МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
В.о. завідувача кафедри інтелектуальних
інформаційних систем, канд. техн. наук, доцент
_____Є. В. Сіденко
«____»_____202_ р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ТЕЛЕГРАМ-БОТ ДЛЯ РОЗПІЗНАВАННЯ УКРАЇНОМОВНОГО ТЕКСТУ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.1710105

Виконав студент 6-го курсу, групи 601
_____ К. Ю. Бурдін
«16» лютого 2023 р.

Керівник: д.ф.-м. наук, професор
_____ Е. А. Лисенков
«16» лютого 2023 р.

Миколаїв – 2023

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри інтелектуальних інформаційних систем, канд. техн. наук, доцент
_____ Є. В. Сіденко
« » 20 р.

ЗАВДАННЯ

на магістерську кваліфікаційну роботу

Бурдіну Кирилу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема магістерської кваліфікаційної роботи «Телеграм-бот для розпізнавання україномовного тексту за допомогою нейронних мереж».
Керівник роботи Лисенков Едуард Анатолійович, д.ф.-м. наук, професор.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «03» листопада 2022 р. № 199
2. Строк подання студентом роботи __ лютого 2023 р.
3. Вхідні (початкові) дані до роботи: користувацькі зображення з україномовним текстом. Очікуваний результат: система розпізнання україномовного тексту.
4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):
   - аналіз сучасного стану задачі;
   - огляд існуючих аналогів;
   - огляд існуючих методів розпізнання тексту;
5. Перелік графічного матеріалу: презентація.
6. Завдання до спеціальної частини: аналіз вимог до умов праці, заходи з техніки безпеки та охорона праці під час роботи у випадку настання надзвичайних ситуацій

7. Консультанти:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис**
Спеціальна частина з охорони праці	д-р. біол. наук., професор Григор'єва Л.І.	
Методична частина	д. ф.-м. наук, професор Лисенков Е.А.	

Керівник роботи __ д. ф.-м. наук, професор Лисенков Е.А._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Бурдін К. Ю._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «____» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
Виконання магістерської кваліфікаційної роботи

Тема: Телеграм-бот для розпізнавання україномовного тексту за допомогою нейронних мереж

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми МКР. Подання заяви на затвердження теми МКР	01.09.2022	20.10.2022	Виконано
2	Отримання завдання на виконання МКР	21.10.2022	10.11.2022	Виконано
3	Складання календарного плану на період виконання МКР	11.11.2022	15.11.2022	Виконано
4	Огляд літератури за темою дослідження	16.11.2022	27.11.2022	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до МКР	28.11.2022	18.12.2022	Виконано
6	Аналіз предметної області та розробка технічного завдання	19.12.2022	12.01.2023	Виконано
7	Опис фахової частини МКР	13.01.2023	25.01.2023	Виконано
8	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2023	02.02.2023	Виконано
9	Попередній захист МКР на засіданні комісії кафедри	03.02.2023	03.02.2023	Виконано
10	Корегування роботи за результатами попереднього захисту	04.02.2023	06.02.2023	Виконано
11	Остаточне оформлення пояснювальної записки та слайдів доповіді для захисту	07.02.2023	09.02.2023	Виконано
12	Подання МКР рецензенту	09.02.2023	10.02.2023	Виконано
13	Рецензування МКР	11.02.2023	12.02.2023	Виконано
14	Подання МКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.02.2023	16.02.2023	Виконано
15	Захист МКР перед екзаменаційною комісією (ЕК)	22.02.2023	23.02.2023	

Розробив студент ___Бурдін Кирило Юрійович_____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи ____ д. ф.-м. наук, професор Лисенков Е.А._ _____
(*наук. ступінь, вчене звання, прізвище та ініціали)*

«__»листопада 2022 р.

АНОТАЦІЯ

Магістерської кваліфікаційної роботи студента групи 601 ЧНУ ім. Петра Могили

Бурдіна Кирила Юрійовича

Тема: «Телеграм-бот для розпізнавання україномовного тексту за допомогою нейронних мереж»

Об'єкт дослідження – процес розпізнавання україномовного тексту за допомогою нейронних мереж.

Предмет дослідження – методи дослідження розпізнавання україномовного текст.

Мета дослідження – розробка телеграм-бота для розпізнавання україномовного тексту за допомогою нейронних мереж.

Робота складається з зі вступу, 3 розділів та висновку.

У першому розділі розглядається сфера розпізнавання текстових символів, наявні інструменти для їх розпізнання.

У другому розділі проведено аналіз нейронних технологій і обрано інструментальні засоби розробки застосунку.

У третьому розділі описано реалізацію застосунку для розпізнавання україномовного тексту.

В результаті розроблено телеграм-бот для розпізнавання україномовного тексту.

Магістерська дипломна робота містить 86 сторінок, 38 рисунків та 41 використаних джерел.

Ключові слова: C#, Telegram, розпізнавання тексту.

ABSTRACT

master's qualification work of a student of group 601 Petro Mohyla Black SeaNational University

Burdin Kirill Yurievich

Topic: «Telegram bot for recognizing Ukrainian text using neural networks»

The object of the research is the process of recognizing Ukrainian-language text using neural networks.

The subject of the study is research methods of Ukrainian-language text recognition.

The purpose of the research is to develop a Telegram bot for recognizing Ukrainian-language text using neural networks.

The work consists of an introduction, 3 chapters and a conclusion.

The first chapter examines the field of handwritten materials and the available tools for their recognition.

In the second section, an analysis of neural technologies was carried out and instrumental means of application development were selected.

The third chapter describes the implementation of the application for recognizing Ukrainian-language text.

As a result, a Telegram bot was developed for recognizing Ukrainian-language text.

The master's thesis contains 86 pages, 38 figures and 41 used sources.

Keywords: C#, Telegram, text recognition.

ЗМІСТ

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«ТЕЛЕГРАМ-БОТ ДЛЯ РОЗПІЗНАВАННЯ УКРАЇНОМОВНОГО ТЕКСТУ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ»

Спеціальність 122 «Комп’ютерні науки»

122 – МКР – 601.1710105

Виконав студент 6-го курсу, групи 601
_____ К. Ю. Бурдін
«16» лютого 2023 р.

Керівник: д.ф.-м.н, професор
_____ Е. А. Лисенков
«16» лютого 2023 р.

Миколаїв – 2023

ВСТУП

Нейроні мережі починають використовуються все більшій кількості сфер, в яких їх використовують починаючи від допомоги у захисті інформації на мобільних пристроях і закінчуючи прогнозуванням економічних або природних процесів.

Розпізнавання тексту є дуже складним завданням з теоретичної та практичної точки зору. Людина, наприклад, використовує весь спектр знань і досвіду. Він визначає текст із набору смислових сигналів, ідентифікує кожен символ, визначає характеристики символів і на основі свого досвіду приходить до висновку про значення символу та тексту загалом.

Актуальність даного дослідження полягає у необхідності підвищення ефективності сервісів розпізнання текстів з зображень.

Метою дослідження є проведення дослідження стану технологій розпізнавання текстових даних з зображень різних форматів.

Об’єкт дослідження є методи розпізнавання тексту, алгоритми машинного навчання, придатні для виявлення тексту на зображеннях.

Предметом дослідження є обробка зображень з метою виділення, класифікації та подальшого використання текстової інформації з зображень.

Апробація результатів дослідження. Основні положення дипломної роботи, викладено в матеріалах Всеукраїнській науково-практичній конференції молодих вчених, аспірантів і студентів «Інформаційні технології та інженерія».

Завдання розпізнавання тексту поділено на підзадачі: відфільтрувати зображення від шуму, видалити не текстові зображення із текстових, вибрати ознаки символів та порівняти ці ознаки із збереженими зразками. Кожне завдання містить багато рішень, лише деякі з яких є більш-менш оптимальними.

В даній роботі всі дослідження і експерименти проводилися з набором реальних даних задля досягнення максимальної ефективності в задачі розпізнання україномовних текстів за допомогою нейромереж.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ. ПОСТАНОВКА ЗАДАЧІ

1.1 Постановка задачі

Завдання розпізнавання тексту є залишається актуальним на сьогоднішній день, тому що не існує стовідсоткової універсальної системи з розпізнавання тексту.

Головна проблема, яку вирішує розпізнавання друкованого введення – економія часу. На те, щоб вручну передрукувати текст, потрібно витратити велику кількість часу, при цьому ця робота швидко втомлює. Комп'ютерні програми можуть значно полегшити таку повсякденну працю.

Система розпізнання тексту буде складатися з додатка, який оброблятиме вхідні зображення та оформляти текст в електронному вигляді. Головне завдання нейромережі – розпізнавання символів на зображеннях та виявляти їх аналоги в електронному варіанті.

Розпізнавання тексту включає такі етапи [1]:

– вхідне зображення повинно бути очищене від шуму і приведено до вигляду, що дозволяє ефективно виділяти символи і розпізнавати їх;
– система розбиває зображення на блоки тексту, ґрунтуючись на особливостях його вирівнювання;
– зображення з текстом поділиться на зображення рядків, а потім на зображення символів для того, щоб обробити кожен символ окремо;
– зображення символу може оброблятися повністю, для цього воно порівнюється з наявними шаблонами, розпізнається за допомогою нейронних мереж;
– на виході з'являється можливий варіант літери.

1.2 Огляд та аналіз наявних аналогів

ABBYY FineReader (рис. 1.1.). Дана програма розробляється з 1993 року. Програма реалізує метод оптичного розпізнання. Додаток працює зі сканером, підтримує формат зображень jpg, jpeg, png, gif, bmp, перекладає pdf у Word, Excell [2].

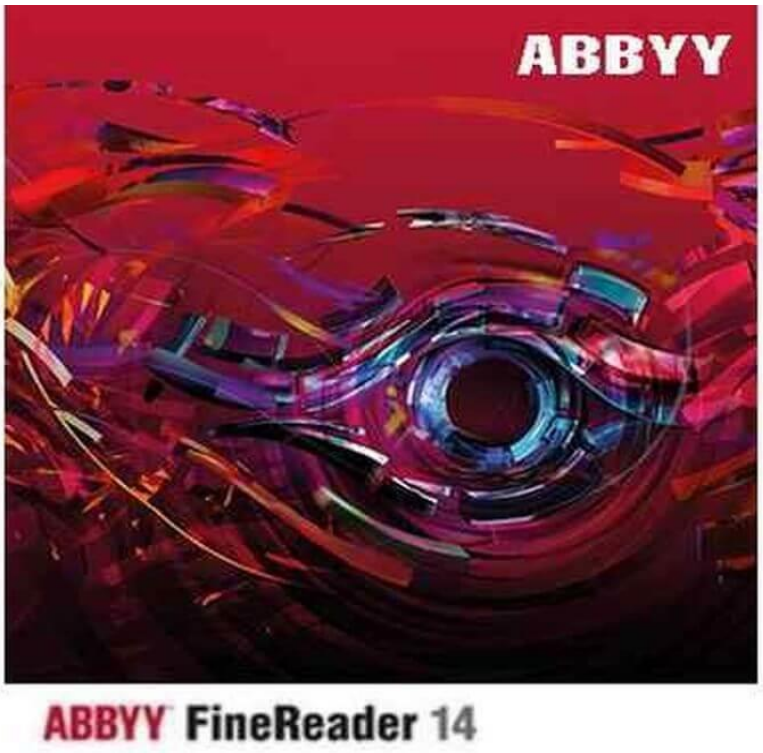

Рисунок 1.1 – ABBYY FineReader

Особливості програми:

– остання версія програми розпізнає рукописний текст 192 мовами, при цьому для 48 мов є підтримка перевірки орфографії;
– програма підтримує роботу з багатьма форматами;
– здібність стискати оригінальні файли зображень без явної втрати якості;
– повнофункціональна пробна версія;
– синхронізація з зовнішніми сервісами;
– програма визнана експертами та користувачами. Їй неодноразово присуджували різні нагороди.

Єдиний недолік програми полягає в тому, що вона платна. Тому вона підійде для тих, в кого є постійна потреба в функції розпізнавання тексту.

SimpleOCR рис. 1.2.). Дана програма менш досконала ніж попередня але основну функцію – розпізнання рукописних матеріалів, виконує. Існує дві версії програми – безкоштовна та платна. Безкоштовна версія містить усі потрібні звичайному користувачу функції, такі як: розпізнання і конвертація рукописних матеріалів в потрібний формат [3].

Дана програма має такі переваги:

- немає серверних систем для вивчення чи підтримки, лише настільна програма;
- не покладайтеся на власні технології для доступу до ваших файлів;
- використовує вбудовані дозволи на файли Active Directory;
- використовуйте наявні права доступу або редагуйте їх із SimpleView;
- файли можна ділитися з іншими програмами;
- немає необхідності імпортувати та позначати файли вручну;
- використовуйте існуючі плани резервного копіювання.

Рисунок 1.2 – SimpleOCR

PDFelemnt PRO – комплексне вирішення завдань роботи з PDF файлами. Дана програма на рівних конкурує зі своїми аналогами. Є багато функцій, які можуть знадобитися при роботі з текстом, у тому числі функція розпізнавання тексту [4].

PDFelement PRO створено для роботи з форматом PDF. Безкоштовна версія програми дозволяє редагувати, анотувати, створювати, об'єднувати та розділяти PDF-файли. Розпізнавання тексту доступне лише у платній версії, але якість та результати роботи залишаються на високому рівні.

За допомогою програми можна робити документи конфіденційними, можна створювати готові шаблони, ставити штами і т.д. Загалом програма підійде більше тим, хто працює з PDF файлами. Хоча в ній є функція розпізнавання рукописного введення, вона тут не є головною.

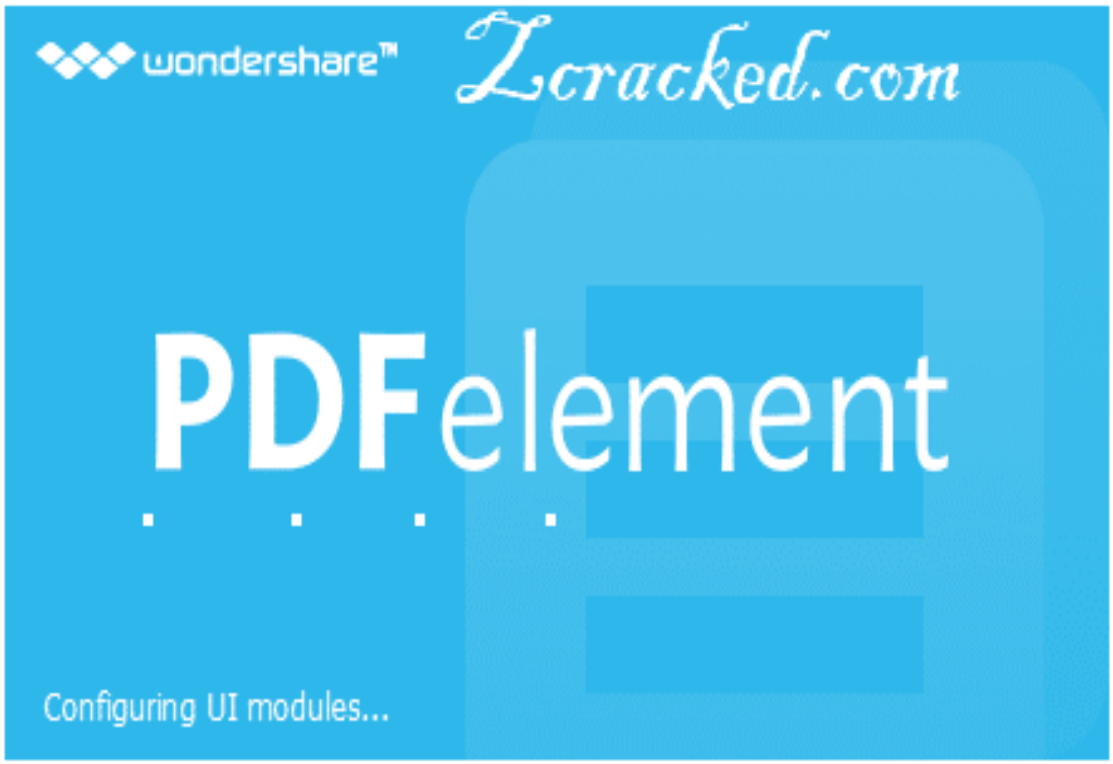

Рисунок 1.3 – PDFelemnt PRO

OCR Desktop (Free Online OCR, рис. 1.4.). Основні особливості програми полягає в тому, що її можна використовувати в онлайн-режимі, при цьому повністю безкоштовна. Програма знадобиться тим користувачам, кому потрібно розпізнати текст і створити його цифрову версію [5].

Інтернет-сервіс працює із такими форматами як: PDF, JPEG, PNG, GIF та ін. Завантаживши документ, можливо з високою точністю перевести рукописний текст на друкований формат. Тексти розпізнаються нейромережею, якому на навчання у розпізнаванні текстів надали 4 мільйони прикладів. Завдяки цьому точність розпізнавання висока. Безкоштовність і робота в режимі онлайн - чудовий привід використовувати цю програму, якщо потрібно розпізнати рукописні матеріали.

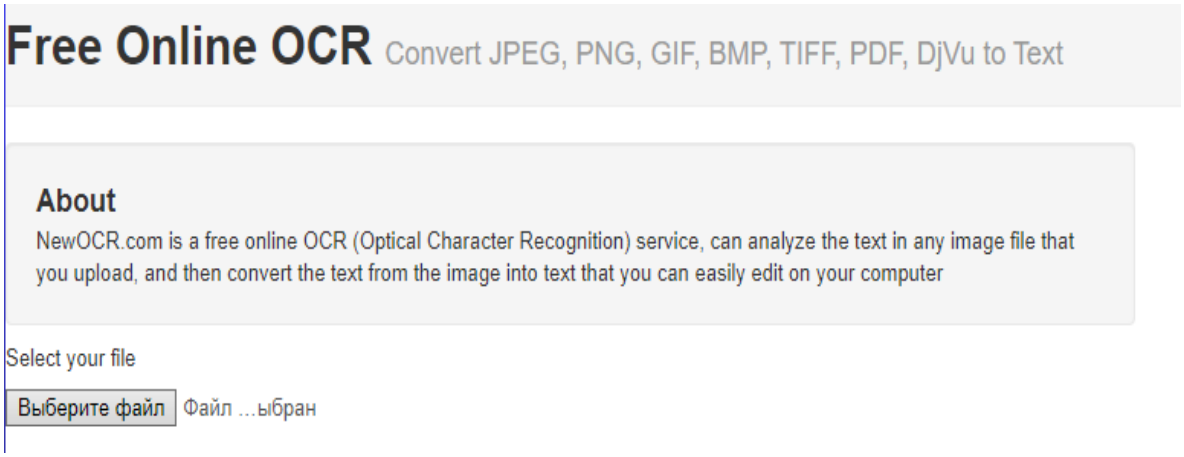

Рисунок 1.4 – Free Online OCR

TopOCR. Інше онлай-рішення в області розпізнавання тексту. Раніше програма була безкоштовною, але на сьогоднішній час її потрібно придбати, щоб використовувати [6]. З іншого боку, вона має переваги:

- нейромережа для роботи з текстами — одна з найновітніших, на гідному рівні конкурує з аналогами;
- TopOCR підтримує конвертацію із зображення у формат документа;
- розробники представили оригінальний девайс - спеціальну камеру, яка може автоматично розпізнати текст на аркуші паперу, досить просто піднести його до камери.

Рисунок 1.5 – TopOCR

Висновки до розділу 1

У першому розділі було визначено, що предметом дослідження являються системи розпізнання тексту. За допомогою проведення аналізу існуючих рішень було виявлено основні сучасні системи розпізнання тексту та проведений їхній аналіз. Була сформована задача, яка буде проаналізована під час роботи з магістерською дисертацією. Було вказано призначення розробки системи розпізнання тексту, цілі та задачі, які повинні бути досягнуті під час роботи з системою розпізнання тексту. Наступний розділі буде присвячений вибору інструментів для розробки.

2 МЕТОДИ ТА ІФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Нейронні мережі

Нейронні мережі є одним із напрямків наукового дослідження в рамках дисципліни створення штучного інтелекту (ШІ), що базується на імітації нервової системи людини. Включаючи здатність виправляти помилки та навчатись. Усе це має дозволити імітувати функціонування людського розуму.

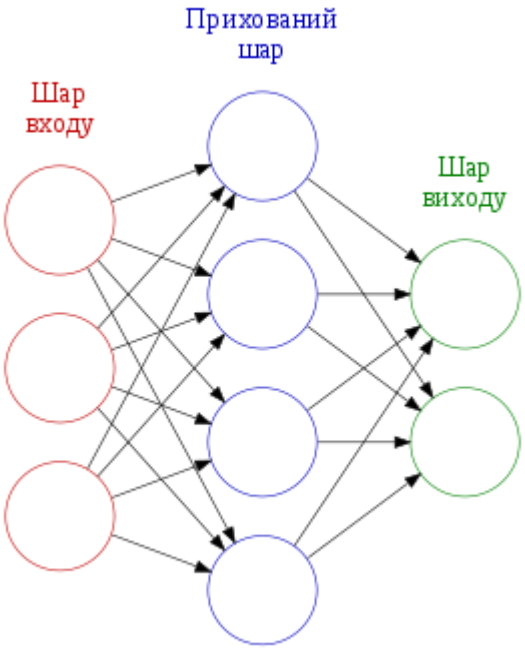

Рисунок 2.1 – Штучна нейронна мережа

Органічний нейрон – це клітина, що включає ядро, тіло і відростки, і вона тісно пов’язана із сотнями інших нейронів. Через це з’єднання передаються електрохімічні імпульси, які переводять всю нейронну мережу у стан збудження або навпаки спокою. Наприклад, приємна і водночас хвилююча подія генерує електрохімічний імпульс у нейронній мережі, розташованій в нашій голові, що призведе до її взбудження. Як кінцевий результат того факту, нейронна мережа в нашому мозку передасть збудження на інші органи нашого тіла, і може призвести до підвищеного серцебиття, частішого моргання очима тощо [7].

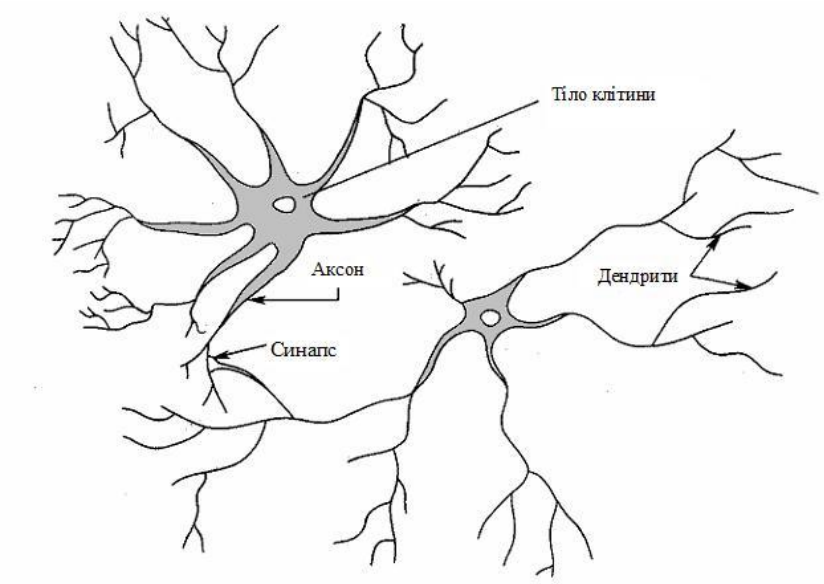

Рисунок 2.2 – Органічний нейрон

Штучна нейронна мережа (ІНС, НС) – це паралельно розподілена структура обробки інформації, що складається з нейронів, пов'язаних між собою. Модель нейронної мережі в Програмування є машинною інтерпретацією головного мозку. Центральна нервова система людини представлена спинним та головним мозком, функціональність яких здійснюється за допомогою величезного кількості нейронів, які пов'язані між собою синаптичними зв'язками та можуть передавати інформацію за допомогою імпульсів [8].

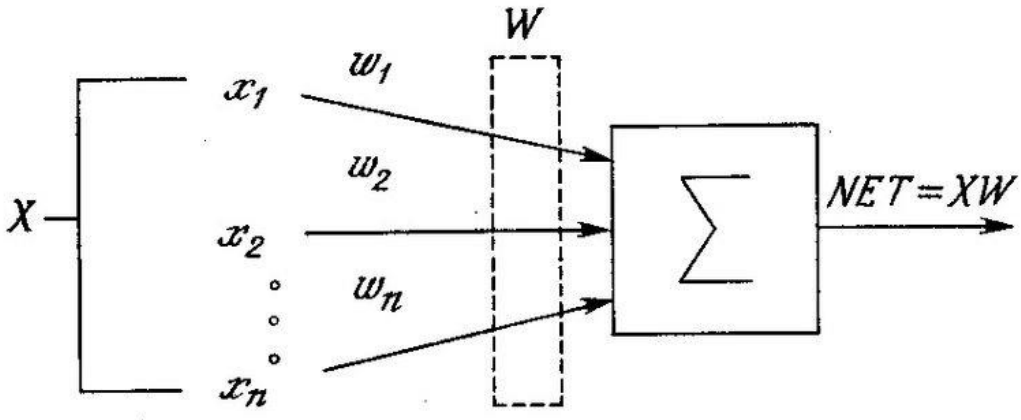

Рисунок 2.3 – Штучний нейрон

Користь штучних нейронних мереж зростає з кожним роком, на сьогоднішній день вони використовуються в таких сферах як:

– машинне навчання, тобто свого роду штучний інтелект. Він повністю базується на навчанні штучному інтелекту насамперед на прикладі мільйонів подібних завдань. В даний час Google, Bing Baidu, і Google, активно впроваджують машинну навчання в пошукові системи. Отже, базуючись на сотнях тисяч пошукових запитів, які всі ми щодня здійснюємо в Google, їхні дослідницькі алгоритми дають нам максимально релевантні результати, щоб ми могли виявити саме те, що шукаємо;

– у робототехніці нейронні мережі використовуються в рамках розробки чисельних алгоритмів для роботів;

– архітектори комп’ютерних гаджетів використовують нейронні мережі, щоб вирішити проблему паралельних обчислень;

– за допомогою нейронних мереж математики можуть вирішувати різноманітні складні математичні задачі.

В цілому для різних задач застосовуються різні види і типи нейронних мереж, серед яких можна виділити:

- багатошаровий персептрон;
- згорткові нейронні мережі;
- рекурентні нейронні мережі.

Багатошаровий персептрон складається з багатьох сенсорних вузлів, які формують вхідний рівень; один або більше прихованих шарів обчислювальних нейронів і один вихідний шар нейронів. Вхідний сигнал поширюється по мережі безпосереднім шляхом, від рівня до рівня [9].

Багатошарові перцептрони ефективно використовуються для вирішення різних складних проблем. При цьому навчання з вчителем завершується за допомогою такого алгоритму, як набір правил зворотного поширення

помилок.

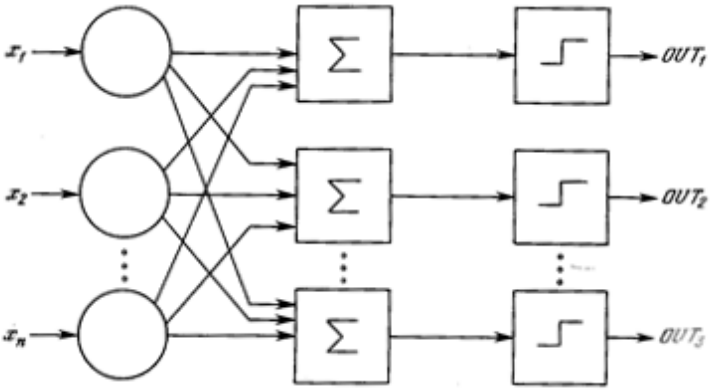

Рисунок 2.4 – Багатошаровий персептрон

Згорткові мережі є одним з найпопулярніших різновидів штучних нейронних мереж. Вони підтвердили свою ефективність у розпізнаванні образів (відео та зображень), системах рекомендацій та обробки мови.

Згорткові нейронні мережі масштабуються і можуть використовуватися для розпізнавання образів будь-якого масштабу.

Ці мережі використовують об’ємні тривимірні нейрони. В межах одного шару нейрони найефективніше зв’язуються за допомогою невеликого поля, яке називається рецептивним шаром.

Нейрони суміжних шарів пов'язані за допомогою механізму просторової локалізації. Робота багатьох таких шарів забезпечується використанням унікальних нелінійних фільтрів, які реагують на дедалі більше пікселів.

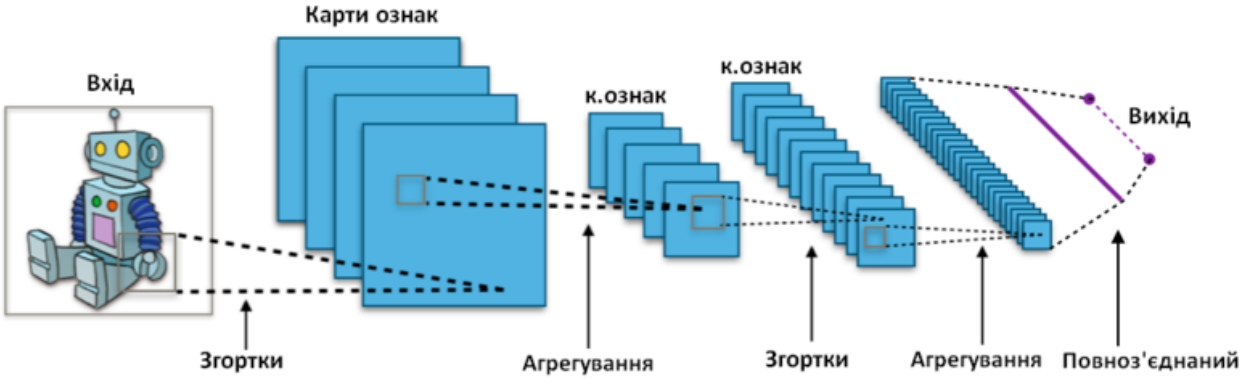

Рисунок 2.5 – Архітектура згорткової нейронної мережі

Рекурентні нейронні мережі – це категорія штучних нейронних мереж, у яких зв’язки між вузлами утворюють орієнтований у часі граф. Це створює внутрішній стан мережі, який дозволяє їй демонструвати динамічну поведінку в часі. На відміну від нейронних мереж з прямим поширенням, РНМ можуть використовувати свою внутрішню пам’ять для обробки довільних послідовностей вхідних даних. Це робить їх актуальними для виконання задач, які включають розпізнавання несегментованого безперервного рукописного тексту та мовлення [10].

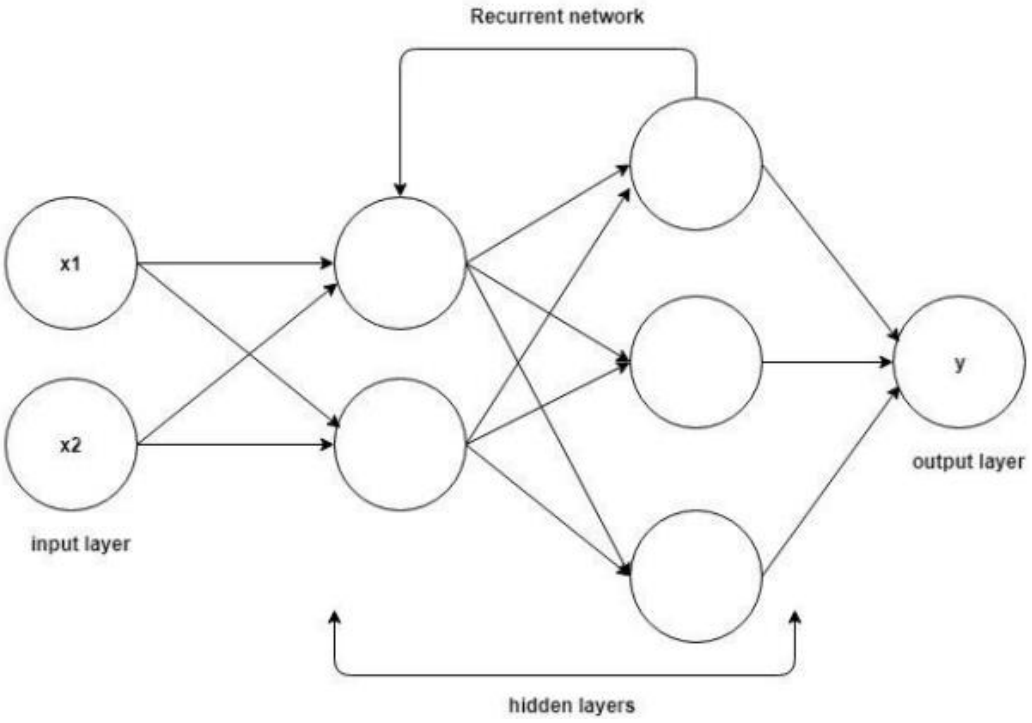

Рисунок 2.6 – Загальна схема рекурентної нейронної мережі

2.2 Оптичне розпізнавання символів

Оптичний індивідуальний зчитувач (OCR) – це цифрове або механічне перетворення зображень надрукованого, рукописного або сканованого текстового вмісту в машинно-кодований текст, будь то сканований документ, зображення файлу, фотографія або з текст субтитрів, накладеного на зображення [11].

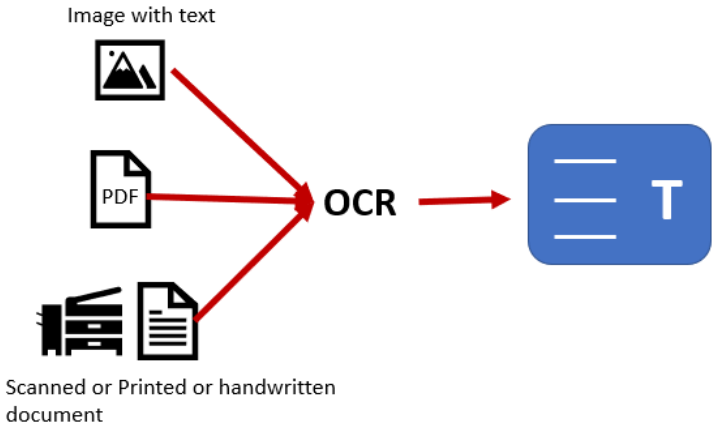

Рисунок 2.7 – Оптичне розпізнання символів

Широко використовується як форма запису даних із друкованих паперових записів – будь то паспортні документи, рахунки-фактури, виписки фінансових установ, квитанції, пошта, роздруківки статичних фактів чи будь-яка відповідна документація – це звичайний метод оцифрування опублікованих текстів для того, щоб їх можна було редагувати в електронному вигляді, шукати, зберігати більш компактно, відображати в Інтернеті та використовувати в машинних процесах разом із когнітивними обчисленнями, машинним перекладом, текстовий вмістом у мовлення, ключовими данні і аналіз тексту. OCR – це дисципліна, яка вивчає розпізнавання образів, штучний інтелект і комп’ютерне бачення.

У 1929 році Густав Таушек отримав патент на підхід до оптичного розпізнання текстового вмісту в Німеччині, після чого Гендель отримав патент на свій підхід у Сполучених Штатах у 1933 році. У 1935 році Таушек також отримав патент США на свій метод. Пристрій Таушека став механічним пристроєм, який використовував шаблони та фотодетектор.

У 1950 році Девід Х. Шепард, криптоаналітик з Агентства безпеки збройних сил США, ознайомившись із труднощами перетворення друкованих повідомлень на машинну мову для обробки комп’ютером, створив гаджет, який вирішує цю проблему. Отримавши патент США, він представив його у Washington Daily News (27 квітня 1951) і New York Times (26 грудня 1953). Потім Шепард заснував компанію що розробляє інтелектуальні машини, яка

незабаром випустила перші в світі комерційні оптичні індивідуальнісистеми розпізнання.

Перший промисловий пристрій було встановлено на Reader's Digest у 1955 році. Другий пристрій був проданий Standard Oil для перегляду і обробки кредитних карток. Інші системи, обладнані Shepard, були запропоновані в 1950-ті роки разом із сканером веб-сторінок для ВВС США для перевірки та телетайпу машинописних повідомлень. Пізніше IBM отримала ліцензію на патенти Шепарда.

Приблизно в 1965 році Reader's Digest і RCA розпочали співпрацю, щоб створити пристрій для зчитування записів, використовуючи оптичне розрізнання тексту, призний для оцифровки серійних номерів купонів Reader's Digest, які повертаються з рекламних оголошень. Для документів, надрукованих барабанним принтером RCA, використовувався унікальний шрифт OCR-A Машина для читання документів працювала безпосередньо з комп'ютером RCA 301 (один із перших масових комп’ютерів). Швидкість гаджета досягла 1500 документів у хвилину: він перевіряв кожен звіт, крім тих, які не міг ефективно обробити. Поштова служба Сполучених Штатів використовує машини OCR для сортування пошти з 1965 року, основані на генерації, розробленій дослідником Яковом Рабіновим. У Європі основним підприємством, яке застосувало машини OCR, було Британське поштове відомство. Пошта Канади використовує оптичні індивідуальні системи розпізнавання з 1971 року. На початковому етапі центр сортування за допомогою оптичного пристрою розпізнавання отримував дані одержувача та друкував штрих-код. Він наноситься спеціальним чорнилом, яке добре видно в ультрафіолеті. Це зроблено для того, щоб уникнути плутанини з полем адреси яке може бути всюди на конверті.

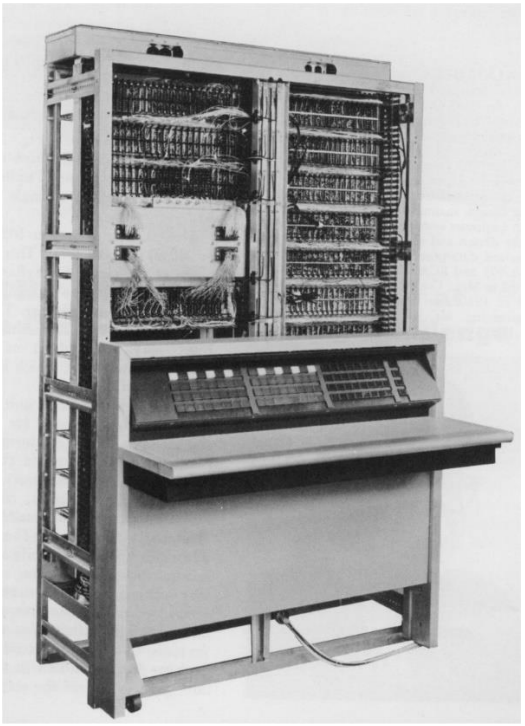

Рисунок 2.8 – RCA 301

У 1974 році Рей Курцвейл заснував компанію Kurzweil Computer Products і почав працювати над вдосконаленням першої системи оптичного розпізнавання символів, здатного розпізнавати текстовий вміст, надрукований будь-яким шрифтом. Курцвейл вважав, що кращим застосуванням цієї технології було б створення системи читання для сліпих, яка могла б дозволити сліпим людям мати комп’ютер, який міг би читати текст вголос. Цей пристрій потребував відкриття відразу двох технологій — планшетного ПЗЗ-сканера та синтезатора тексту в мову. Остаточний продукт був представлений 13 січня 1976 року під час прес-конференції під головуванням Курцвейла та лідерів Національної федерації сліпих.

Рисунок 2.9 – Персональна версія машини для читання Курцвейла

У 1978 році компанія Kurzweil Computer Products почала рекламувати бізнес-версію комп'ютерного програмного забезпечення OCR. Через два роки Курцвейл продав свою компанію Xerox, яка зацікавилася додатковою комерціалізацією систем розпізнання тексту. Kurzweil Computer Products стала дочірньою компанією Xerox, відомою як Scansoft.

У ранніх версіях потрібно було навчати із зображеннями кожного символу та працювати над одним шрифтом за раз. Удосконалені структури, здатні на високу ступінь точності розпізнавання для більшості шрифтів, насправді є звичайним явищем, а також мають підтримку для різноманітних форматів даних цифрових зображень. Деякі системи здатні відтворювати відформатований вихід, який дуже наближений до первинної сторінки, що включає в собі зображення, стовпці і різні нетекстові компоненти.

Класифікують різні види технологій OCR на основі їх використання та застосування. Нижче наведено лише деякі приклади:

Програми простого оптичного розпізнавання символів. Простий механізм OCR застосовує безліч різних шаблонів, що зберігаються, шрифтів і зображень тексту в якості шаблонів. Програмне забезпечення OCR використовує алгоритми зіставлення шаблонів для порівняння зображень тексту з внутрішньою базою даних. Підхід, у якому система зіставляє текст слово за словом, називається оптичним розпізнаванням слів. Він має свої обмеження, оскільки існує практично необмежену кількість шрифтів та стилів почерку, і кожен окремий тип не може бути врахований та збережений у базі даних.

Програми інтелектуального розпізнавання символів. Сучасні системи OCR використовують технологію інтелектуального розпізнавання символів (ICR) для зчитування тексту так само, як це робить людина. Вони використовують передові методи машинного навчання людських навичок читання. Система машинного навчання, звана нейронною мережею, аналізує текст багатьох рівнях, багаторазово обробляючи зображення. Вона шукає різні атрибути зображення (криві, лінії, перетину та петлі) і поєднує результати різних рівнів аналізу для отримання остаточного результату. Незважаючи на те, що ICR обробляє зображення за символами, процес не займає багато часу, а результати виходять за лічені секунди.

Інтелектуальне розпізнавання слів. Інтелектуальні системи розпізнавання слів працюють за тим самим принципом, що й ICR, але обробляють зображення цілих слів без попереднього виділення символів у зображенні.

Оптичне розпізнавання знаків. Оптичне розпізнавання символів дозволяє ідентифікувати логотипи, водяні знаки та інші позначення документа.

Банківська сфера використовує OCR для обробки та перевірки документів щодо кредитів, депозитних чеків та інших фінансових операцій. Така перевірка дозволила підвищити ефективність боротьби з шахрайством та зміцнити безпеку транзакцій. Наприклад, BlueVine, фінансова технологічна

компанія, що надає фінансування малому та середньому бізнесу, використовувала Amazon Textract, хмарний сервіс OCR для розробки продукту, за допомогою якого малі бізнеси в США можуть швидко отримати доступ до кредитів за Програмою захисту заробітної плати в рамках пакету заходів щодо стимулювання економіки в умовах COVID-19. Amazon Textract автоматично обробляв та аналізував десятки тисяч форм на день, завдяки чому BlueVine змогла допомогти кільком тисячам підприємств отримати кошти та зберегти понад 400 000 робочих місць.

У системі охорони здоров'я OCR використовується для обробки історій хвороби пацієнтів, включаючи лікувальні процедури, аналізи, лікарняні картки та страхові виплати. OCR допомагає оптимізувати робочий процес та скоротити обсяг ручної роботи у лікарнях, а також підтримувати актуальність записів. Наприклад, компанія nib Group забезпечує медичне страхування понад 1 мільйон австралійців та щодня отримує тисячі заявок на виплату страхового відшкодування за отримання медичних послуг. Клієнти компанії можуть сфотографувати свій медичний рахунок та відправити його через мобільний додаток nib. Amazon Textract автоматично обробляє ці зображення, що дозволяє компанії набагато швидше розглядати заявки.

Логістичні компанії використовують OCR для більш ефективного відстеження етикеток на упаковках, рахунках, квитаціях та інших документах. Наприклад, компанія Foresight Group використовує Amazon Texttract для автоматизації обробки рахунків у SAP. Введення таких документів вручну віднімало багато часу і призводило до помилок, оскільки співробітникам Foresight надходило вводити дані в кілька систем бухгалтерського обліку. Завдяки Amazon Texttract програмне забезпечення компанії Foresight стало більш точно розраховувати символи на різних носителях і підвищити ефективність ведення бізнесу компанії.

2.3 Telegram

Універсально доступна безкоштовна, кросплатформна, зашифрована, хмарна та централізована служба обміну миттєвими повідомленнями (IM) – це Telegram Messenger. Крім того, програма пропонує низку інших функцій, включаючи VoIP, обмін файлами, ключовий чат і відеодзвінки, а також додаткові чати з наскрізним шифруванням. 14 серпня 2013 року та 20 жовтня 2013 року він був випущений для Android. П’ять центрів обробки даних розташовані по всьому світу, а функціональний центр Telegram знаходиться в Дубаї, Об’єднані Арабські Емірати. Для реєстрації потрібен пристрій iOS або Android і робочий номер телефону, існує багато клієнтських програм, доступних для настільних і мобільних систем, включаючи стандартні програми для Android, iOS, Windows, mac і Linux. Крім того, протокол Telegram використовується великою кількістю неофіційних клієнтів, а також двома стандартними веб-додатками, WebK і WebZ [12].

Рисунок 2.10 – Telegram

Telegram пропонує додаткові чати з наскрізним шифруванням. Інтернет-провайдери та інші треті сторони в мережі не можуть отримати дані, оскільки хмарні розмови та групи зашифровані між клієнтом і сервером. Користувачі можуть ділитися необмеженою кількістю зображень, документів, місць розташування споживачів, анімованих наклейок, контактів і звукових файлів. Користувачі також можуть створювати голосові та відеодзвінки. За програмами також можуть стежити користувачі.

Telegram мав понад 500 мільйонів активних користувачів щомісяця станом на січень 2021 року. Станом на кінець серпня 2021 року він отримав 1 мільярд завантажень, що робить його додатком з найбільшою кількістю завантажень у світі. Дослідження показують, що хвиля Telegram у січні 2021 року була здебільшого викликана деплатформуванням Parler великими технологічними компаніями. У червні 2022 року кількість активних користувачів Telegram перевищила 700 мільйонів щомісяця. Того ж тижня було представлено Telegram Premium, додаткову платну підписку з деякими додатковими функціями.

Telegram випередив WhatsApp і Facebook Messenger і став найпопулярнішим додатком для обміну миттєвими повідомленнями в Молдові, Йорданії, Вірменії, Азербайджані, Казахстані, Киргизстані, Камбоджі, Ефіопії, та Україні.

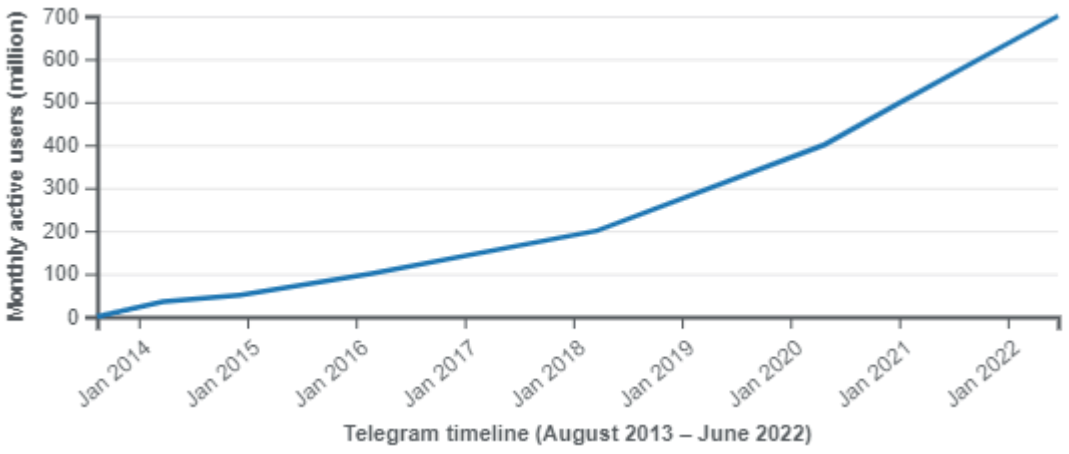

Рисунок 2.11 – Ріст популярності Telegram

Месенджер використовує кілька протоколів кодування для створення протоколу MTProto. Під час надсилання повідомлень протоколу в мережу AES використовує ключ, відомий як клієнту, так і серверу, щоб зашифрувати їх за допомогою алгоритмів RSA-2048 і DH-2048. Крім того, використовуються криптографічні хеш-суми SHA-1 і MD5.

Лише «секретні» чати (Secret Chats), які доступні з 8 жовтня 2013 року, забезпечують захист від перехоплення сервером Telegram пересланих повідомлень. Цей режим використовує алгоритм AES-256 у режимі IGE (Infinite Garble Extension) для пересланих повідомлень для реалізації шифрування, де лише відправник і одержувач мають спільний ключ (наскрізне шифрування). На відміну від стандартного налаштування, повідомлення в секретних чатах ніколи не розшифровуються сервером і зберігаються лише в історії двох пристроїв, створених чатом.

Якщо ви використовуєте інтелектуальну версію iOS або Android під час обміну файлами, ви можете надсилати файли з пристрою та шукати медіаконтент в Інтернеті. Передана інформація може займати лише 1,5 Гб. Коли з’єднання обривається, програмне забезпечення використовує систему для завантаження файлів.

У мультичатах можуть брати участь до 200 осіб, а з листопада 2015 року до супергруп можуть приєднатися до 1000 осіб.

У Telegram доступна бот-платформа. Боти здатні виконувати широкий спектр завдань, включаючи модерацію команди, покупки, платежі, розваги та пошук в Інтернеті чи державних реєстрах. Зв’язок ділиться користувачем Telegram і комп’ютерною програмою стороннього розробника.

Компоненти інтерфейсу месенджера дозволяють користувачеві спілкуватися з ботом, надсилаючи повідомлення, натискаючи команди та кнопки, а також використовуючи автоматичний режим. Особистий чат (традиційний спосіб), груповий чат і так званий вбудований режим — це три способи взаємодії користувачів і ботів Telegram:

− приватний чат – найпопулярніший спосіб. Лише в двох ситуаціях –

авторизація стороннього додатку через Telegram і подання заявки на приєднання до чату;

- деякі боти можуть бути учасниками груп Бот, наприклад, може проводити ігри, середні повідомлення або підтримувати розмову в групах;
- інлайн-режим нагадує інтерфейс пошукової системи. У полі введення інформації користувач вводить запит, який починається з короткого імені бота. Потім користувачі можуть вибрати та надіслати один із результатів.

Деякі боти можуть бути учасниками каналів. У каналах ботів застосовують здебільшого для запланованого розміщення повідомлень. З такими ботами користувач взаємодіє через приватний чат або вебінтерфейс.

Рисунок 2.12 – BotFather

2.4 OpenCV

OpenCV — це бібліотека з відкритим кодом, яка містить кілька сотень алгоритмів комп'ютерного зору. У документі описано так званий OpenCV 2. x API, який, по суті, є C++ API, на відміну від OpenCV 1. x API на основі C (C API застарів і не тестується з компілятором "C" після випуску OpenCV 2.4) [13].

Рис. 2.13 – OpenCV

OpenCV має гнучку структуру, що означає, що пропозиція включає деякі спільні або динамічні бібліотеки. Доступні такі компоненти:
основні функції (ядро) - невеликий модуль, що визначає основні структури даних, включаючи щільний багатовимірний масив Mat і фундаментальні функції, які використовуються всіма додатковими модулями;
обробка зображень (imgproc) — модуль обробки зображень, який включає лінійну та нелінійну фільтрацію зображень, геометричні перетворення зображення (зміна розміру, афінне та перспективне викривлення, загальне перевідображення на основі таблиці), перетворення простору кольорів, гістограми тощо;

- аналіз відео (фільм) – компонент аналізу відео, який включає в себе оцінку руху, віднімання фону та алгоритми відстеження об'єктів;

- калібрування камери та 3D-реконструкція (calib3d) - базові алгоритми геометрії кількох ракурсів, калібрування однієї та стереокамери, оцінка пози об’єкта, алгоритми стереовідповідності та елементи 3D-реконструкції;
- 2D Features Framework (features2d) – детектори, дескриптори та відповідники дескрипторів;
- виявлення об'єктів (objdetect) - ідентифікація об'єктів і ситуацій заданих курсів (наприклад, обличчя, очі, чашки, люди, машини і так далі);
- графічний інтерфейс високого рівня (highgui) - простий у використанні інтерфейс для простих можливостей інтерфейсу користувача;
- Video I/O (videoio) – простий у використанні інтерфейс для захоплення відео та відеокодеків;
- деякі інші допоміжні компоненти, такі як FLANN і Google перевірки оболонки, прив'язки Python та інші.

Офіційно запущений у 1999 році проект OpenCV був перш за все зусиллями Intel Research, спрямованими на розвиток додатків із інтенсивним використанням процесора, частиною серії проектів, включаючи трасування променів у реальному часі та 3D-дисплеї.] 4] Основні учасники проекту включали ряд проектів. експертів з оптимізації в Intel Russia, а також команди Intel Performance Library. На початку OpenCV цілі проекту описувалися як:

- попереднє дослідження цілей, надаючи не лише відкритий, але й оптимізований сценарій для основних засобів зору. Більше не потрібно винаходити колесо;
- розповсюджуйте знання про бачення, надаючи популярну інфраструктуру, на якій розробники можуть спиратися, щоб код був легше доступним і придатним для передачі;

- удосконалюйте промислові програми на основі візуалізації, надаючи зручний, оптимізований за продуктивністю сценарій, доступний безкоштовно – з дозволом, який не вимагає відкритості чи незалежності коду.

Друге бета-видання OpenCV було опубліковано на Конференції IEEE з комп'ютерного бачення та розпізнавання образів у 2000 році, а п'ять бета-версій було випущено між 2001 і 2005 роками. Друга версія 1.0 була випущена в 2006 році. Версія 1.1 вийшла у жовтні 2008 року.

Наступний великий випуск OpenCV був випущений у жовтні 2009 року. OpenCV 2 містить значні зміни в інтерфейсі C++, спрямовані на простіші, більш безпечні шаблони, нові функції та кращі реалізації для існуючих з точки зору продуктивності. Офіційні випуски тепер відбуваються кожні шість місяців.

У серпні 2012 року підтримку OpenCV взяла на себе некомерційна організація OpenCV.org, яка підтримує сайт розробників і користувачів.

У травні 2016 року Intel підписала угоду про придбання Itseez, провідного розробника OpenCV.

Хоча основний інтерфейс OpenCV написаний на мові програмування C++, він все ще має старіший, менш розширений інтерфейс C. Інтерфейс C++ містить усі новітні інновації та алгоритми. Python, Java та MATLAB/Octave мають прив'язки до мови. Онлайн-документація містить інтерфейс прикладного програмування (API) для цих інтерфейсів. Щоб сприяти сприйняттю ширшою аудиторією, було створено буклети багатьма мовами. Прив'язки JavaScript для певного набору функцій OpenCV стали доступними у версії 3.4. js для використання на платформах веб-сайтів.

2.5 Emgu CV

Emgu CV – це крос-платформна обгортка над OpenCV, написана на C#. Використовується у завданнях комп'ютерного зору та обробки зображень. Може використовуватись під Windows, Linux, Mac OS, iOS, Android. Сумісна з будь-якою мовою .NET [14].

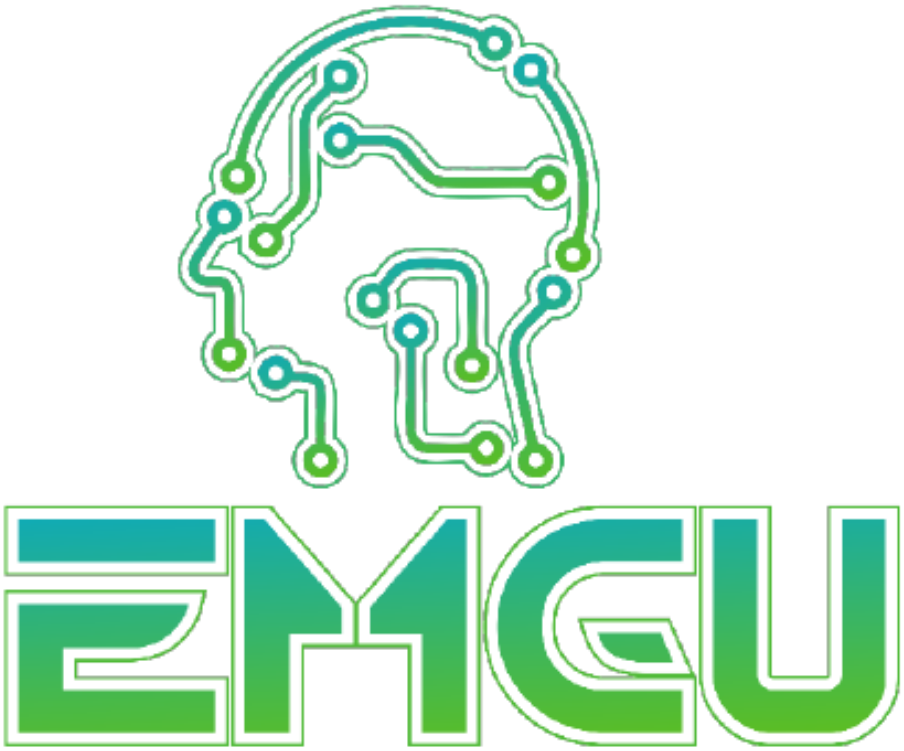

Рисунок 2.14 – Emgu CV

2.6 Мова програмування C#

C# — це мова програмування загального призначення високого рівня, яка підтримує кілька парадигм. C# охоплює статичну типізацію, жорстку типізацію, лексичну область видимості, імперативну, декларативну, функціональну, загальну, об’єктно-орієнтовану (на основі класів) і компонентно-орієнтоване програмування [15].

Мова програмування C# була розроблена Андерсом Хейлсбергом із Microsoft у 2000 році та пізніше була затверджена як міжнародний стандарт Ecma (ECMA-334) у 2002 році та ISO/IEC (ISO/IEC 23270) у 2003 році.

Microsoft представила C# разом із .NET. Framework і Visual Studio, обидва з яких були закритими. У той час Microsoft не мала порожніх довідкових матеріалів. Через чотири роки, у 2004 році, розпочався безкоштовний проект із відкритим вихідним кодом під назвою Mono, який надавав міжплатформний компілятор і середовище виконання для мови програмування C#. Через десять років Microsoft випустила Visual Studio Code (редактор коду), Roslyn (компілятор) і платформу unified.NET (фреймворк програмного забезпечення), усі вони підтримують C# і є безкоштовними, мають відкритий код і є кросплатформними. Mono приєднався до Microsoft, але не був об’єднаний з .NET.

Станом на листопад 2022 року найновішою стабільною версією мови є C# 11.0, яка була випущена в 2022 році в .NET 7.0.

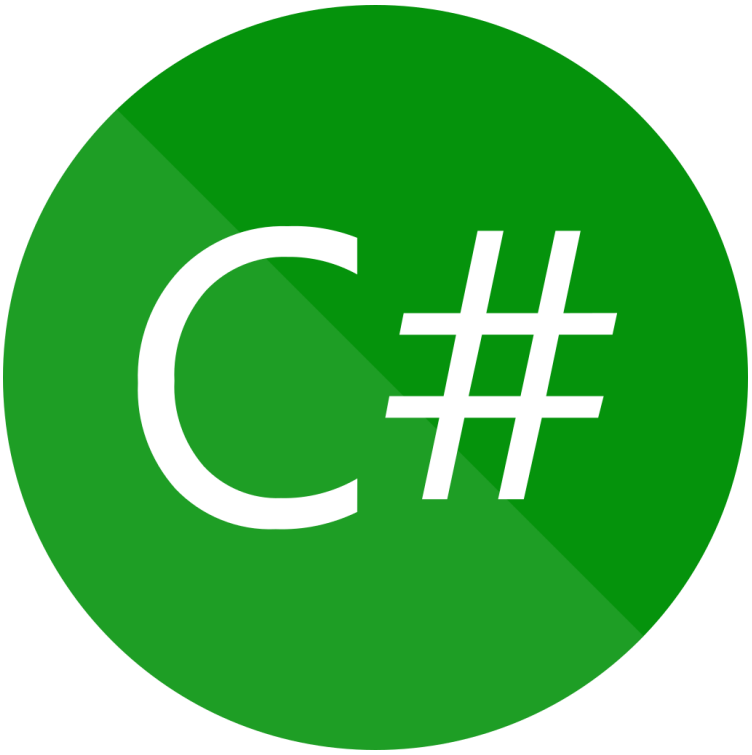

Рисунок 2.15 – Мова програмування C#

Стандарт Ecma перераховує такі цілі дизайну для C#:

- ця мова проста, розширена, об’єктно-орієнтована мова програмування загального призначення;
- мова та її реалізації мають забезпечувати підтримку принципів розробки програмного забезпечення, таких як перевірка типів, перевірка меж масиву, виявлення спроб використати неініціалізовані змінні та автоматизоване збирання сміття. Стійкість програмного забезпечення, довговічність і продуктивність програміста є важливими;
- мова призначена для використання в розробці програмних компонентів, придатних для реалізації в розподіленому середовищі;
- портативність дуже важлива для вихідного коду та розробників, особливо тих, хто вже знайомий із C та C++;
- підтримка інтернаціоналізації дуже важлива;
- C# призначений для написання програм як для розміщених, так і для вбудованих систем, починаючи від дуже великих, які використовують складні операційні системи, і закінчуючи дуже маленькими, що мають спеціальні функції;
- незважаючи на те, що додатки C# мають бути економічними щодо вимог до пам’яті та процесорної потужності, ця мова не була призначена для прямої конкуренції за продуктивністю із C або мовою асемблера.

Бібліотеки класів спочатку були створені за допомогою системи компілятора керованого коду "Simple Managed C" (SMC) під час розробки .NET Framework. Андерс Хейлсберг організував групу в січні 1999 року, щоб створити «Cool», або «C-подібну об'єктно-орієнтовану мову», таку назву отримала нова мова в той час. Microsoft вважала, що щодо збереження остаточної назви мови "Cool", але вирішила відмовитися від неї через

міркування торгової марки. Мова, бібліотеки класів і ASP були перейменовані на C# до того часу, коли проект .NET було оприлюднено на конференції професійних розробників у липні 2000 року. C# отримав середовище виконання NET.

Гейлсберг, який є головним розробником Microsoft і провідним архітектором C#, раніше працював над створенням Visual J++, Embarcadero Delphi (раніше CodeGear Delphi), Inprise Delphi та Turbo Pascal. В інтерв'ю та технічних документах він стверджував, що основи Common Language Runtime (CLR) були обумовлені недоліками в більшості популярних мов програмування (таких як C++, Java, Delphi та Smalltalk), які, у свою чергу, спонукали до появи мови C#. дизайн.

З моменту випуску версії 2.0 у листопаді 2005 року мови C# і Java розвивалися дедалі різними шляхами, що робить їх двома різними мовами. Одне з перших значних відхилень сталося з додаванням генериків до обох мов із дуже різними реалізаціями. Реифікація використовується в C# для забезпечення загальних об'єктів, які можна використовувати з генерацією коду під час завантаження класу, як і будь-який інший клас. Розширення LINQ, які були випущені з C# 3.0 і його підтримкою з лямбда-виразів, методів розширення та анонімних типів, є кульмінацією кількох значущих доповнень, зроблених C++ для програмування у функціональному стилі. Ці функції дають змогу програмістам на C# використовувати замикання та інші методи функціонального програмування, коли це принесе користь їх застосуванню. Щоб покращити читабельність і зручність обслуговування, розробники можуть використовувати розширення LINQ і функціональний імпорт, щоб зменшити кількість шаблонного коду, необхідного для рутинних завдань, таких як запити до бази даних, аналіз XML-файлів або пошук структури даних.

Енді, якого назвали на честь Андерса Хейлсберга, раніше був талісманом C#. 29 січня 2004 року виведено на пенсію.

Рисунок 2.16 – Маскот C#

C# був спочатку розглянутий підкомітетом ISO/IEC JTC 1 SC 22, перш ніж був відкликаний і отримав добро у 2006 році. З цією версією 23270: 2006 відкликано та затверджено.

Варіант мови C, призначений для інкрементної компіляції, вперше отримав назву C# від Microsoft у 1988 році. Пізніше назву було змінено, оскільки цей проект не був завершений.

Натхненням для назви послужив нотний запис, який визначає, що написана нота має бути на півтону вищою за висотою. Це можна порівняти з мовою програмування C++, де "++" означає, що змінну потрібно оцінити перед збільшенням на 1. Мова є приростом C++, про що свідчить схожість символу різкості з лігатурою з чотирьох "+". " символи.

Цифровий знак було вибрано так, щоб нагадувати різкий символ у письмовій назві мови програмування через технічні обмеження відображення та відсутність різких символів на більшості розкладок клавіатури. Специфікація мови C# ECMA-334 враховує цю угоду.

Багато інших мов .NET, які є варіаціями існуючих мов, наприклад J# (мова, розроблена Microsoft, похідна від Java 1.1), A# від Ada та функціональна мова програмування F#, використовували суфікс «sharp». Оскільки тепер підтримується вся мова Eiffel, оригінальну назву для .NET-версії мови скасовано. Бібліотеки також використовували суфікс, включаючи Gtk#, .NET-обгортку для GTK та інших бібліотек GNOME, і Cocoa#.

2.7 Visual Studio

Інтегроване середовище розробки Microsoft (IDE) називається Visual Studio. Він використовується для створення веб-додатків, мобільних додатків, веб-сервісів і веб-сайтів. Інструменти розробки програмного забезпечення Microsoft, такі як Windows API, Forms, Presentation Foundation, Windows Store і Microsoft Silverlight, використовуються Visual Studio. Він може створювати як рідний, так і керований код [16].

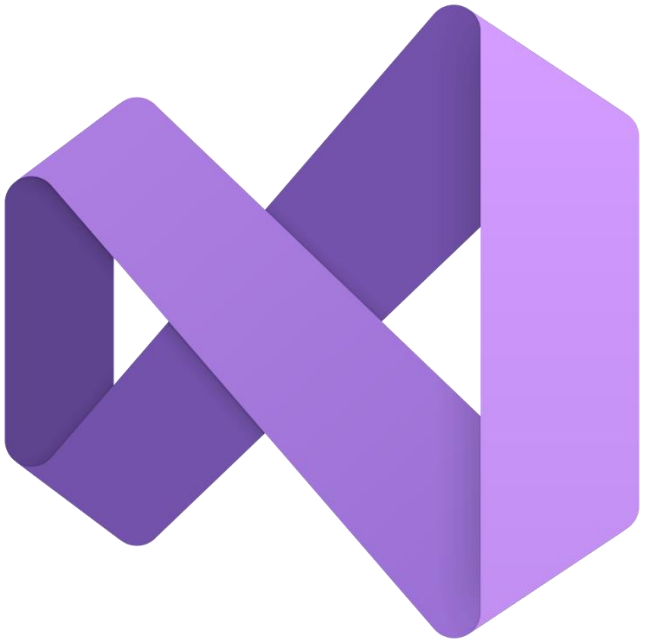

Рисунок 2.17 – Visual Studio

Рефакторинг коду та редактор коду, сумісний із IntelliSense, включені до Visual Studio. Інтегрований налагоджувач працює як налагоджувач на рівні джерела та як налагоджувач на рівні машини. Інші вбудовані інструменти включають профайлер коду, конструктор для створення програм графічного інтерфейсу користувача, веб-дизайнер, конструктор класів і конструктор схем бази даних. Він приймає плагіни, які розширюють функціональні можливості майже на всіх рівнях, включаючи додавання підтримки для систем керування вихідним кодом і додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для доменних мов або наборів інструментів для інших аспектів життєвого циклу розробки програмного забезпечення.

Visual Studio дозволяє розміщувати лише функції, які були закодовані як VSPackage; він не підтримує жодної конкретної мови програмування, рішення чи інструменту. Інтерфейс пропонує три служби: SVsSolution, який дає змогу нумерувати проекти та рішення, MVsUIShell який забезпечує функціональність інтерфейсу користувача (а також вкладки, панелі інструментів і вікна інструментів), а також реєстрацію VSPackages. Крім того, IDE також відповідає за координацію та забезпечення зв’язку між службами. VSPackages використовуються для всіх редакторів, виробників, типів проектів і багатьох інструментів. Щоб отримати VSPackages, Visual Studio використовує COM.

The Managed Package Framework (MPF), колекція керованих оболонок навколо COM-інтерфейсів, які дозволяють створювати пакети на будь-якій сумісній мові CLI, також включено до Visual Studio SDK. Але MPF не пропонує всіх функцій, які пропонує COM-інтерфейс Visual Studio. Потім ці служби можна використовувати для створення додаткових пакетів, які покращують функціональність Visual Studio IDE.

Для додавання підтримки мов програмування використовується окремий VSPackage, відомий як Language Service. Реалізація VSPackage може включати підтримку різних функцій шляхом реалізації різних інтерфейсів, які визначає мовна служба. Таким чином можна додати колір синтаксису, висновок оператора, відповідність фігурних дужок, спливаючі підказки, списки

учасників і маркери помилок для фонової компіляції. Функції будуть доступні для мови, якщо реалізовано інтерфейс. Сервіси Word пропонуються для кожної мови. Компілятор мови або код компілятора можуть повторно використовуватися реалізаціями. Або власний код, або керований код можна використовувати для реалізації служб Word. Для рідного коду можна використовувати або рідний інтерфейс COM, або Babel Framework (компонент Visual Studio SDK).

Visual Studio не містить жодної вбудованої підтримки керування джерелами, але визначає два альтернативні способи інтеграції систем керування джерелами з IDE. VSPackage для контролю пакетів може запропонувати унікальний інтерфейс користувача. Плагін керування джерелом, з іншого боку, використовує звичайний користувальницький інтерфейс Visual Studio та MSSCCI для реалізації різноманітних джерел. Спочатку Visual SourceSafe було інтегровано з Visual Studio 6.0 за допомогою MSSCCI, але пізніше для його відкриття використовувався SDK. MSSCCI 1.1 використовувався Visual Studio .NET у 2002 році, а Microsoft SCCI 12 – у 2003 році. Версія MSSCCI 1.3 використовується у Visual Studio 2005, 2008 та 2010 і додає підтримку синхронного запуску, перейменування та видалення розповсюдження.

Для змінних, функцій, методів, циклів і запитів LINQ Visual Studio містить редактор коду, який підтримує підсвічування синтаксису та завершення коду за допомогою IntelliSense. Під час створення веб-сайтів і веб-додатків IntelliSense сумісний із включеними мовами, а також XML, каскадними таблицями стилів і JavaScript. Над вікном редактора коду, поряд із курсором редагування, з’являється немодальний список із пропозиціями автозаповнення. Його можна тимчасово зробити напівпрозорим у Visual Studio 2008 і пізніших версіях, щоб переглянути код, який він заважає. Усі підтримувані мови використовують редактор коду.

Для швидкої навігації закладки можна встановити в коді за допомогою редактора коду Visual Studio. Окрім звичайного текстового пошуку та пошуку

за регулярними виразами, інші інструменти навігації включають згортання блоків коду та поступовий пошук. Список завдань і багатоелементний буфер обміну також є функціями редактора коду. Редактор коду підтримує фрагменти коду, збережені шаблони для повторюваного коду, які можна вставити в код і налаштувати для проекту, над яким працюється. Крім того, включено засіб керування фрагментами коду. Ці інструменти можна прикріпити збоку від екрана або відкрити у вигляді плаваючих вікон, які автоматично закриваються, коли вони не використовуються. Рефакторинг коду також підтримується редактором коду Visual Studio, включаючи інкапсуляцію членів класу всередині властивостей, перепризначення параметрів, повторне відкриття змінних і методів, а також вилучення інтерфейсу.

У Visual Studio є налагоджувач, який можна використовувати як налагоджувач на рівні машини та джерела. Його можна використовувати для налагодження програм, написаних будь-якою мовою, яка підтримується Visual Studio, і працює як з керованим, так і з рідним кодом. Крім того, він має можливість моніторингу, налагодження та підключення до запущених процесів. Якщо вихідний код запущеного процесу доступний, він показує код під час його виконання. Він може відображати розбирання, якщо вихідний код недоступний. Дампи пам’яті можна створювати та завантажувати пізніше для налагодження за допомогою налагоджувача Visual Studio. Додатково підтримуються багатопотокові програми. Налагоджувач можна налаштувати на запуск, коли програма, яка працює поза середовищем Visual Studio, аварійно завершує роботу.

Точки зупинки, які дозволяють тимчасово призупинити виконання в певному місці, а також спостереження (які відстежують значення змінних у ході виконання), обидві можливі за допомогою Visual Studio Debugger. Точки зупину можуть бути умовними, що означає, що вони спрацьовують, коли виконується умова. Код можна переступати, тобто виконувати один рядок (вихідного коду) за раз. Він може переходити до функції або входити в неї для налагодження, тобто виконання тіла функції недоступне для перевірки вручну.

Налагоджувач підтримує редагування та продовження, що дозволяє редагувати код під час його компіляції. Під час налагодження поточне значення будь-якої змінної відображається у підказці ("data tooltipps"), де його можна змінити за потреби, якщо навести на нього вказівник миші. Налагоджувач Visual Studio забезпечує ручний доступ до певних функцій із вікна інструменту Imediate під час кодування. У вікні Immediate надаються параметри методу.

Висновки до розділу 2

У другому розділі були розглянуті засоби та методи використані для створення телеграм-боту для розпізнання україномовного тексту.

Наступний розділі буде присвячений проектуванню і розробці власної програмної реалізації системи розпізнання тексту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Створення бота

Вданній роботі бібліотеки Telegram.Bot та Telegram.Bot.Extentions.Polling, оновлення будемо отримувати періодично опитуючи Telegram сервер на наявність нових оновлень. Polling простіше в реалізації оскільки не потрібно отримувати SSL-сертифікат і бот можна запустити відразу після написання коду без додаткових проблем.

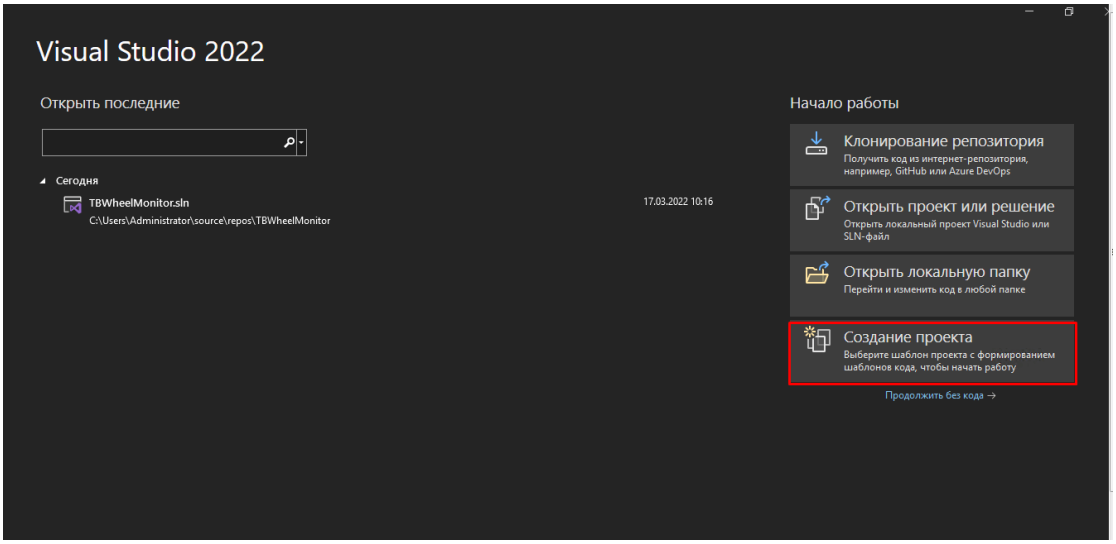

Рисунок 3.1 – Створення проекту

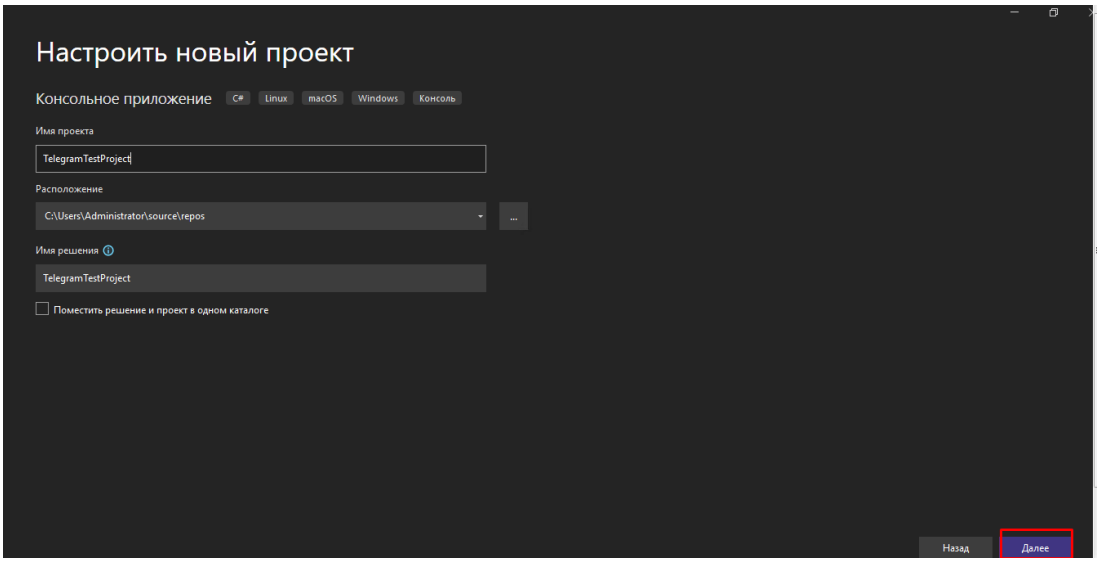

Рисунок 3.2 – налаштування нового проекту

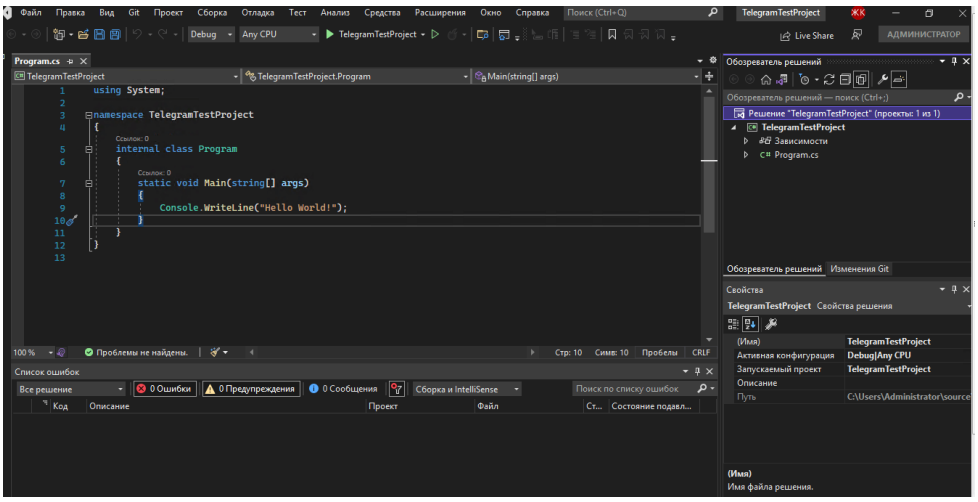

Рисунок 3.3 – Створений проект

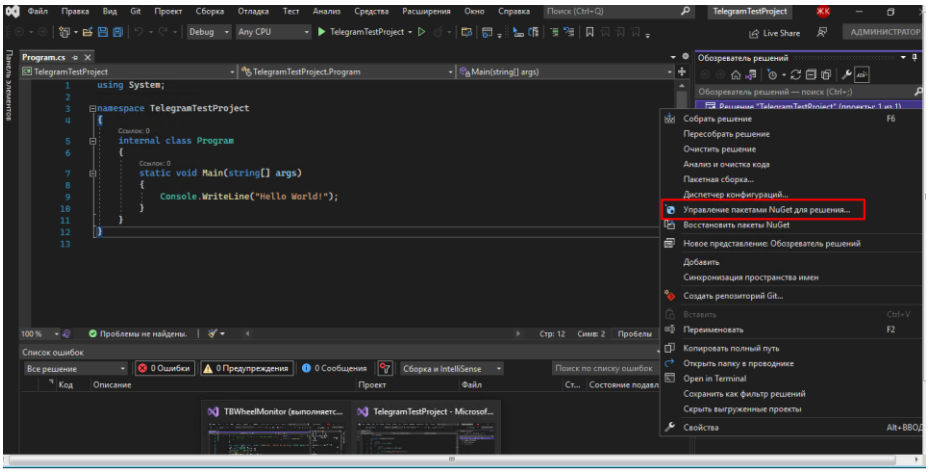

Рисунок 3.4 – Відкриття NuGet

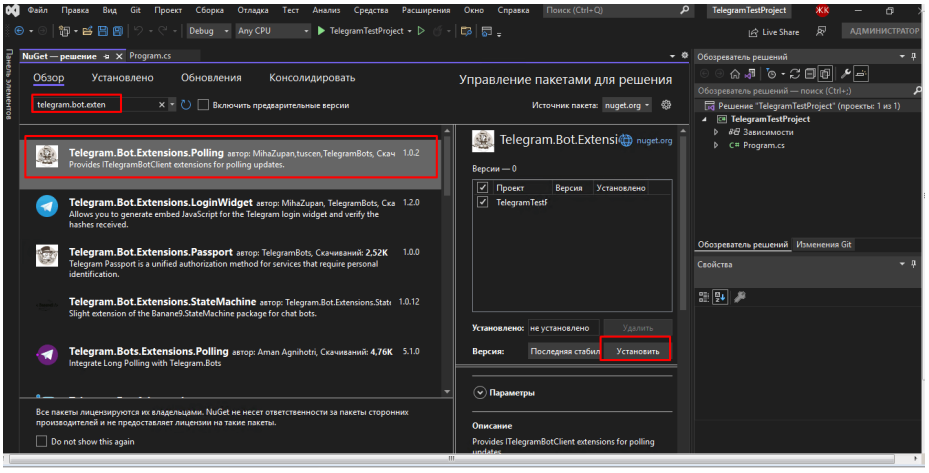

Рисунок 3.5 – Встановлення бібліотек

Створюємо telegram бот у BotFather. Копіюємо його api key для роботи. Знаходимо в telegram BotFather, відправляємо йому /newbot, назву та логін бота. BotFather повинен нам надати API key, який ми повинні вставити в код у наступному рядку:

static ITelegramBotClient bot = New TelegramBotClient("TOKEN");

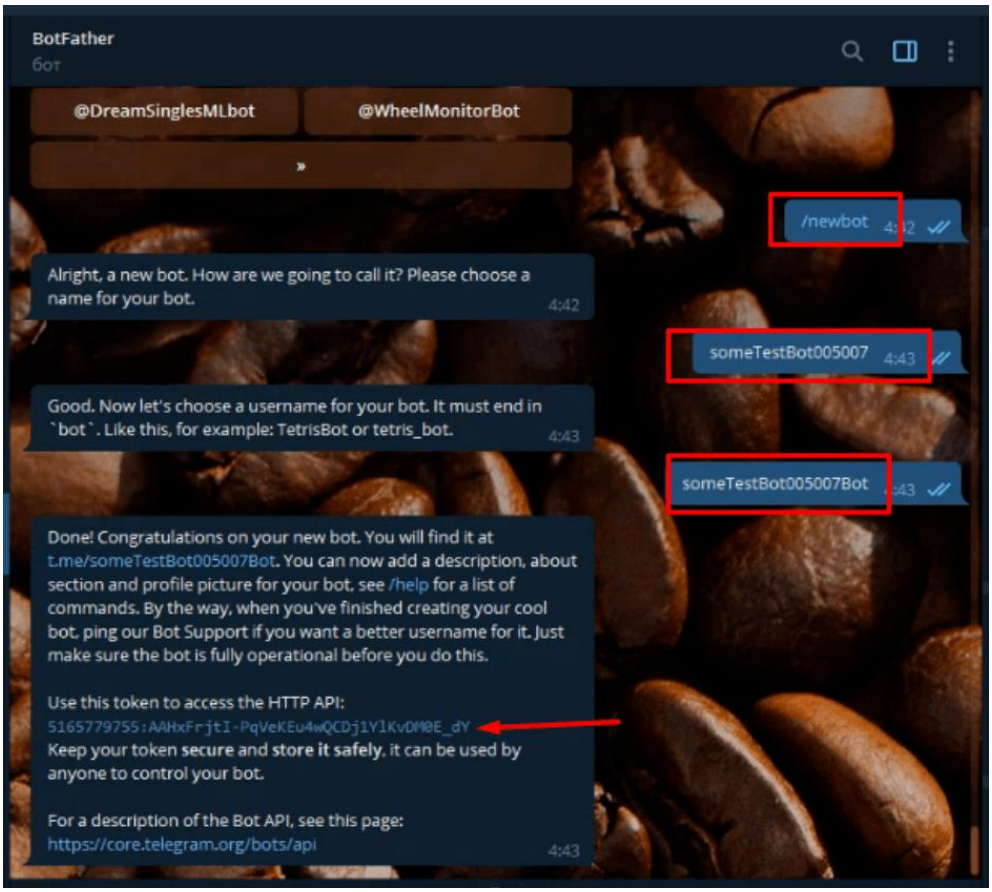

Рисунок 3.6 – Отримання API key

3.2 Збереження зображення

Головне завдання, яке має виконувати бот – це реагувати на повідомлення, які надсилає йому користувач. Звичайно, розробники telegram'a заклали можливість відстеження та реагування на багато інших подій.

За допомогою наступного коду здійснюється збереження фото відправленого користувачем для подальшого розпізнання тексту зображеного на ньому.

```
var fileId = update.Message.Photo.Last().FileId;
                        var fileInfo = await botClient.GetFileAsync(fileId);
                        var filePath = fileInfo.FilePath;

                        string destinationFilePath =
"C:\\Users\\stalk\\Desktop\\DiplomBot\\DiplomBot\\Temp\\downloaded.jpg";

                        using (Stream fileStream =
System.IO.File.OpenWrite(destinationFilePath))
                        {
                            await botClient.DownloadFileAsync(
                                filePath: filePath,
                                destination: fileStream,
                                cancellationToken: cancellationToken);
                              };
```

3.3 Розпізнавання тексту зображення

За допомогою наступного коду здійснюється розпізнання тексту з зображення надісланого користувачем. Розпізнаний текст повертається користувачеві в якості повідомлення від бота.

```
private static string RecognisingImage(string path)
        {
            string text = string.Empty;
            Tesseract tesseract = new
Tesseract(@"C:\Users\stalk\Desktop\DiplomBot\DiplomBot\Models", "UA",
OcrEngineMode.TesseractLstmCombined);
            tesseract.SetImage(new Image<Bgr, byte>(path));
            tesseract.Recognize();
            text = tesseract.GetUTF8Text();
            tesseract.Dispose();
            return text;
              }
```

В даній роботі використовується згорткова нейронна мережа з функцією активації Relu та 30-ма прихованими шарами .

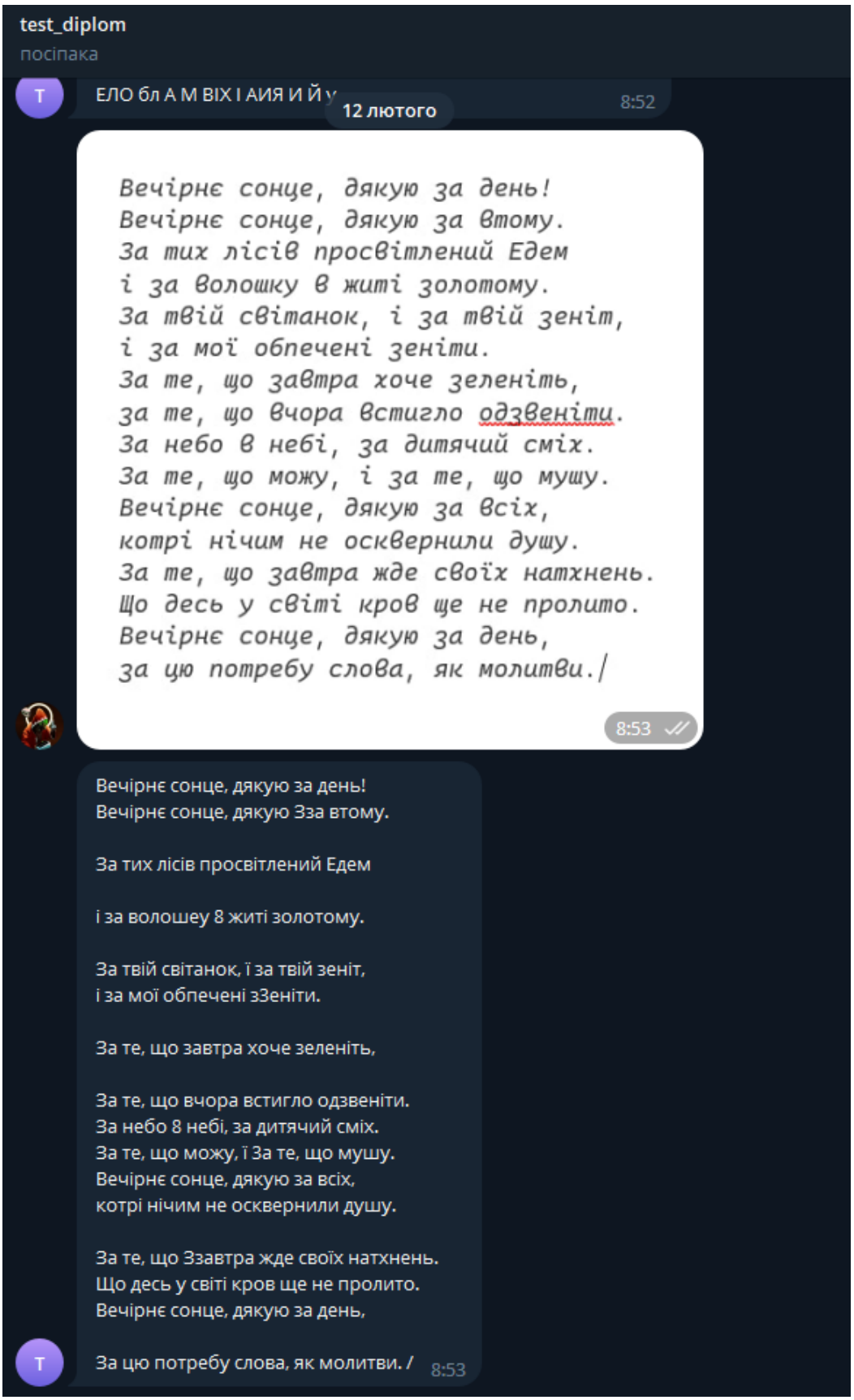

Рисунок 3.7 – Результат розпізнавання

Висновки до розділу 3

В даний час багато вчених займаються дослідженням нейронних мереж, стійкості тих чи інших конфігурацій, проте далеко не всі завдання розпізнавання образів можуть бути вирішені нейронними мережами. А якщо і можуть бути вирішені, то з будь-якими обмеженнями. Незважаючи на те, що штучний нейрон є моделлю біологічного нейрона, він лише апроксимує його властивості, а не точно їх повторює. Тому машина, як і раніше, не в змозі навчатися, як людина.

В ході даного розділу було розглянуте створення телеграм-боту для розпізнавання україномовного тексту та результати його роботи.

ВИСНОВКИ

В рамках випускної кваліфікаційної роботи було вивчено основні методи розпізнавання об'єктів на зображенні, розглянуті особливості проектування та навчання штучних згорткових нейронних мереж, сфери їх застосування та проблеми, що часто зустрічаються.

Згодом було реалізовано процедуру сегментації зображення, обрано згорткову нейронну мережу для розпізнавання символів української мови на зображенні. Під час розробки програми використовували мову програмування C# та засоби бібліотеки Emgu CV.

Використання згорткових нейронних мереж дозволяє будувати якісні алгоритми розпізнавання об'єктів на фотографіях та відеофайлах. Як правило, структуру CHP підбирають відповідно до особливостей задачі та з вимогами до її вирішення. Орієнтація на вузькоспеціалізоване завдання дозволяє отримувати точні та коректні рішення. Також згорткові мережі відрізняються високою продуктивністю та ефективністю. Нині сфер їх застосування стає дедалі більше

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rodriguez-Canosa G. R., Thomas S., Cerro J. et al. A real-time method to detect and track moving objects (DATMO) from unmanned aerial vehicles (UAVs) using a single camera // Remote Sensing. 2012. No 4. P. 1090–1111. 64
2. ABBYY FineReader [Електронний ресурс] – Режим доступу до ресурсу: https://pdf.abbyy.com/ru/finereader-pdf/.
3. SimpleOCR [Електронний ресурс] – Режим доступу до ресурсу: https://pdf.wondershare.net/ad/pdf-editor/ocr.html?gclsrc=aw.ds&&gclid=Cj0KCQiAqbyNBhC2ARIsALDwAsA4ROgK3HY3x1MlFH8s5kn7aCrMqOJsXlkZ8M5SS135moOSJaxc068aAgga3EALw_wcB.
4. PDFelemnt PRO [Електронний ресурс] – Режим доступу до ресурсу: https://pdf.wondershare.net/ad/pdfelement-brand.html?gclsrc=aw.ds&&gclid=Cj0KCQiAqbyNBhC2ARIsALDwAsDGz6QvAnJtnizUP9gZnSm9VTeF-JgvyLXW0_0yQ8nWIWirLDPITrkaAkw4EALw_wcB.
5. Free Online OCR [Електронний ресурс] – Режим доступу до ресурсу: https://www.newocr.com.
6. TopOCR [Електронний ресурс] – Режим доступу до ресурсу: https://www.topocr.com/.
7. НЕЙРОННІ МЕРЕЖІ: ЇХ ЗАСТОСУВАННЯ, РОБОТА [Електронний ресурс] – Режим доступу до ресурсу: https://www.poznavayka.org/uk/nauka-i-tehnika-2/neyronni-merezhi-yih-zastosuvannya-robota/
8. Штучна нейронна мережа [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B0_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0

%B0.

9. Багатошаровий перцептрон [Електронний ресурс] – Режим доступу до ресурсу: http://ni.biz.ua/16/16_3/16_36920_mnogosloyniy-perseptron.html.
10. Рекурентні нейронні мережі [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/7653871/page:9/.
11. Охоцинский Д. Е. Системы технического зрения / Д. Е. Охоцинский, В. М. Златкис. – М. : Наука, 1991. – 201 с. 26. Манкин В. И. Техническое зрение роботов / В. И. Манкин, А. А. Петров, В. С. Титов, Ю. Г. Якушенков. – М. : Машиностроение, 1990. – 272 с
12. Telegram [Електронний ресурс] – Режим доступу до ресурсу: https://telegram.org/faq.
13. OpenCV [Електронний ресурс] – Режим доступу до ресурсу: https://docs.opencv.org/4.7.0/index.html.
14. Emgu.CV [Електронний ресурс] – Режим доступу до ресурсу: https://emgu.com/wiki/files/2.0.0.0/Index.html.
15. Документация по C# [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/ru-ru/dotnet/csharp/.
16. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.
17. Закон України “Про охорону праці” / Законодавство України про охорону праці. - К. Нова редакція 2002 р.
18. Закон України “Про охорону навколишнього природного середовища” – К.: Україна. – 1991. - 59 с.
19. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин/ Зареєстровано в Міністерстві юстиції України 19 квітня 2010 р. за N 293/17588
20. Правила улаштування электроустановок. ПУЕ.– Харків.: Форт – 2011 – 728 с.
21. Гігієнічна класифікація праці за показниками шкідливості та

небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу. Гігієнічні нормативи ГН 3.3.5-8-6.6.1 2002 р. Видання офіційне Київ, 2001 рік – 46 с.

22. НАПБ Б.03.002 – 2007 Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою. НаказМНС від 03.12.2007 №883.

23. ДСанПін 3.3.2.007– 98 Державні санитарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. – К.:ГСЕУ України, 1998 – 21 с.

24. ДСТУ ISO14001 - 97 – 14012-97. Система управления окружающей средой – К.:ДЕРЖСТАНДАРТ УКРАИНЫ – 225 с.

25. Hansen L., Salamon P. Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence. San Diego, 1990. P. 993–1001.

26. Boser, B., Guyon I., Vapnik, V. A training algorithm for optimal margin classifiers. In D. Haussler, editor, proceedings of the 5th Annual ACM Workshop on COLT, pp. 144-152, Pittsburgh, 1992.

27. Breiman, L. Bagging predictors. Machine Learning, Vol. 24 (2), pp. 123-140, 1996.

28. Khaustov P.A. Algorithms for handwritten character recognition based on constructing structural models / P.A. Khaustov // Computer Optics. – 2017. – № 41 (1). – P. 67–78.

29. Моругов А.М. Методы распознавания символов / А.М. Моругов, С.В. Волков // Труды Международного симпозиума «Надежность и качество». – Пенза : ПГУ, 2017. – Т. 1. [Електронний ресурс]. – Режим доступу : https://cyberleninka.ru/article/n/metody-raspoznavaniya-simvolov/viewer.

30. Perceptron [Електронний ресурс]. – Режим доступу :

https://en.wikipedia.org/wiki/Perceptron.

31. Lan H. The Softmax Function, Neural Net Outputs as Probabilities, and Ensemble Classifiers / H.Lan. – 2017 [Електронний ресурс]. – Режим доступу : https://towardsdatascience.com/the-softmax-function-neural-net-outputs-as-probabilities-and-ensemble-classifiers-9bd94d75932.

32. Upadhyay Y. Introduction to Feed Forward Neural Networks / Y.Upadhyay. – 2019 [Електронний ресурс]. – Режим доступу : https://en.wikipedia.org/wiki/Perceptron.

33. Neural M. Networks and Deep Learning / M.Nielsen // Determination Press. – 2015. – P. 224.

34. Optical character recognition [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Optical_character_recognition.

35. Чиркова Т.І. Аналіз існуючих методів розпізнавання друкованих та рукописних текстів / Т.І. Чиркова, Ю.Г. Тендітний, Л.О. Латанська // Матеріали III Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»: збірка наукових праць (30 листопада 2020 року) / Під редакцією Г.О. Райко. – Херсон: Видавництво ФОП Вишемирський В. С., 2020. – с.86-87.

36. Чавчанідзе В.В. Аналітичне рішення задачі формування понять і розпізнавання образів//Збірник доповідей, т.61, №1 / Тбілісі, 1971. – с. 102.

37. Хінтон Дж. Е., Седжновский Т. Дж. Навчання та перенавчання в машинах Больцмана//У "Паралельно розподіленій обробці"/Кембридж, Массачусетс: MIT Press. – вип. 1,ис. 1986. – с.282-317.

38. Сенашова М.Ю. Глава 6. Похибки в нейронних мережах // Нейроінформатика / О.М. Горбань, В.Л. Дунін – Барковський, А.Н. Кірдіна, Е.М. Міркес і ін. Новосибірськ: Наука, 1998. – с.89-91

39. Мозговой А.А. Проблема существующих методик оптического распознавания рукописного текста.

40. Берр, Д. Дж. Експерименти з коннекціоністським читачем тексту//У матеріалах Першого міжнародного з нейронних мереж// Під редакцією М.Коділл і К.Батлер, вип. 4. Сан – Дієго, Каліфорнія: Видавництво SOS – друк, 1987. – с. 24-717.

41. Субботін, С. О. Нейронні мережі : навчальний посібник / С. О. Субботін, А. О. Олійник ; під заг. ред. проф. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2014. – 132 с

ДОДАТОК А

Лістинг коду програми

```
using Telegram.Bot.Polling;
using Telegram.Bot.Types;
using Telegram.Bot;
using Emgu.CV.OCR;
using Emgu.CV.Structure;
using Emgu.CV;
using Telegram.Bot.Types.Enums;


namespace TelegramBotExperiments
{
    class Program
    {
        static string token = "5959928753:AAHhwPcTpf1VgHh4nkGFIgT_JlpjVMIRe7A";
        static ITelegramBotClient bot = new TelegramBotClient(token);

        public static async Task HandleUpdateAsync(ITelegramBotClient botClient,
Update update, CancellationToken cancellationToken)
        {
            Console.WriteLine(Newtonsoft.Json.JsonConvert.SerializeObject(update));

            // Обробка повідомлень користувача
            if (update.Type == Telegram.Bot.Types.Enums.UpdateType.Message)
            {
                var message = update.Message;

                if (message.Type==MessageType.Photo) // перевірка чи це фото
                {
                    try
                    {
                        var fileId = update.Message.Photo.Last().FileId;
                        var fileInfo = await botClient.GetFileAsync(fileId);
                        var filePath = fileInfo.FilePath;

                        string destinationFilePath =
"C:\\Users\\stalk\\Desktop\\DiplomBot\\DiplomBot\\Temp\\downloaded.jpg";

                        using (Stream fileStream =
System.IO.File.OpenWrite(destinationFilePath))
                        {
```

```
                        await botClient.DownloadFileAsync(
                            filePath: filePath,
                            destination: fileStream,
                            cancellationToken: cancellationToken);
                    };
                    await botClient.SendTextMessageAsync(message.Chat,
RecognisingImage(destinationFilePath));
                }
                catch (Exception ex)
                {

Console.WriteLine(Newtonsoft.Json.JsonConvert.SerializeObject(ex));
                }
                return;
            }
        }
    }

    private static string RecognisingImage(string path)
    {
        string text = string.Empty;
        Tesseract tesseract = new
Tesseract(@"C:\Users\stalk\Desktop\DiplomBot\DiplomBot\Models", "UA",
OcrEngineMode.TesseractLstmCombined);
        tesseract.SetImage(new Image<Bgr, byte>(path));
        tesseract.Recognize();
        text = tesseract.GetUTF8Text();
        tesseract.Dispose();
        return text;
    }

    public static async Task HandleErrorAsync(ITelegramBotClient botClient,
Exception exception, CancellationToken cancellationToken)
    {
        // Вивід помилок в консолі у вигляді JSON

Console.WriteLine(Newtonsoft.Json.JsonConvert.SerializeObject(exception));
    }

    static void Main(string[] args)
    {
        Console.WriteLine("Start: " + bot.GetMeAsync().Result.FirstName);

        var cts = new CancellationTokenSource();
```

```
            var cancellationToken = cts.Token;
            var receiverOptions = new ReceiverOptions
            {
                AllowedUpdates = { },
            };
            bot.StartReceiving(
                HandleUpdateAsync,
                HandleErrorAsync,
                receiverOptions,
                cancellationToken
            );
            Console.ReadLine();
        }
    }
}
```