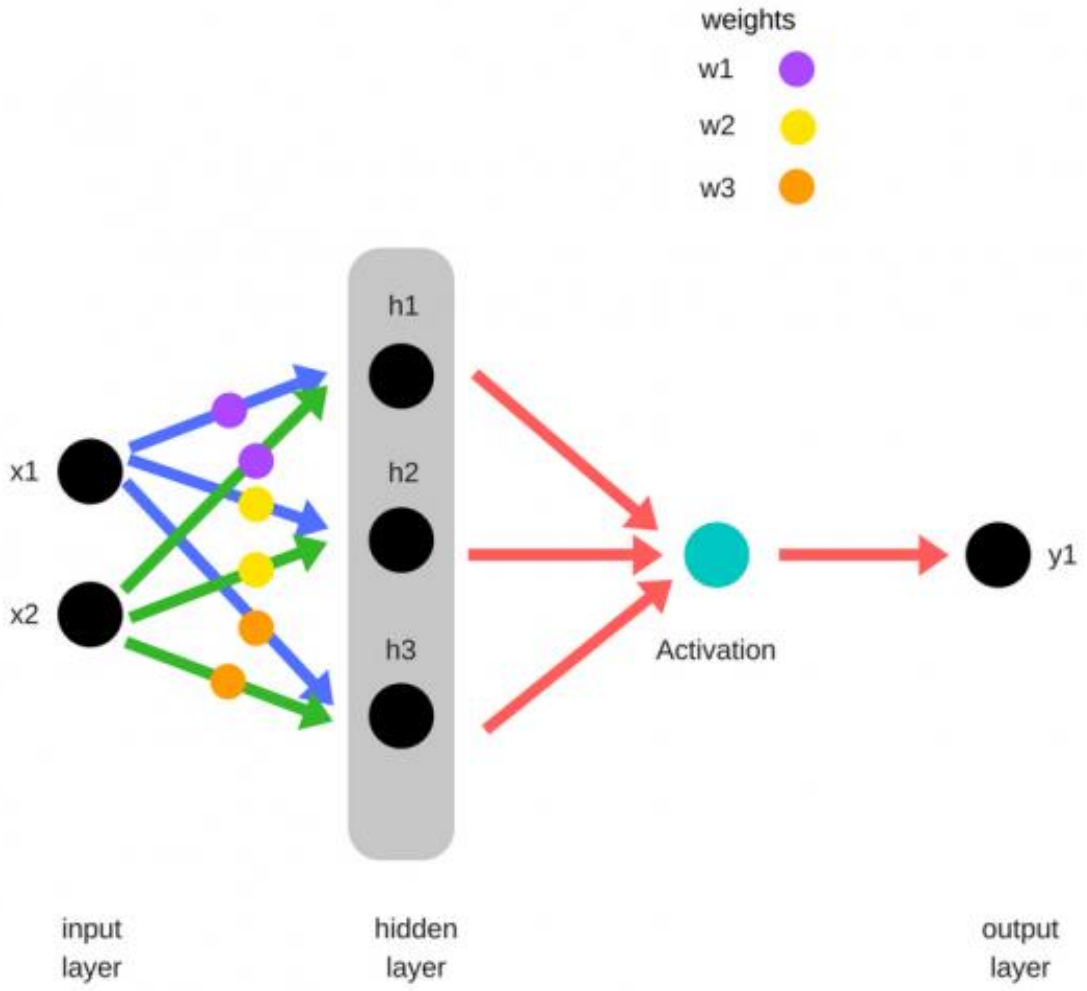

Рисунок 2.3 – Схема етапу прямого поширення помилки

Перший етап прямого поширення працює наступним чином: спочатку задаються довільні, початкові значення ваг w1, w2, w3. Потім ці ваги множаться на вхідний вектор, для формування прихованого шару:

– h1 = (x1 * w1) + (x2 * w1);

– h2 = (x1 * w2) + (x2 * w2);

– h3 = (x1 * w3) + (x2 * w2).

Далі, данні з прихованого шару передаються через деяку нелінійну функцію активації для отримання першого виходу мережі:

y_ = fn(h1 , h2, h3). Після цього настає час другого етапу, етапу зворотного поширення помилки, загальна схема якого наведена на рисунку 2.4.

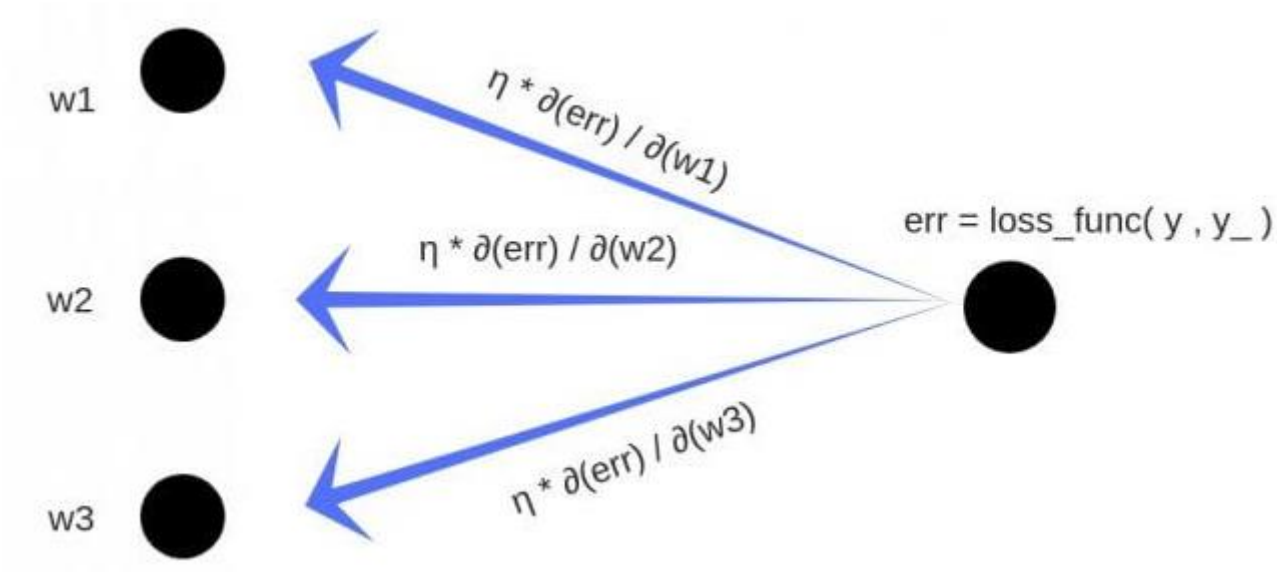

Рисунок 2.4 – Схема зворотного поширення помилки

На цьому етапі, сумарна помилка обчислюється як різниця між очікуваним значенням «y» (з навчального набору) і отриманим значенням «y_» (пораховані на попередньому етапі прямого поширення помилки), що проходять через функцію втрат.

Приватна похідна помилки обчислюється по кожній вазі (ці приватні диференціали відображають внесок кожної ваги в загальну помилку).

Потім ці диференціали множаться на число, так звану швидкість навчання або learning rate (η).

Отриманий результат потім віднімається з відповідних ваг, у результаті вийдуть такі оновлені ваги:

– w1 = w1 - (η * ∂ (err) / ∂ (w1));
– w2 = w2 - (η * ∂ (err) / ∂ (w2));
– w3 = w3 - (η * ∂ (err) / ∂ (w3)).

Можна помітити що цей процес навчання залежить від деяких зовнішніх параметрів, а саме від:

– функції активації;

– швидкості навчання;

– функції втрат.

Цю сукупність параметрів часто називають гіперпараметрами і їх особливість в тому що ці параметри необхідно налаштовувати самостійно.

Функція активації – це один з ключових факторів що впливає на поведінку мережі. Почасти, вона визначає, які нейрони будуть активовані, іншими словами вона визначає інформацію що буде передаватися наступним шарам.

Без функцій активації глибокі мережі втрачають значну частину своєї здатності до навчання. Нелінійність цих функцій відповідає за підвищення ступеня свободи, що дозволяє узагальнювати проблеми високої розмірності в більш низьких вимірах. Нижче наведені приклади поширених функцій активації:

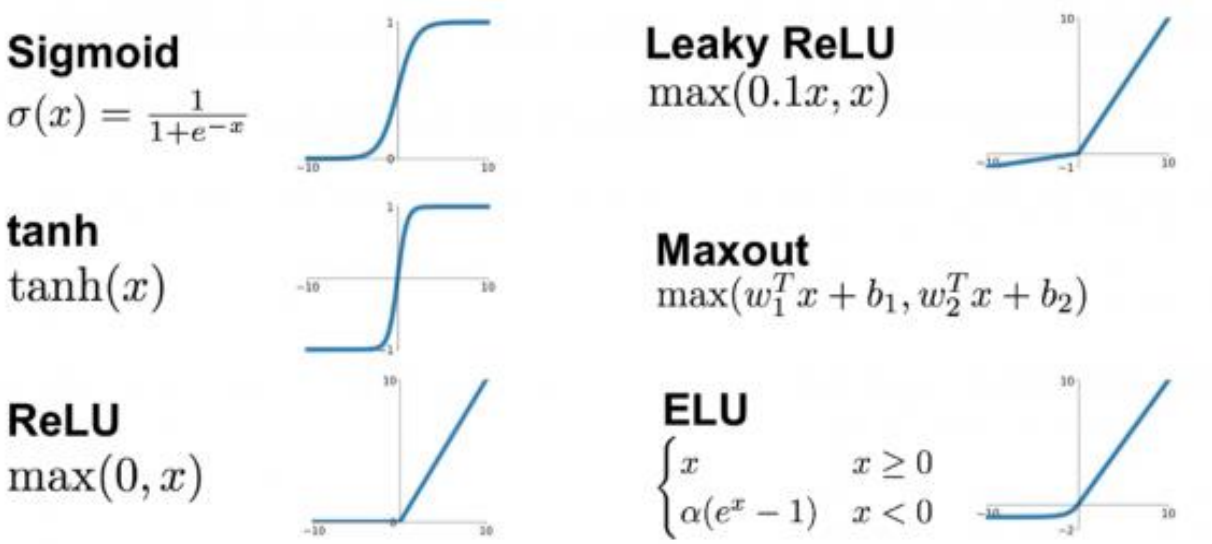

Рисунок 2.5 – Найбільш поширені функції активації

Швидкість навчання є дуже важливим гіперпараметром. Якщо швидкість навчання занадто мала, то навіть після навчання нейронної мережі протягом тривалого часу вона буде далека від оптимальних результатів. Візуалізація використання занадто малої швидкості навчання наведено на рисунку 2.6

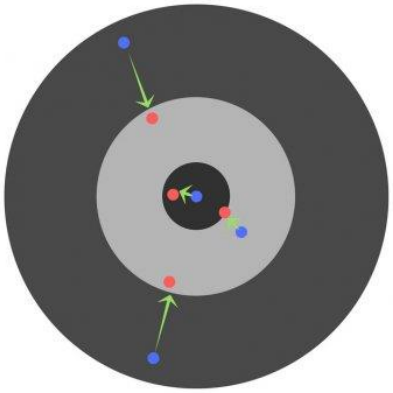

Рисунок 2.6 – Приклад замалої швидкості навчання

У той же час, завелика швидкість навчання теж має негативні наслідки – незважаючи на те що на навчання буде витрачено менше часу, є великі шанси того що ваги у процесі такого навчання «проскочать» свої оптимальні значення, візуалізація такого випадка наведена на рисунку 2.7.

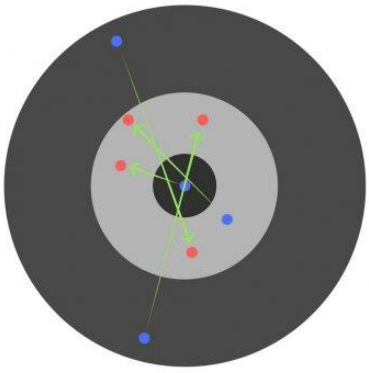

Рисунок 2.7 – Приклад завеликої швидкості навчання

Функція втрат знаходиться в центрі нейронної мережі. Вона використовується для розрахунку помилки між реальними і отриманими відповідями. Авжеж наша мета це мінімізація цієї помилки. Таким чином, функція втрат ефективно наближає навчання нейронної мережі до цієї мети.

Функція втрат вимірює «наскільки гарна» нейронна мережа щодо даної навчальної вибірки і очікуваних відповідей. Вона також може залежати від таких змінних, як ваги і зміщення.

Функція втрат одномірна і не є вектором, оскільки вона оцінює, наскільки добре нейронна мережа працює в цілому.

Найбільш широковідомі функції втрат це:

– квадратична (середньоквадратичне відхилення);

– крос-ентропія;

– експоненціальна (AdaBoost);

– відстань Кульбака - Лейблера або приріст інформації.

Функція втрат в нейронної мережі повинна відповідати двом головним умовам:

– функція втрат повинна бути записана як середнє;

– функція втрат не повинна залежати від будь-яких активаційних значень нейронної мережі, крім значень, які видаються на виході.

Функція витрат розраховується на кожній епохі і повинна йти на спад протягом процесу навчання.

Крім того, функція втрат може виступати у якості індикатору для уникнення такого явища як перенавчання, або оверфіт – негативного явища, що виникає, коли алгоритм навчання виробляє передбачення, які занадто близько або точно відповідають конкретному набору даних і тому не підходять для застосування алгоритму до додаткових даних або майбутнім спостереженнями. Порівняння випадків недонавчання, добре навченої моделі і перенавчання на прикладі класифікації за допомогою логістичної регресії наведено на рисунку 2.8.

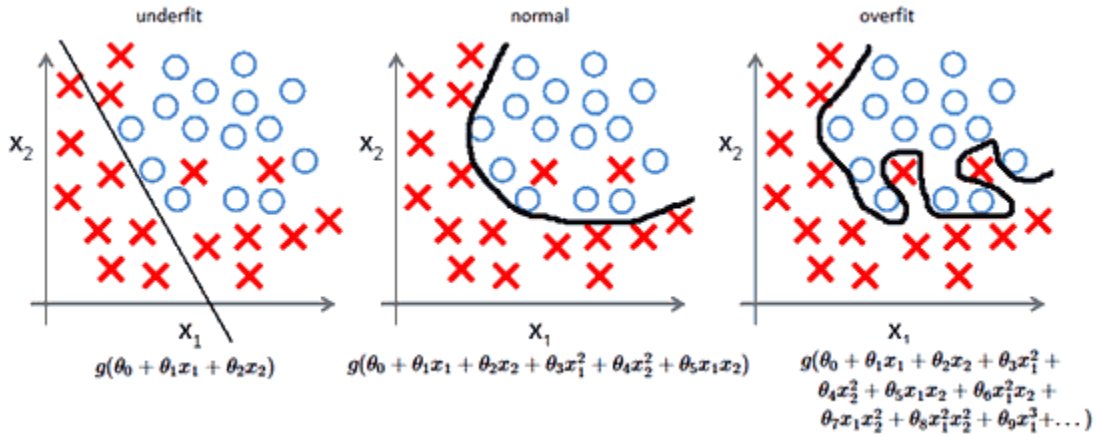

Рисунок 2.8 – Порівняння випадків недонавчання, добре навченої моделі і перенавчання на прикладі логістичної регресії

2.3 Згорткові нейронні мережі

Класичні, повнозв'язні штучні нейронні мережі мають досить суттєві недоліки при вирішенні певних різновидів задач. Головний з них – це величезне число різноманітних вагових коефіцієнтів що будуть визначаються у ході навчання і кількість яких стрімко зростає у відповідності до розмірності вхідних даних. Крім того, за умови використання таких мереж дані зазвичай аналізуються у вигляді одномірного вектору або плоского масиву, що у деяких випадках може потягнути за собою втрату важливої топологічної інформації.

З ціллю усунення таких недоліків повнозв'язних ШНМ був створений новий вид архітектури ШНМ – згорткові нейронні мережі (convolutional neural networks), що являють собою різновид глибоких штучних нейронних мереж.

Їх архітектура була запропонована Яном Лекуном [24] і націлена на ефективне розпізнавання зображень. Свою назву архітектура мережі отримала через наявність ключової для даної архітектури операції згортки, суть якої в тому, що кожен фрагмент зображення множиться на матрицю (ядро) згортки поелементно, а результат сумується і записується в

аналогічну позицію вихідного зображення. В архітектуру мережі закладені апріорні знання з предметної області комп'ютерного зору: піксель зображення сильніше пов'язаний з сусіднім (локальна кореляція) і об'єкт на зображенні може зустрітися в будь-якій частині зображення.

Особливу увагу згорткові нейронні мережі отримали після конкурсу ImageNet, який відбувся в жовтні 2012 року і був присвячений класифікації об'єктів на фотографіях. У конкурсі було потрібно розпізнавання образів в 1000 категорій. Переможець цього конкурсу – Алекс Крижевський, використовуючи згорткову нейронну мережу, значно перевищив за результативністю інших учасників. Цей успіх застосування згорткових нейронних мереж до класифікації зображень привів до безлічі спроб використовувати даний метод для вирішення інших задач. У наш час, згорткові штучні нейронні мережі стали вже класичним підходом для задач пов’язаних з аналізом зображень.

Згорткові нейронні мережі були побудовані як прототип зорової кори мозку. Зорова кора людини має невеликі ділянки клітин, які чутливі до певних областей поля зору. Ця ідея була розглянута дослідниками Хьюбелом і Візелем в 1962 році, вони у ході свого експерименту показали, що деякі мозкові нервові клітини активувались тільки при візуальному сприйнятті границь певної орієнтації. Наприклад, деякі нейрони активувалися, коли сприймали вертикальні границі, а деякі – горизонтальні або діагональні.

У ході своїх експериментів Хьюбел і Візель з'ясували, що всі ці нейрони зосереджені в вигляді стрижневої архітектури і спільною роботою формують візуальне сприйняття. Ця ідея спеціалізованих компонентів всередині системи що призначені для вирішення конкретних завдань (за прикладом клітин зорової кори, які шукають специфічні характеристики) і використовують машини, це і є основною ідеєю згорткових нейронних мереж [25].

Дані, що подаються на вхід згорткової нейронної мережі подаються у вигляді двомірної матриці – що досить зручно для задач пов’язаних з обробкою зображень. У задачі класифікації зображень, на момент своєї появи, згорткові ШНМ перевищили результат інших, класичних ШНМ на 10-15% і станом на сьогодні ще все залишається основним методом обробки зображень. Але є і інші способи застосування згорткових мереж: вони можуть бути застосовані до звуку, який представлений візуально як спектрограма, до аналітики тексту і інших задач, за умови конвертації вхідних даних до вигляду двомірної матриці [26].

Архітектура згорткової нейронної мережі зазвичай являє собою чергування згортальних шарів (convolution layers), субдіскретізірующіх шарів (subsampling layers) і повнозв'язних шарів (fully connected layer) на виході.

Сутність операції згортки, на основі якої і працюють ці шари згортки є наступною: у ході кожної ітерації ми розглядаємо деяку, відносно невелику, область (фрагмент) з вхідних даних, наприклад 3x3, і використовуємо відносно нього деяке ядро згортки яке являє собою матрицю аналогічного розміру, відносно розглядаємої ділянки вхідних даних.

Це ядро згортки поелементно множтися на кожен фрагмент вхідних даних, а вже після завершення цієї операції всі отримані за допомогою цього множення елементи складаються. При цьому кожний з нейронів працює зі своїм фрагментом вхідних даних, який у свою чергу вже може частково перетинатися із фрагментом сусіднього. У подальшому, у нейроні буде застосована деяка функція активації відносно цього розрахованого значення.

Таким чином, ядро являє собою фільтр або вікно, яке ковзає по всій області попередньої карти і знаходить певні ознаки об'єктів.

Схематичне зображення операції згортки наведена на рисунку 2.9.

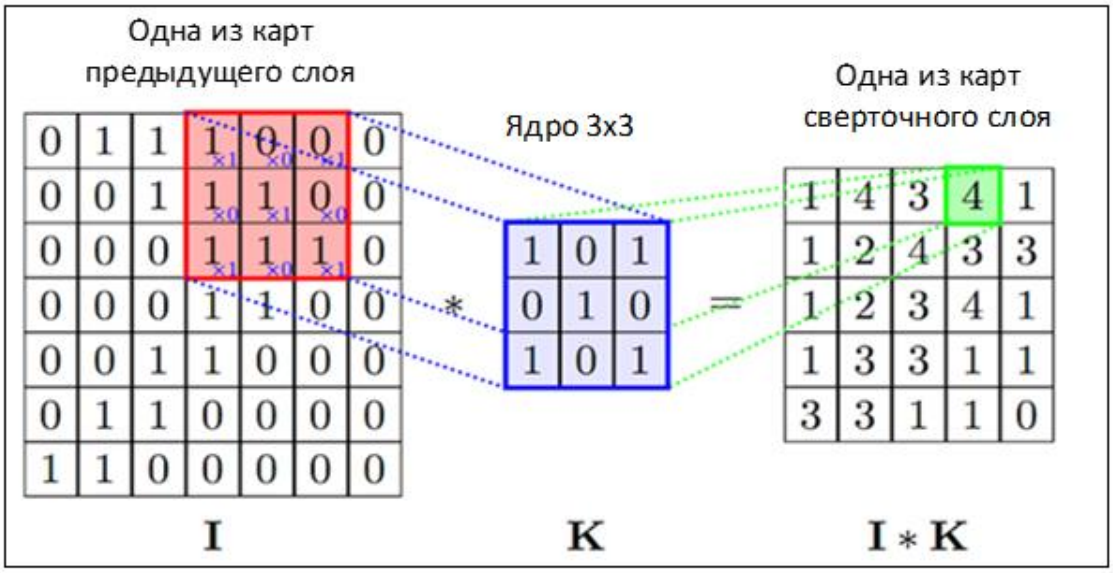

Рисунок 2.9 – Схематичне зображення операції згортки

Слід зауважити що такий обхід вхідних даних у ході операції згортки може зіткнутися с проблемою на границях зображення і зазвичай цю проблему вирішують за допомогою спеціального прийому під назвою Padding.

Padding – це техніка ціль якої вирішити проблему можливої обрізки нашого ковзаючого вікна на границях вхідної матриці. Крайні пікселі ніколи не виявляються в центрі вікна (ядра), тому що для цього вікна немає значень за границями зображення. Це зовсім не ідеальний варіант, так як ми хочемо, щоб розмір на виході дорівнював вхідному.

Padding додає до країв підроблені (fake) пікселі (зазвичай нульового значення, внаслідок цього до них застосовується термін «zero padding», але також можливі варіанти розрахунку значень цих пікселів на основі значень на границі нашої матриці). Таким чином, ядро при прослизанні дозволяє непідробним пикселям знаходитися в центрі ядра, а потім поширюється на підроблені пікселі за межами краю, створюючи вихідну матрицю того ж розміру, що і вхідна.

Приклад використання техніки padding наведена на рисунку 2.10.

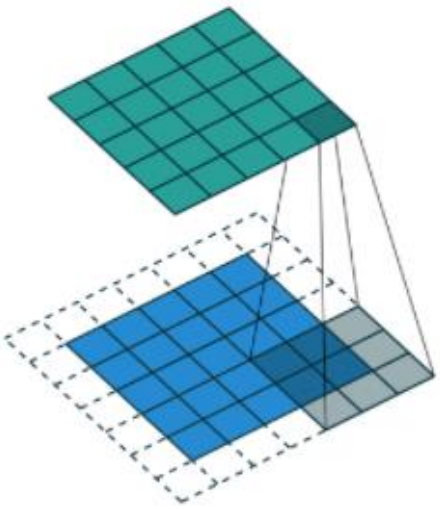

Рисунок 2.10 – Додавання відступу на границях матриці при операції згортки

Слід відмитити, що згорткові мережі використовують однакові ядра згортки для обробки різних областей вхідних даних, що призводить до того що певна характеристика або ознака шукається на різних ділянках вхідного зображеня. Якщо така шукана характеристика наявна у деякому фрагменті вхідних даних що аналізується у цей момент вікном пошуку, операція згортки на виході видасть число що буди мате відносно більше значення. Напроти, у випадку якщо шукана характеристика відсутня, вихідне число буде меншим. Крім того, використання однакових вагових коефіцієнтів у ядрах згортки значно зменшує загальну кількість вагових коефіцієнтів значення яких визначаються у ході навчання нейроної мережі

У більшості випадків розмір ядра знаходится в межах від 3x3 до 11x11. Як правило, замаленький розмір ядра призводить до того що воно не зможе виділити необхідні ознаки у вхідних даних і навпроти, якщо ця розмірність занадто велика, черезмірно збільшується кількість зв’язків між нейронами, що призводить до ускладнення навчання [27]. Крім того, має місце залежність розмірів ядра згортки відносно формату вхідних даних так як для даних більш високої розмірності у свою чергу потребується більша розмірність ядера згортки. Слід відмітити і те, що розмірність ядер

згортки намагються обрати такий що розмір карт згорткового шару був у свою чергу парним з метою уникнення втрати інформації у процесі зменшення розміру у шарах субдискретизації, процес роботи яких буде описаний нижче.

В нейронних мережах розмірність ядер згортки розраховується у автоматичному режимі у процесі навчання – штучна нейронна мережа самостійно обирає потрібну розмірність ядра згортки у відповідності до оброблюємих вхідних даних.

Слід зазначити що розмір ядра згорткової нейронної мережі визначає кількість ознак, що об'єднуються для отримання нового ознаки на виході.

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

Число карт визначається задачею що вирішується за допомогою ШНМ. Більш комплексні задачі потребують більшого числа карт ознак. З ростом кількісті карт підвищується якість розпізнавання але у той же час також збільшується і обчислювальна складність навчання такої мережі. Зазвичай їх кількість збільшуюється після кожного шару. Перші шари мережі що мають меншу кількість карт ознак виділяють більш прості ознаки (наприклад, межі об’єктів на зображенні), а вже наступні шари виділяють більш складні ознаки, базуючись на простих [26], [27].

Розмір усіх карт згорткового шару визначається за формулою 2.1.

$$(w,h) = (mW - kW + 1, mH - kH + 1). \quad (2.1)$$

де (w, h) – розмір згорткової карти;

mW – ширина попередньої карти;

mH – висота попередньої карти;

kW – ширина ядра згортки;

kH – висота ядра згортки.

У подальшому, переслідуючи мету зменшення розмірності в згорткових нейронних мережах використовуються шари субдискретизації (subsampling). Головна мета таких шарів це зменшення розмірності карт попереднього шару. У випадках коли на попередніх операціях згортки вже були успішно виявлені якісь ознаки, можна сказати що у рамках подальших етапів обробки настільки докладні дані вже для нас черезмірні і саме тому вони будуть ущільнені до менш докладних. Крім того, така фільтрація вже непотрібних деталей допомагає уникати перенавчання. Шари субдискретизації подібно до згорткових мають у наявності карти, кількість яких співпадає з попереднім згортковим шаром.

Сутність операцій субдискретизації полягає у зменшенні розмірності сформованих карт ознак. Кожний нейрон з шару субдискретизації має підключення до декількох нейронів з попереднього шару, у більшості випадків до фрагменту з розмірами 2x2 нейрони, але і можливе використання фрагментів і інших розмірностей. У CNN архітектурі мережі у більшшості випадків припускається те що інформація про сам факт наявності певної шуканої ознаки значно важливіша за точне знання її координат, тому з кількох сусідніх нейронів карти ознак за допомогою певної функції (наприклад максимуму, коли ми обираємо лише максимальне значення, а інші ігноруються) розраховується лише одне значення, яке у свою чергу і приймається за один нейрон вже ущільненої карти ознак, яка завдяки цьому ущільненню вже має меншу розмірность.

Окрім наведеної вище функції максимуму, у якості функцій субдискретизації можливе використання і інших функцій – наприклад досить популярними є функції середнього значення і L2-нормування. Але все ж таки практика показує що у більшості практичних виипадків найкращим чином проявляють себе саме шари субдискретизації що використовують функцію максимуму, яка найбільш яскраво виділяє шукані ознаки і тому використовуються у більшості типових архітектур [28].

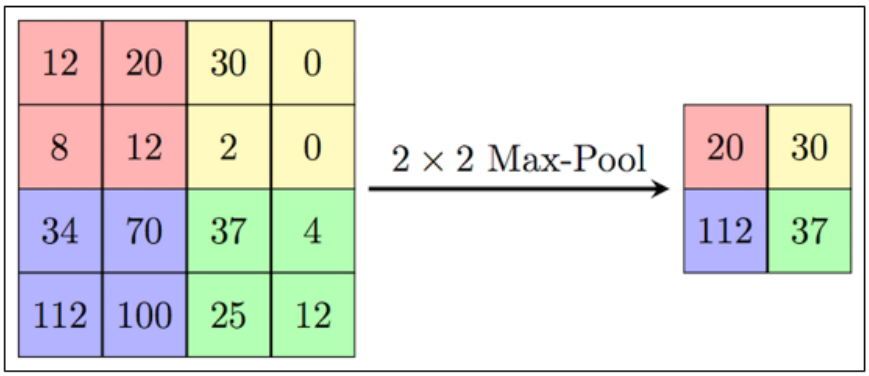

Рисунок 2.11 – Робота шару субдискретизації з функцією максимуму

У більшості випадків, кожна з карт має розмірність ядер 2x2, яке очевидно зменшує розмірність попередніх карт згорткового шару у 2 рази. У той же час, у випадку великих об’ємів (розмір) вхідних даних може мати місце використання ядер шарів субдискретизації більших розмірів, наприклад 4x4, але подібне зростання розмірів ядер слід застовувати уважно, так як воно може призвести до втрати великих обсягів інформації. При цьому слід зазначити, що у ході сканування ядрами шарів субдискретизації карт попереднього шарів, на відмінну від схожої операції на згортковому шарі, скануюче ядро шару субдискретизації не перетинається у ході свого «ковзання».

Слід зазначити що замість таких шарів субдискретизації для вирішення задачі зменшення розмірности карт ознак можуть використовуватися інші прийоми, наприклад використання шагу (stride).

Ідея stride полягає в тому, щоб пропустити деякі області, над якими ковзає ядро. Крок 1 по суті є стандартною згороткою. Крок 2 означає, що прольоти відбуваються через кожні два пікселя, пропускаючи всі інші прольоти в процесі і зменшуючи їх кількість приблизно в 2 рази, крок 3 означає пропуск 3-х пікселів, скорочуючи кількість в 3 рази і т.п. Деякі сучасні мережі, наприклад архітектури ResNet, повністю відмовляються від субдіскрітізірующіх шарів у внутрішніх шарах і використовують замість них stride для зменшення розмірності карт на виході.

В будь-якому випадку, після декількіх циклів чергування шарів згортки і субдискретизації настає час класифікатора, який у свою чергу базуючись на ознаках знайденими згортковою мережею виконує беспосередньо классифікацію – визначення до якого конкретного вихідного класу належить об’єкт. На початку є необхідність перетворення отриманого двовимірного представлення в плоске. В якості класифікатора зазвичай використовується послідовність повнозв’язних шарів. Вихідний, шар містить кількість нейронів що точно відповідає кількості вихідних класів, кожний з яких говорить про належність об’єкту до певного класу.

У сукупності, процес роботи згорткової нейронної мережі можна інтерпретувати як перехід від більш конкретних і простих особливостей або ознак даних до більш вискорівневих і абстрактних, схожих на шукані, по мірі проходження все нових, більш глибоків шарів мережі, аж до виділення остаточних вихідних понять високого рівня. У ході цього процесу згорткові мережі виконують самоналаштування і самостійно виробляє необхідну ієрархію абстрактних ознак (у вигляді послідовності карт ознак), фільтруючи незначнідеталі і навпроти виділяючи істотні.

Авжеж така інтерпретація має скоріше метафоричний і ілюстративний характер. По факту ці «ознаки», що виробляються згорткової мережею у більшості випадків настільки малозрозумілі і важкі для інтерпретації дюдиною, що на практиці при використанні подібних систем людям не рекомендується навіть намагатися зрозуміти зміст цих ознак або спробувати їх «підправити» у ручному режимі, а у якості заміни таких втручань слід приділати увагу вдосконаленню структури і архітектури мережі у сукупності з її параметрами навчання.

У випадках коли побудуванна модель ігнорує істотні, цільові ознаки слід замислитися про достатність і якісність даних для навчання, або розглянути можливість того що структура мережі має суттєві недоліки і як наслідок система не спроможна виділити достатньо ефективні ознаки для даних явищ [28].

3 ЕКСПЕРИМЕНТАЛЬНЕ МОДЕЛЮВАННЯ ТА НАВЧАННЯ МОДЕЛІ

3.1 Вибір програмних засобів

В якості основної мови програмування для виконання поставленого завдання була обрана мова Python.

Станом на сьогоднішній день саме Python став стандартною мовою програмування у сфері машинного навчання. Навіть якщо не виділяти окремо сферу машинного навчання, Python – це дуже популярна високорівнева мова програмування з динамічною типізацією, що за рівнем популярності навіть конкурує с такою мовою як Javascript. Вона досить проста для написання і читання коду, її використання знижує загальну вартість розробки та обслуговування програм.

Крім простоти, у Python є ще один плюс – він досить легко взаємодіє з іншими мовами, особливо з C і C++. Тому у процесі популяризації машинного навчання він досить швидко був доповнений безліччю фреймворків та бібліотек які спрощують процес написання коду і скорочують час на розробку. Наприклад у наукових розрахунках дуже широко використовується Numpy, в просунутих обчисленнях – SciPy, в добуванні і аналізі даних – SciKit-Learn. Ці бібліотеки працюють для таких фреймворків, як TensorFlow, CNTK і Apache Spark. Існує фреймворк для Python, розроблений спеціально для машинного навчання – це PyTorch.

Фахівці зайняті у сфері машинного навчання та data science у більшості випадків займаються збіром, аналізом і систематизацією даних, а потім, на основі отриманої інформації створюють алгоритми і навчають моделі штучного інтелекту. Python найкраще підходить для виконання таких завдань, тому що за синтаксисом він досить простий і зрозумілої в

порівнянні з іншими мовами. І крім того він має непогану продуктивність при обробці даних.

Ще одна перевага Python – це гарний рівень підтримки і якісна документація. Зараз існує безліч корисних ресурсів про Python, на яких програміст може отримати допомогу і консультацію, перебуваючи на будь-якому етапі розробки.

У якості основного середовища розробки на мові Python було використано Jupyter Notebook. Jupyter Notebook є графічною веб-оболонкою для IPython, яка розширює ідею консольного підходу до інтерактивних обчислень.

IPython, на якому і базується Jupyter Notebook, дозволяє підключатися безлічі клієнтів до одного обчислювальному ядру і, завдяки своїй архітектурі, може працювати в паралельному кластері.

У Jupyter notebook ви можете 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.п.

Таким чином, Jupyter Notebook – це доволі потужний інструмент для розробки та подання проектів у сферах Data Science і Machine Learning в інтерактивному вигляді. Він дозволяє зручно об'єднати програмний код, результати його виконання, додаткову довільну інформацію у різноманітних форматів у єдиний зручний документ, при цьому дозволяючи зручно структурувати всю цю інформацію шляхом розбиття вхідного коду на послідовність окремих кроків.

Такий покроковий, ітеративний підхід забезпечує досить швидкий, послідовний процес розробки, оскільки вивід для кожного блоку показується миттєво. Саме тому інструмент став настільки популярним в середовищі Data Science і Machine Learning за останній час. Яскравим прикладом цього може слугувати те що велика частина Kaggle Kernels

(роботи учасників конкурсів на платформі Kaggle) на сьогодні створені за допомогою Jupyter Notebook.

З метою полегшення процесу аналізу даних використовувалась бібліотека Pandas. Станом на сьогодні, Pandas це найбільш популярна бібліотека для подібних задач для мови Python.

Цей програмний пакет забезпечує швидкі, гнучкі та виразні структури даних, покликані зробити роботу зі структурованими (табличними, багатовимірними, потенційно неоднорідними) та даними часових рядів одночасно легкою та інтуїтивно зрозумілою. Він має мету стати фундаментальним будівельним елементом високого рівня для практичного аналізу даних у реальному світі на мові Python. Окрім того, вона має більш широку мету – стати найпотужнішим та найгнучкішим інструментом аналізу / маніпуляції з відкритим кодом, доступним для більшості сучасних мов програмування.

Pandas добре підходить для роботи з широким спектром різноманітних форматів даних:

– табличні дані з неоднорідно набраними стовпцями, як у таблиці SQL або таблиці Excel;

– впорядковані та невпорядковані (не обов'язково фіксовані частоти) дані часових рядів;

– дані довільної матриці (однорідно введені або неоднорідні) з мітками рядків і стовпців;

– дані довільної матриці (однорідно введені або неоднорідні) з мітками рядків і стовпців;

– будь-яка інша форма спостережних / статистичних наборів даних.

З метою спрощення роботи по веденню розрахунків у матричному та векторному вигляді була використана бібліотека NumPy. Головні можливості що надає використання NumPy це підтримка багатовимірних масивів (включаючи матриці) у сукупності з підтримкою високорівневих математичних функцій, призначених для роботи з багатовимірними

масивами. Проблема стандартних, вбудованих у Python математичних алгоритмів така ж як і проблема будь яких алгоритмів що були реалізовані на інших інтерпретованих мовах – вони часто працюють набагато повільніше тих же алгоритмів, реалізованих на компільованих мовах (наприклад C або Java). Бібліотека NumPy вирушує цю проблему швидкодії інтерпретованих мов і надає реалізації обчислювальних алгоритмів (у вигляді функцій і операторів), що оптимізовані для роботи з багатовимірними масивами. В результаті будь-який алгоритм, який може бути виражений у вигляді послідовності операцій над масивами (матрицями) і реалізований з використанням NumPy, працює так само швидко, як еквівалентний код, що виконується у середовищі MATLAB.

У якості основної бібліотеки що використовується для створення і навчання нашої штучної нейронної мережі був обраний Keras. Keras надає простий і зручний спосіб створення моделей глибокого навчання. Її автор, François Chollet, розробив її для того, щоб максимально прискорити і спростити процес створення нейронних мереж. Він зосередив свою увагу на розширюваності, модульності, мінімалізмі і інтеграції з мовою програмування Python. Keras можна використовувати з GPU і CPU; вона підтримує як і Python 2, так і Python 3. Keras внес великий внесок в популяризацію використання глибокого навчання і штучного інтелекту, шляхом спрощення процесу роботи з сучасними алгоритми глибокого навчання які раніше були досить важкими для використання.

3.2 Підготовка набору даних для навчання

Для навчання штучної нейронної мережі необхідно зібрати набір даних, датасет на якому мережа буде здійснювати своє навчання і оцінку своєї роботи, наш фінальний датасет збирається з 2-х різних джерел, одне з яких буде використане для отримання зображень рентгенограм випадків

вірусної пневмонії COVID-19, а інше для випадків бактеріальної пневмонії і випадків коли пневмонія не виявлена, ці 3 вида випадків відповідають вихідним класам нашої штучної мережі.

Для отримання зразків зображень рентгенограм з випадками вірусної пневмонії COVID-19 був використаний датасет COVID-19 image data collection який можна знайти за адресою https://github.com/ieee8023/covid-chestxray-dataset.

Після фільтрації цього датасету за параметрами «finding» – назва захворювання, нас цікавить тільки «COVID-19» і «modality», цей параметр повинен бути «X-ray» – з метою фільтрації КТ зображень, що присутні у цьому датасету. Після фільтрації по цим 2-м параметрам отримуємо 253 зображення рентгенограм з випадками COVID-19.

Для отримання зразків зображень рентгенограм з випадками бактеріальної пневмонії був використаний наступний набір https://data.mendeley.com/datasets/rscbjbr9sj/2. Цей набір даних вже розділений на папки «NORMAL» і «PNEUMONIA» які і будуть використані для отримання випадків коли пневмонія не виявлена і випадків бактеріальної пневмонії відповідно.

З метою збереження балансу набору даних відносно вихідних класів ми беремо тільки по 253 (відповідно до числа зображень рентгенограм з випадками COVID-19) випадково відібраних зображень рентгенограм випадків відсутності пневмонії і бактеріальної пневмонії.

Перед безпосереднім використанням отриманий датасет піддається додатковому препроцесінгу – приведення вхідних зображень до розміру 224*224 пікселів, зміна цвітового простору на RGB і розділення всього датасету на тренувальну і валідаційну вибірку у співвідношенні 80% / 20% відповідно.

Додатково, для доповнення даних було використано клас генератора зображень (ImageDataGenerator) з бібліотеки Keras з метою додавання випадкової ротації зображення в межах 15%.

3.3 Розробка топології мережі

У більшості випадків навчання штучної нейронної мережі може зайняти досить довгий час. До вирішення цієї проблеми є 2 основних підходи, перший являє собою звичайне нарощення обчислювальних потужностей, у більшості випадків для таких цілей використовується GPU, але у останній час наряду з ними стрімко зростає популярність використання більш спеціалізованих інтегральних схем спеціального призначення (ASIC) і одного їх різновиду, TPU – тензорних процесорів .

Інший можливий варіант – використання готової штучної нейронної мережі, яка вже була навчена до нас. Такі мережі називають попередньо навченими. Вони вже раніше були успішно навчені під вирішення деякої задачі протягом довгого часу. Після завершення навчання дані про архітектуру цієї мережі і її розраховані вагові коефіцієнти були збережені і зроблені загальнодоступними. Ці дані можна завантажити вже у власний проект та використовувати для вирішення певних своїх задач.

В бібліотеці Keras попередньо навчені нейронні мережі містяться в модулі Keras Applications. Keras Applications містить значну кількість популярних попередньо навчених мереж, більша частина з них використовується для розпізнавання об’єктів на зображеннях.

З метою оптимізації процесу навчання нашої моделі було прийняте рішення про використання технології transfer learning і побудови нашої згорткової штучної нейронної мережі на основі однієї з попередньо тренованих штучних нейронних мереж, представлених у модулі Keras Applications. Незважаючи на те, що перелік же містить деякі данні щодо результативності таких мереж, ці данні можуть досить сильно змінитися в залежності від контексту і предметної області використання таких мереж. Перелік цих, попередньо тренованих мереж, що містяться у модулі Keras Applications наведений на рисунку 3.1.

Available models

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
EfficientNetB0	29 MB	-	-	5,330,571	-
EfficientNetB1	31 MB	-	-	7,856,239	-
EfficientNetB2	36 MB	-	-	9,177,569	-
EfficientNetB3	48 MB	-	-	12,320,535	-
EfficientNetB4	75 MB	-	-	19,466,823	-
EfficientNetB5	118 MB	-	-	30,562,527	-
EfficientNetB6	166 MB	-	-	43,265,143	-
EfficientNetB7	256 MB	-	-	66,658,687	-

Рисунок 3.1 Попередньо треновані ШНМ у модулі Keras Applications

Оскільки теоретичним шляхом досить важко оцінити наскільки кожна з перерахованих на рисунку 3.1 мереж підійде для вирішення нашої задачі було прийняте рішення о підборі найкращої для вирішення поставленої задачі з них шляхом тестування. У зв’язку з тим що деякі типи штучних попередньо навчених мереж представлені у великій кількості модифікацій було прийняте рішення не тестувати всі модифікації а найбільш потужні і нові з них. Таким чином, для тестування були відібрані наступні мережі: Xception, VGG16, VGG19, ResNet152V2, InceptionResNetV2, MobileNetV2, DenseNet201, NASNetLarge, EfficientNet81.

У якості функції помилки була обрана бінарна крос-ентропія, яка гарно підходить до задач класифікації зображень, подібних до нашої.

У якості метрики оцінки роботи мережі була обрана точність (accuracy) – доля вірних спроб класифікації нашою штучною мережею.

З 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 Adam.

Для адаптації попередньо навчених мереж до нашої задачі вивід цих мереж проходить через низку шарів наведених на рисунку 3.2.

```
AveragePooling2D(pool_size =(2, 2))(headmodel)
Flatten(name ='Flatten')(headmodel)
Dense(64, activation = 'relu')(headmodel)
Dropout(0.5)(headmodel)
Dense(3, activation = 'softmax')(headmodel)
```

Рисунок 3.2 Останні, вихідні шари побудованої ШНМ

Результати тестування використання різних попередньо тренованих ШНМ наведені у таблиці 3.1.

Таблиця 3.1 – Точність і помилка мережі при використанні різних попередньо тренованих ШНМ

Назва ШНМ	Точність	Помилка
Xception	0.9310	0.1105
VGG16	0.9365	0.1145
VGG19	0.9750	0.0633
ResNet152V2	0.9620	0.1921
InceptionResNetV2	0.9560	0.1336
MobileNetV2	0.9410	0.1732
DenseNet201	0.9375	0.1385
NASNetLarge	0.9410	0.1539
EfficientNet81	0.9475	0.1532

За результатами тесту було прийнято рішення щодо використання попередньо навченої мережі VGG19. Ця ШНМ комбінує достатньо гарні показники результативності (точність і значення функції втрат) і одночасно невелику кількість шарів у своїй архітектурі, що забезпечує гарні показники швидкодії нашої мережі.

З метою вибору оптимального оптимізатора для нашої задачі було проведено порівняння результатів роботи мережі на тестових даних при використанні трьох найбільш популярних оптимізаторів доступних у Keras – SGD, Rmsprop та Adam. Результати тестування відображено у таблиці 3.2.

Таблиця 3.2 – Точність і помилка мережі при використанні різних оптимізаторів

Оптимізатор	Точність	Помилка
SGD	0.9465	0.1278
Rmsprop	0.9420	0.1383
Adam	0.9750	0.0633

Як видно з цієї таблиці найкращі результати наша ШНМ показує при використанні оптимізатора Adam.

Крім того, дослідним шляхом була обрана оптимальна кількість епох навчання. На результатах цього тесту було виконане порівняння результатів навчання мережі при різній кількості епох – 20, 40, 60, 80, 100. За результатами цього досліду був зроблений висновок що оптимальна кількість епох навчання нашої ШНМ складає 60 епох, тому що на меншій кількості епох наша мережа демонструє нижчу точність на тестовій вибірці, а на більших значеннях кількості епох навчання цей параметр вже не демонструє зростання точності.

Остаточна архітектура нашої побудованої ШНМ створеної за допомогою бібліотеки Keras наведено на рисунку 3.3.

```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
average_pooling2d_1 (Average (None, 3, 3, 512)         0
_________________________________________________________________
Flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                294976
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 195
=================================================================
Total params: 20,319,555
Trainable params: 295,171
Non-trainable params: 20,024,384
```

Рисунок 3.3 – Структура побудованої ШНМ

3.4 Аналіз результатів

В результаті проведених робіт була створена модель для реалізації класифікації рентгенівських зображень. Вона була реалізована за допомогою мови програмування Python і бібліотеки машинного навчання Keras і дозволяє подати на свій вхід рентгенівські зображення, кожне з яких у подальшому буде класифіковане нейронною мережею і буде віднесене до одного з трьох вихідних класів – бактеріальна або вірусна пневмонія і норма.

Завдяки невеликому об’єму вхідних даних і використанню попередньо тренованої мережі VGG-19 навчання моделі на платформі Kaggle зайняло досить невеликий час – 16 хвилин і 51 секунд.

Хоча якісне навчання типової згорткової ШНМ для вирішення подібних задач зазвичай потребує більшого вхідного датасету і часу, використання технології transfer learning у цьому випадку допомогло в оптимізації таких витрат. Як результат, побудована модель демонструючи високу швидкодію у процесі своєї роботи у сукупності з високими показниками точності була навчена на відносно невеликому датасеті за досить короткий час у 17 хвилин (за умови використання графічного прискорювача). Вклад в ці показники отриманий завдяки використанню попередньо тренованої мережі є досить суттєвим – це можна побачити проаналізувавши данні представлені вище, на рисунку 3.3, з більш ніж 20 мільйонів параметрів ШНМ ми навчали лише 300 тисяч.

Після успішного завершення навчання побудованої штучної згорткової мережі, загальна точність виміряна на навчальному набору даних склала 97.50%, а на перевірочній вибірці (validation data), яка не використовується у процесі навчання – 96.71%. У той же час значення функції втрат (loss function) відповідно склали 0.0374 для навчальної вибірки і 0.0633 для перевірочної вибірки (validation loss). Консольні логи останніх епох навчання бібліотеки Keras наведені на рисунку 3.4.

```
Epoch 1/100
75/75 [==============================] - 8s 103ms/step - loss: 0.4393 - accuracy: 0.7012 - val_loss: 0.2402 - val_accuracy: 0.8750
Epoch 2/100
75/75 [==============================] - 8s 108ms/step - loss: 0.2649 - accuracy: 0.8397 - val_loss: 0.1466 - val_accuracy: 0.9145
Epoch 3/100
75/75 [==============================] - 7s 96ms/step - loss: 0.1919 - accuracy: 0.9015 - val_loss: 0.1086 - val_accuracy: 0.9605
Epoch 4/100
75/75 [==============================] - 7s 95ms/step - loss: 0.1902 - accuracy: 0.8948 - val_loss: 0.1174 - val_accuracy: 0.9671
Epoch 5/100
75/75 [==============================] - 7s 98ms/step - loss: 0.1733 - accuracy: 0.9098 - val_loss: 0.1201 - val_accuracy: 0.9408
Epoch 6/100
75/75 [==============================] - 7s 94ms/step - loss: 0.1401 - accuracy: 0.9349 - val_loss: 0.0760 - val_accuracy: 0.9671
Epoch 7/100
75/75 [==============================] - 8s 103ms/step - loss: 0.1382 - accuracy: 0.9316 - val_loss: 0.1185 - val_accuracy: 0.9276
Epoch 8/100
75/75 [==============================] - 7s 99ms/step - loss: 0.1297 - accuracy: 0.9249 - val_loss: 0.0724 - val_accuracy: 0.9605
Epoch 9/100
75/75 [==============================] - 7s 99ms/step - loss: 0.1263 - accuracy: 0.9282 - val_loss: 0.0847 - val_accuracy: 0.9737
Epoch 10/100
75/75 [==============================] - 7s 98ms/step - loss: 0.1131 - accuracy: 0.9366 - val_loss: 0.0757 - val_accuracy: 0.9605
Epoch 11/100
75/75 [==============================] - 8s 105ms/step - loss: 0.1117 - accuracy: 0.9466 - val_loss: 0.0922 - val_accuracy: 0.9539
Epoch 12/100
75/75 [==============================] - 7s 99ms/step - loss: 0.1154 - accuracy: 0.9432 - val_loss: 0.0625 - val_accuracy: 0.9671
Epoch 13/100
75/75 [==============================] - 7s 95ms/step - loss: 0.1050 - accuracy: 0.9366 - val_loss: 0.0666 - val_accuracy: 0.9803
Epoch 14/100
75/75 [==============================] - 7s 99ms/step - loss: 0.1082 - accuracy: 0.9416 - val_loss: 0.0827 - val_accuracy: 0.9605
Epoch 15/100
75/75 [==============================] - 8s 107ms/step - loss: 0.0829 - accuracy: 0.9482 - val_loss: 0.0545 - val_accuracy: 0.9605
Epoch 16/100
75/75 [==============================] - 7s 96ms/step - loss: 0.1227 - accuracy: 0.9332 - val_loss: 0.0666 - val_accuracy: 0.9671
Epoch 17/100
75/75 [==============================] - 7s 98ms/step - loss: 0.0889 - accuracy: 0.9499 - val_loss: 0.0706 - val_accuracy: 0.9737
Epoch 18/100
75/75 [==============================] - 7s 100ms/step - loss: 0.1147 - accuracy: 0.9282 - val_loss: 0.0736 - val_accuracy: 0.9671
Epoch 19/100
75/75 [==============================] - 7s 96ms/step - loss: 0.0876 - accuracy: 0.9399 - val_loss: 0.0553 - val_accuracy: 0.9737
Epoch 20/100
75/75 [==============================] - 8s 106ms/step - loss: 0.0932 - accuracy: 0.9449 - val_loss: 0.0547 - val_accuracy: 0.9539
Epoch 21/100
75/75 [==============================] - 7s 98ms/step - loss: 0.1124 - accuracy: 0.9249 - val_loss: 0.0697 - val_accuracy: 0.9605
Epoch 22/100
75/75 [==============================] - 7s 97ms/step - loss: 0.1013 - accuracy: 0.9332 - val_loss: 0.0481 - val_accuracy: 0.9737
Epoch 23/100
75/75 [==============================] - 7s 93ms/step - loss: 0.1101 - accuracy: 0.9232 - val_loss: 0.0618 - val_accuracy: 0.9539
Epoch 24/100
75/75 [==============================] - 8s 105ms/step - loss: 0.1307 - accuracy: 0.9282 - val_loss: 0.0615 - val_accuracy: 0.9737
Epoch 25/100
75/75 [==============================] - 7s 99ms/step - loss: 0.1175 - accuracy: 0.9316 - val_loss: 0.0545 - val_accuracy: 0.9803
Epoch 26/100
75/75 [==============================] - 7s 96ms/step - loss: 0.1035 - accuracy: 0.9399 - val_loss: 0.0650 - val_accuracy: 0.9539
Epoch 27/100
75/75 [==============================] - 7s 99ms/step - loss: 0.1072 - accuracy: 0.9282 - val_loss: 0.0663 - val_accuracy: 0.9539
Epoch 28/100
75/75 [==============================] - 8s 100ms/step - loss: 0.0937 - accuracy: 0.9416 - val_loss: 0.0548 - val_accuracy: 0.9737
Epoch 29/100
75/75 [==============================] - 7s 97ms/step - loss: 0.0741 - accuracy: 0.9566 - val_loss: 0.0538 - val_accuracy: 0.9605
Epoch 30/100
75/75 [==============================] - 7s 98ms/step - loss: 0.0711 - accuracy: 0.9583 - val_loss: 0.0417 - val_accuracy: 0.9868
Epoch 31/100
75/75 [==============================] - 7s 99ms/step - loss: 0.0708 - accuracy: 0.9533 - val_loss: 0.0664 - val_accuracy: 0.9605
Epoch 32/100
75/75 [==============================] - 7s 96ms/step - loss: 0.0943 - accuracy: 0.9366 - val_loss: 0.0515 - val_accuracy: 0.9671
Epoch 33/100
75/75 [==============================] - 8s 107ms/step - loss: 0.0825 - accuracy: 0.9466 - val_loss: 0.0363 - val_accuracy: 0.9737
Epoch 34/100
75/75 [==============================] - 8s 100ms/step - loss: 0.0922 - accuracy: 0.9399 - val_loss: 0.0779 - val_accuracy: 0.9671
Epoch 35/100
75/75 [==============================] - 7s 96ms/step - loss: 0.0733 - accuracy: 0.9533 - val_loss: 0.0349 - val_accuracy: 0.9868
Epoch 36/100
75/75 [==============================] - 7s 96ms/step - loss: 0.0681 - accuracy: 0.9633 - val_loss: 0.0709 - val_accuracy: 0.9539
Epoch 37/100
75/75 [==============================] - 8s 106ms/step - loss: 0.0815 - accuracy: 0.9499 - val_loss: 0.0492 - val_accuracy: 0.9737
Epoch 38/100
75/75 [==============================] - 7s 99ms/step - loss: 0.0842 - accuracy: 0.9499 - val_loss: 0.0557 - val_accuracy: 0.9605
Epoch 39/100
75/75 [==============================] - 7s 95ms/step - loss: 0.0891 - accuracy: 0.9482 - val_loss: 0.0642 - val_accuracy: 0.9803
Epoch 40/100
75/75 [==============================] - 7s 99ms/step - loss: 0.0642 - accuracy: 0.9583 - val_loss: 0.0347 - val_accuracy: 0.9737
```

Рисунок 3.4 – Вивід бібліотеки Keras в процесі навчання мережі

Було побудовано комбінований графік зміни значення функції втрат і точності мережі для навчального і валідаційного сетів в процесі навчання, цей графік наведений на рисунку 3.5. На даних графіках видно, що під час навчання мережі на кожній з епох помилка на наборі даних для навчання і наборі даних для перевірки поступово зменшується, а точність зростає. Цей прогрес навчання (у вигляді зменшення значення функції помилки і зростання точності) закінчується у районі 40-ї епохи, тому на цьому етапі можна зупинити подальше навчання, так як воно не призводить до поліпшення результативності роботи моделі.

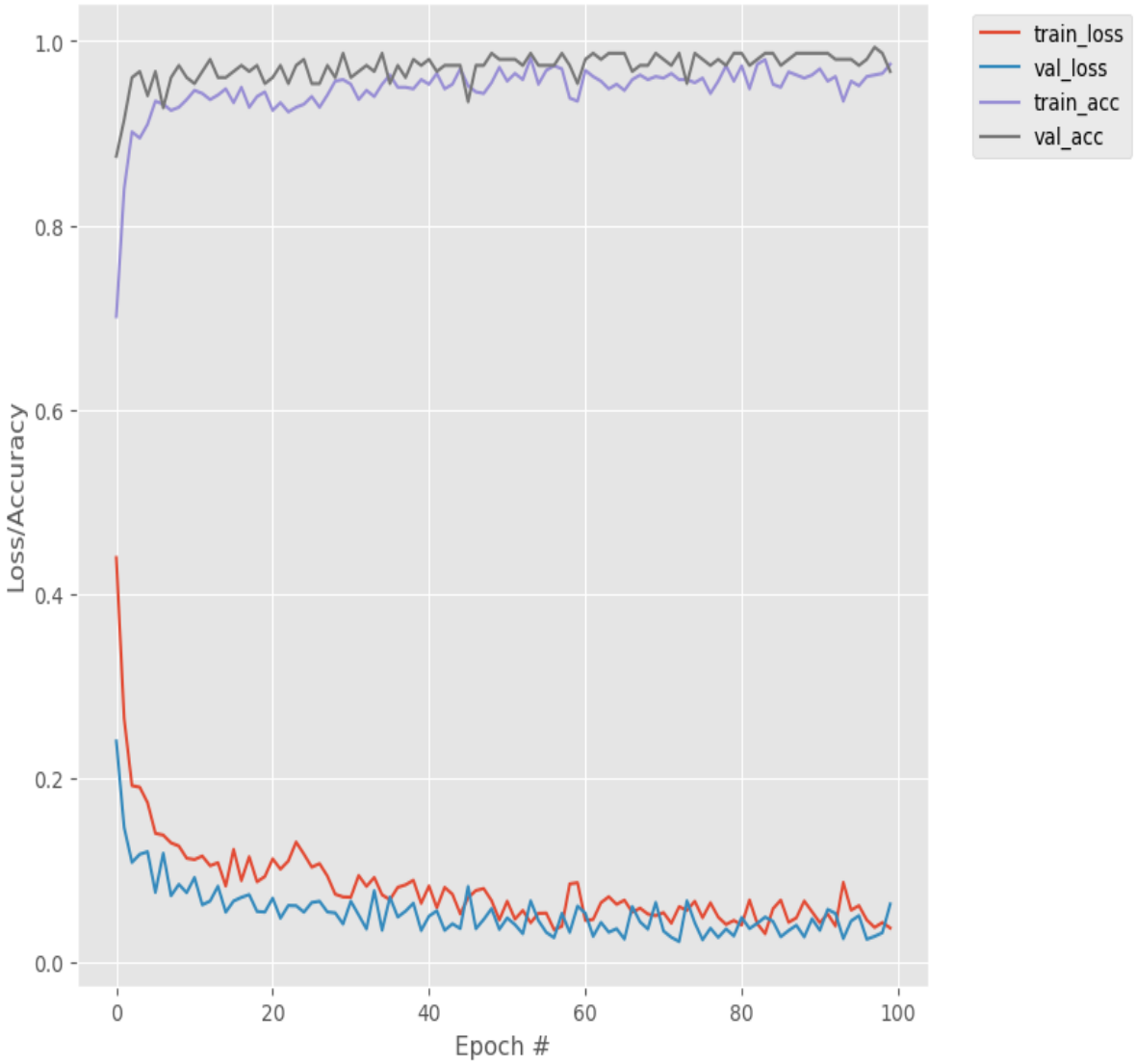

Рисунок 3.5 – Графіки зміни функції помилки і точності в ході навчання

Крім того, були розраховані інші показники що характеризують якість навчання мережі, наприклад точність (precision), відгук (recall) та f-міру (f-measure). Значення цих мір за класами наведені на рисунку 3.6.

```
[INFO] evaluating network...
              precision    recall  f1-score   support

   bacterial       0.98      0.94      0.96        51
      normal       0.96      0.96      0.96        51
       viral       0.96      1.00      0.98        50

    accuracy                           0.97       152
   macro avg       0.97      0.97      0.97       152
weighted avg       0.97      0.97      0.97       152
```

Рисунок 3.6 – Значення мір якості навчання за класами

Розподіл відсотків вірних відповідей за вихідними класами наведений на рисунку 3.7.

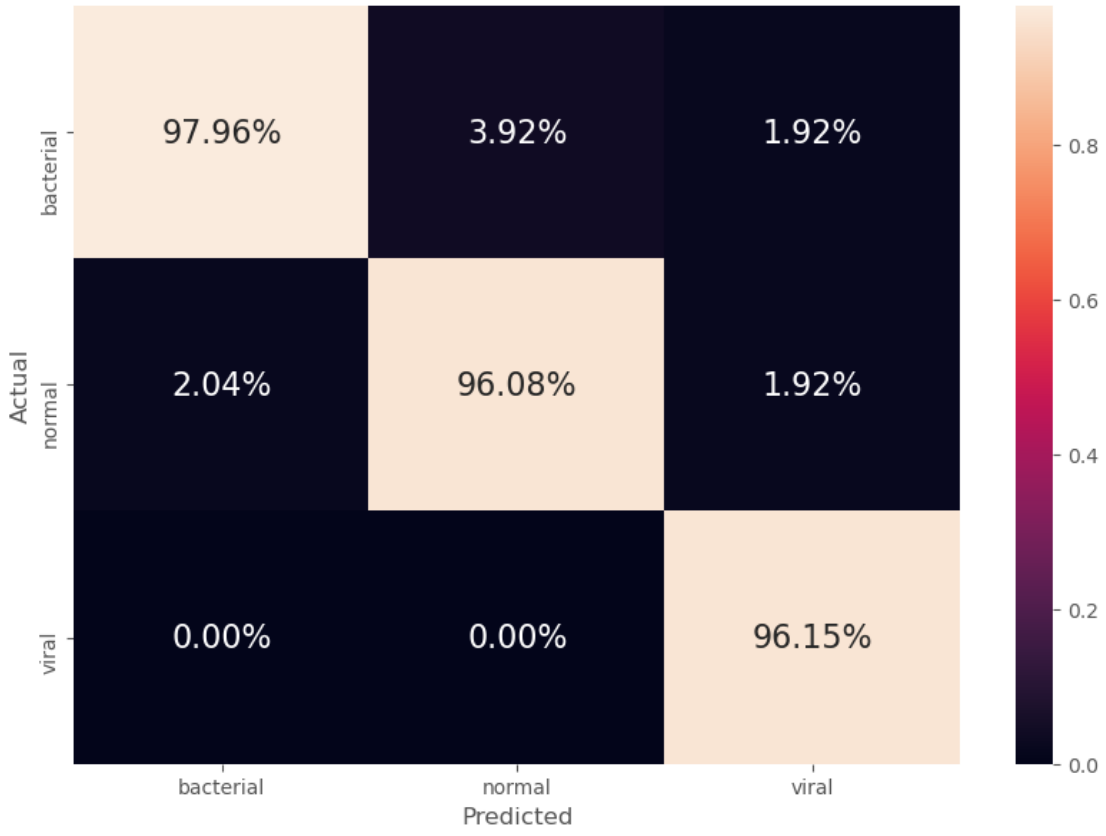

Рисунок 3.7 – Розподіл відсотків вірних відповідей за вихідними класами

Приклад результатів класифікації знімків наведено на рисунку 3.8.

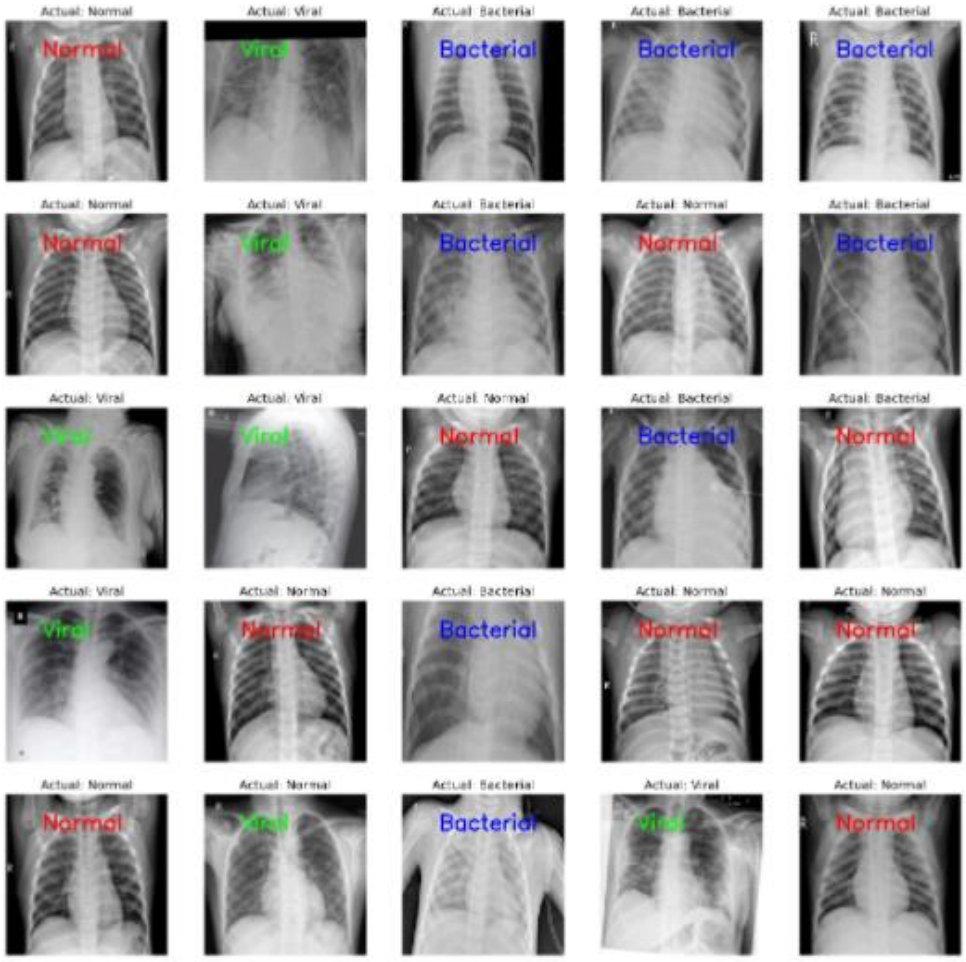

Рисунок 3.8 – Приклад результатів класифікації рентгенівських знімків

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 було виконано. Вимоги що до точності мережі більшої за 90% також виконано, розроблена модель демонструє точність близьку до 96%.

Таким чином, комбінуючи високу швидкодію і точність у сукупності з відносно малим часом необхідним на навчання (завдяки використанню попередньо мережі і transfer learning), розроблена ШНМ становить гідну конкуренцію розглянутим у розділі 1.2 вже існуючим подібним системам.

ВИСНОВКИ

В рамках даної кваліфікаційної роботи було досліджено використання штучних нейронних мереж в задачі діагностики респіраторних захворювань за допомогою аналізу рентгенівських знімків, що передбачає можливість за рентгенівським знімком пацієнта автоматично визначити відсутність патологій або наявність на ньому вірусної або бактеріальної пневмонії.

Було проведено попередній аналіз предметної області, а також огляд актуальності і сучасного стану проблематики респіраторних захворювань як і у світі загалом, так і в Україні. Були розглянуті вже існуючи системи націлені на вирішення подібних задач.

У ході проведення теоретичних досліджень були розглянуті загальні принципи роботи штучних нейронних мереж і методи їх навчання, їх можливості щодо вирішення поставленої задачі класифікації зображень

Після загального огляду різноманітних типів штучних нейронних мереж було прийняте рішення про використання згорткових нейронних мереж як найбільш відповідних и придатних до вирішення поставленої задачі. Був здійснений огляд згорткових нейронних мереж, їх особливостей і принципів роботи.

У подальшому, в ході експериментального моделювання і навчання моделі ми побудували модель на основі згорткової нейронної мережі. Для цього був обраний вже класичний для сфери машинного навчання інструментарій у вигляді мови програмування Python, бібліотеки машинного навчання Keras і додаткових допоміжних бібліотек, таких як Pandas і Numpy. Було зібрано датасет для подальшого навчання побудованої ШНМ що нараховує 253 зображення для кожного з вихідних класів.

З метою спрощення процесу навчання і збільшення продуктивності мережі було прийнято рішення щодо використання технології transfer

learning шляхом побудови нашої мережі на основі попередньо навченої ШНМ, її добудови і донавчнання відповідно до нашої задачі і предметної області за допомогою зібраного раніше датасета. Був проведений порівняльний аналіз різних архітектури попередньо навчених мереж, здійснений вибір найбільш придатної з них і проектування вже нашої ШНМ на її основі.

Після побудови загальної архітектури ШНМ було проведено її налаштування шляхом підбору параметрів навчання і оптимального оптимізатора, після чого було проведено безпосереднє навчання побудованої ШНМ і оцінка її результативності.

Головні показники продуктивності створеної моделі є задовільними, швидкість класифікації зображення становить значно менше ніж задане завданням обмеження в одну секунду і у той же час точність класифікації становить 96%, що також задовольняє обмеженню заданому в завдані у 90% точності і становить гідну конкуренцію найкращім із вже існуючих подібних систем, які хоча і мають схожу результативність, але, у той же час, мають більш ускладнену структуру відносно нашої і були навчанні на більш великих датасетах – і, як наслідок, потребують значно більше ресурсів на процес свого навчання.

З метою покращення результативності створеної моделі можна розширити навчальний датасет і зкомбінувати безпосередній аналіз рентгенівських знімків з додатковими відомостями про пацієнта (наприклад стать, вік і т.п).

Створена у ході даної роботи модель може бути застосована у якості ядра автоматизованих систем діагностики респіраторних захворювань за допомогою аналізу рентгенівських знімків, які можуть значно полегшити роботу лікарів рентгенологів, збільшити її продуктивність і точність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). URL: http://image-net.org/challenges/LSVRC/2012/results.html#t1 (дата звернення: 15.04.2021).

2. World Health Organization Health Emergency Dashboard. URL: https://covid19.who.int/region/euro/country/ua (дата звернення: 15.04.2021).

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

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. *The Lancet*. 2020. P. 2– 5.

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. 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. London,

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. Анил К. Джейн, Жианчанг Мао, К М. Моиуддин. Введение в искусственные нейронные сети. *Открытые системы* – 1997. № – 4. С. 3.

15. W.S. McCulloch and W. Pitts, A logical Calculus of Ideas Immanent in Nervous Activity, Bull. Mathematical Biophysics, Vol. 5, 1943, p. 115-133.

16. R.Rosenblatt. Principles of Neurodynamics, Spartan Books, New York, 1962.

17. M. Mitnsky and S. Papert, Perceptrons: An Introduction to Computational Geometry, MIT Press, Cambridge, Mass., 1969.

18. J.J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilitie. in Proc. National Academy of Sciencies, USA 79, 1982, p. 2554-2558.

19. P. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, Phd Thesis, Dept. of Applied Mathematics, Harvard University, Cambridge, Mass., 1974.

20. D.E. Rumelhart and J.L. McClelland, Parallel Distributed Processing: Exploration in the Microstructure of Cognition, MIT Press, Cambridge, Mass., 1986.

21. J.A. Anderson and E. Rosenfeld. Neurocomputing: Foundation of Research, MIT Press, Cambridge, Mass, 1988.

22. Горбань А.Н. Обобщенная аппроксимационная теорема и вычислительные возможности нейронных сетей. *Сибирский журнал вычислительной математики*, 1998, т. 1, № 1. С. 12–24.

23. Joshua Bengio Yann LeCun, Leon Bottou, Patrick Haffner. Gradient-based learning applied to document recognition. YIEEE. — 1998.

24. A. Krizhevsky, I. Sutskever, G. Hinton. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 25. Curran Associates, Inc. 2012.

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

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

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

28. 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.

29. Terziyan, V. Semantic and Agent Technologies for Distributed e-Health. *Social and Health Care Data-Processing Day*. University of Jyvaskyla. Keynote Speech. 28.05.2009.