

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА
АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

**АВТОМАТИЧНЕ РОЗПІЗНАВАННЯ ХВОРОБ
СІЛЬСЬКОГОСПОДАРСЬКИХ РОСЛИН**

Текстова частина до курсової роботи
за спеціальністю „Комп’ютерні науки” 122

Керівник курсової роботи
Кандидат фіз.-мат. наук, доцент

Афонін А.О.

(прізвище та ініціали)

(підпис)

“ ____ ” _____ 2021 р.

Виконав студент 1 курсу

Кундік К.В.

(прізвище та ініціали)

“ ____ ” _____ 2020 р.

Київ 2021

ЗМІСТ

ВСТУП.....	5
РОЗДІЛ 1.....	9
1.1 Аналіз сучасного стану питання та обґрунтування теми.....	9
1.2 Огляд існуючих аналогів розробки	9
1.3 Постановка задачі.....	11
РОЗДІЛ 2.....	12
2.1 Основні концепції машинного навчання	12
2.2 Проблеми та завдання, що вирішує машинне навчання	16
2.3 Основні концепції глибинного навчання.....	18
2.4 Проблема класифікації зображень.....	22
РОЗДІЛ 3.....	25
3.1 Опис датасету для навчання.....	25
3.2 Проектування моделі нейронної мережі та створення коду для навчання	27
3.3 Навчання вже існуючих архітектур нейронних мереж	31
3.4 Використання сервісу Google AutoML Vision.....	34
3.5 Написання коду для телеграм боту з використанням навчених моделей.....	38
3.6 Тестування програми та результати її виконання.....	39
ВИСНОВКИ	43
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	44
ДОДАТКИ.....	46
Додаток №1	46
Додаток №2	47

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультимедійних систем,
канд. фіз.-мат. наук, доцент

_____ О. П. Жежерун

(підпис)

“ ____ ” _____ 202_ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту _____ Кундіку Кирилу

_____ 1 _____ курсу факультету інформатики

ТЕМА: Автоматичне розпізнавання хвороб сільськогосподарських рослин

Вихідні дані:

Зміст ТЧ до курсової роботи:

Вступ

Розділ 1. Аналіз предметної області. Постановка завдання курсової роботи

Розділ 2. Дослідження сучасних технологій машинного навчання та глибинного
навчання

Розділ 3. Опис практичної частини роботи

Висновки

Список джерел

Додатки

Дата видачі “ ____ ” _____ 202_ р.

Керівник _____ Завдання отримано _____

Календарний план виконання курсової роботи

Тема: Автоматичне розпізнавання хвороб сільськогосподарських рослин

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	листопад 2020 р.	
2.	Огляд літератури за темою роботи	листопад-грудень 2020 р.	
3.	Оволодіння теоретичними та практичними навичками роботи глибинним навчанням	грудень 2020 р.	
4.	Розробка архітектури моделі нейронної мережі та її тренування	січень-березень 2021 р.	
5.	Написання пояснювальної роботи	березень 2021 р.	
6.	Створення слайдів для доповіді та написання доповіді	квітень 2021 р.	
7.	Надання роботи керівнику для перевірки	квітень 2021 р.	
8.	Корегування роботи за результатами перевірки керівником	квітень 2021 р.	
9.	Остаточне оформлення пояснювальної роботи та слайдів	травень 2021 р.	
10.	Подання роботи на кафедру для перевірки на плагіат	травень 2021 р.	
11.	Захист курсової роботи	травень 2021 р.	

Студент Кундік К.В.

Керівник Афонін А.О.

“ ” _____ р.

ВСТУП

Відповідно до дослідження ООН лише в Індії, близько 70% врожаю хворіє кожного року з різними ступенями важкості. Хвороби рослин різко знижують якість та кількість врожаю, що знищує потенційні переваги сільського господарства. Зменшення втрат врожаю через хвороби дозволить фермерам виробляти продукти харчування ефективніше, приносячи користь сільськогосподарській галузі та навколишньому середовищу. Хвороби рослин надзвичайно важливі для виявлення та належного лікування на початковому етапі, оскільки невилікована хвороба може вразити все господарство та зруйнувати його.

На даний момент фермерам доводиться періодично ретельно аналізувати кожен сільськогосподарську культуру, щоб виявляти хвороби, що є надзвичайно складним і трудомістким завданням (особливо, коли кількість сільськогосподарських культур величезна).

Виявлення хвороби за допомогою автоматичної техніки є корисним, оскільки це зменшує об'ємну роботу спостереження за фермами сільськогосподарських культур. Варіації симптомів, які вказують на хвороби рослини, можуть бути неправильно діагностовані через труднощі їх визначення. Автоматизована система, розроблена для виявлення хвороб рослин за зовнішнім виглядом та візуальними симптомами, може бути корисною для любителів садівничого процесу, а також для навчених фахівців як система перевірки в діагностиці хвороб. Досягнення в галузі комп'ютерного зору дають можливість розширити та вдосконалити практику захисту рослин та розширити ринок програм комп'ютерного зору в галузі землеробства.

Мета та завдання курсової роботи

Мета: Створити модель нейронної мережі для визначення хвороб рослин за зображеннями їхніх листків, що спростить пошук та визначення лікування, профілактики хвороб таких рослин.

Завдання: Розробити сервіс, який стане швидким інтерактивним довідником для сільськогосподарських малих підприємств та фермерів. Реалізувати визначення хвороб рослин за зображенням їхніх уражених листків та видачу інформації щодо лікування та профілактики.

Об'єкт дослідження

Розробка моделі нейронної мережі з використанням фреймворку tensorflow на Python.

Предмет дослідження

Сучасні архітектури глибоких нейронних мереж, що можуть бути використані для вирішення даної задачі.

Практичне значення одержаних результатів

Розроблений сервіс буде використовуватись сільськогосподарськими малими підприємствами та фермерами для визначення хвороб рослин за їхніми ураженими листками, що значно полегшить їхню роботу та процеси, а також підвищить рівень врожайності.

Джерела дослідження

Під час написання курсової роботи було досліджено наукові статті про машинне навчання та архітектури нейронних мереж, а також способи їхнього застосування для вирішення різних завдань. Проаналізовано

інтернет-ресурси, які містять інформації про рейтинги найкращих глибинних нейронних мереж та сфери, на яких вони продемонстрували найкращі результати. Під час роботи було вивчено аналоги для визначення можливих проблем та аналізу шляхів їх уникнення.

Структура роботи

Курсова робота складається з анотації, вступу, трьох основних розділів та підрозділів, висновків, списку використаних джерел, додатків.

Перший розділ курсової роботи доповнює думки, що були попередньо висвітлені у вступі до роботи. Розглянуто існуючі аналоги та проведено детальний аналіз щодо незначного поширення додатків, які вирішують поставлену проблему. Визначено задачу та описано кроки для подальшої розробки.

У другому розділі описані основні концепти машинного навчання та глибинного навчання. Описано теорію розробки нових архітектур глибинних нейронних мереж та їх можливостей, сучасні способи їх застосування на практиці та кращі практики їх використання для передбачення на реальних прикладах. Детально описані проблеми, що вивчають ці сфери комп'ютерних наук, а також виокремлено проблему класифікації зображень як цільове завдання, що вирішується моделями для практичної частини.

Третій розділ є детальним описом практичної частини роботи. У розділі детально описано проектування нової архітектури глибинної нейронної мережі. Описано реалізацію програмного інтерфейсу для тренування моделей машинного навчання, створення звітів їх навчання, роботи з передбаченнями. Також описано використання існуючих архітектур нейронних мереж для вирішення поставленого завдання.

Наведено опис розробки телеграм боту як зразок використання розроблених моделей на реальних прикладах.

РОЗДІЛ 1

1.1 Аналіз сучасного стану питання та обґрунтування теми

У країнах, що розвиваються економіка головним чином залежить від сільського господарства. В більшості таких країнах сільське господарство є основним джерелом їжі, а також основним джерелом доходу для фермерів. Отже, втрата культивованих культур через хворобу дійсно означатиме значну втрату доходу для фермерів і значного сектору економіки всієї держави, і якщо це станеться у масових масштабах, це призведе до дефіциту продовольства. Сільське господарство також служить вирішенням багатьох екологічних проблем, забезпечує середовища існування, зберігає екосистему, а також грає роль у кругообігу води.

Проблема ефективного захисту рослин від хвороб тісно пов'язана з проблемами сталого розвитку сільського господарства та кліматичних змін. У навколишньому середовищі присутні різні патогени, які серйозно впливають на посіви та ґрунт, в якому висаджена рослина, впливаючи тим самим на виробництво врожаю. На рослинах і посівах спостерігаються різні хвороби. Основною ідентифікацією ураженої рослини або культури є її листя. Різноманітні кольорові плями та малюнки на листі дуже корисні для виявлення хвороби.

Та поки не було створено єдиного засобу, який би задовольняв потреби більшості фермерів, для вирішення проблеми виявлення хвороб рослин на ранніх та проміжних стадіях активної фази хвороб для ефективного знешкодження, лікування та подальшої профілактики сільськогосподарських культур.

1.2 Огляд існуючих аналогів розробки

В ході аналізу було виявлено, що на даний момент проблема далеко від вирішення і знайти додаток, який визначав хвороби

сільськогосподарських рослин дуже складно. Більшість із знайдених матеріалів були у вигляді вихідного коду програм, для запуску якого необхідно мати достатню кількість технічних навичок, або наукові статті, що описують архітектуру моделей нейронних мереж та ін. Останнє вимагає не лише технічних навичок для запуску коду, а й навичок для написання коду на фреймворках для машинного навчання, використання відповідного датасету для навчання, та безпосередньо необхідність тренування моделей, підбирання параметрів для збільшення ефективності і точності.

Єдиним додатком, що має схожий функціонал для вирішення проблеми ідентифікації хвороб рослин, є PictureThis. [1] Розробники заявляють про більш ніж 10 тис. видів рослин для розпізнавання та надання вичерпної інформації щодо наявних хвороб. Для використання достатньо або завантажити фотографію з альбому телефону, або сфотографувати безпосередньо з додатку. З основного сайту програмного забезпечення вже стає зрозумілим, що основною ціллю розробки було задоволення потреб садівників кімнатних рослин, а інформація щодо сільськогосподарських культур та визначення їх хвороб на сайті відсутня. Під час використання дійсно було підтверджено попередню думку щодо кімнатних рослин: точність їх розпізнавання достатньо велика (близько 85% на зображеннях кімнатних рослин з інтернету), а інформація, що надається щодо їх догляду покриває всі необхідні процеси. Щодо хворих кімнатних рослин тестування додатку показало значне зменшення точності до майже 60% на зображеннях з інтернету хворих кімнатних рослин. Остаточним доказом неефективності додатку стала перевірка на сільськогосподарських культурах: менше ніж 17.5% точності на рослинах із хворобами. Таким чином з'ясовано, що використання додатку сільськогосподарськими підприємствами і фермерами є неможливими через занадто низьку точність. Єдиним випадком, коли можливо його використовувати – це знаходження інформації щодо хвороб таких рослин, але таку інформацію легко знайти у

перевірених наукових джерелах разом з відповідним доглядом та лікуванням за ними. Також слід виділити те, що додаток працює лише на мобільних пристроях з операційними системами iOS або Android і тільки при наявному підключення до інтернету, що можливо створить певні незручності для сільськогосподарників в певних регіонах без зв'язку відповідної якості.

Підсумком аналізу стає те, що поки вирішення цієї проблеми в повному обсязі не існує.

1.3 Постановка задачі

Виходячи із детального дослідження ресурсів та аналізу предметної області було деталізовано задачу, на якій базується реалізація практичної частини роботи.

Постановка задачі:

1. Дослідити сучасні концепти застосування машинного навчання та глибинного навчання, проаналізувати архітектури моделей нейронних мереж.
2. Підготувати зображення сільськогосподарських культур з поширеними хворобами для створення датасету для тренувань.
3. Провести аналіз датасету, попередню обробку та нормалізацію.
4. Розробити архітектуру глибинної нейронної мережі
5. Використати для навчання популярні архітектури моделей глибинних мереж, що мають високу точність у вирішеннях схожих задач.
6. Розробити телеграм бот, що буде використовувати натреновану модель для передбачення хвороб сільськогосподарських рослин на основі зображень користувачів та надавати вичерпну інформацію щодо лікування та профілактики виявлених хвороб.

РОЗДІЛ 2

2.1 Основні концепції машинного навчання

Коли програмні комп'ютери були вперше винайдені, люди задавались питанням, чи можуть такі машини стати розумними за сто років до того, як їх було побудовано [2]. Сьогодні штучний інтелект (ШІ) - це процвітаюче поле, що має безліч практичних додатків та активних дослідницьких тем. Люди прагнуть до інтелектуального програмного забезпечення для автоматизації повсякденної праці, розуміння мови чи зображень, постановки діагнозів у медицині та підтримки основних наукових досліджень. У перші дні штучного інтелекту сфера швидко вирішувала проблеми, які є інтелектуально важкими для людей, але відносно простими для комп'ютерів - проблеми, які можна описати за допомогою списку формальних математичних правил. Справжній виклик штучному інтелекту виявився вирішенням завдань, які є досить простими для виконання людьми, але людям важко їх формально описати - проблеми, які люди вирішують інтуїтивно, і які відчуються автоматично, як розпізнавання вимовлених слів чи облич на зображеннях. Вирішенням саме таких складних завдань для програмування і займається розділ штучного інтелекту в більшій мірі.

Багато ранніх успіхів ШІ відбувались у відносно офіційних середовищах і не вимагали від комп'ютерів знань про світ. Наприклад, система гри в шахи Deep Blue від IBM перемогла чемпіона світу Гаррі Каспарова в 1997 році. Звичайно, шахи - це дуже простий світ, що містить лише шістдесят чотири місця та тридцять дві фігури, які можуть рухатися лише жорстко обмеженими способами. Розробка успішної шахової стратегії є приголомшливим досягненням, але проблема полягає не в складності опису набору шахових фігур та допустимих переміщень до комп'ютера.

Шахи можна повністю описати дуже коротким переліком цілком формальних правил, які програміст легко надати заздалегідь. [4]

Як не дивно, абстрактні та формальні завдання, які є одними з найскладніших розумових починань для людини, є одними з найпростіших для комп'ютера. Комп'ютери давно змогли перемогти навіть найкращого шахіста, але лише нещодавно почали відповідати деяким здібностям пересічних людей розпізнавати предмети або мову. Повсякденне життя людини вимагає величезних знань про світ. Значна частина цих знань є суб'єктивною та інтуїтивно зрозумілою, а тому її важко сформулювати формально. Комп'ютери повинні досягти цих самих знань в повному обсязі, щоб поводитись розумно. Однією з ключових проблем штучного інтелекту є те, як перенести ці неформальні знання в комп'ютер.

Труднощі, з якими стикаються системи, що покладаються на жорстко закодовані знання, свідчать про те, що системи ШІ потребують здатності здобувати власні знання, витягуючи зразки з необроблених даних. Ця можливість відома як машинне навчання. Впровадження комп'ютерів, що підтримують машинне навчання, для вирішення проблем, пов'язаних із пізнанням реального світу, та прийняття суб'єктивних рішень. Простий алгоритм машинного навчання, який називається логістичною регресією, може визначити чи рекомендувати кесарів розтин [3]. Простий алгоритм машинного навчання, який називається наївним Байєсом, може відокремити електронну пошту зі спамом.

Ефективність цих простих алгоритмів машинного навчання в значній мірі залежить від подання даних, які вони отримують. Наприклад, коли використовується логістична регресія, щоб рекомендувати кесарів розтин, система ШІ не обстежує пацієнта безпосередньо. Натомість лікар повідомляє системі кілька важливих відомостей, таких як наявність або відсутність рубця на матці. Кожна інформація, включена до подання пацієнта, відома як особливість. Логістична регресія дізнається, як кожна з

цих особливостей пацієнта співвідноситься з різними результатами. Однак це не може впливати на те, як якісь функції визначаються. Якби логістична регресія отримала МРТ-сканування пацієнта, а не формалізований звіт лікаря, він не міг би робити корисні прогнози. Окремі пікселі при МРТ мають незначну кореляцію з будь-якими ускладненнями, які можуть виникнути під час пологів. [3]

Ця залежність від уявлень є загальним явищем, яке проявляється у всій інформатиці та навіть у повсякденному житті. В інформатиці такі операції, як пошук колекції даних, можуть тривати експоненційно швидше, якщо колекція структурована та індексована розумно.

Багато завдань штучного інтелекту можна вирішити, розробивши правильний набір функцій, які потрібно витягнути для цього завдання, а потім надавши ці функції простому алгоритму машинного навчання. Наприклад, корисною характеристикою ідентифікації динаміка за звуком є оцінка розміру голосового тракту спікера. Ця функція дає чітку підказку щодо того, чи є доповідачем чоловік, жінка чи дитина.

Однак для багатьох завдань важко знати, які особливості слід виділити. Наприклад, припустимо, що ми хотіли б написати програму для виявлення автомобілів на фотографіях. Ми знаємо, що автомобілі мають колеса, тому, можливо, ми хотіли б використовувати наявність колеса як особливість. На жаль, складно описати, як саме виглядає колесо з точки зору значень пікселів. Колесо має просту геометричну форму, але його зображення може ускладнюватися тінями, що падають на колесо, сонцем, що блищить металевими частинами колеса, крилом автомобіля або предметом на передньому плані, що закриває частину колеса, і так далі.

Одним із вирішенням цієї проблеми є використання машинного навчання для виявлення не тільки відображення від подання до виводу, але й самого подання. Цей підхід відомий як репрезентативне навчання. Вивчені подання часто дають набагато кращі показники, ніж це можна

отримати за допомогою ручних розроблених подань. Вони також дозволяють системам ШІ швидко адаптуватися до нових завдань, з мінімальним втручанням людини. Алгоритм репрезентативного навчання може виявити хороший набір функцій для простого завдання за лічені хвилини або для складного завдання за години чи місяці. Ручне проектування функцій для складного завдання вимагає великої кількості людського часу та сил; це може зайняти десятиліття для цілого співтовариства дослідників. [4]

Основним прикладом алгоритму репрезентативного навчання є автокодер. Автокодер - це комбінація функції кодера, яка перетворює вхідні дані у різне представлення, та функції декодера, яка перетворює нове подання назад у вихідний формат. Автокодери навчені зберігати якомога більше інформації, коли введення проходить через кодер, а потім декодер, але вони також навчені робити нове подання з різними додатковими властивостями. Різні типи автокодерів спрямовані на досягнення різних видів властивостей.

При розробці функцій або алгоритмів для вивчення особливостей, мета, як правило, полягає у виділенні факторів варіації, що пояснюють спостережувані дані. У цьому контексті використовують слово «фактори» для позначення окремих джерел впливу; фактори, як правило, не поєднуються множенням. Такі фактори часто не є величинами, які безпосередньо спостерігаються. Натомість вони можуть існувати або як неспостережувані об'єкти, або як неспостережувані сили у фізичному світі, які впливають на спостережувані величини. Вони також можуть існувати як конструкції в людській свідомості, що надають корисні спрощувальні пояснення або виведені причини спостережуваних даних. Їх можна сприймати як концепції або абстракції, які допомагають зрозуміти багату мінливість даних. При аналізі запису мовлення фактори варіації включають вік мовця, його стать, його акцент та слова, які вони говорять. При аналізі

зображення автомобіля фактори варіації включають положення автомобіля, його колір, а також кут і яскравість сонця. [4]

Основним джерелом труднощів у багатьох реальних додатках штучного інтелекту є те, що багато факторів зміни впливу впливають на кожен окремий фрагмент даних, який ми можемо спостерігати. Окремі пікселі на зображенні червоного автомобіля можуть бути дуже близькими до чорного вночі. Форма силуету автомобіля залежить від кута огляду. Більшість програм вимагає від нас розмежувати фактори варіації та відкинути ті, про які ми не дбаємо.

Звичайно, може бути дуже важко витягти такі абстрактні риси високого рівня з необроблених даних. Багато з цих факторів варіації, такі як акцент мовця, можна визначити лише за допомогою складного, майже на людському рівні розуміння даних. Коли отримати уявлення майже так само важко, як вирішити вихідну проблему, репрезентативне навчання, на перший погляд, не допоможе у вирішенні цієї проблеми.

2.2 Проблеми та завдання, що вирішує машинне навчання

Завдання машинного навчання, як правило, класифікуються на три широкі категорії, залежно від характеру навчального "сигналу" або "зворотного зв'язку", доступних для системи навчання.

Це:

- Навчання з наглядом. Комп'ютер представлений прикладами входів та їх бажаними результатами, наданими "вчителем", і мета полягає у вивченні загального правила, яке відображає входи на виходи. Простими словами це можна описати як навчання на даних, що були заздалегідь розмічені і комп'ютерний алгоритм має співставити вхідні дані з розміченими правильними/бажаними результатами.

- Навчання без нагляду. Алгоритму навчання не присвоюються мітки, що залишає його самостійно знаходити структуру у своєму введенні.

Навчання без нагляду може бути самоціллю (виявлення прихованих закономірностей у даних) або засобом досягнення мети.

- Навчання з підкріпленням. Комп'ютерна програма взаємодіє з динамічним середовищем, в якому вона повинна виконати певну мету (наприклад, керувати транспортним засобом), без того, щоб вчитель явно сказав їй, чи наближається алгоритм до успіху чи ні. Інший приклад - навчитися грати в гру, граючи проти суперника. [5]

Між контрольованим та неконтрольованим навчанням є напівконтрольоване навчання, де вчитель подає неповний навчальний сигнал: навчальний набір з відсутністю деяких (часто багатьох) цільових результатів. Трансдукція - це особливий випадок цього принципу, коли на час вивчення відома вся сукупність проблемних випадків, за винятком того, що частина цілей відсутня.

Серед інших категорій проблем машинного навчання, навчання вчитися засвоює власне індуктивне упередження на основі попереднього досвіду. Розвиваюче навчання, розроблене для навчання роботів, генерує власні послідовності (також звані як навчальні програми) навчальних ситуацій для кумулятивного набуття репертуару нових навичок шляхом автономного самодослідження та соціальної взаємодії з вчителями-людьми та використання таких механізмів керівництва, як активне навчання, дозрівання, рухова синергія та імітація.

Інша категоризація завдань машинного навчання виникає, коли враховується бажаний результат машинно навченої системи:

- У класифікації вхідні дані поділяються на два або більше класів, і той, хто навчається, повинен створити модель, яка призначає невидимі вхідні дані одному (або багатозначній класифікації) або більше з цих класів. Зазвичай з цим вирішують під наглядом. Фільтрація спаму є прикладом класифікації, де вхідними даними є повідомлення електронної пошти (або інші), а класи - "спам" та "не спам".

- В регресії, яка також є контрольованою проблемою, результати є неперервними, а не дискретними.
- При кластеризації набір вхідних даних слід розділити на групи. На відміну від класифікації, групи заздалегідь невідомі, що робить це, як правило, завданням без нагляду.
- Оцінка щільності визначає розподіл вхідних даних у певному просторі.
- Зменшення розмірності спрощує вхідні дані, відображаючи їх у простір нижчих розмірів. Моделювання тем - це пов'язана проблема, коли програмі надається перелік документів, що стосуються людської мови, і доручається з'ясувати, які документи охоплюють подібні теми. [5]

2.3 Основні концепції глибокого навчання

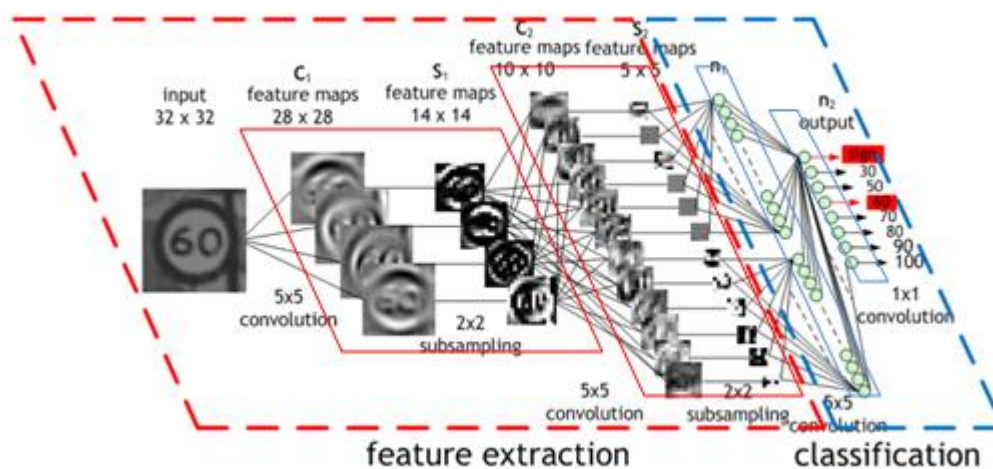


Рис 2.1. Зображення послідовної обробки зображення нейронною мережею [6]

Глибоке навчання вирішує центральну проблему в репрезентативному навчанні шляхом введення уявлень, які виражаються в термінах інших, більш простих уявлень. Глибоке навчання дозволяє комп'ютеру будувати складні концепції з більш простих концепцій.

Рисунок 2.1 показує, як система глибокого навчання може представляти концепцію образу дорожнього знаку, поєднуючи більш прості поняття, такі як кути, тіні та контури, які, у свою чергу, визначаються з точки зору різних країв.

Основним прикладом моделі глибокого навчання є глибока мережа прямого пересилання або багат шаровий персептрон (MLP). Багат шаровий персептрон - це просто математична функція, що відображає деякий набір вхідних значень у вихідні. Функція формується шляхом складання багатьох простих функцій. Можна думати про кожне застосування різних математичних функцій як про нове представлення вхідних даних. [4]

Ідея вивчення правильного подання даних дає єдиний погляд на глибоке навчання. Інший погляд на глибоке навчання полягає в тому, що глибина дозволяє комп'ютеру вивчати багатоступеневу комп'ютерну програму. Кожен шар представлення можна сприймати як стан пам'яті комп'ютера після паралельного виконання іншого набору інструкцій. Мережі з більшою глибиною можуть виконувати більше інструкцій послідовно. Послідовні інструкції надають велику потужність, оскільки пізніші інструкції можуть посилатися на результати попередніх. Згідно з цим поглядом на глибоке навчання, не вся інформація в активаціях шару обов'язково кодує фактори варіації, що пояснюють вхідні дані. Представлення також зберігає інформацію про стан, яка допомагає виконати програму, яка може зрозуміти введені дані. Ця інформація про стан може бути аналогом лічильника або показчика в традиційній комп'ютерній програмі. Це не має нічого спільного із вмістом введення, зокрема, але допомагає моделі організувати його обробку.

Глибокі мережі прямого пересилання, які також називаються нейронними мережами прямого пересилання, або багат шарові персептрони (MLP), є типовими моделями глибокого навчання. Метою

мережі прямого пересилання є наближення деякої функції f^* . Наприклад, для класифікатора $y = f^*(x)$ відображає вхідні дані x до категорії y . Мережа переадресації визначає відображення $y = f(x; \theta)$ та вивчає значення параметрів θ , що призводять до найкращого наближення функції. [4]

Ці моделі називаються моделями прямого пересилання, оскільки інформація протікає через функцію, яка обчислюється від x , через проміжні обчислення, що використовуються для визначення, і, нарешті, до вихідного y . Немає з'єднань зворотного зв'язку, в яких виходи моделі подаються назад в себе. Коли нейронні мережі прямої передачі розширюються, включаючи з'єднання зворотного зв'язку, вони називаються рекурентними нейронними мережами.

Мережі прямого зв'язку мають надзвичайне значення для практиків машинного навчання. Вони складають основу багатьох важливих комерційних додатків. Наприклад, згорткові мережі, що використовуються для розпізнавання об'єктів за фотографіями, є спеціалізованими видами мереж прямого зв'язку. Мережі прямого зв'язку - це концептуальна сходинка на шляху до періодичних мереж, які забезпечують багато програм природної мови.

Нейронні мережі прямого зв'язку називаються мережами, оскільки вони, як правило, представлені складанням безлічі різних функцій. Модель пов'язана з спрямованим ациклічним графіком, що описує, як функції складаються разом. Наприклад, маємо три функції $f(1)$, $f(2)$ та $f(3)$, з'єднані ланцюжком, щоб утворити $f(x) = f(3)(f(2)(f(1)(x)))$. Ці ланцюгові структури є найбільш часто використовуваними структурами нейронних мереж. У цьому випадку $f(1)$ називається першим рівнем мережі, $f(2)$ - другим рівнем тощо. Загальна довжина ланцюга визначає глибину моделі. З цієї термінології виникла назва “глибоке навчання”. Кінцевий рівень мережі прямої передачі називається вихідним рівнем. Під час навчання нейронних мереж алгоритм рухається від $f(x)$ до $f^*(x)$. Дані навчання дають

шумні, приблизні приклади $f^*(x)$, які оцінюються в різних навчальних пунктах. Кожен приклад x супроводжується міткою $y \approx f^*(x)$. На навчальних прикладах безпосередньо вказується, що повинен робити вихідний рівень у кожній точці x ; він повинен давати значення, близьке до y . Поведінка інших рівнів не визначається безпосередньо навчальними даними. Алгоритм навчання повинен вирішити, як використовувати ці рівні для отримання бажаного результату, але дані навчання не говорять про те, що повинен робити кожен окремий рівень. Натомість алгоритм навчання повинен вирішити, як використовувати ці рівні, щоб найкраще реалізувати наближення f^* . Оскільки навчальні дані не показують бажаного результату для кожного з цих шарів, вони називаються прихованими шарами. [4]

Нарешті, ці мережі називають нейронними, оскільки вони були натхнені неврологією. Кожен прихований рівень мережі, як правило, має векторну оцінку. Розмірність цих прихованих шарів визначає ширину моделі. Кожен елемент вектора можна інтерпретувати як такий, що відіграє роль, аналогічну нейрону. Замість того, щоб думати про шар як про те, що він представляє єдину функцію вектор-вектор, ми можемо також думати про шар, що складається з багатьох одиниць, які діють паралельно, кожна представляє функцію вектор-значення. Кожна одиниця нагадує нейрон в тому сенсі, що він отримує вхідні дані від багатьох інших одиниць і обчислює своє власне значення активації. Ідея використання багатьох шарів векторно-значущих уявлень береться з неврології. Вибір функцій $f^{(i)}(x)$, що використовуються для обчислення цих уявлень, також вільно керується нейронауковими спостереженнями про функції, які обчислюють біологічні нейрони. Однак сучасні дослідження нейронних мереж керуються багатьма математичними та інженерними дисциплінами, і мета нейронних мереж полягає не в ідеальному моделюванні нейронної біологічної мережі. Найкраще думати про мережі прямого пересилання як про машини

наближення функцій, які призначені для досягнення статистичного узагальнення, час від часу отримуючи деякі уявлення з того, що ми знаємо з неврології, а не як про моделі функцій біологічних мереж.

Одним із способів зрозуміти мережі зворотного зв'язку є, для початку, лінійні моделі та розгляд способу подолання їх обмежень. Лінійні моделі, такі як логістична регресія та лінійна регресія, можуть бути ефективними і надійними, або в закритій формі, або з опуклою оптимізацією. Лінійні моделі також мають очевидний недолік, що ємність моделі обмежена лінійними функціями, тому модель не може зрозуміти взаємодії між будь-якими двома вхідними змінними. Щоб розширити лінійні моделі для представлення нелінійних функцій x , ми можемо застосувати лінійну модель не до самого x , а до трансформованого входу $\varphi(x)$, де φ - нелінійне перетворення. Крім того, ми можемо застосувати зміну ядра, щоб отримати нелінійний алгоритм навчання, заснований на неявному застосуванні φ -відображенні. Тоді можна думати про φ як про надання набору ознак, що описують x , або як про надання нового представлення для x . [4]

2.4 Проблема класифікації зображень

З бурхливим розвитком технологій мобільного Інтернету все більше інформації про зображення зберігається в Інтернеті. Зображення стало ще одним важливим носієм мережевої інформації після тексту. У цьому контексті дуже важливо використовувати комп'ютер, щоб розумно класифікувати та розпізнати ці зображення та зробити їх кращими для використання. На початковому етапі класифікації та розпізнавання зображень люди в основному використовують цю технологію для задоволення деяких допоміжних потреб, наприклад, функція обличчя Baidu може допомогти користувачам знайти найбільш подібну знаменитість.

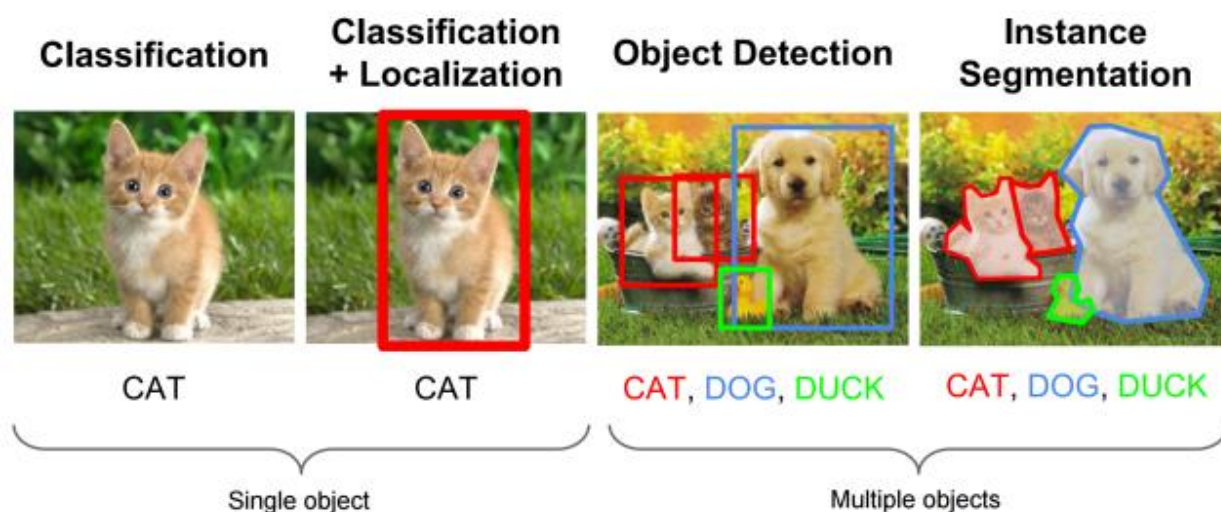


Рис. 2.2. Зображення основних завдань глибокого навчання в сфері комп'ютерного зору [7]

У галузі класифікації зображень традиційні алгоритми машинного навчання, такі як K-Nearest Neighbor (KNN) та Support Vector Machine (SVM), широко застосовуються для вирішення проблем класифікації і особливо добре працюють на невеликих наборах даних. Однак, звертаючись до великого набору даних із більш складними функціями та класами, глибоке навчання переважає традиційне машинне навчання. [8]

Останнім часом трансферне навчання стає більш поширеним, ніж будь-коли раніше, оскільки воно може заощадити велику кількість обчислювальних та часових ресурсів для розвитку глибоких мереж з самого початку. Попередньо навчені моделі глибокого навчання можуть бути відправною точкою для вирішення суміжних проблем. Оскільки цілі класифікації, такі як рослини чи об'єкти побутового світу, мають занадто багато категорій, і деякі їх особливості подібні, так що машинне навчання та глибоке навчання будуть корисними, щоб допомогти дослідникам класифікувати заплутані класи.

Через переважання смартфонів сьогодні як користувачі мобільних телефонів, так і індустрія штучного інтелекту розраховують на

впровадження в мобільних додатках систем для досягнення зручності. Попередній широко використовуваний метод класифікації зображень за допомогою мобільних пристроїв (який все ще використовується за певних обставин) полягав у збереженні моделі на сервері. Класи та результати передаються між сервером та мобільними пристроями і це спричиняє затримку через двостороннє спілкування, також виникає проблеми конфіденційності даних.

РОЗДІЛ 3

3.1 Опис датасету для навчання

Для навчання було використано датасет компанії PlantVillage, яка займається розробками в сферах ШІ. [9] Датасет першочергово було створено компанією для проведення хакатону щодо розробки моделі для визначення хвороб сільськогосподарських рослин. Він містить 38 класів для більш ніж 15 видів рослин, що є найбільш популярними агрокультурами серед фермерів Південної Африки та Північної Америки такі як кукурудза, томати, картопля та ін. в т.ч. фруктові культури та ягоди. Датасет містить майже 65 тисяч вже розмічених зображень, що дає змогу відразу переходити до його обробки і безпосередньої роботи з ним.

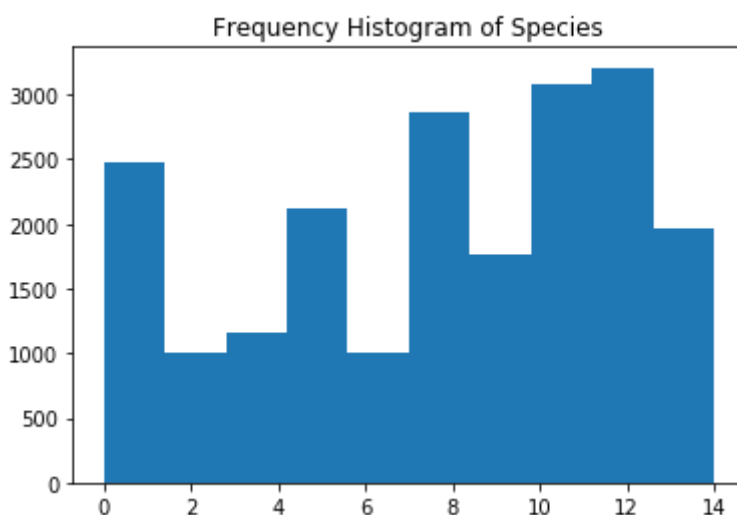


Рис. 3.1. Розподіл класів використаного датасету

З Рисунок 3.1. можна зрозуміти, що розподіл класів достатньо збалансований проте існують деякі класи де значно не вистачає представників. Таким чином, було вирішено застосувати компонент бібліотеки keras ImageDataGenerator, що дозволяє доповнити існуючий датасет новими зображеннями. Ці нові зображення генеруються штучно завдяки різним афінним перетворенням, зміною контрастності, яскравості і т.д.

Застосування та конфігурацію цього компонента можна побачити в наступному фрагменті коду:

```
self._aug = ImageDataGenerator(
    rotation_range=25,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest"
)
```



Рис. 3.2. Приклад зображень класу ‘Tomato Bacterial Sport’ (томата, що має на листях ознаки бактеріальної хвороби)

Після розширення і класового балансування датасету необхідно перейти до етапу нормалізації даних. Нормалізація відбувається за допомогою Python бібліотек `opencv-python` та `numpy`. [10][11] Ці потужні бібліотеки дають змогу оперувати і змінювати зображення надаючи високорівневий програмний інтерфейс та методи лінійної алгебри.

Сам процес нормалізації відбувається в два етапи:

1. Зчитування зображення за допомогою бібліотеки `opencv-python`. Та змінення його розмірів відповідно до вхідного шару моделей. Під час навчання моделей було визначено розмір 220 на 220 пікселів, який дозволяв помістити одну партію зображень (`batch size`) в оперативну пам'ять для виконання одного кроку навчання моделі.
2. Приведення зображень в тип `numpy.array` для застосування функцій лінійної алгебри та роботи моделей з ними. Та безпосередньо нормалізація пікселів зображень до розподілу їх значень в межах від 0 до 1 за допомогою ділення цих значень на 255.



Рис. 3.3. Приклад зображень різних класів після застосування нормалізації.

Після нормалізації та розширення датасету можна переходити до процесу проектування моделей та їх навчання.

3.2 Проектування моделі нейронної мережі та створення коду для навчання

Для проектування та навчання моделей було обрано сучасний і популярний фреймворк для машинного навчання `tensorflow` та його розширення `keras`. [12]

За допомогою фреймворку було створено власну модель та спеціальний програмний інтерфейс, що дозволяє легко інтегрувати нові

архітектури моделей для навчання, створення звітів після їх навчання, роботи з оцінками моделей та їх подальшого використання для передбачень.

Програмний інтерфейс був розроблений задля виділення основних функцій навчання та роботи з моделями. Його можна використовувати у двох режимах: навчання та передбачення зі вже навченою моделлю. Для навчання було виокремлено функції нормалізації даних, розширення датасету, ініціалізації моделей (ініціалізація шарів та інших допоміжних функцій), вибір оптимізатора (за замовчуванням використовується один з найпопулярніших оптимізаторів машинного навчання Adam), визначення функції помилки та метрики навчання (за замовчуванням для функції помилки використовується Binary cross entropy loss function, а для метрики навчання – точність), сам процес навчання з виводом інформації щодо поточного кроку та епохи навчання, створення графіків навчання і підсумкових результатів та збереження натренованих параметрів моделі. Для передбачення було створено функції завантаження параметрів моделей, їх перевірка та валідація і функції для роботи з передбаченнями.

Фрагмент коду розробленого програмного інтерфейсу для підготовки моделей до навчання:

```
self._aug = ImageDataGenerator(
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
    zoom_range=0.2, horizontal_flip=True,
    fill_mode="nearest"
)
self._label_binarizer = LabelBinarizer()
self._image_labels =
self._label_binarizer.fit_transform(self._label_list)
self._np_image_list = np.array(self._image_list,
dtype=np.float16) / 225.0
print("[INFO] Splitting data to train, test")
self._x_train, self._x_test, self._y_train, self._y_test =
train_test_split(
```

```

        self._np_image_list, self._image_labels, test_size=0.2,
random_state=42
    )
    self._model = self.implementations()[self._name](
        (Config.height, Config.width, Config.depth),
len(self._label_binarizer.classes_), 0.0001, self._weights
    ).init()

```

Фрагмент коду для навчання моделей:

```

    opt = Adam(learning_rate=Config.INIT_LR, decay=Config.INIT_LR /
Config.EPOCHS)
    self._model.compile(loss="binary_crossentropy", optimizer=opt,
metrics=["accuracy"])
    print("[INFO] training network...")
    self._history = self._model.fit_generator(
        self._aug.flow(self._x_train, self._y_train,
batch_size=Config.BS),
        validation_data=(self._x_test, self._y_test),
        steps_per_epoch=len(self._x_train) // Config.BS,
        epochs=Config.EPOCHS, verbose=1
    )

```

Таким чином створений програмний інтерфейс дозволяє швидко і легко інтегрувати нові архітектури нейронних мереж і значно спрощує роботу зі звітами їх навчанням, а також роботи з їхнім подальшим використанням для передбачень та роботи з реальними зображеннями.

Наступним етапом було створення власної архітектури нейронної мережі, що буде мати високі показники навчання та оцінки після навчання. Власна модель отримала назву EasyNet, оскільки має невелику кількість параметрів і шарів. Що дозволяє їй достатньо швидко навчити і одразу отримати перші результати. У додатку №1 наведено зображення архітектури нейронної мережі EasyNet за допомогою орієнтованого графа TensorBoard. У додатку №2 наведено зображення загальної схеми для навчання з використанням розробленого програмного інтерфейсу.

Використання архітектури займає приблизно 150 хвилин для навчання та видає максимально 83.3% точності на тренувальних даних та

максимально 81.5% на тестових даних для оцінювання моделі. Нижче буде описані інші моделі і ця буде їм програвати в точностях та інших метриках, але час навчання буде значно меншим. Наприклад, популярна складна архітектура AlexNet при таких самих конфігураціях машини, що використовувалася для навчання, має розрахунковий час навчання в більше ніж 250 годин. Саме через такі значення розрахункового часу навчання змусили перенести навчання з локального комп'ютера на спеціалізовані дроплет від DigitalOcean для нейронних мереж. Про це буде описано нижче.

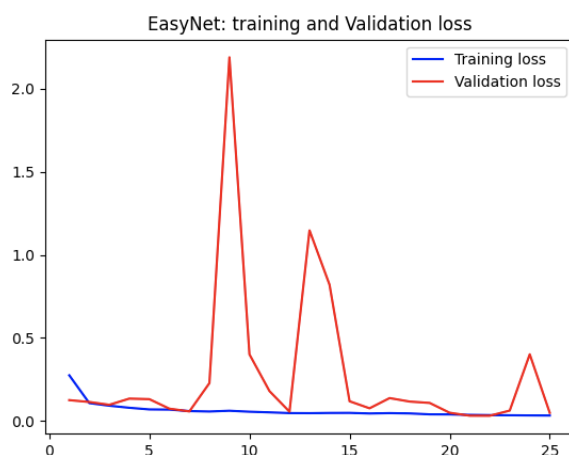


Рис. 3.4. Зображення графіків функції помилки на тренувальних та тестових даних під час навчання. Ось абсцис містить значення кількості епох тренуваної моделі, а ординат – значення функції помилки.

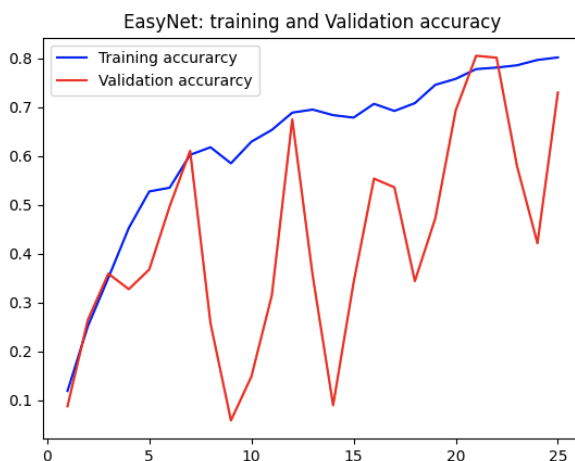


Рис. 3.5. Зображення графіків функції метрики (точності) на тренувальних та тестових даних під час навчання. Ось абсцис містить значення кількості епох тренуваної моделі, а ординат – значення функції метрики (точності).

3.3 Навчання вже існуючих архітектур нейронних мереж

Для навчання вже існуючих архітектур нейронних мереж було використано розроблений і описаний вище програмний інтерфейс, в який легко було інтегрувати моделі. Таким чином було обрано наступні архітектури: AlexNet, DenseNet121, LeNet, VggNet16, XceptionNet. Оскільки архітектури цих нейронних мереж значно складніші, вони мають набагато більшу кількість параметрів. Тому для їх навчання було прийнято рішення використати сервіс DigitalOcean.

Сервіс надає багато можливостей для створення серверної інфраструктури в хмарі, а також окремий функціонал для навчання моделей. Такий функціонал включає в себе спеціальні одиниці для виснажливих обчислень, що мають достатню кількість процесорів та оперативної пам'яті. Для навчання було обрано дроплет з 16 процесорами та 32 Гб оперативної пам'яті. Через це навчання вищеописаної моделі зменшилося зі 150 хвилин до лише 22.

Choose a plan

[Help me choose](#)

SHARED CPU	DEDICATED CPU			
Basic	General Purpose	CPU-Optimized	Memory-Optimized	Storage-Optimized NEW

Compute-optimized virtual machines with dedicated hyper-threads from best in class Intel processors. Best for CPU-intensive applications like CI/CD, video encoding and transcoding, machine learning, ad serving, batch processing, and active front-end web and application servers.

\$40/mo \$0.060/hour 4 GB / 2 CPUs 25 GB SSD Disk 4 TB transfer	\$80/mo \$0.119/hour 8 GB / 4 CPUs 50 GB SSD Disk 5 TB transfer	\$160/mo \$0.238/hour 16 GB / 8 CPUs 100 GB SSD Disk 6 TB transfer	\$320/mo \$0.476/hour 32 GB / 16 CPUs 200 GB SSD Disk 7 TB transfer	\$640/mo \$0.952/hour 64 GB / 32 CPUs 400 GB SSD Disk 9 TB transfer
--	--	---	--	--

Рис. 3.6. Зображення доступних видів обчислювальних одиниць для машинного навчання на DigitalOcean. [13]

З переваг цього сервісу також слід виділити те, що для старту навчання немає необхідності розбиратися з нуля з новим сервісом, оскільки DigitalOcean надає доступ до серверної машини напряму через командний рядок. Це дуже полегшує запуск навчання, якщо все було підготовлено заздалегідь локально. Тому для початку треба було лише завантажити на сервер необхідні скрипти та дані датасету.

Загалом навчання всіх моделей зайняло ~44 години використовуючи описаний вище сервер. Та єдиною моделлю, яку не вдалося дотренувати до кінця, була AlexNet. Проблема з нею заключається в тому, що її архітектура значно складніша в порівнянні з усіма іншими, велика кількість параметрів значно підвищує час необхідний для навчання та кількість оперативної пам'яті. Саме останнє не дало змогу завершити процес тренування моделі. Оскільки на приблизно 10 та 11 епохах через перевищення кількості використаної оперативної пам'яті процес навчання був примусово завершений oom killer'ом, що відповідає за використані ресурси процесами.

З натренованих моделей найкращу точність мала XceptionNet: 99.3% на тренувальних даних та 93.4% на тестових. Найгіршими виявились LeNet та VggNet16. LeNet мала низьку точність 65.6% на тренувальних даних та 72.1% на тестових через свою достатньо просту та застарілу архітектуру. VggNet16 взагалі не підійшла під цю проблему та мала найнижчу точність в менше ніж 4% на тренувальних та тестових даних. Через це вони точно не підходять для вирішення поточного завдання.

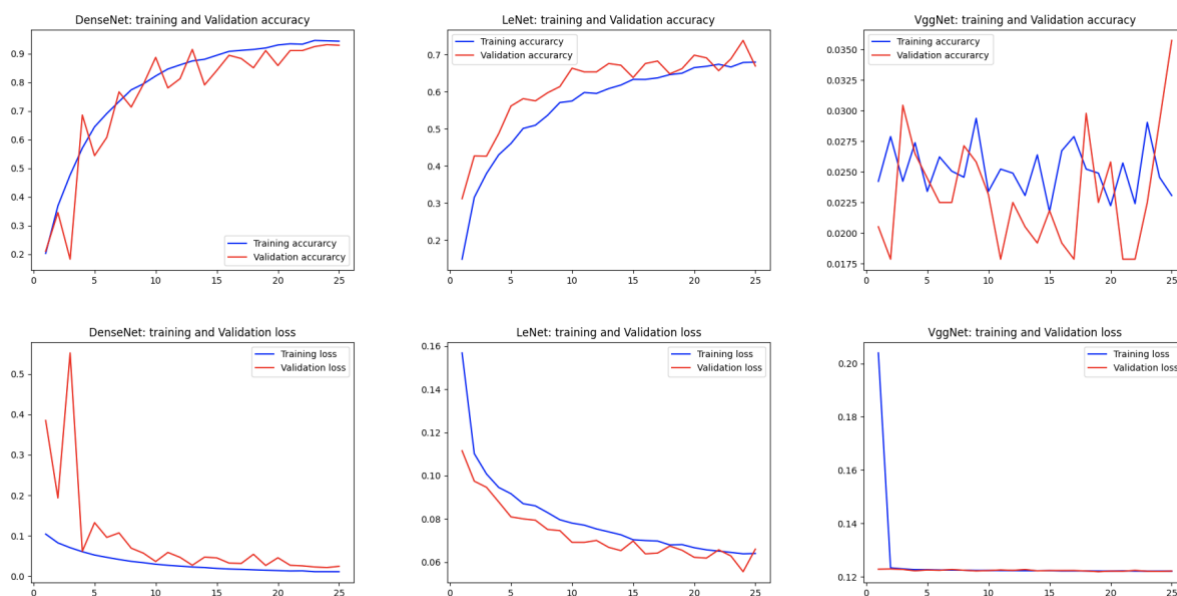


Рис. 3.9. Зображення графіків функції метрики (точності) та функції помилки на тренувальних та тестових даних під час навчання. Ось абсцис містить значення кількості епох тренуваної моделі, а ординат – значення функцій. Для моделей (зліва-направо) - DenseNet121, LeNet, VggNet16.

Наступними етапами для вдосконалення точностей моделей та збільшення їх якості під час використання на реальних прикладах може бути: розширення датасету та/або його заміна на більш новий і такий, що буде містити більшу кількість класів для класифікації; підбирання гіперпараметрів моделей для досягнення максимальної точності кожної з моделей; спроба використати інші функції помилки та оцінки; спроба використати інші моделі глибокого навчання.

3.4 Використання сервісу Google AutoML Vision

Наступним кроком для підвищення точностей вирішення поточної задачі класифікації зображень хвороб рослин за їхнім листям було визначено використати сторонній сервіс для навчання моделей. Оскільки попередньо було вже розроблено нову архітектуру та використано вже

існуючі архітектури нейронних мереж, вирішено скористатися значно іншим рішенням не схожим на попередні.

Таким рішенням стало використання сервісу від Google який називається AutoML Vision. Цей продукт є одним з додатків програми Google AutoML, що розрахована на мінімізацію часу розробників машинного навчання для роботи з моделями. AutoML бере на себе відповідальність за розширення датасету, його нормалізацію та весь процес попередньої обробки, навчання та оцінювання моделей, підбирання параметрів для підвищення якості моделей, підбирання функцій помилок та оцінок. Загалом, цей продукт повністю звільняє розробника від написання будь-якої частини коду, що стосується машинного навчання та відповідно не потребує від нього певних навичок чи знань ШІ. Тобто єдина необхідна річ для швидкого старту з цим сервісом – це лише наявність у розробника датасету та розуміння задачі, що має вирішуватися машинним навчанням.

Для використання цього додатку достатньо мати Google акаунт та можливість створювати проєкти в Google Cloud Console. Далі необхідно створити новий проєкт або використати вже існуючий проєкт користувача і включити в ньому функцію AutoML Vision API. В меню сервісу достатньо завантажити датасет та Google автоматично почне процес навчання моделі. Після завершення навчання можна буде оглянути метрики моделі та вибрати яким чином користувач хоче її використовувати в подальшому (процес деплою). З доступних використань Google пропонує достатню кількість варіантів серед яких: деплой в хмару з використанням REST сервісів Google, збереження моделі у форматі tensorflow для її подальшого запуску на сервері та збереження моделі у форматі tensorflow mobile для її подальшого запуску на мобільних пристроях. [15]

Під час створення нового датасету інтуїтивно зрозумілий інтерфейс підказує користувачу яке завдання машинного навчання він хоче вирішити. У випадку класифікації хвороб рослин за зображенням їх листків – це

Single-Label Classification, тобто класифікація одного класу хворобу за одним зображенням.

Select your model objective

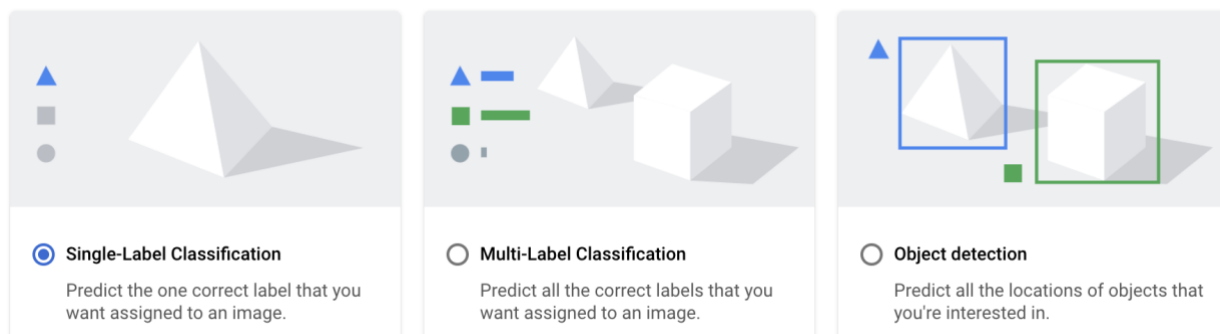


Рис. 3.10. Етап створення нового датасету в сервісі Google AutoML Vision [15]

Після чого Google самостійно проаналізує датасет та підготує всю необхідну інформацію та ресурси для початку навчання.

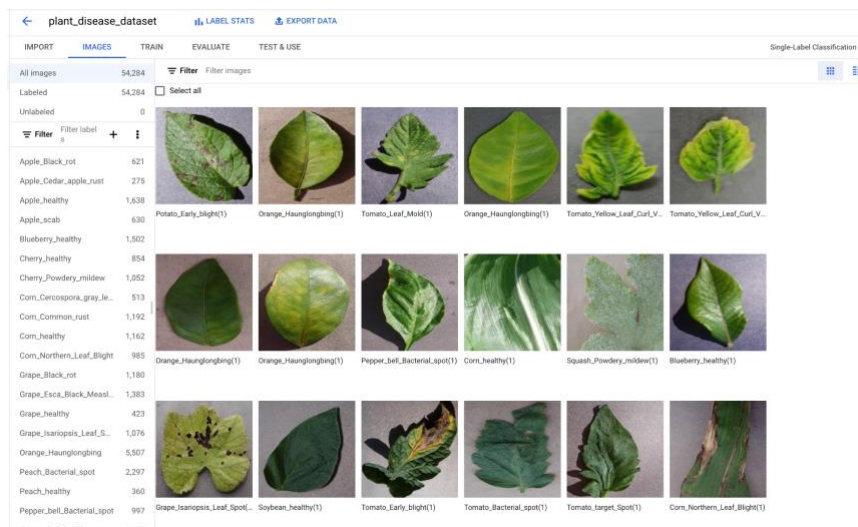


Рис. 3.11. Вигляд імпортованого датасету, що вже був проаналізований та підготовлений до навчання моделі

Train new model

1 Define your model

2 Optimize model for

Goal	Package size	Accuracy	Latency for Google Pixel 2
<input type="radio"/> Higher accuracy	6 MB	Higher	360 ms
<input checked="" type="radio"/> Best trade-off	3.2 MB	Medium	150 ms
<input type="radio"/> Faster predictions	0.6 MB	Lower	56 ms

Please note that prediction latency estimates are for guidance only. Actual latency will depend on your network connectivity.

[CONTINUE](#)

3 Set a node hour budget

Рис. 3.12. Опції, що доступні для конфігурації навчання моделі Google AutoML Vision [15]

All labels

Total images	48,861
Test items	5,423
Precision ?	99.82%
Recall ?	99.61%

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.
[Learn more about these metrics and graphs.](#)

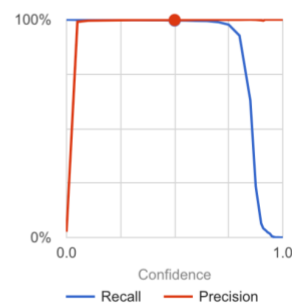
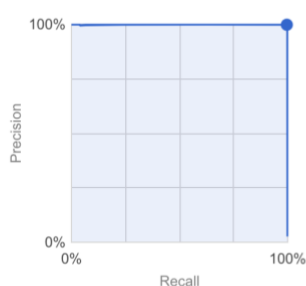


Рис. 3.13. Метрики моделі після її навчання

Після завершення процесу тренування та ознайомлення з метриками навченої моделі, вона буде доступна відповідно до опції деплою, що була обрана попередньо. В рамках цього дослідження натренована модель була задеплойована для локального використання та перевірена на тестових даних для підтвердження результатів, які можна побачити вище на малюнку метрик в інтерфейсі Google AutoML Vision.

Таким чином з'ясовано, що Google AutoML достатньо потужний сервіс що продемонстрував дуже високі результати якості тренування моделей нейронних мереж і яким можна скористатися у випадках, коли

немає часу або недостатньо знань і навичок для роботи з машинним навчанням.

3.5 Написання коду для телеграм боту з використанням навчених моделей

Останнім етапом дослідження необхідно було створити безпосередньо додаток, що буде використовувати попередньо натреновані моделі і буде доступним для використання будь-якому користувачу. Головним критерієм були зручність використання сервісу та швидкість його розробки і подальшого доповнення. Через це було вирішено створити спеціальний телеграм бот, що буде звертатися до моделей машинного навчання та видавати користувачу результат відпрацювання моделей з додатковими інформаціями щодо хвороб рослин.

Для розробки було обрано фреймворк `aiogram` для розробки телеграм ботів на `python` з використанням стандартної бібліотеки асинхронного програмування `asyncio`. [16] Оскільки передбачення моделей на реальних прикладах (`inference time`) може займати достатньо багато часу, було вирішено мінімізувати час на всі інші операції. Для цього було встановлено оптимізовану бібліотеку для роботи з форматами `JSON`, що називається `ujson`. Також бібліотеку `uvloop`, що надає можливість використовувати покращений `event loop` для асинхронних операцій. Та інші прискорювачі асинхронного коду, що працює з мережею, які об'єднані в пакет `aiohttp[speedups]`. [17]

Фрагмент коду, що відповідає за обробку надісланого зображення від користувача та отримання інформації від моделі щодо передбаченого класу хвороби рослини:

```
dest = io.BytesIO()
await message.photo[-1].download(dest)
prediction = await predictor.predict(dest)
confidence, prediction_class = prediction["predictions"][0]
```

Фрагмент коду, що відповідає за обробку зображення та взаємодію з ендпоінтом моделі машинного навчання:

```
start_time = time.monotonic()
encoded_image = self._preprocess_image(image_bytes, 256, 256)
image_id = str(uuid.uuid4())
instances = {'instances': [{'image_bytes': {'b64':
str(encoded_image)}, 'key': image_id}]}
response = await self._fetch(instances)
inference_time = time.monotonic() - start_time
```

Таким чином було створено телеграм бот, що покладається лише на асинхронний код задля зменшення затримок мережі та взаємодії з моделями та швидкого реагування на повідомлення користувачів.

3.6 Тестування програми та результати її виконання

Для більш детального ознайомлення з результатом практичної частини роботи, розглянемо функціонал, що доступний в телеграм боті. В описі чат-боту поданий короткий опис його можливостей. Знайти бот у пошуці в месенджері Telegram можна ввівши його назву: «Plant Disease Detection Bot» або за його юзернеймом «@PlantDiseaseDetectionBot».

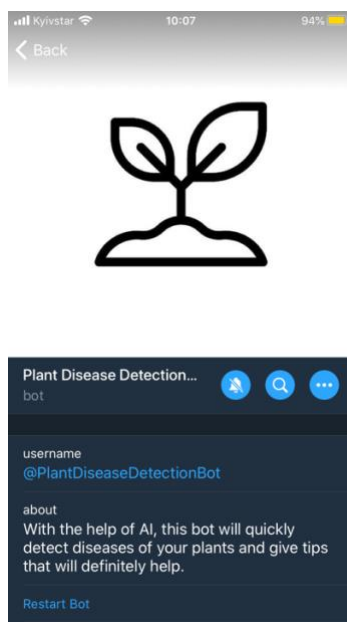


Рис. 3.14. Вигляд профілю чат-боту в месенджері Telegram

Оскільки Telegram має достатньо інтуїтивно зрозумілий інтерфейс користувача, проблем з використанням боту виникнути не мають. Під час запуску бота командою «start» виводиться інформація про те як використовувати чат-бот і що для цього необхідно.

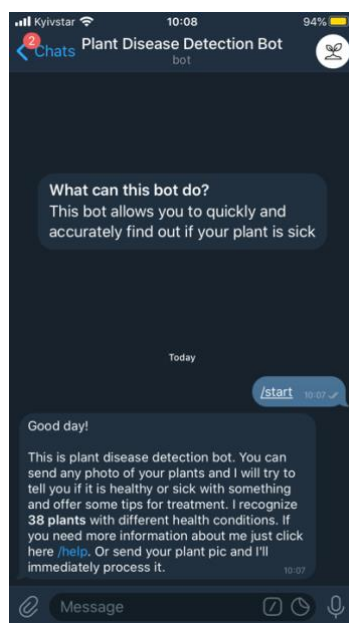


Рис. 3.15. Зображення результату виконання команди «start» в боті

Для отримання більш детальної інформації щодо доступних функцій в боті необхідно скористатися командою «help». Результатом якої буде інтерактивна міні-презентація безпосередньо в месенджері в приватних повідомленнях користувача з ботом. Презентація включає в собі пояснення щодо даних, які використовувалися для навчання моделі нейронної мережі, загальна інформація щодо машинного навчання і яким чином воно було застосоване для вирішення проблеми розпізнавання хвороб рослин, метрики моделі, інформація щодо доступних видів рослин і хвороб для їх класифікації.



Рис. 3.16. Зображення результату виконання команди «help» в боті

Для отримання передбачення щодо хвороби сільськогосподарської культури достатньо лише відправити фотографію боту у зручний користувачеві спосіб. Отримання інформації щодо відпрацювання моделі (класифікації зображення) та інформації щодо лікування, профілактики та догляду за рослиною відбувається за часом до 1 секунди (в середньому - 0.3 секунди).

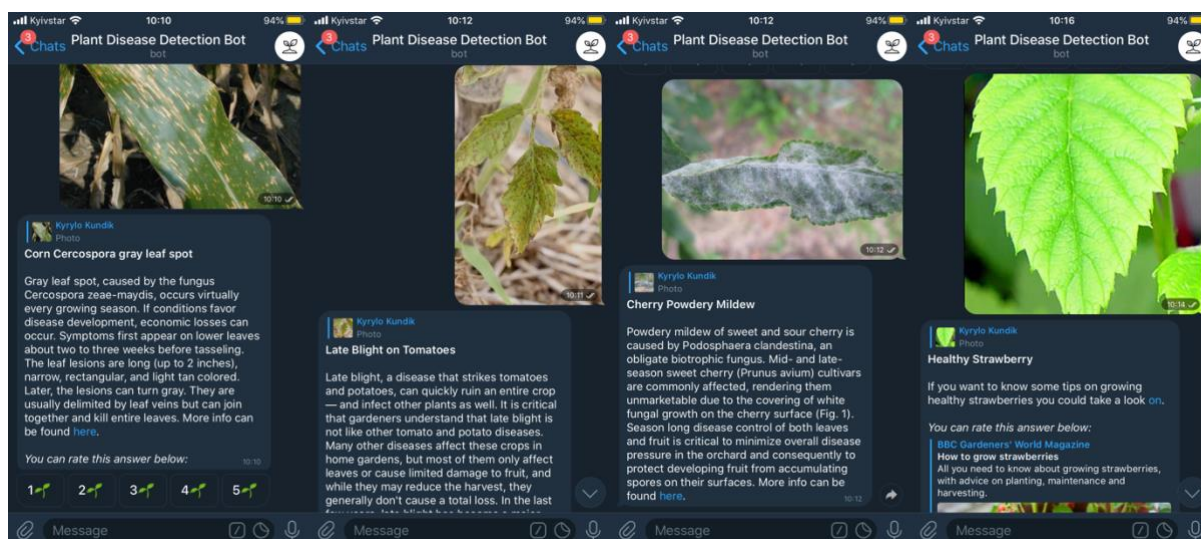


Рис. 3.17. Зображення результатів передбачень чат-бота за фотографіями

Після отримання передбачення від чат-боту у користувача є змога оцінити відповідь за шкалою від 1 до 5. Вся інформація зберігається для її подальшої обробки та конфігурації точностей моделей, підвищення якості відповідей від боту.

Таким чином результат тестування показало, що розроблений Telegram чат-бот разом з моделлю нейронної мережі високої точності повністю виконують поставлену задачу виявлення хвороб сільськогосподарських рослин за допомогою їх зображень, а також надає користувачам вичерпну інформацію щодо запобіганню, поширенню, лікуванню та профілактики таких хвороб.

ВИСНОВКИ

Результатом курсової роботи стала модель нейронної мережі, основною ціллю якої є класифікація хвороб сільськогосподарських рослин за вхідними зображеннями.

Під час написання курсової роботи було досліджено сучасні методи та технології машинного та глибинного навчання, їх практичне застосування в комерційних та дослідницьких проєктах. Досліджено необхідні етапи для роботи з датасетами для тренування моделей штучного інтелекту, їх розширення, нормалізація та обробка перед навчанням.

У третьому розділі курсової роботи детально описано етап проектування нейронної мережі для класифікації хвороб сільськогосподарських рослин. Описані інші популярні архітектури машинного навчання та способи їх застосування. Розроблено програмний інтерфейс, що дозволяє легко інтегрувати нові моделі, створювати звіти щодо їх навчання та спрощує подальшу роботу з їхніми передбаченнями на реальних прикладах застосування.

Для використання моделей на реальних прикладах було окремо розроблено чат-бот для месенджеру Telegram, який інтегрує модель машинного навчання. Для передбачень було використано модель XceptionNet, що має точність 93.4% на валідаційному датасеті проти 81.5% точності моделі власної архітектури. Це зумовлено меншою кількістю параметрів та глибиною архітектури.

В перспективі, бот може значно полегшити роботу сільськогосподарських підприємств та фермерів, що не мають достатньо коштів для найму відповідного персоналу для виявлення хвороб рослин, оскільки моделі нейронної мережі мають велику точність передбачення за зображеннями, а сам бот надає вичерпну інформацію щодо хвороб.

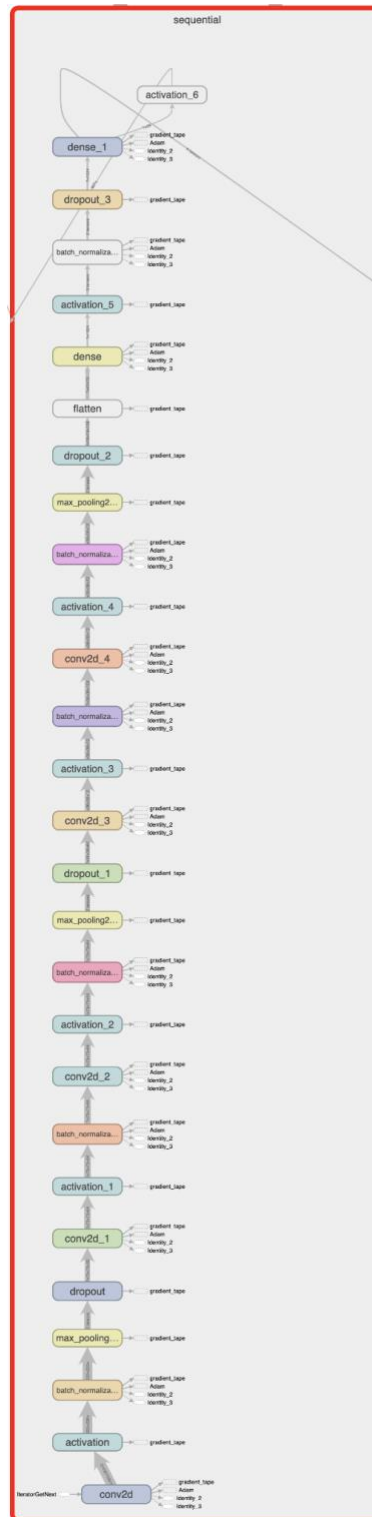
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. [Електронний ресурс] PictureThis: Botanist in your pocket. Online plant encyclopedia and plant identifier. <https://picturethisai.com>
2. [Електронний ресурс] Ada Lovelace Day: We should never forget the first computer programmer. <https://www.independent.co.uk/life-style/women/ada-lovelace-day-first-computer-programmer-forgotten-women-a8557416.html>
3. [Електронний ресурс] Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study. <https://europepmc.org/article/med/2342742>
4. [Електронний ресурс] Deep Learning Book by MIT Press. <https://www.deeplearningbook.org/>
5. [Електронний ресурс] Introduction to Machine Learning. The Wikipedia guide. <http://www.datascienceassn.org/sites/default/files/Introduction%20to%20Machine%20Learning.pdf>
6. [Електронний ресурс] NVidia Developer Blog. Deep Learning in a Nutshell. <https://developer.nvidia.com/blog/deep-learning-nutshell-core-concepts/>
7. [Електронний ресурс] Object Detection and Image Classification with YOLO. <https://www.kdnuggets.com/2018/09/object-detection-image-classification-yolo.html>
8. [Електронний ресурс] Towards Image Classification with Machine Learning Methodologies for Smartphones. <https://www.mdpi.com/2504-4990/1/4/59>
9. [Електронний ресурс] PlantVillage Dataset for Plant Disease Recognition. <https://plantvillage.psu.edu/>
10. [Електронний ресурс] Python OpenCV library. <https://opencv.org/>

11. [Электронный ресурс] Python NumPy library. <https://numpy.org/>
12. [Электронный ресурс] Python Tensorflow framework.
<https://www.tensorflow.org/>
13. [Электронный ресурс] DigitalOcean cloud services.
<https://www.digitalocean.com/>
14. [Электронный ресурс] Xception: Deep Learning with Depthwise Separable Convolutions.
https://openaccess.thecvf.com/content_cvpr_2017/papers/Chollet_Xception_Deep_Learning_CVPR_2017_paper.pdf
15. [Электронный ресурс] Google Cloud AutoML service.
<https://cloud.google.com/automl>
16. [Электронный ресурс] Aiogram fully asynchronous Python framework for Telegram API. <https://docs.aiogram.dev/en/latest/index.html>
17. [Электронный ресурс] Aiohttp asynchronous HTTP client/server for Python. <https://docs.aiohttp.org/en/stable/>
18. [Электронный ресурс] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems by Aurelien Geron.
<https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/>

ДОДАТКИ

Додаток №1



Додаток №2

